University of the Philippines Manila
College of Arts and Sciences
Department of Physical Sciences and Mathematics

# DynastyMap: An Interactive Data Visualization Tool on Political Dynasties in the Philippines

A special problem in partial fulfillment
of the requirements for the degree of
**Bachelor of Science in Computer Science**

Submitted by:

Lean Godffrey M. Rada

Aug 2015

Permission is given for the following people to have access to this SP:

| | |
|---|---|
| Available to the general public | Yes |
| Available only after consultation with author/SP adviser | No |
| Available only to those bound by confidentiality agreement | No |

## ACCEPTANCE SHEET

The Special Problem entitled "DynastyMap: An Interactive Data Visualization Tool on Political Dynasties in the Philippines" prepared and submitted by Lean Godffrey M. Rada in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

**Ma. Sheila A. Magboo, M.Sc.**
Adviser

**EXAMINERS:**

| | **Approved** | **Disapproved** |
|---|---|---|
| 1. Gregorio B. Baes, Ph.D. (*candidate*) | _____ | _____ |
| 2. Avegail D. Carpio, M.Sc. | _____ | _____ |
| 3. Perlita E. Gasmen, M.Sc. (*candidate*) | _____ | _____ |
| 4. Ma. Sheila A. Magboo, M.Sc. | _____ | _____ |
| 5. Vincent Peter C. Magboo, M.D., M.Sc. | _____ | _____ |
| 6. Bernie B. Terrado, M.Sc. (*candidate*) | _____ | _____ |

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

**Ma. Sheila A. Magboo, M.Sc.**
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences and Mathematics

**Marcelina B. Lirazan, Ph.D.**
Chair
Department of Physical Sciences and Mathematics

**Alex C. Gonzaga, Ph.D., Dr.Eng.**
Dean
College of Arts and Sciences

**Abstract**

Political dynasties are prevalent in Philippine politics. The relationship between the prevalence of political dynasties and other aspects of the country, such as poverty, may be studied using data visualization. In this paper, DynastyMap, a web-based map data visualization system was presented as a tool for studying data related to political dynasties in the Philippines. The choropleth map and the tag map were used as data visualization methods, with the choropleth showing various datasets and the tag map showing dynasty data. Using these visualizations with the right datasets, DynastyMap can be an effective tool for the study or presentation of political dynasty data in the Philippines.

*Keywords*: data visualization, political dynasty, geographic information system

# Contents

# List of Figures

# List of Tables

# I. Introduction

## A. Background of the Study

Political dynasties are active in the Philippines. [1] A political dynasty is a group of elected persons who belong to the same family. [2] The prevalence of political dynasties in the Philippines is notably high. According to a study, the share of representatives with dynastic links reaches 70% in the Philippines. [3]

Are political dynasties on the rise? If one observes the elections, many candidates belong to political families. Some of the known political families are the Binays, the Marcoses, and the Ampatuans. Which families are growing? Which ones are shrinking? How does their influence affect the areas they govern?

There is a view that political dynasties significantly contribute to corruption in the Philippine government, and consequently, the deterioration of socio-economic development in the Philippines. [3] There is also a view that the presence of dynasties does not correlate with the incidence of poverty in an area. [4] How do political dynasties affect various aspects of the country?

To answer these questions, one may look into data and statistics. Philippine statistics and data are publicly available from several agencies and government bodies. Election records can be obtained from the Commission on Elections (COMELEC). Various Philippine statistics can be obtained from the Philippine Statistics Authority (PSA) and the National Statistics Office (NSO).

One of the means to study data is the use of data visualization. Data visualization is an application of computing to provide visual presentations of data. Data visualization is an important tool in research. It can reveal patterns and trends which can provide insights and a deeper understanding of the data. There are several computer software available that can generate visualizations of data. [5] Office spreadsheet applications, the R software environment, and the D3 software library are some examples of tools that can visualize data.

Data visualizations are recently becoming more dynamic with the use of ani-

mation and interactivity. Interaction and animation can provide more engagement with the information, allow for easier multidimensional data representation, and can give the viewers a clearer perspective on the data. Animations can be used for temporal data to give a view of the dynamics and trends of the data. Interactivity can be used for data exploration, such as focusing on parts of the data (zooming in). [6] An example of interactive visualization software is Gapminder's Trendalyzer, which produces animated and interactive bubble charts. [7]

## B. Statement of the Problem

Without a data visualization tool, study of data may be impaired. Data visualizations help in understanding the data by presenting information in an visual way.

There are existing tools for data visualization. However, there were none available for the special purpose of computing and visualizing political dynasties and indicators from raw election records, and additionally visualizing other user-provided datasets along with them.

Therefore, there was a need for a data visualization system that provides these necessary functions.

## C. Objectives of the Study

The project aimed to develop an online interactive data visualization tool for computing and visualizing political dynasties and their relationship with user-provided regional Philippine data, such as regional unemployment rates and regional GDP.

The following specific functions are provided by the system:

1. Compute dynasty indicator variables for political dynasties from the found dynasties. These will be used for visualization. These indicators will be discussed in detail in the third chapter of this paper.

2. Allow registered users to:

(a) Upload their own Philippine geographical dataset.

i. Upload a file in an accepted format (CSV) containing the dataset. The specifications of the format is in the fourth chapter of this paper.

ii. Manually enter the data using a web form interface.

3. Allow any user to:

(a) Visualize data in layers overlaid onto the map of the Philippines.

i. Choropleth map for the visualization of any chosen variable from user-uploaded Philippine geographical data and any of the dynasty indicator variables.

ii. Tag map for the visualization of local dynasties.

iii. Display additional information when selecting an area in the map.

(b) Visualize data in different levels of detail.

i. Regional level.

ii. Provincial level.

iii. Municipal level.

iv. Barangay level.

(c) Select the year to visualize.

(d) Animate the visualization (show successive visualizations of consecutive years).

(e) Register for an account.

4. Allow administrators to:

(a) Add/delete election results data.

i. Upload a file in a tabular format (CSV) containing election results data. The format is defined in the fourth chapter of this paper.

ii. Manually enter the data through a web form interface.

(b) Add/delete datasets uploaded by users.

(c) Add/edit/delete the political dynasty associations.

(d) Add/edit/delete user accounts.

## D. Significance of the Project

Unlike static data visualizations, interactive animated visualizations may perform better. Interactivity encourages exploratory data analysis. Animated visualizations are appropriate for temporal data. It provides a richer experience, which can make the user gain more knowledge about the data being studied.

In addition to visualization of data, the system also processes data. From election records and political dynasty data, it computes indicators for political dynasty prevalence. The system could still be used for future data without changing the system code, assuming the Philippines retains its political structure.

The system can be helpful for political studies in the Philippines. The system can help discover new insights about political dynasties in the Philippines, aid in generating specific hypotheses on political dynasties, and act as a starting point of deeper research into the subject.

The system can be made available to the public Internet for the general public to view. The information presented by the system will be useful for journalists and the general public. The general public can use this to make informed decisions on who to vote or who to not vote. The system can be used to help or urge policy makers to form plans to address the political dynasty situation in the Philippines. The system can also be used to assess the effectiveness of anti-political dynasty laws that may come up in the future.

## E. Scope and Limitations

1. The scope of the system is limited to the Philippines.

2. This is not a replacement for statistical analysis software.

3. The tool provides visualizations only. It is up to the user to interpret the visualization presented.

4. The system accepts only quantitative or numerical datasets from the users, such as population and employment rate.

5. The system is limited to elected officials only.

6. The system does not consider political affiliations among family members, such as party affiliations or inter-family divisions. For example, if two individuals belong to the same family but are opponents of each other, they are still considered as members of the same political dynasty.

7. This tool does not get the data on its own. The administrators provide the election records that they shall compile from COMELEC and the users shall provide their own user datasets.

8. If parts of the election records are missing, then those missing officials will not be included in the visualization.

## F. Assumptions

1. The administrator provides accurate election records data.

2. Accurate family associations are provided to the system by the administrator. The administrator may do its own research to obtain accurate data.

3. The users have datasets in the proper format for uploading. The data format is described in detail in the fourth chapter.

4. The basic structure of local government units in the Philippines does not change radically. There will always be regions, provinces, municipalities, cities, and barangays.

# II. Review of Related Literature

There have been numerous data visualization projects and systems created for various purposes.

In 2012, the Department of Science and Technology (DOST) installed a responsive program called the Nationwide Operational Assessment of Hazards (NOAH) for disaster prevention and mitigation. NOAH aims to improve disaster management of local governments through the use of advanced science and technology. The program provides geo-hazard vulnerability maps, real-time weather maps, simulations, and other maps. NOAH gets the data from various sources, such as sensors and agencies. These are presented using visualization maps overlaid on a geographical map of the Philippines by Google Maps. The tool provides a web interface found at noah.dost.gov.ph where users can view such visualizations of real-time hazard data. [8]

Oak Ridge National Laboratory (ORNL) implemented a WebGIS to present information on global datasets archived through the ORNL Distributed Active Archive Center (DAAC). The researchers at ORNL used the Open Geospatial Consortium (OGC) standards to facilitate data-sharing and collaboration between multiple data sources. The current web interface can be found at webgis.ornl.gov/webgis/global. The interface provides visualizations for various global data, such as land cover, elevation, soil types, ecosystem, and climate maps. [9][10]

Mondrian is an interactive data visualization software for the Java platform. Mondrian can produce maps, histograms, barcharts, and scatterplots. It can also create special plots for high-dimensional data: mosaic plots for categorical data and parallel coordinate plots for continuous data. [11]

A study by Slingsby et al (2007) [12] used tag clouds and tag maps to visualize usage logs of a business directory by mobile telephone users. They used Yahoo's tag map applet and Google Earth. The tag map showed the relative prominence and local concentrations of business queries through the use of tags overlaid onto a geographical map. The tags are positioned at particular locations in particular

sizes according to the data being visualized. [12]

SOM Visualize is a software for visualizing multivariate regional health data in the Philippines. It used self-organizing maps to visualize patterns in the data and it also shows a geographical representation of the data through GIS visualization. A self-organizing map is an algorithm that reduces multidimensional data into a low-dimensional representation. [13]

HEALTH GeoJunction is a web application for finding scientific publications with geographical, temporal, and thematic filtering. It uses computational reasoning methods to extract place-time-concept information from papers and provides an interface for place-time-concept queries, filtering of queries, and contextualization of results. [14]

HealthMap is a project that aims to provide real-time information on disease outbreaks around the world. The web application, located at `healthmap.org`, provides real-time visualizations for disease outbreak monitoring and surveillance of emerging public health threats. The software utilizes several online sources, such as ProMED, World Health Organization, Google News, and more to gather its data. The data is then processed for visualization which is presented in the web interface.

Trendalyzer is an online interactive data visualization application created by Gapminder Foundation. It generates animated bubble charts of various global socio-economic data. Bubble charts are used to show three dimensions of data. Animation over time adds a fourth dimension to the visualization. The application is used to show trends within data. [7]

Tolentino (2012) [15] created an application that generates crisis maps from crowd-sourced data to help in disaster relief. Laguna, Philippines was the scope area of the study. The technologies used were Facebook and Google Maps. Information to be mapped was gathered from social media users through a Facebook application. The collected data was then integrated with Google Maps to generate a visualization of the disaster. The application was found at `apps.facebook.`

com/lagunacrisismap (It is not available anymore). [15]

Limosa is an interactive system for visualization of geographic user interests in Twitter. Geographic user interests were mined from Twitter users using their visited locations and mentioned locations. The system provides various visualizations displaying information in different contexts, such as the geographic regions visited by a certain user, the top topics and related users in a particular region, and the top geographic topics related to a term. [16]

Hauger and Schedl (2014) [17] created a system that maps the music listening patterns of Twitter users. The system mines microblog data using the Twitter Streaming API, processes the data using clustering, and visualizes the resulting information on a map through a web interface using the Google Maps API. [17]

TwitInfo is a system for aggregating and visualizing events on Twitter. It provides a web interface for exploration of the timeline and geolocation of events on Twitter. The information is updated in realtime by capturing live streams from Twitter. [18]

Dengue-GIS is a web-based surveillance geographic information system for the collection, analysis, and reporting of dengue cases in Mexico. The system implements epidemiological and entomological surveillance tools and functionalities for the integrated analyses of the geo-referenced input data. The system generates visualizations that show the geographic distribution of cases and transmission risk maps. [19]

ConTraffic is a web GIS for the interactive visualization of container movements. It tracks cargo containers around the globe for monitoring. The system features geographical visualization of container events and semantic summarization of records. [20]

Taggram combines texts and maps for geo-tagged data exploration. It uses the concept of tag clouds or tag maps to visualize labels in maps. Tag positions are computed to fit inside an arbitrarily-shaped region. Data exploration for other tags is handled through the use of fisheye-menu based interaction and dynamic

display of tags. [21]

Indiemapper is a geographic data visualization tool that allows users to upload their own datasets for visualization. It is used to make thematic maps, such as bivariate maps, choropleth maps, proportional symbol maps, dot density maps, and cartograms. These maps can then be exported into image formats. [22]

# III. Theoretical Framework

## A. Political dynasty

There are many definitions of political dynasties. Generally, a political dynasty is defined as "a family or group that maintains power for several generations." [23] Specific definitions of political dynasties differ in the details, such as what constitutes a family, how far into past generations can individuals be related, how distant can relations be, whether physical distance or blood distance, and the consideration of relations other than bloodline relations.

The Anti-Political Dynasty Bill in the Philippines defines a political dynasty as existing when two or more related persons "run simultaneously for elective public office within the same municipality, city, or province," and also when a person related to an incumbent official "holds or runs an elective office simultaneously with the incumbent official within the same province" or succeeds the incumbent elective official. Related persons are defined as one person being the spouse of the other, being a brother or sister, and being a direct descendant or ascendant (in other words, second civil degree of consanguinity or affinity). Half-blood, illegitimate, and foster relations qualify as "related." [24]

## B. Dynastic measures

In order to quantify political dynasties, Mendoza et al (2013) [2] developed the following indicators: DYNSHA, DYNLAR, and DYNHERF.

DYNSHA is the measure of the share of dynasties in elected positions for each province. It indicates the number of dynastic officials in a province. They don't need to belong to the same political dynasty to be counted. For this variable, dynasties are defined as those elected officials in 2010 with relatives in 2010 and 2007. For example, if there are three dynastic officials elected in a province, then the DYNSHA value for this province is three (3).

DYNLAR indicates the number of positions occupied by the largest dynasty

in each province. This is different from the DYNSHA indicator. For example, if there are six dynastic offcials elected in a province, and four of them belong to the same dynasty, while the other two belongs to another dynasty, then the DYNLAR value for this case is four (4).

DYNHERF is a variant of the Hirschman-Herfindahl index applied to political dynasties. This is essentially "the sum of squared shares of the total positions of each political dynasty in each province." For example, suppose there are three dynasties in a province, with seven dynastic officials elected. Four officials are from one dynasty, two are from the second dynasty, and one is from the third dynasty. Then the DYNHERF value for this province is $4^2 + 2^2 + 1^2 = 21$. DYNHERF is "a more nuanced indicator of the concentration of political power because it will put a corresponding greater weight on those political dynasties with larger shares of the total positions (while also considering the other "fat" dynasties in the province)."

These three indicators are measured per province in the original study. For this project, the indicators were adapted for the regional, municipal, and barangay levels, for different years, and for creating a per dynasty indicator.

## C. Choropleth map

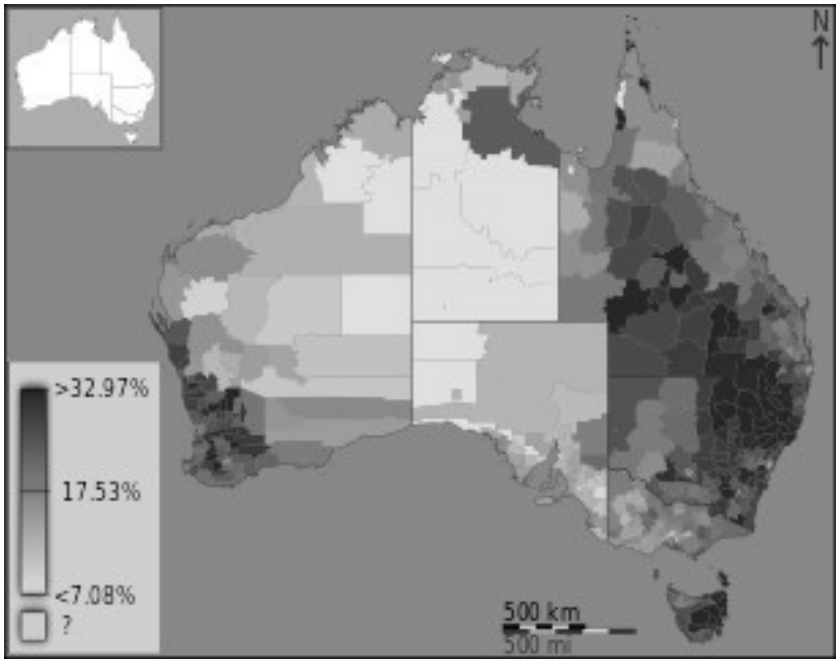


Figure 1: Choropleth map showing the fraction of Australians that identified as Anglican in 2011.
©Toby Hudson based on data from the Australian Bureau of Statistics.

A choropleth is "a map showing the distribution of a phenomenon, usually using various colors. Color gradations are correlated to the density per unit area of the phenomenon." [25] It can be used to show geographical variables. Figure 1 shows an example of a choropleth map. It shows a map of Australia with different regions colored according to the value of the statistic in that region. The legend on the lower-left corner shows the values corresponding to the colors used.

## D. Tag cloud

Figure 2: Tag cloud on a map of the United States showing relative frequencies of surnames arranged by geographical location.
Retreived from http://ngm.nationalgeographic.com/2011/02/geography/usa-surnames-interactive. ©National Geographic Society.

Tag clouds are visualizations of text. It shows words or terms in different font sizes as to show the relative frequency or weight of each word or term. [26] It can be used to show the relative importance or relevance of different tags. Tags can also be color-coded to represent other variables, such as the category of the tag. The positioning of the tags in a tag cloud can also be used to convey other information, e.g., by grouping related tags.

Figure 3 shows a tag cloud of surnames in the United States. The surnames are arranged by the geographical location they occur in, with high-frequency names appearing relatively larger. The map only shows the top 25 surnames in each state. The apparent density of the names correlate with the population, thus, the east is denser than the central area.

## E. Philippine political structure

The Philippines is divided into political units in four hierarchical levels: region, province, city or municipality, and barangay.

The barangay is the smallest government unit in the Philippines. Elective positions in the barangay level include the punong barangay, seven sangguniang barangay members, and the sangguniang kabataan positions. [27]

Barangays are grouped under cities or municipalities. The city or municipality is the next level of political subdivision. There are different types of cities and municipalities in the Philippine government: component cities, independent cities, and municipalities. Component cities and municipalities are grouped under provinces, while independent cities are independent of the province. For the elective offices, municipalities and cities have a mayor, a vice-mayor, and sanggunian members. [27]

Provinces are the next level of administrative division. A province is an aggregation of cities or municipalities. As previously stated, there are independent cities which are not part of a province. Independent cities are on the same administrative level as provinces. The elective provincial offices are: the governor, the vice-governor, sangguniang panlalawigan members, and congressional positions. [27]

Provinces are grouped into regions. The region is the largest administrative division of the Philippines. Regions have no local government, except for autonomous regions.

Autonomous regions are different. Autonomous regions have local government at the regional level. The autonomous regional government can also choose to enact its own local government code, or apply the Local Government Code of the Philippines, as described in the previous paragraphs. [27] The only autonomous region in the Philippines at the time of writing is the Autonomous Region in Muslim Mindanao (ARMM). The ARMM has its own government. Elective officials of the ARMM are: the Regional Governor, the Vice-Governor, and members of

the Regional Assembly. The lower local government units in ARMM (provinces, municipalities, etc.) still follows the national Local Government Code. [28] As of writing, there is an ongoing process to replace the ARMM with a new government, the Bangsamoro political entity, which could potentially redefine the structure of local government units in the region. [29]

## F. Philippine Standard Geographic Code

The Philippine Standard Geographic Code (PSGC) [30] is the standard for the coding of geographic areas of the Philippines. It was developed by the National Statistical Coordination Board (NSCB) in coordination with other government bodies. One of the uses of the PSGC is for the development of databases and geographic information systems. [31]

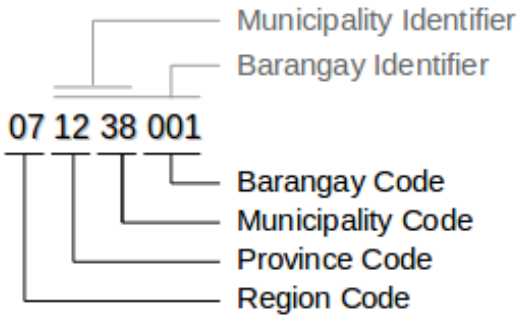


Figure 3: The structure of the PSGC

The string consists of nine digits. The first two digits is the Region Code, which identifies an administrative region. [32]

The next two digits is the Province Code, followed by the two-digit Municipality Code. The Province Code and the Municipality Code constitute the four-digit Municipality Identifier, which identifies a municipality or a city. [32]

The remaining three digits is the Barangay Code. The Barangay Identifier is composed of the Municipality Identifier and the Barangay Code. The Barangay Code is dependent on the Municipality Identifier to fully identify one barangay. [32]

There are special cases in PSGC coding. For the case of cities independent from a province, the independent city is assigned its own Province Code in addition to the Municipality Code, i.e., the independent city has its own "province." For example, Cotabato City is coded as 129804000. The city's Municipality Identifier is 9804. The Municipality Code 04 represents Cotabato City and the Province Code 98 represents Cotabato City as well. The barangay level is unchanged. [30]

In the National Capital Region (NCR), the districts act as "provinces" in PSGC coding. Each of the four districts in NCR is assigned its own Province Code. [30] Another exception is the first district of NCR, which is the City of Manila. In PSGC, Manila is assigned a province level with a Province Code of 39. There are no sub-municipalities under Manila, hence, "places" are used for the municipal levels instead. Ermita, Quiapo, and Tondo are such "places" in Manila. The barangay level is unchanged. [30]

## G. Geographic information system

A geographic information system (GIS) is an information system that primarily features a geospatial reference system. A GIS is designed to work with geographically referenced data, such as industrial sites in a city, bus routes of a transportation company, and disease outbreak clusters in an area.

Geographic information systems have means of geographic data input, storage, retrieval and query; geographic data analysis and spatial statistics; and data reporting, such as maps. These functionalities are integrated together in a comprehensive GIS.

There is a multitude of geographic information systems these days developed for various specific purposes to fit certain demands. Such uses are urban planning, demographic monitoring and forecasting, epidemic surveillance, and disaster management and response. There are also general-purpose GIS software packages that can be applied in a wide range of individual needs. [33]

## H. GeoJSON

GeoJSON is a format for encoding geographical features. It is built upon the JSON format. GeoJSON objects can encode points, polygons, and other geographical features using a data structure defined by the format. GeoJSON objects can also contain metadata, or additional properties for a geographical feature.

A GeoJSON object can be any of the following types: "Point", "MultiPoint", "LineString", "MultiLineString", "Polygon", "MultiPolygon", "GeometryCollection", "Feature", and "FeatureCollection". The type string is stored in the "type" attribute of the object.

Geometry objects are GeoJSON objects with a type that is neither "Feature" nor "FeatureCollection". Geometry objects with a type other than "GeometryCollection" must have a "coordinates" attribute, which defines the shape or position of the geometry. The value of the "coordinates" depend of the type of the geometry object. GeometryCollection-type objects must contain a "geometries" attribute containing an array of other geometry objects.

An example geometry object, which encodes a polygon, follows:

```
{
  "type": "Polygon",
  "coordinates": [
    [ [100.0, 0.0], [101.0, 0.0], [101.0, 1.0],
      [100.0, 1.0], [100.0,0.0] ]
  ]
}
```

Feature objects are GeoJSON objects with the type "Feature". A feature object contains a geometry object, a "properties" attribute, and an optional "id" attribute. The properties attribute can be any JSON object, and is used for additional properties of the geographical feature, such as its name. FeatureCollection objects, similar to GeometryCollection objects, are objects that contain an array of feature objects, stored in a "features" attribute. [34]

# IV. Design and Implementation

## A. Use case diagram

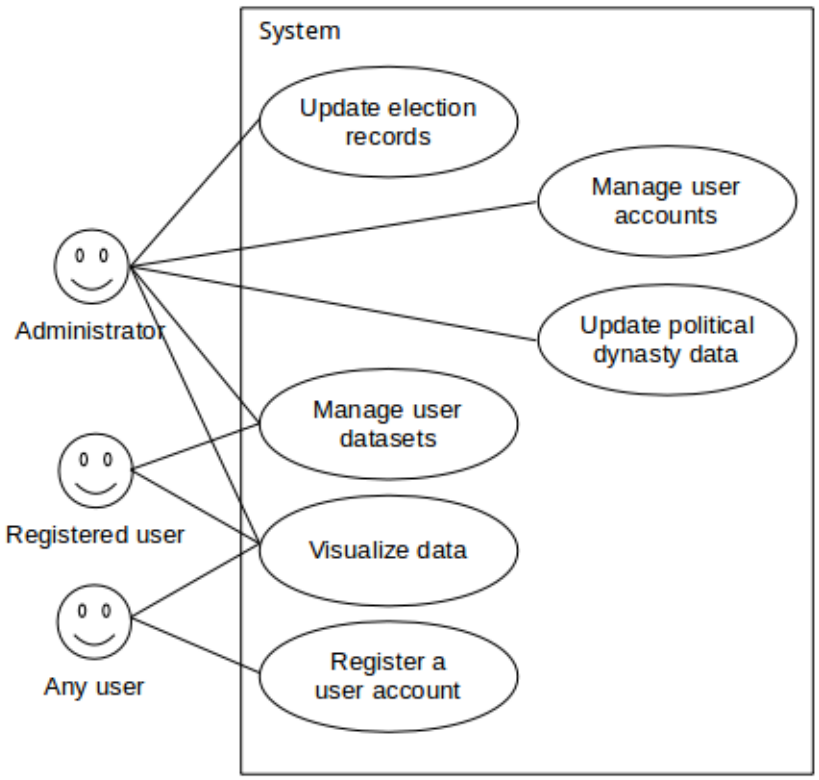


Figure 4: Overview use case diagram

There are three types of users: the administrator, the registered user, and the unregistered user (any user).

Administrators perform administrative tasks. They can manage user accounts and user datasets. Deleting a user account will also delete all the user datasets uploaded by that user. They are also in charge of updating other data of the system, which are the election records and the dynasty data.

Updating election records can be done by uploading a file containing the records or manually entering data through a user interface. Administrators shall enter the individuals who are or have been elected, the instances when they were elected, their positions at those instances, party affiliations, and various related information.

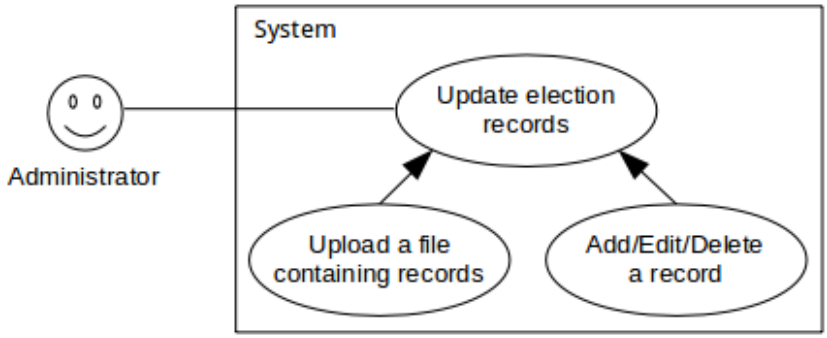


Figure 5: Use case diagram for updating election records

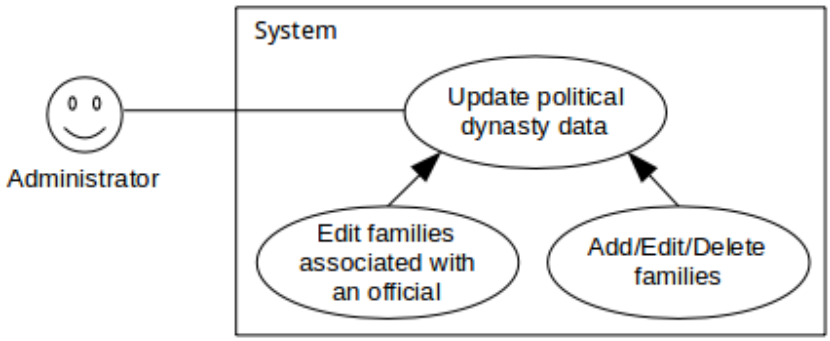


Figure 6: Use case diagram for updating political family associations

The administrator can add or remove family associations of an official, and add, edit, or delete families.

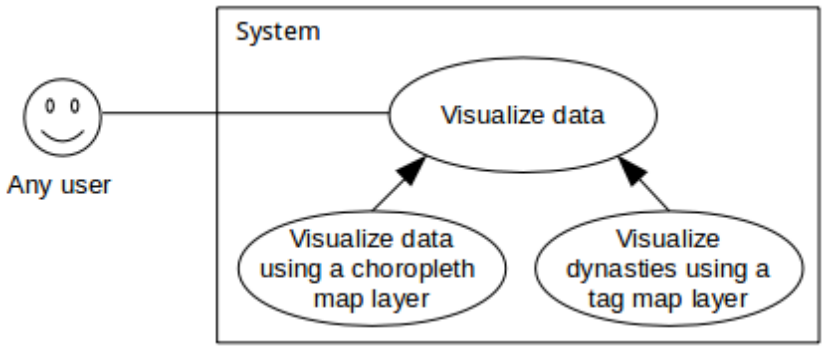


Figure 7: Use case diagram for visualizing data

Any user can use the visualization functions of the software. The visualizations provided by the system are interactive and animated choropleth and tag maps. The data to be visualized come from the user-provided datasets and the system-generated dynasty data. The choropleth map will be used to visualize variables from the user datasets or the dynasty indicator variables. The tag map will be

used to visualize local dynasties. The user can choose which data or variable to use and which types of visualization to generate.

Any user can also register a user account to become a registered user. Registration requires a username and a password. Once registered, they can login and act as a registered user.

Registered users are users who are logged in using their user account. They can upload datasets and, like any other user, visualize the data in the system. The uploaded datasets will be available for everyone to use for visualization.

## B. Context diagram

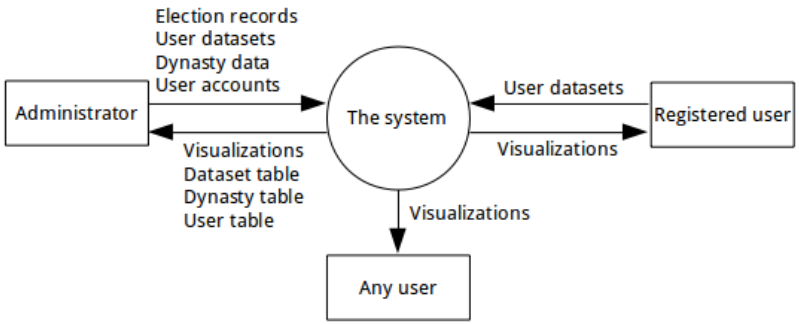


Figure 8: High-level context diagram of the system

The system interfaces with three types of entities: the administrators, the registered users, and the unregistered users (any user).

The administrators are the system's main source of election records data. It is out the scope of the system how administrators get the election records.

Datasets other than the election records and the dynasty data can be uploaded into the system by any registered user or administrator. These datasets are called user-provided datasets or user datasets.

Visualizations provided by the system can be viewed by any user.

## C. Entity relationship diagram

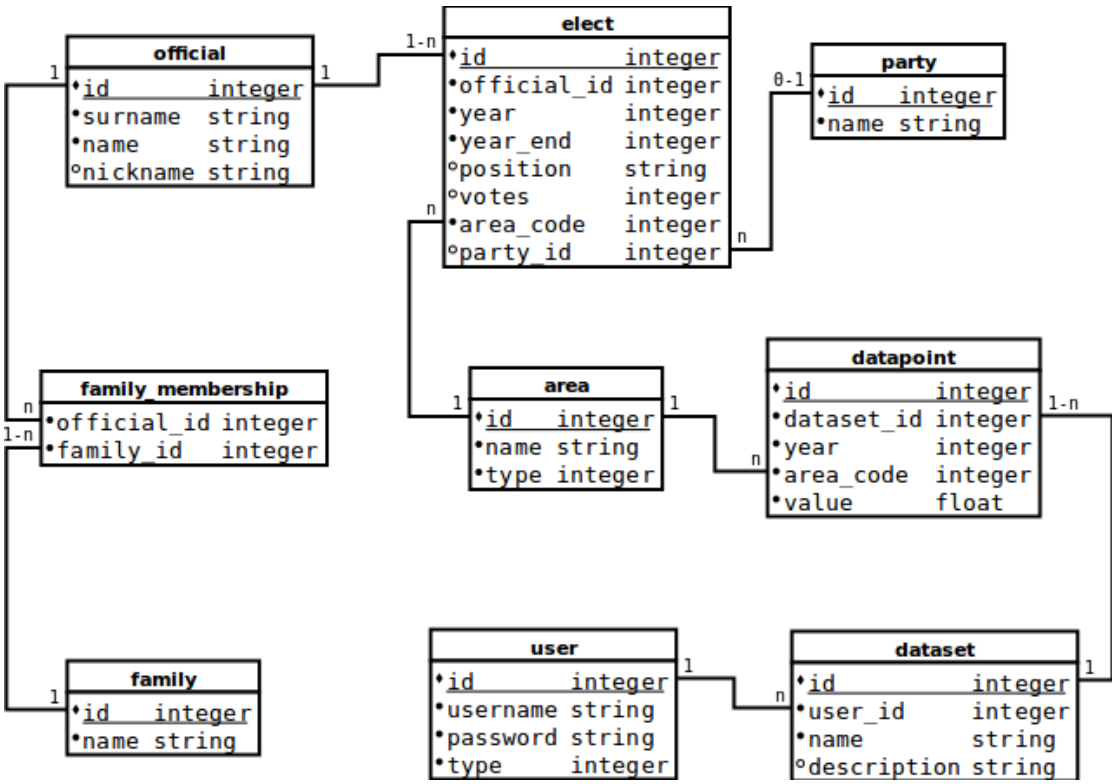


Figure 9: Entity relationship diagram of the system

## D. Data dictionary

| Field | Type | Description |
|---|---|---|
| id | integer | Primary key |
| surname | string | Surname of the official |
| name | string | Full name of the official excluding the surname in the following format: "[Given name] [Middle name]" |
| nickname | string | Nickname of the official |

Table 1: official table

| Field | Type | Description |
|---|---|---|
| id | integer | Primary key |
| name | string | Name of the family. This is usually the common surname of most of its members. |

Table 2: family table

| Field | Type | Description |
|---|---|---|
| official_id | integer | Primary key component, foreign key to *official* table |
| family_id | integer | Primary key component, foreign key to *family* table |

Table 3: family_membership table

| Field | Type | Description |
|---|---|---|
| id | integer | Primary key |
| official_id | integer | Foreign key to *official* table. The elected official |
| year | integer | Year elected |
| year_end | integer | Year the official's post ended |
| position | string | Description of the position where the official was elected |
| votes | integer | Votes obtained |
| area_code | integer | Foreign key to *area* table. The province or region where the official was elected |
| party_id | integer | Foreign key to *party* table. The party list where the official belongs |

Table 4: elect table

| Field | Type | Description |
|---|---|---|
| id | integer | Primary key |
| name | string | Name of the party list |

Table 5: party table

| Field | Type | Description |
|---|---|---|
| id | integer | Primary key |
| code | integer | Area's code in the Philippine Standard Geographic Code (PSGC) [30] |
| name | string | Name of the area |
| type | integer | Type/level of the area according to PSGC. The following coding is used: 0 – Region, 1 – Province, 2 – Municipal, 3 – Barangay. |

Table 6: area table

| Field | Type | Description |
|---|---|---|
| id | integer | Primary key |
| user_id | integer | Foreign key to *user* table. The user that uploaded this dataset |
| type | integer | Type of the dataset, whether per area or per family (tags). The following coding is used: 0 - Area, 1 - Family. |
| name | string | Name of the dataset |
| description | string | Description of the dataset |

Table 7: area table

| Field | Type | Description |
|---|---|---|
| id | integer | Primary key |
| dataset_id | integer | Foreign key to *dataset* table. The area-type dataset where this data point belongs |
| year | integer | Year of the data point |
| area_code | integer | Foreign key to *area* table. The province or area being described by this data point |
| value | float | Value of the data point |

Table 8: datapoint table

| Field | Type | Description |
|---|---|---|
| id | integer | Primary key |
| dataset_id | integer | Foreign key to *dataset* table. The family-type dataset where this data point belongs |
| year | integer | Year of the data point |
| area_code | integer | Foreign key to *area* table. The province or area being described by this data point |
| family_id | integer | Foreign key to *family* table. The family being described by this data point |
| value | float | Value of the data point |

Table 9: tagdatapoint table

| Field | Type | Description |
|---|---|---|
| id | integer | Primary key |
| username | string | Username |
| active | boolean | Whether the user is activated or deactivated |
| pw_hash | string | Password hash |
| type | integer | Type of account. The following coding is used: 0 – Regular user, 1 – Administrator. |
| salt | string | Salt for the password hash |

Table 10: user table

## E. Wireframe

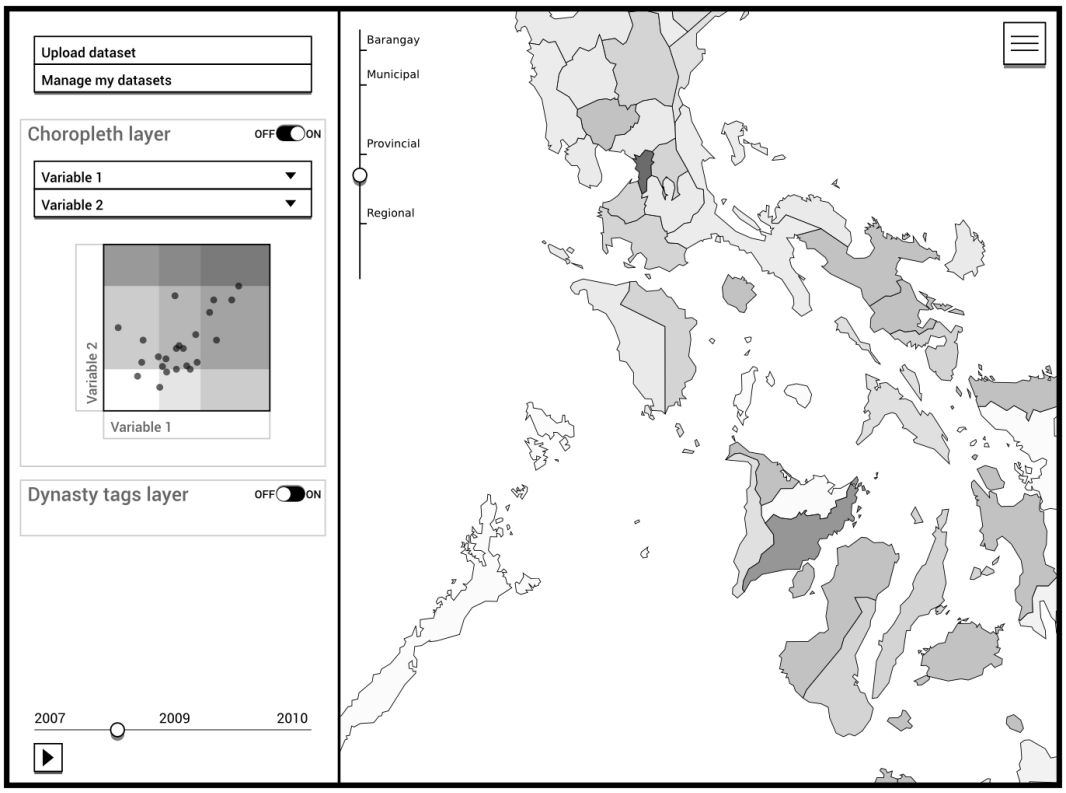


Figure 10: Wireframe model of the main client interface. The bivariate choropleth map layer is activated, showing two variables on the map. A scatter plot of the data is shown.

The choropleth layer was used to visualize the user-provided data and the dynasty indicator variables on the Philippine map. The choropleth map can show either a single variable or two variables at a time. The variables can be selected from the user-provided datasets or from any of the three dynasty indicator variables (DYNSHA, DYNLAR, and DYNHERF) computed by the system from the dynasty data. Users can set the level of detail (regional, provincial, municipal, or barangay level) of the visualization.

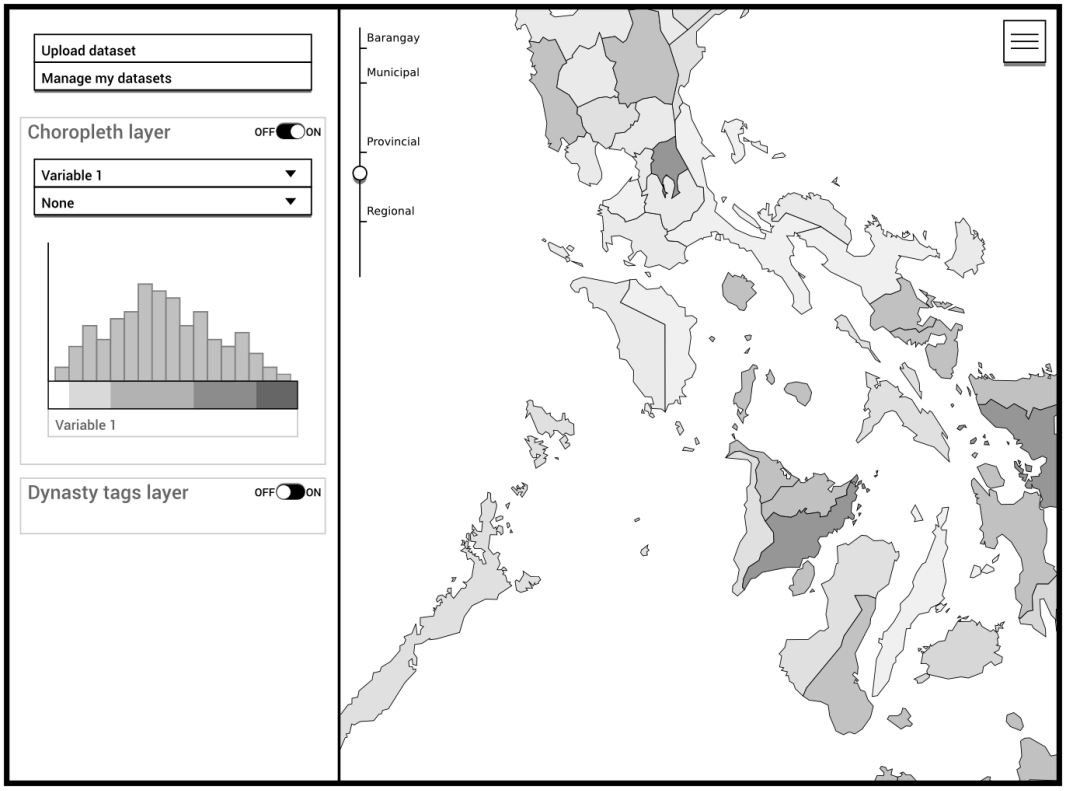


Figure 11: The univariate choropleth map shows a single variable on the map. A histogram of the data is shown.

The tag map was used to visualize local dynasties on the Philippine map. The tag map was overlaid on the choropleth map. One tag represents one local dynasty. The tag size is proportional to the associated local dynasty's size, computed using the LocalDynastySize variable (which is described in a later section of this chapter). Tags are positioned according to the geographical location of the associated dynasty. Dynasties spanning more than one area will have separate tags for each area. Users can set the level of detail to display.

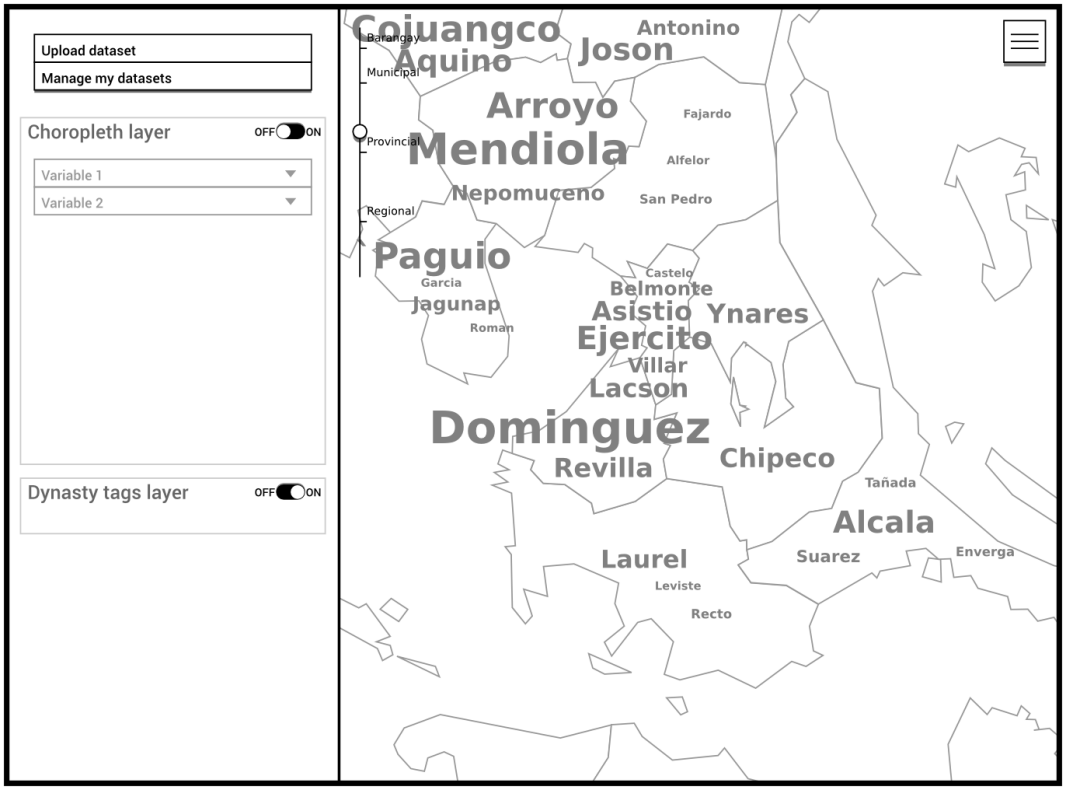


Figure 12: The tag map layer shows the relative sizes of dynasties of different areas.

## F. Indicators

There are several political dynasty indicators to be computed by the system. To define these computations, some functions and variables are introduced.

Let $F$ be the set of all political families, and $A$ be the set of all areas (regions, provinces, cities, municipalities, or barangays).

The function $Subarea(a)$ equals the set of subareas under area $a$. The PSGC's hierarchical organization was used to determine which areas are subareas of an area.

The function $Officials(a, t)$ equals the set of officials elected in area $a$ at time $t$.

$Members(f, t)$ equals the set of individuals belonging to political family $f$ who have been elected on or after time $t$.

An additional function $LocalDynastySize(a, f, t)$ is the set of officials elected in area $a$ at time $t$ and belonging to political dynasty $f$. It can be computed using the following form:

$$LocalDynastySize(a, f, t) = Officials(a, t) \cap Members(f, t)$$

### F..1 DYNSHA

$DYNSHA$ [2] is an indicator variable for the share of dynastic positions in an area.

$$DYNSHA(a, t) = \left| Officials(a, t) \cap \bigcup_{f \in F} Members(f, t) \right| \div |Officials(a, t)|$$

### F..2 DYNLAR

$DYNLAR$ [2] is the share of positions of the largest dynasty in an area. (Note: $LocalDynastySize$ is abbreviated as $LDS$.)

Let $F = f_1, f_2, \ldots, f_n$,

$$DYNLAR(a, t) = \frac{max\left[|LDS(a, f_1, t)|, |LDS(a, f_2, t)|, \ldots, |LDS(a, f_n, t)|\right]}{|Officials(a, t)|}$$

### F..3 DYNHERF

$DYNHERF$ [2] is the sum of squared shares of the total positions of each political dynasty in an area.

$$DYNHERF(a, t) = \sum_{f \in F} \left(LDS(a, f, t) \div |Officials(a, t)|\right)^2$$

### F..4 Measures of dynasty size

The $DYNSHA$, $DYNLAR$, and $DYNHERF$ variables describe the dynastic share in an area. Using these variables as basis, additional indicator variables were developed for the purpose of describing the size of a dynasty in an area.

*RecursiveDynastySize* (*RDS*) is the number of elected members of a political dynasty in an area, in all subareas of that area, and in all subareas of subareas. This can be used to show the size of political dynasties per area. The definition is recursive:

$$RDS(a, f, t) = |LDS(a, f, t)| + \sum_{s \in Subarea(a)} RDS(s, f, t)$$

The recursion stops at the barangay level since the *Subarea* function is null in the case of barangays.

*TotalDynastySize* is the number of elected members of a political dynasty in all areas. This can be used to show the top dynasties in the country.

$$TotalDynastySize(f, t) = \sum_{a \in A} |LDS(a, f, t)|$$

## G. Datasets format

The system accepts user datasets in a comma-separated value (CSV) file format. The fields of the file are area and years. Specifically, the first column is the area field and the succeeding columns are fields for each year in the dataset. The first line of the file will contain the column labels, and the succeeding lines will contain the data rows. Missing data is encoded by an empty string. The Philippine Standard Geographic Code (PSGC) [30] was used for the coding of areas.

A sample of the format looks like the following (data sampled from gross enrollment ratio for public elementary schools):

```
area,2011,2010,2009,2008,2007,2006,2005
010000000,99.45,98.42,98.47,98.38,98.05,98.32,90.84
020000000,106.86,105.04,104.47,104.13,103.02,102.76,88.56
030000000,91.85,91.90,92.44,92.44,92.64,92.95,91.97
```

Which represents the following data:

| area code | 2011 | 2010 | 2009 | 2008 | 2007 | 2006 | 2005 |
|---|---|---|---|---|---|---|---|
| 010000000 | 99.45 | 98.42 | 98.47 | 98.38 | 98.05 | 90.32 | 90.84 |
| 020000000 | 106.86 | 105.04 | 104.47 | 104.13 | 103.02 | 102.76 | 88.56 |
| 030000000 | 91.85 | 91.90 | 92.44 | 92.44 | 92.64 | 92.95 | 91.97 |

Table 11: Example dataset CSV file

Not all datasets use PSGC. Area strings can be the common names of the areas, such as "NCR", "Region IV-A", or "Abra". The system is able to convert these informal names into PSGC internally using a mapping. However, all cases may not be handled since there are multiple ways to name an area.

## H. Election records format

The format of the election records file follows a CSV file format. The fields are the area code, the starting year, position, surname of the official, the rest of the name of the official (the given name with the middle name), the official's nickname, the official's party, and finally the votes obtained. The required fields are the area code, the year, and the official's name fields. Non-required fields can be left blank using an empty string. The party field of independent officials, those without party affiliation during that election, is encoded using an empty string or the string "IND".

An sample line of the format follows:

```
060402000,2010,COUNCILOR,FELICIANO,CIRIACO TOLENTINO,ACOY,NP,6502
```

This line encodes one election record, for Ciriaco "Acoy" Tolentino Feliciano, councilor of Balete, Aklan for the term of 2010-2013, affiliated with NP, with 6502 votes during that election.

## I. GeoJSON format

The system allows for updating the geometry of the map using the GeoJSON format. There are four levels of geometry: regional level, provincial level, municipal

level, and barangay level. For each level, the user can update the geometry of the areas by uploading a GeoJSON file that uses the following format: For each Feature that encodes one area, it shall have a "PSGC" property in its "properties", with the value equal to the PSGC code of that area.

An example Feature object follows:

```
{
  "type": "Feature",
  "geometry": {
    "type": "Polygon",
    "coordinates": [
      [ [100.0, 0.0], [101.0, 0.0], [101.0, 1.0],
      [100.0, 1.0], [100.0, 0.0] ]
    ]
  },
  "properties": {
    "PSGC": 120423000
  }
}
```

The PSGC code is required for all Feature objects that define an area. Other properties are allowed, but they are just ignored by the system.

## J. PSGC table format

The system allows for the bulk uploading of area data. Area data can be uploaded in a CSV file format that follows these specifications:

One area is encoded in one row or line of the file. The row must contain the nine-digit PSGC code, and the area's name. The order does not matter. The row may contain other fields, and the system will try to determine the appropriate fields. The file may also contain rows that don't encode any area, such as empty rows and header rows. The system just ignores these invalid rows. Duplicate

PSGC code is resolved by using the row the appears last in the file.

A sample of the PSGC table file follows:

```
PHILIPPINE STANDARD GEOGRAPHIC CODE (PSGC),,
"As of June 30, 2015",,
RegProvMunBgy,RuralUrban*,Name
130000000,,NATIONAL CAPITAL REGION (NCR)
133900000,,"NCR, CITY OF MANILA, FIRST DISTRICT (Not a Province)"
133900000,,CITY OF MANILA
133901000,,TONDO I / II
133901001,U,Barangay 1
133901002,U,Barangay 2
133901003,U,Barangay 3
```

This sample is the first 10 lines of the official PSGC table obtained from NSCB. In this example, the first three lines of the file will be discarded. The second field (RuralUrban) will also be ignored by the system. The multiple rows that encode for 133900000 will be resolved to the last row, which is the "CITY OF MANILA" row.

The flexibility in the format allows the user to directly upload the official PSGC table which can be obtained from the NSCB website. However, it is recommended that the file consistently contain only valid rows with only the PSGC and the name fields, in that order, to avoid parsing errors.

## K. Technical architecture

The server software requires at least the following:

1. One gigabyte of memory
2. Apache 2.4
3. MySQL 5.5
4. PHP 5.6

5. Python 2.7
    (a) python-requests
    (b) python-geojson
6. Internet connectivity

The client is recommended to have the following to be able to use the system well:

1. Two gigabytes of memory
2. Two-gigahertz processor
3. Decent graphics processing unit
4. Modern web browser (e.g. Chrome 43, Firefox 38)
5. Reliable Internet connectivity

# V. Results

## A. Login page

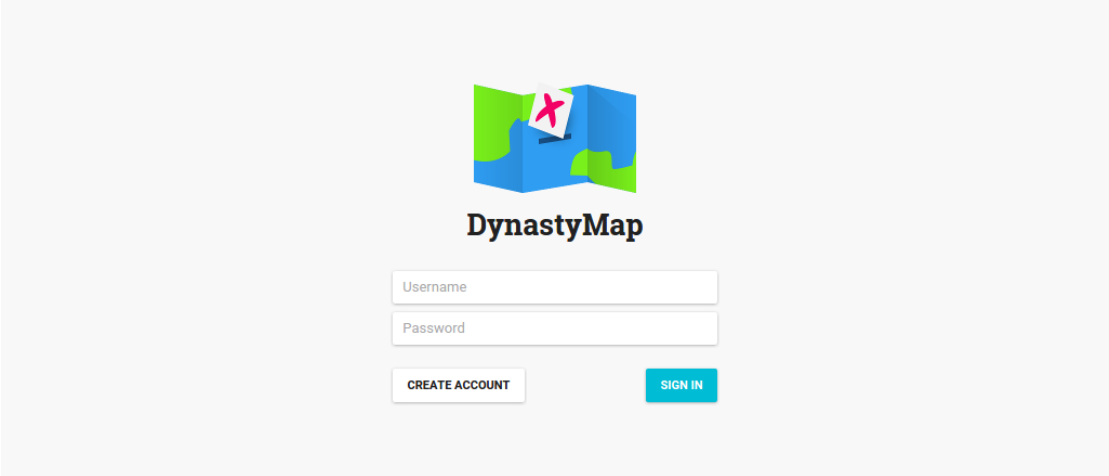


Figure 13: Login page

Fig. 13 shows the login page for users and administrators who want to log in. Login and registration requires a username and a password.

## B. Administration pages

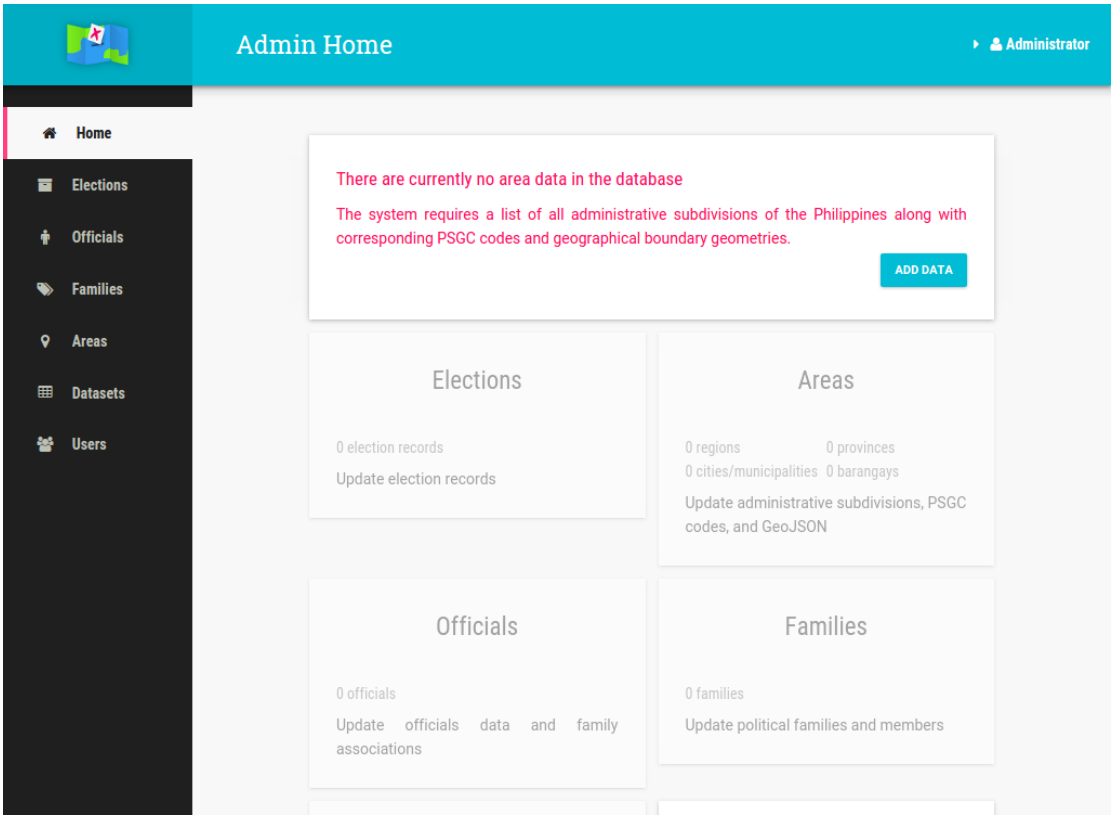


Figure 14: Administrator home page

Fig. 14 shows the home page of the system administrator. The home page shows messages about issues the administrator needs to address, such as missing required data. The sidebar menu contains a list of all the administration pages: Elections, Officials, Families, Areas, Datasets, and Users. Each option leads to a page for performing various administrative tasks.

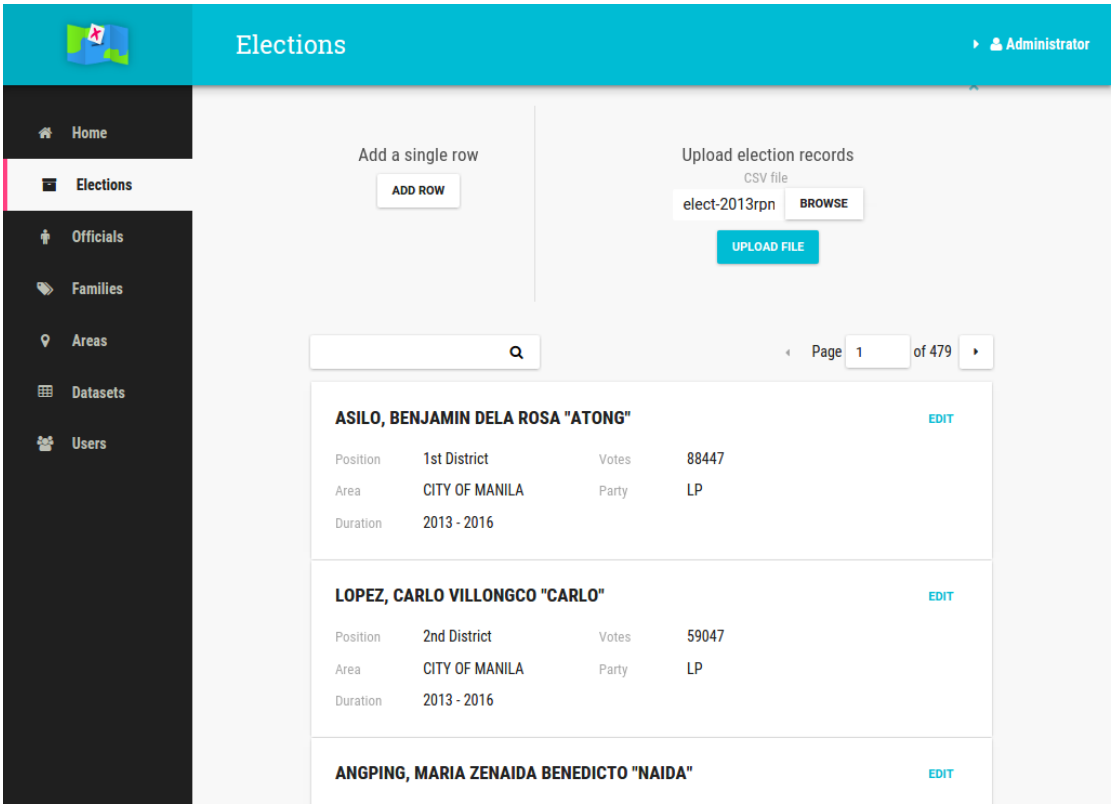


Figure 15: Elections list page

Fig. 15 shows the elections page for managing election records data. The page lists all election records stored in the system. The administrator can add more data using either the single row option or the file upload option. Any row of data can be edited or deleted in-place.

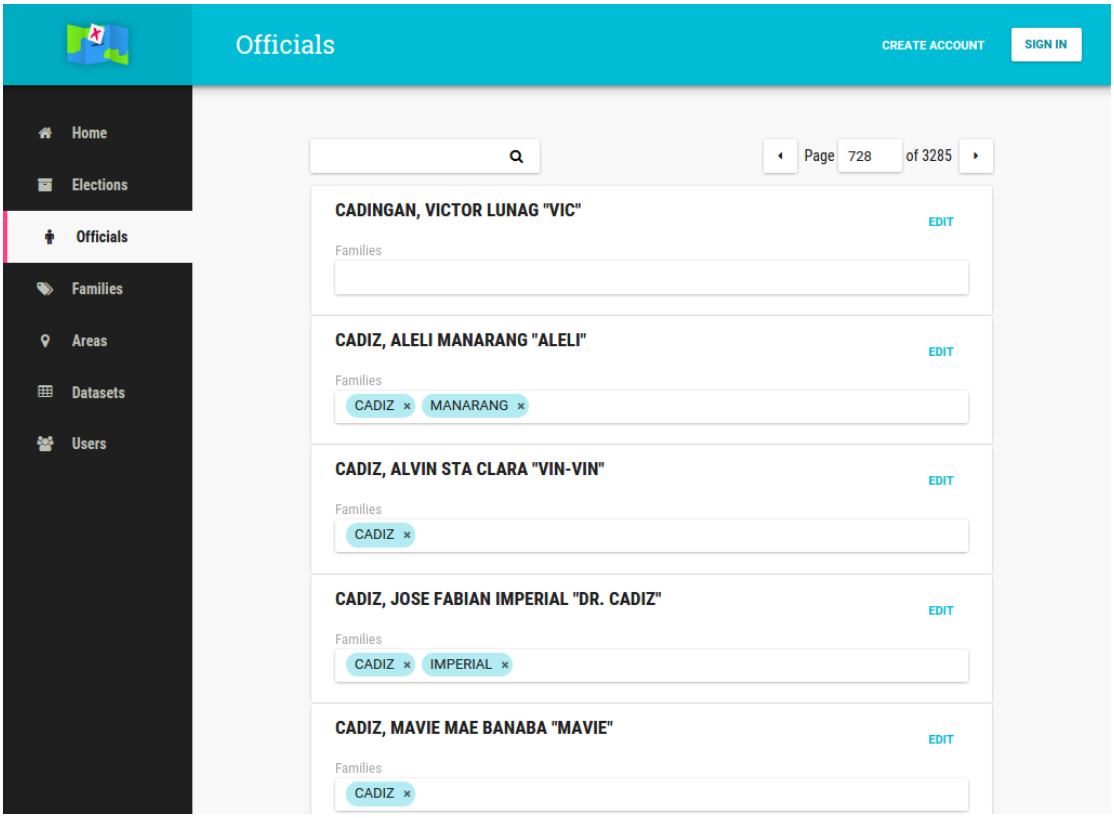


Figure 16: Officials list page

Fig. 16 shows the officials page for managing officials data and family associations (dynasty) data. There is a list of all elected officials stored in the system. The families associated with each official is shown and can be modified using the token list input. Officials' names can be edited using the Edit button.

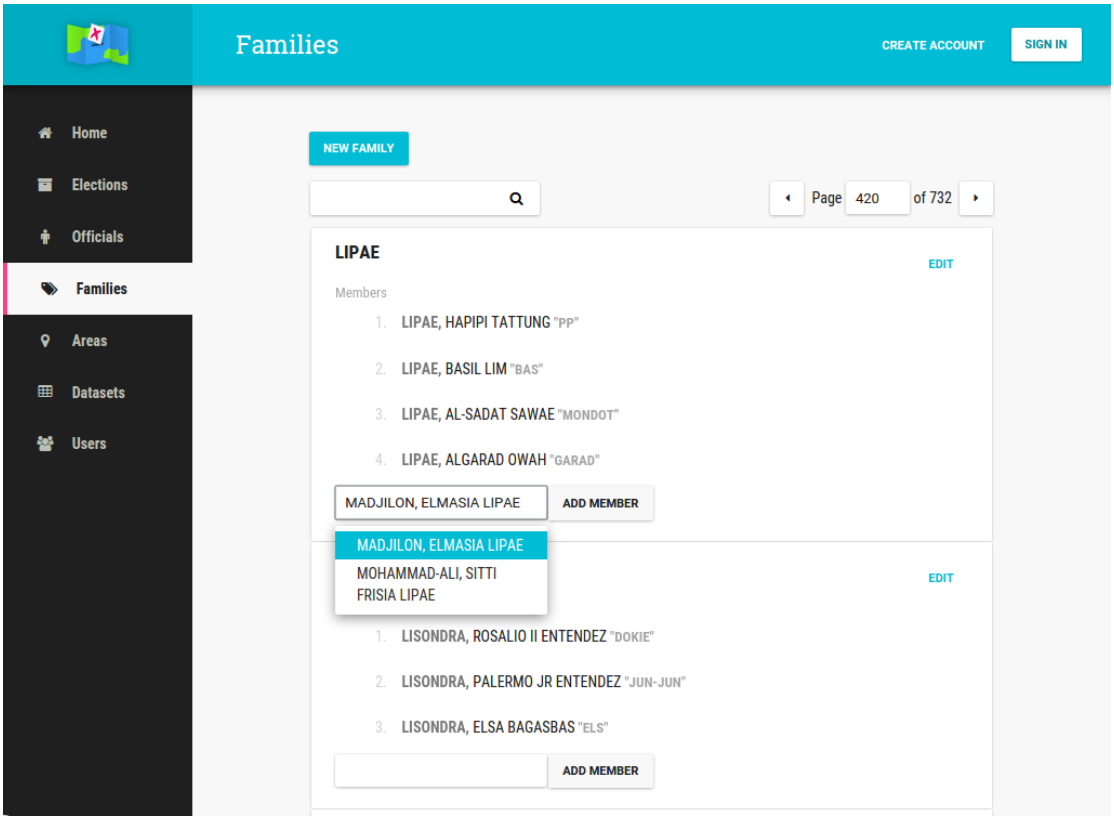


Figure 17: Families list page

Fig. 17 shows the families page for managing family data and family members data. Families and their members are listed. Families can be added, edited, and removed. Individual family members can be added and removed.

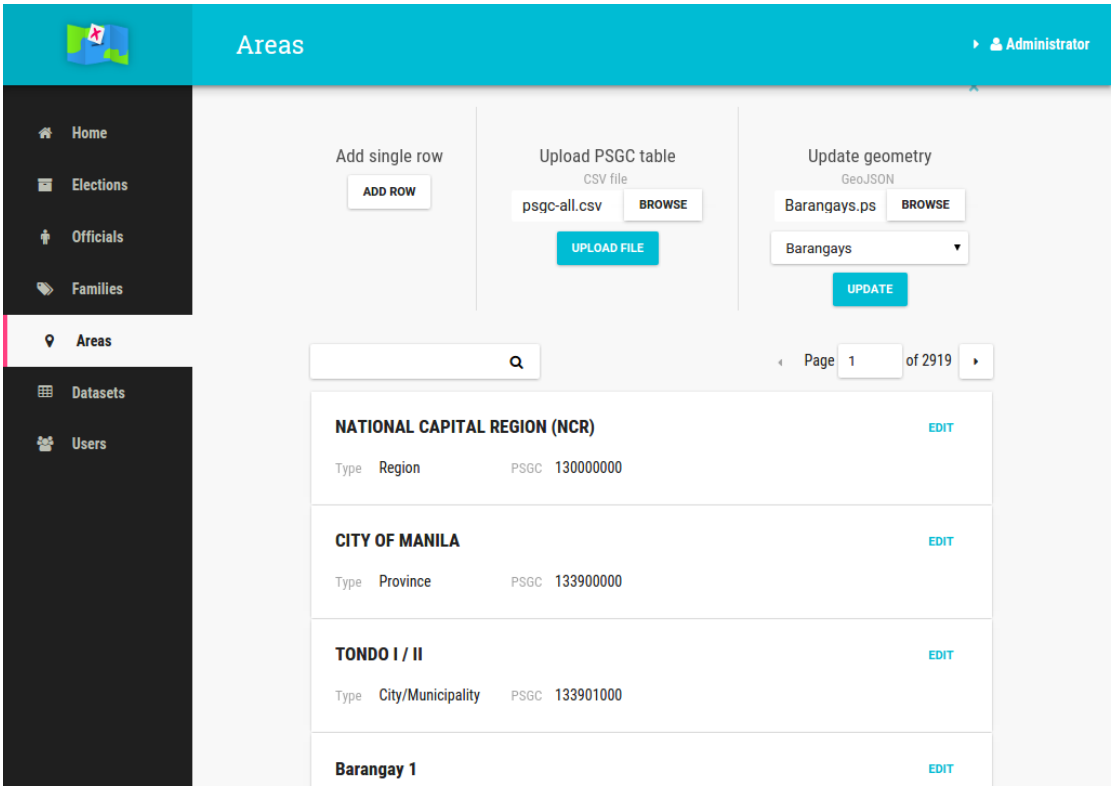


Figure 18: Areas list page

Fig. 18 shows the areas page for managing area data. Area data can be added using the single row option or the file upload option. The areas GeoJSON geometry can also be updated through a file upload method. Finally, a list of all areas allow editing and removal of individual areas.

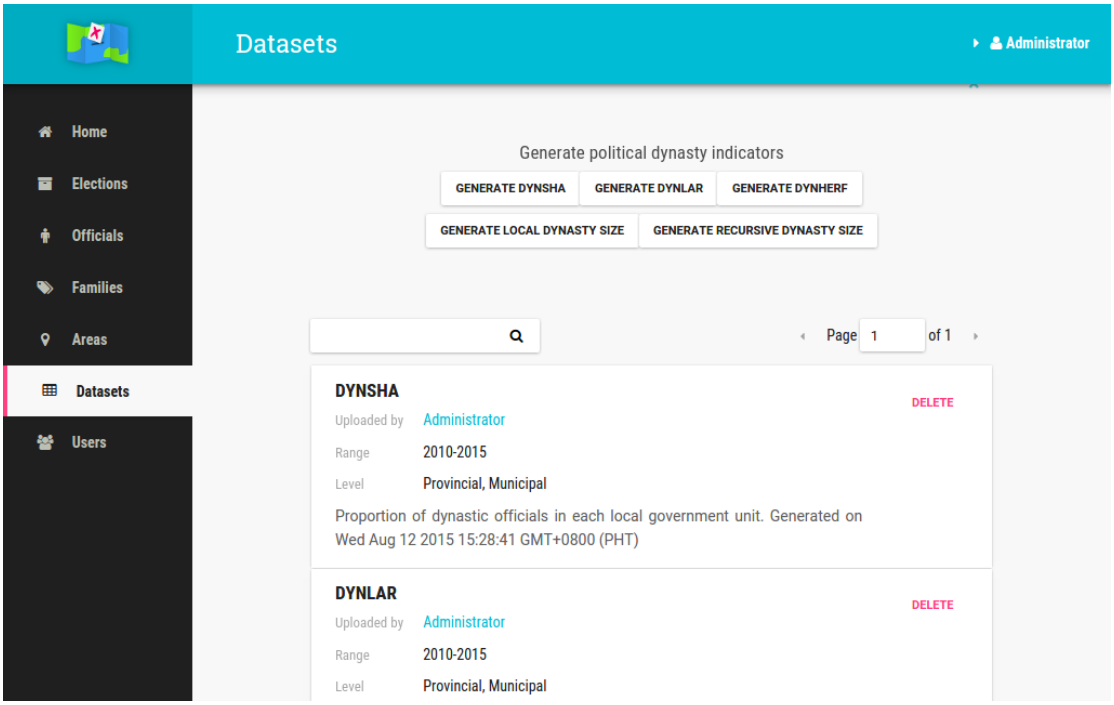


Figure 19: Datasets list page

Fig. 19 shows the datasets list page for managing datasets. User-uploaded datasets as well as system-generated datasets are listed in this page. Individual datasets can be deleted. The system provides five options to generate datasets: DYNSHA, DYNLAR, DYNHERF, LocalDynastySize, and RecursiveDynastySize, corresponding to the different political dynasty indicators. Choosing an option will cause the system to generate the chosen indicator variable as a new dataset.

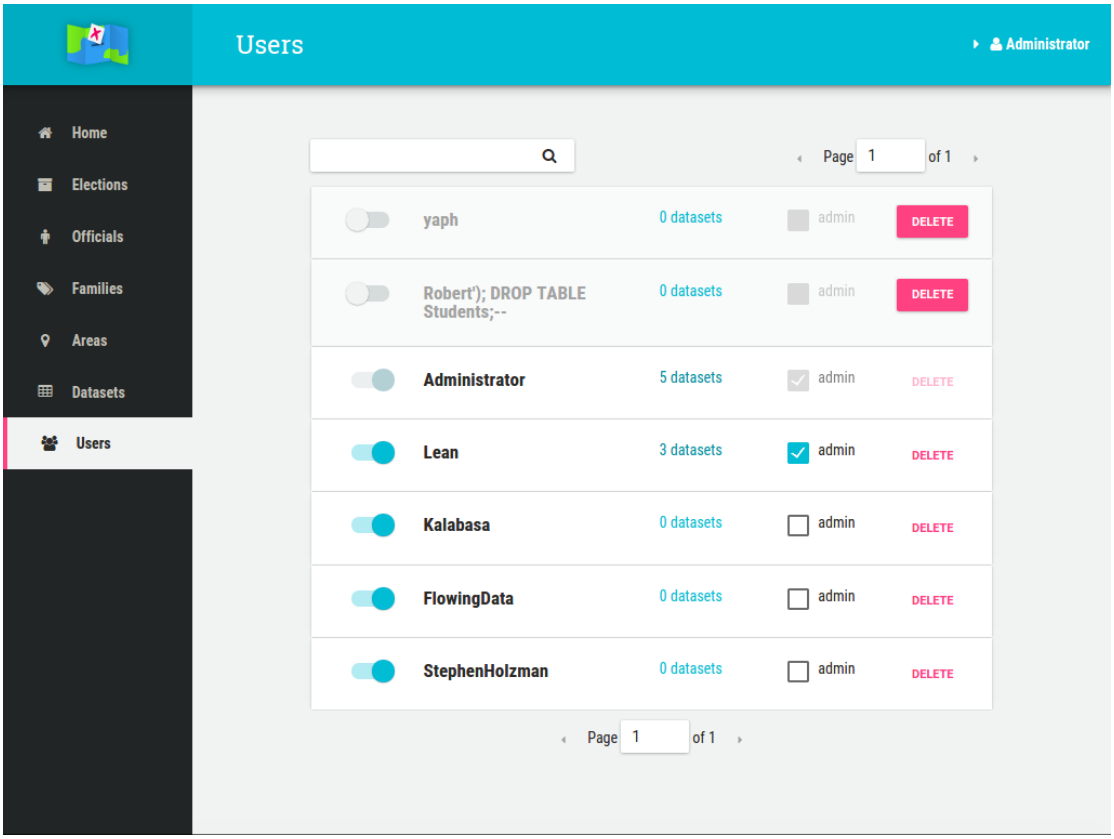


Figure 20: Users list page

Fig. 20 shows the users management page. Registered users of the system are listed in the page. The administrator can delete any user except itself. It can also change the type of any user except itself from a regular user to an administrator. It can approve or deactivate users to change their upload privileges. For each user there is a link to the user's list of datasets.

## C. Main interface

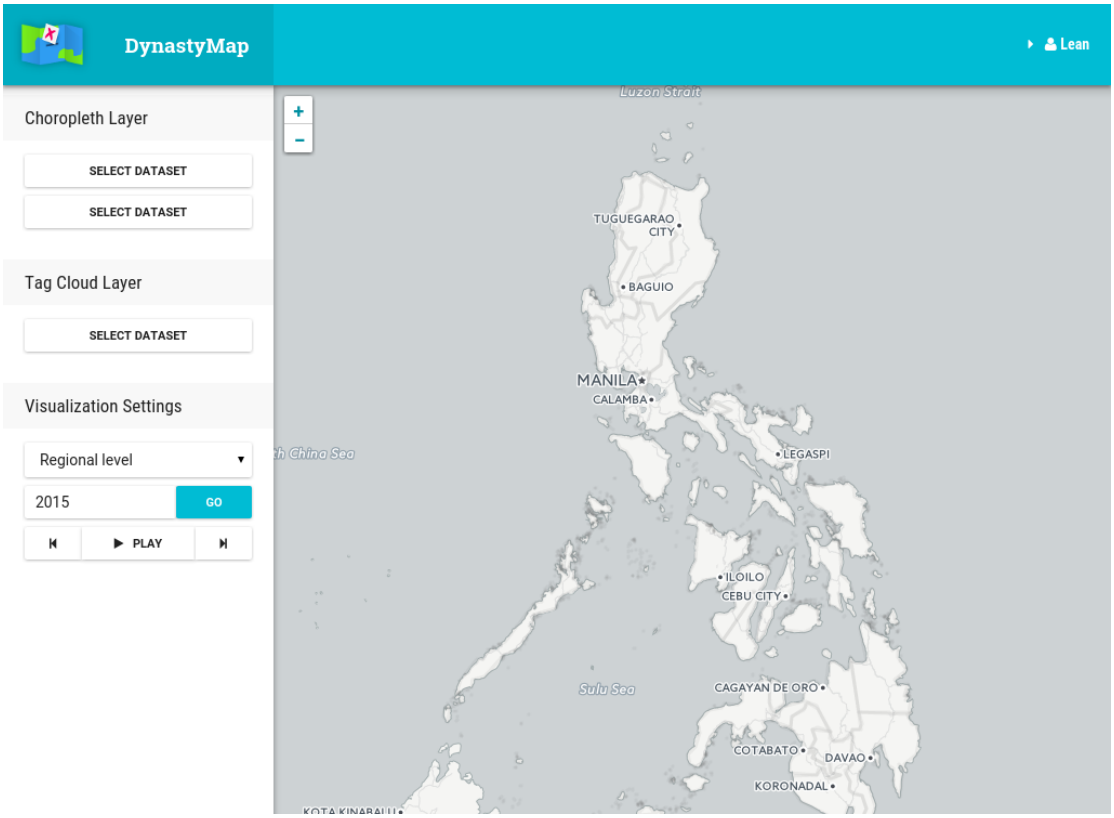


Figure 21: Main page in its initial state

Fig. 21 shows the main page of the application. On the left sidebar is a set of controls for controlling the map visualization, which is shown on the right side, on the major portion of the application. The top header bar shows the login status of the current user.

The left sidebar features three sections: Choropleth Layer, Tag Cloud Layer, and Visualization Settings. Each section controls different components of the map visualization.

The Visualization Settings section controls the level of detail of the visualization and the year to display in the visualization. It includes a Play button for the automatic changing of years through the range of allowed years.

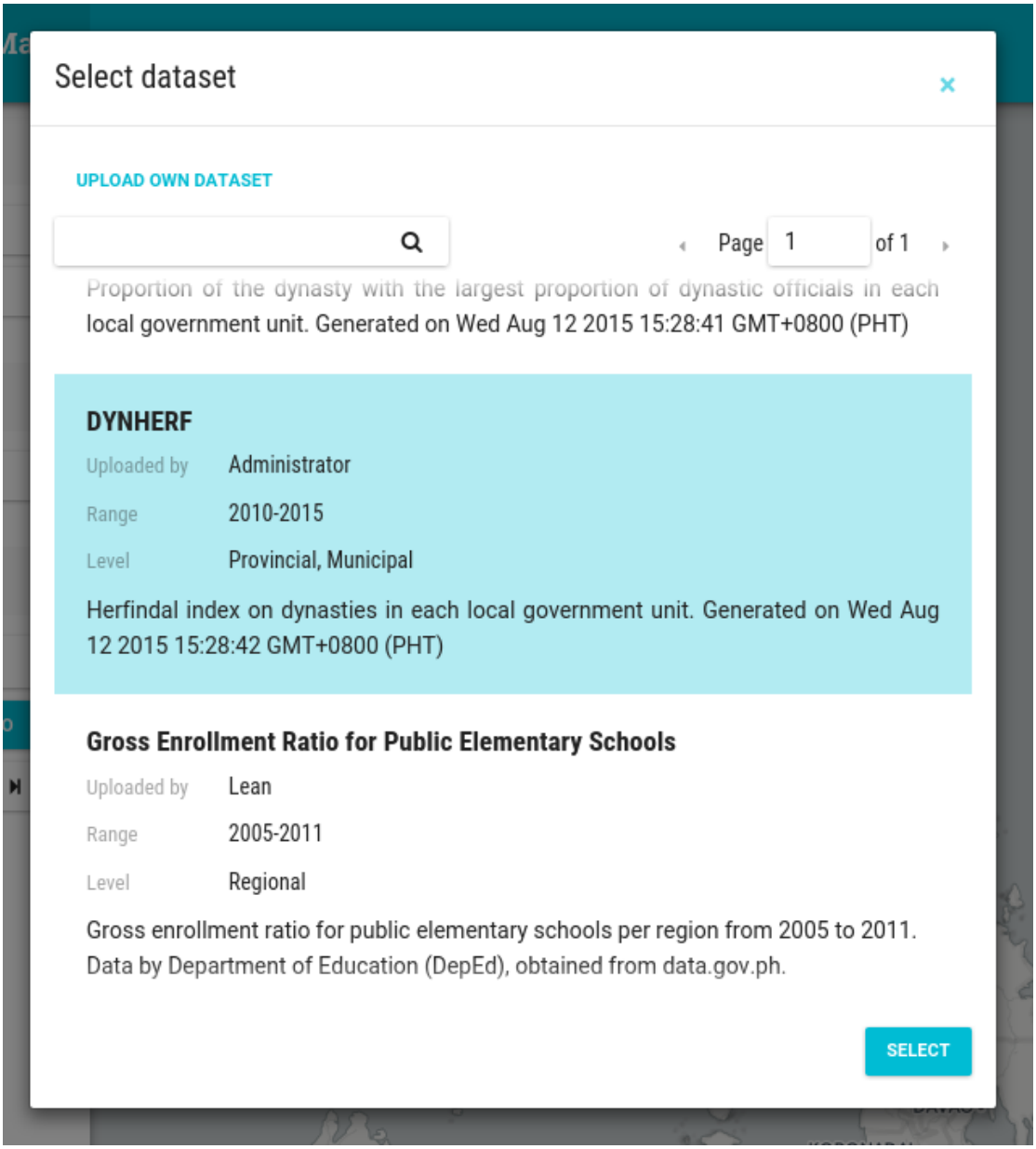


Figure 22: Dataset selection modal

When the user clicks one of the Select Dataset buttons in the sidebar, a dataset selection modal will be displayed as shown in Fig. 22. The dataset selection modal provides an interface for users to select the dataset they wish to visualize. There is also an option for users to upload their own dataset, which will lead them to the datasets upload page.

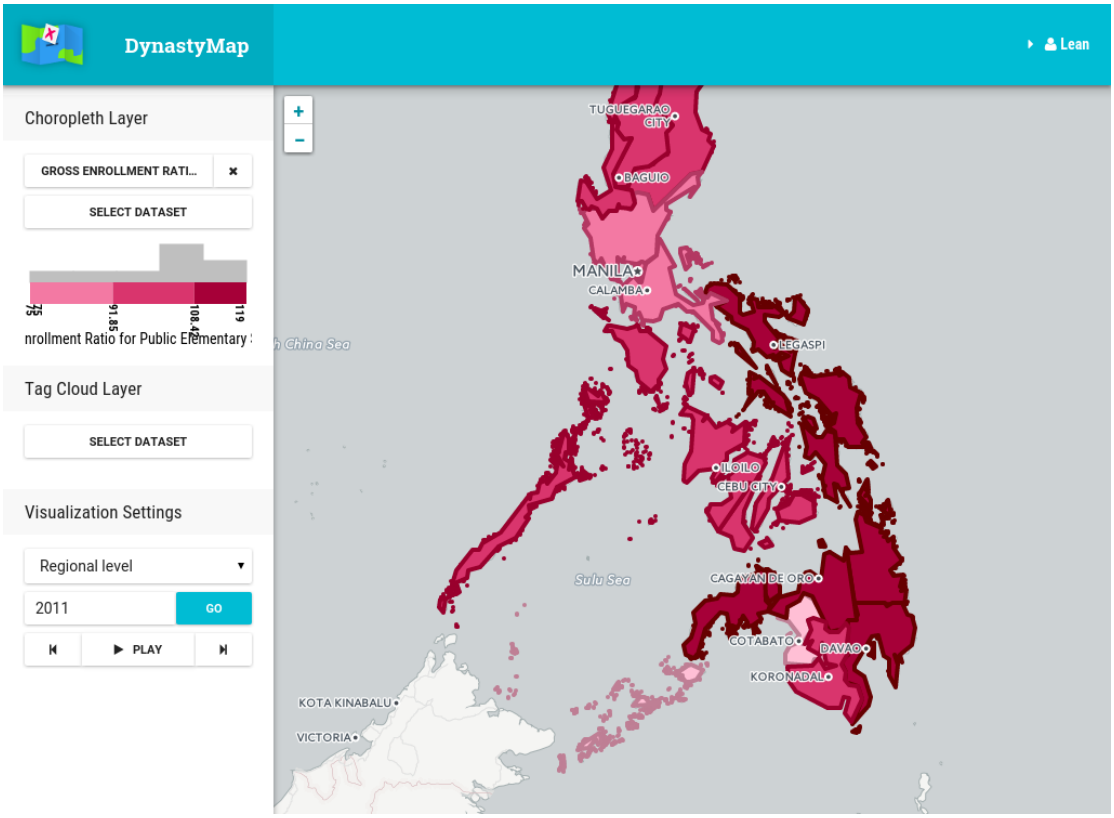


Figure 23: Univariate choropleth visualization

Fig. 23 shows the main interface after selecting a dataset for the choropleth layer. A choropleth layer will be visualized on the viewport when one or more datasets are selected for the choropleth layer. A histogram of the data for the current level and year is shown on the sidebar. The choropleth color legend is also displayed. The coloring scheme/classification is generated using the Jenks natural breaks optimization.

The variable represented by the choropleth map in Fig. 23 is the DYNHERF variable, which indicates the size of local dynasties and the amount of competition between them.

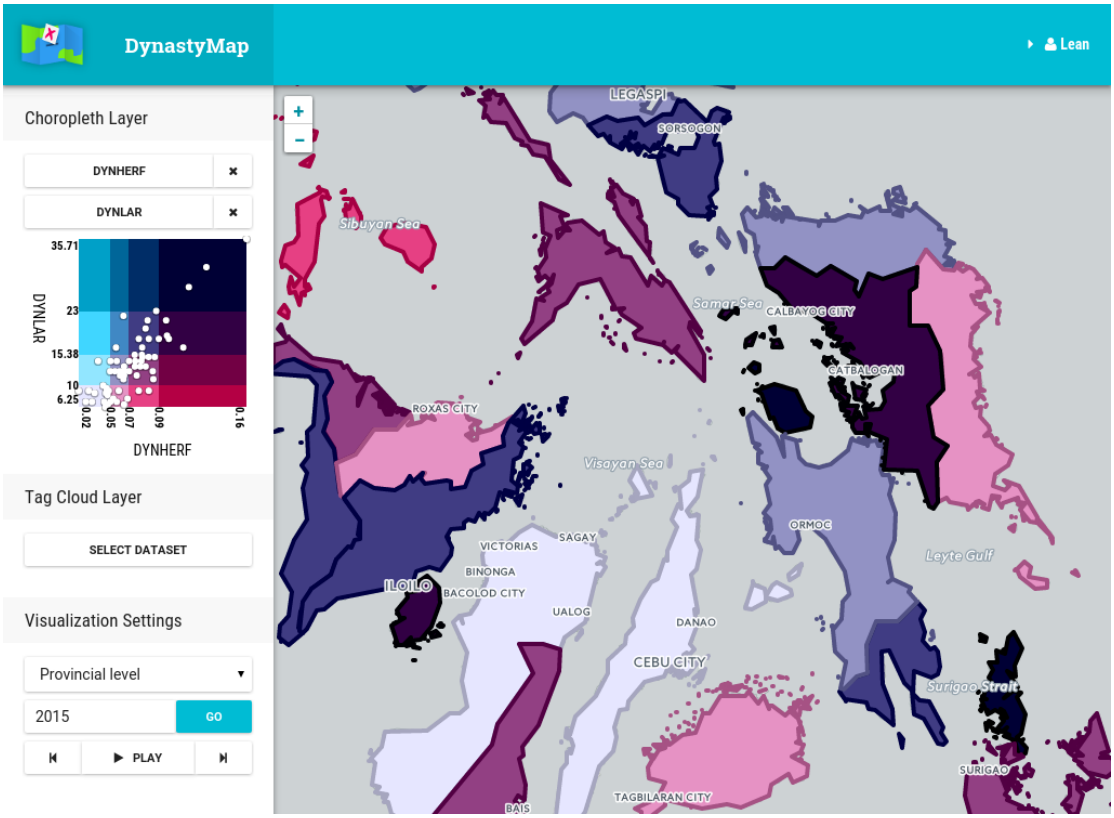


Figure 24: Bivariate choropleth visualization

When two datasets are selected for the choropleth layer, the visualization will become bivariate, as shown in Fig. 24. The choropleth legend becomes a two-dimensional legend, with a scatterplot of the data overlaid on top of the two-dimensional color legend. The color scheme is a combination of the two color schemes of the two datasets.

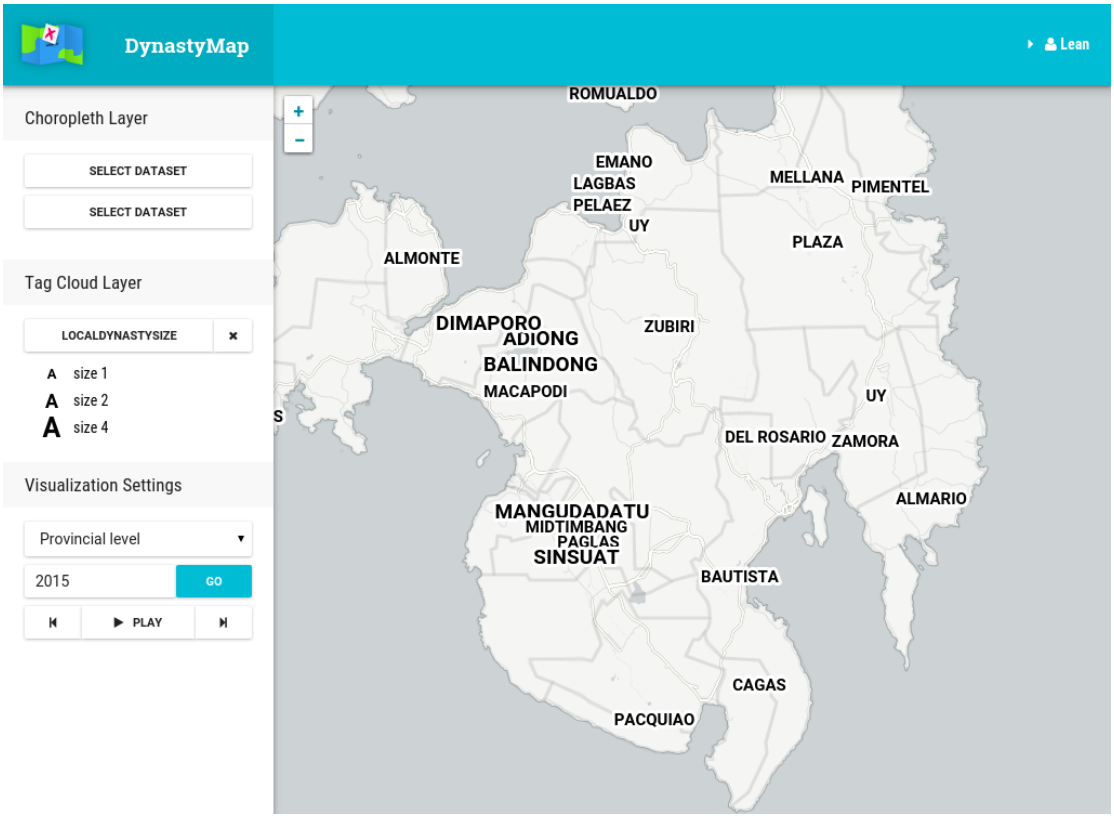


Figure 25: Tag map visualization

Fig. 25 shows the tag cloud layer of the visualization. The datasets that can be selected for the tag map layer are the political dynasty datasets LocalDynastySize and RecursiveDynastySize. A size legend is shown on the sidebar. A tag in the map corresponds to a local dynasty in one area. Each tag is positioned in the general location of the area it is associated with. The size of the tag is proportional to the size of the local dynasty represented by the tag. Small dynasties were not displayed to prevent crowding.

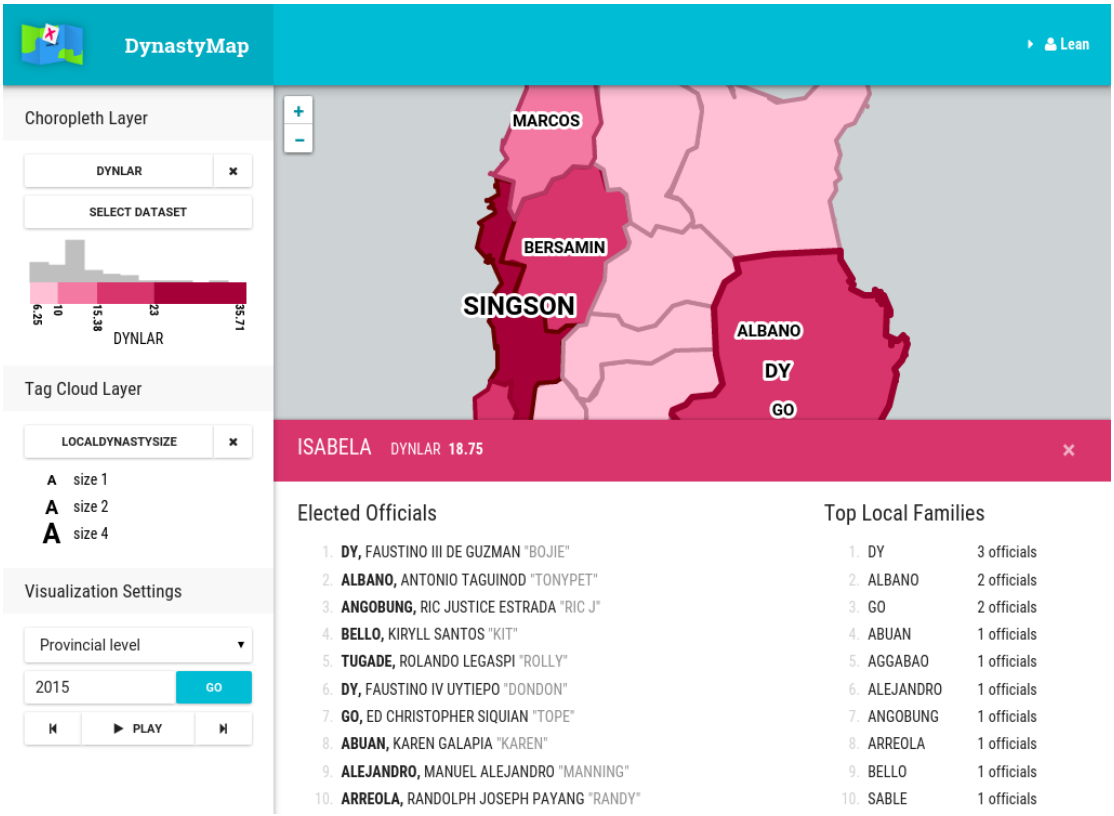


Figure 26: Main interface with all the layers active and the area details bar

Fig. 26 shows the main interface with all the layers activated. The bottom bar is displayed when the user clicks on any of the areas. It shows details of the selected area, such as the list of elected officials and the top local families in the area. It also shows the value of the current choropleth variable for that area.

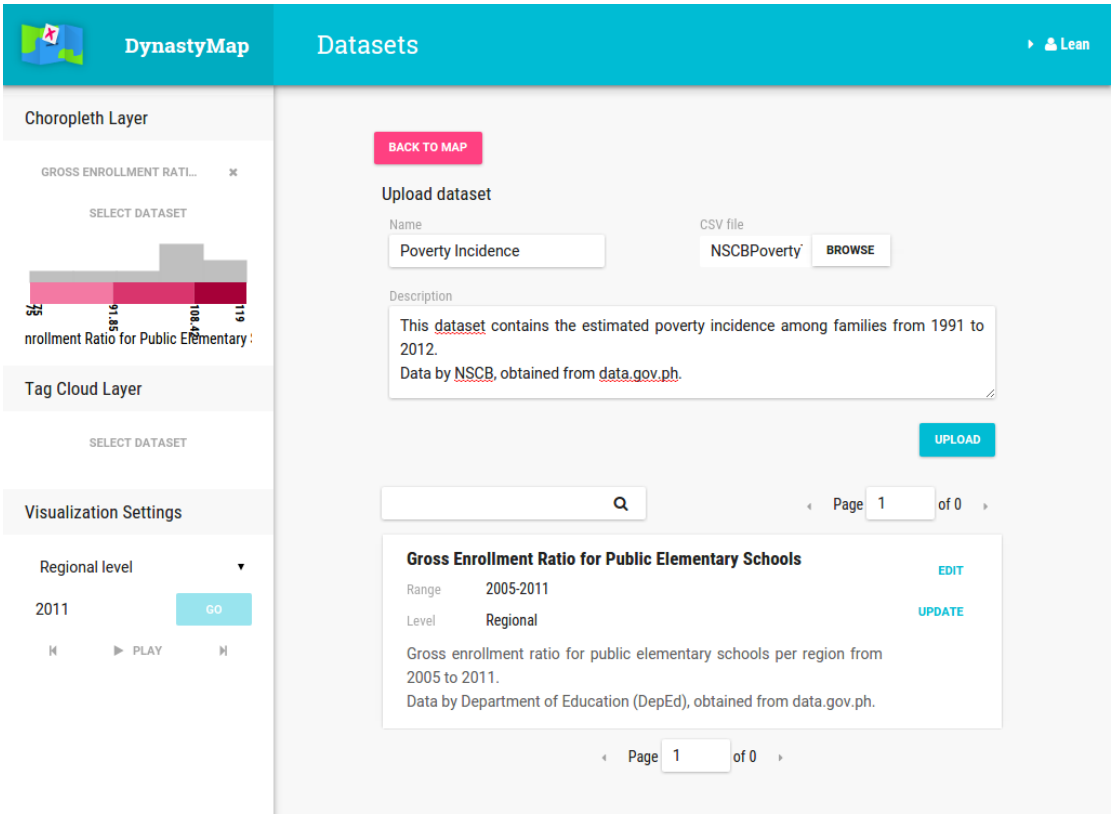


Figure 27: User datasets list page

When the user selects the "Upload Own Dataset" option from the dataset selection modal, the datasets page is displayed, as shown in Fig. 27. This page shows all of the datasets uploaded by the current user, and a form to create and upload new datasets. Each dataset can be edited and updated.

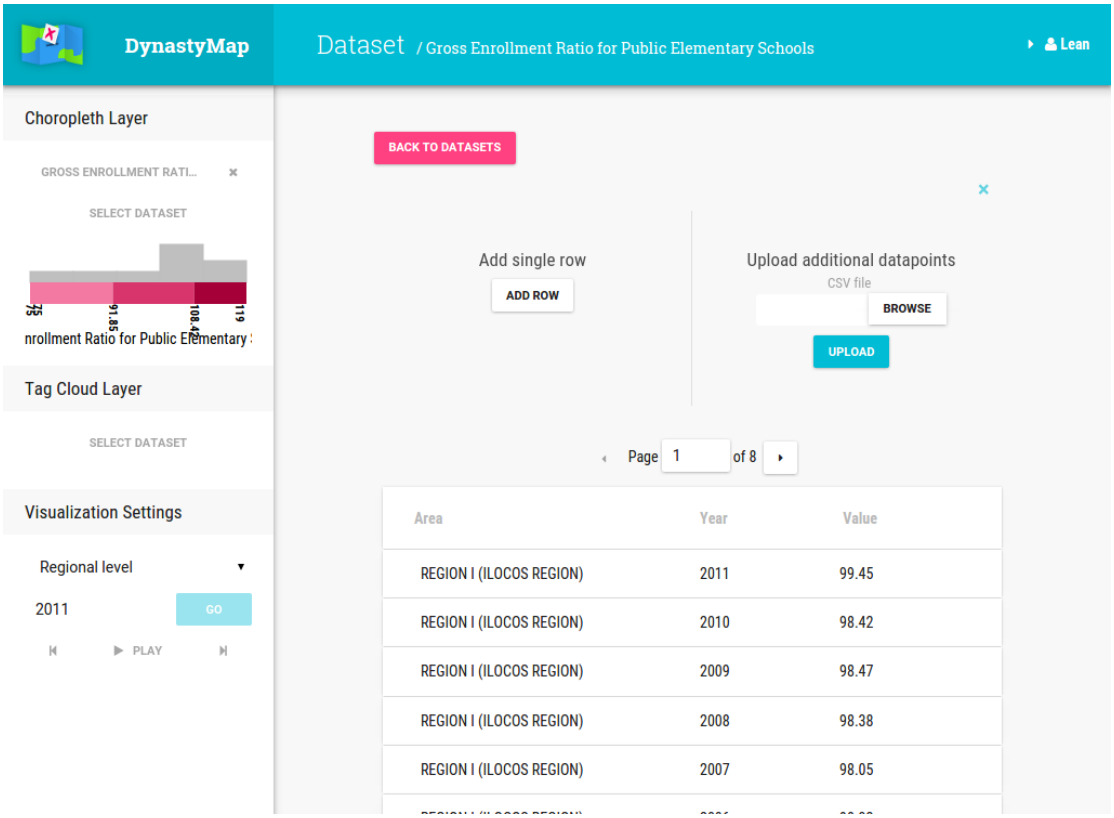


Figure 28: Dataset update page

When the user clicks on the dataset "Update" button, the dataset update page is displayed, as shown in Fig. 28. Here, users are able to edit individual datapoints and add more datapoints to the dataset.

# VI. Discussions

The DynastyMap system was proposed and developed as an interactive data visualization tool on political dynasties in the Philippines. The main features of the system are the computation of several indicators for political dynasty prevalence in local government units, and data visualization using choropleth maps for various datasets and tag maps for dynasty data. DynastyMap also includes administration tools for managing the data used by the system. Software libraries and frameworks were used to aid in development and to avoid reinventing the wheel.

Using the web application, users can view visualizations of various datasets available in the system. They can also upload their own datasets for visualization. Political dynasty datasets are also available by the system. The user can explore the visualization by zooming and panning around, changing the year or the level of detail, and by clicking areas to reveal details.

Styles and animations were created for a smoother and more functional experience using parts of the CSS frameworks PureCSS and Bootstrap. Google's material design were used as loose guidelines for the user interface style and patterns. The SCSS extension for CSS was used to organize styles and accelerate development.

The client software was built as a single-page application (SPA) with the goal of providing a fluid interface similar to a desktop or native application. The RequireJS library and the React framework were used to achieve this goal.

RequireJS is a library that allows modular JavaScript programming. The use this library shortened development time by allowing code organization through the use of modules or separate files. The library also allows on-demand loading of JavaScript modules, which is really useful in a single-page web application like this.

React is a JavaScript library for building user interfaces. React is component-based, allowing code reuse. An example of this are the components that display

the name of an official and the pagination controls, which are used in different parts of the application. React also features a one-way data flow. The displayed components are strictly a function of the underlying data. Whenever the data is changed, the components automatically change to reflect the data. React made easier the development of a dynamic web application such as this one.

The client application uses the MVC software architecture. It is common for web applications to use the MVC pattern. However, in this case, the server was not involved. The client application implements the MVC pattern. This was made easier with the use of Backbone.js, a JavaScript "back-end" framework that provides structure to web applications through the use of data models (as in MVC), data collections, and events, while synchronizing data to the server. Combining the Backbone framework with the React UI framework enabled the implementation of the MVC architecture of the application. Backbone handles data, while React renders them.

On top of this base structure lie the main features of the system, one of which is data visualization. All visualizations are generated by the client application. The mainly map-based visualizations include choropleth maps and tag maps. These are rendered as layers on top of a map of the country – the basemap. The Leaflet and the D3 JavaScript libraries were utilized to render the visualizations.

The Leaflet library provides interactive maps. It handles the parsing of the GeoJSON and the drawing of polygons on the map, which are then colored to create the choropleth layer. It also handles the drawing of the basemap. The basemaps are provided free-to-use by the CartoDB map service. CartoDB basemaps were chosen over other map services since these basemaps were specifically designed for data visualization. These basemaps are greyscale with low contrast, which is suitable for overlaying with data visualization.

The D3 library was used to render the tag map. D3 provides easy binding of data to graphics elements, which enabled easy manipulation of the tag map visualization. The tag map rendered by D3 was then added onto the Leaflet map

as a layer. D3 was also used to render the scatterplot, the histogram, and the legend.

Several other useful libraries were utilized, such as jenks, underscore, and jQuery, and some were created for the system as needed, such as validator.js and the number formatter mini-library.

The server, on the other hand, uses a smaller number of libraries: the Slim framework and the Medoo database library.

Medoo is a lightweight PHP database framework. Medoo was used to interface with the MySQL database. The library provides wrapper functions for common commands, such as select, insert, and update, and is flexible enough for the execution of complex commands.

On top of Medoo is the entity abstraction layer, where database rows are abstracted into PHP objects for easier manipulation and for consistency and code reuse in saving and loading of data. The entity system was developed from scratch with only Medoo as the helper library.

The Slim framework was used to make the API that the client uses to communicate with the server. The API follows a REST design, which tightly integrates with the client's Backbone library. Since the client already implements most of the MVC logic, the server was somewhat reduced to a thin server. Client requests are directly mapped to database requests, and results are directly returned to the clients, with minimal processing involved. The server essentially acts as a data server, with a few exceptions, specifically the data processing portion of the server.

PHP and Python were used for data processing in the server. PHP was used for generating the SQL queries that define the computation for the dynasty indicator variables. PHP was only used for the pre- and post-processing of the data.

Python was used for processing GeoJSON uploaded by the administrator. The GeoJSON is chopped up into tiles, similar to how the map is divided into tiles, following the same tiling conventions. This allows the client to only request the relevant parts of the GeoJSON geometry. The idea is to only load and show a

part of the geometry at once. As an example, the Philippine barangays GeoJSON file is about 60 megabytes. Loading this into the client requires a long time and a substantial amount of computing resource. By dividing the GeoJSON into tiles, the client can just request the necessary parts of the barangays GeoJSON. The Python script uses the Python GeoJSON library for parsing and writing in GeoJSON format.

Some server tools were created during the development of the system, primarily for testing purposes, and possibly for use by system administrators. One is a web scraper for election records, which parses the COMELEC website for election results and saves them to a CSV file. The COMELEC website has no public data API, so scraping was the method used to quickly retrieve election records.

Another tool is a generator of mappings of common names (e.g. Manila) to PSGC (e.g. 133900000). This script parses the official PSGC table from NSCB and generates a fuzzy mapping of common names to PSGC codes and then writes them to a JSON file or a PHP file that can be readily included in a server script. This is useful since only few datasets use the PSGC encoding for areas. The mapping generated from the March 2015 PSGC table from NSCB was used by the current server code to parse common names found in user-uploaded datasets.

The third tool is a GeoJSON PSGC adder, which just reads a GeoJSON and attaches appropriate PSGC codes to each area, using the JSON mapping generated by the previous tool. Since most of the available Philippine GeoJSON files online don't include the PSGC code of areas, this script will be useful for automatically adding the PSGC codes required by the system.

Lastly, a tool for automatically generating political dynasty associations based on the surnames of officials was created for testing purposes. This enabled the generation of large dynasty data without the need to research into the lives of each of the thousands of officials in the database.

DynastyMap can be compared with other web map visualization software, such as IndieMapper or the Philippine Geoportal map viewer. IndieMapper uses the

proprietary Flash technology, while DynastyMap uses standard modern web technologies, such as HTML, JS, CSS, and SVG. PH Geoportal also uses standard web technologies, however, user-uploaded datasets are not allowed. In DynastyMap (and in IndieMapper), users can upload their own datasets for visualization. Still, the Geoportal has a large amount of Philippine datasets from to choose from, due to the fact that it is maintained by a government agency. However, that data is not publicly available. The DynastyMap system, on the other hand, allows access to datasets, as well as all of the election records and the dynasty data, through the client-agnostic server REST API. Finally, the distinguishing feature of DynastyMap is that it has a focus on dynasty data in the Philippines, which is apparent from the dynasty tag map visualization and the dynasty indicator datasets it offers.

DynastyMap was developed in the hope that it would be a useful tool for studying data related to political dynasties in the Philippines. By comparing and visualizing relevant datasets, questions about political dynasties in the Philippines may be answered, or at least indirectly answered, since this system may be viewed as just a preliminary data analysis and exploration tool. With the Anti-Political Dynasty bill in the works, the system may even be useful in the future in determining whether that law was effective in banning or reducing dynasties, and whether some social problems die with them in the process.

# VII. Conclusions

In this paper, DynastyMap, a web-based map data visualization system on political dynasties in the Philippines was presented as a tool for studying data related to political dynasties. The choropleth map and the tag map were used as the data visualization method, with the choropleth showing various datasets and the tag map showing dynasty data. The DYNSHA, DYNLAR, and DYNHERF metrics adapted from literature were used for measuring dynasties. The LocalDynastySize and RecursiveDynastySize metrics were also created using the previous metrics as basis. The PSGC standard was used for encoding of areas and the GeoJSON format for the describing area geometry. The web application was developed using several major JavaScript frameworks to accelerate development, include useful features, and add interactivity. Using the visualizations with the right datasets, DynastyMap can be an effective tool for the study or presentation of political dynasty data in the Philippines.

# VIII. Recommendations

Having completed the objectives of the study, DynastyMap still was lacking in some aspects. In particular, the automation aspects, optimization, and visualization can be improved.

For automation, the system can be made to interface directly with the COMELEC database, so it can obtain election records without the need for an external entity to relay the data from one system to the other. Election records would then be updated every election and the system will be up-to-date automatically.

Another area for improvement in automation is the association of officials to families. Currently the association is manually done through the administrator interface, per official. However, with the growing number of officials, this is not feasible for a human to do. A crude solution would be to use the surnames of the officials as basis for the families, which was already implemented in a script tool that was developed for testing. However, this approach is not accurate enough. An automated way of accurately associating individuals to families would be a great improvement to the system.

Optimization areas include the dynasty indicator generation, particularly the Recursive Dynasty Size computation; the tag cloud rendering, especially when there is a large amount of tags; memory management in the client application, which handles large amounts of datasets; and overall optimizations to the system, especially in the handling of large data.

A specific issue to address is in the visualization of tag maps. The placement of tags exhibit a few problems, from overlapping tags to misplaced tags. An algorithm for the smart placement of tags within the area polygons would improve the quality of the visualizations. There are already algorithms for packing rectangles (or words) into arbitrary polygons. The challenge is to implement them in a dynamic and real-time manner for this system that handles a large amount of tags.

More types of visualization can be added, such as animated charts of political

families, showing various variables such as total members, total votes obtained, over time. Possible types of charts are bubble charts or simple line graphs.

Lastly, to address some of the security issues in the system, user registration and uploading could be restricted. Currently, users can upload any dataset they wish to upload. There is no limit to the amount of data they can store into the system. Although user accounts can be deactivated, this unlimited upload is undesirable. A possible solution is to limit the number of datasets per user, or amount of datapoints per user. Another is to store datasets temporarily, and delete them afterwards, for example, when the user logs out.

# IX. Bibliography


[1] The Center for People Emporwerment in Governance (CenPEG), "Political clans are more entrenched after mid-term polls," *Issue Analysis*, no. 2, 2013.

[2] R. Mendoza, E. B. Jr., V. Venida, and D. Yap, "Political dynasties and poverty: Resolving the "chicken or the egg" question," 2013.

[3] R. Mendoza, E. B. Jr., V. Venida, and D. Yap, "An empirical analysis of political dynasties in the 15th philippine congress," *Philippine Political Science Journal*, vol. 33, no. 2, 2012.

[4] J. D. Santos, "Political dynasties growing virtually unopposed." vote2013.verafiles.org/political-dynasties-growing-virtually-unopposed, 2013.

[5] S. Orford, R. Harris, and D. Dorling, "Geography: Information visualization in the social sciences: A state-of-the-art review," *Social Science Computer Review*, vol. 17, no. 3, pp. 289–304, 1999.

[6] C. Bajaj, ed., *DATA VISUALIZATION TECHNIQUES*, ch. Preface. John Wiley & Sons, Ltd, 1998.

[7] H. Rosling, "GapMinder Foundation." www.gapminder.org, 2009.

[8] A. M. F. Lagmay, "Disseminating near real-time hazards information and flood maps in the philippines through web-gis," *DOST-Project NOAH Open-File reports*, vol. 1, pp. 28–36, 2012.

[9] Y. Wei, S. K. Santhana-Vannan, and R. Cook, "Discover, visualize, and deliver geospatial data through ogc standards-based webgis system," in *2009 17th International Conference on Geoinformatics*, Institute of Electrical and Electronics Engineers (IEEE), Aug. 2009.

[10] R. L. Graham, S. K. Santhana Vannan, U. Dadi, R. C. Cook, S. K. Holladay, T. W. Beaty, and B. E. Wilson, "Ornl daac webgis: A web-based gis system for visualizing and distributing biogeochemical and ecological datasets," in *American Geophysical Union, Fall Meeting 2007*, American Geophysical Union, Dec. 2007.

[11] M. Theus, "Interactive data visualization using mondrian," *Journal of Statistical Software*, vol. 7, no. 11, 2002.

[12] A. Slingsby, J. Dykes, J. Wood, and K. Clarke, "Interactive tag maps and tag clouds for the multiscale exploration of large spatio-temporal datasets," pp. 497–504, Institute of Electrical and Electronics Engineers (IEEE), 2007.

[13] M. L. Y. Ghany, "Visualization of multivariate health data using self-organizing maps," 2012.

[14] A. M. MacEachren, M. S. Stryker, I. J. Turton, and S. Pezanowski, "Health geojunction: place-time-concept browsing of health publications," *International Journal of Health Geographics*, vol. 9, no. 23, 2010.

[15] K. A. S. Tolentino, "Disaster relief in laguna: A geographical information system through crowdsourcing on facebook," 2012.

[16] J. Vosecky, D. Jiang, and W. Ng, "Limosa: A system for geographic user interest analysis in twitter," in *16th International Conference on Extending Database Technology*, 2013.

[17] D. Hauger and M. Schedl, *Adaptive Multimedia Retrieval: Semantics, Context, and Adaptation*, ch. Exploring Geospatial Music Listening Patterns in Microblog Data, pp. 133–146. Springer International Publishing, 2014.

[18] A. Marcus, M. S. Bernstein, O. Badar, D. R. Karger, S. Madden, and R. C. Miller, "Twitinfo: aggregating and visualizing microblogs for event exploration," in *Annual conference on human factors in computing systems (CHI '11)*, pp. 227–236, 2011.

[19] J. E. Hernández-Ávila, M.-H. Rodríguez, R. Santos-Luna, V. Sánchez-Castañeda, S. Román-Pérez, V. H. Ríos-Salgado, and J. A. Salas-Sarmiento, "Nation-wide, web-based, geographic information system for the integrated surveillance and control of dengue fever in mexico," *PLoS ONE*, vol. 8, no. 8, 2013.

[20] T. V. Dimitrova, A. Tsois, and E. Camossi, "Visualization of container movements through a web-based geographical information system," in *2013 European Intelligence and Security Informatics Conference (EISIC)*, pp. 182–185, 2013.

[21] D.-Q. Nguyen and H. Schumann, "Taggram: Exploring geo-data on maps through a tag cloud-based visualization," *Information Visualisation*, vol. 4, pp. 322,328, 2010.

[22] Axis Maps LLC, "Indiemapper." www.indiemapper.com, 2010.

[23] political dynasty, *American Heritage® Dictionary of the English Language*. 5 ed., 2011.

[24] "16th Congress House Bill No. 837: An Act Defining Political Dynasty and Prohibiting the Establishment Thereof." www.congress.gov.ph/download/basic_16/HB00837.pdf, 2013.

[25] choropleth map, *McGraw-Hill Dictionary of Scientific & Technical Terms*. 6 ed., 2003.

[26] tag cloud, *Dictionary.com Unabridged*.

[27] "Republic Act No. 7160: An Act Providing For A Local Government Code." www.gov.ph/1991/10/10/republic-act-no-7160, 1991.

[28] "Republic Act No. 6734: An Act Providing For An Organic Act For The Autonomous Region In Muslim Mindanao." www.comelec.gov.ph/uploads/References/RelatedLaws/ElectionLaws/ARMM/RA6734.pdf, 1989.

[29] "16th Congress House Bill No. 4994: Bangsamoro Basic Law." www.gov.ph/2014/09/10/house-bill-no-4994-bbl, 2014.

[30] National Statistical Coordination Board (NSCB), "Philippine Standard Geographic Codes as of March 2015," Mar. 2015.

[31] "NSCB Resolution No. 6 of 1997." www.nscb.gov.ph/resolutions/1997/6.asp, 1997.

[32] "GIS Cookbook for LGUs." cookbook.hlurb.gov.ph/5-09-01structure-philippine-standard-geographic-code-psgc.

[33] K. Foote and M. Lynch, "Lecture notes in The Geographer's Craft Project." www.colorado.edu/geography/gcraft/notes/intro/intro.html, 2000.

[34] H. Butler, M. Daly, A. Doyle, S. Gillies, T. Schaub, and C. Schmidt, "The GeoJSON Format Specification." geojson.org/geojson-spec.html, 2008.

# X. Appendix

## A. Source Code

*.htaccess*

```
SetEnvIf Authorization "(.*)"
    HTTP_AUTHORIZATION=$1

php_value upload_max_filesize 64M
php_value post_max_size 66M
```

*.user.ini*

```
upload_max_filesize = 64M

post_max_size = 40M
```

*Makefile*

```
all: css/index.css css/app.css css/admin.css
    css/login.css js-build/
clean:
  rm css/app.css css/admin.css css/login.css
  rm -r js-build/

css/index.css: scss/index.scss scss/_*
  scss scss/index.scss css/index.css

css/app.css: scss/app.scss scss/_*
  scss scss/app.scss css/app.css

css/admin.css: scss/admin.scss scss/_*
  scss scss/admin.scss css/admin.css

css/login.css: scss/login.scss scss/_*
  scss scss/login.scss css/login.css

js-build/: $(shell find js-src/)
  mkdir -p js-build/
  cp -r -t js-build/ js-src/*
  jsx --harmony --extension jsx js-src/ js-
      build/
  find js-build/ -type f -name "*.js" -print0
      | xargs -0 sed -i 's/jsx!//g'
  find js-build/ -type f -name "*.jsx" -delete
```

*admin.html*

```
<!DOCTYPE html>
<html>
<head>
  <title>DynastyMap</title>

  <link rel="shortcut icon" type="image/png"
      href="favicon.png"/>

  <link rel="stylesheet" type="text/css" href
      ="css/pure-min.css" />
  <link rel="stylesheet" type="text/css" href
      ="css/bootstrap.min.css" />
  <link rel="stylesheet" type="text/css" href
      ="css/admin.css" />

  <script src="js-src/lib/require.js"></script
      >
  <script src="js-src/config.js"></script>
</head>
<body>
  <div id="main">
    <div id="header" class="header layout-row
        "></div>
    <div id="sidebar" class="sidebar layout-
        col scroll-y"></div>
    <div id="body" class="body layout-row
        layout-col scroll-y"></div>
  </div>
  <script src="js-build/admin.js"></script>
</body>
</html>
```

*api.php*

```
<?php
require 'php/init.include.php';

use \Dynavis\Database;
use \Dynavis\Core\Entity;
use \Dynavis\Core\NotFoundException;
use \Dynavis\Core\DataException;

use \Dynavis\Model\Official;
use \Dynavis\Model\Family;
use \Dynavis\Model\Party;
use \Dynavis\Model\Area;
use \Dynavis\Model\Elect;
use \Dynavis\Model\User;
use \Dynavis\Model\Dataset;
use \Dynavis\Model\Datapoint;
use \Dynavis\Model\TagDatapoint;
use \Dynavis\Model\Token;

use \Dynavis\DataProcessor;

\Slim\Route::setDefaultConditions([
  "id" => "\d+",
  "id_" => "\d+",
  "code" => "\d{8,9}",
  "level" => "region|province|municipality|
      barangay",
  "zoom" => "\d+",
  "x" => "\d+",
  "y" => "\d+",
]);

$app = new \Slim\Slim(["debug" => true]);

$auth_admin = authenticator(["roles" => ["
    admin"]]);

$auth_username = authenticator(["
    username_match" => true]);
$auth_token = authenticator(["token_match" =>
    true]);
$auth_username_or_admin = authenticator(["
    username_match" => true, "roles" => ["
    admin"]]);

//-----------------------------------------------------------

// GET requests
//-----------------------------------------------------------

$app->get("/officials", function () {
    generic_get_list("Official"); } )->name("
    officials");
$app->get("/officials/:id", "get_official");
$app->get("/officials/:id/families", "
    get_official_families");
$app->get("/families", function () {
    generic_get_list("Family"); } );
$app->get("/families/:id", "get_family");
$app->get("/families/:id/officials", "
    get_family_officials");
$app->get("/parties", function () {
    generic_get_list("Party"); } );
$app->get("/parties/:id", function ($id) {
    generic_get_item("Party", $id); });
$app->get("/parties/:id/elections", "
    get_party_elections");
$app->get("/areas", "get_areas")->name("areas
    ");
$app->get("/areas/id/:id", "get_area");
$app->get("/areas/:code", "code_to_id", "
    get_area");
```

```
$app->get("/areas/id/:id/elections", "
    get_area_elections");
$app->get("/areas/:code/elections", "
    code_to_id", "get_area_elections");
$app->get("/elections", function () {
    generic_get_list("Elect"); } );
$app->get("/elections/:id", function ($id) {
    generic_get_item("Elect", $id); } );
$app->get("/users", $auth_admin, function () {
     generic_get_list("User"); } );
$app->get("/users/:username",
    $auth_username_or_admin, "get_user");
$app->get("/users/:username/datasets", "
    get_user_datasets");
$app->get("/users/:username/datasets/:id", "
    get_user_dataset");
$app->get("/users/:username/datasets/:id/
    datapoints", "get_user_dataset_datapoints
    ");
$app->get("/users/:username/datasets/:id_/
    datapoints/:id", "
    get_user_dataset_datapoint");
$app->get("/datasets", "get_datasets");
$app->get("/tokens/:id", $auth_token, "
    get_token");
$app->get("/geojson/:level/:zoom/:x/:y", "
    get_geojson");


//-------------------------------------------------------

// POST requests
//-------------------------------------------------------



$app->post("/officials", $auth_admin, "
    post_official");
$app->post("/officials/:id/families",
    $auth_admin, "post_official_family");
$app->post("/families", $auth_admin, function
    () { generic_post_item("Family", "
    families"); } );
$app->post("/families/:id/officials",
    $auth_admin, "post_family_official");
$app->post("/parties", $auth_admin, function
    () { generic_post_item("Party", "parties
    "); } );
$app->post("/areas", $auth_admin, "post_area")
    ;
$app->post("/elections", $auth_admin, "
    post_election");
$app->post("/users", "post_user");
$app->post("/users/:username/datasets",
    $auth_username, "post_user_dataset");
$app->post("/users/:username/datasets/:id/
    datapoints", $auth_username, "
    post_user_dataset_datapoint");
$app->post("/tokens", "post_token");
$app->post("/geojson/:level", $auth_admin, "
    post_geojson");
$app->post("/generate-indicator", $auth_admin,
     "generate_indicator");


//-------------------------------------------------------

// PUT requests
//-------------------------------------------------------



$app->map("/officials/:id", $auth_admin,
    function ($id) { generic_put_item("
    Official", $id); } )->via("PUT", "PATCH")
    ;
$app->map("/families/:id", $auth_admin,
    function ($id) { generic_put_item("Family
    ", $id); } )->via("PUT", "PATCH");
$app->map("/parties/:id", $auth_admin,
    function ($id) { generic_put_item("Party
    ", $id); } )->via("PUT", "PATCH");
$app->map("/areas/id/:id", $auth_admin, "
    put_area")->via("PUT", "PATCH");
$app->map("/areas/:code", $auth_admin, "
    code_to_id", "put_area")->via("PUT", "
    PATCH");
$app->map("/elections/:id", $auth_admin,
    function ($id) { generic_put_item("Elect
    ", $id); } )->via("PUT", "PATCH");
$app->map("/users/:username",
    $auth_username_or_admin, "put_user")->via
    ("PUT", "PATCH");
$app->map("/users/:username/datasets/:id",
    $auth_username, "put_user_dataset")->via
    ("PUT", "PATCH");
$app->map("/users/:username/datasets/:id_/
    datapoints/:id", $auth_username, "
    put_user_dataset_datapoint")->via("PUT",
    "PATCH");
```

```
//-------------------------------------------------------

// DELETE requests
//-------------------------------------------------------


$app->delete("/officials/:id", $auth_admin,
    function ($id) { generic_delete_item("
    Official", $id); } );
$app->delete("/officials/:official_id/families
    /:id", $auth_admin, "
    delete_official_family");
$app->delete("/families", $auth_admin,
    function () { generic_delete_all("Family
    "); } );
$app->delete("/families/:id", $auth_admin,
    function ($id) { generic_delete_item("
    Family", $id); } );
$app->delete("/families/:official_id/officials
    /:id", $auth_admin, "
    delete_family_official");
$app->delete("/parties/:id", $auth_admin,
    function ($id) { generic_delete_item("
    Party", $id); } );
$app->delete("/areas", $auth_admin, function
    () { generic_delete_all("Area"); } );
$app->delete("/areas/id/:id", $auth_admin, "
    delete_area");
$app->delete("/areas/:code", "code_to_id",
    $auth_admin, "delete_area");
$app->delete("/elections", $auth_admin, "
    delete_all_elections");
$app->delete("/elections/:id", $auth_admin,
    function ($id) { generic_delete_item("
    Elect", $id); } );
$app->delete("/users/:username",
    $auth_username_or_admin, "delete_user");
$app->delete("/users/:username/datasets/:id",
    $auth_username_or_admin, "
    delete_user_dataset");
$app->delete("/users/:username/datasets/:id_/
    datapoints/:id", $auth_username, "
    delete_user_dataset_datapoint");
$app->delete("/tokens/:id", $auth_token,
    function ($id) { generic_delete_item("
    Token", $id); } );

$app->run();


//-------------------------------------------------------

// Function definitions
//-------------------------------------------------------


function defaults($params, $defaults){
  foreach ($params as $key => $value) {
    $defaults[$key] = $value;
  }
  return $defaults;
}

function code_to_id($route) {
  global $app;
  $code = $route->getParams()["code"];
  try {
    $area = Area::get_by_code((int) $code);
  }catch(NotFoundException $e) {
    $app->halt(404, $e->getMessage());
  }
  $app->is_id_from_code = true;
  $app->redirect($app->request->getRootUri() .
       preg_replace("/(?=\/)?\d{8,9}(?=\/)
      ?/", "id/" . $area->get_id(), $app->
      request->getResourceUri()));
}

function authenticator($options) {
  global $app;
  return function($route) use ($options, $app)
       {
    $auth = $app->request->headers->get("
        Authorization");

    if(!isset($auth)) {
      $app->halt(401);
    }

    $matches = array();
    preg_match('/Token token="(.*)"/', $auth,
        $matches);
    if(isset($matches[1])){
      $token_string = $matches[1];
    }else{
      $app->halt(401);
    }
```

```
    try {
      $token = \Dynavis\Model\Token::
          get_by_token($token_string);
    }catch(NotFoundException $e) {
      $app->halt(404);
    }
    if(!$token || !$token->valid()) {
      $app->halt(401);
    }

    $user = $token->get_user();
    $role = ["user", "admin"][$user->type];

    $allow = false;
    if(isset($options["roles"]) && in_array(
        $role, $options["roles"])) {
      $allow = true;
    }else if(isset($options["username_match"])
        && $route->getParam("username") ===
        $user->username) {
      $allow = true;
    }else if(isset($options["token_match"]) &&
        (int) $route->getParam("id") ===
        $token->get_id()) {
      $allow = true;
    }

    if(!$allow) {
      $app->halt(403);
    }

    $app->user_id = $user->get_id();
  };
}

function normalize_query($query_string) {
  return preg_split("/\W+/", $query_string,
      null, PREG_SPLIT_NO_EMPTY);
}


// Generic

function generic_get_list($class) {
  global $app;
  $class = "\\Dynavis\\Model\\" . $class;

  $params = defaults($app->request->get(), [
    "count" => 0,
    "start" => 0,
    "q" => null,
    "qnorm" => true,
  ]);

  $start = (int) $params["start"];
  $count = (int) $params["count"];
  if(!is_null($params["q"])) {
    $query = $params["qnorm"]
      ? normalize_query($params["q"])
      : [$params["q"]];
  }

  $result = isset($query)
    ? $class::query_items($count, $start,
        $query)
    : $class::list_items($count, $start);
  if(!$result) {
    $app->halt(400, "Invalid request
        parameters.");
  }
  $list = array_map(
    function ($data) use ($class) {
      return new $class($data, false);
    },
    $result["data"]
  );
  $total = $result["total"];

  $end = $start + count($list);
  if($end > $total) $end = $total;

  echo json_encode([
    "total" => $total,
    "start" => $start,
    "end" => $end,
    "data" => $list,
  ]);
}

function generic_get_item($class, $id) {
  global $app;
  $class = "\\Dynavis\\Model\\" . $class;
  try {
    $item = new $class((int) $id);
  }catch(NotFoundException $e) {
    $app->halt(404);
  }
  echo json_encode($item);
}

function generic_post_item($class, $name) {
  global $app;
  $class = "\\Dynavis\\Model\\" . $class;

  $data = json_decode($app->request->getBody()
      , TRUE);
  if(is_null($data)) {
    $app->halt(400, "Malformed data.");
  }

  $item = new $class();
  foreach ($class::FIELDS as $field) {
    if(!isset($data[$field])) {
      $app->halt(400, "Incomplete data. " .
          $field);
    }
    $item->$field = $data[$field];
  }
  try {
    $item->save();
  }catch(DataException $e) {
    $app->halt(400, "Invalid data. " . $e->
        getMessage());
  }

  $app->response->setStatus(201);
  echo json_encode($item);
}

function generic_put_item($class, $id) {
  global $app;
  $class = "\\Dynavis\\Model\\" . $class;

  $data = json_decode($app->request->getBody()
      , TRUE);
  if(is_null($data)) {
    $app->halt(400, "Malformed data.");
  }

  try {
    $item = new $class((int) $id);
  }catch(NotFoundException $e) {
    $app->halt(404);
  }
  foreach ($data as $key => $value) {
    if(!in_array($key, $class::FIELDS)) {
      $app->halt(400, "Invalid property. " .
          $key);
    }
    $item->$key = $value;
  }
  try {
    $item->save();
  }catch(DataException $e) {
    $app->halt(400, "Invalid data. " . $e->
        getMessage());
  }

  echo json_encode($item);
}

function generic_delete_item($class, $id) {
  global $app;
  $class = "\\Dynavis\\Model\\" . $class;
  try {
    $item = new $class((int) $id);
  }catch(NotFoundException $e) {
    $app->halt(404);
  }
  try {
    $item->delete();
  }catch(DataException $e) {
    $app->halt(400, $e->getMessage());
  }

  $app->response->setStatus(204);
}

function generic_delete_all($class) {
  global $app;
  $class = "\\Dynavis\\Model\\" . $class;
  try {
    $class::delete_all();
  }catch(DataException $e) {
    $app->halt(400, $e->getMessage());
  }
  $app->response->setStatus(204);
}


// Officials

function get_official($id) {
  global $app;
  try {
    $official = new Official((int) $id);
  }catch(NotFoundException $e) {
    $app->halt(404);
```

```
  }
  echo json_encode($official);
}

function get_official_families($id) {
  global $app;
  try {
    $official = new Official((int) $id);
  }catch(NotFoundException $e) {
    $app->halt(404);
  }
  $families = $official->get_families();

  echo json_encode([
    "total" => count($families),
    "data" => $families,
  ]);
}

function post_official() {
  global $app;
  $data = json_decode($app->request->getBody()
      , TRUE);
  if(is_null($data)) {
    $app->halt(400, "Malformed data.");
  }

  if(!isset($data["surname"], $data["name"]))
      {
    $app->halt(400, "Incomplete data.");
  }

  $official = new Official();
  $official->surname = $data["surname"];
  $official->name = $data["name"];
  $official->nickname = isset($data["nickname
      "]) ? $data["nickname"] : null;
  try {
    $official->save();
  }catch(DataException $e) {
    $app->halt(400, "Invalid data. " . $e->
        getMessage());
  }

  $app->response->setStatus(201);
  echo json_encode($official);
}

function post_official_family($id) {
  global $app;
  $data = json_decode($app->request->getBody()
      , TRUE);
  if(is_null($data)) {
    $app->halt(400, "Malformed data.");
  }

  try {
    $official = new Official((int) $id);
  }catch(NotFoundException $e) {
    $app->halt(404);
  }

  Database::get()->pdo->beginTransaction();

  if(isset($data["id"])) {
    try {
      $family = new Family((int) $data["id"]);
    }catch(NotFoundException $e) {
      Database::get()->pdo->rollback();
      $app->halt(400, "Invalid family ID.");
    }
  }else if(isset($data["name"])) {
    $family = new Family();
    $family->name = $data["name"];
    try{
      $family->save();
    }catch(DataException $e) {
      Database::get()->pdo->rollback();
      $app->halt(400, "Invalid family data. "
          . $e->getMessage());
    }
  }else{
    Database::get()->pdo->rollback();
    $app->halt(400, "Incomplete data.");
  }

  try {
    $family->add_member($official);
  }catch(DataException $e) {
    Database::get()->pdo->rollback();
    $app->halt(400, $e->getMessage());
  }

  Database::get()->pdo->commit();

  $app->response->setStatus(201);
  echo json_encode($family);
}

function delete_official_family($official_id,
    $family_id) {
  global $app;

  Database::get()->pdo->beginTransaction();

  try {
    $official = new Official((int)
        $official_id);
    $family = new Family((int) $family_id);
    $family->remove_member($official);
  }catch(NotFoundException $e) {
    Database::get()->pdo->rollback();
    $app->halt(404);
  }catch(DataException $e) {
    Database::get()->pdo->rollback();
    $app->halt(404);
  }

  Database::get()->pdo->commit();

  $app->response->setStatus(204);
}


// Families

function get_family($id) {
  global $app;
  try {
    $family = new Family((int) $id);
  }catch(NotFoundException $e) {
    $app->halt(404);
  }
  echo json_encode($family);
}

function get_family_officials($id) {
  global $app;

  $params = defaults($app->request->get(), [
    "year" => False,
  ]);

  try {
    $family = new Family((int) $id);
  }catch(NotFoundException $e) {
    $app->halt(404);
  }
  $members = $family->get_members($params["
      year"]);

  echo json_encode([
    "total" => count($members),
    "data" => $members,
  ]);
}

function post_family_official($id) {
  global $app;
  $data = json_decode($app->request->getBody()
      , TRUE);
  if(is_null($data)) {
    $app->halt(400, "Malformed data.");
  }

  try {
    $family = new Family((int) $id);
  }catch(NotFoundException $e) {
    $app->halt(404);
  }

  if(!isset($data["id"])) {
    $app->halt(400, "Incomplete data.");
  }

  try {
    $official = new Official((int) $data["id
        "]);
  }catch(NotFoundException $e) {
    $app->halt(400, "Invalid official ID.");
  }

  try {
    $family->add_member($official);
  }catch(DataException $e) {
    $app->halt(400, $e->getMessage());
  }

  $app->response->setStatus(201);
  echo json_encode($official);
}

function delete_family_official($family_id,
    $official_id) {
  return delete_official_family($official_id,
      $family_id);
}
```

```
// Parties

function get_party_elections($id) {
  global $app;
  try {
    $party = new Party((int) $id);
  }catch(NotFoundException $e) {
    $app->halt(404);
  }
  $elections = $party->get_elections();

  echo json_encode([
    "total" => count($elections),
    "data" => $elections,
  ]);
}


// Areas

function get_areas() {
  global $app;
  $params = defaults($app->request->get(), [
    "count" => 0,
    "start" => 0,
    "level" => null,
    "q" => null,
    "qnorm" => true,
  ]);

  $start = (int) $params["start"];
  $count = (int) $params["count"];

  if(isset($params["level"])) $level = $params
      ["level"];
  else $level = null;

  if(is_null($params["q"])) {
    $query = null;
  }else{
    $query = $params["qnorm"]
      ? normalize_query($params["q"])
      : [$params["q"]];
  }

  $result = Area::list_areas($count, $start,
      $level, $query);
  if(!$result) {
    $app->halt(400, "Invalid request
        parameters.");
  }
  $areas = array_map(
    function ($data) {
      return new Area($data, false);
    },
    $result["data"]
  );
  $total = $result["total"];

  $end = $start + count($areas);
  if($end > $total) $end = $total;

  echo json_encode([
    "total" => $total,
    "start" => $start,
    "end" => $end,
    "data" => $areas,
  ]);
}

function get_area($id) {
  global $app;
  try {
    $area = new Area((int) $id, !$app->
        is_id_from_code);
  }catch(NotFoundException $e) {
    $app->halt(404);
  }
  echo json_encode($area);
}

function get_area_elections($id) {
  global $app;
  try {
    $area = new Area((int) $id, !$app->
        is_id_from_code);
  }catch(NotFoundException $e) {
    $app->halt(404);
  }
  $elections = $area->get_elections();

  echo json_encode([
    "total" => count($elections),
    "data" => $elections,
  ]);
}

function put_area($id) {
  global $app;
  $data = json_decode($app->request->getBody()
      , TRUE);
  if(is_null($data)) {
    $app->halt(400, "Malformed data.");
  }

  $area = new Area((int) $id, !$app->
      is_id_from_code);
  if(!$area) {
    $app->halt(404);
  }

  if(isset($data["level"])) {
    $data["type"] = [
      "region" => 0,
      "province" => 1,
      "municipality" => 2,
      "barangay" => 3,
    ][$data["level"]];
    unset($data["level"]);
  }

  foreach ($data as $key => $value) {
    if(!in_array($key, Area::FIELDS)) {
      $app->halt(400, "Invalid property. " .
          $key);
    }
    $area->$key = $value;
  }
  try {
    $area->save();
  }catch(DataException $e) {
    $app->halt(400, "Invalid data. " . $e->
        getMessage());
  }

  echo json_encode($area);
}

function post_area() {
  global $app;

  if(isset($_FILES["file"])) {
    return post_areas_file($_FILES["file"]);
  }

  $data = json_decode($app->request->getBody()
      , TRUE);
  if(is_null($data)) {
    $app->halt(400, "Malformed data.");
  }

  if(!isset($data["code"], $data["name"],
      $data["level"])) {
    $app->halt(400, "Incomplete data.");
  }

  $area = new Area();
  $area->code = (int) $data["code"];
  $area->name = $data["name"];
  $area->type = [
    "region" => 0,
    "province" => 1,
    "municipality" => 2,
    "barangay" => 3,
  ][$data["level"]];
  try {
    $area->save();
  }catch(DataException $e) {
    $app->halt(400, "Invalid data. " . $e->
        getMessage());
  }

  $app->response->setStatus(201);
  echo json_encode($area);
}

function post_areas_file($file) {
  global $app;
  Database::get()->pdo->beginTransaction();
  try {
    Area::file($file);
  }catch(DataException $e) {
    Database::get()->pdo->rollBack();
    $app->halt(400, "Invalid file. " . $e->
        getMessage());
  }
  Database::get()->pdo->commit();
  $app->response->setStatus(201);
  $app->response->headers->set("Location",
      $app->urlFor("areas"));
}

function delete_area($id) {
  global $app;
  try {
    $area = new Area((int) $id);
  }catch(NotFoundException $e) {
```

```
    $app->halt(404);
  }
  try {
    $area->delete();
  }catch(DataException $e) {
    $app->halt(400, $e->getMessage());
  }

  $app->response->setStatus(204);
}


// Elections

function post_election() {
  global $app;

  if(isset($_FILES["file"])) {
    return post_elections_file($_FILES["file
        "]);
  }

  $data = json_decode($app->request->getBody()
      , TRUE);
  if(is_null($data)) {
    $app->halt(400, "Malformed data.");
  }


  if(!isset($data["official_id"], $data["year
      "], $data["year_end"], $data["area_code
      "])) {
    $app->halt(400, "Incomplete data.");
  }

  try {
    $official = new Official((int) $data["
        official_id"]);
  }catch(NotFoundException $e) {
    $app->halt(400, "Invalid official ID.");
  }
  try{
    $area = Area::get_by_code((int) $data["
        area_code"]);
  }catch(NotFoundException $e) {
    $app->halt(400, "Invalid area code.");
  }
  try{
    $party = isset($data["party_id"]) ? new
        Party((int) $data["party_id"]) : null
        ;
  }catch(NotFoundException $e) {
    $app->halt(400, "Invalid party ID.");
  }

  $elect = new Elect(null, [
    "official" => $official,
    "area" => $area,
    "party" => $party,
  ]);
  $elect->year = (int) $data["year"];
  $elect->year_end = (int) $data["year_end"];
  $elect->position = isset($data["position"])
      ? $data["position"] : null;
  $elect->votes = isset($data["votes"]) ? (int
      ) $data["votes"] : null;
  try {
    $elect->save();
  }catch(DataException $e) {
    $app->halt(400, "Invalid data. " . $e->
        getMessage());
  }

  $app->response->setStatus(201);
  echo json_encode($elect);
}

function post_elections_file($file) {
  global $app;
  Database::get()->pdo->beginTransaction();
  try {
    Elect::file($file);
  }catch(DataException $e) {
    Database::get()->pdo->rollBack();
    $app->halt(400, "Invalid file. " . $e->
        getMessage());
  }
  Database::get()->pdo->commit();
  $app->response->setStatus(201);
  $app->response->headers->set("Location",
      $app->urlFor("officials"));
}

function delete_all_elections() { // Deletes
    EVERYTHING! except areas,datasets,etc
  global $app;
  Elect::delete_all();
  Party::delete_all();
  Database::get()->query("delete from " .
      Family::TABLE_FAMILY_MEMBERSHIP);
  Official::delete_all();
  Family::delete_all();
  $app->response->setStatus(204);
}


// Users

function get_user($username) {
  global $app;
  try {
    $user = User::get_by_username($username);
  }catch(NotFoundException $e) {
    $app->halt(404, $e->getMessage());
  }
  echo json_encode($user);
}

function post_user() {
  global $app;
  $data = json_decode($app->request->getBody()
      , TRUE);
  if(is_null($data)) {
    $app->halt(400, "Malformed data.");
  }

  if(!isset($data["username"], $data["password
      "])) {
    $app->halt(400, "Incomplete data.");
  }

  $user = new User();
  $user->active = false;
  $user->username = $data["username"];
  $user->set_password($data["password"]);
  $user->type = 0;
  try {
    $user->save();
  }catch(DataException $e) {
    $app->halt(400, "Invalid data. " . $e->
        getMessage());
  }

  $app->response->setStatus(201);
  echo json_encode($user);
}

function put_user($username) {
  global $app;
  $data = json_decode($app->request->getBody()
      , TRUE);
  if(is_null($data)) {
    $app->halt(400, "Malformed data.");
  }

  try {
    $user = User::get_by_username($username);
  }catch(NotFoundException $e) {
    $app->halt(404, $e->getMessage());
  }

  if(array_key_exists("active", $data)) {
    $user->active = (bool) $data["active"];
  }

  if(array_key_exists("password", $data)) {
    if(!isset($data["old_password"])) {
      $app->halt(400, "No old_password
          parameter found. The current
          password is needed to change
          password.");
    }else if(!$user->check_password($data["
        old_password"])) {
      $app->halt(400, "Invalid old password.")
          ;
    }
    $user->set_password($data["password"]);
  }

  if(array_key_exists("role", $data)) {
    switch ($data["role"]) {
      case "user": $type = 0; break;
      case "admin": $type = 1; break;
      default: $app->halt(400, "Invalid role.
          " . $data["role"]);
    }
    $user->type = $type;
  }

  try {
    $user->save();
  }catch(DataException $e) {
    $app->halt(400, "Invalid data. " . $e->
        getMessage());
  }

  echo json_encode($user);
```

```
}

function delete_user($username) {
  global $app;
  try {
    $user = User::get_by_username($username);
  }catch(NotFoundException $e) {
    $app->halt(404, $e->getMessage());
  }

  try {
    $user->delete();
  }catch(DataException $e) {
    $app->halt(400, $e->getMessage());
  }

  $app->response->setStatus(204);
}


// Datasets

function get_datasets() {
  global $app;
  $params = defaults($app->request->get(), [
    "count" => 0,
    "start" => 0,
    "type" => null,
    "q" => null,
    "qnorm" => true,
  ]);

  $start = (int) $params["start"];
  $count = (int) $params["count"];

  if(isset($params["type"])) $type = $params["
      type"];
  else $type = null;

  if(is_null($params["q"])) {
    $query = null;
  }else{
    $query = $params["qnorm"]
      ? normalize_query($params["q"])
      : [$params["q"]];
  }

  $result = Dataset::list_datasets($count,
      $start, $type, $query);
  if(!$result) {
    $app->halt(400, "Invalid request
        parameters.");
  }
  $areas = array_map(
    function ($data) {
      return new Dataset($data, false);
    },
    $result["data"]
  );
  $total = $result["total"];

  $end = $start + count($areas);
  if($end > $total) $end = $total;

  echo json_encode([
    "total" => $total,
    "start" => $start,
    "end" => $end,
    "data" => $areas,
  ]);
}

function get_user_datasets($username) {
  global $app;
  $params = defaults($app->request->get(), [
    "count" => 0,
    "start" => 0,
    "type" => null,
  ]);

  $start = (int) $params["start"];
  $count = (int) $params["count"];

  try {
    $user = User::get_by_username($username);
  }catch(NotFoundException $e) {
    $app->halt(404, $e->getMessage());
  }

  $type = $params["type"];
  $datasets = $user->get_datasets($count,
      $start, $type);
  $total = $user->count_datasets($type);

  $end = $start + count($datasets);
  if($end > $total) $end = $total;

  echo json_encode([
    "total" => $total,
    "start" => $start,
    "end" => $end,
    "data" => $datasets,
  ]);
}

function get_user_dataset($username,
    $dataset_id) {
  global $app;
  try {
    $user = User::get_by_username($username);
    $dataset = new Dataset((int) $dataset_id);
  }catch(NotFoundException $e) {
    $app->halt(404, $e->getMessage());
  }

  if($dataset->user_id != $user->get_id()) {
    $app->halt(404);
  }

  echo json_encode($dataset);
}

function get_user_dataset_datapoint($username,
     $dataset_id, $datapoint_id) {
  global $app;
  try {
    $user = User::get_by_username($username);
    $dataset = new Dataset((int) $dataset_id);
  }catch(NotFoundException $e) {
    $app->halt(404, $e->getMessage());
  }

  if($dataset->user_id != $user->get_id()) {
    $app->halt(404);
  }
  try {
    $datapoint = new Datapoint((int)
        $datapoint_id);
  }catch(NotFoundException $e) {
    $app->halt(404);
  }

  if($datapoints->dataset_id != $dataset->
      get_id()) {
    $app->halt(404);
  }

  echo json_encode($datapoint);
}

function get_user_dataset_datapoints($username
    , $dataset_id) {
  global $app;
  $params = defaults($app->request->get(), [
    "count" => 0,
    "start" => 0,
    "type" => null,
    "year" => null,
  ]);

  $start = (int) $params["start"];
  $count = (int) $params["count"];
  $year = is_null($params["year"]) ? null : (
      int) $params["year"];

  try {
    $user = User::get_by_username($username);
    $dataset = new Dataset((int) $dataset_id);
  }catch(NotFoundException $e) {
    $app->halt(404, $e->getMessage());
  }

  if($dataset->user_id != $user->get_id()) {
    $app->halt(404);
  }

  echo json_encode($dataset->get_points($count
      , $start, $year));
}

function post_user_dataset($username) {
  global $app;
  $data = json_decode($app->request->getBody()
      , TRUE);
  if(is_null($data)) {
    $app->halt(400, "Malformed data.");
  }

  if(!isset($data["name"], $data["description
      "], $data["type"])) {
    $app->halt(400, "Incomplete data.");
  }

  try {
    $user = User::get_by_username($username);
  }catch(NotFoundException $e) {
    $app->halt(404, $e->getMessage());
  }
```

```
  if(!$user->active) {
    $app->halt(403, "Deactivated users are not
        allowed to upload data.");
  }

  $total_data = Database::get()->count(Dataset
      ::TABLE, [
    "[><]" . Datapoint::TABLE => [Dataset::
        PRIMARY_KEY => "dataset_id"],
    "[><]" . TagDatapoint::TABLE => [Dataset::
        PRIMARY_KEY => "dataset_id"],
  ], "*", [
    Dataset::TABLE . ".user_id" => $user->
        get_id()
  ]);
  if($total_data > 20) {
    $app->halt(403, "Data limit reached.");
  }

  $dataset = new Dataset(null, ["user" =>
      $user]);
  $dataset->name = $data["name"];
  $dataset->type = [
    "area" => 0,
    "tag" => 1,
  ][$data["type"]];
  $dataset->description = $data["description
      "];
  try {
    $dataset->save();
  }catch(DataException $e) {
    $app->halt(400, "Invalid data. " . $e->
        getMessage());
  }

  $app->response->setStatus(201);
  echo json_encode($dataset);
}

function post_user_dataset_datapoint($username
    , $dataset_id) {
  global $app;

  if(isset($_FILES["file"])) {
    return post_dataset_file($username,
        $dataset_id, $_FILES["file"]);
  }

  $data = json_decode($app->request->getBody()
      , TRUE);
  if(is_null($data)) {
    $app->halt(400, "Malformed data.");
  }

  try {
    $user = User::get_by_username($username);
    $dataset = new Dataset((int) $dataset_id);
  }catch(NotFoundException $e) {
    $app->halt(404, $e->getMessage());
  }

  if($dataset->user_id != $user->get_id()) {
    $app->halt(404);
  }

  if(!$user->active) {
    $app->halt(403, "Deactivated users are not
        allowed to upload data.");
  }

  $total_data = Database::get()->count(Dataset
      ::TABLE, [
    "[>]" . Datapoint::TABLE => [Dataset::
        PRIMARY_KEY => "dataset_id"],
    "[>]" . TagDatapoint::TABLE => [Dataset::
        PRIMARY_KEY => "dataset_id"],
  ], "*", [
    Dataset::TABLE . ".user_id" => $user->
        get_id()
  ]);
  if($total_data > 20) {
    $app->halt(403, "Data limit reached.");
  }

  if(!isset($data["year"], $data["area_code"],
      $data["value"])) {
    $app->halt(400, "Incomplete data.");
  }

  try {
    $area = Area::get_by_code((int) $data["
        area_code"]);
  }catch(NotFoundException $e) {
    $app->halt(400, "Invalid area code.");
  }

  $datapoint = new Datapoint(null, [
    "dataset" => $dataset,
    "area" => $area,
  ]);
  $datapoint->year = $data["year"];
  $datapoint->value = $data["value"];
  try {
    $datapoint->save();
  }catch(DataException $e) {
    $app->halt(400, "Invalid data. " . $e->
        getMessage());
  }

  $app->response->setStatus(201);
  echo json_encode($datapoint);
}

function post_dataset_file($username,
    $dataset_id, $file) {
  global $app;

  try {
    $user = User::get_by_username($username);
    $dataset = new Dataset((int) $dataset_id);
  }catch(NotFoundException $e) {
    $app->halt(404, $e->getMessage());
  }

  Database::get()->pdo->beginTransaction();
  try {
    $dataset->file($_FILES["file"]);
  }catch(DataException $e) {
    Database::get()->pdo->rollBack();
    $app->halt(400, "Invalid file. " . $e->
        getMessage());
  }
  Database::get()->pdo->commit();

  $app->response->setStatus(201);
}

function put_user_dataset($username,
    $dataset_id) {
  global $app;
  $data = json_decode($app->request->getBody()
      , TRUE);
  if(is_null($data)) {
    $app->halt(400, "Malformed data.");
  }

  try {
    $user = User::get_by_username($username);
    $dataset = new Dataset((int) $dataset_id);
  }catch(NotFoundException $e) {
    $app->halt(404, $e->getMessage());
  }

  if($dataset->user_id != $user->get_id()) {
    $app->halt(404);
  }

  foreach ($data as $key => $value) {
    if(!in_array($key, Dataset::FIELDS)) {
      $app->halt(400, "Invalid property. " .
          $key);
    }
    $dataset->$key = $value;
  }
  try {
    $dataset->save();
  }catch(DataException $e) {
    $app->halt(400, "Invalid data. " . $e->
        getMessage());
  }

  echo json_encode($dataset);
}

function put_user_dataset_datapoint($username,
    $dataset_id, $datapoint_id) {
  global $app;
  $data = json_decode($app->request->getBody()
      , TRUE);
  if(is_null($data)) {
    $app->halt(400, "Malformed data.");
  }

  try{
    $user = User::get_by_username($username);
    $dataset = new Dataset((int) $dataset_id);
    $datapoint = new Datapoint((int)
        $datapoint_id);
    if($dataset->user_id != $user->get_id() ||
        $datapoint->dataset_id !=
        $dataset_id) {
      $app->halt(404);
    }
  }catch(NotFoundException $e) {
    $app->halt(404, $e->getMessage());
  }
```

```
  foreach ($data as $key => $value) {
    if(!in_array($key, Datapoint::FIELDS)) {
      $app->halt(400, "Invalid property. " .
          $key);
    }
    $datapoint->$key = $value;
  }
  try {
    $datapoint->save();
  }catch(DataException $e) {
    $app->halt(400, "Invalid data. " . $e->
        getMessage());
  }

  echo json_encode($datapoint);
}

function delete_user_dataset($username,
    $dataset_id) {
  global $app;
  try {
    $user = User::get_by_username($username);
    $dataset = new Dataset((int) $dataset_id);
  }catch(NotFoundException $e) {
    $app->halt(404, $e->getMessage());
  }

  if($dataset->user_id != $user->get_id()) {
     $app->halt(404);
  }

  try {
    $dataset->delete();
  }catch(DataException $e) {
    $app->halt(400, "Invalid data. " . $e->
        getMessage());
  }

  $app->response->setStatus(204);
}

function delete_user_dataset_datapoint(
    $username, $dataset_id, $datapoint_id) {
  global $app;

  try{
    $user = User::get_by_username($username);
    $dataset = new Dataset((int) $dataset_id);
    $datapoint = new Datapoint((int)
        $datapoint_id);
    if($dataset->user_id != $user->get_id() ||
        $datapoint->dataset_id !=
        $dataset_id) {
      $app->halt(404);
    }
  }catch(NotFoundException $e) {
    $app->halt(404, $e->getMessage());
  }

  try {
    $datapoint->delete();
  }catch(DataException $e) {
    $app->halt(400, $e->getMessage());
  }

  $app->response->setStatus(204);
}


// Tokens

function get_token($id) {
  global $app;
  try {
    $token = new Token((int) $id);
  }catch(NotFoundException $e) {
    $app->halt(404);
  }
  $token->refresh();

  Token::cleanup();

  echo json_encode($token);
}

function post_token() {
  global $app;
  $data = json_decode($app->request->getBody()
      , TRUE);
  if(is_null($data)) {
    $app->halt(400, "Malformed data.");
  }

  if(!isset($data["username"], $data["password
      "])) {
    $app->halt(400, "Incomplete data.");
  }

  try{
    $user = User::get_by_username($data["
        username"]);
  }catch(NotFoundException $e) {
    $app->halt(401, "Invalid username or
        password.");
  }
  if(!$user->check_password($data["password"])
      ) {
    $app->halt(401, "Invalid username or
        password.");
  }

  $token = new Token(null, ["user" => $user]);
  $token->save();

  Token::cleanup();

  $app->response->setStatus(201);
  echo json_encode($token);
}


// GeoJSON

function get_geojson($level, $zoom, $x, $y) {
  global $app;

  $target_zoom = [ // These zoom levels must
      match with the client
    "region" => 0,
    "province" => 8,
    "municipality" => 10,
    "barangay" => 12,
  ][$level];

  $s = pow(2, $target_zoom - $zoom);
  $x = floor((int)$x * $s);
  $y = floor((int)$y * $s);

  $url = dirname($app->request->getRootUri())
      . "/data/$level/$x/$y.json";
  $path = __DIR__ . "/data/$level/$x/$y.json";

  if(file_exists($path)) {
    $app->response->setStatus(302);
    $app->response->headers->set("Location",
        $url);
  }else{
    echo "null";
  }
}

function post_geojson($level) {
  global $app;

  if(!isset($_FILES["file"])) {
    $app->halt(400, "No file uploaded.");
  }

  $file = $_FILES["file"];
  $error = $file["error"];
  if($error != UPLOAD_ERR_OK) {
    switch ($error) {
      case UPLOAD_ERR_INI_SIZE:
        $message = "The uploaded file exceeds
            the upload_max_filesize directive
            in php.ini";
        break;
      case UPLOAD_ERR_FORM_SIZE:
        $message = "The uploaded file exceeds
            the MAX_FILE_SIZE directive that
            was specified in the HTML form";
        break;
      case UPLOAD_ERR_PARTIAL:
        $message = "The uploaded file was only
            partially uploaded";
        break;
      case UPLOAD_ERR_NO_FILE:
        $message = "No file was uploaded";
        break;
      case UPLOAD_ERR_NO_TMP_DIR:
        $message = "Missing a temporary folder
            ";
        break;
      case UPLOAD_ERR_CANT_WRITE:
        $message = "Failed to write file to
            disk";
        break;
      case UPLOAD_ERR_EXTENSION:
        $message = "File upload stopped by
            extension";
        break;

      default:
        $message = "Unknown upload error";
        break;
    }
    throw new \RuntimeException("File upload
        error. " . $message);
```

```
  }

  if(!is_uploaded_file($file["tmp_name"])) {
    throw new \RuntimeException("Invalid file
        .");
  }

  $size = $file["size"];
  if($size == 0) {
    throw new DataException("No file or empty
        file was uploaded.");
  }

  $dir = "./data";
  if(!is_dir($dir)) {
    if(!mkdir($dir, 0775)) {
      throw new \RuntimeException("Cannot
          create data directory.");
    }
  }

  $dest = "$dir/$level.json";
  if(!copy($file["tmp_name"], $dest)) {
    throw new \RuntimeException("Cannot save
        file!");
  }

  exec(escapeshellcmd("./scripts/
      process_geojson.py " . escapeshellarg(
      $dest) . " " . escapeshellarg($level) .
       " &"));

  $app->response->setStatus(204);
}


// Generate indicator

function generate_indicator() {
  global $app;

  $data = json_decode($app->request->getBody()
      , TRUE);
  if(is_null($data)) {
    $app->halt(400, "Malformed data.");
  }

  if(!isset($data["username"], $data["
      indicator"], $data["description"])) {
    $app->halt(400, "Incomplete data.");
  }

  try {
    $user = User::get_by_username($data["
        username"]);
  }catch(NotFoundException $e) {
    $app->halt(400, $e->getMessage());
  }
  $indicator = $data["indicator"];
  $description = $data["description"];

  try{
    $dataset = DataProcessor::
        generate_indicator($indicator,
        $description, $user);
  }catch(DataException $e) {
    $app->halt(403, "Unsuccessful. Make sure
        the necessary data, such as election
        records, political dynasty
        associations, and complete area data,
         are in-place. " . $e->getMessage());
  }

  $app->response->setStatus(201);
  echo json_encode($dataset);
}
```

*app.html*

```
<!DOCTYPE html>
<html>
<head>
  <title>DynastyMap</title>

  <link rel="shortcut icon" type="image/png"
      href="favicon.png"/>

  <link rel="stylesheet" type="text/css" href
      ="css/pure-min.css" />
  <link rel="stylesheet" type="text/css" href
      ="css/bootstrap.min.css" />
  <link rel="stylesheet" type="text/css" href
      ="css/leaflet.css" />
  <link rel="stylesheet" type="text/css" href
      ="css/app.css" />

  <script src="js-src/lib/require.js"></script
      >
  <script src="js-src/config.js"></script>
</head>
<body>
  <div id="main">
    <div id="header" class="header layout-row
        "></div>
    <div id="sidebar" class="sidebar layout-
        col scroll-y"></div>
    <div id="body" class="body layout-row
        layout-col scroll-y"></div>
    <div id="infobar" class="infobar layout-
        row scroll-y"></div>
  </div>
  <script src="js-build/app.js"></script>
</body>
</html>
```

*dynastymap.sql*

```
-- phpMyAdmin SQL Dump
-- version 4.4.12deb1
-- http://www.phpmyadmin.net
--
-- Host:
-- Generation Time: Aug 19, 2015 at 09:42 PM
-- Server version:
-- PHP Version:

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=
    @@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=
    @@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=
    @@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Database: `dynavis`
--

--
    ------------------------------------------------------------

--
-- Table structure for table `area`
--

CREATE TABLE IF NOT EXISTS `area` (
  `id` int(11) NOT NULL,
  `code` int(9) NOT NULL,
  `name` varchar(48) COLLATE utf8_unicode_ci
      NOT NULL,
  `type` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=
    utf8_unicode_ci;

--
    ------------------------------------------------------------

--
-- Table structure for table `datapoint`
--

CREATE TABLE IF NOT EXISTS `datapoint` (
  `id` int(11) NOT NULL,
  `dataset_id` int(11) NOT NULL,
  `year` int(11) NOT NULL,
  `area_code` int(11) NOT NULL,
  `value` double DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=
    utf8_unicode_ci;

--
    ------------------------------------------------------------

--
-- Table structure for table `dataset`
--

CREATE TABLE IF NOT EXISTS `dataset` (
```

```
  `id` int(11) NOT NULL,
  `user_id` int(11) NOT NULL,
  `type` int(11) NOT NULL,
  `name` varchar(128) COLLATE utf8_unicode_ci
      NOT NULL,
  `description` text COLLATE utf8_unicode_ci
      NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=
    utf8_unicode_ci;

--
-- Table structure for table `elect`
--

CREATE TABLE IF NOT EXISTS `elect` (
  `id` int(11) NOT NULL,
  `official_id` int(11) NOT NULL,
  `year` int(11) NOT NULL,
  `year_end` int(11) NOT NULL,
  `position` varchar(48) COLLATE
      utf8_unicode_ci DEFAULT NULL,
  `votes` int(11) DEFAULT NULL,
  `area_code` int(11) NOT NULL,
  `party_id` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=
    utf8_unicode_ci;

--
-- Table structure for table `family`
--

CREATE TABLE IF NOT EXISTS `family` (
  `id` int(11) NOT NULL,
  `name` varchar(32) COLLATE utf8_unicode_ci
      NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=
    utf8_unicode_ci;

--
-- Table structure for table `
    family_membership`
--

CREATE TABLE IF NOT EXISTS `family_membership`
    (
  `official_id` int(11) NOT NULL,
  `family_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=
    utf8_unicode_ci;

--
-- Table structure for table `official`
--

CREATE TABLE IF NOT EXISTS `official` (
  `id` int(11) NOT NULL,
  `surname` varchar(32) COLLATE
      utf8_unicode_ci NOT NULL,
  `name` varchar(48) COLLATE utf8_unicode_ci
      NOT NULL,
  `nickname` varchar(24) COLLATE
      utf8_unicode_ci DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=
    utf8_unicode_ci;

--
-- Table structure for table `party`
--

CREATE TABLE IF NOT EXISTS `party` (
  `id` int(11) NOT NULL,
  `name` varchar(32) COLLATE utf8_unicode_ci
      NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=
    utf8_unicode_ci;

--
-- Table structure for table `tag_datapoint`
--

CREATE TABLE IF NOT EXISTS `tag_datapoint` (
  `id` int(11) NOT NULL,
  `dataset_id` int(11) NOT NULL,
  `year` int(11) NOT NULL,
  `area_code` int(11) NOT NULL,
  `family_id` int(11) NOT NULL,
  `value` double DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=
    utf8_unicode_ci;

--
-- Table structure for table `token`
--

CREATE TABLE IF NOT EXISTS `token` (
  `id` int(11) NOT NULL,
  `user_id` int(11) NOT NULL,
  `token` char(128) COLLATE utf8_unicode_ci
      NOT NULL,
  `expiry` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=
    utf8_unicode_ci;

--
-- Table structure for table `user`
--

CREATE TABLE IF NOT EXISTS `user` (
  `id` int(11) NOT NULL,
  `username` varchar(48) COLLATE
      utf8_unicode_ci NOT NULL,
  `active` tinyint(1) NOT NULL DEFAULT '0',
  `pw_hash` char(60) CHARACTER SET ascii
      COLLATE ascii_bin NOT NULL,
  `type` int(11) NOT NULL,
  `salt` char(128) CHARACTER SET ascii COLLATE
      ascii_bin NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=
    utf8_unicode_ci;

--
-- Indexes for dumped tables
--

--
-- Indexes for table `area`
--
ALTER TABLE `area`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `code` (`code`);

--
-- Indexes for table `datapoint`
--
ALTER TABLE `datapoint`
  ADD PRIMARY KEY (`id`),
  ADD KEY `dataset_id` (`dataset_id`),
  ADD KEY `area_code` (`area_code`);

--
-- Indexes for table `dataset`
--
ALTER TABLE `dataset`
  ADD PRIMARY KEY (`id`),
  ADD KEY `user_id` (`user_id`);

--
-- Indexes for table `elect`
--
ALTER TABLE `elect`
  ADD PRIMARY KEY (`id`),
  ADD KEY `official_id` (`official_id`),
  ADD KEY `area_code` (`area_code`),
  ADD KEY `party_id` (`party_id`);

--
-- Indexes for table `family`
--
ALTER TABLE `family`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `family_membership`
--
ALTER TABLE `family_membership`
  ADD PRIMARY KEY (`official_id`,`family_id`),
```

```
  ADD KEY `official_id` (`official_id`),
  ADD KEY `family_membership_ibfk_2` (`
      family_id`);

--
-- Indexes for table `official`
--
ALTER TABLE `official`
  ADD PRIMARY KEY (`id`),
  ADD KEY `Fullname` (`surname`,`name`,`
      nickname`);

--
-- Indexes for table `party`
--
ALTER TABLE `party`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `tag_datapoint`
--
ALTER TABLE `tag_datapoint`
  ADD PRIMARY KEY (`id`),
  ADD KEY `dataset_id` (`dataset_id`),
  ADD KEY `area_code` (`area_code`),
  ADD KEY `family_id` (`family_id`);

--
-- Indexes for table `token`
--
ALTER TABLE `token`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `token` (`token`),
  ADD KEY `user_id` (`user_id`);

--
-- Indexes for table `user`
--
ALTER TABLE `user`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `username` (`username`);

--
-- AUTO_INCREMENT for dumped tables
--


--
-- AUTO_INCREMENT for table `area`
--
ALTER TABLE `area`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
--
-- AUTO_INCREMENT for table `datapoint`
--
ALTER TABLE `datapoint`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
--
-- AUTO_INCREMENT for table `dataset`
--
ALTER TABLE `dataset`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
--
-- AUTO_INCREMENT for table `elect`
--
ALTER TABLE `elect`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
--
-- AUTO_INCREMENT for table `family`
--
ALTER TABLE `family`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
--
-- AUTO_INCREMENT for table `official`
--
ALTER TABLE `official`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
--
-- AUTO_INCREMENT for table `party`
--
ALTER TABLE `party`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
--
-- AUTO_INCREMENT for table `tag_datapoint`
--
ALTER TABLE `tag_datapoint`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
--
-- AUTO_INCREMENT for table `token`
--
ALTER TABLE `token`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
--
-- AUTO_INCREMENT for table `user`
--
ALTER TABLE `user`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
--
-- Constraints for dumped tables
--


--
-- Constraints for table `datapoint`
--
ALTER TABLE `datapoint`
  ADD CONSTRAINT `datapoint_ibfk_1` FOREIGN
      KEY (`dataset_id`) REFERENCES `dataset`
       (`id`) ON DELETE CASCADE ON UPDATE
      CASCADE,
  ADD CONSTRAINT `datapoint_ibfk_2` FOREIGN
      KEY (`area_code`) REFERENCES `area` (`
      code`) ON UPDATE CASCADE;

--
-- Constraints for table `dataset`
--
ALTER TABLE `dataset`
  ADD CONSTRAINT `dataset_ibfk_1` FOREIGN KEY
      (`user_id`) REFERENCES `user` (`id`) ON
       DELETE CASCADE ON UPDATE CASCADE;

--
-- Constraints for table `elect`
--
ALTER TABLE `elect`
  ADD CONSTRAINT `elect_ibfk_1` FOREIGN KEY (`
      official_id`) REFERENCES `official` (`
      id`) ON UPDATE CASCADE,
  ADD CONSTRAINT `elect_ibfk_3` FOREIGN KEY (`
      party_id`) REFERENCES `party` (`id`) ON
       UPDATE CASCADE,
  ADD CONSTRAINT `elect_ibfk_4` FOREIGN KEY (`
      area_code`) REFERENCES `area` (`code`)
      ON UPDATE CASCADE;

--
-- Constraints for table `family_membership`
--
ALTER TABLE `family_membership`
  ADD CONSTRAINT `family_membership_ibfk_1`
      FOREIGN KEY (`official_id`) REFERENCES
      `official` (`id`) ON DELETE CASCADE ON
      UPDATE CASCADE,
  ADD CONSTRAINT `family_membership_ibfk_2`
      FOREIGN KEY (`family_id`) REFERENCES `
      family` (`id`) ON DELETE CASCADE ON
      UPDATE CASCADE;

--
-- Constraints for table `tag_datapoint`
--
ALTER TABLE `tag_datapoint`
  ADD CONSTRAINT `tag_datapoint_ibfk_1`
      FOREIGN KEY (`dataset_id`) REFERENCES `
      dataset` (`id`) ON DELETE CASCADE ON
      UPDATE CASCADE,
  ADD CONSTRAINT `tag_datapoint_ibfk_3`
      FOREIGN KEY (`family_id`) REFERENCES `
      family` (`id`) ON DELETE CASCADE ON
      UPDATE CASCADE,
  ADD CONSTRAINT `tag_datapoint_ibfk_4`
      FOREIGN KEY (`area_code`) REFERENCES `
      area` (`code`) ON UPDATE CASCADE;

--
-- Constraints for table `token`
--
ALTER TABLE `token`
  ADD CONSTRAINT `token_ibfk_1` FOREIGN KEY (`
      user_id`) REFERENCES `user` (`id`) ON
      DELETE CASCADE ON UPDATE CASCADE;

/*!40101 SET CHARACTER_SET_CLIENT=
    @OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=
    @OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=
    @OLD_COLLATION_CONNECTION */;
```

*index.html*

```
<!DOCTYPE html>
<html>
<head>
  <title>DynastyMap</title>

  <link rel="shortcut icon" type="image/png"
      href="favicon.png"/>

  <link rel="stylesheet" type="text/css" href
      ="css/pure-min.css" />
```

```
  <link rel="stylesheet" type="text/css" href
      ="css/index.css" />
</head>
<body>
  <div class="splash-container">
    <div class="splash">
      <div class="pure-g">
        <div class="pure-u-1 logo-container">
          <div class="logo-large"></div>
          <div class="logo-type">DynastyMap</
              div>
        </div>
        <div class="pure-u-1 splash-title">An
            Interactive Data Visualization
            Tool on Political Dynasties in
            the Philippines</div>
        <div class="pure-u-1 launch-container
            ">
          <div><a href="app"><button class="
              splash-button button button-
              primary no-decor">Launch App</
              button></a></div>
          <div>or <a href="login">Sign In</a
              ></div>
        </div>
      </div>
    </div>
  </div>
  <div class="pure-g content">
    <div class="pure-u-1 text">
      <h1>Create Visualizations</h1>
      DynastyMap lets you create interactive
          visualizations of Philippine
          political dynasty data and
          geographic data.
    </div>
    <div class="pure-u-1-2 img-box fill-1" id
        ="img-choropleth">
      <img src="img/choropleth.png"/>
      <div class="img-box-caption">Visualize
          geographic data using choropleth
          maps.</div>
    </div>
    <div class="pure-u-1-2 img-box fill-1" id
        ="img-bivariate">
      <img src="img/bivariate.png"/>
      <div class="img-box-caption">Compare and
          contrast geographic data using
          bivariate choropleth maps.</div>
    </div>
    <div class="pure-u-1 dark">
      <div class="pure-u-1 text">
        <h1>Visualize Political Dynasties</h1>
        <p>Political dynasties are active in
            the Philippines, but how
            prevalent are they? In what ways
            do they affect the country?</p>
        <p>DynastyMap enables you to visualize
            and compare local political
            dynasties with Philippines
            geographic datasets such as
            poverty, population, and crime
            rate.</p>
      </div>
    </div>
    <div class="pure-u-1-2 img-box fill-2" id
        ="img-tag-map">
      <img src="img/tag-map.png"/>
      <div class="img-box-caption">See the
          relative prevalence of local
          political dynasties using the tag
          cloud map.</div>
    </div>
    <div class="pure-u-1-2 img-box fill-2" id
        ="img-tag-map-overlay">
      <img src="img/tag-map-overlay.png"/>
      <div class="img-box-caption">Compare
          local political dynasties with
          geographic data using layers of
          visualization.</div>
    </div>
    <div class="pure-u-1 text">
      <h1>Use Your Own Data</h1>
      <p>Got your own dataset that you wish to
          compare with political dynasties?
          Do you wish to visualize the
          results of a survey?</p>
      <p>DynastyMap allows approved users to
          upload their own geographic
          datasets for visualization and
          comparison with other datasets.</p>
    </div>
    <div class="pure-u-1-2 img-box fill-3" id
        ="img-upload">
      <img src="img/upload.png"/>
      <div class="img-box-caption">Use any
          Philippine geographical dataset at
          any administrative level &dash;
          regional, provincial, municipal,
          and barangay levels.</div>
    </div>
    <div class="pure-u-1-2 img-box fill-3" id
        ="img-upload-result">
      <img src="img/upload-result.png"/>
      <div class="img-box-caption">Visualize
          your own datasets along with other
          datasets for comparison.</div>
    </div>
    <div class="pure-u-1 about-section">
      <div class="text">
        <h1>About DynastyMap</h1>
        <p>DynastyMap is an interactive data
            visualization tool on political
            dynasties in the Philippines. The
            software uses choropleth maps
            and tag cloud maps to visualize
            Philippine datasets. Geographic
            datasets are uploaded by the
            users, but the political dynasty
            datasets are provided by the
            system.</p>
        <p>The software adapts the DYNSHA,
            DYNLAR, and DYNHERF metrics
            developed by Mendoza et al<a href
            ="#[1]">[1]</a> for quantifying
            local political dynasties. These
            metrics are used by the software
            to provide the political dynasty
            visualization.</p>
        <p>This project was developed by Lean
            Rada as a degree requirement (
            Special Problem) in B.S. Computer
            Science at the University of the
            Philppines Manila.</p>
        <ol class="text-small">
          <li><a name="[1]"></a>R. Mendoza, R.
              Beja Jr., V. Venida, and D.
              Yap, &ldquo;Political dynasties
              and poverty: Resolving the &
              ldquo;chicken or the egg&rdquo;
              question,&rdquo; 2013</li>
        </ol>
      </div>
    </div>
    <footer class="pure-u-1 footer-section">
      <p class="text-small pull-left">
        <a href="login">Login</a> &middot;
        <a href="admin">Admin</a> &middot;
        <a href="app">App</a>
      </p>
      <p class="text-small pull-right">&copy;
          2015</p>
      <div class="text-center">
        <img class="up-logo" src="img/up.png
            "/>
      </div>
    </footer>
  </div>
</body>
</html>
```

*js-src/AdminApp.jsx*

```
"use strict";
define(function(require){
  var Backbone = require("backbone"),
    React = require("react"),
    Events = require("minivents"),
    Token = require("model/Token"),
    InstanceCache = require("InstanceCache"),
    AdminRouter = require("jsx!AdminRouter"),
    Header = require("jsx!view/Header"),
    Sidebar = require("jsx!view/admin/Sidebar
        ");

  var AdminApp = function() {
    this.token = new Token();
    InstanceCache.set("Token", "session", this
        .token, false);

    this.bus = {
      router: new Events(),
    };

    this.router = new AdminRouter({bus: this.
        bus});
    this.token.on("change", this.check_login,
        this);
  };

  AdminApp.prototype.start = function() {
```

```
    if(!this.check_login()) return;

    this.header = React.render(<Header title="
        Dashboard" bus={this.bus} />,
        document.getElementById("header"));
    this.sidebar = React.render(<Sidebar bus={
        this.bus} />, document.getElementById
        ("sidebar"));

    Backbone.history.start();
  };

  AdminApp.prototype.check_login = function()
      {
    function go(url) {
      React.unmountComponentAtNode(document.
          getElementById("header"));
      React.unmountComponentAtNode(document.
          getElementById("sidebar"));
      React.unmountComponentAtNode(document.
          getElementById("body"));
      window.location.href = url;
    }

    if(!this.token.get_user()) {
      go(".");
      return false;
    }else if(this.token.get_user_role() &&
        this.token.get_user_role() != "admin
        ") {
      go("login?n=admin" + encodeURIComponent(
          window.location.hash));
      return false;
    }
    return true;
  };

  return AdminApp;
});
```

*js-src/AdminRouter.jsx*

```
"use strict";
define(["backbone", "react", "InstanceCache",
    "jsx!view/Spinner"], function(Backbone,
    React, InstanceCache, Spinner) {
  return Backbone.Router.extend({
    initialize: function(options) {
      this.bus = options.bus;
      this.listenTo(this, "route", this.
          on_route);
      this.official_collection = null;
      this.family_collection = null;
      this.election_collection = null;
      this.area_collection = null;
      this.dataset_collection = {};
      this.user_collection = null;
    },

    on_route: function(route, params) {
      React.render(<Spinner/>, document.
          getElementById("body"));
      this.bus.router.emit("route", {
        route: route,
        params: params,
      });
    },

    routes: {
      "": "home",
      "home": "home",
      "officials": "officials",
      "families": "families",
      "elections": "elections",
      "areas": "areas",
      "(users/:username/)datasets": "datasets
          ",
      "users": "users",
    },
    home: function() {
      require([
        "jsx!view/admin/HomePanel"
      ], function(HomePanel) {
        React.render(<HomePanel />, document.
            getElementById("body"));
      }.bind(this));
    },
    officials: function() {
      require([
        "model/OfficialCollection", "jsx!view/
            admin/OfficialsPanel"
      ], function(OfficialCollection,
          OfficialsPanel) {
        this.official_collection = this.
            official_collection || new
            OfficialCollection(null, {
            per_page: 8});
        React.render(<OfficialsPanel
            collection={this.
            official_collection} />, document
            .getElementById("body"));
        this.official_collection.resetParams()
            ;
        this.official_collection.fetch();
      }.bind(this));
    },
    families: function() {
      require([
        "model/FamilyCollection", "jsx!view/
            admin/FamiliesPanel"
      ], function(FamilyCollection,
          FamiliesPanel) {
        this.family_collection = this.
            family_collection || new
            FamilyCollection(null, {per_page:
            6});
        React.render(<FamiliesPanel collection
            ={this.family_collection} />,
            document.getElementById("body"));
        this.family_collection.resetParams();
        this.family_collection.fetch();
      }.bind(this));
    },
    elections: function() {
      require([
        "model/ElectionCollection", "jsx!view/
            admin/ElectionsPanel"
      ], function(ElectionCollection,
          ElectionsPanel) {
        this.election_collection = this.
            election_collection || new
            ElectionCollection(null, {
            per_page: 8});
        React.render(<ElectionsPanel
            collection={this.
            election_collection} />,
            document.getElementById("body"));
        this.election_collection.resetParams()
            ;
        this.election_collection.fetch();
      }.bind(this));
    },
    datasets: function(username) {
      require([
        "model/DatasetCollection", "jsx!view/
            admin/DatasetsPanel"
      ], function(DatasetCollection,
          DatasetsPanel) {
        this.dataset_collection[username] =
            this.dataset_collection[username]
            || new DatasetCollection(null,
            username ? {username: username} :
            null);
        React.render(<DatasetsPanel collection
            ={this.dataset_collection[
            username]} />, document.
            getElementById("body"));
        this.dataset_collection[username].
            resetParams();
        this.dataset_collection[username].
            fetch();
      }.bind(this));
    },
    areas: function(username) {
      require([
        "model/AreaCollection", "jsx!view/
            admin/AreasPanel"
      ], function(AreaCollection, AreasPanel)
          {
        this.area_collection = this.
            area_collection || new
            AreaCollection();
        React.render(<AreasPanel collection={
            this.area_collection} />,
            document.getElementById("body"));
        this.area_collection.resetParams();
        this.area_collection.fetch();
      }.bind(this));
    },
    users: function() {
      require([
        "model/UserCollection", "jsx!view/
            admin/UsersPanel"
      ], function(UserCollection, UsersPanel)
          {
        this.user_collection = this.
            user_collection || new
            UserCollection();
        React.render(<UsersPanel collection={
            this.user_collection} />,
```

```
            document.getElementById("body"));
        this.user_collection.resetParams();
        this.user_collection.fetch();
      }.bind(this));
```

*js-src/ChoroplethDataProcessor.js*

```
"use strict";
define(["underscore", "jenks", "model/Area"],
    function(_, jenks, Area) {
  var CDP = function(bus) {
    this.bus = bus;
    this.datasets = null;
    this.level = null;
    this.year = null;

    this.bus.main_settings.on("update", this.
        update_main.bind(this));
    this.bus.choropleth_settings.on("update",
        this.update_dataset.bind(this));
  };

  CDP.prototype.update_main = function(
      settings) {
    if(settings.level) this.level = settings.
        level;
    if(settings.year) this.year = settings.
        year;
    this.update();
  };
  CDP.prototype.update_dataset = function(
      settings) {
    this.datasets = [settings.dataset1,
        settings.dataset2];
    this.update();
  };

  CDP.prototype.update = _.debounce(function()
      {
    if(!this.year || !this.level || !this.
        datasets) return;

    var callback = _.after(2, function() {
      var processed = _.map(this.datasets,
          function(d, i) {
        if(d) return this.process(d, i);
        return null;
      }, this);
      this.bus.choropleth_data.emit("update",
          processed);
    }.bind(this));

    _.each(this.datasets, function(d) {
      if(!d || d.get_datapoints(this.year).
          size()){
        callback();
      }else{
        d.get_datapoints(this.year).fetch({
          success: callback
        });
      }
    }, this);
  }, 0);

  CDP.prototype.process = function(dataset, i)
      {
    var datapoints = dataset.get_datapoints(
        this.year)
      .filter(function(p) {
        return Area.get_level(p.get("area_code
            ")) == this.level;
      }, this);
    if(!datapoints.length) return null;
    var scales_hex = [
      ["#FFBFD5", "#F379A3", "#D9356D", "#
          A60038"],
      ["#BFF8FF", "#79E5F3", "#35C6D9", "#0094
          A6"],
    ];

    var scales = _.map(scales_hex, function(
        scale_hex) {
      return _.map(scale_hex, function(hex) {
        var rr = hex.substr(1,2);
        var gg = hex.substr(3,2);
        var bb = hex.substr(5,2);
        return {r:parseInt(rr,16), g:parseInt(
            gg,16), b:parseInt(bb,16)};
      });
    });

    return {
      name: dataset.get("name"),
      min_year: dataset.get("min_year"),
      max_year: dataset.get("max_year"),
      year: this.year,
      datapoints: datapoints,
      classes: this.calculate_breaks(
          datapoints, 4, dataset, this.level,
          this.year),
      color_scale: scales[i],
    };
  };

  CDP.prototype.calculate_breaks = _.memoize(
      function(datapoints, n, dataset, level,
          year) {
    var data = _.chain(datapoints)
      .filter(function(p) {
        return p.get("value") !== null;
      })
      .map(function(p) { return p.get("value")
          ; })
      .value();

    if(data.length > n + 1) {
      var breaks = jenks(data, n);
      if(breaks && !_.some(breaks, function(b)
          { return isNaN(b); })) return
          breaks;
    }

    var min = _.min(data);
    var max = _.max(data);
    breaks = [];
    for (var i = 0; i <= n; i++) {
      breaks.push(min + (max - min) * i / n);
    }
    return breaks;
  }, function(datapoints, n, dataset, level,
      year) {
    return n + ":" + dataset.id + "|" + level
        + "|" + year;
  });

  return CDP;
});
```

*js-src/InstanceCache.js*

```
"use strict";
(function() {

var MODEL_PATHS = {
  "Official": "model/OfficialSingle",
  "Family": "model/FamilySingle",
  "Area": "model/Area",
  "Party": "model/Party",
  "Election": "model/ElectionSingle",
  "Dataset": "model/Dataset",
  "User": "model/User",
  "Token": "model/Token",
};

var MODEL_PATHS_VALUES = []
for(var key in MODEL_PATHS) {
  MODEL_PATHS_VALUES.push(MODEL_PATHS[key]);
}

define(["require", "underscore", "bloodhound
    "].concat(MODEL_PATHS_VALUES), function(
    require, _, Bloodhound) {
  var create_hound = function(path) {
    return new Bloodhound({
      queryTokenizer: Bloodhound.tokenizers.
          nonword,
      datumTokenizer: Bloodhound.tokenizers.
          obj.nonword,
      remote: {
        prepare: function(query, settings) {
          query.q = query.string; delete query
              .string;
          query.count = query.limit; delete
              query.limit;
          return _.extend(settings, {
            url: "api.php/" + path + "?" + _.
                map(_.keys(query), function(k
                ) {
              var v = query[k];
```

```
            if(typeof v == "boolean") v = v
                ? 1 : 0;
            return encodeURIComponent(k) +
                "=" + encodeURIComponent(v)
                ;
          }).join("&"),
        });
      },
      url: "api.php/" + path + "?count=1",
      transform: function(data) { return
          data.data; },
    },
  });
};

var InstanceCache = function() {
  this.model_paths = MODEL_PATHS;
  this.active_list = [];
  this.hash = {};
  this.models = {};
  this.hounds = {
    "Official": create_hound("officials"),
    "Family": create_hound("families"),
    "Area": create_hound("areas"),
    "Party": create_hound("parties"),
  };
};

InstanceCache.prototype.get = function(name,
    id, fetch) {
  if(id !== id) return null; // for NaN

  this.set_active(name, id);
  this.remove_expired();

  if(this.hash[name]) {
    if(this.hash[name][id]) {
      return this.hash[name][id];
    }
  }else{
    this.hash[name] = {};
  }

  if(!this.models[name]) {
    this.models[name] = require(this.
        model_paths[name]);
  }

  var instance = this.hash[name][id] = new
      this.models[name]();
  instance.set(instance.idAttribute, id);

  if(fetch) instance.fetch();

  return instance;
};

InstanceCache.prototype.get_existing =
    function(name, id) {
  if(this.hash[name] && this.hash[name][id])
      return this.hash[name][id];
  return null;
};

InstanceCache.prototype.set = function(name,
    key, value, expires) {
  if(typeof expires === "undefined") expires
      = true;

  if(!this.hash[name]) this.hash[name] = {};
  this.hash[name][key] = value;
  if(expires) this.set_active(name, key);
};

InstanceCache.prototype.delete = function(
    name, key) {
  if(this.hash[name]) delete this.hash[name
      ][key];
};

InstanceCache.prototype.search = function(
    name, query, callback, async) {
  if(typeof query === "string") {
    query = {
      string: query,
      limit: 6,
    };
  }

  var callback_sync, callback_async;
  if(async) {
    callback_sync = callback;
    callback_async = async;
  }else{
    callback_sync = callback_async = (
        function() {
      var fin = false;
      var exec_count = 0;
      var collected_data = [];
      return function(data) {
        if(!fin) {
          exec_count++;
          collected_data = collected_data.
              concat(data);
          if (collected_data.length >= query
              .limit || exec_count == 2) {
            fin = true;
            callback(data);
          }
        }
      };
    })();
  }

  this.hounds[name].search(query,
      callback_sync, callback_async);
};

InstanceCache.prototype.set_active =
    function(name, key, time) {
  if (typeof time === "undefined") time =
      60000;

  var obj = _.findWhere(this.active_list, {
      name: name, key: key});
  if(obj) {
    obj.expiry += time;
  }else{
    obj = {
      name: name,
      key: key,
      expiry: Date.now() + time,
    };
    this.active_list.push(obj);
  }
};

InstanceCache.prototype.remove_expired =
    function() {
  var now = Date.now();
  _.each(this.active_list, function(obj) {
    if(obj.expiry < now) {
      if(this.hash[obj.name]) delete this.
          hash[obj.name][obj.key];
    }
  }, this);
  this.active_list = _.reject(this.
      active_list, function(obj) {
    return obj.expiry < now;
  });
};

return new InstanceCache();
});

})();
```

*js-src/LoginApp.jsx*

```
"use strict";
define(function(require){
  var Backbone = require("backbone"),
    React = require("react"),
    Events = require("minivents"),
    Token = require("model/Token"),
    InstanceCache = require("InstanceCache"),
    LoginPage = require("jsx!view/login/
        LoginPage");

  var LoginApp = function() {
    this.token = new Token();
    InstanceCache.set("Token", "session", this
        .token, false);
  };

  LoginApp.prototype.start = function() {
    if(this.token.get_user()) {
      this.on_login();
    }else{
      this.login = React.render(<LoginPage
          model={this.token} onLogin={this.
          on_login.bind(this)} />, document.
          getElementById("body"));
      Backbone.history.start();
    }
  };

  LoginApp.prototype.on_login = function() {
    var target = (location.search.split("n=")
        [1]||"").split("&")[0] || (this.token
        .get_user_role() == "admin" ? "admin"
```

```
        : "app");
    window.location.href = decodeURIComponent(
        target);
  };
```

*js-src/MainApp.jsx*

```
"use strict";
define(function(require){
  var Backbone = require("backbone"),
    React = require("react"),
    Events = require("minivents"),
    Token = require("model/Token"),
    InstanceCache = require("InstanceCache"),
    MainRouter = require("jsx!MainRouter"),
    Header = require("jsx!view/Header"),
    Sidebar = require("jsx!view/main/Sidebar")
        ,
    Infobar = require("jsx!view/main/Infobar")
        ,
    ChoroplethDataProcessor = require("
        ChoroplethDataProcessor"),
    TagCloudDataProcessor = require("
        TagCloudDataProcessor");

  var MainApp = function() {
    this.token = new Token();
    InstanceCache.set("Token", "session", this
        .token, false);
    this.bus = {
      router: new Events(),

      main_settings: new Events(),
      choropleth_settings: new Events(),
      tagcloud_settings: new Events(),

      choropleth_data: new Events(),
      tagcloud_data: new Events(),
    };

    new ChoroplethDataProcessor(this.bus);
    new TagCloudDataProcessor(this.bus);

    this.router = new MainRouter({bus: this.
        bus});
    this.token.on("change", this.check_login,
        this);
  };

  MainApp.prototype.start = function() {
    this.header = React.render(<Header title
        ="" bus={this.bus} />, document.
        getElementById("header"));
    this.sidebar = React.render(<Sidebar bus={
        this.bus} />, document.getElementById
        ("sidebar"));
    this.infobar = React.render(<Infobar bus={
        this.bus} />, document.getElementById
        ("infobar"));

    Backbone.history.start();
  };

  MainApp.prototype.check_login = function() {
    if(Backbone.history.getFragment() !== ""
        && !this.token.get_user()) {
      this.router.navigate("", true);
      return false;
    }
    return true;
  };

  return MainApp;
});
```

*js-src/MainRouter.jsx*

```
"use strict";
define(["backbone", "react", "InstanceCache",
    "jsx!view/Spinner"], function(Backbone,
    React, InstanceCache, Spinner) {
  return Backbone.Router.extend({
    initialize: function(options) {
      this.bus = options.bus;
      this.listenTo(this, "route", this.
          on_route);
    },

    on_route: function(route, params) {
      React.render(<Spinner/>, document.
          getElementById("body"));
      this.bus.router.emit("route", {
        route: route,
        params: params,
      });
    },

    routes: {
      "": "main",
      "datasets": "datasets",
      "datasets/:username/:id": "datapoints",
    },
    main: function() {
      var that = this;
      require([
        "jsx!view/main/MapPanel"
      ], function(MapPanel) {
        React.render(<MapPanel bus={that.bus}
            />, document.getElementById("body
            "));
      });
    },
    datasets: function() {
      var that = this;
      require([
        "model/DatasetCollection", "jsx!view/
            main/DatasetsPanel"
      ], function(DatasetCollection,
          DatasetsPanel) {
        var token = InstanceCache.get_existing
            ("Token", "session");
        var user = token ? token.get_user() :
            null;
        if(!user) {
          window.location.href = "login?n=app"
              + encodeURIComponent(window.
              location.hash);
          return;
        }

        var dataset_collection = new
            DatasetCollection(null, {type: "
            area", username: user.get("
            username")});
        dataset_collection.fetch();
        React.render(<DatasetsPanel collection
            ={dataset_collection}/>, document
            .getElementById("body"));
      });
    },
    datapoints: function(username, id) {
      var that = this;
      require([
        "model/Dataset", "model/
            DatapointPageableCollection", "
            jsx!view/main/DatapointsPanel"
      ], function(Dataset,
          DatapointPageableCollection,
          DatapointsPanel) {
        var token = InstanceCache.get_existing
            ("Token", "session");
        var user = token ? token.get_user() :
            null;
        if(!user) {
          window.location.href = "login?n=app"
              + encodeURIComponent(window.
              location.hash);
          return;
        }
        var dataset = new Dataset({username:
            username, id: id});
        var datapoint_collection = new
            DatapointPageableCollection(null,
            {dataset: dataset});
        dataset.fetch({
          success: function(model) {
            datapoint_collection.fetch();
          },
        });
        React.render(<DatapointsPanel model={
            dataset} collection={
            datapoint_collection}/>, document
            .getElementById("body"));
      });
    },
  });
});
```

*js-src/TagCloudDataProcessor.js*

```
"use strict";
define(["underscore", "jenks", "model/Area", "
    ChoroplethDataProcessor"], function(_,
    jenks, Area, CDP) {
  var TCDP = function(bus) {
    this.bus = bus;
    this.dataset = null;
    this.level = null;
    this.year = null;

    this.bus.main_settings.on("update", this.
        update_main.bind(this));
    this.bus.tagcloud_settings.on("update",
        this.update_dataset.bind(this));
  };

  TCDP.prototype.update_main = function(
      settings) {
    if(settings.level) this.level = settings.
        level;
    if(settings.year) this.year = settings.
        year;
    this.update();
  };
  TCDP.prototype.update_dataset = function(
      settings) {
    this.dataset = settings.dataset;
    this.update();
  };

  TCDP.prototype.update = _.debounce(function
      () {
    if(!this.year || !this.level) return;

    var callback = function(){
      var processed = this.dataset ? this.
          process(this.dataset) : null;
      this.bus.tagcloud_data.emit("update",
          processed);
    }.bind(this);

    if(!this.dataset || this.dataset.
        get_datapoints(this.year).size()){
      callback();
    }else{
      this.dataset.get_datapoints(this.year).
          fetch({
        success: callback
      });
    }
  }, 0);

  TCDP.prototype.process = function(dataset) {
    var datapoints = dataset.get_datapoints(
        this.year)
      .filter(function(p) {
        return Area.get_level(p.get("area_code
            ")) == this.level;
      }, this);
    if(!datapoints.length) return null;

    return {
      name: dataset.get("name"),
      min_year: dataset.get("min_year"),
      max_year: dataset.get("max_year"),
      year: this.year,
      datapoints: datapoints,
      classes: this.calculate_breaks(
          datapoints, 3, dataset, this.level,
          this.year),
    };
  };

  TCDP.prototype.calculate_breaks = CDP.
      prototype.calculate_breaks;

  return TCDP;
});
```

*js-src/admin.js*

```
"use strict";
require(["jsx!AdminApp"], function(AdminApp){
  var app = new AdminApp();

  app.start();
});
```

*js-src/app.js*

```
"use strict";
require(["jsx!MainApp"], function(MainApp){
  var app = new MainApp();

  app.start();
});
```

*js-src/config.js*

```
"use strict";
require.config({
  // urlArgs: "bust=" + (new Date()).getTime()
      ,

  baseUrl: "js-build",

  paths: {
    "backbone": "lib/backbone",
    "backbone-pagec": "lib/backbone-pagec",
    "jquery.bez": "lib/jquery.bez.min",
    "bloodhound": "lib/bloodhound",
    "bootstrap": "lib/bootstrap.min",
    "config.map": "config.map",
    "d3": "lib/d3.min",
    "jenks": "lib/jenks",
    "jquery": "lib/jquery-2.1.4.min",
    "jsx": "lib/jsx",
    "JSXTransformer": "lib/JSXTransformer",
    "leaflet": "lib/leaflet",
    "localStorage": "lib/localStorage",
    "minivents": "lib/minivents.min",
    "numf": "lib/numf",
    "react": "lib/react",
    "react.backbone": "lib/react.backbone",
    "text": "lib/text",
    "typeahead": "lib/typeahead.jquery",
    "underscore": "lib/underscore-min",
    "validator": "lib/validator",
  },

  shim : {
    "backbone": {
      deps: ["underscore", "jquery"],
      exports: "Backbone"
    },
    "backbone-pagec": {
      deps: ["backbone"],
      exports: "Backbone.PageableCollection"
    },
    "bootstrap": { deps: ["jquery"], },
    "jenks": { exports: "jenks" },
    "JSXTransformer": "JSXTransformer",
    "localStorage": { exports: "localStorage"
        },
    "minivents": { exports: "Events" },
    "react": { exports: "React" },
    "react.backbone": { deps: ["react", "
        backbone"] },
    "underscore": { exports: "_" },
  },

  config: {
    jsx: {
      fileExtension: ".jsx",
      harmony: true,
    }
  },
});

requirejs.onError = function (err) {
  console.error(err);
  if (err.requireType === 'timeout') {
    console.error('modules: ' + err.
        requireModules);
  }
};
```

*js-src/config.map.sample.js*

```
"use strict";
define(function() {
  return {
    url: "http://{s}.example.com/{z}/{x}/{y}.
        png",
    attribution: "&copy;",
  };
});
```

*js-src/lib/backbone-pagec.js*

```
"use strict";
var Backbone = Backbone || {};
(function() {
  Backbone.PageableCollection = Backbone.
      Collection.extend({
    start: 0,
    per_page: 15,
    total: 0,

    initialize: function(models, options) {
      options = options || {};
      this.reset = true;
      this.per_page = options.per_page || this
          .per_page;
      this.total = options.total || this.total
          ;
      this.params = options.params || {};
    },

    getPage: function() {
      return Math.ceil(this.start/this.
          per_page);
    },
    getTotalPages: function() {
      return Math.ceil(this.total/this.
          per_page);
    },

    resetParams: function() {
      this.params = {};
    },
    setParams: function(params) {
      this.params = params;
    },

    next: function(options) {
      if(this.start + this.per_page >= this.
          total) {
        this.start = this.per_page < this.
            total ? this.total - this.
            per_page : 0;
        console.error("Error next. No pages
            left.");
        return null;
      }else{
        this.start += this.per_page;
      }
      return this.fetch(options);
    },
    prev: function(options) {
      if(this.start <= 0) {
        this.start = 0;
        console.error("Error prev. No pages
            left.");
        return null;
      }else if(this.start <= this.per_page) {
        this.start = 0;
      }else{
        this.start -= this.per_page;
      }
      return this.fetch(options);
    },
    page: function(page, options) {
      if(page < 0 || page * this.per_page >
          this.total) {
        console.error("Error page. Out of
            bounds.");
        return null;
      }
      this.start = page * this.per_page;
      return this.fetch(options);
    },

    fetch: function(options) {
      var that = this;

      options = options || {};
      options.data = options.data || {}
      if(this.params) {
        _.each(this.params, function(v,k) {
          if(!(k in options.data)) options.
              data[k] = v;
        });
      }

      if(options.data.start === undefined)
          options.data.start = this.start;
      if(options.data.count === undefined)
          options.data.count = this.per_page;

      var success = options.success;
      options.success = function(c,r,o) {
        that.total = r.total;
        if(success) success(c,r,o);
      };

      return Backbone.Collection.prototype.
          fetch.call(this, options);
    },
  });
})();
```

*js-src/lib/jenks.js*

```
// # [Jenks natural breaks optimization](http
    ://en.wikipedia.org/wiki/
    Jenks_natural_breaks_optimization)
//
// Implementations: [1](http://danieljlewis.
    org/files/2010/06/Jenks.pdf) (python),
// [2](https://github.com/vvoovv/djeo-jenks/
    blob/master/main.js) (buggy),
// [3](https://github.com/simogeo/geostats/
    blob/master/lib/geostats.js#L407) (works)
function jenks(data, n_classes) {

    // Compute the matrices required for Jenks
        breaks. These matrices
    // can be used for any classing of data
        with `classes <= n_classes`
    function getMatrices(data, n_classes) {

        // in the original implementation,
            these matrices are referred to
        // as `LC` and `OP`
        //
        // * lower_class_limits (LC): optimal
            lower class limits
        // * variance_combinations (OP):
            optimal variance combinations for
             all classes
        var lower_class_limits = [],
            variance_combinations = [],
            // loop counters
            i, j,
            // the variance, as computed at
                each step in the calculation
            variance = 0;

        // Initialize and fill each matrix
            with zeroes
        for (i = 0; i < data.length + 1; i++)
            {
            var tmp1 = [], tmp2 = [];
            for (j = 0; j < n_classes + 1; j
                ++) {
                tmp1.push(0);
                tmp2.push(0);
            }
            lower_class_limits.push(tmp1);
            variance_combinations.push(tmp2);
        }

        for (i = 1; i < n_classes + 1; i++) {
            lower_class_limits[1][i] = 1;
            variance_combinations[1][i] = 0;
            // in the original implementation,
                9999999 is used but
            // since Javascript has `Infinity
                `, we use that.
            for (j = 2; j < data.length + 1; j
                ++) {
                variance_combinations[j][i] =
                    Infinity;
            }
        }

        for (var l = 2; l < data.length + 1; l
            ++) {

            // `SZ` originally. this is the
                sum of the values seen thus
```

```
// far when calculating variance.
var sum = 0,
    // 'ZSQ' originally. the sum
        of squares of values seen
    // thus far
    sum_squares = 0,
    // 'WT' originally. This is
        the number of
    w = 0,
    // 'IV' originally
    i4 = 0;

// in several instances, you could
    say 'Math.pow(x, 2)'
// instead of 'x * x', but this is
    slower in some browsers
// introduces an unnecessary
    concept.
for (var m = 1; m < l + 1; m++) {

    // 'III' originally
    var lower_class_limit = l - m
        + 1,
        val = data[
            lower_class_limit -
            1];

    // here we're estimating
        variance for each
        potential classing
    // of the data, for each
        potential number of
        classes. 'w'
    // is the number of data
        points considered so far.
    w++;

    // increase the current sum
        and sum-of-squares
    sum += val;
    sum_squares += val * val;

    // the variance at this point
        in the sequence is the
        difference
    // between the sum of squares
        and the total x 2, over
        the number
    // of samples.
    variance = sum_squares - (sum
        * sum) / w;

    i4 = lower_class_limit - 1;

    if (i4 !== 0) {
        for (j = 2; j < n_classes
            + 1; j++) {
            // if adding this
                element to an
                existing class
            // will increase its
                variance beyond
                the limit, break
            // the class at this
                point, setting
                the
                lower_class_limit
            // at this point.
            if (
                variance_combinations
                [l][j] >=
                (variance +
                    variance_combinations
                    [i4][j - 1]))
                    {
                lower_class_limits
                    [l][j] =
                    lower_class_limit
                    ;
                variance_combinations
                    [l][j] =
                    variance +      }
                variance_combinations
                    [i4][j -
                        1];
                }
            }
        }
    }

    lower_class_limits[l][1] = 1;
    variance_combinations[l][1] =
        variance;
}

    // return the two matrices. for just
        providing breaks, only
    // 'lower_class_limits' is needed, but
        variances can be useful to
    // evaluage goodness of fit.
    return {
        lower_class_limits:
            lower_class_limits,
        variance_combinations:
            variance_combinations
    };
}


// the second part of the jenks recipe:
    take the calculated matrices
// and derive an array of n breaks.
function breaks(data, lower_class_limits,
    n_classes) {

    var k = data.length - 1,
        kclass = [],
        countNum = n_classes;

    // the calculation of classes will
        never include the upper and
    // lower bounds, so we need to
        explicitly set them
    kclass[n_classes] = data[data.length -
        1];
    kclass[0] = data[0];

    // the lower_class_limits matrix is
        used as indexes into itself
    // here: the 'k' variable is reused in
        each iteration.
    while (countNum > 1) {
        kclass[countNum - 1] = data[
            lower_class_limits[k][
            countNum] - 2];
        k = lower_class_limits[k][countNum
            ] - 1;
        countNum--;
    }

    return kclass;
}

if (n_classes > data.length) return null;

// sort data in numerical order, since
    this is expected
// by the matrices function
data = data.slice().sort(function (a, b) {
     return a - b; });

// get our basic matrices
var matrices = getMatrices(data, n_classes
    ),
    // we only need lower class limits
        here
    lower_class_limits = matrices.
        lower_class_limits;

// extract n_classes out of the computed
    matrices
return breaks(data, lower_class_limits,
    n_classes);
```

*js-src/lib/localStorage.js*

```
if (!('localStorage' in window)) {
  window.localStorage = {
    _data       : {},
    setItem     : function(id, val) { return
        this._data[id] = String(val); },
    getItem     : function(id) { return this.
        _data.hasOwnProperty(id) ? this._data
            [id] : undefined; },
    removeItem  : function(id) { return delete
         this._data[id]; },
    clear       : function() { return this.
        _data = {}; }
  };
}
```

*js-src/lib/numf.js*

```
"use strict";
define(function(require) {
  return {
    format: function(n) {
      var abs = Math.abs(n);
      var s, m, d;
      if(abs >= 100000000) {
        m = 1/1000000000;
        d = 2;
        s = "B";
      }else if(abs >= 100000) {
        m = 1/1000000;
        d = 1;
        s = "M";
      }else if(abs >= 1000) {
        m = 1/1000;
        d = 1;
        s = "K";
      }else if(abs >= 0.1) {
        m = 1;
        d = 2;
        s = "";
      }else if(abs >= 0.01) {
        return n.toFixed(2);
      }else if(abs > 0){
        return n.toExponential(1);
      }else{
        return "0";
      }

      if((abs*m) % 1 < 0.1) return Math.round(
          n*m) + s;
      else return (n*m).toFixed(d) + s;
    },
  };
});
```

*js-src/lib/validator.js*

```
"use strict";
define(["underscore"], function(_) {
  var Va = function(schema) {
    this.schema = schema || {};
  };

  Va.prototype.lidate = function(object,
      whitelist) {
    var valid = true;
    var messages = {};

    var keys = _.union(_.keys(object), _.keys(
        this.schema));
    if(whitelist) keys = _.pick(keys,
        whitelist);

    _.each(keys, function(k) {
      var validator = this.schema[k];
      if(!validator) {
        valid = false;
        messages[k] = "is an unknown property
            ";
      }
      validator = validator.root;
      var err, value = object[k];
      do {
        if(validator.getErrorMsg && (err =
            validator.getErrorMsg(value,
            object)) !== null) {
          valid = false;
          messages[k] = err;
          break;
        }
        validator = validator.next;
      }while(validator);
    }, this);

    return {
      valid: valid,
      messages: messages,
    };
  };

  Va.lidator = function() {
    return new Va.Node();
  };

  Va.Node = function() {
    this.getErrorMsg = null;
    this.next = null;
    this.root = this;
  };

  Va.Node.prototype._end = function() {
    this.next = new Va.Node();
    this.next.root = this.root;
    return this.next;
  };

  Va.Node.prototype.optional_if = function(
      condition) {
    this.getErrorMsg = function(obj, others) {
      if(condition(obj, others)) this.next =
          null;
      return null;
    };
    return this._end();
  };
  Va.Node.prototype.custom = function(method,
      errorMessage) {
    this.getErrorMsg = function(obj, others) {
      if(!method(obj, others)) return
          errorMessage;
      return null;
    };
    return this._end();
  };
  Va.Node.prototype.any = function() {
    this.getErrorMsg = function(obj) {
      return null;
    };
    return this._end();
  };
  Va.Node.prototype.required = function() {
    this.getErrorMsg = function(obj) {
      if(!obj) return "is required";
      return null;
    };
    return this._end();
  };
  Va.Node.prototype.object = function() {
    this.getErrorMsg = function(obj) {
      if(obj === null) return null;
      if(typeof obj !== "object") return "must
          be an object";
      return null;
    };
    return this._end();
  };
  Va.Node.prototype.string = function(minLen,
      maxLen) {
    minLen = minLen || 0;
    maxLen = maxLen || Infinity;
    this.getErrorMsg = function(obj) {
      if(obj === null) return null;
      if(typeof obj !== "string") return "must
          be a string";
      var clauses = [];
      if(minLen > 0) clauses.push("at least "
          + minLen + " characters long");
      if(maxLen < Infinity) clauses.push("not
          longer than " + maxLen + "
          characters");
      if(obj.length < minLen || obj.length >
          maxLen) return "must be " + clauses
          .join(" but ");
      return null;
    };
    return this._end();
  };
  Va.Node.prototype.integerish = function(min,
      max) {
    min = min || 0;
    max = max || Infinity;
    this.getErrorMsg = function(obj) {
      // http://stackoverflow.com/a/14794066
      var msg = "must be an integer";
      if(obj === null) return null;
      if (isNaN(obj)) return msg;
      var x = parseFloat(obj);
      if((x | 0) !== x) return msg;
      var clauses = [];
      if(min > 0) clauses.push("at least " +
          min);
      if(max < Infinity) clauses.push("not
          greater than " + max);
      if(x < min || x > max) return "must be "
          + clauses.join(" but ");
      return null;
    };
    return this._end();
  };
  Va.Node.prototype.floatish = function(min,
      max) {
    min = min || 0;
    max = max || Infinity;
    this.getErrorMsg = function(obj) {
      // http://stackoverflow.com/a/14794066
      if(obj === null) return null;
      var x = parseFloat(obj);
      if (isNaN(x)) return "must be a number";
      var clauses = [];
```

```
      if(min > 0) clauses.push("at least " +
          min);
      if(max < Infinity) clauses.push("not
          greater than " + max);
      if(x < min || x > max) return "must be "
          + clauses.join(" but ");
      return null;
    };
    return this._end();
  };
  Va.Node.prototype.length = function(a, b) {
    this.getErrorMsg = function(obj) {
      var len = obj.length;
      if(typeof a !== "undefined" && typeof b
          !== "undefined")
        if(len < a || len > b) return "length
            must be between " + a + " and " +
             b;
      if(typeof a !== "undefined")
        if(len !== a) return "must be " + a +
            " long";
      if(len == 0) return "must not be empty";
      return null;
    };
    return this._end();
  };
  Va.Node.prototype.lessThan = function(value)
      {
    this.getErrorMsg = function(obj, others) {
      var y = value, yname = value;
      if(typeof value === "object") {
        var key = value.key;
        y = parseFloat(others[key]);
        yname = key;
      }
      if(parseFloat(obj) >= y) return "must be
          less than " + yname;
      return null;
    };
    return this._end();
  };
  Va.Node.prototype.greaterThan = function(
      value) {
    this.getErrorMsg = function(obj, others) {
      var y = value, yname = value;
      if(typeof value === "object") {
        var key = value.key;
        y = parseFloat(others[key]);
        yname = key;
      }
      if(parseFloat(obj) <= y) return "must be
          greater than " + yname;
      return null;
    };
    return this._end();
  };

  return Va;
});
```

*js-src/login.js*

```
"use strict";
require(["jsx!LoginApp"], function(LoginApp){
  var app = new LoginApp();
  app.start();
});
```

*js-src/mixin/ClickToTopMixin.js*

```
"use strict";
define(["jquery", "react"], function($, React)
    {
  return {
    componentDidMount: function() {
      var $el = $(React.findDOMNode(this));
      $el.click(function(e) {
        var max = -Infinity;
        $el.siblings().each(function(i,s) {
          var z = parseInt($(s).css("z-index")
              , 10);
          if(max < z) max = z;
        });
        if(max === -Infinity) max = 0;
        $el.css("z-index", max + 1);
      });
    },
  };
});
```

*js-src/mixin/ScrollToTopMixin.js*

```
"use strict";
define(["jquery", "react"], function($, React)
    {
  return {
    scroll_to_top: function(smooth) {
      var $el = $(React.findDOMNode(this));
      if(smooth) {}
      $el.parent().scrollTop(0);
    },
  };
});
```

*js-src/mixin/ValidationMixin.js*

```
"use strict";
define(["underscore", "validator"], function(_
    , Va) {
  return {
    getInitialState: function() {
      return {
        validation: {
          valid: true,
          messages: {},
        }
      };
    },

    componentWillMount: function() {
      this._va = new Va(this.
          getValidationSchema());
    },

    resetValidation: function() {
      this.setState({
        validation: {
          valid: true,
          messages: {},
        }
      });
    },

    validate: function(keys) {
      var obj = this.getObjectToValidate();
      var result = this._va.lidate(obj, keys);
      this.setState({validation: result});
      if(result.valid) {
        _.mapObject(obj, function(value, key)
            {
          this._validationResult(key, true,
              null);
        }, this);
      }else{
        _.mapObject(result.messages, function(
            message, key) {
          this._validationResult(key, false,
              message);
        }, this);
      }
      return result.valid;
    },

    validationOverride: function(key, valid,
        message) {
      var result = this.state.validation;
      result.valid = valid;
      result.messages[key] = message;
      this.setState({validation: result});
      this._validationResult(key, valid,
          message);
    },

    _validationResult: function(key, valid,
        message) {
      if(typeof this.validationCallback === "
          function") this.validationCallback(
          key, valid, message);
      if(typeof this.getValidationElementMap
          === "function") {
```

```
        var element = this.
            getValidationElementMap()[key];
        if(element) {
          if(valid) $(element).removeClass("
              validation-error");
          else $(element).addClass("validation
              -error");
        }
      }
    },
  };
});
```

*js-src/model/Area.js*

```
"use strict";
define(["underscore", "backbone"], function(_,
    Backbone) {
  return Backbone.Model.extend({
    urlRoot: "api.php/areas",
    defaults: {
      id: null,
      code: null,
      name: null,
      level: null,
    },
    idAttribute: "code",
    parse: function(r,o) {
      return {
        id: parseInt(r.id, 10),
        code: parseInt(r.code, 10),
        name: r.name,
        level: r.level,
      }
    },

    isNew: function() {
      return !this.has("id");
    },

    sync: function(method, model, options) {
      var base = this.urlRoot.replace(/[^\/]$
          /, "$&/");

      if(this.has("id") || method === "update"
          || method === "patch" || method
          === "delete") {
        // use id url when writing
        var id = this.get("id");
        options = options || {};
        options.url = options.url || base + "
            id/" + encodeURIComponent(id);
      }else if(method === "read") {
        // use code url when reading without
            id
        var code = this.get("code");
        options = options || {};
        options.url = options.url || base +
            encodeURIComponent(code);
      }
      Backbone.Model.prototype.sync.call(this,
          method, model, options);
    },
  }, {
    get_level: function(code) {
      code = ("0"+code).slice(-9);
      if(code.substr(6,3) !== "000") {
        return "barangay";
      }else if(code.substr(4,2) !== "00") {
        return "municipality";
      }else if(code.substr(2,2) !== "00") {
        return "province";
      }else{
        return "region";
      }
    },
  });
});
```

*js-src/model/AreaCollection.js*

```
"use strict";
define(["backbone", "model/Area", "backbone-
    pagec"], function(Backbone, Area) {
  return Backbone.PageableCollection.extend({
    model: Area,
    url: "api.php/areas",

    parse: function(data) {
      return data.data;
    },
  });
});
```

*js-src/model/AreaElectionCollection.js*

```
"use strict";
define(["jquery", "backbone", "model/Election
    "], function($, Backbone, Election) {
  return Backbone.Collection.extend({
    model: Election,
    initialize: function(models, options) {
      this.area = options.area;
      this.area_id = this.area.get("id");
      this.area_code = this.area.get("code");
    },
    url: function() {
      if(this.area_id)
        return "api.php/areas/id/" +
            encodeURIComponent(this.area_id)
            + "/elections";
      else
        return "api.php/areas/" +
            encodeURIComponent(this.area_code
            ) + "/elections";
    },
    parse: function(data) {
      return data.data;
    },
  });
});
```

*js-src/model/Datapoint.js*

```
"use strict";
define(["backbone"], function(Backbone) {
  return Backbone.Model.extend({
    defaults: {
      dataset_id: null,
      year: null,
      area_code: null,
      family_id: null,
      value: null,
    },
    parse: function(r,o) {
      return {
        id: r.id,
        dataset_id: r.dataset_id
          ? parseInt(r.dataset_id, 10)
          : (this.collection && this.
              collection.dataset
            ? parseInt(this.collection.dataset
                .id, 10)
            : null),
        year: parseInt(r.year, 10),
        area_code: parseInt(r.area_code, 10),
        family_id: r.family_id ? parseInt(r.
            family_id, 10) : null,
        value: r.value ? parseFloat(r.value) :
            null,
      };
    },
  });
});
```

*js-src/model/DatapointCollection.js*

```
"use strict";
define(["underscore", "backbone", "model/
    Datapoint"], function(_, Backbone,
    Datapoint) {
  return Backbone.Collection.extend({
    model: Datapoint,
    initialize: function(models, options) {
      this.dataset = options.dataset;
      this.year = options.year;
    },
    url: function() {
      return this.dataset.url() + "/datapoints
          ";
    },
    parse: function(data) {
      return data.data;
    },

    get_max_value: function() {
      if(this.max_value === undefined) {
        this.max_value = this.reduce(function(
            max,p){
          var value = p.get("value");
          if(value !== null && value > max) {
            return value;
          }else{
            return max;
          }
        }, -Infinity);
      }
      return this.max_value;
    },
    get_min_value: function() {
      if(this.min_value === undefined) {
        this.min_value = this.reduce(function(
            min,p){
          var value = p.get("value");
          if(value !== null && value < min) {
            return value;
          }else{
            return min;
          }
        }, Infinity);
      }
      return this.min_value;
    },

    fetch: function(options) {
      options = options || {};
      options.data = options.data || {}
      if(options.data.year === undefined)
          options.data.year = this.year;
      return Backbone.Collection.prototype.
          fetch.call(this, options);
    },
  });
});
```

*js-src/model/DatapointPageableCollection.js*

```
"use strict";
define(["backbone", "model/Datapoint", "
    backbone-pagec"], function(Backbone,
    Datapoint) {
  return Backbone.PageableCollection.extend({
    model: Datapoint,
    initialize: function(models, options) {
      this.dataset = options.dataset;
    },
    url: function() {
      return this.dataset.url() + "/datapoints
          ";
    },
    parse: function(data) {
      return data.data;
    },
  });
});
```

*js-src/model/Dataset.js*

```
"use strict";
define(["require", "backbone", "InstanceCache
    ", "model/DatapointCollection"], function
    (require, Backbone, InstanceCache,
    DatapointCollection) {
  return Backbone.Model.extend({
    urlRoot: function() {
      return "api.php/users/" + this.get("
          username") + "/datasets";
    },
    defaults: {
      username: null,
      type: null,
      name: null,
      description: null,
      min_year: null,
      max_year: null,
      contained_levels: null,
    },
    parse: function(r,o) {
      return {
        id: parseInt(r.id, 10),
        username: r.username,
        type: r.type,
        name: r.name,
        description: r.description,
        min_year: r.min_year ? parseInt(r.
            min_year, 10) : null,
        max_year: r.max_year ? parseInt(r.
            max_year, 10) : null,
        contained_levels: r.contained_levels,
      };
    },

    get_datapoints: function(year) {
      if(!InstanceCache) InstanceCache =
          require("InstanceCache");

      var name = "DatapointCollection";
      var key = this.id + "," + year;
      var datapoints = InstanceCache.
          get_existing(name, key);
      if(!datapoints) {
        datapoints = new DatapointCollection(
            null, {dataset: this, year: year
            });
        InstanceCache.set(name, key,
            datapoints);
      }

      return datapoints;
    },
  });
});
```

*js-src/model/DatasetCollection.js*

```
"use strict";
define(["backbone", "model/Dataset", "backbone
    -pagec"], function(Backbone, Dataset) {
  return Backbone.PageableCollection.extend({
    model: Dataset,
    initialize: function(models, options) {
      if(options && options.username) {
        this.username = options.username;
        this.type = options.type || null;
      }
    },
    url: function() {
      if(this.username) {
        return "api.php/users/" + this.
            username + "/datasets";
      }else{
        return "api.php/datasets";
      }
    },
    parse: function(data) {
      return data.data;
    },

    fetch: function(options) {
      options = options || {};
      if(this.type) {
        options.data = options.data || {}
        options.data.type = options.data.type
            || this.type;
      }
      Backbone.PageableCollection.prototype.
          fetch.call(this, options);
    },
  });
});
```

*js-src/model/Election.js*

```
"use strict";
define(["backbone"], function(Backbone) {
  return Backbone.Model.extend({
    defaults: {
      official_id: null,
      year: null,
      year_end: null,
      position: null,
      votes: null,
      area_code: null,
      party_id: null,
    },
    parse: function(r,o) {
      return {
        id: parseInt(r.id, 10),
        official_id: parseInt(r.official_id,
            10),
        year: parseInt(r.year, 10),
        year_end: parseInt(r.year_end, 10),
        position: r.position,
        votes: r.votes ? parseInt(r.votes, 10)
            : null,
        area_code: parseInt(r.area_code, 10),
        party_id: r.party_id ? parseInt(r.
            party_id, 10) : null,
      };
    },
  });
});
```

*js-src/model/ElectionCollection.js*

```
"use strict";
define(["backbone", "model/Election", "
    backbone-pagec"], function(Backbone,
    Election) {
  return Backbone.PageableCollection.extend({
    model: Election,
    url: "api.php/elections",
    parse: function(data) {
      return data.data;
    },
  });
});
```

*js-src/model/ElectionSingle.js*

```
"use strict";
define(["model/Election"], function(Election)
    {
  return Election.extend({urlRoot: "api.php/
      elections"});
});
```

*js-src/model/Family.js*

```
"use strict";
define(["backbone"], function(Backbone) {
  return Backbone.Model.extend({
    defaults: {
      name: null,
    },
    parse: function(r,o) {
      return {
        id: parseInt(r.id, 10),
        name: r.name,
      };
    },
  });
});
```

*js-src/model/FamilyCollection.js*

```
"use strict";
define(["backbone", "model/Family", "backbone-
    pagec"], function(Backbone, Family) {
  return Backbone.PageableCollection.extend({
    model: Family,
    url: "api.php/families",
    parse: function(data) {
      return data.data;
    },
  });
});
```

*js-src/model/FamilyMemberCollection.js*

```
"use strict";
define(["jquery", "backbone", "model/Official
    "], function($, Backbone, Official) {
  return Backbone.Collection.extend({
    model: Official,
    initialize: function(models, options) {
      this.family_id = options.family_id;
    },
    url: function() {
      return "api.php/families/" +
          encodeURIComponent(this.family_id)
          + "/officials";
    },
    parse: function(data) {
      return data.data;
    },
    post_member: function(official) {
      var that = this;
      $.ajax({
        method: "POST",
        url: that.url(),
        data: JSON.stringify(official),
        processData: false,
        dataType: "json",
        success: function() {
          that.add(official);
        },
        error: function() {
          console.error("Error post_member");
        },
      });
    },
  });
});
```

*js-src/model/FamilySingle.js*

```
"use strict";
define(["model/Family"], function(Family) {
  return Family.extend({urlRoot: "api.php/
      families"});
});
```

*js-src/model/Official.js*

```
"use strict";
define(["backbone"], function(Backbone) {
  return Backbone.Model.extend({
    defaults: {
      surname: null,
      name: null,
      nickname: null,
    },
    parse: function(r,o) {
      return {
        id: parseInt(r.id, 10),
        surname: r.surname,
        name: r.name,
        nickname: r.nickname,
      };
    },
```

```
    get_full_name: function() {
      if(!this.get("surname") || !this.get("
          name")) return null;

      var nickname = null;
      if(this.get("nickname")) {
        nickname = ' "' + this.get("nickname")
            + '"';
      }else{
        nickname = "";
      }
      return this.get("surname") + ", " + this
          .get("name") + nickname;
    },
  });
});
```

*js-src/model/OfficialCollection.js*

```
"use strict";
define(["backbone", "model/Official", "
    backbone-pagec"], function(Backbone,
    Official) {
  return Backbone.PageableCollection.extend({
    model: Official,
    url: "api.php/officials",
    parse: function(data) {
      return data.data;
    },
  });
});
```

*js-src/model/OfficialFamilyCollection.js*

```
"use strict";
define(["jquery", "backbone", "model/Family"],
     function($, Backbone, Family) {
  return Backbone.Collection.extend({
    model: Family,
    initialize: function(models, options) {
      this.official_id = options.official_id;
    },
    url: function() {
      return "api.php/officials/" +
          encodeURIComponent(this.official_id
          ) + "/families";
    },
    parse: function(data) {
      return data.data;
    },
    post_family: function(family) {
      var that = this;
      $.ajax({
        method: "POST",
        url: that.url(),
        data: JSON.stringify(family),
        processData: false,
        dataType: "json",
        success: function() {
          that.add(family);
        },
        error: function() {
          console.error("Error post_family");
        },
      });
    },
  });
});
```

*js-src/model/OfficialSingle.js*

```
"use strict";
define(["model/Official"], function(Official)
    {
  return Official.extend({urlRoot: "api.php/
      officials"});
});
```

*js-src/model/Party.js*

```
"use strict";
define(["backbone"], function(Backbone) {
  return Backbone.Model.extend({
    urlRoot: "api.php/parties",
    defaults: {
      name: null,
    },
    parse: function(r,o) {
      return {
        id: parseInt(r.id, 10),
        name: r.name,
      };
    },
  });
});
```

*js-src/model/PartyCollection.js*

```
"use strict";
define(["backbone", "model/Party", "backbone-
    pagec"], function(Backbone, Party) {
  return Backbone.PageableCollection.extend({
    model: Party,
    url: "api.php/parties",
    parse: function(data) {
      return data.data;
    },
  });
});
```

*js-src/model/Token.js*

```
"use strict";
define(["jquery", "localStorage", "backbone",
    "InstanceCache", "model/User"], function(
    $, localStorage, Backbone, InstanceCache,
     User) {
  return Backbone.Model.extend({
    urlRoot: "api.php/tokens",
    defaults: {
      username: null,
      token: null,
      expiry: null,
    },

    initialize: function() {
      var that = this;

      var cookie = localStorage.getItem("
          dynavis_token");
      if(cookie) {
        var data = JSON.parse(cookie);
        this.set(data);
        $.ajaxSetup({
          headers: {"Authorization": 'Token
              token="' + data.token + '"'}
        });
        this.fetch({
          error: function() {
            $.ajaxSetup();
            that.clear();
            localStorage.removeItem("
                dynavis_token");
          },
        });
      }
    },

    get_user: function() {
      var username = this.get("username");
      if(!username || this.isNew()) return
          null;

      var user = InstanceCache.get_existing("
          User", username);
      if(!user) {
```

```
        user = InstanceCache.get("User",
            username);
        user.fetch({
          success: function() {
            this.trigger("change", this);
          }.bind(this),
        });
      }
      return user;
    },

    get_user_role: function() {
      return this.get_user() ? this.get_user()
          .get("role") : null;
    },

    login: function(username, password,
        success, error) {
      var that = this;
      $.ajax({
        method: "POST",
        url: that.urlRoot,
        data: JSON.stringify({username:
            username, password: password}),
        processData: false,
        dataType: "json",
        success: function(data) {
          $.ajaxSetup({
            headers: {"Authorization": 'Token
                token="' + data.token + '"'}
          });
          that.set(data);
          that.fetch();
          localStorage.setItem("dynavis_token
              ", JSON.stringify(data), data.
              expiry);
          if(success) success(data);
        },
        error: function(xhr) {
          if(error) error(xhr);
        },
      });
    },

    logout: function() {
      var that = this;
      $.ajax({
        method: "DELETE",
        url: that.url(),
        success: function() {
          $.ajaxSetup();
          that.clear();
          localStorage.removeItem("
              dynavis_token");
        },
        error: function() {
          console.error("Error logout");
          that.fetch({
            error: function(m,r,o) {
              if(r.status == 404) {
                $.ajaxSetup();
                that.clear();
                localStorage.removeItem("
                    dynavis_token");
              }
            },
          });
        },
      });
    },
  });
});
```

*js-src/model/User.js*

```
"use strict";
define(["backbone"], function(Backbone) {
  return Backbone.Model.extend({
    urlRoot: "api.php/users",
    defaults: {
      active: null,
      username: null,
      role: null,
    },
    idAttribute: "username",

    parse: function(r,o) {
      return {
        active: r.active != "0",
        username: r.username,
        role: r.role,
      };
    },
  });
});
```

*js-src/model/UserCollection.js*

```
"use strict";
define(["backbone", "model/User", "backbone-
    pagec"], function(Backbone, User) {
  return Backbone.PageableCollection.extend({
    model: User,
    url: "api.php/users",

    parse: function(data) {
      return data.data;
    },
  });
});
```

*js-src/view/AlphabetIndex.jsx*

```
"use strict";
define(["react", "react.backbone"], function(
    React) {
  return React.createBackboneClass({
    render: function() {
      var that = this;
      return (
        <div>
          <button className="button" onClick={
              this.props.getCallback(null)}>
              All</button>
          {_.map("ABCDEFGHIJKLMNOPQRSTUVWXYZ".
              split(""), function(letter) {
            return <button className="button"
                key={letter} onClick={that.
                props.getCallback(letter)}>{
                letter}</button>
          })}
        </div>
      );
    },
  });
});
```

*js-src/view/CollectionCount.jsx*

```
"use strict";
define(["react", "react.backbone"], function(
    React) {
  return React.createBackboneClass({
    render: function() {
      return (
        <span className={this.props.className
            }>{this.collection().size()}</
            span>
      );
    },
  });
});
```

*js-src/view/ConfirmationDialog.jsx*

```
"use strict";
define(["underscore", "jquery", "react", "jsx!
    view/Modal", "bootstrap"], function(_, $,
     React, Modal) {
  var ConfirmationDialog = React.createClass({
    statics: {
      open: function(message, actions) {
        var node = document.createElement("div
            ");
        document.body.appendChild(node);
        var dialog = React.createElement(
            ConfirmationDialog,{
```

```
            message: message,
            actions: actions,
            onClose: function() {
              React.unmountComponentAtNode(node)
                  ;
              document.body.removeChild(node);
            },
          }, null);
          return React.render(dialog, node);
        },
      },

      componentDidMount: function() {
        var $this = $(React.findDOMNode(this));
        $this.modal();
        $this.on("hidden.bs.modal", function() {
          this.props.onClose();
        }.bind(this));
      },

      render: function() {
        var that = this;
        return (
          <div className="modal fade">
            <div className="modal-dialog modal-
                sm">
              <div className="modal-content">
                <div className="modal-body">
                  <div className="text pad">
                    {this.props.message}
                  </div>
                  <div className="clearfix">
                    {_.chain(this.props.actions)
                      .map(function(action, i) {
                        var className = "mar
                            button button-flat
                            ";
                        className += i == 0 ? "
                            pull-left" : " pull
                            -right";

                        if(action.type === "
                            close") {
                          // className += "
                              button-flat";
                        }else if(action.type ===
                            "primary") {
                          className += " button-
                              primary";
                        }else if(action.type ===
                            "secondary") {
                          className += " button-
                              complement";
                        }
                        var original_callback =
                            action.callback;
                        action.callback =
                            function() {
                          if(original_callback)
                              original_callback
                              ();
                          that.close();
                        };

                        return <button className
                            ={className}
                            onClick={action.
                            callback}>{action.
                            display}</button>;
                      })
                      .reverse()
                      .value()}
                  </div>
                </div>
              </div>
            </div>
          );
        },

        close: function() {
          $(React.findDOMNode(this)).modal("hide")
              ;
        },
      });
      return ConfirmationDialog;
    });
```

*js-src/view/EditableName.jsx*

```
"use strict";
define(["react", "react.backbone"], function(
    React) {
  return React.createBackboneClass({
    mixins: [React.addons.LinkedStateMixin],

    getInitialState: function() {
      return {
        name: this.model().get("name"),
      };
    },

    render: function() {
      return (
        <input type="text" valueLink={this.
            linkState("name")} {...this.props
            } />
      );
    },
  });
});
```

*js-src/view/EditableOfficialName.jsx*

```
"use strict";
define(["react", "react.backbone"], function(
    React) {
  return React.createBackboneClass({
    mixins: [React.addons.LinkedStateMixin],

    getInitialState: function() {
      return {
        surname: this.model().get("surname"),
        name: this.model().get("name"),
        nickname: this.model().get("nickname")
            ,
      };
    },

    render: function() {
      var nickname = this.state.nickname;
      return (
        <span className={this.props.className
            }>
          <label className="pure-u-1 pad">
            <div className="label">Surname</
                div>
            <input ref="surname" className="
                pure-u-1" type="text"
                valueLink={this.linkState("
                surname")} required autoFocus
                ={this.props.autoFocus} />
          </label>
          <label className="pure-u-1 pad">
            <div className="label">Name</div>
            <input ref="name" className="pure-
                u-1" type="text" valueLink={
                this.linkState("name")}
                required />
          </label>
          <label className="pure-u-1 pad">
            <div className="label">Nickname</
                div>
            <input ref="nickname" className="
                pure-u-1" type="text"
                valueLink={this.linkState("
                nickname")} />
          </label>
        </span>
      );
    },

    get_input_surname: function() { return
        this.refs.surname; },
    get_input_name: function() { return this.
        refs.name; },
    get_input_nickname: function() { return
        this.refs.nickname; },
  });
});
```

*js-src/view/FileInput.jsx*

```
"use strict";
define(["react"], function(React) {
  return React.createClass({
    getInitialState: function() {
      return {
        filename: null
      };
    },

    render: function() {
      return (
        <span className="file-wrapper">
          <label className="group">
            <span className="group-component
                input" title={this.state.
                filename} readOnly>
              <div className="name">{this.
                  state.filename}</div>
            </span>
            <span className="group-component
                button">Browse</span>
            <input ref="file" type="file"
                onChange={this.handle_change}
                {...this.props} />
          </label>
        </span>
      );
    },

    handle_change: function() {
      var filename = React.findDOMNode(this.
          refs.file).value.replace("C:\\
          fakepath\\","");
      this.setState({filename: filename});
    },

    get_input: function() {
      return React.findDOMNode(this.refs.file)
          ;
    },

    get_filename: function() {
      return this.state.filename;
    },
  });
});
```

*js-src/view/Header.jsx*

```
"use strict";
define(["react", "InstanceCache", "model/
    Dataset", "jsx!view/HeaderSession"],
    function(React, InstanceCache, Dataset,
    HeaderSession) {
  return React.createClass({
    componentDidMount: function() {
      this.props.bus.router.on("route", this.
          on_route);
    },

    componentWillUnmount: function() {
      this.props.bus.router.off("route", this.
          on_route);
    },

    getInitialState: function() {
      return {
        title: this.props.title,
        subtitle: null,
      };
    },

    render: function() {
      document.title = (this.state.title ? (
          this.state.title + (this.state.
          subtitle ? (" / " + this.state.
          subtitle) : "") + " - ") : "") + "
          DynastyMap";
      if(this.state.subtitle) {
        var subtitle = <h3 className="header-
            subtitle">{" / " + this.state.
            subtitle}</h3>
      }
      var token = InstanceCache.get_existing("
          Token", "session");
      return (
        <div className="clearfix">
          <div id="header-logo">
            <a href="." className="no-decor">
              <div className="logo"></div>
              <div className="logo-type">
                  DynastyMap</div>
            </a>
          </div>
          <div id="header-content">
            <h2 className="header-title">{this
                .state.title}</h2>
            {subtitle}
          </div>
          <HeaderSession model={token}/>
        </div>
      );
    },

    set_title: function(title) {
      this.setState({title: title, subtitle:
          null});
    },

    set_subtitle: function(subtitle) {
      this.setState({subtitle: subtitle});
    },

    on_route: function(e) {
      var title_map = {
        "home": "Admin Home",
        "elections": "Elections",
        "officials": "Officials",
        "families": "Families",
        "areas": "Areas",
        "datasets": "Datasets",
        "datapoints": "Dataset",
        "users": "Users",
      };
      this.set_title(title_map[e.route]);

      if(e.route === "datasets" && e.params
          [0]) {
        this.set_subtitle("uploaded by " + e.
            params[0]);
      }
      if(e.route === "datapoints" && e.params
          [0] && e.params[1]) {
        var id = parseInt(e.params[1], 10);
        var dataset = InstanceCache.
            get_existing("Dataset", id)
          || new Dataset({id: id, username: e.
              params[0]});
        dataset.fetch({
          success: function(model) {
            this.set_subtitle(model.get("name
                "));
          }.bind(this)
        });
      }
    },
  });
});
```

*js-src/view/HeaderSession.jsx*

```
"use strict";
define(["jquery", "react", "InstanceCache", "
    react.backbone"], function($, React,
    InstanceCache) {
  return React.createBackboneClass({
    mixins: [React.addons.LinkedStateMixin],

    render: function() {
      var user = this.model() ? this.model().
          get_user() : null;
      if(user) {
        if(!window.location.pathname.match
            (/^.*?\/admin(\.html)?((\?|#).*)?
            $/) && this.model().get_user_role
            () == "admin") {
          var admin_link = <a className="
              button button-flat mar" href="
              admin">Admin</a>;
        }
        if(window.location.pathname.match
            (/^.*?\/app(\.html)?((\?|#).*)?$
            /)) {
          var datasets_link = <a className="
              button button-flat" href="#
              datasets">Datasets</a>;
        }
        return (
          <div className="session">
            <div className="hider">
              <div className="hider-handle
                  hider-handle-toggle">
                <span className="login-text">
                  <i className="fa fa-caret-
                      right fa-fw"/>
```

```
                <i className="fa fa-user fa-
                    fw"/>
                <span className="login-
                    username">{user.get("
                    username")}</span>
              </span>
            </div>
            <div className="hider-content">
              <span className="login-text">
                {datasets_link}
                {admin_link}
                 
              </span>
              <button className="button
                  button-flat" onClick={
                  this.handle_logout}>Sign
                  Out</button>
              <i className="fa fa-caret-left
                   fa-fw"/>
            </div>
          </div>
        </div>
      );
    }else{
      var n = encodeURIComponent(window.
          location.pathname);
      return (
        <div className="session">
          <span className="login-text">
            <a className="button button-flat
                 mar" href={"login?n=" + n
                }>Create Account</a>
            <a className="button mar" href
                ={"login?n=" + n}>Sign In</
                a>
          </span>
        </div>
      );
    }
  },

  handle_logout: function(e) {
    this.model().logout();
  },
 });
});
```

*js-src/view/IndexedPageControls.jsx*

```
"use strict";
define(["react", "jsx!view/AlphabetIndex", "
    jsx!view/PageControls", "react.backbone
    "], function(React, AlphabetIndex,
    PageControls) {
  return React.createBackboneClass({
    getInitialState: function() {
      return {letter: null};
    },

    render: function() {
      return(
        <div>
          <AlphabetIndex
            getCallback={this.index_handler}
                />
          <PageControls
            collection={this.collection()}
            onPrev={this.handle_prev}
            onNext={this.handle_next} />
        </div>
      );
    },

    set_letter: function(letter, options) {
      this.collection().setParams(this.
          get_data(letter));
      this.collection().page(0);
      this.setState({letter: letter});
    },

    index_handler: function(letter) {
      var that = this;
      return function() {
        that.set_letter(letter);
      };
    },

    handle_prev: function() {
      this.collection().prev();
    },
    handle_next: function() {
      this.collection().next();
    },

    get_data: function(letter) {
      if(letter === undefined) letter = this.
          state.letter;
      return letter ? {q: letter, qindex: 1} :
          null;
    },
  });
});
```

*js-src/view/Modal.jsx*

```
"use strict";
define(["jquery", "react", "bootstrap"],
    function($, React) {
  var Modal = React.createClass({
    statics: {
      open: function(title, children) {
        var node = document.createElement("div
            ");
        document.body.appendChild(node);
        var modal = React.createElement(Modal
            ,{
          title: title,
          onClose: function() {
            React.unmountComponentAtNode(node)
                ;
            document.body.removeChild(node);
          },
        }, null, children);
        return React.render(modal, node);
      },
    },

    componentDidMount: function() {
      var $this = $(React.findDOMNode(this));
      $this.modal();
      $this.on("hidden.bs.modal", function() {
        this.props.onClose();
      }.bind(this));
    },

    render: function() {
      return (
        <div className="modal fade">
          <div className="modal-dialog">
            <div className="modal-content">
              <div className="modal-header">
                <button type="button"
                    className="pull-right
                    button button-flat button
                    -close" onClick={this.
                    handle_close}>&times;</
                    button>
                <h4 className="modal-title">{
                    this.props.title}</h4>
              </div>
              <div className="modal-body">
                {this.props.children}
              </div>
            </div>
          </div>
        </div>
      );
    },

    handle_close: function() {
      this.close();
    },

    close: function() {
      $(React.findDOMNode(this)).modal("hide")
          ;
    },
  });
  return Modal;
});
```

*js-src/view/Name.jsx*

```
"use strict";
define(["react", "react.backbone"], function(
    React) {
  return React.createBackboneClass({
    render: function() {
      return (
        <span className={this.props.className
            }>{this.model() ? this.model().
            get("name") : ""}</span>
      );
    },
  });
});
```

*js-src/view/Notification.jsx*

```
"use strict";
define(["underscore", "jquery", "react"],
    function(_, $, React) {
  var NotificationsContainer = React.
      createClass({
    render: function() {
      return (
        <div className="notifications-
            container">
          {this.props.notifications.map(
              function(notif) {
            return (
              <div key={notif.key} className
                  ={"notification
                  notification-" + notif.type
                  } onClick={notif.close.bind
                  (notif)}>
                {notif.contents}
              </div>
            );
          })}
        </div>
      );
    },
  });

  var NotificationManager = {
    counter: 0,
    notifications: [],
  };

  var Notification = function(key, contents,
      type) {
    this.key = key;
    this.contents = contents;
    this.type = type;
    this.timer = null;
    this.active = false;
  };
  Notification.prototype.close = function() {
    NotificationManager.close(this);
  };
  Notification.prototype.set = function(
      contents, type) {
    this.contents = contents;
    if(type) this.type = type;
    NotificationManager.update();
  };

  NotificationManager.open = function(contents
      , timeout, type) {
    var notif = new Notification("n" + (this.
        counter++), contents, type);
    notif.active = true;
    this.notifications.push(notif);
    this.update();

    if(timeout !== 0) {
      notif.timer = setTimeout(function() {
        if(notif.active) this.close(notif);
      }.bind(this), timeout || 9000);
    }

    return notif;
  };
  NotificationManager.replace = function(notif
      , contents, timeout, type) {
    if(notif.active) {
      if(contents) notif.contents = contents;
      if(type) notif.type = type;
      if(notif.timer) clearTimeout(notif.timer
          );
      if(timeout !== 0) {
        notif.timer = setTimeout(function() {
          if(notif.active) this.close(notif);
        }.bind(this), timeout || 9000);
      }
      this.update();
      return notif;
    }else{
      return this.open(contents, timeout, type
          );
    }
  };
  NotificationManager.close = function(notif)
      {
    notif.active = false;
    this.notifications = _.reject(this.
        notifications, function(n) { return n
        .key == notif.key; });
    this.update();
  };
  NotificationManager.update = function() {
    var node = document.getElementById("
        notifications-container");
    if(!node){
      node = document.createElement("div");
      node.id = "notifications-container";
      document.body.appendChild(node);
    }
    React.render(<NotificationsContainer
        notifications={this.notifications}/>,
        node);
  };

  return NotificationManager;
});
```

*js-src/view/OfficialName.jsx*

```
"use strict";
define(["react", "react.backbone"], function(
    React) {
  return React.createBackboneClass({
    render: function() {
      if(this.model()) {
        if(this.model().has("surname"))
          var surname = <span className="
              surname">{this.model().get("
              surname") + ", "}</span>;
        if(this.model().has("name"))
          var name = <span className="name">{
              this.model().get("name")}</span
              >;
        if(this.model().has("nickname"))
          var nickname = <span className="
              nickname">{' "' + this.model().
              get("nickname") + '"'}</span>;
      }
      return (
        <span className={this.props.className
            }>{surname}{name}{nickname}</span
            >
      );
    },
  });
});
```

*js-src/view/PageControls.jsx*

```
"use strict";
define(["react", "react.backbone"], function(
    React) {
  return React.createBackboneClass({
    getInitialState: function() {
      return {input: 1};
    },

    componentWillMount: function() {
      this.collection().on("sync", this.
          on_load_page);
    },
    componentDidUnmount: function() {
      this.collection().off("sync", this.
          on_load_page);
    },

    render: function() {
      var page = this.collection().getPage();
      var pages = this.collection().
          getTotalPages();
      return (
```

```
        <form className={this.props.className}
              onSubmit={this.handle_submit}>
          <button className="button"
            type="button"
            title="Previous page"
            onClick={this.handle_prev}
            disabled={page <= 0}>
            <i className="fa fa-caret-left"/>
          </button>
          <span className="pad">Page <input
               className="input" type="text"
               size="3" value={this.state.
               input} onChange={this.
               handle_change} /> of {pages}</
               span>
          <button className="button"
            type="button"
            title="Next page"
            onClick={this.handle_next}
            disabled={page + 1 >= pages}>
            <i className="fa fa-caret-right"/>
          </button>
          <input className="hidden" type="
               submit" value="Go to page" />
        </form>
      );
    },

    on_load_page: function() {
      this.setState({input: this.collection().
          getPage() + 1});
    },

    handle_change: function(e) {
      this.setState({input: e.target.value});
    },

    handle_submit: function(e) {
      e.preventDefault();
      var new_page = parseInt(this.state.input
          , 10);
      if(isNaN(new_page)) {
        this.setState({input: this.collection
            ().getPage() + 1});
      }else{
        var pages = this.collection().
            getTotalPages();
        if(new_page < 1) {
          new_page = 1;
          this.setState({input: new_page});
        }else if(new_page > pages){
          new_page = pages;
          this.setState({input: new_page});
        }
        this.collection().page(new_page - 1);
      }
    },

    handle_prev: function() {
      if(this.props.onPrev) {
        this.props.onPrev();
      }
      this.collection().prev({success: this.
          on_load_page});
    },

    handle_next: function() {
      if(this.props.onNext) {
        this.props.onNext();
      }
      this.collection().next({success: this.
          on_load_page});
    },
  });
});
```

*js-src/view/PanelToolbar.jsx*

```
"use strict";
define(["jquery", "react"], function($, React)
    {
  return React.createClass({
    componentDidMount: function() {

    },

    render: function() {
      return (
        <div className="panel-toolbar panel-
            toolbar-closed">
          <div className="panel-toolbar-toggle
              ">
            <button className="pull-right
                button button-primary mar"
                onClick={this.toggle}>
              {this.props.toggle_text}
            </button>
          </div>
          <div className="panel-toolbar-
              contents">
            <div className="pure-u-1">
              <button className="pull-right
                  button button-flat button-
                  close mar" onClick={this.
                  toggle}>
                &times;
              </button>
            </div>
            <div className="pure-g">
              {this.props.children}
            </div>
          </div>
        </div>
      );
    },

    toggle: function() {
      $(React.findDOMNode(this)).toggleClass("
          panel-toolbar-closed");
    },
    open: function() {
      $(React.findDOMNode(this)).removeClass("
          panel-toolbar-closed");
    },
    close: function() {
      $(React.findDOMNode(this)).addClass("
          panel-toolbar-closed");
    },
  });
});
```

*js-src/view/SearchControls.jsx*

```
"use strict";
define(["react", "jsx!view/PageControls", "
    react.backbone"], function(React,
    PageControls) {
  return React.createBackboneClass({
    mixins: [React.addons.LinkedStateMixin],

    getInitialState: function() {
      return {
        input: null,
        query: null,
      };
    },

    render: function() {
      var action_button = null;
      if(this.state.query) {
        action_button = <button type="reset"
            onClick={this.handle_cancel}><i
            className="fa fa-close"/></button
            >;
      }else{
        action_button = <button type="submit
            "><i className="fa fa-search"/></
            button>;
      }
      return(
        <div className={this.props.className}>
          <div className="clearfix">
            <form className="search-bar pure-g
                pull-left" onSubmit={this.
                handle_search}>
              <input type="text" placeholder={
                  this.state.query} valueLink
                  ={this.linkState("input")}
                  />
              {action_button}
            </form>
            <PageControls className="pull-
                right" collection={this.
                collection()} />
          </div>
        </div>
      );
    },

    set_query: function(query, options) {
      this.collection().setParams(this.
          get_data(query));
      this.collection().page(0, options);
      this.setState({query: query});
    },
```

```
    handle_cancel: function() {
      this.setState({input: ""});
      this.set_query(null);
    },

    handle_search: function(e) {
      e.preventDefault();
      this.set_query(this.state.input);
    },

    get_data: function(query) {
      if(query === undefined) query = this.
          state.query;
      return query ? {q: query} : null;
    },
  });
});
```

*js-src/view/SliderTransitionGroupChild.jsx*

```
"use strict";
define(["jquery", "react", "jquery.bez"],
    function($, React) {
  var ease_swift = $.bez([0.2, 0, 0.1, 1]);
  return React.createClass({
    componentWillAppear: function(done) {
      done();
    },

    componentWillEnter: function(done) {
      var $el = $(React.findDOMNode(this));
      var $buttons = $el.find("input[type=
          button],input[type=submit],input[
          type=reset],button");
      $buttons.prop("disabled", true);
      $el.hide().animate({
        height: "toggle", opacity: "toggle",
      }, 600, ease_swift, function(){
        $buttons.prop("disabled", false);
        done();
      });
    },

    componentWillLeave: function(done) {
      var $el = $(React.findDOMNode(this));
      $el.find("input[type=button],input[type=
          submit],input[type=reset],button").
          prop("disabled", true);
      $el.animate({
        height: 0, opacity: 0,
      }, 600, ease_swift, done);
    },

    render: function() {
      return <div>{this.props.children}</div>;
    },
  });
});
```

*js-src/view/Spinner.jsx*

```
"use strict";
define(["react"], function(React) {
  return React.createClass({
    // http://codepen.io/CSS3fx/pen/EfByo
    render: function() {
      // var size = this.props.size || 1;
      var size = 1;
      return (
        <svg className={"spinner-circular " +
            this.props.className} height={50
            * size} width={50 * size} styl={
            this.props.style}>
          <circle className="spinner-path" cx
              ={25 * size} cy={25 * size} r
              ={19.9 * size} fill="none"
              strokeWidth={6 * size}
              strokeMiterlimit={10} />
        </svg>
      );
    },
  });
});
```

*js-src/view/Toggle.jsx*

```
"use strict";
define(["jquery", "react"], function($, React)
    {
  return React.createClass({
    render: function() {
      return (
        <span className="toggle" title={this.
            props.title}>
          <label>
            <input className="toggle-checkbox"
                type="checkbox" {...this.
                props} />
            <span className="toggle-lever"></
                span>
          </label>
        </span>
      );
    },
  });
});
```

*js-src/view/TypeaheadInput.jsx*

```
"use strict";
define(["jquery", "react", "InstanceCache", "
    typeahead", "react.backbone"], function($
    , React, InstanceCache) {
  return React.createBackboneClass({
    getInitialState: function() {
      return {
        value: "",
        selected: null,
      };
    },

    onModelChange: function() {
      if(this.model() && this.state.selected
          === this.model()) {
        this.setState({value: this.props.
            display(this.model().toJSON())});
      }
    },

    componentWillMount: function() {
      if(this.model()) {
        this.setState({
          value: this.props.display(this.model
              ().toJSON()),
          selected: this.model(),
        });
      }else{
        this.setState(this.getInitialState());
      }
    },

    componentDidMount: function() {
      var that = this;

      var $input = $(React.findDOMNode(this.
          refs.input));
      $input.typeahead({highlight: true}, {
        limit: 6,
        highlight: true,
        display: that.props.display,
        source: function(q, sync, async) {
          var params = {
            string: q,
            limit: this.limit,
          };
          InstanceCache.search(that.props.for,
              params,
            function(d) { sync(that.
                filter_search(d)); },
            function(d) { async(that.
                filter_search(d)); });
        },
      });
      $input.bind("typeahead:select typeahead:
          autocomplete", function(e, s) {
        that.handle_select(s);
      });

      if(this.props.autoFocus) $input.focus();
    },
```

```
    filter_search: function(data) {
      var that = this;
      if(this.collection()) {
        return _.filter(data, function(item) {
          return that.collection().get(item.id
              ) == null;
        });
      }else{
        return data;
      }
    },

    render: function() {
      return (
        <input className={this.props.className
            } type="text" ref="input" value={
            this.state.value} onChange={this.
            handle_change} placeholder={this.
            props.placeholder} required={this
            .props.required} autoFocus={this.
            props.autoFocus} />
      );
    },

    handle_change: function(e) {
      this.setState({value: e.target.value,
          selected: null});
    },

    handle_select: function(item) {
      this.setState({value: this.props.display
          (item), selected: item});
    },

    get_or_create: function(options) {
      var that = this;
      if(this.state.selected) {
        options.callback(this.state.selected,
            false);
      }else if(this.state.value) {
        var attributes = options.attributes(
            this.state.value);
        if(!attributes) return;

        var normalizer = function(val) {
          return (typeof val === "string") ?
              val.toUpperCase() : val;
        };

        if(!options.search) {
          options.search = _.keys(attributes);
        }
        var match = _.mapObject(_.pick(
            attributes, options.search),
            normalizer);

        // Callback party!! Woo!

        var callback = function(data) {
          var found = _.find(data, function(o)
              {
            return _.isMatch(_.mapObject(o,
                normalizer), match);
          });

          if(found) {
            options.callback(found);
          }else{
            var model = new options.model(
                attributes);
            model.save(null, {
              patch: true,
              wait: true,
              success: function(model) {
                options.callback(model, true);
              },
              error: function() {
                console.error("Error model.
                    save");
              },
            });
          }
        };

        var query = _.values(match).join(" ");
        InstanceCache.search(this.props.for, {
            string: query, limit:1}, callback
            );
      }else{
        options.callback(null, false);
      }
    },

    clear: function() {
      var $input = $(React.findDOMNode(this.
          refs.input));
      $input.typeahead("val", "");
      this.setState(this.getInitialState());
    },
  });
});
```

*js-src/view/Username.jsx*

```
"use strict";
define(["react", "react.backbone"], function(
    React) {
  return React.createBackboneClass({
    render: function() {
      return (
        <span>{this.model() ? this.model().get
            ("username") : ""}</span>
      );
    },
  });
});
```

*js-src/view/ValidationMessages.jsx*

```
"use strict";
define(["underscore", "react"], function(_,
    React) {
  return React.createClass({
    render: function() {
      if(this.props.validation.valid) return <
          div></div>;
      return (
        <div className="validation-messages">
          {_.map(_.pairs(this.props.validation
              .messages), function(pair) {
            return <div className="text">{pair
                [0] + " " + pair[1]}</div>;
          })}
        </div>
      );
    },
  });
});
```

*js-src/view/admin/AreaRow.jsx*

```
"use strict";
define(function(require) {
  var React = require("react", "react.backbone
      "),
    SliderTransitionGroupChild = require("jsx!
        view/SliderTransitionGroupChild"),
    EditableName = require("jsx!view/
        EditableName"),
    TypeaheadInput = require("jsx!view/
        TypeaheadInput"),
    Name = require("jsx!view/Name"),
    ClickToTopMixin = require("mixin/
        ClickToTopMixin"),
    Va = require("validator"),
    ValidationMixin = require("mixin/
        ValidationMixin"),
    ValidationMessages = require("jsx!view/
        ValidationMessages"),
    Notification = require("jsx!view/
        Notification"),
    ConfirmationDialog = require("jsx!view/
        ConfirmationDialog"),
    ReactTransitionGroup = React.addons.
        TransitionGroup;

  return React.createBackboneClass({
    mixins: [React.addons.LinkedStateMixin,
        ValidationMixin, ClickToTopMixin],

    getInitialState: function() {
      return {
        edit: this.model().isNew(),
        code: this.format_code(this.model().
            get("code")),
        level: this.model().get("level") || "
            barangay",
      };
```

```
},

getValidationSchema: function() {
  return {
    name: Va.lidator().required().string()
        ,
    code: Va.lidator().required().length
        (9).integerish(),
    level: Va.lidator().required(),
  };
},
getObjectToValidate: function() {
  return {
    name: this.refs.name.state.name,
    code: this.state.code,
    level: this.state.level,
  };
},
getValidationElementMap: function() {
  return {
    name: React.findDOMNode(this.refs.name
        ),
    code: React.findDOMNode(this.refs.code
        ),
    level: React.findDOMNode(this.refs.
        level),
  };
},
validationCallback: function(key, valid,
    message) {
  // React.findDOMNode(this.refs.save).
      disabled = !valid;
},

render: function() {
  var display = function(item) {
    return item.name;
  };

  var fields = null;
  if(this.model().isNew() || this.state.
      edit) {
    if(!this.model().isNew()){
      var cancel_button = <button
          className="pull-right button
          mar" onClick={this.
          handle_cancel}>Cancel</button>;
    }
    return (
      <div className="edit data-row form">
      <ReactTransitionGroup><
          SliderTransitionGroupChild key
          ="edit">
        <ValidationMessages validation={
            this.state.validation} />
        <div className="pure-g">
          <label className="pure-u-1-2 pad
              ">
            <div className="label">Name</
                div>
            <EditableName className="pure-
                u-1" ref="name" model={
                this.model()} autoFocus
                />
          </label>
          <label className="pure-u-1-2 pad
              ">
            <div className="label">Code</
                div>
            <input ref="code" className="
                pure-u-1" type="text"
                valueLink={this.linkState
                ("code")} required />
          </label>
          <label className="pure-u-1-2 pad
              ">
            <div className="label">Type</
                div>
            <select ref="level" className
                ="pure-u-1" valueLink={
                this.linkState("level")}
                required>
              <option value="region">
                  Region</option>
              <option value="province">
                  Province</option>
              <option value="municipality
                  ">City/Municipality</
                  option>
              <option value="barangay">
                  Barangay</option>
            </select>
          </label>
        </div>
        <div className="pure-g">
          <div className="pure-u-1">
            <button className="pull-right
                button button-primary mar
                " onClick={this.
                handle_save}>Save</button
                >
            {cancel_button}
            <button className="pull-left
                button button-complement
                mar" onClick={this.
                handle_delete}>Delete</
                button>
          </div>
        </div>
      </SliderTransitionGroupChild></
          ReactTransitionGroup>
      </div>
    );
  }else{
    var type = {
      "region": "Region",
      "province": "Province",
      "municipality": "City/Municipality",
      "barangay": "Barangay",
    }[this.model().get("level")];
    return (
      <div className="data-row form">
      <ReactTransitionGroup><
          SliderTransitionGroupChild key
          ="display">
        <div className="pure-g">
          <div className="pure-u-5-6">
            <div className="pure-g">
              <div className="pure-u-1
                  field-group">
                <Name className="pure-u-1
                    field text-large"
                    model={this.model()}
                    />
              </div>
            </div>
            <div className="pure-g">
              <div className="pure-u-1-3
                  field-group">
                <span className="pure-u
                    -1-4 label">Type</
                    span>
                <span className="pure-u
                    -3-4 field">{type}</
                    span>
              </div>
              <div className="pure-u-1-3
                  field-group">
                <span className="pure-u
                    -1-4 label">PSGC</
                    span>
                <span className="pure-u
                    -3-4 field">{this.
                    format_code(this.
                    model().get("code"))
                    }</span>
              </div>
            </div>
          </div>
          <div className="pure-u-1-6">
            <button className="pull-right
                button button-flat"
                onClick={this.handle_edit
                }>Edit</button>
          </div>
        </div>
      </SliderTransitionGroupChild></
          ReactTransitionGroup>
      </div>
    );
  }
},

format_code: function(code) {
  if(!code) return null;
  return ("0" + code).slice(-9);
},

handle_edit: function() {
  this.setState({edit: true});
},

handle_cancel: function() {
  this.setState({edit: false});
},

handle_save: function() {
  var name = this.refs.name.state.name;

  if(!this.validate()) return;

  var code = parseInt(this.state.code, 10)
      ;
  var level = this.state.level;

  this.save(code, name, level);
},
```

```
    save: function(code, name, level) {
      var that = this;

      var new_attributes = {
        code: code,
        name: name,
        level: level,
      };

      var patch = !this.model().has("id")
        ? new_attributes
        : _.omit(new_attributes, function(
            value, key, object) {
          return that.model().get(key) ===
              value;
        });

      if(_.isEmpty(patch)) {
        this.setState({edit: false});
      }else{
        this.model().save(patch, {
          patch: this.model().has("id"),
          wait: true,
          success: function() {
            that.setState({edit: false});
          }
        });
      }
    },

    handle_delete: function() {
      var that = this;

      ConfirmationDialog.open("Are you sure
          you want to delete this area?", [
        {
          display: "Cancel",
          type: "close",
        },
        {
          display: "Delete",
          type: "secondary",
          callback: function() {
            that.model().destroy({
              wait: true,
              error: function(xhr) {
                Notification.open(<span><p><i
                    className="fa fa-
                    exclamation-circle"/>&
                    ensp;Delete error: {xhr.
                    responseText}</p><p>Make
                    sure there are no
                    election records or
                    datapoints that reference
                     this area.</p></span>,
                    null, "error");
              },
            });
          },
        },
      ]);
    },
  });
});
```

*js-src/view/admin/AreasPanel.jsx*

```
"use strict";
define(function(require) {
  var React = require("react", "react.backbone
      "),
    FileInput = require("jsx!view/FileInput"),
    SearchControls = require("jsx!view/
        SearchControls"),
    PageControls = require("jsx!view/
        PageControls"),
    PanelToolbar = require("jsx!view/
        PanelToolbar"),
    AreaRow = require("jsx!view/admin/AreaRow
        "),
    ScrollToTopMixin = require("mixin/
        ScrollToTopMixin"),
    Notification = require("jsx!view/
        Notification"),
    ConfirmationDialog = require("jsx!view/
        ConfirmationDialog"),
    ReactCSSTransitionGroup = React.addons.
        CSSTransitionGroup;

  return React.createBackboneClass({
    mixins: [React.addons.LinkedStateMixin,
        ScrollToTopMixin],

    getInitialState: function() {
      return {
        upload_level: "region",
      };
    },

    componentDidMount: function() {
      this.onModelChange();
    },
    onModelChange: function() {
      if(this.refs.toolbar) {
        if(this.empty_data()) this.refs.
            toolbar.open();
        else this.refs.toolbar.close();
      }
      this.forceUpdate();
    },

    render: function() {
      return (
        <div className="body-panel clearfix">
          <PanelToolbar ref="toolbar"
              toggle_text="Add Data">
            <div className="pure-u-1-4 text-
                center pad">
              <h6>Add single row</h6>
              <button className="button"
                  onClick={this.handle_add}>
                  Add Row</button>
            </div>
            <div className="pure-u-3-8 text-
                center pad">
              <form ref="psgc_form" onSubmit={
                  this.handle_upload_psgc}>
                <h6>Upload PSGC table</h6>
                <div className="label">CSV
                    file</div>
                <div><FileInput ref="file_psgc
                    " type="file" /></div>
                <input className="button
                    button-primary" type="
                    submit" value="Upload
                    File" />
              </form>
            </div>
            <div className="pure-u-3-8 text-
                center pad">
              <form ref="geojson_form"
                  onSubmit={this.
                  handle_upload_geojson}>
                <h6>Update geometry</h6>
                <div className="label">GeoJSON
                    </div>
                <div><FileInput ref="
                    file_geojson" type="file"
                     /></div>
                <select className="pure-u-1
                    input" valueLink={this.
                    linkState("upload_level")
                    } required>
                  <option value="region">
                      Regions</option>
                  <option value="province">
                      Provinces</option>
                  <option value="municipality
                      ">Cities/Municipalities
                      </option>
                  <option value="barangay">
                      Barangays</option>
                </select>
                <input className="button
                    button-primary" type="
                    submit" value="Update" />
              </form>
            </div>
          </PanelToolbar>
          {!this.empty_data() ?
          <div>
            <SearchControls className="mar"
                ref="searcher" collection={
                this.collection()} />
            <ReactCSSTransitionGroup
                transitionName="fade">
              {this.collection().map(function(
                  area) {
                return <AreaRow key={area.cid}
                    model={area} />;
              })}
            </ReactCSSTransitionGroup>
            <PageControls className="text-
                center mar" collection={this.
                collection()} onNext={this.
                scroll_to_top} onPrev={this.
                scroll_to_top} />
            <button className="pull-right
                button button-complement mar"
                 onClick={this.
```

```
                handle_delete_all}>Delete All
                </button>
            </div>
            : null}
          </div>
        );
      },

      empty_data: function() {
        return !this.collection().size() && (!
            this.refs.searcher || this.refs.
            searcher.state.query === null) &&
            this.collection().getPage() === 0;
      },

      handle_delete_all: function() {
        var that = this;
        ConfirmationDialog.open("Are you sure
            you want to delete all family data
            ?", [
          {
            display: "Cancel",
            type: "close",
          },
          {
            display: "Delete All",
            type: "secondary",
            callback: function() {
              var notif = Notification.open(<
                  span><i className="fa fa-
                  circle-o-notch fa-spin"/>&
                  ensp;Deleting all areas...</
                  span>, 0);
              $.ajax({
                url: that.collection().url,
                type: "DELETE",
                success: function(data){
                  if(that.refs.toolbar) that.
                      refs.toolbar.open();
                  that.collection().fetch();
                  Notification.replace(notif, <
                      span><i className="fa fa-
                      trash"/> All areas
                      deleted</span>, null, "
                      success");
                },
                error: function(xhr) {
                  Notification.replace(notif, <
                      span><i className="fa fa-
                      exclamation-circle"/>&
                      ensp;Delete error: {xhr.
                      responseText}</span>,
                      null, "error");
                },
              });
            },
          },
        ]);
      },

      handle_add: function() {
        var that = this;
        if(!this.collection().size() || this.
            refs.searcher.state.query === null
            && this.collection().getPage() ===
            0) {
          this.refs.toolbar.close();
          this.collection().add({}, {at: 0});
        }else{
          this.refs.searcher.set_query(null, {
            complete: function() {
              if(that.refs.toolbar) that.refs.
                  toolbar.close();
              that.collection().add({}, {at: 0})
                  ;
            },
          });
        }
      },

      handle_upload_psgc: function(e) {
        var that = this;
        e.preventDefault();

        var fd = new FormData();
        var file = this.refs.file_psgc.get_input
            ().files[0];
        fd.append("file", file);

        var notif = Notification.open(<span><i
            className="fa fa-circle-o-notch fa-
            spin"/> Uploading {this.refs.
            file_psgc.get_filename()}...</span
            >, 0);

        $.ajax({
          url: this.collection().url,
          data: fd,
          processData: false,
          contentType: false,
          type: "POST",
          success: function(data){
            if(that.refs.psgc_form) React.
                findDOMNode(that.refs.psgc_form
                ).reset();
            if(that.refs.toolbar) that.refs.
                toolbar.close();
            that.collection().fetch();
            Notification.replace(notif, <span><i
                 className="fa fa-check-circle
                "/> Uploaded PSGC file: {
                that.refs.file_psgc.
                get_filename()}</span>, null, "
                success");
          },
          error: function(xhr) {
            Notification.replace(notif, <span><i
                 className="fa fa-exclamation-
                circle"/> Upload error: {
                that.refs.file_psgc.
                get_filename()}: {xhr.
                responseText}</span>, null, "
                error");
          },
        });
      },

      handle_upload_geojson: function(e) {
        var that = this;
        e.preventDefault();

        var fd = new FormData();
        var file = this.refs.file_geojson.
            get_input().files[0];
        fd.append("file", file);

        var notif = Notification.open(<span><i
            className="fa fa-circle-o-notch fa-
            spin"/> Updating {this.refs.
            file_geojson.get_filename()}...</
            span>, 0);

        $.ajax({
          url: "api.php/geojson/" + this.state.
              upload_level,
          data: fd,
          processData: false,
          contentType: false,
          type: "POST",
          success: function(data){
            React.findDOMNode(that.refs.
                geojson_form).reset();
            that.refs.toolbar.close();
            Notification.replace(notif, <span><i
                 className="fa fa-check-circle
                "/> Updated GeoJSON: {that
                .refs.file_geojson.get_filename
                ()}</span>, null, "success");
          },
          error: function(xhr) {
            Notification.replace(notif, <span><i
                 className="fa fa-exclamation-
                circle"/> Update error: {
                that.refs.file_geojson.
                get_filename()}: {xhr.
                responseText}</span>, null, "
                error");
          },
        });
      },
    });
  });
```

*js-src/view/admin/DatasetBox.jsx*

```
"use strict";
define(function(require) {
  var _ = require("underscore"),
    React = require("react", "react.backbone")
        ,
    Notification = require("jsx!view/
        Notification"),
    ConfirmationDialog = require("jsx!view/
        ConfirmationDialog");

  return React.createBackboneClass({
    render: function() {
      var username = this.model().get("
          username");
```

```
        var url_datasets = "#users/" + username
            + "/datasets";
        return (
          <div className="data-row">
            <div className="pure-g">
              <div className="pure-u-5-6">
                <div className="field text-large
                    ">{this.model().get("name")
                    }</div>
                <div className="field-group">
                  <span className="pure-u-1-6
                      label">Uploaded by</span>
                  <span className="pure-u-5-6
                      field">
                    <a href={url_datasets}>{
                        username}</a>
                  </span>
                </div>
                <div className="field-group">
                  <span className="pure-u-1-6
                      label">Range</span>
                  <span className="pure-u-5-6
                      field">
                    {this.model().get("min_year
                        ")}-{this.model().get("
                        max_year")}
                  </span>
                </div>
                <div className="field-group">
                  <span className="pure-u-1-6
                      label">Level</span>
                  <span className="pure-u-5-6
                      field">
                    {_.chain(this.model().get("
                        contained_levels"))
                      .pick(function(v){ return
                          v; }).keys()
                      .map(function(k) {
                        return {
                          "region": "Regional",
                          "province": "
                              Provincial",
                          "municipality": "
                              Municipal",
                          "barangay": "Barangay
                              ",
                        }[k];
                      })
                      .values().join(", ")}
                  </span>
                </div>
                <div className="pure-u-1 field
                    text"><pre>{this.model().
                    get("description")}</pre></
                    div>
              </div>
              <div className="pure-u-1-6">
                <button className="pull-right
                    button button-flat button-
                    complement" onClick={this.
                    handle_delete}>Delete</
                    button>
              </div>
            </div>
          </div>
        );
      },

      handle_delete: function() {
        var that = this;
        ConfirmationDialog.open("Are you sure
            you want to delete this dataset?",
            [
          {
            display: "Cancel",
            type: "close",
          },
          {
            display: "Delete",
            type: "secondary",
            callback: function() {
              var name = that.model().get("name
                  ");
              var notif = Notification.open(<
                  span><i className="fa fa-
                  circle-o-notch fa-spin"/>&
                  ensp;Deleting {name}...</span
                  >, 0);
              that.model().destroy({
                wait: true,
                success: function(){
                  Notification.replace(notif, <
                      span><i className="fa fa-
                      trash"/> {name}
                      deleted</span>);
                },
                error: function(xhr) {
                  Notification.replace(notif, <
                      span><i className="fa fa-
                      exclamation-circle"/>&
                      ensp;Delete error: {xhr.
                      responseText}</span>,
                      null, "error");
                },
              });
            },
          },
        ]);
      },
    });
});
```

*js-src/view/admin/DatasetsPanel.jsx*

```
"use strict";
define(function(require) {
  var React = require("react", "react.backbone
      "),
    InstanceCache = require("InstanceCache"),
    FileInput = require("jsx!view/FileInput"),
    SearchControls = require("jsx!view/
        SearchControls"),
    PageControls = require("jsx!view/
        PageControls"),
    PanelToolbar = require("jsx!view/
        PanelToolbar"),
    DatasetBox = require("jsx!view/admin/
        DatasetBox"),
    ScrollToTopMixin = require("mixin/
        ScrollToTopMixin"),
    Notification = require("jsx!view/
        Notification"),
    ReactCSSTransitionGroup = React.addons.
        CSSTransitionGroup;

  return React.createBackboneClass({
    mixins: [ScrollToTopMixin],

    render: function() {
      if(!this.collection().username) {
        var toolbar = (
          <PanelToolbar ref="toolbar"
              toggle_text="Generate Dataset">
            <div className="pure-u-1 text-
                center pad">
              <h6>Generate political dynasty
                  indicators</h6>
              <button className="button"
                key="btn_DYNSHA"
                onClick={this.
                    indicator_generator.bind(
                    this, "DYNSHA")}>
                Generate DYNSHA
              </button>
              <button className="button"
                key="btn_DYNLAR"
                onClick={this.
                    indicator_generator.bind(
                    this, "DYNLAR")}>
                Generate DYNLAR
              </button>
              <button className="button"
                key="btn_DYNHERF"
                onClick={this.
                    indicator_generator.bind(
                    this, "DYNHERF")}>
                Generate DYNHERF
              </button>
              <br />
              <button className="button"
                key="btn_LocalDynastySize"
                onClick={this.
                    indicator_generator.bind(
                    this, "LocalDynastySize")
                    }>
                Generate Local Dynasty Size
              </button>
              <button className="button"
                key="btn_RecursiveDynastySize"
                onClick={this.
                    indicator_generator.bind(
                    this, "
                    RecursiveDynastySize")}>
                Generate Recursive Dynasty
                    Size
              </button>
            </div>
          </PanelToolbar>
        );
      }
```

```
      return (
        <div className="body-panel">
          {toolbar}
          {!this.empty_data() ?
          <div>
            <SearchControls ref="searcher"
                 className="mar" collection={
                 this.collection()} />
            <ReactCSSTransitionGroup
                 transitionName="fade">
              {this.collection().map(function(
                   dataset) {
                return <DatasetBox key={
                     dataset.cid} model={
                     dataset} />;
              })}
            </ReactCSSTransitionGroup>
            <PageControls className="text-
                 center mar" collection={this.
                 collection()} onNext={this.
                 scroll_to_top} onPrev={this.
                 scroll_to_top} />
          </div>
          :
          <div className="text-center">
            <h1 className="transparent">No
                 Datasets Yet</h1>
            <p className="text text-center">
                 User-uploaded datasets go
                 here.</p>
            <p className="text text-center">
                 The system can also generate
                 datasets based on election
                 records and dynasty data.</p>
          </div>
          }
        </div>
      );
    },

    empty_data: function() {
      return !this.collection().size() && (!
           this.refs.searcher || this.refs.
           searcher.state.query === null) &&
           this.collection().getPage() === 0;
    },

    indicator_generator: function(indicator) {
      var that = this;

      var descriptions = {
        "DYNSHA": "Proportion of dynastic
             officials in each local
             government unit.",
        "DYNLAR": "Proportion of the dynasty
             with the largest proportion of
             dynastic officials in each local
             government unit.",
        "DYNHERF": "Herfindal index on
             dynasties in each local
             government unit.",
        "LocalDynastySize": "Number of members
              of each dynasty in each local
             government unit.",
        "RecursiveDynastySize": "Number of
             members of each dynasty in each
             local government unit including
             members in subdivisions.",
      };

      var token = InstanceCache.get_existing("
           Token", "session");
      var user = token ? token.get_user() :
           null;
      if(!user) return;

      var props = {
        username: user.get("username"),
        indicator: indicator,
        description: descriptions[indicator] +
             " Generated on " + new Date(),
      };

      var notif = Notification.open(<span><i
           className="fa fa-circle-o-notch fa-
           spin"/> Generating {indicator}
            variable...</span>, 0);

      $.ajax({
        method: "POST",
        url: "api.php/generate-indicator",
        data: JSON.stringify(props),
        processData: false,
        dataType: "json",
        success: function(data) {
          if(that.refs.toolbar) that.refs.
               toolbar.close();
          that.collection().fetch();
          Notification.replace(notif, <span><i
               className="fa fa-check-circle
              "/> {indicator} variable
              generated</span>, null, "
              success");
        },
        error: function(xhr) {
          Notification.replace(notif, <span><i
               className="fa fa-exclamation-
              circle"/> Generate error:
              {xhr.responseText}</span>, null
              , "error");
        },
      });
    },
  });
});
```

*js-src/view/admin/ElectionRow.jsx*

```
"use strict";
define(function(require) {
  var React = require("react", "react.backbone
       "),
    InstanceCache = require("InstanceCache"),
    OfficialSingle = require("model/
         OfficialSingle"),
    Party = require("model/Party"),
    SliderTransitionGroupChild = require("jsx!
         view/SliderTransitionGroupChild"),
    TypeaheadInput = require("jsx!view/
         TypeaheadInput"),
    OfficialName = require("jsx!view/
         OfficialName"),
    Name = require("jsx!view/Name"),
    ClickToTopMixin = require("mixin/
         ClickToTopMixin"),
    Va = require("validator"),
    ValidationMixin = require("mixin/
         ValidationMixin"),
    ValidationMessages = require("jsx!view/
         ValidationMessages"),
    ConfirmationDialog = require("jsx!view/
         ConfirmationDialog"),
    ReactTransitionGroup = React.addons.
         TransitionGroup;

  return React.createBackboneClass({
    mixins: [React.addons.LinkedStateMixin,
         ValidationMixin, ClickToTopMixin],

    getInitialState: function() {
      return {
        edit: this.model().isNew(),
        position: this.model().get("position")
             ,
        year: this.model().get("year"),
        year_end: this.model().get("year_end")
             ,
        votes: this.model().get("votes"),
      };
    },

    componentWillMount: function() {
      this.official_name_regex = /^\s*(.+?)\s
           *,\s*(.+?)\s*(?:"(.+?)")?\s*$/;
    },

    getValidationSchema: function() {
      return {
        official_name: Va.lidator()
          .optional_if(function(name, others)
               { return others.official !==
               null; })
          .required().string().custom(function
               (name) {
            var tokens = name.match(this.
                 official_name_regex);
            return tokens && tokens.length >
                 2;
          }.bind(this), "format invalid"),
        official: Va.lidator()
          .optional_if(function(official,
               others) { return others.
               official_name; })
          .required().object(),
        area: Va.lidator().required().object()
             ,
        party: Va.lidator().object(),
        position: Va.lidator().string(),
        year: Va.lidator().required().
             integerish().lessThan({key: "
```

```
        year_end"}),
    year_end: Va.lidator().required().
        integerish(),
    votes: Va.lidator().integerish(),
  };
},
getObjectToValidate: function() {
  return {
    official_name: this.refs.official.
        state.value,
    official: this.refs.official.state.
        selected,
    area: this.refs.area.state.selected,
    party: this.refs.party.state.selected,
    position: this.state.position,
    year: this.state.year,
    year_end: this.state.year_end,
    votes: this.state.votes,
  };
},
getValidationElementMap: function() {
  return {
    official_name: React.findDOMNode(this.
        refs.official),
    official: React.findDOMNode(this.refs.
        official),
    area: React.findDOMNode(this.refs.area
        ),
    party: React.findDOMNode(this.refs.
        party),
    position: React.findDOMNode(this.refs.
        position),
    year: React.findDOMNode(this.refs.year
        ),
    year_end: React.findDOMNode(this.refs.
        year_end),
    votes: React.findDOMNode(this.refs.
        votes),
  };
},
validationCallback: function(key, valid,
    message) {
  // React.findDOMNode(this.refs.save).
      disabled = !valid;
},

render: function() {
  var display = function(item) {
    return item.name;
  };
  var display_official = function(item) {
    return item.surname + ", " + item.name
        ;
  };

  var official_id = this.model().get("
      official_id");
  var area_code = this.model().get("
      area_code");
  var party_id = this.model().get("
      party_id");

  var official = official_id ?
      InstanceCache.get("Official",
      official_id, true) : null;
  var area = area_code ? InstanceCache.get
      ("Area", area_code, true) : null;
  var party = party_id ? InstanceCache.get
      ("Party", party_id, true) : null;

  if(this.model().isNew() || this.state.
      edit) {
    if(!this.model().isNew()){
      var cancel_button = <button
          className="pull-right button
          mar" type="reset" onClick={this
          .handle_cancel}>Cancel</button
          >;
    }
    return (
      <div className="edit data-row form">
      <ReactTransitionGroup><
          SliderTransitionGroupChild key
          ="edit">
        <ValidationMessages validation={
            this.state.validation} />
        <div className="pure-g">
          <label className="pure-u-1-2 pad
              ">
            <div className="label">
                Official</div>
            <TypeaheadInput className="
                pure-u-1"
              for="Official"
              ref="official"
              display={display_official}
              model={official}
              required autoFocus />
          </label>
          <label className="pure-u-1-2 pad
              ">
            <div className="label">
                Position</div>
            <input ref="position"
                className="pure-u-1" type
                ="text" valueLink={this.
                linkState("position")} />
          </label>
        </div>
        <div className="pure-g">
          <label className="pure-u-1-2 pad
              ">
            <div className="label">Area</
                div>
            <TypeaheadInput className="
                pure-u-1"
              for="Area"
              ref="area"
              display={display}
              model={area}
              required />
          </label>
          <label className="pure-u-1-4 pad
              ">
            <div className="label">Year</
                div>
            <input ref="year" className="
                pure-u-1" type="number"
                valueLink={this.linkState
                ("year")} required />
          </label>
          <label className="pure-u-1-4 pad
              ">
            <div className="label">Year
                end</div>
            <input ref="year_end"
                className="pure-u-1" type
                ="number" valueLink={this
                .linkState("year_end")}
                required />
          </label>
        </div>
        <div className="pure-g">
          <label className="pure-u-1-2 pad
              ">
            <div className="label">Votes</
                div>
            <input ref="votes" className="
                pure-u-1" type="number"
                valueLink={this.linkState
                ("votes")} />
          </label>
          <label className="pure-u-1-2 pad
              ">
            <div className="label">Party</
                div>
            <TypeaheadInput className="
                pure-u-1"
              for="Party"
              ref="party"
              display={display}
              model={party} />
          </label>
        </div>
        <div className="pure-g">
          <div className="pure-u-1">
            <button className="pull-right
                button button-primary mar
                " type="submit" onClick={
                this.handle_save}>Save</
                button>
            {cancel_button}
            <button className="pull-left
                button button-complement
                mar" type="button"
                onClick={this.
                handle_delete}>Delete</
                button>
          </div>
        </div>
      </SliderTransitionGroupChild></
          ReactTransitionGroup>
      </div>
    );
  }else{
    var votes = this.model().get("votes");
    return (
      <div className="data-row form">
      <ReactTransitionGroup><
          SliderTransitionGroupChild key
          ="display">
        <div className="pure-g">
          <div className="pure-u-5-6">
            <div className="pure-g field-
                group">
              <OfficialName className="
                  pure-u-1 field text-
                  large" model={official}
```

```
                    />
                </div>
                <div className="pure-g">
                  <div className="pure-u-1-2">
                    <div className="field-
                        group">
                      <span className="pure-u
                          -1-3 label">
                          Position</span>
                      <span className="pure-u
                          -2-3 field">{this.
                          model().get("
                          position")}</span>
                    </div>
                    <div className="field-
                        group">
                      <span className="pure-u
                          -1-3 label">Area</
                          span>
                      <Name className="pure-u
                          -2-3 field" model={
                          area} />
                    </div>
                    <div className="field-
                        group">
                      <span className="pure-u
                          -1-3 label">
                          Duration</span>
                      <span className="pure-u
                          -2-3 field">{this.
                          model().get("year")
                          } - {this.model().
                          get("year_end")}</
                          span>
                    </div>
                  </div>
                  <div className="pure-u-1-2">
                    <div className="field-
                        group">
                      <span className="pure-u
                          -1-3 label">Votes</
                          span>
                      <span className="pure-u
                          -2-3 field">{votes
                          !== null ? votes :
                          ""}</span>
                    </div>
                    <div className="field-
                        group">
                      <span className="pure-u
                          -1-3 label">Party</
                          span>
                      <Name className="pure-u
                          -2-3 field" model={
                          party} />
                    </div>
                  </div>
                </div>
                <div className="pure-u-1-6">
                  <button className="pull-right
                      button button-flat"
                      onClick={this.handle_edit
                      }>Edit</button>
                </div>
              </div>
            </SliderTransitionGroupChild></
                ReactTransitionGroup>
          </div>
      );
    }
  },

  handle_edit: function() {
    this.setState({edit: true});
  },

  handle_cancel: function() {
    this.resetValidation();
    this.setState({edit: false});
  },

  handle_delete: function(e) {
    var that = this;
    ConfirmationDialog.open("Are you sure
        you want to delete this record?", [
      {
        display: "Cancel",
        type: "close",
      },
      {
        display: "Delete",
        type: "secondary",
        callback: function() {
          that.model().destroy({wait: true})
              ;
        },
      },
    ]);
  },

  handle_save: function(e) {
    e.preventDefault();
    var that = this;

    if(!that.validate()) return;

    var dummy = {};
    var party = dummy;
    var official = dummy;
    var created_party = false;
    var created_official = false;

    // Callback hell!!

    var callback = function() {
      if(official !== dummy && party !==
          dummy) {
        var area = that.refs.area.state.
            selected;
        var position = that.state.position
            || null;
        var year = parseInt(that.state.year,
            10);
        var year_end = parseInt(that.state.
            year_end, 10);
        var votes = parseInt(that.state.
            votes, 10);
        that.save(official, year, year_end,
            position, votes, area, party,
            created_party, created_official
            );
      }
    };

    this.refs.party.get_or_create({
      model: Party,
      attributes: function(str) {
        return {name: that.refs.party.state.
            value};
      },
      callback: function(item, created) {
        party = item;
        created_party = created;
        callback();
      },
    });

    this.refs.official.get_or_create({
      model: OfficialSingle,
      attributes: function(str) {
        var tokens = that.refs.official.
            state.value.match(that.
            official_name_regex);
        return {
          surname: tokens[1],
          name: tokens[2],
          nickname: tokens.length > 3 ?
              tokens[3] : null,
        };
      },
      search: ["surname", "name"],
      callback: function(item, created) {
        official = item;
        created_official = created;
        callback();
      },
    });
  },

  save: function(official, year, year_end,
      position, votes, area, party,
      created_party, created_official) {
    var that = this;

    var new_attributes = {
      official_id: official.id,
      year: year,
      year_end: year_end,
      position: position,
      votes: votes,
      area_code: area.code,
      party_id: party ? party.id : null,
    };

    var patch = this.model().isNew()
      ? new_attributes
      : _.omit(new_attributes, function(
          value, key, object) {
        return that.model().get(key) ===
            value;
      });

    if(_.isEmpty(patch)) {
      this.setState({edit: false});
    }else{
      this.model().save(patch, {
        patch: !this.model().isNew(),
```

```
            wait: true,
            success: function() {
              that.setState({edit: false});
            },
            error: function() {
              if(created_party) party.destroy();
              if(created_official) official.
                  destroy();
            },
          });
        }
      },
    });
```

*js-src/view/admin/ElectionsPanel.jsx*

```
"use strict";
define(function(require) {
  var $ = require("jquery"),
    React = require("react", "react.backbone")
        ,
    FileInput = require("jsx!view/FileInput"),
    SearchControls = require("jsx!view/
        SearchControls"),
    PageControls = require("jsx!view/
        PageControls"),
    PanelToolbar = require("jsx!view/
        PanelToolbar"),
    ElectionRow = require("jsx!view/admin/
        ElectionRow"),
    ScrollToTopMixin = require("mixin/
        ScrollToTopMixin"),
    Notification = require("jsx!view/
        Notification"),
    ConfirmationDialog = require("jsx!view/
        ConfirmationDialog"),
    ReactCSSTransitionGroup = React.addons.
        CSSTransitionGroup;

  return React.createBackboneClass({
    mixins: [ScrollToTopMixin],

    componentDidMount: function() {
      this.onModelChange();
    },
    onModelChange: function() {
      if(this.refs.toolbar) {
        if(this.empty_data()) this.refs.
            toolbar.open();
        else this.refs.toolbar.close();
      }
      this.forceUpdate();
    },

    render: function() {
      var that = this;
      return (
        <div className="body-panel clearfix">
          <PanelToolbar ref="toolbar"
              toggle_text="Add Data">
            <div className="pure-u-1-3 text-
                center pad">
              <h6>Add a single row</h6>
              <button className="button"
                  onClick={this.handle_add}>
                  Add Row</button>
            </div>
            <div className="pure-u-2-3 text-
                center pad">
              <form ref="upload_form" onSubmit
                  ={this.handle_upload}>
                <h6>Upload election records</
                    h6>
                <div className="label">CSV
                    file</div>
                <div><FileInput ref="file"
                    type="file" /></div>
                <input className="button
                    button-primary" type="
                    submit" value="Upload
                    File" />
              </form>
            </div>
          </PanelToolbar>
          {!this.empty_data() ?
          <div>
            <SearchControls className="mar"
                ref="searcher" collection={
                this.collection()} />
            <ReactCSSTransitionGroup
                transitionName="fade">
              {this.collection().map(function(
                  election) {
                return <ElectionRow
                  key={election.cid}
                  model={election} />;
              })}
            </ReactCSSTransitionGroup>
            <PageControls className="text-
                center mar" collection={this.
                collection()} onNext={this.
                scroll_to_top} onPrev={this.
                scroll_to_top} />
            <button className="pull-right
                button button-complement mar"
                 onClick={this.
                handle_delete_all}>Delete All
                </button>
          </div>
          : null}
        </div>
      );
    },

    empty_data: function() {
      return !this.collection().size() && (!
          this.refs.searcher || this.refs.
          searcher.state.query === null) &&
          this.collection().getPage() === 0;
    },

    handle_delete_all: function() {
      var that = this;
      ConfirmationDialog.open("Are you sure
          you want to delete all records?", [
        {
          display: "Cancel",
          type: "close",
        },
        {
          display: "Delete All",
          type: "secondary",
          callback: function() {
            var notif = Notification.open(<
                span><i className="fa fa-
                circle-o-notch fa-spin"/>&
                ensp;Deleting all election
                records and related data...</
                span>, 0);
            $.ajax({
              url: that.collection().url,
              type: "DELETE",
              success: function(data){
                if(that.refs.toolbar) that.
                    refs.toolbar.open();
                that.collection().fetch();
                Notification.replace(notif, <
                    span><i className="fa fa-
                    trash"/> All
                    election records and
                    related data deleted</
                    span>, null, "success");
              },
              error: function(xhr) {
                Notification.replace(notif, <
                    span><i className="fa fa-
                    exclamation-circle"/>&
                    ensp;Delete error: {xhr.
                    responseText}</span>,
                    null, "error");
              },
            });
          },
        },
      ]);
    },

    handle_add: function() {
      var that = this;
      if(!this.collection().size() || this.
          refs.searcher.state.query === null
          && this.collection().getPage() ===
          0) {
        this.refs.toolbar.close();
        this.collection().add({}, {at: 0});
      }else{
        this.refs.searcher.set_query(null, {
          complete: function() {
            if(that.refs.toolbar) that.refs.
                toolbar.close();
            that.collection().add({}, {at: 0})
                ;
          },
        });
      }
    },

    handle_upload: function(e) {
```

```
        var that = this;
        e.preventDefault();

        var fd = new FormData();
        var file = this.refs.file.get_input().
            files[0];
        fd.append("file", file);

        var notif = Notification.open(<span><i
            className="fa fa-circle-o-notch fa-
            spin"/> Uploading {this.refs.
            file.get_filename()}...</span>, 0);

        $.ajax({
          url: this.collection().url,
          data: fd,
          processData: false,
          contentType: false,
          type: "POST",
          success: function(data){
            if(that.refs.upload_form) React.
                findDOMNode(that.refs.
                upload_form).reset();
            if(that.refs.toolbar) that.refs.
                toolbar.close();
            that.collection().fetch();
            Notification.replace(notif, <span><i
                 className="fa fa-check-circle
                "/> Uploaded election
                records: {that.refs.file.
                get_filename()}</span>, null, "
                success");
          },
          error: function(xhr) {
            Notification.replace(notif, <span><i
                 className="fa fa-exclamation-
                circle"/> Upload error: {
                that.refs.file.get_filename()}:
                 {xhr.responseText}</span>,
                null, "error");
          },
        });
      },
    });
});
```

*js-src/view/admin/FamiliesPanel.jsx*

```
"use strict";
define(function(require) {
  var React = require("react", "react.backbone
      "),
    SearchControls = require("jsx!view/
        SearchControls"),
    PageControls = require("jsx!view/
        PageControls"),
    FamilyBox = require("jsx!view/admin/
        FamilyBox"),
    ScrollToTopMixin = require("mixin/
        ScrollToTopMixin"),
    Notification = require("jsx!view/
        Notification"),
    ConfirmationDialog = require("jsx!view/
        ConfirmationDialog"),
    ReactCSSTransitionGroup = React.addons.
        CSSTransitionGroup;

  return React.createBackboneClass({
    mixins: [ScrollToTopMixin],

    render: function() {
      var that = this;
      return (
        <div className="body-panel">
          <button className="button button-
              primary mar" onClick={this.
              handle_add}>New Family</button>
          {!this.empty_data() ?
          <div className="clearfix">
            <SearchControls ref="searcher"
                className="mar" collection={
                this.collection()} />
            <ReactCSSTransitionGroup
                transitionName="fade">
              {this.collection().map(function(
                  family) {
                return <FamilyBox key={family.
                    cid} model={family}
                    onDeleteMember={that.
                    handle_delete_official}
                    />;
              })}
            </ReactCSSTransitionGroup>
            <PageControls className="text-
                center mar" collection={this.
                collection()} onNext={this.
                scroll_to_top} onPrev={this.
                scroll_to_top} />
            <button className="pull-right
                button button-complement mar"
                 onClick={this.
                handle_delete_all}>Delete All
                </button>
          </div>
          :
          <div className="text-center">
            <h1 className="transparent">No
                Families Yet</h1>
            <p className="text text-center">
                Add families using the New
                Family button</p>
          </div>
          }
        </div>
      );
    },

    empty_data: function() {
      return !this.collection().size() && (!
          this.refs.searcher || this.refs.
          searcher.state.query === null) &&
          this.collection().getPage() === 0;
    },

    handle_delete_all: function() {
      var that = this;
      ConfirmationDialog.open("Are you sure
          you want to delete all family data
          ?", [
        {
          display: "Cancel",
          type: "close",
        },
        {
          display: "Delete All",
          type: "secondary",
          callback: function() {
            var notif = Notification.open(<
                span><i className="fa fa-
                circle-o-notch fa-spin"/>&
                ensp;Deleting all families
                ...</span>, 0);
            $.ajax({
              url: that.collection().url,
              type: "DELETE",
              success: function(data){
                that.collection().fetch();
                Notification.replace(notif, <
                    span><i className="fa fa-
                    check-circle"/> All
                    families deleted</span>,
                    null, "success");
              },
              error: function(xhr) {
                Notification.replace(notif, <
                    span><i className="fa fa-
                    exclamation-circle"/>&
                    ensp;Delete error: {xhr.
                    responseText}</span>,
                    null, "error");
              },
            });
          },
        },
      ]);
    },

    handle_add: function() {
      var that = this;
      if(!this.collection().size() || this.
          refs.searcher.state.query === null
          && this.collection().getPage() ===
          0) {
        this.collection().add({}, {at: 0});
      }else{
        this.refs.searcher.set_query(null, {
          complete: function() {
            that.collection().add({}, {at: 0})
                ;
          },
        });
      }
    },

    handle_delete_official: function() {
      this.collection().fetch();
    },
  });
});
```

*js-src/view/admin/FamilyBox.jsx*

```
"use strict";
define(function(require) {
  var React = require("react", "react.backbone
      "),
    FamilyMemberCollection = require("model/
        FamilyMemberCollection"),
    SliderTransitionGroupChild = require("jsx!
        view/SliderTransitionGroupChild"),
    EditableName = require("jsx!view/
        EditableName"),
    Name = require("jsx!view/Name"),
    FamilyMemberList = require("jsx!view/admin
        /FamilyMemberList"),
    ClickToTopMixin = require("mixin/
        ClickToTopMixin"),
    Va = require("validator"),
    ValidationMixin = require("mixin/
        ValidationMixin"),
    ValidationMessages = require("jsx!view/
        ValidationMessages"),
    ConfirmationDialog = require("jsx!view/
        ConfirmationDialog"),
    ReactTransitionGroup = React.addons.
        TransitionGroup;

  return React.createBackboneClass({
    mixins: [ValidationMixin, ClickToTopMixin
        ],

    getInitialState: function() {
      return {
        edit: this.model().isNew(),
        members: new FamilyMemberCollection(
            null, {family_id: this.model().id
            }),
      };
    },

    getValidationSchema: function() {
      return {
        name: Va.lidator().required().string()
            ,
      };
    },
    getObjectToValidate: function() {
      return {
        name: this.refs.name.state.name,
      };
    },
    getValidationElementMap: function() {
      return {
        name: React.findDOMNode(this.refs.name
            ),
      };
    },
    validationCallback: function(key, valid,
        message) {
      // React.findDOMNode(this.refs.save).
          disabled = !valid;
    },

    componentWillMount: function() {
      if(!this.model().isNew()) {
        this.state.members.fetch();
        this.state.members.on("remove", this.
            check_empty, this);
      }
    },

    componentDidUnmount: function() {
      if(!this.model().isNew()) {
        this.state.members.off("remove", this.
            check_empty, this);
      }
    },

    render: function() {
      var classes = "data-row";
      var fields = null;
      if(this.model().isNew() || this.state.
          edit) {
        classes += " edit";
        if(!this.model().isNew()){
          var cancel_button = <button
              className="pull-right button
              mar" type="reset" onClick={this
              .handle_cancel}>Cancel</button
              >;
        }
        fields = (
          <form onSubmit={this.handle_save}>
            <ValidationMessages validation={
                this.state.validation} />
            <div className="pure-g form pad">
              <EditableName className="pure-u
                  -1" ref="name" model={this.
                  model()} autoFocus />
            </div>
            <div className="pure-g">
              <div className="pure-u-1">
                <button ref="save" className="
                    pull-right button button-
                    primary mar" type="submit
                    ">Save</button>
                {cancel_button}
                <button className="pull-left
                    button button-complement
                    mar" type="button"
                    onClick={this.
                    handle_delete}>Delete</
                    button>
              </div>
            </div>
          </form>
        );
      }else{
        fields = [
          (<div className="pure-g">
            <Name className="field pure-u-5-6
                text-large" model={this.model
                ()} />
            <div className="pure-u-1-6">
              <button className="pull-right
                  button button-flat" onClick
                  ={this.handle_edit}>Edit</
                  button>
            </div>
          </div>),
          (<FamilyMemberList collection={this.
              state.members} onDeleteMember={
              this.props.onDeleteMember} />)
        ];
      }
      return (
        <div className={classes}>
        <ReactTransitionGroup><
            SliderTransitionGroupChild key={
            classes}>
          {fields}
        </SliderTransitionGroupChild></
            ReactTransitionGroup>
        </div>
      );
    },

    check_empty: function() {
      if(this.state.members.size() == 0) {
        this.model().destroy();
      }
    },

    handle_edit: function() {
      this.setState({edit: true});
    },

    handle_cancel: function() {
      this.resetValidation();
      this.setState({edit: false});
    },

    handle_save: function(e) {
      e.preventDefault();

      if(!this.validate()) return;

      var that = this;
      var new_name = this.refs.name.state.name
          ;
      if(!this.model().isNew() && this.model()
          .get("name") === new_name) {
        this.setState({edit: false});
      }else{
        this.model().save({name: new_name}, {
          patch: !this.model().isNew(),
          wait: true,
          success: function() {
            that.setState({edit: false});
          }
        });
      }
    },

    handle_delete: function(e) {
      var that = this;
      ConfirmationDialog.open("Are you sure
          you want to delete this family?", [
        {
          display: "Cancel",
          type: "close",
        },
        {
          display: "Delete",
```

```
          type: "secondary",
          callback: function() {
            that.model().destroy({wait: true})
                ;
          },
```

```
        },
      ]);
    },
  });
});
```

*js-src/view/admin/FamilyMemberItem.jsx*

```
"use strict";
define(["react", "jsx!view/OfficialName", "
    react.backbone"], function(React,
    OfficialName) {
  return React.createBackboneClass({
    render: function() {
      return (
        <li className="family-member-item
            field clearfix">
          <span className="number pure-u
              -1-12">{this.props.number}.</
              span>
          <OfficialName className="pure-u-5-6"
              model={this.model()} />
          <div className="pure-u-1-12">
            <button className="pull-right
                button button-complement
                button-flat button-close"
                onClick={this.handle_delete
```

```
                }>&times;</button>
          </div>
        </li>
      );
    },

    handle_delete: function() {
      var that = this;
      this.model().destroy({
        wait: true,
        success: function() {
          that.props.onDeleteMember(that.model
              ());
        },
      });
    },
  });
});
```

*js-src/view/admin/FamilyMemberList.jsx*

```
"use strict";
define(["react", "jsx!view/TypeaheadInput", "
    jsx!view/admin/FamilyMemberItem", "react.
    backbone"], function(React,
    TypeaheadInput, FamilyMemberItem) {
  return React.createBackboneClass({
    render: function() {
      var that = this;

      var display =  function(item) {
        return item.surname + ", " + item.name
            ;
      };

      return (
        <div className="pure-g">
          <div className="pure-u-1">
            <div className="label">Members</
                div>
            <ol>
              {this.collection().map(function(
                  official, i) {
                return <FamilyMemberItem key={
                    official.cid} model={
                    official} number={i+1}
                    onDelete={that.props.
                    onDeleteMember} />;
              })}
            </ol>
            <form onSubmit={this.handle_submit
                }>
              <div className="group">
                <div className="group-
                    component">
```

```
                  <TypeaheadInput className="
                      input"
                    for="Official"
                    ref="input"
                    display={display}
                    collection={this.
                        collection()}
                    required />
                </div>
                <input className="group-
                    component button" type="
                    submit" value="Add Member
                    " />
              </div>
            </form>
          </div>
        </div>
      );
    },

    handle_submit: function(e) {
      e.preventDefault();
      if(this.refs.input.state.selected) {
        this.collection().post_member(this.
            refs.input.state.selected);
        this.refs.input.clear();
      }else{
        console.error("None selected.");
      }
    },
  });
});
```

*js-src/view/admin/HomePanel.jsx*

```
"use strict";
define(["underscore", "jquery", "react"],
    function(_, $, React) {
  return React.createClass({
    getInitialState: function() {
      return {
        elections: 8,
        areas: 8,
        officials: 8,
        families: 8,
        datasets: 8,
        users: 8,

        region: 8,
        province: 8,
        municipality: 8,
        barangay: 8,

        area_datasets: 8,
        tag_datasets: 8,
      };
    },

    componentWillMount: function() {
      var that = this;

      var create_callback = function(key) {
        return function(data) {
```

```
          var s = {};
          s[key] = data.total;
          that.setState(s);
        };
      };

      _.each(["elections", "areas", "officials
          ", "families", "datasets", "users
          "], function(x) {
        $.get("api.php/" + x, {count: 1},
            create_callback(x), "json");
      });
      _.each(["region", "province", "
          municipality", "barangay"],
          function(x) {
        $.get("api.php/areas", {level: x,
            count: 1}, create_callback(x), "
            json");
      });
      _.each(["area", "tag"], function(x) {
        $.get("api.php/datasets", {type: x,
            count: 1}, create_callback(x + "
            _datasets"), "json");
      });
    },

    render: function() {
      var tiles = [
```

```
  { name: "elections",
    contents: (
      <div className="tile">
        <h1 className="text-center">
            Elections</h1>
        <div className="pure-u-1 field
            ">{this.state.elections}
            election records</div>
        <p className="text">Update
            election records</p>
      </div>
    ),
  },
  { name: "areas",
    contents: (
      <div className="tile">
        <h1 className="text-center">
            Areas</h1>
        <div className="pure-u-1-2 field
            ">{this.state.region}
            regions</div>
        <div className="pure-u-1-2 field
            ">{this.state.province}
            provinces</div>
        <div className="pure-u-1-2 field
            ">{this.state.municipality}
             cities/municipalities</div
            >
        <div className="pure-u-1-2 field
            ">{this.state.barangay}
            barangays</div>
        <p className="text">Update
            administrative subdivisions
            , PSGC codes, and GeoJSON</
            p>
      </div>
    ),
  },
  { name: "officials",
    contents: (
      <div className="tile">
        <h1 className="text-center">
            Officials</h1>
        <div className="pure-u-1 field
            ">{this.state.officials}
            officials</div>
        <p className="text">Update
            officials data and family
            associations</p>
      </div>
    ),
  },
  { name: "families",
    contents: (
      <div className="tile">
        <h1 className="text-center">
            Families</h1>
        <div className="pure-u-1 field
            ">{this.state.families}
            families</div>
        <p className="text">Update
            political families and
            members</p>
      </div>
    ),
  },
  { name: "datasets",
    contents: (
      <div className="tile">
        <h1 className="text-center">
            Datasets</h1>
        <div className="pure-u-1 field
            ">{this.state.datasets}
            datasets</div>
        <p className="text">Manage user-
            uploaded and system-
            generated datasets</p>
      </div>
    ),
  },
  { name: "users",
    contents: (
      <div className="tile">
        <h1 className="text-center">
            Users</h1>
        <div className="pure-u-1 field
            ">{this.state.users} users
            </div>
        <p className="text">Manage
            administrators and regular
            users</p>
      </div>
    ),
  },
];

var message = null;
var hide = [];
if(this.state.areas == 0) {
  message = message || {
    title: "There are currently no area
         data in the database",
    contents: "The system requires a
         list of all administrative
         subdivisions of the Philippines
          along with corresponding PSGC
         codes and geographical boundary
          geometries.",
    action: "Add Data",
    link: "areas",
  };
  hide.push(_.findWhere(tiles, {name: "
      areas"}));
}
if(this.state.elections == 0) {
  message = message || {
    title: "There are no election
         records currently stored in the
          database",
    contents: "The system requires
         election data to function.
         Election records may be
         obtained from COMELEC.",
    action: "Add Records",
    link: "elections",
  };
  hide.push(_.findWhere(tiles, {name: "
      elections"}));
  hide.push(_.findWhere(tiles, {name: "
      officials"}));
  hide.push(_.findWhere(tiles, {name: "
      families"}));
}
if(this.state.families == 0) {
  message = message || {
    title: "There are no officials
         currently assigned as members
         of any political family",
    contents: "The system requires
         political family associations
         for the data visualization.",
    action: "Add Families",
    link: "families",
  };
  hide.push(_.findWhere(tiles, {name: "
      families"}));
}
if(this.state.tag_datasets == 0){
  message = message || {
    title: "There are currently no
         political dynasty indicators",
    contents: "Political dynasty
         indicators are used as input
         for the data visualization.
         These indicators are generated
         from the political family data
         stored in the database.",
    action: "Generate Indicators",
    link: "datasets",
  };
  if(this.state.datasets == 0){
    hide.push(_.findWhere(tiles, {name:
         "datasets"}));
  }
}

if(message) {
  var mel = (
    <div className="pure-g">
      <a href={"#" + message.link}
           className="pure-u-1 tile-
           container highlight">
        <div className="tile clearfix">
          <h6>{message.title}</h6>
          <p className="text">{message.
               contents}</p>
          <button className="pull-right
               button button-primary">{
               message.action}</button>
        </div>
      </a>
    </div>
  );
}

return (
  <div className="body-panel">
    {mel}
    <div className="pure-g">
      {tiles.map(function(tile) {
        var classes = "pure-u-1-2 tile-
             container";
        if(_.indexOf(hide, tile) >= 0)
             classes += " faded";
        return <a key={tile.name} href
             ={"#" + tile.name}
             className={classes}>{tile.
             contents}</a>;
```

```
            })}
          </div>
        </div>
      );
```

*js-src/view/admin/OfficialFamilyList.jsx*

```
"use strict";
define(["react", "model/Family", "jsx!view/
    TypeaheadInput", "jsx!view/admin/
    OfficialFamilyToken", "react.backbone"],
    function(React, Family, TypeaheadInput,
    OfficialFamilyToken) {
  return React.createBackboneClass({
    filterSearch: function(data) {
      var that = this;
      return _.filter(data, function(item) {
        return that.collection().get(item.id)
            == null;
      });
    },

    render: function() {
      var that = this;

      var display =  function(item) {
        return item.name;
      };

      return (
        <div className="pure-g">
          <form className="pure-u-1" onSubmit
              ={this.handle_submit}>
            <div className="label">Families</
                div>
            <div className="input token-list-
                input clearfix" onClick={this
                .handle_focus}>
              {this.collection().map(function(
                  family) {
                return <OfficialFamilyToken
                    key={family.cid} model={
                    family} />;
              })}
              <div className="token-input"
                  onKeyDown={this.handle_key
                  }>
                <TypeaheadInput className="
                    token-input-typeahead
                    typeahead-nostyle"
                  for="Family"
                  ref="input"
                  display={display}
                  collection={this.collection
                      ()}
                  required />
              </div>
            </div>
            <div className="token-submit">
              <input className="button" type="
                  submit" value="Add" />
            </div>
          </form>
        </div>
      );
    },

    handle_key: function(e) {
      var size = this.collection().size();
      if(!e.target.value.length && e.keyCode
          == 8 && size) {
        this.collection().at(-1).destroy({wait
            : true});
      }
    },

    handle_focus: function(e) {
      setTimeout(function(){ React.findDOMNode
          (this.refs.input).focus(); }.bind(
          this), 0);
    },

    handle_submit: function(e) {
      e.preventDefault();
      var that = this;
      this.refs.input.get_or_create({
        model: Family.extend({urlRoot: this.
            collection().url()}),
        attributes: function(str) {
          return {name: that.refs.input.state.
              value};
        },
        callback: function(item, created) {
          if(created) {
            that.collection().add(item);
          }else{
            that.collection().post_family(item
                );
          }
          that.refs.input.clear();
        },
      });
    },
  });
});
```

*js-src/view/admin/OfficialFamilyToken.jsx*

```
"use strict";
define(["react", "react.backbone"], function(
    React) {
  return React.createBackboneClass({
    render: function() {
      return (
        <span className="token">
          <span>{this.model().get("name")}</
              span>
          <button className="button-close"
              type="button" onClick={this.
              handle_delete}>&times;</button>
        </span>
      );
    },

    handle_delete: function() {
      this.model().destroy({wait: true});
    },
  });
});
```

*js-src/view/admin/OfficialRow.jsx*

```
"use strict";
define(function(require) {
  var React = require("react", "react.backbone
      "),
    OfficialFamilyCollection = require("model/
        OfficialFamilyCollection"),
    SliderTransitionGroupChild = require("jsx!
        view/SliderTransitionGroupChild"),
    EditableOfficialName = require("jsx!view/
        EditableOfficialName"),
    OfficialName = require("jsx!view/
        OfficialName"),
    OfficialFamilyList = require("jsx!view/
        admin/OfficialFamilyList"),
    ClickToTopMixin = require("mixin/
        ClickToTopMixin"),
    Va = require("validator"),
    ValidationMixin = require("mixin/
        ValidationMixin"),
    ValidationMessages = require("jsx!view/
        ValidationMessages"),
    ReactTransitionGroup = React.addons.
        TransitionGroup;

  return React.createBackboneClass({
    mixins: [ValidationMixin, ClickToTopMixin
        ],

    getInitialState: function() {
      return {
        edit: this.model().isNew(),
        families: new OfficialFamilyCollection
            (null, {official_id: this.model()
            .id}),
      };
    },

    componentWillMount: function() {
      this.state.families.fetch();
    },

    getValidationSchema: function() {
      return {
```

```
      surname: Va.lidator().required().
          string(),
      name: Va.lidator().required().string()
          ,
      nickname: Va.lidator().string(),
    };
  },
  getObjectToValidate: function() {
    return {
      surname: this.refs.name.state.surname,
      name: this.refs.name.state.name,
      nickname: this.refs.name.state.
          nickname,
    };
  },
  getValidationElementMap: function() {
    return {
      surname: React.findDOMNode(this.refs.
          name.get_input_surname()),
      name: React.findDOMNode(this.refs.name
          .get_input_name()),
      nickname: React.findDOMNode(this.refs.
          name.get_input_nickname()),
    };
  },
  validationCallback: function(key, valid,
       message) {
    // React.findDOMNode(this.refs.save).
        disabled = !valid;
  },

  render: function() {
    var classes = "data-row";
    var fields = null;
    if(this.model().isNew() || this.state.
        edit) {
      classes += " edit";
      fields = (
        <form onSubmit={this.handle_save}>
          <ValidationMessages validation={
              this.state.validation} />
          <div className="pure-g form">
            <EditableOfficialName className
                ="pure-u-1" ref="name"
                model={this.model()}
                autoFocus />
          </div>
          <div className="pure-g">
            <div className="pure-u-1">
              <button className="pull-right
                  button button-primary mar
                  " type="submit" onClick={
                  this.handle_save}>Save</
                  button>
              <button className="pull-right
                  button mar" type="button"
                   onClick={this.
                  handle_cancel}>Cancel</
                  button>
            </div>
          </div>
        </form>
      );
    }else{
      fields = [
        (<div className="pure-g form">
          <OfficialName className="field
              pure-u-5-6 text-large" model
              ={this.model()} />
          <div className="pure-u-1-6">
            <button className="pull-right
                button button-flat" onClick
                ={this.handle_edit}>Edit</
                button>
          </div>
        </div>),
        (<OfficialFamilyList collection={
            this.state.families} />)
      ];
    }
    return (
      <div className={classes}>
      <ReactTransitionGroup><
          SliderTransitionGroupChild key={
          classes}>
        {fields}
      </SliderTransitionGroupChild></
          ReactTransitionGroup>
      </div>
    );
  },

  handle_edit: function() {
    this.setState({edit: true});
  },

  handle_cancel: function() {
    this.resetValidation();
    this.setState({edit: false});
  },

  handle_save: function(e) {
    e.preventDefault();
    var that = this;

    if(!this.validate()) return;

    var patch = this.model().isNew()
      ? this.refs.name.state
      : _.omit(this.refs.name.state,
          function(value, key, object) {
        return that.model().get(key) ===
            value;
      });

    if(_.isEmpty(patch)) {
      this.setState({edit: false});
    }else{
      this.model().save(patch, {
        patch: !this.model().isNew(),
        wait: true,
        success: function() {
          that.setState({edit: false});
        }
      });
    }
  },
});
});
```

*js-src/view/admin/OfficialsPanel.jsx*

```
"use strict";
define(["react", "jsx!view/SearchControls", "
    jsx!view/PageControls", "jsx!view/admin/
    OfficialRow", "mixin/ScrollToTopMixin", "
    react.backbone"], function(React,
    SearchControls, PageControls, OfficialRow
    , ScrollToTopMixin) {
  var ReactCSSTransitionGroup = React.addons.
      CSSTransitionGroup;
  return React.createBackboneClass({
    mixins: [ScrollToTopMixin],

    render: function() {
      var that = this;
      return (
        <div className="body-panel">
          {!this.empty_data() ?
          <div>
            <SearchControls className="mar"
                ref="searcher" collection={
                this.collection()} />
            <ReactCSSTransitionGroup
                transitionName="fade">
              {this.collection().map(function(
                  official) {
                return <OfficialRow key={
                    official.cid} model={
                    official} />;
              })}
            </ReactCSSTransitionGroup>
            <PageControls className="text-
                center mar" collection={this.
                collection()} onNext={this.
                scroll_to_top} onPrev={this.
                scroll_to_top} />
          </div>
          :
          <div className="text-center">
            <h1 className="transparent">No
                Data</h1>
            <a href="#elections" className="
                text text-center">Add
                election records first</a>
          </div>
          }
        </div>
      );
    },

    empty_data: function() {
      return !this.collection().size() && (!
          this.refs.searcher || this.refs.
          searcher.state.query === null) &&
          this.collection().getPage() === 0;
    },
  });
});
```

*js-src/view/admin/Sidebar.jsx*

```
"use strict";
define(["react"], function(React) {
  return React.createClass({
    componentDidMount: function() {
      this.props.bus.router.on("route", this.
          on_route);
    },

    componentWillUnmount: function() {
      this.props.bus.router.off("route", this.
          on_route);
    },

    render: function() {
      return (
        <div>
          <ul ref="menu" className="menu">
            <li ref="home"><a href="#"><i
                 className="fa fa-home fa-fw
                "/>  Home</a></li>
            <li ref="elections"><a href="#
                 elections"><i className="fa
                 fa-archive fa-fw"/> 
                 Elections</a></li>
            <li ref="officials"><a href="#
                 officials"><i className="fa
                 fa-male fa-fw"/> 
                 Officials</a></li>
            <li ref="families"><a href="#
                 families"><i className="fa fa
                 -tags fa-fw"/>  Families
                 </a></li>
            <li ref="areas"><a href="#areas"><
                 i className="fa fa-map-marker
                  fa-fw"/>  Areas</a></li
                 >
            <li ref="datasets"><a href="#
                 datasets"><i className="fa fa
                 -table fa-fw"/> 
                 Datasets</a></li>
            <li ref="users"><a href="#users"><
                 i className="fa fa-users fa-
                 fw"/>  Users</a></li>
          </ul>
        </div>
      );
    },

    on_route: function(e) {
      $(React.findDOMNode(this.refs.menu)).
          children(".active").removeClass("
          active indirect");
      var $li = $(React.findDOMNode(this.refs[
          e.route]));
      $li.addClass("active");
      if(e.params[0]) {
        $li.addClass("indirect");
      }
    },
  });
});
```

*js-src/view/admin/UserRow.jsx*

```
"use strict";
define(function(require) {
  var React = require("react", "react.backbone
      "),
    InstanceCache = require("InstanceCache"),
    Toggle = require("jsx!view/Toggle"),
    CollectionCount = require("jsx!view/
        CollectionCount"),
    ConfirmationDialog = require("jsx!view/
        ConfirmationDialog");

  return React.createBackboneClass({
    getInitialState: function() {
      return {
        datasets_count: 0,
      };
    },

    componentWillMount: function() {
      $.get("api.php/users/" + this.model().
          get("username") + "/datasets", {
          count: 1}, function(data){
        this.setState({datasets_count: data.
            total});
      }.bind(this), "json");
    },

    render: function() {
      var active = this.model().get("active");
      var username = this.model().get("
          username");
      var url_datasets = "#users/" + username
          + "/datasets";
      return (
        <div className={"data-row " + (active
            ? "active" : "inactive")}>
          <div className="pure-g">
            <div className="pure-u-1-8 pad">
              <label>
                <Toggle title="active status"
                     checked={active} disabled
                    ={this.user_in()}
                    onChange={this.
                    handle_toggle_active} />
              </label>
            </div>
            <div className="pure-u-3-8 pad
                 field text-large">{username
                }</div>
            <div className="pure-u-1-6 pad">
              <a href={url_datasets}>{this.
                  state.datasets_count}
                  datasets</a>
            </div>
            <div className="pure-u-1-6 pad
                 clearfix">
              <label className="pull-right">
                <input type="checkbox" checked
                    ={this.model().get("role
                    ")==="admin"} disabled={
                    this.user_in() || !active
                    } onChange={this.
                    handle_toggle_admin} />
                <span> admin</span>
              </label>
            </div>
            <div className="pure-u-1-6
                 clearfix">
              <button className={"pull-right
                   button button-complement" +
                    (active ? " button-flat" :
                    "")} disabled={this.
                   user_in()} onClick={this.
                   handle_delete}>Delete</
                   button>
            </div>
          </div>
        </div>
      );
    },

    user_in: function() {
      var token = InstanceCache.get_existing("
          Token", "session");
      var user = token ? token.get_user() :
          null;
      if(!user) return false;
      var username = user.get("username");
      return username == this.model().get("
          username");
    },

    handle_toggle_active: function(e) {
      if(!this.user_in()) {
        this.model().save({active: !this.model
            ().get("active")}, {wait: true});
      }
    },

    handle_toggle_admin: function(e) {
      if(!this.user_in()) {
        if(this.model().get("role") === "admin
            ") {
          this.model().save({role: "user"}, {
              wait: true});
        }else{
          this.model().save({role: "admin"}, {
              wait: true});
        }
      }
    },

    handle_delete: function(e) {
      if(!this.user_in()) {
        var that = this;
        ConfirmationDialog.open("Are you sure
            you want to delete this user and
            all of its datasets?", [
          {
            display: "Cancel",
            type: "close",
          },
```

```
          {
            display: "Delete",
            type: "secondary",
            callback: function() {
              that.model().destroy({wait: true
                  });
            },
```

```
            },
          ]);
        }
      },
    });
});
```

*js-src/view/admin/UsersPanel.jsx*

```
"use strict";
define(["react", "jsx!view/SearchControls", "
    jsx!view/PageControls", "jsx!view/admin/
    UserRow", "mixin/ScrollToTopMixin", "
    react.backbone"], function(React,
    SearchControls, PageControls, UserRow,
    ScrollToTopMixin) {
  var ReactCSSTransitionGroup = React.addons.
      CSSTransitionGroup;
  return React.createBackboneClass({
    mixins: [ScrollToTopMixin],

    render: function() {
      return (
        <div className="body-panel">
          <SearchControls className="mar"
              collection={this.collection()}
              />
          <div>
```

```
          {/*<ReactCSSTransitionGroup
              transitionName="fade">*/}
            {this.collection().map(function(
                user) {
              return <UserRow key={user.cid}
                  model={user}/>;
            }, this)}
          {/*</ReactCSSTransitionGroup>*/}
          </div>
          <PageControls className="text-center
              mar" collection={this.
              collection()} onNext={this.
              scroll_to_top} onPrev={this.
              scroll_to_top} />
        </div>
      );
    },
  });
});
```

*js-src/view/login/LoginPage.jsx*

```
"use strict";
define(function(require) {
  var $ = require("jquery"),
    React = require("react.backbone"),
    Notification = require("jsx!view/
        Notification");

  return React.createBackboneClass({
    mixins: [React.addons.LinkedStateMixin],

    getInitialState: function() {
      return {
        username: (location.search.split("
            username=")[1]||"").split("&")[0]
            || "",
      };
    },

    initialize: function() {
      this.clicked = null;
    },

    onModelChange: function() {
      if(this.props.onLogin && this.model().
          get_user()) {
        this.props.onLogin();
      }
    },

    render: function() {
      return (
        <div className="login-container">
          <div className="login-box">
            <div className="pure-g">
              <a className="pure-u-1 no-decor"
                  href=".">
                <div className="logo-large"></
                    div>
                <div className="logo-type">
                    DynastyMap</div>
              </a>
            </div>
            <form ref="form" className="pure-g
                " onSubmit={this.
                handle_submit}>
              <div className="pure-u-1">
                <input className="input" id="
                    username" type="text"
                    placeholder="Username"
                    valueLink={this.linkState
                    ("username")} required />
              </div>
              <div className="pure-u-1">
                <input className="input" id="
                    password" type="password"
                    placeholder="Password"
                    required />
              </div>
              <div className="pure-u-1 login-
                  buttons">
                <input className="pull-right
                    button button-primary"
                    name="login" type="submit
                    " onClick={this.
```

```
                    click_login} value="Sign
                    In" />
                <input className="pull-left
                    button" name="register"
                    type="submit" onClick={
                    this.click_register}
                    value="Create Account" />
              </div>
            </form>
            <div className="pure-g">
              <div className="pure-u-1">
                <a className="pull-right text-
                    small" href=".">Go to the
                    home page</a>
              </div>
            </div>
          </div>
        </div>
      );
    },

    click_login: function() {
      this.clicked = "login";
    },
    click_register: function() {
      this.clicked = "register";
    },

    handle_submit: function(e) {
      e.preventDefault();
      this["handle_" + this.clicked](e);
      this.clicked = null;
    },

    handle_register: function() {
      var $el = $(this.el());
      var username = $el.find("#username").val
          ();
      var password = $el.find("#password").val
          ();
      $.ajax({
        method: "POST",
        url: "api.php/users",
        data: JSON.stringify({username:
            username, password: password}),
        processData: false,
        dataType: "json",
        success: function(data) {
          this.model().login(username,
              password, null, function(xhr) {
            Notification.open(<span><i
                className="fa fa-exclamation-
                circle"/> Login error: {
                xhr.responseText}</span>,
                null, "error");
          });
        }.bind(this),
        error: function(xhr) {
          Notification.open(<span><p><i
              className="fa fa-exclamation-
              circle"/> Registration
              error: {xhr.responseText}</p><p
              >Your username is possibly
              already in use.</p></span>,
```

```
            null, "error");
        },
      });
    },

    handle_login: function(e) {
      e.preventDefault();
      var $el = $(this.el());
      var username = $el.find("#username").val
          ();
      var password = $el.find("#password").val
          ();
      this.model().login(username, password,
          null, function(xhr) {
        Notification.open(<span><i className="
            fa fa-exclamation-circle"/> 
            Login error: {xhr.responseText}</
            span>, null, "error");
      });
    },
  });
});
```

*js-src/view/main/AreaElectionsList.jsx*

```
"use strict";
define(function(require){
  var _ = require("underscore"),
    React = require("react", "react.backbone")
        ,
    InstanceCache = require("InstanceCache"),
    OfficialFamilyCollection = require("model/
        OfficialFamilyCollection"),
    Name = require("jsx!view/Name"),
    OfficialName = require("jsx!view/
        OfficialName"),
    OfficialFamilyList = require("jsx!view/
        main/OfficialFamilyList");

  return React.createBackboneClass({
    getInitialState: function() {
      return {
        year: new Date().getFullYear(),
      };
    },

    componentDidMount: function() {
      this.props.bus.main_settings.on("update
          ", this.on_main_settings);
    },

    componentWillUnmount: function() {
      this.props.bus.main_settings.off("update
          ", this.on_main_settings);
    },

    render: function() {
      if(this.collection()) {
        var list = this.collection().chain()
          .filter(function(e){
            return e.get("year") == this.state
                .year || e.get("year") < this
                .state.year && this.state.
                year < e.get("year_end");
          }, this)
          .map(function(e, i) {
            var official = InstanceCache.get("
                Official", e.get("official_id
                "));
            if(!official.has("name")) official
                .fetch();
            return (
              <li key={official.cid} className
                  ="area-elections-item">
                <span className="pure-u-1-12
                    number">{(i+1) + "."}</
                    span>
                <OfficialName className="pure-
                    u-11-12" model={official
                    }/>
              </li>
            );
          }, this)
          .value();
      }
      return (
        <div className="area-elections-list">
          <h4 className="title">Elected
              Officials</h4>
          {list && list.length ? <ol>{list}</
              ol> : <span>None</span>}
        </div>
      );
    },

    on_main_settings: function(settings) {
      if(settings.year) this.setState({year:
          settings.year});
    },
  });
});
```

*js-src/view/main/AreaFamiliesList.jsx*

```
"use strict";
define(function(require){
  var _ = require("underscore"),
    React = require("react", "react.backbone")
        ,
    InstanceCache = require("InstanceCache"),
    OfficialFamilyCollection = require("model/
        OfficialFamilyCollection"),
    Name = require("jsx!view/Name");

  return React.createBackboneClass({
    getInitialState: function() {
      return {
        year: new Date().getFullYear(),
        families: [],
      };
    },

    componentWillReceiveProps: function(
        nextProps) {
      if(!nextProps.collection) {
        this.setState({families: []});
      }
    },

    onModelChange: function() {
      this.setState({families: []});
      this.collection().chain()
        .filter(function(e){
          return e.get("year") == this.state.
              year || e.get("year") < this.
              state.year && this.state.year <
               e.get("year_end");
        }, this)
        .each(function(e) {
          var official_id = e.get("official_id
              ");
          var name = "OfficialFamilyCollection
              ";
          var families = InstanceCache.
              get_existing(name, official_id)
              ;
          if(families) {
            this.setState({families: this.
                state.families.concat(
                families.models)});
          }else{
            families = new
                OfficialFamilyCollection(null
                , {official_id: official_id})
                ;
            InstanceCache.set(name,
                official_id, families);
            families.fetch({
              success: function() {
                this.setState({families: this.
                    state.families.concat(
                    families.models)});
              }.bind(this),
            });
          }
        }, this);
    },

    componentDidMount: function() {
      this.props.bus.main_settings.on("update
          ", this.on_main_settings);
    },

    componentWillUnmount: function() {
      this.props.bus.main_settings.off("update
          ", this.on_main_settings);
    },

    render: function() {
      return (
        <div className="area-families-list">
          <h4 className="title">Top Local
              Families</h4>
```

```
        {this.collection() && this.
             collection().size() ? <ol>
            {_.chain(this.state.families)
              .groupBy(function(f){ return f
                   .id; })
              .pairs()
              .sortBy(function(p){ return -p
                   [1].length; })
              .map(function(p, i){
                return (
                  <li key={p[0]} className="
                       area-families-item">
                    <span className="pure-u
                         -1-6 number">{(i+1)
                          + "."}</span>
                    <span className="pure-u
                         -5-6">
                      <Name className="pure-
                           u-1-2 name" model
                           ={p[1][0]}/>
                    <span className="pure-
                         u-1-2 count">{p
                                        [1].length + "
                                        officials"}</span
                                        >
                                    </span>
                                  </li>
                                );
                              })
                              .value()
                          }
                        </ol>
                        : <span>None</span>}
                    </div>
                );
            },

            on_main_settings: function(settings) {
              if(settings.year) this.setState({year:
                   settings.year});
            },
        });
    });
```

*js-src/view/main/ChoroplethLegend.jsx*

```
"use strict";
define(function(require) {
  var _ = require("underscore"),
    $ = require("jquery"),
    numf = require("numf"),
    React = require("react");

  return React.createClass({
    statics: {
      combine_colors: function(c1, c2) {
        return _.mapObject(c1, function(v,k) {
          var w = c2[k];
          // return Math.ceil((v + w) / 2); //
               alpha
          // return Math.min(v, w); // darken
          // return Math.ceil(v * w / 255); //
               multiply

          // v = 255 * 0.2 + v * 0.8;
          // w = 255 * 0.2 + w * 0.8;
          // return Math.min(Math.ceil(v * w *
               1.5 / 255), 255); // lighten
              then multiply then brighten

          v = Math.min(v * 1.2, 255);
          w = Math.min(w * 1.2, 255);
           return Math.ceil(v * w / 255); //
               brighten then multiply
        });
      },
    },

    getInitialState: function() {
      return {data: [null, null]};
    },

    componentDidMount: function() {
      this.width = $(React.findDOMNode(this).
           parentElement).width();
      this.props.bus.choropleth_data.on("
           update", this.update);
    },

    componentWillUnmount: function() {
      this.props.bus.choropleth_data.off("
           update", this.update);
    },

    render: function() {
      var margin = 5;
      var legend_width = this.width ? this.
           width - margin * 2 : 0;
      var legend_height = 0;

      if(_.every(this.state.data)) {
        legend_height = legend_width;

        var data_x = this.state.data[0];
        var data_y = this.state.data[1];
        var min_x = data_x.classes[0];
        var min_y = data_y.classes[0];
        var range_x = data_x.classes[data_x.
             classes.length - 1] - min_x;
        var range_y = data_y.classes[data_y.
             classes.length - 1] - min_y;

        // join x & y data on area code
        var sorted_x = _.sortBy(data_x.
             datapoints, function(d){ return d
             .get("area_code"); });
        var sorted_y = _.sortBy(data_y.
             datapoints, function(d){ return d
             .get("area_code"); });
        var ix = 0, iy = 0;
        var code_x = sorted_x[ix].get("
             area_code"),
          code_y = sorted_y[iy].get("area_code
               ");
        var joined_datapoints = [];
        while(ix < sorted_x.length && iy <
             sorted_y.length) {
          if(code_x == code_y) {
            joined_datapoints.push([sorted_x[
                 ix], sorted_y[iy]]);
            ix++; iy++;
            if(ix < sorted_x.length) code_x =
                 sorted_x[ix].get("area_code")
                 ;
            if(iy < sorted_y.length) code_y =
                 sorted_y[iy].get("area_code")
                 ;
          }else if(code_x < code_y){
            ix++;
            if(ix < sorted_x.length) code_x =
                 sorted_x[ix].get("area_code")
                 ;
          }else{ // code_y < code_x
            iy++;
            if(iy < sorted_y.length) code_y =
                 sorted_y[iy].get("area_code")
                 ;
          }
        }

        var scatterplot_margin = 45;
        var scatterplot_width = legend_width -
             scatterplot_margin;
        var scatterplot_height = legend_height
             - scatterplot_margin;
        var scatterplot_points = _.chain(
             joined_datapoints)
          .map(function(pair){ return {x: pair
               [0].get("value"), y: pair[1].
               get("value")}; })
          .uniq(false, function(point){ return
               "p" +
            Math.floor(point.x *
                 scatterplot_width / range_x)
                 + "," +
            Math.floor(point.y *
                 scatterplot_height / range_y)
                 ; })
          .value();

        var g = (
          <g transform={"translate("+margin
               +","+margin+")"}>
            <g id="scatterplot-class-colors">
              {_.map(_.initial(data_x.classes)
                   , function(cx,i) {
                var x = scatterplot_margin + (
                     range_x == 0 ? 0 :
                     scatterplot_width * (cx -
                      min_x) / range_x);
                var w = range_x == 0 ?
                     scatterplot_width :
                     scatterplot_width * (
                     data_x.classes[i+1] - cx)
                      / range_x;
                var cx = data_x.color_scale[i
                     ];

                return _.map(_.rest(data_y.
                     classes), function(cy,j)
```

```
          {
        var y = scatterplot_height *
             (range_y == 0 ? 0 : (1
             - (cy - min_y) /
            range_y));
        var h = range_y == 0 ?
            scatterplot_height :
            scatterplot_height * (
            cy - data_y.classes[j])
             / range_y;
        var cy = data_y.color_scale[
            j];
        var c = this.constructor.
            combine_colors(cx, cy);

        return <rect key={j}
          className="scatterplot-
              class-color"
          x={Math.floor(x)} y={Math.
              floor(y)} width={Math
              .ceil(w)} height={
              Math.ceil(h)}
          fill={"rgb("+c.r+","+c.g
              +","+c.b+")"}/>;
      }, this);
    }, this)}
  </g>
  <g id="scatterplot-point-shadows">
    {_.map(scatterplot_points,
        function(point, i) { //
        shadows
      var x = scatterplot_margin + (
          range_x == 0 ? 0 :
          scatterplot_width * (
          point.x - min_x) /
          range_x);
      var y = (range_y == 0 ? 0 :
          scatterplot_height * (1 -
           (point.y - min_y) /
          range_y));
      return <circle key={i}
        className="scatterplot-point
            -shadow"
        cx={x} cy={y} r={4}/>
    }, this)}
  </g>
  <g id="scatterplot-points">
    {_.map(scatterplot_points,
        function(point, i) {
      var x = scatterplot_margin + (
          range_x == 0 ? 0 :
          scatterplot_width * (
          point.x - min_x) /
          range_x);
      var y = (range_y == 0 ? 0 :
          scatterplot_height * (1 -
           (point.y - min_y) /
          range_y));
      return <circle key={i}
        className="scatterplot-point
            "
        cx={x} cy={y} r={3}/>
    }, this)}
  </g>
  <g id="scatterplot-labels-x"
      transform={"translate("+
      scatterplot_margin+","+
      scatterplot_height+")"}>
    {_.map(data_x.classes, function(
        c,i) {
      var x = scatterplot_width * (
          range_x == 0 ? 0 : (c -
          min_x) / range_x);
      return <text key={i}
        className="scatterplot-label
            scatterplot-label-x"
        x={x} y={0}>
        {numf.format(c)}
      </text>;
    }, this)}
  </g>
  <g id="scatterplot-title-x">
    <text className="scatterplot-
        title scatterplot-title-x"
        x={scatterplot_margin +
        scatterplot_width/2} y={
        legend_height}>{data_x.name
        }</text>
  </g>
  <g id="scatterplot-labels-y"
      transform={"translate("+
      scatterplot_margin+","+0+")
      "}>
    {_.map(data_y.classes, function(
        c,i) {
      var y = (range_y == 0 ?
          scatterplot_height :
          scatterplot_height * (1 -
           (c - min_y) / range_y));
          return <text key={i}
            className="scatterplot-label
                scatterplot-label-y"
            x={0} y={y}>
            {numf.format(c)}
          </text>;
        }, this)}
      </g>
      <g id="scatterplot-title-y">
        <text className="scatterplot-
            title scatterplot-title-y"
            x={0} y={scatterplot_height
            /2}>{data_y.name}</text>
      </g>
    </g>
  );
}else{
  var index = _.findIndex(this.state.
      data);
  if(index >= 0) {
    legend_height = legend_width / 2;

    var dataset = this.state.data[index
        ];

    var min = dataset.classes[0];
    var max = dataset.classes[dataset.
        classes.length - 1];

    // Freedmain-Diaconis rule for
        optimal number of bins in a
        histogram
    var bin_size = 2 * this.iqr(dataset.
        datapoints) * Math.pow(dataset.
        datapoints.length, -1/3);

    var counts = _.countBy(dataset.
        datapoints, function(p) {
      return Math.floor((p.get("value")
          - min) / bin_size);
    });
    var min_bin = Math.floor(min /
        bin_size);
    var max_bin = Math.floor(max /
        bin_size);
    var bins = 1 + max_bin - min_bin;

    var min_count = 0;//_.min(counts);
    var max_count = _.max(counts);

    var histogram_width = legend_width;
    var histogram_height = legend_height
        - 60;
    var bar_height = 20;

    var g = (
      <g transform={"translate("+margin
          +","+margin+")"}>
        <g id="histogram-bars">
          {_.map(_.pairs(counts),
              function(m) {
            var bin = m[0];
            var count = m[1];
            var w = histogram_width /
                bins;
            var h = histogram_height * (
                count - min_count) / (1
                + max_count -
                min_count);
            return <rect key={m}
              className="histogram-bar"
              x={Math.floor(w * bin)} y
                  ={histogram_height -
                  h}
              width={Math.ceil(w)}
                  height={h} />;
          })}
        </g>
        <g id="histogram-class-colors"
            transform={"translate
            ("+0+","+histogram_height
            +")"}>
          {_.map(_.initial(dataset.
              classes), function(c,i) {
            var range = max - min;
            var x = range == 0 ? 0 :
                legend_width * (c - min
                ) / range;
            var w = range == 0 ?
                legend_width :
                legend_width * (dataset
                .classes[i+1] - c) /
                range;
            var c = dataset.color_scale[
                i];
            return <rect key={i}
              className="histogram-class
                  -color"
```

```
                    x={x} width={w} height={
                        bar_height}
                    fill={"rgb("+c.r+","+c.g
                        +","+c.b+")"}/>;
                }, this)}
              </g>
              <g id="histogram-labels"
                  transform={"translate
                  (")+0+","+histogram_height
                  +")"}>
                {_.map(dataset.classes,
                    function(c,i) {
                  var range = max - min;
                  var x = range == 0 ? 0 :
                      legend_width * (c - min
                      ) / range;
                  return <text key={i}
                    className="histogram-label
                        "
                    x={x} y={bar_height}>
                    {numf.format(c)}
                  </text>;
                }, this)}
              </g>
              <g id="histogram-title">
                <text className="histogram-
                    title" x={legend_width/2}
                    y={legend_height}>{
                    dataset.name}</text>
              </g>
            </g>
          );
        }
      }
```

*js-src/view/main/ChoroplethSettingsPane.jsx*

```
"use strict";
define(function(require) {
  var _ = require("underscore"),
    React = require("react"),
    DatasetCollection = require("model/
        DatasetCollection"),
    Modal = require("jsx!view/Modal"),
    DatasetChooser = require("jsx!view/main/
        DatasetChooser"),
    ChoroplethLegend = require("jsx!view/main/
        ChoroplethLegend");

  return React.createClass({
    getInitialState: function() {
      return {
        dataset1: null,
        dataset2: null,
      };
    },

    shouldComponentUpdate: function(nextProps,
        nextState) {
      return !_.isEqual(nextProps, this.props)
          || !_.isEqual(nextState, this.
          state);
    },

    componentWillUpdate: function(nextProps,
        nextState) {
      this.props.bus.choropleth_settings.emit
          ("update", {
        dataset1: nextState.dataset1,
        dataset2: nextState.dataset2,
      });
    },

    render: function() {
      var datasets_list = [];
      var datasets = [this.state.dataset1,
          this.state.dataset2];
      for (var i = 0; i < datasets.length; i
          ++) {
        var dataset = datasets[i];
        if(dataset) {
          datasets_list.push(
            <div key={i} className="pure-u-1
                group group-no-table">
              <button className="pure-u-5-6
                  group-component one-line
                  button" onClick={this.
                  select_handler(i+1)}
                  disabled={this.props.
                  disabled}>{dataset.get("
                  name")}</button>
              <button className="pure-u-1-6
                  group-component button"
                  onClick={this.
                  remove_handler(i+1)}
                  disabled={this.props.
                  disabled}><i className="fa
                  fa-close"/></button>
            </div>
          );
        }else{
          datasets_list.push(
            <div key={i} className="pure-u-1">
              <button className="pure-u-1
                  button" onClick={this.
                  select_handler(i+1)}
                  disabled={this.props.
                  disabled}>Select Dataset</
                  button>
            </div>
          );
        }
      }
      return (
        <div className="pane">
          <h6 className="pane-header">
              Choropleth Layer</h6>
          <div className="pane-content">
            {datasets_list}
            <ChoroplethLegend bus={this.props.
                bus}/>
          </div>
        </div>
      );
    },

    remove_handler: function(i) {
      var that = this;
      return function() {
        that.select_dataset(i, null);
      };
    },

    select_handler: function(i) {
      var that = this;
      return function() {
        var dataset_collection = new
            DatasetCollection();
        dataset_collection.fetch({data: {type:
            "area"}});
        that.modal = Modal.open("Select a
            dataset to visualize", (
          <DatasetChooser collection={
              dataset_collection} onSelect={
              function(dataset) {
            that.select_dataset(i, dataset);
            that.modal.close();
            that.modal = null;
          }}/>
        ));
      };
    },

    select_dataset: function(i, dataset) {
      var s = {}
      s["dataset" + i] = dataset;
      this.setState(s);
    },

    return <svg width={legend_width + margin
        * 2} height={legend_height +
        margin * 2}>{g}</svg>;
  },

  update: function(data) {
    this.setState({data: data});
  },

  iqr: function(datapoints) {
    var values = _.chain(datapoints)
      .filter(function(p){ return p.get("
          value") !== null; })
      .map(function(p){ return p.get("value
          "); })
      .sortBy()
      .value();

    var q1i = (values.length - 1) * 0.25;
    var v1 = values[Math.floor(q1i)];
    var v1n = values[Math.floor(q1i) + 1];
    var q1 = v1 + (v1n - v1) * (q1i % 1);

    var q3i = (values.length - 1) * 0.75;
    var v3 = values[Math.floor(q3i)];
    var v3n = values[Math.floor(q3i) + 1];
    var q3 = v3 + (v3n - v3) * (q3i % 1);

    return q3 - q1;
  },
});
});
```

```
    },
  });
```

*js-src/view/main/DatapointRow.jsx*

```
"use strict";
define(function(require) {
  var _ = require("underscore"),
    React = require("react", "react.backbone")
        ,
    InstanceCache = require("InstanceCache"),
    SliderTransitionGroupChild = require("jsx!
        view/SliderTransitionGroupChild"),
    Name = require("jsx!view/Name"),
    TypeaheadInput = require("jsx!view/
        TypeaheadInput"),
    ClickToTopMixin = require("mixin/
        ClickToTopMixin"),
    Va = require("validator"),
    ValidationMixin = require("mixin/
        ValidationMixin"),
    ValidationMessages = require("jsx!view/
        ValidationMessages"),
    Notification = require("jsx!view/
        Notification"),
    ReactTransitionGroup = React.addons.
        TransitionGroup;

  return React.createBackboneClass({
    mixins: [React.addons.LinkedStateMixin,
        ValidationMixin, ClickToTopMixin],

    getInitialState: function() {
      return {
        edit: this.model().isNew(),
        year: this.model().get("year"),
        value: this.model().get("value"),
      };
    },

    getValidationSchema: function() {
      return {
        area: Va.lidator().required().object()
            ,
        year: Va.lidator().required().
            integerish(),
        value: Va.lidator().required().
            floatish(),
      };
    },
    getObjectToValidate: function() {
      return {
        area: this.refs.area.state.selected,
        year: this.state.year,
        value: this.state.value,
      };
    },
    getValidationElementMap: function() {
      return {
        area: React.findDOMNode(this.refs.area
            ),
        year: React.findDOMNode(this.refs.year
            ),
        value: React.findDOMNode(this.refs.
            value),
      };
    },
    validationCallback: function(key, valid,
        message) {
      // React.findDOMNode(this.refs.save).
          disabled = !valid;
    },

    render: function() {
      var display = function(item) {
        return item.name;
      };
      var area_code = this.model().get("
          area_code");
      var area = area_code ? InstanceCache.get
          ("Area", area_code, true) : null;

      if(this.model().isNew() || this.state.
          edit) {
        if(!this.model().isNew()){
          var cancel_button = <button
              className="pull-right button
              mar" onClick={this.
              handle_cancel}>Cancel</button>;
        }
        return (
          <div className="edit data-row form">
          <ReactTransitionGroup><
              SliderTransitionGroupChild key
              ="edit">
            <ValidationMessages validation={
                this.state.validation} />
            <div className="pure-g">
              <label className="pure-u-1-3 pad
                  ">
                <TypeaheadInput className="
                    pure-u-1"
                  for="Area"
                  ref="area"
                  display={display}
                  model={area}
                  placeholder="Area"
                  required autoFocus />
              </label>
              <label className="pure-u-1-3 pad
                  ">
                <input ref="year" className="
                    pure-u-1" type="number"
                    placeholder="Year"
                    valueLink={this.linkState
                    ("year")} required />
              </label>
              <label className="pure-u-1-3 pad
                  ">
                <input ref="value" className="
                    pure-u-1" type="number"
                    placeholder="Value"
                    valueLink={this.linkState
                    ("value")} required />
              </label>
            </div>
            <div className="pure-g">
              <div className="pure-u-1">
                <button className="pull-right
                    button button-primary mar
                    " onClick={this.
                    handle_save}>Save</button
                    >
                {cancel_button}
                <button className="pull-left
                    button button-complement
                    mar" onClick={this.
                    handle_delete}>Delete</
                    button>
              </div>
            </div>
          </SliderTransitionGroupChild></
              ReactTransitionGroup>
          </div>
        );
      }else{
        return (
          <div className="data-row clicky form
              " onClick={this.handle_edit}>
          <ReactTransitionGroup><
              SliderTransitionGroupChild key
              ="display">
            <button className="pure-g full-
                button">
              <div className="pure-u-1-2">
                <Name className="field pad"
                    model={area} />
              </div>
              <div className="pure-u-1-4">
                <span className="field pad">{
                    this.model().get("year")
                    }</span>
              </div>
              <div className="pure-u-1-4">
                <span className="field pad">{
                    this.model().get("value")
                    }</span>
              </div>
            </button>
          </SliderTransitionGroupChild></
              ReactTransitionGroup>
          </div>
        );
      }
    },

    handle_edit: function() {
      this.setState({edit: true});
    },

    handle_cancel: function() {
      this.resetValidation();
      this.setState({edit: false});
    },

    handle_save: function() {
      var that = this;

      if(!this.validate()) return;
```

```
      var new_attributes = {
        year: parseInt(this.state.year, 10),
        area_code: this.refs.area.state.
            selected.code,
        value: parseFloat(this.state.value),
      };

      var patch = this.model().isNew()
        ? new_attributes
        : _.omit(new_attributes, function(
            value, key, object) {
          return that.model().get(key) ===
              value;
        });

      if(_.isEmpty(patch)) {
        this.setState({edit: false});
      }else{
        this.model().save(patch, {
          patch: !this.model().isNew(),
          wait: true,
          success: function() {
            that.setState({edit: false});
          },
          error: function(m,r,o) {
            Notification.open(<span><i
                className="fa fa-exclamation-
                circle"/> Save error: {r
                .responseText}</span>, null,
                "error");
          },
        });
      }
    },

    handle_delete: function(e) {
      this.model().destroy({wait: true});
    },
  });
});
```

*js-src/view/main/DatapointsPanel.jsx*

```
"use strict";
define(["react", "model/Dataset", "jsx!view/
    FileInput", "jsx!view/SearchControls", "
    jsx!view/PageControls", "jsx!view/
    PanelToolbar", "jsx!view/main/
    DatapointRow", "mixin/ScrollToTopMixin",
    "jsx!view/Notification", "react.backbone
    "], function(React, Dataset, FileInput,
    SearchControls, PageControls,
    PanelToolbar, DatapointRow,
    ScrollToTopMixin, Notification) {
  var ReactCSSTransitionGroup = React.addons.
      CSSTransitionGroup;
  return React.createBackboneClass({
    mixins: [React.addons.LinkedStateMixin,
        ScrollToTopMixin],

    getInitialState: function() {
      return {
        name: this.model().get("name"),
        description: this.model().get("
            description"),
      };
    },

    componentDidMount: function() {
      this.onModelChange();
    },
    onModelChange: function() {
      if(this.refs.toolbar) {
        if(this.empty_data()) this.refs.
            toolbar.open();
        else this.refs.toolbar.close();
      }
      this.forceUpdate();
    },

    render: function() {
      var table_header = (
        <div className="data-table-header">
          <div className="pure-g">
            <div className="pure-u-1-2 pad">
                Area</div>
            <div className="pure-u-1-4 pad">
                Year</div>
            <div className="pure-u-1-4 pad">
                Value</div>
          </div>
        </div>
      );
      return (
        <div className="body-panel">
          <div className="clearfix">
            <a className="pull-left button
                button-complement" href="#
                datasets">Back to Datasets</a
                >
          </div>
          <PanelToolbar ref="toolbar"
              toggle_text="Add Data">
            <div className="pure-u-1-2 text-
                center pad">
              <h6>Add single row</h6>
              <button className="button"
                  onClick={this.handle_add}>
                  Add Row</button>
            </div>
            <div className="pure-u-1-2 text-
                center pad">
              <form ref="form" onSubmit={this.
                  handle_upload}>
                <h6>Upload additional
                    datapoints</h6>
                <div className="label">CSV
                    file</div>
                <div><FileInput ref="file"
                    type="file" /></div>
                <input className="button
                    button-primary" type="
                    submit" value="Upload" />
              </form>
            </div>
          </PanelToolbar>
          {!this.empty_data() ?
          <div>
            {/*<SearchControls ref="searcher"
                className="mar" collection={
                this.collection()} />*/}
            <PageControls className="text-
                center mar" collection={this.
                collection()} />
            {table_header}
            <ReactCSSTransitionGroup
                transitionName="fade">
              {this.collection().map(function(
                  datapoint) {
                return <DatapointRow key={
                    datapoint.cid} model={
                    datapoint} />;
              })}
            </ReactCSSTransitionGroup>
            <PageControls className="text-
                center mar" collection={this.
                collection()} onNext={this.
                scroll_to_top} onPrev={this.
                scroll_to_top} />
          </div>
          : null}
        </div>
      );
    },

    empty_data: function() {
      return !this.collection().size() && /*(!
          this.refs.searcher || this.refs.
          searcher.state.query === null) &&
          */this.collection().getPage() ===
          0;
    },

    handle_add: function() {
      var that = this;
      if(/*this.refs.searcher.state.query ===
          null && */this.collection().getPage
          () === 0) {
        this.refs.toolbar.close();
        this.collection().add({}, {at: 0});
      }else{
        /*this.refs.searcher.set_query(null, {
          complete: function() {
            that.refs.toolbar.close();
            that.collection().add({}, {at: 0})
                ;
          },
        });*/
        this.collection().page(0, {
          complete: function() {
            if(that.refs.toolbar) that.refs.
                toolbar.close();
            that.collection().add({}, {at: 0})
                ;
          },
        });
      }
    },
```

```
    handle_upload: function(e) {
      var that = this;
      e.preventDefault();

      var fd = new FormData();
      var file = this.refs.file.get_input().
          files[0];
      fd.append("file", file);

      var notif = Notification.open(<span><i
          className="fa fa-circle-o-notch fa-
          spin"/> Uploading {this.refs.
          file.get_filename()}...</span>, 0);

      $.ajax({
        url: this.model().url() + "/datapoints
            ",
        data: fd,
        processData: false,
        contentType: false,
        type: "POST",
        success: function(data){
          if(that.refs.form) React.findDOMNode
              (that.refs.form).reset();
          if(that.refs.toolbar) that.refs.
              toolbar.close();
          that.collection().fetch();
          Notification.replace(notif, <span><i
              className="fa fa-check-circle
              "/> Uploaded datapoints: {
              that.refs.file.get_filename()
              }</span>, null, "success");
        },
        error: function(xhr) {
          Notification.replace(notif, <span><i
              className="fa fa-exclamation-
              circle"/> Datapoints
              upload error: {that.refs.file.
              get_filename()}: {xhr.
              responseText}</span>, null, "
              error");
        },
      });
    },
  });
});
```

*js-src/view/main/DatasetBox.jsx*

```
"use strict";
define(function(require) {
  var _ = require("underscore"),
    React = require("react", "react.backbone")
        ,
    SliderTransitionGroupChild = require("jsx!
        view/SliderTransitionGroupChild"),
    EditableName = require("jsx!view/
        EditableName"),
    ClickToTopMixin = require("mixin/
        ClickToTopMixin"),
    Va = require("validator"),
    ValidationMixin = require("mixin/
        ValidationMixin"),
    ValidationMessages = require("jsx!view/
        ValidationMessages"),
    Notification = require("jsx!view/
        Notification"),
    ConfirmationDialog = require("jsx!view/
        ConfirmationDialog"),
    ReactTransitionGroup = React.addons.
        TransitionGroup;

  return React.createBackboneClass({
    mixins: [React.addons.LinkedStateMixin,
        ValidationMixin, ClickToTopMixin],

    getInitialState: function() {
      return {
        edit: this.model().isNew(),
        description: this.model().get("
            description"),
      };
    },

    getValidationSchema: function() {
      return {
        name: Va.lidator().required().string()
            ,
        description: Va.lidator().required().
            string(),
      };
    },
    getObjectToValidate: function() {
      return {
        name: this.refs.name.state.name,
        description: this.state.description,
      };
    },
    getValidationElementMap: function() {
      return {
        name: React.findDOMNode(this.refs.name
            ),
        description: React.findDOMNode(this.
            refs.description),
      };
    },
    validationCallback: function(key, valid,
        message) {
      // React.findDOMNode(this.refs.save).
          disabled = !valid;
    },

    render: function() {
      var fields = null;
      if(this.model().isNew() || this.state.
          edit) {
        if(!this.model().isNew()){
          var cancel_button = <button
              className="pull-right button
              mar" onClick={this.
              handle_cancel}>Cancel</button>;
        }
        return (
          <div className="edit data-row form">
          <ReactTransitionGroup><
              SliderTransitionGroupChild key
              ="edit">
            <ValidationMessages validation={
                this.state.validation} />
            <div className="pure-g">
              <label className="pure-u-1 pad">
                <div className="label">Name</
                    div>
                <EditableName className="pure-
                    u-1" ref="name" model={
                    this.model()} autoFocus
                    />
              </label>
              <label className="pure-u-1 pad">
                <div className="label">
                    Description</div>
                <textarea ref="description"
                    className="pure-u-1 text"
                     valueLink={this.
                    linkState("description")}
                     required />
              </label>
            </div>
            <div className="pure-g">
              <div className="pure-u-1">
                <button className="pull-right
                    button button-primary mar
                    " onClick={this.
                    handle_save}>Save</button
                    >
                {cancel_button}
                <button className="pull-left
                    button button-complement
                    mar" onClick={this.
                    handle_delete}>Delete</
                    button>
              </div>
            </div>
          </SliderTransitionGroupChild></
              ReactTransitionGroup>
          </div>
        );
      }else{
        return (
          <div className="data-row form">
          <ReactTransitionGroup><
              SliderTransitionGroupChild key
              ="display">
            <div className="pure-g">
              <div className="pure-u-5-6">
                <div className="field text-
                    large">{this.model().get
                    ("name")}</div>
                <div className="field-group">
                  <span className="pure-u-1-6
                      label">Range</span>
                  <span className="pure-u-5-6
                      field">
                    {this.model().get("
                        min_year")}-{this.
                        model().get("max_year
                        ")}
                  </span>
                </div>
```

```
                <div className="field-group">
                  <span className="pure-u-1-6
                       label">Level</span>
                  <span className="pure-u-5-6
                       field">
                    {_.chain(this.model().get
                         ("contained_levels"))
                      .pick(function(v){
                           return v; }).keys()
                      .map(function(k) {
                        return {
                          "region": "Regional
                               ",
                          "province": "
                               Provincial",
                          "municipality": "
                               Municipal",
                          "barangay": "
                               Barangay",
                        }[k];
                      })
                      .values().join(", ")}
                  </span>
                </div>
                <div className="pure-u-1 field
                     text"><pre>{this.model()
                    .get("description")}</pre
                    ></div>
              </div>
              <div className="pure-u-1-6">
                <button className="pull-right
                     button button-flat"
                     onClick={this.handle_edit
                     }>Edit</button>
                <a href={"#datasets/" + this.
                     model().get("username") +
                      "/" + this.model().id}>
                  <button className="pull-
                       right button button-
                       flat">Update</button>
                </a>
              </div>
            </div>
          </SliderTransitionGroupChild></
               ReactTransitionGroup>
        </div>
      );
    }
  },

  handle_edit: function() {
    this.setState({edit: true});
  },

  handle_cancel: function() {
    this.resetValidation();
    this.setState({edit: false});
  },

  handle_save: function() {
    var that = this;

    if(!this.validate()) return;

    var new_attributes = {
      name: this.refs.name.state.name,
      description: this.state.description,
    };

    var patch = this.model().isNew()
      ? new_attributes
      : _.omit(new_attributes, function(
          value, key, object) {
        return that.model().get(key) ===
            value;
      });

    if(_.isEmpty(patch)) {
      this.setState({edit: false});
    }else{
      this.model().save(patch, {
        patch: !this.model().isNew(),
        wait: true,
        success: function() {
          that.setState({edit: false});
        },
        error: function(m,r,o) {
          Notification.open(<span><i
               className="fa fa-exclamation-
               circle"/> Save error: {r
               .responseText}</span>, null,
               "error");
        },
      });
    }
  },

  handle_delete: function(e) {
    var that = this;
    ConfirmationDialog.open("Are you sure
         you want to delete this dataset?",
         [
      {
        display: "Cancel",
        type: "close",
      },
      {
        display: "Delete",
        type: "secondary",
        callback: function() {
          var name = that.model().get("name
               ");
          var notif = Notification.open(<
               span><i className="fa fa-
               circle-o-notch fa-spin"/>&
               ensp;Deleting {name}...</span
               >, 0);
          that.model().destroy({
            wait: true,
            success: function(){
              Notification.replace(notif, <
                   span><i className="fa fa-
                   trash"/> {name}
                   deleted</span>);
            },
            error: function(xhr) {
              Notification.replace(notif, <
                   span><i className="fa fa-
                   exclamation-circle"/>&
                   ensp;Delete error: {xhr.
                   responseText}</span>,
                   null, "error");
            },
          });
        },
      },
    ]);
  },
});
});
```

*js-src/view/main/DatasetChoice.jsx*

```
"use strict";
define(["require", "react.backbone"], function
     (require) {
  var _ = require("underscore"),
    Backbone = require("backbone"),
    React = require("react", "react.backbone")
        ;

  return React.createBackboneClass({
    render: function() {
      if(this.props.selected) {
        var info = [
          (<div className="field-group">
            <span className="pure-u-1-6 label
                 ">Uploaded by</span>
            <span className="pure-u-5-6 field
                 ">{this.model().get("username
                 ")}</span>
          </div>),
          (<div className="field-group">
            <span className="pure-u-1-6 label
                 ">Range</span>
            <span className="pure-u-5-6 field
                 ">
              {this.model().get("min_year")}-{
                   this.model().get("max_year
                   ")}
            </span>
          </div>),
          (<div className="field-group">
            <span className="pure-u-1-6 label
                 ">Level</span>
            <span className="pure-u-5-6 field
                 ">
              {_.chain(this.model().get("
                   contained_levels"))
                .pick(function(v){ return v;
                     }).keys()
                .map(function(k) {
                  return {
                    "region": "Regional",
                    "province": "Provincial",
                    "municipality": "Municipal
                         ",
                    "barangay": "Barangay",
                  }[k];
                })
                .values().join(", ")}
```

```
            </span>
          </div>),
        ];
      }
      return (
        <div className={"pure-g dataset-
            chooser-item" + (this.props.
            selected ? " selected" : "")}
            onClick={this.props.onClick}>
          <div className="pure-u-1">
              <div className="field text-large
                  ">{this.model().get("name")
                  }</div>
              {info}
              <div className="pure-u-1 field
                  text"><pre>{this.model().
                  get("description")}</pre></
                  div>
          </div>
        </div>
      );
    },
  });
});
```

*js-src/view/main/DatasetChooser.jsx*

```
"use strict";
define(function(require) {
  var React = require("react", "react.backbone
      "),
    SearchControls = require("jsx!view/
        SearchControls"),
    DatasetChoice = require("jsx!view/main/
        DatasetChoice");

  return React.createBackboneClass({
    getInitialState: function() {
      return {
        selected: null,
      };
    },

    render: function() {
      var that = this;
      return (
        <div>
          <div className="clearfix">
            <a className="pull-left button
                button-flat" href="#datasets"
                 onClick={this.handle_close}>
                Upload Own Dataset</a>
          </div>
          <SearchControls collection={this.
              collection()}/>
          <div className="dataset-chooser-list
              -container">
            <div className="dataset-chooser-
                list">
              {this.collection().map(function(
                  dataset) {
                return <DatasetChoice key={
                    dataset.cid} model={
                    dataset} selected={that.
                    state.selected===dataset}
                     onClick={function(){
                        that.click_dataset(
                        dataset); }}/>;
              })}
            </div>
            <div className="scroll-edge-fade
                "></div>
          </div>
          <div className="clearfix">
            <button className="pull-right
                button button-primary"
                onClick={this.handle_select}
                disabled={!this.state.
                selected}>Select</button>
          </div>
        </div>
      );
    },

    click_dataset: function(dataset) {
      if(this.state.selected === dataset) {
        // double-click
        this.props.onSelect(this.state.
            selected);
      }else{
        this.setState({selected: dataset});
      }
    },

    handle_close: function() {
      this.props.onSelect(null);
    },

    handle_select: function() {
      this.props.onSelect(this.state.selected)
          ;
    },
  });
});
```

*js-src/view/main/DatasetsPanel.jsx*

```
"use strict";
define(function(require) {
  var React = require("react", "react.backbone
      "),
    Dataset = require("model/Dataset"),
    FileInput = require("jsx!view/FileInput"),
    SearchControls = require("jsx!view/
        SearchControls"),
    PageControls = require("jsx!view/
        PageControls"),
    DatasetBox = require("jsx!view/main/
        DatasetBox"),
    Notification = require("jsx!view/
        Notification"),
    ReactCSSTransitionGroup = React.addons.
        CSSTransitionGroup;

  return React.createBackboneClass({
    mixins: [React.addons.LinkedStateMixin],

    getInitialState: function() {
      return {
        name: null,
        description: null,
      };
    },

    componentDidMount: function() {
      this.onModelChange();
    },
    onModelChange: function() {
      if(this.empty_data()) {
        if(this.refs.toolbar) this.refs.
            toolbar.open();
      }
      this.forceUpdate();
    },

    render: function() {
      return (
        <div className="body-panel">
          <div className="clearfix">
            <a className="pull-left button
                button-complement" href="#">
                Back to Map</a>
          </div>
          <div className="pure-u-1 pad">
            <form ref="form" onSubmit={this.
                handle_upload}>
              <h6>Upload dataset</h6>
              <div className="pure-g">
                <div className="pure-u-1-2 pad
                    ">
                  <div className="label">Name
                      </div>
                  <input className="input" ref
                      ="name" type="text"
                      valueLink={this.
                      linkState("name")}
                      required />
                </div>
                <div className="pure-u-1-2 pad
                    ">
                  <div className="label">CSV
                      file</div>
                  <div><FileInput ref="file"
                      type="file" /></div>
                </div>
              </div>
              <div className="pure-g">
                <div className="pure-u-1 pad">
                  <div className="label">
                      Description</div>
                  <textarea className="input
                      text" ref="description"
                       valueLink={this.
                      linkState("description
                      ")} required />
```

```
                </div>
              </div>
              <div className="pure-g">
                <div className="pure-u-1 pad">
                  <input className="pull-right
                        button button-primary"
                        type="submit" value="
                       Upload" />
                </div>
              </div>
            </form>
          </div>
          {!this.empty_data() ?
          <div>
            <SearchControls ref="searcher"
                 className="mar" collection={
                 this.collection()} />
            <ReactCSSTransitionGroup
                 transitionName="fade">
              {this.collection().map(function(
                   dataset) {
                return <DatasetBox key={
                     dataset.cid} model={
                     dataset} />;
              })}
            </ReactCSSTransitionGroup>
            <PageControls className="text-
                 center mar" collection={this.
                 collection()} onNext={this.
                 scroll_to_top} onPrev={this.
                 scroll_to_top} />
          </div>
          : null}
        </div>
      );
    },

    empty_data: function() {
      return !this.collection().size() && (!
           this.refs.searcher || this.refs.
           searcher.state.query === null) &&
           this.collection().getPage() === 0;
    },

    handle_upload: function(e) {
      var that = this;
      e.preventDefault();

      if(!this.state.name) {
        console.error("No name");
        return;
      }

      var fd = new FormData();
      var file = this.refs.file.get_input().
           files[0];
      fd.append("file", file);

      var username = this.collection().
           username;
      var dataset = new Dataset({
        username: username,
        name: this.state.name,
        type: "area",
        description: this.state.description,
      });

      var notif = Notification.open(<span><i
           className="fa fa-circle-o-notch fa-
           spin"/> Creating dataset...</
           span>, 0);

      dataset.save(null, {
        success: function(model) {
          if(!file) {
            if(that.refs.form) React.
                 findDOMNode(that.refs.form).
                 reset();
            that.collection().fetch();
            that.setState(that.getInitialState
                 ());
            Notification.replace(notif, <span
                 ><i className="fa fa-check-
                 circle"/> Empty dataset
                 created</span>, null, "
                 success");
            window.location.href = "#datasets
                 /" + username + "/" + model.
                 id;
          }else{
            Notification.replace(notif, <span
                 ><i className="fa fa-circle-o
                 -notch fa-spin"/> 
                 Uploading {that.refs.file.
                 get_filename()}...</span>, 0)
                 ;
            $.ajax({
              url: dataset.url() + "/
                   datapoints",
              data: fd,
              processData: false,
              contentType: false,
              type: "POST",
              success: function(data){
                if(that.refs.form) React.
                     findDOMNode(that.refs.
                     form).reset();
                that.collection().fetch();
                that.setState(that.
                     getInitialState());
                Notification.replace(notif, <
                     span><i className="fa fa-
                     check-circle"/> 
                     Uploaded dataset: {that.
                     refs.file.get_filename()
                     }</span>, null, "success
                     ");
              },
              error: function(xhr) {
                dataset.destroy();
                Notification.replace(notif, <
                     span><i className="fa fa-
                     exclamation-circle"/>&
                     ensp;Datapoints upload
                     error: {that.refs.file.
                     get_filename()}: {xhr.
                     responseText}</span>,
                     null, "error");
              },
            });
          }
        },
        error: function(m,r,o) {
          Notification.replace(notif, <span><i
               className="fa fa-exclamation-
               circle"/> Dataset upload
               error: {r.responseText}</span>,
               null, "error");
        },
      });
    },
  });
});
```

*js-src/view/main/Infobar.jsx*

```
"use strict";
define(function(require) {
  var _ = require("underscore"),
    numf = require("numf"),
    React = require("react"),
    InstanceCache = require("InstanceCache"),
    AreaElectionCollection = require("model/
         AreaElectionCollection"),
    Name = require("jsx!view/Name"),
    AreaElectionsList = require("jsx!view/main
         /AreaElectionsList"),
    AreaFamiliesList = require("jsx!view/main/
         AreaFamiliesList"),
    ChoroplethLegend = require("jsx!view/main/
         ChoroplethLegend");

  return React.createClass({
    getInitialState: function() {
      return {
        selected: null,
        data: [null, null],
      };
    },

    componentDidMount: function() {
      this.props.bus.router.on("route", this.
           on_route);
      this.props.bus.main_settings.on("select
           ", this.on_select);
      this.props.bus.main_settings.on("update
           ", this.on_main_settings);
      this.props.bus.choropleth_data.on("
           update", this.on_choropleth_data);
    },

    componentWillUnmount: function() {
      this.props.bus.router.off("route", this.
           on_route);
      this.props.bus.main_settings.off("select
           ", this.on_select);
      this.props.bus.main_settings.off("update
           ", this.on_main_settings);
      this.props.bus.choropleth_data.off("
           update", this.on_choropleth_data);
```

```
},

render: function() {
  var area = null, elections = null;
  if(this.state.selected && this.state.
      selected.area_code) {
    area = InstanceCache.get("Area", this.
        state.selected.area_code, true);
    elections = new AreaElectionCollection
        (null, {area: area});
    elections.fetch();

    var variables = _.map(_.filter(this.
        state.data), function(d) {
      var datapoints = this.
          filter_datapoints(d.datapoints,
           area.get("code"));
      return {
        value: datapoints.length ?
            datapoints[0].get("value") :
            null,
        dataset: d,
      };
    }, this);
    var bg = this.get_color(variables);
    if(bg) {
      var fg = this.get_text_color(bg);
      var titlebar_style = {
        backgroundColor: "rgb("+bg.r+","+
            bg.g+","+bg.b+")",
        color: "rgb("+fg.r+","+fg.g+","+fg
            .b+")"
      };
    }
    var area_bar = (
      <span>
        <h3 key="title" className="inline
            "><Name model={area}/></h3>
        <span key="variables">{_.map(
            variables, function(v, i) {
          return (
            <span key={i} title={v.value}
                className="var-info">
              <span className="var-name">{
                  v.dataset.name}</span>
              <span className="var-value
                  ">{v.value ? numf.
                  format(v.value) : "No
                  Data"}</span>
            </span>
          );
        }, this)}</span>
      </span>
    );
  }
  return (
    <div className="pure-g">
      <div className="pure-u-1 infobar-
          title" style={titlebar_style}>
        <div className="pure-u-4-5">
          {area_bar}
        </div>
        <div className="pure-u-1-5">
          <button className="pull-right
              button button-flat button-
              close" style={_.pick(
              titlebar_style, "color")}
              onClick={this.handle_close
              }>&times;</button>
        </div>
      </div>
      <div className="pure-u-1 infobar-
          content">
        <div className="pure-u-2-3">
          <AreaElectionsList bus={this.
              props.bus} collection={
              elections}/>
        </div>
        <div className="pure-u-1-3">
          <AreaFamiliesList bus={this.
              props.bus} collection={
              elections}/>
        </div>
      </div>
    </div>
  );
},

filter_datapoints: function(datapoints,
    area_code) {
  area_code = ("0" + area_code);
  var match_start = area_code.substr
      (2-9,2) === "00" ? 0 : 2;
  var area_code_match = area_code.substr(
      match_start-9);
  return datapoints.filter(function(p) {
    return ("0"+p.get("area_code")).substr
        (match_start-9) ==
        area_code_match;
  });
},

on_select: function(selected) {
  if(this.state.selected && selected &&
      this.state.selected.area_code ==
      selected.area_code) {
    if(this.is_hidden()) this.show();
  }else{
    this.setState({selected: null});
    if(selected) {
      if(this.is_hidden()) {
        this.show();
        setTimeout(function() {
          this.setState({selected:
              selected});
        }.bind(this), 400);
      }else{
        this.setState({selected: selected
            });
      }
    }
  }
},

on_main_settings: function(settings) {
  if(settings.year) this.forceUpdate();
},

on_choropleth_data: function(data) {
  this.setState({data: data});
},

on_route: function(e) {
  this.hide();
},

handle_close: function() {
  this.hide();
  if(this.state.selected) setTimeout(this.
      state.selected.on_close, 400);
},

show: function() {
  $(React.findDOMNode(this).parentNode).
      addClass("show");
},

hide: function() {
  $(React.findDOMNode(this).parentNode).
      removeClass("show");
},

is_hidden: function() {
  return !$(React.findDOMNode(this).
      parentNode).hasClass("show");
},

get_color: function(variables) {
  var black = {r:20, g:20, b:20};
  var color = null;
  _.each(variables, function(variable, i)
      {
    if(!variable) return;
    var value = variable.value;
    if(!value) return;
    var scale = variable.dataset.
        color_scale;
    var classes = variable.dataset.classes
        ;
    if(scale.length + 1 !== classes.length
        ) return;

    var class_color = black;
    for (var i = 1; i < classes.length; i
        ++) {
      var min = classes[i - 1];
      var max = classes[i];
      if(min <= value && value <= max) {
        class_color = scale[i - 1];
        break;
      }
    };
    color = color ? ChoroplethLegend.
        combine_colors(color, class_color
        ) : class_color;
  }, this);
  return color;
},

get_text_color: function(background_color)
    {
  var o = Math.round((background_color.r *
      299 + background_color.g * 587 +
      background_color.b * 114) /1000);
  return (o > 125) ? {r:0,g:0,b:0} : {r
      :255,g:255,b:255}; //http://www.w3.
      org/TR/AERT#color-contrast
},
```

```
  });
```

*js-src/view/main/MapPanel.jsx*

```
"use strict";
define(["react", "underscore", "leaflet", "
    config.map", "view/main/map/
    ChoroplethLayer", "view/main/map/
    TagCloudLayer"], function(React, _, L,
    config, ChoroplethLayer, TagCloudLayer) {
  return React.createClass({
    getInitialState: function() {
      return {
        year: 2013,
      }
    },

    componentWillMount: function() {
      this._interval = setInterval(function()
          {
        this.geojson_cache = {}; // clear
            cache once in a while
      }.bind(this), 6 * 60 * 1000); // 6
          minutes
    },

    componentWillUnmount: function() {
      clearInterval(this._interval);

      this.choropleth.destruct();
      this.tagcloud.destruct();

      this.map.off("viewreset moveend zoomend
          ", this.update_view);
      this.map = null;

      this.props.bus.tagcloud_data.off("update
          ", this.on_tagcloud_data);
      this.props.bus.main_settings.off("update
          ", this.on_main_settings);
    },

    componentDidMount: function() {
      var that = this;

      var tile_layer = L.tileLayer(config.url,
          {
        attribution: config.attribution,
      });

      this.map = L.map(this.getDOMNode(), {
        center: [13, 122],
        zoom: 6,
        layers: [tile_layer],
        minZoom: 5,
        maxZoom: 16,
        maxBounds: [
          [-1, 109],
          [27, 135]
        ],
      });

      this.geojson_cache = {};
      this.url_added = {};
      this.hash_added = {};
      this.ajax_requests = [];

      this.choropleth = new ChoroplethLayer(
          this.props.bus);
      this.choropleth.addTo(this.map);

      this.tagcloud = new TagCloudLayer(this.
          props.bus);
      this.tagcloud.addTo(this.map);

      this.labels = L.tileLayer("http://{s}.
          basemaps.cartocdn.com/
          light_only_labels/{z}/{x}/{y}.png")
          ;
      this.labels.addTo(this.map);
      this.map.getPanes().shadowPane.
          appendChild(this.labels.
          getContainer());
      this.labels.getContainer().style.
          pointerEvents = "none";

      this.last_level = "region";
      this.target_zoom = 0;
      this.update_view();

      this.map.on("viewreset moveend zoomend",
          this.update_view);
      this.props.bus.tagcloud_data.on("update
          ", this.on_tagcloud_data);
      this.props.bus.main_settings.on("update
          ", this.on_main_settings);
    },

    render: function() {
      return <div className="map-panel"></div
          >;
    },

    reset_geojson: function(level) {
      this.url_added = {};
      this.hash_added = {};
      _.each(this.ajax_requests, function(r){
          r.abort(); });
      this.ajax_requests = [];
    },

    add_geojson: function(geojson_url) {
      var geojson = this.geojson_cache[
          geojson_url];
      if(geojson) {
        if(geojson !== true) {
          var hash = geojson.hash;
          if(!this.hash_added[hash]) {
            this.hash_added[hash] = true;
            this.choropleth.add_geojson(
                geojson);
            this.tagcloud.add_geojson(geojson)
                ;
          }
        }
      }else{
        if(this.url_added[geojson_url]) return
            ;
        this.url_added[geojson_url] = true;
        var request = $.get(geojson_url)
          .success(function(data) {
            this.url_added[geojson_url] = true
                ;
            if(typeof data === "object") {
              var options = _.defaults(this.
                  choropleth.geojson_options,
                    {
                onEachFeature: this.
                    on_each_feature
              });
              var geojson = this.geojson_cache
                  [geojson_url] = L.geoJson(
                  data, options);
              geojson.hash = this.hash_geojson
                  (data);
              this.add_geojson(geojson_url);
            }else{
              this.geojson_cache[geojson_url]
                  = true;
            }
          }.bind(this))
          .fail(function(){
            this.url_added[geojson_url] =
                false;
          }.bind(this));

        this.ajax_requests.push(request);
      }
    },

    on_each_feature: function(feature, layer)
        {
      layer.on({
        click: function(e) {
          this.select(feature, layer);
        }.bind(this),
      });
    },

    select: function(feature, layer) {
      if(feature && layer) {
        this.props.bus.main_settings.emit("
            select", {
          area_code: parseInt(feature.
              properties.PSGC, 10),
          layer: layer,
          on_close: function() {
            this.select(null, null);
          }.bind(this),
        });
      }else{
        this.props.bus.main_settings.emit("
            select", null);
      }
    },

    hash_geojson: function(geojson) {
      var sum = 0;
      _.each(geojson.features, function(
          feature) {
```

```
        sum = (sum << 1) ^ (parseInt(feature.
            properties.PSGC, 10) | 0);
      });
      return sum;
    },

    on_tagcloud_data: function(data) {
      this.labels.getContainer().style.display
          = data ? "none" : "block";
    },

    update_view: function() {
      var zoom = this.map.getZoom();
      if(zoom != this.last_zoom) this.props.
          bus.main_settings.emit("update", {
          zoom: zoom});
      this.last_zoom = zoom;

      var level;
      if(zoom >= 12) {
        level = "barangay";
      }else if(zoom >= 10) {
        level = "municipality";
      }else if(zoom >= 8) {
        level = "province";
      }else{
        level = "region";
      }

      var bounds = this.map.getBounds().pad
          (0.4);
      var nw = bounds.getNorthWest();
      var se = bounds.getSouthEast();

      var nw_tile = {x: this.long2tile(nw.lng,
          this.target_zoom), y: this.
          lat2tile(nw.lat, this.target_zoom)
          };
      var se_tile = {x: this.long2tile(se.lng,
          this.target_zoom), y: this.
          lat2tile(se.lat, this.target_zoom)
          };

      var t = 0;
      for (var x = se_tile.x; x >= nw_tile.x;
          x--) {
        for (var y = se_tile.y; y >= nw_tile.y
            ; y--) {
          var path = this.target_zoom + "/" +
              x + "/" + y;
          setTimeout(this.add_geojson, t +=
              10, "api.php/geojson/" + this.
              last_level + "/" + path);
        }
      }
    },

    on_main_settings: function(settings) {
      if(!settings.level) return;

      var level = settings.level;
      if(level != this.last_level) this.
          reset_geojson(level);
      this.last_level = level;

      this.target_zoom = { // These zoom
          levels must match with the server
        "region": 0,
        "province": 8,
        "municipality": 10,
        "barangay": 12,
      }[level];

      var visual_zoom = {
        "region": {target: 6, min: 5, max: 7},
        "province": {target: 8, min: 6, max:
            10},
        "municipality": {target: 10, min: 8,
            max: 13},
        "barangay": {target: 12, min: 11, max:
            16},
      }[level];

      this.map.options.minZoom = visual_zoom.
          min;
      this.map.options.maxZoom = visual_zoom.
          max;

      if(this.map.getZoom() !== visual_zoom.
          target) {
        this.map.setZoom(visual_zoom.target);
      }else{
        this.update_view();
      }
    },

    long2tile: function(lon,zoom) { return (
        Math.floor((lon+180)/360*Math.pow(2,
        zoom))); },
    lat2tile: function(lat,zoom)  { return (
        Math.floor((1-Math.log(Math.tan(lat*
        Math.PI/180) + 1/Math.cos(lat*Math.PI
        /180))/Math.PI)/2 *Math.pow(2,zoom)))
        ; },
  });
});
```

*js-src/view/main/OfficialFamilyList.jsx*

```
"use strict";
define(function(require) {
  var React = require("react", "react.backbone
      "),
    Name = require("jsx!view/Name");

  return React.createBackboneClass({
    render: function() {
      return (
        <div>
          {this.collection().map(function(o) {
            return <Name key={o.cid} model={o
                }/>;
          })}
        </div>
      );
    },
  });
});
```

*js-src/view/main/Sidebar.jsx*

```
"use strict";
define(function(require) {
  var _ = require("underscore"),
    React = require("react"),
    InstanceCache = require("InstanceCache"),
    Area = require("model/Area"),
    ChoroplethSettingsPane = require("jsx!view
        /main/ChoroplethSettingsPane"),
    TagCloudSettingsPane = require("jsx!view/
        main/TagCloudSettingsPane");

  return React.createClass({
    mixins: [React.addons.LinkedStateMixin],

    getInitialState: function() {
      return {
        year: new Date().getFullYear(),
        year_input: new Date().getFullYear(),
        level: "region",
        allowed_levels: {
          province: true,
          region: true,
          municipality: true,
          barangay: true,
        },
        playing: false,
        disabled: false,
      };
    },

    componentWillMount: function() {
      this.props.bus.main_settings.emit("
          update", {
        year: this.state.year,
        level: this.state.level,
      });

      this.min_year = 0;
      this.max_year = this.state.year;

      this.choropleth_data = [null, null];
      this.tagcloud_data = null;
    },

    componentDidMount: function() {
      this.props.bus.router.on("route", this.
          on_route);
      this.props.bus.choropleth_settings.on("
          update", this.
          update_choropleth_settings);
      this.props.bus.tagcloud_settings.on("
          update", this.
          update_tagcloud_settings);
    },
```

```
componentWillUnmount: function() {
  this.props.bus.router.off("route", this.
      on_route);
  this.props.bus.choropleth_settings.off("
      update", this.
      update_choropleth_settings);
  this.props.bus.tagcloud_settings.off("
      update", this.
      update_tagcloud_settings);
},

componentWillUpdate: function(nextProps,
    nextState) {
  var delta = _.omit(_.pick(nextState, "
      year", "level"), function(value,
      key, object) {
    return this.state[key] === value;
  }.bind(this));
  if(!_.isEmpty(delta)) this.props.bus.
      main_settings.emit("update", delta)
      ;
},

on_route: function(e) {
  this.setState({disabled: e.route !== "
      main"});
  if(e.route == "main") {
    this.setState({level: "region"});
  }
},

update_choropleth_settings: function(
    settings) {
  this.choropleth_data = [settings.
      dataset1, settings.dataset2];
  this.on_update();
},
update_tagcloud_settings: function(
    settings) {
  this.tagcloud_data = settings.dataset;
  this.on_update();
},
on_update: function() {
  var datasets = _.filter(this.
      choropleth_data.concat(this.
      tagcloud_data));
  if(_.isEmpty(datasets)) {
    this.min_year = 0;
    this.max_year = new Date().getFullYear
        ();
    this.setState({
      allowed_levels: _.mapObject(this.
          state.allowed_levels, function
          (){ return true; })
    });
    return;
  }

  var year = this.state.year;
  this.min_year = _.max(datasets, function
      (d){ return d.get("min_year"); }).
      get("min_year");
  this.max_year = _.min(datasets, function
      (d){ return d.get("max_year"); }).
      get("max_year");
  if(this.state.year > this.max_year) {
    year = this.max_year;
    this.setState({year: this.max_year,
        year_input: this.max_year});
  }else if(this.state.year < this.min_year
      ) {
    year = this.min_year;
    this.setState({year: this.min_year,
        year_input: this.min_year});
  }

  var allowed_levels = _.reduce(datasets,
      function(m,d){
    return _.mapObject(m, function(v,k){
      return v || d.get("contained_levels
          ")[k];
    });
  }, _.mapObject(this.state.allowed_levels
      , function(){ return false; }));

  if(!_.isEqual(allowed_levels, this.state
      .allowed_levels)) {
    this.setState({allowed_levels:
        allowed_levels});
  }
  if(!allowed_levels[this.state.level]) {
    this.setState({level: _.chain(
        allowed_levels).keys().find(
        function(k){ return
        allowed_levels[k]; }).value()});
  }
},

render: function() {
  var settings_pane = (
    <div key="pane_settings" className="
        pane">
      <h6 className="pane-header">
          Visualization Settings</h6>
      <div className="pane-content pure-g
          ">
        <select className="pure-u-1 input"
            value={this.state.level}
            onChange={this.
            handle_change_level} required
            disabled={this.state.
            disabled}>
          {this.state.allowed_levels.
              region ? <option value="
              region">Regional level</
              option> : null}
          {this.state.allowed_levels.
              province ? <option value="
              province">Provincial level
              </option> : null}
          {this.state.allowed_levels.
              municipality ? <option
              value="municipality">
              Municipal level</option> :
              null}
          {this.state.allowed_levels.
              barangay ? <option value="
              barangay">Barangay level</
              option> : null}
        </select>
        <form className="pure-u-1 group
            form" onSubmit={this.
            handle_submit_year}>
          <input className="pure-u-2-3
              group-component" type="
              number" valueLink={this.
              linkState("year_input")}
              required disabled={this.
              state.disabled} />
          <input className="pure-u-1-3
              group-component button-
              primary" type="submit"
              value="Go" disabled={this.
              state.disabled} />
        </form>
        <div className="pure-u-1 group">
          <button className="pure-u-1-4
              group-component button"
              onClick={this.
              handle_backward} disabled={
              this.state.disabled}><i
              className="fa fa-step-
              backward"/></button>
          {this.state.playing
            ? <button className="pure-u
                -1-2 group-component
                button" onClick={this.
                handle_pause} disabled={
                this.state.disabled}><i
                className="fa fa-pause
                "/>  Pause</button>
            : <button className="pure-u
                -1-2 group-component
                button" onClick={this.
                handle_play} disabled={
                this.state.disabled}><i
                className="fa fa-play"/>&
                ensp; Play</button>}
          <button className="pure-u-1-4
              group-component button"
              onClick={this.
              handle_forward} disabled={
              this.state.disabled}><i
              className="fa fa-step-
              forward"/></button>
        </div>
      </div>
    </div>
  );
  return (
    <div>
      <ChoroplethSettingsPane key="
          pane_choropleth" bus={this.
          props.bus} disabled={this.state
          .disabled} />
      <TagCloudSettingsPane key="
          pane_tagcloud" bus={this.props.
          bus} disabled={this.state.
          disabled} />
      {settings_pane}
    </div>
  );
},

handle_change_level: function(e) {
  this.setState({level: e.target.value});
},
```

```
    handle_submit_year: function(e) {
      if(e) e.preventDefault();
      var year = parseInt(this.state.
          year_input, 10);
      if(year < this.min_year) {
        year = this.min_year;
      }else if(year > this.max_year) {
        year = this.max_year;
      }
      this.setState({year: year, year_input:
          year});
    },

    handle_play: function() {
      var to_step = _.after(2, function() {
        this.props.bus.choropleth_data.off("
            update", to_step);
        this.props.bus.tagcloud_data.off("
            update", to_step);
        this.setState({year_input: this.state.
            year});
        setTimeout(step.bind(this), 1000);
      }.bind(this));

      this.setState({playing: true});
      if(this.state.year >= this.max_year) {
        this.props.bus.choropleth_data.on("
            update", to_step);
        this.props.bus.tagcloud_data.on("
            update", to_step);
        this.setState({year: this.min_year});
      }else{
        step.call(this);
      }

      function step() {
        var year = this.state.year + 1;
        this.setState({year: year});

        if(year >= this.max_year) {
          this.setState({playing: false,
              year_input: year});
        }else{
          this.props.bus.choropleth_data.on("
              update", to_step);
          this.props.bus.tagcloud_data.on("
              update", to_step);
        }
      }
    },

    handle_pause: function() {
      this.setState({playing: false});
    },

    handle_backward: function() {
      this.setState({year: this.min_year,
          year_input: this.min_year});
    },

    handle_forward: function() {
      this.setState({year: this.max_year,
          year_input: this.max_year});
    },
  });
});
```

*js-src/view/main/TagCloudLegend.jsx*

```
"use strict";
define(function(require) {
  var _ = require("underscore"),
    $ = require("jquery"),
    React = require("react");

  return React.createClass({
    getInitialState: function() {
      return {data: null};
    },

    componentDidMount: function() {
      this.width = $(React.findDOMNode(this).
          parentElement).width();
      this.props.bus.tagcloud_data.on("update
          ", this.update);
    },

    componentWillUnmount: function() {
      this.props.bus.tagcloud_data.off("update
          ", this.update);
    },

    render: function() {
      var margin = 5;
      var legend_width = this.width ? this.
          width - margin * 2 : 0;
      var legend_height = 0;

      if(this.state.data) {
        var font_size_func = function(size) {
            return Math.sqrt(0.8 * size) + "
            em"; };
        var max = this.state.data.classes[this
            .state.data.classes.length - 1];
        var l2max = Math.ceil(Math.log(max) /
            Math.LN2);
        var step = 25;
        legend_height = l2max * step;

        var g = (
          <g transform={"translate("+margin
              +","+margin+")"}>
            {_.map(_.range(0, l2max), function
                (i) {
              var n = Math.ceil(Math.pow(2, i)
                  );
              return (
                <g key={i} transform={"
                    translate(0,"+(i*step)+")
                    "}>
                  <text
                    className="map-tag
                        tagcloud-legend"
                    x={20} y={10}
                    textAnchor="end"
                    fontSize={font_size_func(n
                        )}>
                    A
                  </text>
                  <text
                    className="tagcloud-legend
                        "
                    x={40} y={10}
                    textAnchor="start">
                    {"size " + n}
                  </text>
                </g>
              );
            })}
          </g>
        );
      }

      return <svg width={legend_width + margin
          * 2} height={legend_height +
          margin * 2}>{g}</svg>;
    },

    update: function(data) {
      this.setState({data: data});
    },
  });
});
```

*js-src/view/main/TagCloudSettingsPane.jsx*

```
"use strict";
define(function(require) {
  var _ = require("underscore"),
    React = require("react"),
    DatasetCollection = require("model/
        DatasetCollection"),
    Modal = require("jsx!view/Modal"),
    DatasetChooser = require("jsx!view/main/
        DatasetChooser"),
    TagCloudLegend = require("jsx!view/main/
        TagCloudLegend");

  return React.createClass({
    getInitialState: function() {
      return {
        dataset: null,
      };
    },

    shouldComponentUpdate: function(nextProps,
        nextState) {
      return !_.isEqual(nextProps, this.props)
          || !_.isEqual(nextState, this.
          state);
    },
```

```
    componentWillUpdate: function(nextProps,
         nextState) {
      this.props.bus.tagcloud_settings.emit("
           update", {
        dataset: nextState.dataset,
      });
    },

    render: function() {
      var text = "Select Dataset";
      var selection_button = null;
      if(this.state.dataset) {
        var selection_button = (
          <div className="pure-u-1 group group
               -no-table">
            <button className="pure-u-5-6
                 group-component one-line
                 button" onClick={this.
                 handle_select} disabled={this
                 .props.disabled}>{this.state.
                 dataset.get("name")}</button>
            <button className="pure-u-1-6
                 group-component button"
                 onClick={this.handle_remove}
                 disabled={this.props.disabled
                 }><i className="fa fa-close
                 "/></button>
          </div>
        );
      }else{
        var selection_button = (
          <div className="pure-u-1">
            <button className="pure-u-1 button
                 " onClick={this.handle_select
                 } disabled={this.props.
                 disabled}>Select Dataset</
                 button>
          </div>
        );
      }
      return (
        <div className="pane">
          <h6 className="pane-header">Tag
               Cloud Layer</h6>
          <div className="pane-content">
            {selection_button}
            <TagCloudLegend bus={this.props.
                 bus}/>
          </div>
        </div>
      );
    },

    handle_select: function() {
      var that = this;
      var dataset_collection = new
           DatasetCollection();
      dataset_collection.fetch({data: {type: "
           tag"}});
      that.modal = Modal.open("Select a
           dataset to visualize", (
        <DatasetChooser collection={
             dataset_collection} onSelect={
             function(dataset) {
          that.select_dataset(dataset);
          that.modal.close();
          that.modal = null;
        }}/>
      ));
    },

    handle_remove: function() {
      this.select_dataset(null);
    },

    select_dataset: function(dataset) {
      this.setState({dataset: dataset});
    },
  });
});
```

*js-src/view/main/map/ChoroplethLayer.js*

```
"use strict";
define(["underscore", "leaflet", "model/Area",
     "jsx!view/main/ChoroplethLegend"],
    function(_, L, Area, ChoroplethLegend) {
  return L.LayerGroup.extend({
    initialize: function(bus) {
      L.LayerGroup.prototype.initialize.call(
           this);
      var that = this;

      this.bus = bus;

      this._style_neutral = {
        weight: 3,
        opacity: 0.1,
        color: "#7f7f7f",
        fillOpacity: 0,
        fillColor: "#c7c7c7",
        className: "map-polygon visible",
      };
      this._style_colored = _.defaults({
        weight: 4,
        opacity: 1,
        color: "#ffffff",
        fillOpacity: 1,
        fillColor: "#dfdfdf",
      }, this._style_neutral);
      this._style_highlight = {
        weight: 6,
        opacity: 1,
      };

      this.geojson_options = {
        smoothFactor: 2.4,
        style: this._style_neutral,
      };

      this._geojson_number = 0;
      this._geojson = [];
      this._dataset_number = 0;
      this._datasets = [];

      this.map = null;
      this.selected_layer = null;

      this.on_data = this.on_data.bind(this);
      this.on_main_settings = this.
           on_main_settings.bind(this);
      this.on_select = this.on_select.bind(
           this);

      this.bus.choropleth_data.on("update",
           this.on_data);
      this.bus.main_settings.on("update", this
           .on_main_settings);
      this.bus.main_settings.on("select", this
           .on_select);
      this.bus.main_settings.on("select", this
           .on_select);
    },

    destruct: function() {
      this.bus.choropleth_data.off("update",
           this.on_data);
      this.bus.main_settings.off("update",
           this.on_main_settings);
      this.bus.main_settings.off("select",
           this.on_select);
    },

    onAdd: function (map) {
      this.map = map;
      L.LayerGroup.prototype.onAdd.call(this,
           map);
    },

    on_main_settings: function(settings) {
      if(settings.level) this.reset_geojson();
    },

    on_data: function(data) {
      this._dataset_number++;
      this._datasets = data;
      this.reset_polygons();
    },

    on_select: function(selected) {
      if(this.selected_layer) {
        this.selected_layer.setStyle(this.
             compute_polygon_style(this.
             selected_layer, false));
      }
      if(selected) {
        var layer = selected.layer;
        this.selected_layer = layer;
        layer.setStyle(this.
             compute_polygon_style(layer, true
             ));
        layer.bringToFront();
      }
    },

    reset_geojson: function() {
      this._geojson_number++;
      this._geojson = [];
    },
```

```
    add_geojson: function(geojson) {
      this._geojson.push(geojson);

      var t = 0;
      var layers = geojson.getLayers();
      for (var i = 0; i < layers.length; i++)
           {
        layers[i].variables = [];
        setTimeout(this.colorize_polygon.bind(
             this), t += 10, layers[i], this.
             _geojson_number);
      }
    },

    reset_polygons: function() {
      var t = 0;
      for (var i = this._geojson.length - 1; i
           >= 0; i--) {
        var layers = this._geojson[i].
             getLayers();
        for (var j = 0; j < layers.length; j
             ++) {
          setTimeout(this.colorize_polygon.
               bind(this), t += 10, layers[j],
                this._geojson_number);
        }
      }
    },

    // Sets polygon style based on the dataset
    colorize_polygon: function(poly, gn){
      loop.call(this, poly, gn, 0, true);
      function loop(poly, gn, dn, add) {
        var intersects = poly.getBounds().pad
             (10).intersects(this.map.
             getBounds());
        if(intersects && this._geojson_number
             == gn) {
          if(this._dataset_number != dn) {
            dn = this._dataset_number;
            var area_code = parseInt(poly.
                 feature.properties.PSGC, 10);
            var level = Area.get_level(
                 area_code);
            poly.variables = [];
            _.each(this._datasets, function(
                 dataset) {
              if(dataset) {
                var value = null;
                var filtered = this.
                     filter_datapoints(dataset
                     .datapoints, area_code);
                if(filtered.length) {
                  value = filtered[0].get("
                       value");
                }
                poly.variables.push({dataset:
                     dataset, level: level,
                     value: value});
              }else{
                poly.variables.push(null);
              }
            }, this);
            poly.setStyle(this.
                 compute_polygon_style(poly,
                 false));
          }
          if(add) {
            add = false;
            this.addLayer(poly);
          }
        }else{
          if(!add) {
            add = true;
            this.removeLayer(poly);
          }
        }
        if(this._geojson_number == gn || !add)
             setTimeout(loop.bind(this),
             1000, poly, gn, dn, add);
      }
    },

    filter_datapoints: function(datapoints,
         area_code) {
      area_code = ("0" + area_code);
      var match_start = area_code.substr
           (2-9,2) === "00" ? 0 : 2;
      var area_code_match = area_code.substr(
           match_start-9);
      return datapoints.filter(function(p) {
        return ("0"+p.get("area_code")).substr
             (match_start-9) ===
             area_code_match;
      });
    },

    compute_polygon_style: function(poly,
         highlight) {
      var style = null;

      var colored = _.some(this._datasets) &&
           _.every(this._datasets, function(d)
            {
        if(!d) return true;
        return _.some(poly.variables, function
             (v) {
          return v && d == v.dataset && v.
               value !== null;
        });
      });

      if(colored) {
        var color = this.get_color(poly.
             variables);
        var darker = _.mapObject(color,
             function(v) {
          return Math.max(0, v - 64);
        });
        style = _.defaults({
          color: "rgb("+Math.floor(darker.r)
               +","+Math.floor(darker.g)+","+
               Math.floor(darker.b)+")",
          fillColor: "rgb("+Math.floor(color.r
               )+","+Math.floor(color.g)+","+
               Math.floor(color.b)+")",
        }, this._style_colored);
      }else{
        style = this._style_neutral;
      }

      if(highlight) {
        style = _.extend(_.clone(style), this.
             _style_highlight);
      }

      return style;
    },

    get_color: function(variables) {
      var black = {r:20, g:20, b:20};
      var color = null;
      _.each(variables, function(variable, i)
           {
        if(!variable) return;
        var value = variable.value;
        if(!value) return;
        var scale = variable.dataset.
             color_scale;
        var classes = variable.dataset.classes
             ;
        if(scale.length + 1 !== classes.length
             ) return;

        var class_color = black;
        for (var i = 1; i < classes.length; i
             ++) {
          var min = classes[i - 1];
          var max = classes[i];
          if(min <= value && value <= max) {
            class_color = scale[i - 1];
            break;
          }
        };
        color = color ? ChoroplethLegend.
             combine_colors(color, class_color
             ) : class_color;
      }, this);
      return color;
    },
  });
});
```

*js-src/view/main/map/TagCloudLayer.js*

```
"use strict";
define(["underscore", "d3", "leaflet", "
     InstanceCache", "view/main/map/
     ChoroplethLayer"], function(_, d3, L,
     InstanceCache, ChoroplethLayer) {
  return L.LayerGroup.extend({
    initialize: function(bus) {
      L.LayerGroup.prototype.initialize.call(
           this);

      this.bus = bus;

      this._geojson_number = 0;
      this._geojson = [];
      this._dataset_number = 0;
```

```
    this._dataset = null;

    this._redraw_callback = _.debounce(this.
        redraw.bind(this), 200);

    this.map = null;
    this.minimum_size = 4;

    this.on_data = this.on_data.bind(this);
    this.on_main_settings = this.
        on_main_settings.bind(this);

    this.bus.tagcloud_data.on("update", this
        .on_data);
    this.bus.main_settings.on("update", this
        .on_main_settings);
  },

  destruct: function() {
    this.bus.tagcloud_data.off("update",
        this.on_data);
    this.bus.main_settings.off("update",
        this.on_main_settings);
  },

  onAdd: function (map) {
    this.map = map;
    map.on("move", _.throttle(this.redraw.
        bind(this), 100));

    d3.select("#tag-cloud-overlay").remove()
        ;
    d3.select(this.map.getPanes().shadowPane
        )
      .append("svg")
        .attr("id", "tag-cloud-overlay")
        .style("pointer-events", "none")
        .style("position", "absolute")
      .append("g")
        .attr("id", "tag-cloud-container")
        .attr("class", "leaflet-zoom-hide");

    L.LayerGroup.prototype.onAdd.call(this,
        map);
  },

  on_main_settings: function(settings) {
    if(settings.level) this.reset_geojson();
    if(settings.zoom) this.
        calculate_minimum_size();
    this.redraw();
  },

  on_data: function(data) {
    this._dataset_number++;
    this._dataset = data;
    this.calculate_minimum_size();
    this.reset_tags();
  },

  calculate_minimum_size: function() {
    this.minimum_size = 10 / this.map.
        getZoom();
    if(this._dataset) this.minimum_size =
        Math.max(this.minimum_size, this.
        _dataset.classes[2]);
  },

  reset_geojson: function() {
    this._geojson_number++;
    this._geojson = [];
    this.remove_tags();
  },

  add_geojson: function(geojson) {
    this._geojson.push(geojson);

    var t = 0;
    var layers = geojson.getLayers();
    for (var i = 0; i < layers.length; i++)
        {
      layers[i].tags = [];
      if(this._dataset) {
        setTimeout(this.tag_poly.bind(this),
            t += 10, layers[i], this.
            _geojson_number);
      }
    }
  },

  reset_tags: function() {
    var t = 0;
    for (var i = this._geojson.length - 1; i
        >= 0; i--) {
      var layers = this._geojson[i].
          getLayers();
      for (var j = 0; j < layers.length; j
          ++) {
        setTimeout(this.tag_poly.bind(this),
            t += 10, layers[j], this.
            _geojson_number);
      }
    }
  },

  remove_tags: function() {
    for (var i = this._geojson.length - 1; i
        >= 0; i--) {
      var layers = this._geojson[i].
          getLayers();
      for (var j = 0; j < layers.length; j
          ++) {
        layers[j].tags = [];
      }
    }
  },

  tag_poly: function(poly, gn) {
    loop.call(this, poly, gn, 0, true);
    function loop(poly, gn, dn, add) {
      var intersects = poly.getBounds().pad
          (10).intersects(this.map.
          getBounds());
      if(intersects && this._geojson_number
          == gn) {
        if(this._dataset_number != dn) {
          dn = this._dataset_number;

          poly.tags = [];
          if(this._dataset) {
            var area_code = parseInt(poly.
                feature.properties.PSGC,
                10);
            var datapoints = this.
                filter_datapoints(this.
                _dataset.datapoints,
                area_code, this.
                minimum_size);
            for (var i = 0; i < datapoints.
                length; i++) {
              var p = datapoints[i];
              var family = InstanceCache.get
                  ("Family", p.get("
                  family_id"));
              poly.tags.push({
                data: p,
                area_code: area_code,
                family: family,
                poly: poly,
              });
            }
          }
        }
        if(add) {
          add = false;
          poly.tags_visible = true;
          this._redraw_callback();
        }
      }else{
        if(!add) {
          add = true;
          poly.tags_visible = false;
          this._redraw_callback();
        }
      }
      if(this._geojson_number == gn || !add)
          setTimeout(loop.bind(this),
          1000, poly, gn, dn, add);
    }
  },

  filter_datapoints: function(datapoints,
      area_code, minimum_size) {
    area_code = ("0" + area_code);
    var match_start = area_code.substr
        (2-9,2) === "00" ? 0 : 2;
    var area_code_match = area_code.substr(
        match_start-9);
    return datapoints.filter(function(p) {
      return p.get("value") >= minimum_size
          && ("0"+p.get("area_code")).
          substr(match_start-9) ===
          area_code_match;
    });
  },

  position_tags: function(poly) {
    var bounds = poly.getBounds();
    var top = bounds.getNorth();
    var bottom = bounds.getSouth();
    var left = bounds.getWest();
    var right = bounds.getEast();
    var center = bounds.getCenter();
    _.each(poly.tags, function(tag) {
      var lat = top*0.75 + bottom*0.25 + (
          bottom - top)*0.5 * ((poly.tags.
          indexOf(tag) + 0.5) / poly.tags.
```

```
            length);
        var lng = left*0.6 + right*0.4 + (
            right - left)*0.2 * ((tag.family.
            id % 16) / 16);
        tag.coords = L.latLng(lat, lng);
      });
    },

    redraw: function() {
      var bounds = this.map.getBounds();
      var size = this.map.getSize();
      var top_left = this.map.
          latLngToLayerPoint(bounds.
          getNorthWest());
      var svg = d3.select("#tag-cloud-overlay
          ");
      svg.style("width", size.x + "px")
        .style("height", size.y + "px")
        .style("left", top_left.x + "px")
        .style("top", top_left.y + "px");
      var g = d3.select("#tag-cloud-container
          ");
      g.attr("transform", "translate(" + (-
          top_left.x) + "," + (-top_left.y) +
          ")");

      var tags = [];
      for (var i = this._geojson.length - 1; i
          >= 0; i--) {
        var layers = this._geojson[i].
            getLayers();
        for (var j = 0; j < layers.length; j
            ++) {
          this.position_tags(layers[j]);
          if(layers[j].tags_visible) Array.
              prototype.push.apply(tags,
              layers[j].tags);
        }
      }

      var filtered_tags = _.chain(tags)
        .filter(function(tag) {
          if(tag.data.get("value") < this.
              minimum_size) return false;
          return true;
        }, this)
        .map(function(tag) {
          if(!tag.family.has("name")) this.
              family_fetch_enqueue(tag.family
              );
          var point = this.map.
              latLngToLayerPoint(tag.coords);
          return {
            key: tag.area_code + "|" + tag.
                family.id,
            x: point.x,
            y: point.y,
            size: tag.data.get("value"),
            family: tag.family,
          };
        }, this)
        .value();

      var text_func = function(tag) {
        if(tag.family.has("name")) return tag.
            family.get("name");
        return "..."; // TODO: spinner
      };
      var transform_func = function(tag) {
          return "translate(" + tag.x + "," +
            tag.y + ")"; };

      var font_size_func = function(tag) {
          return Math.sqrt(0.8 * tag.size) +
          "em"; };
      var opacity_func= function(tag) { return
           tag.family.has("name") ? 1 : 0.5;
          };

      var tags_data = g.selectAll("g").data(
          filtered_tags, function(tag){
          return tag.key; });

      // add new tags
      var entered_tags = tags_data.enter().
          append("g");
      entered_tags.append("text").attr("class
          ", "map-tag-stroke");
      entered_tags.append("text").attr("class
          ", "map-tag");
      entered_tags
        .attr("transform", transform_func)
        .style("opacity", 0);

      // update all tags
      tags_data
        .style("font-size", font_size_func)
        .transition().duration(300)
        .attr("transform", transform_func)
        .style("opacity", opacity_func);
      tags_data.selectAll("text")
        .text(text_func);

      // remove old tags
      tags_data.exit()
        .transition().duration(300)
        .style("opacity", 0)
        .remove();
    },

    family_fetch_enqueue: function(family) {
      this.fetch_queue = this.fetch_queue ||
          [];

      if(family.has("name")) return;
      this.fetch_queue.push({model: family,
          options: {success: function() {
        this._redraw_callback();
      }.bind(this)}});

      if(this.fetch_queue.length == 1) {
        dequeue(this.fetch_queue);
      }

      function dequeue(queue) {
        if(!queue.length) return;
        var next = queue.shift();
        next.model.fetch({
          success: function(m,r,o) {
            if(next.options.success) next.
                options.success(m,r,o);
            dequeue(queue);
          },
          error: function(m,r,o) {
            if(next.options.error) next.
                options.error(m,r,o);
            dequeue(queue);
          }
        });
      }
    },
  });
});
```

*login.html*

```
<!DOCTYPE html>
<html>
<head>
  <title>Sign In - DynastyMap</title>

  <link rel="shortcut icon" type="image/png"
      href="favicon.png"/>

  <link rel="stylesheet" type="text/css" href
      ="css/pure-min.css" />
  <link rel="stylesheet" type="text/css" href
      ="css/login.css" />

  <script src="js-src/lib/require.js"></script
      >
  <script src="js-src/config.js"></script>
</head>
<body>
  <div id="main">
    <div id="body" class="layout-row layout-
        col"></div>
  </div>
  <script src="js-build/login.js"></script>
</body>
</html>
```

*php/Dynavis/Core/DataException.class.php*

```
<?php
namespace Dynavis\Core;

class DataException extends \RuntimeException
    {}
```

*php/Dynavis/Core/Entity.class.php*

```php
<?php
namespace Dynavis\Core;
use \PDO;
use \Dynavis\Database;

abstract class Entity implements \
    JsonSerializable{
  // Table name
  const TABLE = null;

  // Array of non-autoincrement fields in the
      database table
  const FIELDS = null;

  // Name of primary key field
  const PRIMARY_KEY = "id";

  // Fields to query when searching
  const QUERY_FIELDS = ["id"];

  // ID of this entity
  private $_id = null;

  // Actual data in the database; synchronized
  private $_data = null;

  public function __construct($id_or_data =
      null, $check = true) {
    if(isset($id_or_data)) {
      if(is_array($id_or_data)) { // assume
          assoc array
        $this->_data = [];
        foreach ($id_or_data as $k => $v) {
          if(in_array($k, static::FIELDS, true
              )) {
            $this->$k = $v;
            if(!$check) $this->_data[$k] = $v;
          }
        }
        if(!$check && isset($id_or_data[static
            ::PRIMARY_KEY])) {
          $this->_id = $id_or_data[static::
              PRIMARY_KEY];
        }
      }else{ // not data, then id
        if(!$check || static::has($id_or_data)
            ) {
          $this->_id = $id_or_data;
        }else{
          throw new NotFoundException("Entity
              ID is not in the database. " .
              get_class($this), 1);
        }
      }
    }else{
      $this->_data = [];
    }
  }

  public static function has($id) {
    return Database::get()->has(static::TABLE,
        [static::PRIMARY_KEY => $id]);
  }

  public static function count() {
    return Database::get()->count(static::
        TABLE);
  }

  public static function list_items($count,
      $start) {
    if($count < 0 || $start < -1) return false
        ;
    $limit = $count == 0 ? null : ["LIMIT" =>
        [(int) $start , (int) $count]];
    return [
      "total" => static::count(),
      "data" => Database::get()->select(static
          ::TABLE, array_merge(static::FIELDS
          , [static::PRIMARY_KEY]), $limit),
    ];
  }

  public static function query_items($count,
      $start, $query) {
    if($count < 0 || $start < -1) return false
        ;
    if(!is_null($query) && empty($query))
        return ["total" => 0, "data" => []];

    $where = " 1 ";
    if(is_null($query)) {
      $uquery = [];
    }else{
      $search_clause = " 1 ";
      $uquery = array_unique($query);
      foreach ($uquery as $k => $v) {
        $word_conditions = " 0 ";
        foreach (static::QUERY_FIELDS as $f) {
          $word_conditions .= " or $f like :
              query_$k";
        }
        $search_clause .= " and (
            $word_conditions) ";
      }
      $where .= " and ($search_clause) ";
    }

    function bind($statement, $uquery) {
      foreach ($uquery as $k => $v) {
        $statement->bindValue(":query_$k", "%
            $v%", PDO::PARAM_STR);
      }
    }

    $count_query = " select count(*) "
      . " from " . static::TABLE
      . " where $where ";

    $count_st = Database::get()->pdo->prepare(
        $count_query);
    bind($count_st, $uquery);
    $count_st->execute();
    $total = (int) $count_st->fetch()[0];

    $select_query = " select " . join(",",
        array_merge(static::FIELDS, [static::
        PRIMARY_KEY]))
      . " from " . static::TABLE
      . " where $where ";
    if($count != 0) {
      $select_query .= " limit :start , :count
          ";
    }

    $select_st = Database::get()->pdo->prepare
        ($select_query);
    bind($select_st, $uquery);
    if($count != 0) {
      $select_st->bindParam(":start", $start,
          PDO::PARAM_INT);
      $select_st->bindParam(":count", $count,
          PDO::PARAM_INT);
    }
    $select_st->execute();

    return [
      "total" => $total,
      "data" => $select_st->fetchAll(),
    ];
  }

  public static function delete_all() {
    $ret = Database::get()->query("delete from
        " . static::TABLE);
    if(!$ret){
      throw new DataException("Error deleting
          entities from the database. " .
          get_called_class());
    }
  }

  public function save() {
    if(isset($this->_id)) {
      $this->update();
    }else{
      $this->insert();
    }
  }

  public function delete() {
    if(!isset($this->_id)){
      throw new \RuntimeException("Cannot
          delete a new entity.");
    }

    $ret = Database::get()->delete(static::
        TABLE, [static::PRIMARY_KEY => $this
        ->_id]);

    if(!$ret){
      throw new DataException("Error deleting
          entity from the database. " .
          get_class($this));
    }

    $this->_data = null;
  }

  public function get_id() { return $this->_id
      ; }

  protected function load() {
    if(is_null($this->_data) && isset($this->
        _id)) {
      $this->_data = Database::get()->get(
          static::TABLE, static::FIELDS, [
          static::PRIMARY_KEY => $this->_id])
```

```
        ;
      // TODO: cast values to field types
      foreach (static::FIELDS as $f) {
        $this->$f = $this->_data[$f];
      }
    }
  }

  public function jsonSerialize() {
    if(is_null($this->_id)){
      throw new \RuntimeException("Cannot
          serialize unsaved entity.");
    }

    if(is_null($this->_data)) $this->load();
    $data = $this->_data;
    $data[static::PRIMARY_KEY] = $this->_id;
    return $data;
  }

  public function __get($prop) {
    if(!in_array($prop, static::FIELDS, true)
        && !property_exists($this, $prop)){
      throw new \RuntimeException("Property is
          not accessible. " . get_class(
          $this) . "::" . $prop);
    }

    $this->load();
    return $this->$prop;
  }

  public function __set($prop, $value) {
    if(!in_array($prop, static::FIELDS, true)
        && !property_exists($this, $prop)){
      throw new \RuntimeException("Property is
          not accessible. " . get_class(
          $this) . "::" . $prop);
    }

    $this->load();
    $this->$prop = $value;
  }

  private function update() {
    if(is_null($this->_data)) return;

    $update_data = [];
    foreach (static::FIELDS as $f) {
      if($this->_data[$f] != $this->$f) {
        $update_data[$f] = $this->$f;
      }
    }

    if(!empty($update_data)){
      $ret = Database::get()->update(static::
          TABLE, $update_data, [static::
          PRIMARY_KEY => $this->_id]);

      if(!$ret) {
        throw new DataException("Error
            updating entity in the database.
            " . get_class($this));
      }

      foreach ($update_data as $key => $value)
          {
        $this->_data[$key] = $value;
      }
    }
  }

  private function insert() {
    $insert_data = [];
    foreach (static::FIELDS as $f) {
      if(!property_exists($this, $f)) {
        $this->$f = null;
      }
      $insert_data[$f] = $this->$f;
    }

    $ret = Database::get()->insert(static::
        TABLE, $insert_data);

    if(!in_array(static::PRIMARY_KEY, static::
        FIELDS)) {
      if(!$ret) {
        throw new DataException("Error adding
            entity to the database. " .
            get_class($this));
      }

      $this->_id = (int) $ret;
    }else{
      $this->_id = (int) $insert_data[static::
          PRIMARY_KEY];
    }

    $this->_data = $insert_data;
  }
}
```

*php/Dynavis/Core/NotFoundException.class.php*

```
<?php
namespace Dynavis\Core;

class NotFoundException extends \
    RuntimeException {}
```

*php/Dynavis/Core/RefEntity.class.php*

```
<?php
namespace Dynavis\Core;

abstract class RefEntity extends Entity {
  function __construct($id_or_data = null,
      $check_or_refs = false, $check = true)
      {
    if(!is_array($check_or_refs)) {
      if(is_bool($check_or_refs)){
        $check = $check_or_refs;
      }else{
        throw new \InvalidArgumentException("
            Invalid second parameter. Needs
            boolean.");
      }
    }
    parent::__construct($id_or_data, $check);
    if(is_array($check_or_refs)) $this->set(
        $check_or_refs);
  }

  abstract public function set($references);
}
```

*php/Dynavis/DataProcessor.class.php*

```
<?php
namespace Dynavis;
use \PDO;
use \Dynavis\Database;
use \Dynavis\Model\Official;
use \Dynavis\Model\Family;
use \Dynavis\Model\Area;
use \Dynavis\Model\Elect;
use \Dynavis\Model\Dataset;
use \Dynavis\Model\Datapoint;
use \Dynavis\Model\TagDatapoint;
use \Dynavis\Model\User;

class DataProcessor {
  const INDICATORS = [
    "DYNSHA" => ["calculate_dynsha", ["code
        "]],
    "DYNLAR" => ["calculate_dynlar", ["code
        "]],
    "DYNHERF" => ["calculate_dynherf", ["code
        "]],
    "LocalDynastySize" => ["
        calculate_localdynastysize", ["code",
        "id"]],
    "RecursiveDynastySize" => ["
        calculate_recursivedynastysize", ["
        code", "id"]],
  ];

  private static function save_partial(
      $dataset, $insert_data) {
    $values_string = "(" . join("),(",
        array_map(
      function ($row) {
        return join(",", array_map(
          function ($x) {
            return is_null($x) ? "NULL" :
                Database::get()->quote($x);
          },
```

```
            $row
          ));
        },
        $insert_data
    )) . ")";

    if($dataset->type === 0) {
      $ret = Database::get()->query("insert
          into " . Datapoint::TABLE . " (
          dataset_id,year,area_code,value)
          values " . $values_string);
    }else{
      $ret = Database::get()->query("insert
          into " . TagDatapoint::TABLE . " (
          dataset_id,year,area_code,family_id
          ,value) values " . $values_string);
    }

    if(!$ret) {
      Database::get()->pdo->rollBack();
      throw new \Dynavis\Core\DataException("
          Cannot insert into database. " .
          Database::get()->error()[2]);
    }
  }

  public static function generate_indicator(
      $name, $description, $user) {
    // Intensive data processing ahead, set a
        longer time limit
    set_time_limit(120); // 2 minutes

    $p = static::INDICATORS[$name];
    $calc_function = $p[0];
    $variables = $p[1];

    Database::get()->pdo->beginTransaction();

    $dataset = new Dataset(null, ["user" =>
        $user]);
    $dataset->name = $name;
    $dataset->description = $description;
    $dataset->type = count($variables) - 1; //
        FIXME: Dangerous!
    $dataset->save();

    $did = $dataset->get_id();
    $min_year = Database::get()->min(Elect::
        TABLE, "year");
    $max_year = Database::get()->max(Elect::
        TABLE, "year_end") - 1;

    $insert_data = [];
    $count = 0;

    for($t = $min_year; $t <= $max_year; $t++)
        {
      $subresult = static::$calc_function($t);
      foreach ($subresult as $result_row) {
        $insert_row = [
          $did,
          $t,
        ];
        foreach ($variables as $v) {
          $insert_row[] = $result_row[$v];
        }
        $insert_row[] = $result_row[$name];

        $insert_data[] = $insert_row;
        if($count++ > 1000) {
          static::save_partial($dataset,
              $insert_data);
          $insert_data = [];
          $count = 0;
        }
      }
    }

    static::save_partial($dataset,
        $insert_data);

    Database::get()->pdo->commit();

    return $dataset;
  }

  private static function calculate_dynsha(
      $year) {
    // DYNSHA(a,t) = | Officials(a,t) union
        IsMembers(t) |
    $query =
      " select "
        . Area::TABLE . ".code "
        . " , count(" . Family::
            TABLE_FAMILY_MEMBERSHIP . ".
            family_id) AS Size "
        . " , count(*) AS Total "
        . " , 100 * count(" . Family::
            TABLE_FAMILY_MEMBERSHIP . ".
            family_id) / count(*) AS DYNSHA "
      . " from " . Area::TABLE

      . " inner join " . Elect::TABLE
        . " on " . Area::TABLE . ".code = " .
            Elect::TABLE . ".area_code "
      . " inner join " . Official::TABLE
        . " on " . Elect::TABLE . ".
            official_id = " . Official::TABLE
            . "." . Official::PRIMARY_KEY
      . " left join " . Family::
          TABLE_FAMILY_MEMBERSHIP
        . " on " . Official::TABLE . "." .
            Official::PRIMARY_KEY . " = " .
            Family::TABLE_FAMILY_MEMBERSHIP .
            ".official_id "

      . " where "
        . Elect::TABLE . ".year <= :year "
        . " and " . Elect::TABLE . ".year_end
            > :year "
      . " group by " . Area::TABLE . ".code ";

    $st = Database::get()->pdo->prepare($query
        );
    $st->bindParam(":year", $year, PDO::
        PARAM_INT);
    $st->execute();
    return $st->fetchAll();
  }

  private static function calculate_dynlar(
      $year) {
    // DYNLAR(a,t) = max over f in F [ |
        DynOfficials(a,f,t)| ]

    // Gets the total elected of each area ->
        Total
    $subsubquery =
      " select "
        . Area::TABLE . ".code "
        . " , count(*) AS Total "
      . " from " . Area::TABLE

      . " inner join " . Elect::TABLE
        . " on " . Area::TABLE . ".code = " .
            Elect::TABLE . ".area_code "
      . " inner join " . Official::TABLE
        . " on " . Elect::TABLE . ".
            official_id = " . Official::TABLE
            . "." . Official::PRIMARY_KEY

      . " where "
        . Elect::TABLE . ".year <= :year "
        . " and " . Elect::TABLE . ".year_end
            > :year "
      . " group by " . Area::TABLE . ".code ";

    // Gets the share of each dynasty in an
        area -> Size
    $subquery =
      " select "
        . " code "
        . " , " . Family::TABLE . "." . Family
            ::PRIMARY_KEY
        . " , count( " . Family::TABLE . "." .
            Family::PRIMARY_KEY . ") AS Size
            "
        . " , Total "
      . " from "
        . " ( " . $subsubquery . " ) T "

      . " inner join " . Elect::TABLE
        . " on T.code = " . Elect::TABLE . ".
            area_code "
      . " inner join " . Official::TABLE
        . " on " . Elect::TABLE . ".
            official_id = " . Official::TABLE
            . "." . Official::PRIMARY_KEY
      . " left join " . FAMILY::
          TABLE_FAMILY_MEMBERSHIP
        . " on " . Official::TABLE . "." .
            Official::PRIMARY_KEY . " = " .
            Family::TABLE_FAMILY_MEMBERSHIP .
            ".official_id "
      . " left join " . Family::TABLE
        . " on " . Family::TABLE . "." .
            Family::PRIMARY_KEY . " = " .
            Family::TABLE_FAMILY_MEMBERSHIP .
            ".family_id "

      . " where "
        . Elect::TABLE . ".year <= :year "
        . " and " . Elect::TABLE . ".year_end
            > :year "
      . " group by "
        . " code "
        . " , " . Family::
            TABLE_FAMILY_MEMBERSHIP . ".
```

```
            family_id "
      . " order by Size desc ";

    // Gets the dynasty with the largest share
        in each area
    // (this following query works because
        $subquery is ordered by Size
        descending)
    $query =
      " select code, id, Size, Total, 100 *
          Size / Total AS DYNLAR "
      . " from ( " . $subquery . " ) T2 "
      . " group by code ";

    $st = Database::get()->pdo->prepare($query
        );
    $st->bindParam(":year", $year, PDO::
        PARAM_INT);
    $st->execute();
    return $st->fetchAll();
  }

  private static function calculate_dynherf(
      $year) {
    // DYNHERF(a,t) = sum over f in F [
        DynOfficials(a,f,t)^2 ]

    // Gets the total elected of each area ->
        Total
    $subsubquery =
      " select "
        . Area::TABLE . ".code "
        . " , count(*) AS Total "
      . " from " . Area::TABLE

      . " inner join " . Elect::TABLE
        . " on " . Area::TABLE . ".code = " .
            Elect::TABLE . ".area_code "
      . " inner join " . Official::TABLE
        . " on " . Elect::TABLE . ".
            official_id = " . Official::TABLE
            . "." . Official::PRIMARY_KEY

      . " where "
        . Elect::TABLE . ".year <= :year "
        . " and " . Elect::TABLE . ".year_end
            > :year "
      . " group by " . Area::TABLE . ".code ";

    // Gets the share of each dynasty in an
        area -> Size
    $subquery =
      " select "
        . " code "
        . " , count( " . Family::TABLE . "." .
            Family::PRIMARY_KEY . ") AS Size
            "
        . " , Total "
      . " from "
        . " ( " . $subsubquery . " ) T "

      . " inner join " . Elect::TABLE
        . " on T.code = " . Elect::TABLE . ".
            area_code "
      . " inner join " . Official::TABLE
        . " on " . Elect::TABLE . ".
            official_id = " . Official::TABLE
            . "." . Official::PRIMARY_KEY
      . " left join " . FAMILY::
          TABLE_FAMILY_MEMBERSHIP
        . " on " . Official::TABLE . "." .
            Official::PRIMARY_KEY . " = " .
            Family::TABLE_FAMILY_MEMBERSHIP .
            ".official_id "
      . " left join " . Family::TABLE
        . " on " . Family::TABLE . "." .
            Family::PRIMARY_KEY . " = " .
            Family::TABLE_FAMILY_MEMBERSHIP .
            ".family_id "

      . " where "
        . Elect::TABLE . ".year <= :year "
        . " and " . Elect::TABLE . ".year_end
            > :year "
      . " group by "
        . " code "
        . " , " . Family::
            TABLE_FAMILY_MEMBERSHIP . ".
            family_id ";

    // Computes the Herfindahl index
    $query =
      " select code, Total, SUM(POW(Size /
          Total, 2)) AS DYNHERF "
      . " from ( " . $subquery . " ) T2 "
      . " group by code ";

    $st = Database::get()->pdo->prepare($query
        );
    $st->bindParam(":year", $year, PDO::
        PARAM_INT);
    $st->execute();
    return $st->fetchAll();
  }

  private static function
      calculate_localdynastysize($year) {
    // LocalDynastySize(a,f,t) = |
        DynOfficials(a,f,t) |
    $query =
      " select "
        . Area::TABLE . ".code "
        . " , " . Family::TABLE . "." . Family
            ::PRIMARY_KEY
        . " , COUNT(*) AS LocalDynastySize "
      . " from " . Area::TABLE

      . " inner join " . Elect::TABLE
        . " on " . Area::TABLE . ".code = " .
            Elect::TABLE . ".area_code "
      . " inner join " . Official::TABLE
        . " on " . Elect::TABLE . ".
            official_id = " . Official::TABLE
            . "." . Official::PRIMARY_KEY
      . " inner join " . Family::
          TABLE_FAMILY_MEMBERSHIP
        . " on " . Official::TABLE . "." .
            Official::PRIMARY_KEY . " = " .
            Family::TABLE_FAMILY_MEMBERSHIP .
            ".official_id "
      . " inner join " . Family::TABLE
        . " on " . Family::TABLE . "." .
            Family::PRIMARY_KEY . " = " .
            Family::TABLE_FAMILY_MEMBERSHIP .
            ".family_id "

      . " where "
        . Elect::TABLE . ".year <= :year "
        . " and " . Elect::TABLE . ".year_end
            > :year "
      . " group by "
        . Area::TABLE . ".code "
        . " , " . Family::TABLE . "." . Family
            ::PRIMARY_KEY;

    $st = Database::get()->pdo->prepare($query
        );
    $st->bindParam(":year", $year, PDO::
        PARAM_INT);
    $st->execute();
    return $st->fetchAll();
  }

  private static function
      calculate_recursivedynastysize($year) {
    // RecursiveDynastySize(a,f,t) =
        LocalDynastySize(a,f,t) + sum over s
        in S(a) [ RecursiveDynastySize(s,f,t)
        ]
    $lds = static::calculate_localdynastysize(
        $year);

    $area_code_exists = [];

    $rds_index = [];
    foreach ($lds as $l) {
      $area = (int) $l["code"];
      $family = (int) $l["id"];
      $value = $l["LocalDynastySize"];
      while($area) {
        if(!array_key_exists($area, $rds_index
            )) {
          $rds_index[$area] = [];
        }
        if(!array_key_exists($family,
            $rds_index[$area])) {
          $rds_index[$area][$family] = [
            "code" => $area,
            "id" => $family,
            "RecursiveDynastySize" => 0,
          ];
        }
        $rds_index[$area][$family]["
            RecursiveDynastySize"] += $value;

        // TODO: use parent_code in the Area
            table first
        $area = Area::extract_subcodes($area)
            ["parent_code"];
        if(!array_key_exists($area,
            $area_code_exists)) {
          if(Area::has_code($area)) {
            $area_code_exists[$area] = true;
          }else{
            break;
          }
        }
      }
```

```
    }

    $rds = [];
    foreach ($rds_index as $area => $families)
        {
      foreach ($families as $family => $row) {
        $rds[] = $row;
      }
    }

    return $rds;
  }
```

*php/Dynavis/Database.class.php*

```
<?php
namespace Dynavis;

class Database {
  private static $medoo;
  public static function get() {
    if(!isset(self::$medoo)) self::$medoo =
        new \medoo(DB_CONFIG);
    return self::$medoo;
  }

  public static function normalize_string($str
      ) {
    return preg_replace("/[[:space:]]+/", " ",
        trim($str));
  }
}
```

*php/Dynavis/Model/Area.class.php*

```
<?php
namespace Dynavis\Model;
use \PDO;
use \Dynavis\Database;
use \Dynavis\PSGC;

class Area extends \Dynavis\Core\Entity {
  const TABLE = "area";
  const FIELDS = ["code", "name", "type"];
  const QUERY_FIELDS = ["code", "name"];

  public static function get_by_code($code) {
    $ret = Database::get()->get(static::TABLE,
        static::PRIMARY_KEY, [
      "code" => $code,
    ]);
    if($ret === false) throw new \Dynavis\Core
        \NotFoundException("Code does not
        exist. " . $code);
    return new Area((int) $ret, false);
  }

  public static function has_code($code) {
    return Database::get()->has(static::TABLE,
        ["code" => $code]);
  }

  public static function list_areas($count,
      $start, $level = null, $query = null) {
    if($count < 0 || $start < -1) return false
        ;
    if(!is_null($query) && empty($query))
        return ["total" => 0, "data" => []];

    switch ($level) {
      case "region": $type = 0; break;
      case "province": $type = 1; break;
      case "municipality": $type = 2; break;
      case "barangay": $type = 3; break;
      case null: break;
      default: return false; break;
    }

    $where = " 1 ";
    if(!is_null($level)) {
      $where .= " and type = :type ";
    }
    if(is_null($query)) {
      $uquery = [];
    }else{
      $search_clause = " 1 ";
      $uquery = array_unique($query);
      foreach ($uquery as $k => $v) {
        $word_conditions = " 0 ";
        foreach (static::QUERY_FIELDS as $f) {
          $word_conditions .= " or $f like :
              query_$k";
        }
        $search_clause .= " and (
            $word_conditions) ";
      }
      $where .= " and ($search_clause) ";
    }

    $count_query = " select count(*) "
      . " from " . static::TABLE
      . " where $where ";

    $count_st = Database::get()->pdo->prepare(
        $count_query);
    foreach ($uquery as $k => $v) {
      $count_st->bindValue(":query_$k", "%$v
          %", PDO::PARAM_STR);
    }
    if(!is_null($level)) $count_st->bindParam
        (":type", $type, PDO::PARAM_INT);
    $count_st->execute();
    $total = (int) $count_st->fetch()[0];

    $select_query = " select " . join(",",
        array_merge(static::FIELDS, [static
        ::PRIMARY_KEY]))
      . " from " . static::TABLE
      . " where $where ";
    if($count != 0) {
      $select_query .= " limit :start , :count
          ";
    }

    $select_st = Database::get()->pdo->prepare
        ($select_query);
    foreach ($uquery as $k => $v) {
      $select_st->bindValue(":query_$k", "%$v
          %", PDO::PARAM_STR);
    }
    if(!is_null($level)) $select_st->bindParam
        (":type", $type, PDO::PARAM_INT);
    if($count != 0) {
      $select_st->bindParam(":start", $start,
          PDO::PARAM_INT);
      $select_st->bindParam(":count", $count,
          PDO::PARAM_INT);
    }
    $select_st->execute();

    return [
      "total" => $total,
      "data" => $select_st->fetchAll(),
    ];
  }

  public static function count_areas($level =
      null) {
    switch ($level) {
      case "region": $type = 0; break;
      case "province": $type = 1; break;
      case "municipality": $type = 2; break;
      case "barangay": $type = 3; break;
      case null: break;
      default: return false; break;
    }

    $where = null;
    if(!is_null($level)) {
      $where = ["type" => $type];
    }

    return Database::get()->count(static::
        TABLE, $where);
  }

  public function get_elections() {
    $fields = array_map(function($f) {
      return Elect::TABLE . ".$f";
    }, array_merge(Elect::FIELDS, [Elect::
        PRIMARY_KEY]));

    return array_map(
      function ($data) {
        return new Elect($data, false);
      },
      Database::get()->select(Elect::TABLE, [
        "[><]" . static::TABLE => ["area_code"
            => "code"]
      ], $fields, [
        static::TABLE . "." . static::
            PRIMARY_KEY => $this->get_id()
      ])
```

```
  );
}

public function get_officials($year = False)
    {
  $fields = array_map(function($f) {
    return Official::TABLE . ".$f";
  }, array_merge(Official::FIELDS, [Official
      ::PRIMARY_KEY]));

  $query =
    " select " . join(",", $fields)
    . " from " . Official::TABLE

    . " inner join " . Elect::TABLE
      . " on " . Official::TABLE . "." .
          Official::PRIMARY_KEY . " = " .
          Elect::TABLE . ".official_id "
    . " inner join " . static::TABLE
      . " on " . Elect::TABLE . ".area_code
          = " . static::TABLE . ".code "

    . " where "
      . static::TABLE . "." . static::
          PRIMARY_KEY . " = " . Database::
          get()->quote($this->get_id());

  if($year) {
    $query .= " and year <= " . Database::
        get()->quote($year)
      . " and year_end > " . Database::get()
          ->quote($year);
  }

  return array_map(
    function ($data) {
      return new Official($data, false);
    },
    Database::get()->query($query)->fetchAll
        ()
  );
}

public function jsonSerialize() {
  $data = parent::jsonSerialize();
  $data["level"] = ["region", "province", "
      municipality", "barangay"][$data["
      type"]];
  unset($data["type"]);
  return $data;
}

public function save() {
  // normalize strings
  $this->name = Database::normalize_string(
      $this->name);

  $type = (int) $this->type;
  if($type < 0 || $type >= 4) {
    throw new \Dynavis\Core\DataException("
        Invalid type. " . $type);
  }

  if(!strlen($this->name)) {
    throw new \Dynavis\Core\DataException("
        Empty name.");
  }

  parent::save();
}

public static function file($file) {
  $error = $file["error"];
  if($error != UPLOAD_ERR_OK) {
    throw new \RuntimeException("File upload
        error.");
  }

  $size = $file["size"];
  if($size == 0) {
    throw new \Dynavis\Core\DataException("
        No file uploaded.");
  }

  $handle = fopen($file["tmp_name"], "r");
  if ($handle === FALSE) {
    throw new \RuntimeException("Error
        reading file.", 1);
  }

  $data = [];
  while (($row = fgetcsv($handle)) !== FALSE
      ) {
    $data[] = $row;
  }
  fclose($handle);

  $insert_data = [];
  $r = count($data);
  for ($i = 0; $i < $r; $i++) {
    $row = $data[$i];

    $entry = ["code" => null, "name" => null
        ];
    foreach ($row as $value) {
      if(is_null($entry["code"]) && strlen(
          $value) == 9 && intval($value)) {
        $entry["code"] = $value;
      }else{
        if(is_null($entry["name"]) || strlen
            ($value) > strlen($entry["name
            "])) {
          $entry["name"] = $value;
        }
      }
    }
    if(!is_null($entry["code"])) {
      if(is_null($entry["name"])) $entry["
          name"] = "(Unnamed)";
      $insert_data[] = static::process_row(
          $entry, $i);
    }
  }

  $values_string = "(" . join("),(",
      array_map(
    function ($row) {
      return join(",", array_map(
        function ($x) {
          return is_null($x) ? "NULL" :
              Database::get()->quote($x);
        },
        $row
      ));
    },
    $insert_data
  )) . ")";

  $ret = Database::get()->query("replace
      into " . static::TABLE . " (code,name
      ,type) values " . $values_string);

  if(!$ret) {
    throw new \Dynavis\Core\DataException("
        Error adding file data to database
        .");
  }

  // TODO: What about parent_codes (they're
      null)
}

private static function process_row($entry,
    $row) {
  $code = (int) $entry["code"];
  return [
    $code,
    Database::normalize_string($entry["name
        "]),
    static::extract_level($code),
  ];
}

public static function extract_level($code)
    {
  // Assuming valid PSGC $code
  $str = str_pad($code, 9, "0", STR_PAD_LEFT
      );

  $province_code = substr($str, 2, 2);
  $municipality_code = substr($str, 4, 2);
  $barangay_code = substr($str, 6, 3);

  if($province_code === "00") return 0;
  if($municipality_code === "00") return 1;
  if($barangay_code === "000") return 2;
  return 3;
}

public static function extract_subcodes(
    $code) {
  // Assuming valid PSGC $code
  $str = str_pad($code, 9, "0", STR_PAD_LEFT
      );

  $parent_code = null;
  $mun_id = null;
  $bar_id = null;

  $province_code = substr($str, 2, 2);
  if($province_code === "00") {
    // region.parent = NULL
    $parent_code = null;
  }else{
    $region_code = substr($str, 0, 2);
    $municipality_code = substr($str, 4, 2);
    if($municipality_code === "00") {
```

```
        // province.parent = region
        $parent_code = intval($region_code .
            "0000000");
      }else{
        $mun_id = $province_code .
            $municipality_code;
        $barangay_code = substr($str, 6, 3);
        if($barangay_code === "000") {
          // municipality.parent = province
          $parent_code = intval($region_code .
              $province_code . "00000");
        }else{
          // barangay.parent = municipality
          $bar_id = $mun_id . $barangay_code;
          $parent_code = intval($region_code .
              $mun_id . "000");
        }
      }
    }

    return [
      "parent_code" => $parent_code,
      "mun_id" => $mun_id,
      "bar_id" => $bar_id,
    ];
  }

  public static function get_by_name($name) {
    $names = static::psgc_get_names($name);

    $candidates = [];
    foreach ($names as $n) {
      if(array_key_exists($n, PSGC::MAP)) {
        $c = count(PSGC::MAP[$n]);
        foreach (PSGC::MAP[$n] as $area_code)
            {
          if(!isset($candidates[$area_code]))
              {
            $candidates[$area_code] = 0;
          }
          $level = static::extract_level(
              $area_code);
          $candidates[$area_code] += (4.0 -
              $level)/$c;
        }
      }
    }

    if(empty($candidates)) return null;

    $highest = null;
    foreach ($candidates as $area_code =>
        $prob) {
      if(!$highest || $candidates[$highest] <
          $prob) {
        $highest = $area_code;
      }
    }
    return static::get_by_code($highest);
  }

  /* area name normalization functions */

  private static function psgc_get_names($s) {
    $split = preg_split("/[()]|\s+-\s+/", $s,
        0, PREG_SPLIT_NO_EMPTY);
    $names = [];
    foreach ($split as $value) {
      $nv = static::psgc_normalize($value);
      if($nv) $names[] = $nv;
    }
    $add = [];
    foreach($names as $n){
      $len = strlen($n);
      if(substr_compare($n, "city", $len-4, 4)
          === 0) {
        $add[] = substr($n, $len-4);
      }
    }
    return $names + $add;
  }

  private static function psgc_normalize($s) {
    $s = iconv('UTF-8', 'ASCII//TRANSLIT', $s)
        ;
    $s = preg_replace("/-/", " ", $s);
    $s = strtolower(preg_replace_callback(
        "/\b(?=[CLXVI]+\b)(C){0,3}(XC|XL|L?X
            {0,3})(IX|IV|V?I{0,3})\b/",
        function($matches) {
          $numeral_map = [[100, "C"], [90, "XC
              "], [50, "L"], [40, "XL"], [10,
              "X"], [9, "IX"], [5, "V"], [4,
              "IV"], [1, "I"]];
          $n = strtoupper($matches[0]);
          $i = $result = 0;
          foreach ($numeral_map as $pair) {
            $integer = $pair[0];
            $numeral = $pair[1];
            $len = strlen($numeral);
            while(substr($n, $i, $len) ==
                $numeral){
              $result += $integer;
              $i += $len;
            }
          }
          return strval($result);
        },
        $s
    ));
    $s = preg_replace("/\bcity\s+of(.+)/", "
        $1city", $s);
    $s = preg_replace("/\bbarangay\b/", "bgy",
        $s);
    $s = preg_replace("/\bpoblacion\b/", "pob
        ", $s);
    $s = preg_replace("/\b(general|heneral|hen
        )\b/", "gen", $s);
    $s = preg_replace("#[*-.'/\s]+|of|the#",
        "", $s);
    return $s;
  }
}
```

*php/Dynavis/Model/Datapoint.class.php*

```
<?php
namespace Dynavis\Model;

class Datapoint extends \Dynavis\Core\
    RefEntity {
  const TABLE = "datapoint";
  const FIELDS = ["dataset_id", "year", "
      area_code", "value"];
  const QUERY_FIELDS = null;

  public function set($param) {
    $dataset = $param["dataset"];
    $area = $param["area"];

    if(is_null($dataset->get_id()) || is_null(
        $area->get_id())) {
      throw new \RuntimeException("The dataset
          or the area is not yet stored in
          the database.");
    }

    $area->load();
    $this->dataset_id = $dataset->get_id();
    $this->area_code = $area->code;
  }
}
```

*php/Dynavis/Model/Dataset.class.php*

```
<?php
namespace Dynavis\Model;
use \Dynavis\Database;
use \Dynavis\PSGC;

class Dataset extends \Dynavis\Core\RefEntity
    {
  const TABLE = "dataset";
  const FIELDS = ["user_id", "type", "name", "
      description"];
  const QUERY_FIELDS = ["name", "description
      "];

  public function set($param) {
    $user = $param["user"];

    if(!is_null($user) && is_null($user->
        get_id())) {
      throw new \RuntimeException("The user is
          not yet stored in the database.");
    }

    $this->user_id = is_null($user) ? null :
        $user->get_id();
  }

  public static function list_datasets($count,
      $start, $type = null, $query = null) {
    if($count < 0 || $start < -1) return false
        ;
```

```
    if(!is_null($query) && empty($query))
        return ["total" => 0, "data" => []];

    switch ($type) {
      case "area": $type = 0; break;
      case "tag": $type = 1; break;
      case null: break;
      default: return false; break;
    }

    $where = [];
    if(!is_null($type)) {
      $where["type"] = $type;
    }
    if(!is_null($query)) {
      $where = ["AND" => array_merge($where,
          ["name[~]" => array_unique($query)
          ])];
    }

    $total = Database::get()->count(static::
        TABLE, $where);
    if($count != 0) {
      $where["LIMIT"] = [(int) $start , (int)
          $count];
    }

    return [
      "total" => $total,
      "data" => Database::get()->select(static
          ::TABLE, array_merge(static::FIELDS
          , [static::PRIMARY_KEY]), $where),
    ];
  }

  public function get_points($count = 0,
      $start = 0, $year = null) {
    if($count < 0 || $start < -1) return false
        ;

    $this->load();
    $class = [
      "\\Dynavis\\Model\\Datapoint",
      "\\Dynavis\\Model\\TagDatapoint"
    ][$this->type];

    $join = [
      "[><]" . static::TABLE => ["dataset_id"
          => static::PRIMARY_KEY]
    ];

    $fields = array_merge($class::FIELDS, [
        $class::TABLE . "." . $class::
        PRIMARY_KEY]);
    unset($fields[0]);

    $where = [
      static::TABLE . "." . static::
          PRIMARY_KEY => $this->get_id(),
    ];

    if(!is_null($year)) {
      $where = ["AND" => array_merge($where, [
          $class::TABLE . ".year" => $year])
          ];
    }

    $total = Database::get()->count($class::
        TABLE, $join, "*", $where);
    if($count != 0) {
      $where["LIMIT"] = [(int) $start , (int)
          $count];
    }

    return [
      "total" => $total,
      "data" => Database::get()->select($class
          ::TABLE, $join, $fields, $where)
    ];
  }

  public function jsonSerialize() {
    $data = parent::jsonSerialize();

    $data["username"] = (new User((int) $data
        ["user_id"], false))->username;
    unset($data["user_id"]);

    $class = [
      "\\Dynavis\\Model\\Datapoint",
      "\\Dynavis\\Model\\TagDatapoint"
    ][$data["type"]];

    $join = ["[><]" . static::TABLE => ["
        dataset_id" => static::PRIMARY_KEY]];
    $where = [static::TABLE . "." . static::
        PRIMARY_KEY => $this->get_id()];
    $data["min_year"] = Database::get()->min(
        $class::TABLE, $join, $class::TABLE .
        ".year", $where);
    $data["max_year"] = Database::get()->max(
        $class::TABLE, $join, $class::TABLE .
        ".year", $where);

    $levels = ["region", "province", "
        municipality", "barangay"];
    $contained_levels = [];
    $join = [
      "[><]" . static::TABLE => ["dataset_id"
          => static::PRIMARY_KEY],
      "[><]" . Area::TABLE => ["area_code" =>
          "code"],
    ];
    foreach ($levels as $key => $value) {
      $contained_levels[$value] = Database::
          get()->has($class::TABLE, $join, ["
          AND" => [
        static::TABLE . "." . static::
            PRIMARY_KEY => $this->get_id(),
        Area::TABLE . ".type" => $key,
      ]]);
    }
    $data["contained_levels"] =
        $contained_levels;

    $data["type"] = ["area", "tag"][$data["
        type"]];

    return $data;
  }

  public function file($file) {
    $error = $file["error"];
    if($error != UPLOAD_ERR_OK) {
      throw new \RuntimeException("File upload
          error.");
    }

    $size = $file["size"];
    if($size == 0) {
      throw new \Dynavis\Core\DataException("
          No file uploaded.");
    }

    $handle = fopen($file["tmp_name"], "r");
    if ($handle === FALSE) {
      throw new \RuntimeException("Error
          reading file.", 1);
    }

    $data = [];
    while (($row = fgetcsv($handle)) !== FALSE
        ) {
      $data[] = $row;
    }
    fclose($handle);

    $header = $data[0];
    $c = count($header);
    for ($i = 1; $i < $c; $i++) {
      $year = (int) $header[$i];
      if($year === 0) {
        throw new \Dynavis\Core\DataException
            ("Invalid year format in header
            row. " . $header[i]);
      }
      $header[$i] = $year;
    }

    $insert_data = [];

    $r = count($data);
    for ($i = 1; $i < $r; $i++) {
      $row = $data[$i];
      if(count($row) !== $c) {
        throw new \Dynavis\Core\DataException
            ("Incorrect number of columns in
            row. Expected " . $c . ". Got " .
            count($row) . " at row " . ($i +
            1) . ": " . join(",", $row));
      }

      $area_code = (int) $row[0];
      try {
        $area = Area::get_by_code($area_code);
      }catch(\Dynavis\Core\NotFoundException
          $e) {
        $area = Area::get_by_name($row[0]);
        if(!$area) {
          $why = $area_code ? "Invalid PSGC
              code. " : "Name not recognized.
              ";
          throw new \Dynavis\Core\
              DataException("Invalid area
              format. " . $why . $row[0] . "
              at row " . ($i + 1));
        }
        $area_code = $area->code;
```

```
      }

      for ($j = 1; $j < $c; $j++) {
        if(!preg_match("/^(\+|-)?((\d{0,65}\.\
            d{0,30})|\d{1,65})$/", $row[$j]))
              {
          throw new \Dynavis\Core\
              DataException("Invalid number
              format. " . $row[$j] . " at row
               " . ($i + 1));
        }
        $insert_data[] = [
          "dataset_id" => $this->get_id(),
          "year"  => $header[$j],
          "area_code" => $area_code,
          "value" => strlen($row[$j]) ? $row[
              $j] : null,
        ];
      }
    }

    $values_string = "(" . join("),(",
        array_map(
      function ($row) {
        return join(",", array_map(
          function ($x) {
            return is_null($x) ? "NULL" :
                Database::get()->quote($x);
          },
          $row
        ));
      },
      $insert_data
    )) . ")";

    $ret = Database::get()->query("insert into
         " . Datapoint::TABLE . " (dataset_id
        ,year,area_code,value) values " .
        $values_string);

    if(!$ret) {
      throw new \Dynavis\Core\DataException("
          Error adding file data to the
          database.");
    }
  }
}
```

*php/Dynavis/Model/Elect.class.php*

```
<?php
namespace Dynavis\Model;
use \PDO;
use \Dynavis\Database;

class Elect extends \Dynavis\Core\RefEntity {
  const TABLE = "elect";
  const FIELDS = [
    "official_id",
    "year",
    "year_end",
    "position",
    "votes",
    "area_code",
    "party_id",
  ];
  const QUERY_FIELDS = ["year", "position"];

  public function set($param) {
    $official = $param["official"];
    $area = $param["area"];
    $party = $param["party"];

    if(is_null($official->get_id()) || is_null
        ($area->get_id()) || (!is_null($party
        ) && is_null($party->get_id()))) {
      throw new \RuntimeException("The
          official, the area, or the party is
           not yet stored in the database.");
    }

    $area->load();
    $this->official_id = $official->get_id();
    $this->area_code = $area->code;
    $this->party_id = is_null($party) ? null :
         $party->get_id();
  }

  public static function list_items($count,
      $start) {
    return static::query_items($count, $start)
        ;
  }

  public static function query_items($count,
      $start, $query = null) {
    if($count < 0 || $start < -1) return false
        ;
    if(!is_null($query) && empty($query))
        return ["total" => 0, "data" => []];

    $classes = ["Elect", "Official", "Area", "
        Party"];
    $joins = "";

    $where = " 1 ";
    if(is_null($query)) {
      $uquery = null;
    }else{
      $joins = " inner join " . Official::
          TABLE
          . " on " . static::TABLE . ".
              official_id = " . Official::
              TABLE . "." . Official::
              PRIMARY_KEY
        . " inner join " . Area::TABLE
        . " on " . static::TABLE . ".
            area_code = " . Area::TABLE .
            ".code "
        ." left join " . Party::TABLE
        . " on " . static::TABLE . ".
            party_id = " . Party::TABLE .
            "." . Party::PRIMARY_KEY;

      $search_clause = " 0 ";
      $uquery = array_unique($query);
      foreach ($classes as $class) {
        $full_class = "\\Dynavis\\Model\\
            $class";
        foreach ($full_class::QUERY_FIELDS as
            $f) {
          $field_conditions = " 1 ";
          foreach ($uquery as $k => $v) {
            $field_conditions .= " and " .
                $full_class::TABLE . ".$f
                like :query_{$class}_{$f}_{$k
                }";
          }
          $search_clause .= " or (
              $field_conditions) ";
        }
      }
      $where .= " and ($search_clause) ";
    }

    function bind($statement, $uquery,
        $classes) {
      if(!is_null($uquery)) {
        foreach ($classes as $class) {
          $full_class = "\\Dynavis\\Model\\
              $class";
          foreach ($full_class::QUERY_FIELDS
              as $f) {
            foreach ($uquery as $k => $v) {
              $statement->bindValue(":query_{
                  $class}_{$f}_{$k}", "%$v%",
                  PDO::PARAM_STR);
            }
          }
        }
      }

      $count_query = " select count(*) "
        . " from " . static::TABLE
        . $joins
        . " where $where ";

      $count_st = Database::get()->pdo->prepare(
          $count_query);
      bind($count_st, $uquery, $classes);
      $count_st->execute();
      $total = (int) $count_st->fetch()[0];

      $fields = array_map(function($f) {
        return static::TABLE . ".$f";
      }, array_merge(static::FIELDS, [static::
          PRIMARY_KEY]));

      $select_query = " select " . join(",",
          $fields)
        . " from " . static::TABLE
        . $joins
        . " where $where ";
      if($count != 0) {
        $select_query .= " limit :start , :count
            ";
      }
```

```
    $select_st = Database::get()->pdo->prepare
        ($select_query);
    bind($select_st, $uquery, $classes);
    if($count != 0) {
      $select_st->bindParam(":start", $start,
          PDO::PARAM_INT);
      $select_st->bindParam(":count", $count,
          PDO::PARAM_INT);
    }
    $select_st->execute();

    return [
      "total" => $total,
      "data" => $select_st->fetchAll(),
    ];
  }

  public function save() {
    if($this->year >= $this->year_end) {
      throw new \Dynavis\Core\DataException("'
          year' must be less than 'year_end
          '.");
    }

    // normalize strings
    $this->position = $this->position && trim(
        $this->position) ? Database::
        normalize_string($this->position) :
        null;

    // Official cannot be in two posts
        simultaneously
    $official_overlaps = Database::get()->
        select(static::TABLE, static::
        PRIMARY_KEY, ["AND" => [
      static::PRIMARY_KEY . "[!]" => $this->
          get_id(),
      "year[<]" => $this->year_end,
      "year_end[>]" => $this->year,
      "official_id" => $this->official_id,
    ]]);

    if(count($official_overlaps)) {
      throw new \Dynavis\Core\DataException("
          No official can be in two posts
          simultaneously."
        . " Official ID: " . $this->
            official_id
        . " Conflicts: " . join(", ",
            array_map(function($id) {
          return $id;
        }, $official_overlaps)));
    }

    parent::save();
  }

  public function delete() {
    $official = new Official((int) $this->
        official_id, false);
    parent::delete();
    $official->autodelete();
  }

  public static function file($file) {
    $error = $file["error"];
    if($error != UPLOAD_ERR_OK) {
      throw new \RuntimeException("File upload
          error.");
    }

    $size = $file["size"];
    if($size == 0) {
      throw new \Dynavis\Core\DataException("
          No file uploaded.");
    }

    $handle = fopen($file["tmp_name"], "r");
    if ($handle === FALSE) {
      throw new \RuntimeException("Error
          reading file.", 1);
    }

    $data = [];
    while (($row = fgetcsv($handle)) !== FALSE
        ) {
      $data[] = $row;
    }
    fclose($handle);

    $file_fields = ["area", "year", "position
        ", "surname", "name", "nickname", "
        party", "votes"];
    $num_fields = count($file_fields);

    $insert_data = [];

    $r = count($data);
    for ($i = 0; $i < $r; $i++) {
      $row = $data[$i];
      if(count($row) !== $num_fields) {
        throw new \Dynavis\Core\DataException
            ("Incorrect number of columns in
            row. Expected " . $num_fields .
            ". Got " . count($row) . " at row
             " . ($i + 1) . ": " . join(",",
            $row));
      }
      $entry = [];
      foreach ($row as $key => $value) {
        $entry[$file_fields[$key]] = $value;
      }
      $insert_data[] = static::process_row(
          $entry, $i);
    }

    // No checking for overlaps. Assume
        correct data.

    $values_string = "(" . join("),(",
        array_map(
      function ($row) {
        return join(",", array_map(
          function ($x) {
            return is_null($x) ? "NULL" :
                Database::get()->quote($x);
          },
          $row
        ));
      },
      $insert_data
    )) . ")";

    $ret = Database::get()->query("insert into
        " . static::TABLE . " (official_id,
        year,year_end,position,votes,
        area_code,party_id) values " .
        $values_string);

    if(!$ret) {
      throw new \Dynavis\Core\DataException("
          Error adding file data to database
          .");
    }
  }

  private static function process_row($entry,
      $row) {
    $surname = Database::normalize_string(
        $entry["surname"]);
    $name = preg_replace("/[.,]/", " ",
        Database::normalize_string($entry["
        name"]));
    $official = Official::get_by_name($surname
        , $name);
    if(!$official) {
      $official = new Official();
      $official->surname = $surname;
      $official->name = $name;
      $official->nickname = $entry["nickname
          "];
      $official->save();
    }

    $year = (int) $entry["year"];
    if(!$year) {
      throw new \Dynavis\Core\DataException("
          Invalid year format. " . $entry["
          year"] . " at row " . ($row + 1));
    }
    // The length of term for LGU positions is
        three (3) years.
    $year_end = $year + 3; // TODO: how about
        special elections or rescheduled
        elections

    $position = $entry["position"];
    $position = $position && trim($position) ?
        Database::normalize_string($position
        ) : null;

    $votes = (int) $entry["votes"];
    if(!is_numeric($votes)) {
      throw new \Dynavis\Core\DataException("
          Invalid votes format. " . $entry["
          votes"] . " at row " . ($row + 1));
    }

    $area = null;
    $area_code = (int) $entry["area"];
    if($area_code) {
      try{
        $area = Area::get_by_code($area_code);
      }catch(\Dynavis\Core\NotFoundException
          $e) {
        throw new \Dynavis\Core\DataException
            ("Invalid area code. " . $entry["
            area"] . " at row " . ($row + 1))
```

```
            ;
        }
      }else{
        $area = Area::get_by_name($entry["area
            "]);
        if(!$area) {
          throw new \Dynavis\Core\DataException
              ("Invalid area name. " . $entry["
              area"] . " at row " . ($row + 1))
              ;
        }
      }

      $party = null;
      if(!empty($entry["party"]) && $entry["
          party"] !== "IND") { // IND = coding
          for independent
        try{
          $party = Party::get_by_name($entry["
              party"]);
        }catch(\Dynavis\Core\NotFoundException
            $e) {
          $party = new Party();
          $party->name = $entry["party"];
          $party->save();
        }
      }

      return [
        $official->get_id(),
        $year,
        $year_end,
        $position,
        $votes,
        $area->code,
        is_null($party) ? null : $party->get_id
            (),
      ];
    }
  }
```

*php/Dynavis/Model/Family.class.php*

```
<?php
namespace Dynavis\Model;
use \Dynavis\Database;

class Family extends \Dynavis\Core\Entity {
  const TABLE = "family";
  const FIELDS = ["name"];
  const QUERY_FIELDS = ["name"];

  const TABLE_FAMILY_MEMBERSHIP = "
      family_membership";

  public static function get_by_name($name) {
    $ret = Database::get()->get(static::TABLE,
        static::PRIMARY_KEY, [
      "name" => Database::normalize_string(
          $name),
    ]);
    if($ret === false) throw new \Dynavis\Core
        \NotFoundException("Name not found.
        $name");
    return new Family((int) $ret, false);
  }

  public function add_member($official) {
    if($this->is_member($official)) {
      throw new \Dynavis\Core\DataException("
          Cannot add an already-added member
          .");
    }

    Database::get()->insert(static::
        TABLE_FAMILY_MEMBERSHIP, [
      "official_id" => $official->get_id(),
      "family_id" => $this->get_id(),
    ]);
  }

  public function get_members($year = False) {
    $fields = array_map(function($f) {
      return Official::TABLE . ".$f";
    }, array_merge(Official::FIELDS, [Official
        ::PRIMARY_KEY]));

    $query =
      " select " . join(",", $fields)
      . " from " . Official::TABLE

      . " inner join " . static::
          TABLE_FAMILY_MEMBERSHIP
        . " on " . Official::TABLE . "." .
            Official::PRIMARY_KEY . " = " .
            static::TABLE_FAMILY_MEMBERSHIP .
            ".official_id"
      . " inner join " . static::TABLE
        . " on " . static::
            TABLE_FAMILY_MEMBERSHIP . ".
            family_id = " . static::TABLE .
            "." . static::PRIMARY_KEY;

    if($year) {
      $query .=
        " inner join " . Elect::TABLE . " e1 "
        . "on " . Official::TABLE . "." .
            Official::PRIMARY_KEY . " = e1.
            official_id "
        . " left join " . Elect::TABLE . " e2
            "
        . " on e1.official_id = e2.
            official_id and e1.year > e2.
            year ";
    }

    $query .= " where " . static::TABLE . "."
        . static::PRIMARY_KEY . " = " .
        Database::get()->quote($this->get_id
        ());

    if($year) {
      $query .=
        " and e2." . Elect::PRIMARY_KEY . " is
            null"
        . " and e1.year <= " . Database::get()
            ->quote($year);
    }

    return array_map(
      function ($data) {
        return new Official($data, false);
      },
      Database::get()->query($query)->fetchAll
          ()
    );
  }

  public function count_members() {
    return Database::get()->count(Official::
        TABLE, [
      "[><]" . static::TABLE_FAMILY_MEMBERSHIP
          => [Official::PRIMARY_KEY => "
          official_id"],
      "[><]" . static::TABLE => [static::
          TABLE_FAMILY_MEMBERSHIP . ".
          family_id" => static::PRIMARY_KEY],
    ], Official::TABLE . "." . Official::
        PRIMARY_KEY, [
      static::TABLE . "." . static::
          PRIMARY_KEY => $this->get_id()
    ]);
  }

  public function is_member($official) {
    if(is_null($official->get_id()) or is_null
        ($this->get_id())) {
      throw new \RuntimeException("The
          official or the family is not yet
          stored in the database.");
    }

    return Database::get()->has(static::
        TABLE_FAMILY_MEMBERSHIP, [ "AND" => [
      "official_id" => $official->get_id(),
      "family_id" => $this->get_id(),
    ]]);
  }

  public function remove_member($official) {
    if(!$this->is_member($official)) {
      throw new \Dynavis\Core\DataException("
          Cannot remove a non-member.");
    }

    $ret = Database::get()->delete(static::
        TABLE_FAMILY_MEMBERSHIP, [ "AND" => [
      "official_id" => $official->get_id(),
      "family_id" => $this->get_id(),
    ]]);

    if(!$ret) {
      throw new \Dynavis\Core\DataException("
          Error removing family membership
          from the database.");
    }

    $this->autodelete();
  }
```

```
  public function autodelete() {
    if($this->count_members() == 0) {
      $this->delete();
      return true;
    }
    return false;
  }

  public function save() {
    // normalize strings
    $this->name = Database::normalize_string(
        $this->name);

    if(!strlen($this->name)) {
      throw new \Dynavis\Core\DataException("
          Empty name.");
    }

    parent::save();
  }
}
```

*php/Dynavis/Model/Official.class.php*

```
<?php
namespace Dynavis\Model;
use \Dynavis\Database;

class Official extends \Dynavis\Core\Entity {
  const TABLE = "official";
  const FIELDS = ["surname", "name", "nickname
      "];
  const QUERY_FIELDS = ["surname", "name", "
      nickname"];

  const TABLE_FAMILY_MEMBERSHIP = "
      family_membership";

  public function get_families() {
    $fields = array_map(function($f) {
      return Family::TABLE . ".$f";
    }, array_merge(Family::FIELDS, [Family::
        PRIMARY_KEY]));

    return array_map(
      function ($data) {
        return new Family($data, false);
      },
      Database::get()->select(Family::TABLE, [
        "[><]" . static::
            TABLE_FAMILY_MEMBERSHIP => [
            Family::PRIMARY_KEY => "family_id
            "],
        "[><]" . static::TABLE => [static::
            TABLE_FAMILY_MEMBERSHIP . ".
            official_id" => static::
            PRIMARY_KEY],
      ], $fields, [
        static::TABLE . "." . static::
            PRIMARY_KEY => $this->get_id()
      ])
    );
  }

  public function get_elections() {
    $fields = array_map(function($f) {
      return Elect::TABLE . ".$f";
    }, array_merge(Elect::FIELDS, [Elect::
        PRIMARY_KEY]));

    return array_map(
      function ($data) {
        return new Elect($data, false);
      },
      Database::get()->select(Elect::TABLE,
          $fields, [
        Elect::TABLE . ".official_id" => $this
            ->get_id()
      ])
    );
  }

  public function save() {
    // normalize strings
    $this->surname = Database::
        normalize_string($this->surname);
    $this->name = Database::normalize_string(
        $this->name);
    $this->nickname = trim($this->nickname) ?
        Database::normalize_string($this->
        nickname) : null;

    if(!strlen($this->surname) || !strlen(
        $this->name)) {
      throw new \Dynavis\Core\DataException("
          Required fields empty. (surname,
          name)");
    }

    parent::save();
  }

  public function autodelete() {
    $elections = Database::get()->count(Elect
        ::TABLE, [
      Elect::TABLE . ".official_id" => $this->
          get_id()
    ]);

    if($elections === 0) {
      $this->delete();
      return true;
    }
    return false;
  }

  public function delete() {
    Database::get()->pdo->beginTransaction();

    try{
      // Remove membership before deletion
      foreach ($this->get_families() as $f) {
        $f->remove_member($this);
      }

      parent::delete();
    }catch(Exception $e) {
      Database::get()->pdo->rollback();
      throw e;
    }

    Database::get()->pdo->commit();
  }

  public static function get_by_name($surname,
      $name) {
    $ret = Database::get()->get(static::TABLE,
        static::PRIMARY_KEY, ["AND" => [
      "surname" => Database::normalize_string(
          $surname),
      "name" => Database::normalize_string(
          $name),
    ]]);
    if(!$ret) return null;
    return new Official((int) $ret, false);
  }
}
```

*php/Dynavis/Model/Party.class.php*

```
<?php
namespace Dynavis\Model;
use \Dynavis\Database;

class Party extends \Dynavis\Core\Entity {
  const TABLE = "party";
  const FIELDS = ["name"];
  const QUERY_FIELDS = ["name"];

  public function get_elections() {
    $fields = array_map(function($f) {
      return Elect::TABLE . ".$f";
    }, array_merge(Elect::FIELDS, [Elect::
        PRIMARY_KEY]));

    return array_map(
      function ($data) {
        return new Elect($data, false);
      },
      Database::get()->select(Elect::TABLE, [
        "[><]" . static::TABLE => ["party_id"
            => static::PRIMARY_KEY]
      ], $fields, [
        static::TABLE . "." . static::
            PRIMARY_KEY => $this->get_id()
      ])
    );
  }

  public static function get_by_name($name) {
    $ret = Database::get()->get(static::TABLE,
        static::PRIMARY_KEY, [
      "name" => Database::normalize_string(
          $name),
```

```
    ]);
    if($ret === false) throw new \Dynavis\Core
        \NotFoundException("Name not found.
        $name");
    return new Party((int) $ret, false);
  }

  public function save() {
    // normalize strings
    $this->name = Database::normalize_string(
        $this->name);
    if(!strlen($this->name)) {
      throw new \Dynavis\Core\DataException("
          Empty name.");
    }

    parent::save();
  }
}
```

*php/Dynavis/Model/TagDatapoint.class.php*

```
<?php
namespace Dynavis\Model;

class TagDatapoint extends \Dynavis\Core\
    RefEntity {
  const TABLE = "tag_datapoint";
  const FIELDS = ["dataset_id", "year", "
      area_code", "family_id", "value"];
  const QUERY_FIELDS = null;

  public function set($param) {
    $dataset = $param["dataset"];
    $area = $param["area"];
    $family = $param["family"];
    if(is_null($dataset->get_id()) || is_null(
        $area->get_id()) || is_null($family->
        get_id())) {
      throw new \RuntimeException("The dataset
          , the area, or the family is not
          yet stored in the database.");
    }

    $area->load();
    $this->dataset_id = $dataset->get_id();
    $this->area_code = $area->code;
    $this->family_id = $family->get_id();
  }
}
```

*php/Dynavis/Model/Token.class.php*

```
<?php
namespace Dynavis\Model;
use \Dynavis\Database;

class Token extends \Dynavis\Core\RefEntity {
  const TABLE = "token";
  const FIELDS = [
    "user_id",
    "token",
    "expiry",
  ];
  const QUERY_FIELDS = null;

  public function set($param) {
    $user = $param["user"];

    if(is_null($user->get_id())) {
      throw new \RuntimeException("The user is
          not yet stored in the database.");
    }

    $this->user_id = $user->get_id();
    $this->token = base64_encode(
        openssl_random_pseudo_bytes(96));
    $this->refresh();
  }

  public static function get_by_token($token)
      {
    $ret = Database::get()->get(static::TABLE,
        static::PRIMARY_KEY, ["token" =>
        $token]);
    if($ret === false) throw new \Dynavis\Core
        \NotFoundException("Token not found.
        $token");
    return new Token((int) $ret, false);
  }

  public function get_user() {
    $this->load();
    return new User($this->user_id, false);
  }

  public function valid() {
    $this->load();
    $now = new \DateTime();
    $expiry = new \DateTime($this->expiry);
    return $now < $expiry;
  }

  public function set_expiry($datetime) {
    $this->load();
    $this->expiry = $datetime->format("Y-m-d H
        :i:s");
  }

  public function refresh() {
    $this->set_expiry(new \DateTime("+10 days
        "));
  }

  public function jsonSerialize() {
    $data = parent::jsonSerialize();
    $data["username"] = (new User((int) $data
        ["user_id"], false))->username;
    unset($data["user_id"]);
    return $data;
  }

  public static function cleanup() {
    $dt = new \DateTime("-10 hours");
    Database::get()->delete(static::TABLE, ["
        expiry[<]" => $dt->format("Y-m-d H:i:
        s")]);
  }
}
```

*php/Dynavis/Model/User.class.php*

```
<?php
namespace Dynavis\Model;
use \Dynavis\Database;

class User extends \Dynavis\Core\Entity {
  const TABLE = "user";
  const FIELDS = [
    "active",
    "username",
    "pw_hash",
    "type",
    "salt",
  ];
  const QUERY_FIELDS = ["username"];

  public static function get_by_username(
      $username) {
    $ret = Database::get()->get(static::TABLE,
        static::PRIMARY_KEY, [
      "username" => Database::normalize_string
          ($username)
    ]);
    if($ret === false) throw new \Dynavis\Core
        \NotFoundException("Username not
        found. $username");
    return new User((int) $ret, false);
  }

  public function count_datasets($type = null)
      {
    $where = [
      static::TABLE . "." . static::
          PRIMARY_KEY => $this->get_id()
    ];

    if(!is_null($type)) {
      switch ($type) {
        case "area": $type = 0; break;
        case "tag": $type = 1; break;
        default: $app->halt(400, "Invalid type
            . " . $type);
      }
      $where = ["AND" => array_merge($where,
          ["type" => $type])];
```

```
    }

    return Database::get()->count(Dataset::
        TABLE, [
      "[><]" . static::TABLE => ["user_id" =>
          static::PRIMARY_KEY]
    ], Dataset::TABLE . "." . Dataset::
        PRIMARY_KEY, $where);
  }

  public function get_datasets($count, $start,
      $type = null) {
    if($count < 0 || $start < -1) return false
        ;

    $where = [
      static::TABLE . "." . static::
          PRIMARY_KEY => $this->get_id()
    ];

    if(!is_null($type)) {
      switch ($type) {
        case "area": $type = 0; break;
        case "tag": $type = 1; break;
        default: $app->halt(400, "Invalid type
            . " . $type);
      }
      $where = ["AND" => array_merge($where, [
          Dataset::TABLE . ".type" => $type])
          ];
    }

    if($count != 0) {
      $where["LIMIT"] = [(int) $start , (int)
          $count];
    }

    $fields = array_map(function($f) {
      return Dataset::TABLE . ".$f";
    }, array_merge(Dataset::FIELDS, [Dataset::
        PRIMARY_KEY]));

    return array_map(
      function ($data) {
        return new Dataset($data, false);
      },
      Database::get()->select(Dataset::TABLE,
          [
        "[><]" . static::TABLE => ["user_id"
            => static::PRIMARY_KEY]
      ], $fields, $where)
    );
  }

  public function set_password($password) {
    $this->load();
    $this->salt = base64_encode(
        mcrypt_create_iv(96));
    $this->pw_hash = $this->hash_password(
        $password);
  }

  public function check_password($password) {
    $this->load();
    if(is_null($this->pw_hash) || is_null(
        $this->salt)) {
      throw new \RuntimeException("Password
          not set yet.");
    }

    return $this->pw_hash == $this->
        hash_password($password);
  }

  private function hash_password($password) {
    $this->load();
    return password_hash($password . substr(
        $this->salt,4,4), PASSWORD_BCRYPT, ["
        salt" => $this->salt]);
  }

  public static function list_items($count,
      $start) {
    if($count < 0 || $start < -1) return false
        ;
    $where = $count == 0 ? [] : ["LIMIT" => [(
        int) $start , (int) $count]];
    $where["ORDER"] = "active ASC";
    return [
      "total" => static::count(),
      "data" => Database::get()->select(static
          ::TABLE, array_merge(static::FIELDS
          , [static::PRIMARY_KEY]), $where),
    ];
  }

  public function save() {
    if(static::count() == 0) {
      // First user is admin
      $this->active = true;
      $this->type = 1;
    }
    parent::save();
  }

  public function jsonSerialize() {
    $data = parent::jsonSerialize();
    $data["active"] = $data["active"] != 0;
    $data["role"] = ["user", "admin"][$data["
        type"]];
    unset($data["id"]);
    unset($data["type"]);
    unset($data["pw_hash"]);
    unset($data["salt"]);
    return $data;
  }
}
```

*php/db_config.include.sample.php*

```
<?php
const DB_CONFIG = [
  'database_type' => 'mysql',
  'database_name' => 'datab',
  'server' => 'localhost',
  'username' => 'user',
  'password' => 'pass',
  'charset' => 'utf8',
];
```

*php/init.include.php*

```
<?php
require_once __DIR__ . '/lib/Slim/Slim.php';
require_once __DIR__ . '/lib/medoo.min.php';
require_once __DIR__ . '/db_config.include.php
    ';

\Slim\Slim::registerAutoloader();

spl_autoload_register(function ($class) {
  $className = ltrim($class, '\\');
  $fileName  = __DIR__ . '/';
  $namespace = '';
  if ($lastNsPos = strripos($className, '\\'))
      {
    $namespace = substr($className, 0,
        $lastNsPos);
    $className = substr($className, $lastNsPos
        + 1);
    $fileName  .= str_replace('\\',
        DIRECTORY_SEPARATOR, $namespace) .
        DIRECTORY_SEPARATOR;
  }
  $fileName .= str_replace('_',
      DIRECTORY_SEPARATOR, $className) . '.
      class.php';

  if (file_exists($fileName)) {
    require $fileName;
  }
});
```

*scripts/.htaccess*

```
Deny from all
```

*scripts/add-geojson-psgc.py*

```
#!/usr/bin/env python
import sys
import logging
import pprint
import argparse
import csv
from os.path import commonprefix
import json
import unicodedata
import re

KEYS = [
  ["REGION"],
  ["PROVINCE"],
  ["NAME_2", "VARNAME_2"],
  ["NAME_3", "VARNAME_3"]
]

logging.basicConfig()
logger = logging.getLogger("agp")

def main():
  logger.setLevel(logging.DEBUG)

  parser = argparse.ArgumentParser(description
      ="Add PSGC to GeoJSON features.")
  parser.add_argument("-p", "--psgc", required
      =True, help="input csv file containing
      PSGC reference")
  parser.add_argument("-g", "--geojson",
      required=True, help="input GeoJSON file
      ")
  parser.add_argument("-o", "--output",
      required=True, help="output GeoJSON
      file")

  params = parser.parse_args()

  cat_map = {}
  name_map = [{} for _ in range(4)]
  psgc_uniq_map = [{} for _ in range(4)]
  hier = {"name":"root","area":"000000000","
      children":{}}
  jump = {}
  with (sys.stdin if params.psgc == "-" else
      open(params.psgc, "r")) as in_psgc:
    chain = [hier]
    last_area = None
    last_name = None
    last_level = -1
    level = 0
    for row in csv.reader(in_psgc):
      area = None
      name = ""
      for cell in (unicode(x, "utf-8") for x
          in row):
        if len(cell) == 9:
          try:
            int(cell)
            area = cell
            continue
          except ValueError:
            pass
        if len(cell) > len(name):
          name = cell
      if area:
        last_level = level
        if area[2:4] == "00":
          level = 0 # Reg
        elif area[4:6] == "00":
          level = 1 # Prov
        elif area[6:9] == "000":
          level = 2 # Mun
        else:
          level = 3 # Bar

        comlen = [0,2,4,6][level]
        if last_area and area[:comlen] !=
            last_area[:comlen]:
          chain.pop()
          last_level = -1

        if level > last_level:
          chain.append(chain[-1]["children"][
              last_name])
        elif level < last_level:
          for _ in range(level,last_level):
            chain.pop()

        names = get_names(name)
        main = names[0]
        for n in names:
          if not n in chain[-1]:
            main = n
            break
        for n in set(names):
          if n in name_map[level]:
            name_map[level][n] = None
          else:
            name_map[level][n] = main

        if main in psgc_uniq_map[level]:
          psgc_uniq_map[level][main] = None
        else:
          psgc_uniq_map[level][main] = area

        cat = ".".join([x["name"] for x in
            chain] + [main])
        cat_map[cat] = area

        children = chain[-1]["children"]
        child = jump[area] = {
          "name": main,
          "area": area,
          "children": {}
        }
        for n in names:
          if n in children and main != n:
            other = children[n]
            if other and other["area"] != area
                and other["name"] != n:
              children[n] = None
            else:
              children[n] = child

        last_area = area
        last_name = main

psgc_uniq_map = [{k:v for k,v in x.iteritems
    () if v} for x in psgc_uniq_map]
name_map = [{k:v for k,v in x.iteritems() if
    v} for x in name_map]

# print get_main_name(u"Daraga", 2, name_map
    )
# pprint.pprint({k:v for k,v in name_map[2].
    iteritems() if k!=v})
# return

geojson = None
with (open(params.geojson, "r")) as
    in_geojson:
  geojson = json.load(in_geojson)

for feature in geojson["features"]:
  props = feature["properties"]

  values = [None] * len(KEYS)
  for i,keys in enumerate(KEYS):
    values[i] = set([x for l in [props[x].
        split("|") for x in keys if x in
        props and props[x]] for x in l])

  level = None
  for i,v in enumerate(values):
    if v:
      level = i

  name = get_main_name(values[level], level,
      name_map)
  cat = ".".join(["root"] + [get_main_name(x
      , i, name_map) for i,x in enumerate(
      values[:level])] + [name])

  if cat in cat_map:
    area = cat_map[cat]
  elif name in psgc_uniq_map[level]:
    area = psgc_uniq_map[level][name]
  else:
    area = None
    chain = []
    p = hier["children"]
    for i in range(0,level+1):
      aname = get_main_name(values[i], i,
          name_map)
      if aname in p and p[aname]:
        print i,aname
        if i == level:
          area = p[aname]["area"]
          break
        p = p[aname]["children"]
      elif aname in psgc_uniq_map[i]:
        p = jump[psgc_uniq_map[i][aname]]["
            children"]
      else:
        input = None
        while input == None:
          print("Can't find " + aname + " ["
              + "RPMB"[i] + "]")
          pprint.pprint(values)
          print("Enter PSGC for " + list(
              values[i])[0] + " [" + ["
              Region","Province","
              Municipality","Barangay"][i]
              + "]:")
          input = raw_input("\t")
          if not input:
            print("None")
```

```
            break
          elif len(input) != 9 or input not
              in jump:
            input = None
            print("Invalid")
            continue
        if not input:
          break
        psgc_uniq_map[i][aname] = input
        if i == level:
          area = input
          break
        p = jump[input]["children"]
      chain.append(aname)

  if area:
    logger.debug(str(area) + " " + name)
    props["PSGC"] = area
  else:
    print("Can't find " + name + " [" +
        RPMB"[level] + "]")
    pprint.pprint(values)

with open(params.output, "w") as output:
  json.dump(geojson, output)

main_name_cache = [{} for _ in range(4)]

def get_main_name(strings, level, name_map):
  if not strings:
    return None
  for s in strings:
    if s in main_name_cache[level]:
      return main_name_cache[level][s]
  names = [x for l in [get_names(s) for s in
      strings] for x in l]
  main = names[0]
  for n in names:
    if n in name_map[level]:
      main = name_map[level][n]
  for s in strings:
    main_name_cache[level][s] = main
  return main

def get_names(s):
  names = filter(None, [normalize(x) for x in
      [s] + filter(None, re_split.split(s))])
  add = []
  if names[0].endswith("city"):
    add.append(names[0])
    names[0] = names[0][:-4]
  for n in names:
    if n.endswith("city"):
      add.append(n[:-4])
  return names + add

re_split = re.compile(r"[()]|\s+-\s+")
re_roman = re.compile(r"\b(?=[CLXVI]+\b)(C)
    {0,3}(XC|XL|L?X{0,3})(IX|IV|V?I{0,3})\b")
re_trash = re.compile(r"[*-._'/\s]+|of|the|
    capital")

def normalize(s):
  s = str(unicodedata.normalize("NFKD",
      unicode(s)).encode("ascii", "ignore"))
  s = re.sub(r"-", r" ", s)
  s = re_roman.sub(roman_to_int, s).lower()
  s = re.sub(r"\bcity\s+of(.+)", r"\1city", s)
  s = re.sub(r"\bbarangay\b", "bgy", s)
  s = re.sub(r"\bpoblacion\b", "pob", s)
  s = re.sub(r"\b(general|heneral|hen)\b", "
      gen", s)
  s = re_trash.sub("", s)
  return s

numeral_map = zip(
  (100, 90, 50, 40, 10, 9, 5, 4, 1),
  ("C", "XC", "L", "XL", "X", "IX", "V", "IV",
      "I")
)

def roman_to_int(match):
  n = match.group(0).upper()
  i = result = 0
  for integer, numeral in numeral_map:
    while n[i:i + len(numeral)] == numeral:
      result += integer
      i += len(numeral)
  return str(result)

main()
```

*scripts/autogen_family.php*

```
<?php
require '../php/init.include.php';

use \Dynavis\Database;
use \Dynavis\DataProcessor;
use \Dynavis\Model\Official;
use \Dynavis\Model\Family;

$officials = Official::list_items(0, 0);
Database::get()->pdo->beginTransaction();
try {
  $total = $officials["total"];
  $i = 0;
  foreach ($officials["data"] as $o) {
    print "$i/$total\n"; $i++;
    $official = new Official((int) $o["id"],
        false);
    try {
      $family = Family::get_by_name($official
          ->surname);
    }catch(\Dynavis\Core\NotFoundException $e)
        {
      $family = new Family();
      $family->name = $official->surname;
      $family->save();
    }
    $family->add_member($official);
  }

  // Delete single-member families
  Database::get()->query(
    "DELETE FROM family
    WHERE
      id IN (SELECT
        T.family_id
      FROM
        (SELECT
          family_id
        FROM
          family_membership
        INNER JOIN family ON id = family_id
        GROUP BY id
        HAVING COUNT(*) = 1) T)"
  );
}catch(Exception $e) {
  Database::get()->pdo->rollBack();
  throw $e;
}
Database::get()->pdo->commit();
```

*scripts/generate-psgc-mapping.py*

```
#!/usr/bin/env python
import sys
import logging
import argparse
import csv
import json
import unicodedata
import re

logging.basicConfig()
logger = logging.getLogger("gpm")

def main():
  logger.setLevel(logging.DEBUG)

  parser = argparse.ArgumentParser(description
      ="Generate PSGC mapping.")
  parser.add_argument("-i", "--input", help="
      input csv file containing PSGC
      reference")
  parser.add_argument("-o", "--output", help="
      output file (json default)")
  parser.add_argument("--php", action="
      store_true", help="output in PHP array
      format")

  params = parser.parse_args()

  area_map = {}
  with (open(params.input, "r") if params.
      input else sys.stdin) as infile:
    for row in csv.reader(infile):
      area = None
      name = ""
      for cell in (unicode(x, "utf-8") for x
          in row):
        if len(cell) == 9:
          try:
```

```
            int(cell)
            area = cell
            continue
          except ValueError:
            pass
        if len(cell) > len(name):
          name = cell
      if area:
        names = get_names(name)
        for n in set(names):
          if not n in area_map:
            area_map[n] = []
          area_map[n].append(area)

  with (open(params.output, "w") if params.
      output else sys.stdout) as outfile:
    if params.php:
      outfile.write("<?php\n$PSGC_MAP=["+ ",".
          join(
          ['"' + k + '"=>[' + ",".join(
            [str(int(a)) for a in v]
          ) + ']' for k,v in area_map.
              iteritems()]
        ) + "];\n?>")
    else:
      json.dump(area_map, outfile)

def get_names(s):
  names = filter(None, [normalize(x) for x in
      [s] + filter(None, re_split.split(s))])
  add = []
  if names[0].endswith("city"):
    add.append(names[0])
    names[0] = names[0][:-4]
  for n in names:
    if n.endswith("city"):
      add.append(n[:-4])
  return names + add

re_split = re.compile(r"[()]|\s+-\s+")
re_roman = re.compile(r"\b(?=[CLXVI]+\b)(C)
    {0,3}(XC|XL|L?X{0,3})(IX|IV|V?I{0,3})\b")
re_trash = re.compile(r"[*-._'/\s]+|of|the")

def normalize(s):
  s = str(unicodedata.normalize("NFKD",
      unicode(s)).encode("ascii", "ignore"))
  s = re.sub(r"_", r" ", s)
  s = re_roman.sub(roman_to_int, s).lower()
  s = re.sub(r"\bcity\s+of(.+)", r"\1city", s)
  s = re.sub(r"\bbarangay\b", "bgy", s)
  s = re.sub(r"\bpoblacion\b", "pob", s)
  s = re.sub(r"\b(general|heneral|hen)\b", "
      gen", s)
  s = re_trash.sub("", s)
  return s

numeral_map = zip(
  (100, 90, 50, 40, 10, 9, 5, 4, 1),
  ("C", "XC", "L", "XL", "X", "IX", "V", "IV",
      "I")
)

def roman_to_int(match):
  n = match.group(0).upper()
  i = result = 0
  for integer, numeral in numeral_map:
    while n[i:i + len(numeral)] == numeral:
      result += integer
      i += len(numeral)
  return str(result)

main()
```

*scripts/process_geojson.py*

```
#!/usr/bin/env python
import sys
import os
import shutil
import errno
import logging
import collections
import geojson
import math

DATA_DIR = "./data/"
LOG_FILE = DATA_DIR + "log"

try:
  file_handle = os.open(LOG_FILE, os.O_CREAT |
      os.O_EXCL | os.O_WRONLY)
except OSError as e:
  if e.errno == errno.EEXIST:
    pass
  else:
    raise

logging.basicConfig(filename=LOG_FILE)
logger = logging.getLogger("process_geojson")
logger.addHandler(logging.StreamHandler())
logger.setLevel(logging.DEBUG)

def main():
  filepath = sys.argv[1]
  level = sys.argv[2]

  zoom = { # These zoom levels must match the
      server
    "region": 0,
    "province": 8,
    "municipality": 10,
    "barangay": 12,
  }[level]

  tiles = collections.defaultdict(list)

  # Read GeoJson file
  with open(filepath, "r") as infile:
    gj = geojson.loads(infile.read())

    # For each feature, put into appropriate
        tile
    for feature in gj.features:
      bounds = (sys.maxint,sys.maxint, -sys.
          maxint-1,-sys.maxint-1) # min x,y;
          max x,y
      for tile in [deg2num(lat,lon,zoom) for
          lon,lat in geojson.utils.coords(
          feature)]:
        x = tile[0]
        y = tile[1]
        bounds = (
          min(bounds[0], tile[0]),
          min(bounds[1], tile[1]),
          max(bounds[2], tile[0]),
          max(bounds[3], tile[1])
        )
      center = ((bounds[0]+bounds[2])/2,(
          bounds[1]+bounds[3])/2)
      key = (int(center[0]), int(center[1]))
      tiles[key].append(feature)

    # Save tiles
    root = os.path.join(DATA_DIR, level)
    if(os.path.exists(root)):
      shutil.rmtree(root)
    for tile,features in tiles.iteritems():
      fc = geojson.FeatureCollection(features)

      dirpath = os.path.join(root, str(tile
          [0]))
      filepath = os.path.join(dirpath, str(
          tile[1]) + ".json")

      mkdir_p(dirpath, 0775)
      with os.fdopen(os.open(filepath, os.
          O_CREAT | os.O_EXCL | os.O_WRONLY,
          0664), "w") as outfile:
        outfile.write(geojson.dumps(fc))

# http://wiki.openstreetmap.org/wiki/
    Slippy_map_tilenames#Python
def deg2num(lat_deg, lon_deg, zoom):
  lat_rad = math.radians(lat_deg)
  n = 2.0 ** zoom
  xtile = int((lon_deg + 180.0) / 360.0 * n)
  ytile = int((1.0 - math.log(math.tan(lat_rad
      ) + (1 / math.cos(lat_rad))) / math.pi)
       / 2.0 * n)
  return (xtile, ytile)

# http://stackoverflow.com/a/600612
def mkdir_p(path, mode):
  try:
    os.makedirs(path, mode)
  except OSError as exc: # Python >2.5
    if exc.errno == errno.EEXIST and os.path.
        isdir(path):
      pass
    else: raise

main()
```

*scripts/scrape-elections.py*

```
#!/usr/bin/env python
import logging
import argparse
import requests
from bs4 import BeautifulSoup
import sys
import csv

email = "None"

logging.basicConfig()
logger = logging.getLogger("scraper")

def main():
  logger.setLevel(logging.DEBUG)

  parser = argparse.ArgumentParser(description
      ="Scrape the COMELEC election results
      .")
  parser.add_argument("--congress", action="
      store_true", help="find congressional
      elections")

  parser.add_argument("year", metavar="YEAR",
      type=int, help="year of election")

  input_group = parser.
      add_mutually_exclusive_group()
  input_group.add_argument("-c", "--code",
      type=int, help="PSGC of the area")
  input_group.add_argument("-i", "--input",
      help="input csv file containing PSGC")

  parser.add_argument("-o", "--output", help="
      output csv file")

  params = parser.parse_args()

  with open("scrape-elections.conf", "r") as f
      :
    for line in f:
      key,value = line.strip().split("=", 1)
      if key.strip() == "email":
        global email
        email = value.strip()
        logger.debug("email " + email)

  with (open(params.output, "w") if params.
      output else sys.stdout) as outfile:
    if params.input:
      with (sys.stdin if params.input == "-"
          else open(params.input, "r")) as
          infile:
        line = 0
        for row in csv.reader(infile):
          line += 1
          logger.debug("file line " + str(line
              ))
          area = None
          name = ""
          for cell in (unicode(x, "utf-8") for
              x in row):
            if len(cell) == 9:
              try:
                int(cell)
                area = cell
                continue
              except ValueError:
                pass
            if len(cell) > len(name):
              name = cell
          if area and area[2:4] != "00":
            logger.debug("scrape " + area + "
                " + name)
            scrape(params.year, area, False,
                outfile)
            if area[4:] == "00000":
              scrape(params.year, area, True,
                  outfile)

    else:
      scrape(params.year, params.code, params.
          congress, outfile)

def scrape(year, area, congressional, outfile)
    :
  sid = str(year) + " " + str(area) + (" [C]"
      if congressional else "");

  url, data = get_request_props(year, area,
      congressional)
  if not url or not data:
    logger.error("[" + sid + "] Invalid
        parameters")
    sys.exit(1)

  try:
    headers = {
      "User-Agent": "scraperbot/0 (" + email +
          ")",
      "From": email
    }
    request = requests.post(url, data=data,
        headers=headers)
  except requests.ConnectionError as e:
    logger.error("[" + sid + "] Error
        requesting")
    logger.exception(e)
    return False

  try:
    elections = parse_text(request.text)
  except Exception as e:
    logger.error("[" + sid + "] Error parsing
        ")
    logger.exception(e)
    logger.debug(request.text)
    return False
  if elections is None:
    logger.warning("[" + sid + "] No results")
    return False
  elif not elections:
    logger.error("[" + sid + "] Malformed text
        ")
    logger.debug(request.text)
    return False

  area_str = str(area).zfill(9)
  elections = [(area_str, year) + tuple([s.
      encode("utf-8") for s in e]) for e in
      elections]
  csv.writer(outfile).writerows(elections)
  return True

def get_request_props(year, area,
    congressional=False):
  year = int(year)
  area_string = str(area).zfill(9)
  region_code = area_string[0:2]
  province_code = area_string[2:4]
  municipality_code = area_string[4:6]
  barangay_code = area_string[6:9]
  municipality_id = province_code +
      municipality_code
  barangay_id = municipality_id +
      barangay_code

  data = {}

  if province_code == "00": # Region
    url = None
    data["region"] = region_code
  elif municipality_code == "00": # Province
    category = "cong" if congressional else "
        prov";
    if year <= 2007:
      url = "http://www.comelec.gov.ph/tpl/
          ResultsScripts/" + str(year)[-2:] +
          "search" + category + ".php"
    elif year == 2010:
      url = "http://www.comelec.gov.ph/tpl/
          ResultsScripts/search" + category +
          ".php"
    else:
      url = "http://www.comelec.gov.ph/tpl/
          ResultsScripts/search" + category +
          str(year) + ".php"
    data["province"] = province_code
    if congressional:
      data["hidden_cong"] = 1
    else:
      data["hidden_prov"] = 1
  elif barangay_code == "000": # Municipality
    if year <= 2007:
      url = "http://www.comelec.gov.ph/tpl/
          ResultsScripts/" + str(year)[-2:] +
          "searchcity.php"
    elif year == 2010:
      url = "http://www.comelec.gov.ph/tpl/
          ResultsScripts/searchcity.php"
    else:
      url = "http://www.comelec.gov.ph/tpl/
          ResultsScripts/searchcity" + str(
          year) + ".php"
    data["province"] = province_code
    data["municipality"] = municipality_code
    data["hidden_prov"] = 1
  else: # Barangay
    if year == 2010:
      url = "http://www.comelec.gov.ph/tpl/
          ResultsScripts/searchbarangay.php"
      data["hidden_bar"] = 1
    else:
      url = "http://www.comelec.gov.ph/tpl/
          ResultsScripts/" + str(year) + "
          BskeSearch.php"
      data["hidden_prov"] = 1
```

```
    data["region"] = region_code
    data["province"] = province_code
    data["municipality"] = municipality_id
    data["barangay"] = barangay_id

  return (url, data)

def parse_text(text):
  soup = BeautifulSoup(text)
  rows = soup.find_all("tr")
  if not rows:
    if len(soup.find_all("p")) == 1:
      return None
    else:
      return []
  headers = [th.get_text() for th in rows[0].
      find_all("th")]
  values = {}
  elections = []
  for row in rows[1:]:
    cells = row.find_all("td")
    for i in range(len(cells)):
      text = cells[i].get_text().strip()
      if text or i > 0:
        values[headers[i]] = text
      if "colspan" in cells[i].attrs:
        i += int(cells[i].attrs["colspan"])
    if len(cells) == len(headers):
      # elect tuple: (area, year, position,
          surname, name, nickname, party,
          votes)
      surname, name = [x.replace(r"/[^\P{P
          }-]+/", "").strip() for x in values
          ["NAME"].split(",",1)]
      position = values["POSITION"] if "
          POSITION" in values else ""
      nickname = values["NICKNAME"] if "
          NICKNAME" in values else ""
      party = values["PARTY AFFILIATION"] if "
          PARTY AFFILIATION" in values else
          ""
      votes = values["VOTES OBTAINED"] if "
          VOTES OBTAINED" in values else ""
      elections.append((position, surname,
          name, nickname, party, votes))
  return elections

main()
```

*scripts/scrape-elections.sample.conf*

```
email = example@example.com
```

*scss/_checkbox.scss*

```
// Adapted from Materialize CSS
// https://github.com/Dogfalo/materialize/blob
    /43d43c27960ba7fd6eff16b4daadcd7e2dba6294
    /sass/components/_form.scss
/***************
   Checkboxes
***************/

/* CUSTOM CSS CHECKBOXES */
form p {
  margin-bottom: 10px;
  text-align: left;
}
form p:last-child {
  margin-bottom: 0;
}

/* Remove default checkbox */
[type="checkbox"]:not(:checked),
[type="checkbox"]:checked {
  position: absolute;
  left: -9999px;
  visibility: hidden;
}


// Checkbox Styles
[type="checkbox"]:not(.toggle-checkbox) {

  // Text Label Style
  + * {
    position: relative;
    padding-left: 28px;
    display: inline-block;

    -webkit-user-select: none; /* webkit (
        safari, chrome) browsers */
    -moz-user-select: none; /* mozilla
        browsers */
    -khtml-user-select: none; /* webkit (
        konqueror) browsers */
    -ms-user-select: none; /* IE10+ */
  }

  &:not(:disabled) {
    cursor: pointer;
  }
  &:disabled + * {
    opacity: 0.6;
  }

  /* checkbox aspect */
  + *:before {
    content: '';
    position: absolute;
    top: 0;
    left: 0;
    width: 18px;
    height: 18px;
    z-index: 0;
    border: 2px solid $color-primary-2;
    border-radius: 1px;
    margin-top: 2px;
  }

  &:not(:checked):disabled + *:before {
    border: none;
    background-color: $color-primary-2;
  }

  &:checked {
    + *:before {
      top: -4px;
      left: -3px;
      width: 12px; height: 22px;
      border-top: 2px solid transparent;
      border-left: 2px solid transparent;
      border-right: 2px solid $color-primary
          -1;
      border-bottom: 2px solid $color-primary
          -1;
      transform: rotate(40deg);
      transform-origin: 100% 100%;
    }

    &:disabled + *:before {
      border-right: 2px solid white;
      border-bottom: 2px solid white;
    }

  }

  /* Indeterminate checkbox */
  &:indeterminate {
    +label:before {
      left: -10px;
      top: -11px;
      width: 10px; height: 22px;
      border-top: none;
      border-left: none;
      border-right: 2px solid $color-primary
          -1;
      border-bottom: none;
      transform: rotate(90deg);
      transform-origin: 100% 100%;
    }

    // Disabled indeterminate
    &:disabled + *:before {
      border-right: 2px solid desaturate(
          $color-primary-1, 100%);
      background-color: transparent;
    }
  }

  // General
  + *:after {
    border-radius: 2px;
  }
  + *:before,
  + *:after {
    content: '';
    left: 0;
    position: absolute;
    /* .1s delay is for check animation */
    transition: border .25s, background-color
        .25s, width .20s .1s, height .20s .1s
        , top .20s .1s, left .20s .1s;
    z-index: 1;
  }
```

```
  // Unchecked style
  &:not(:checked) + *:before {
    width: 0;
     height: 0;
     border: 3px solid transparent;
     left: 6px;
     top: 10px;

     -webkit-transform: rotateZ(37deg);
             transform: rotateZ(37deg);
     -webkit-transform-origin: 20% 40%;
             transform-origin: 100% 100%;
  }
  &:not(:checked) + *:after {
    height: 20px;
    width: 20px;
    background-color: transparent;
    border: 2px solid desaturate($color-
        primary-1, 100%);
    top: 0px;
    z-index: 0;
  }
  // Checked style
  &:checked {
    + *:before {
      top: 0;
      left: 1px;
      width: 8px;
      height: 13px;
      border-top: 2px solid transparent;
      border-left: 2px solid transparent;
      border-right: 2px solid white;
      border-bottom: 2px solid white;
      -webkit-transform: rotateZ(37deg);
      transform: rotateZ(37deg);
      -webkit-transform-origin: 100% 100%;
      transform-origin: 100% 100%;
    }
    + *:after {
      top: 0px;
      width: 20px;
      height: 20px;
      border: 2px solid $color-primary-1;
      background-color: $color-primary-1;
      z-index: 0;
    }
  }
  // Disabled style
  &:disabled:not(:checked) + *:before {

    background-color: transparent;
    border: 2px solid transparent;
  }
  &:disabled:not(:checked) + *:after {
    border-color: transparent;
    background-color: mix(desaturate($color-
        primary-1,100%), white, 40%);
  }
  &:disabled:checked + *:before {
    background-color: transparent;

  }
  &:disabled:checked + *:after {
    background-color: mix(desaturate($color-
        primary-1,100%), white, 40%);
    border-color: mix(desaturate($color-
        primary-1,100%), white, 40%);
  }

}
```

*scss/_common.scss*

```
@import "fonts";
@import "reset";

$header-height: 75px;
$sidebar-width: 200px;

$color-text: #2B2A28;
$color-background: #F1F3F3;

$color-text-alt: #BFBEB4;
$color-background-alt: #222526;

$color-primary-1: #00BCD4;
$color-primary-2: #B2EBF2;
$color-complement: #FF4081;
$color-text-primary-1: #ffffff;
$color-text-primary-2: #212121;
$color-text-complement: #ffffff;

$ease-swift: cubic-bezier(0.2, 0, 0, 1);

@import "typeahead";
@import "drop-shadow";

* { box-sizing: border-box }

html,body { margin: 0; }
.layout-row, .layout-col { overflow: hidden;
    position: absolute; }
.layout-row { left: 0; right: 0; }
.layout-col { top: 0; bottom: 0; }
.scroll-x { overflow-x: auto; }
.scroll-y { overflow-y: auto; }

$sidebar-width: 250px;

.header {
  height: $header-height;
  z-index: 200;
}
.sidebar {
  width: $sidebar-width;
  top: $header-height;
  z-index: 100;
}
.body {
  top: $header-height;
  left: $sidebar-width;
  z-index: 0;
}

.clearfix:after {
  content: " ";
  visibility: hidden;
  display: block;
  height: 0;
  clear: both;
}
```

```
html,body {
  color: $color-text;
  background-color: $color-background;
  font-family: "Roboto Condensed", sans-serif;
  font-size: 0.94em;
}
a {
  text-decoration: none;
  color: $color-primary-1;
  &:link {
    color: $color-primary-1;
  }
  &:visited {
    color: darken($color-primary-1, 10%);
  }
  &:hover {
    text-decoration: underline;
    color: lighten($color-primary-1, 20%);
  }
  &:active {
    color: darken($color-primary-1, 10%);
  }
  &.no-decor, &.no-decor:hover {
    text-decoration: none;
  }
}

h1,h2 {
  line-height: 180%;
  font-size: 160%;
  font-weight: lighter;
}
h3,h4 {
  line-height: 180%;
  font-size: 140%;
}
h5,h6 {
  line-height: 180%;
  font-size: 120%;
}

pre {
  white-space: pre-wrap;
  word-wrap: break-word;
}

.text {
  line-height: 1.6em;
  text-align: justify;
  font-family: "Roboto", sans-serif;
  fonr-size: 95%;
}

.text-large {
  font-size: 120%;
  font-weight: bold;
}
```

```
.text-small {
  opacity: 0.8;
  font-size: 80%;
}

.text-center {
  text-align: center;
}

.pull-left {
  float: left !important;
}
.pull-right {
  float: right !important;
}

.one-line {
  overflow: hidden;
  white-space: nowrap;
  text-overflow: ellipsis;
}

.hidden {
  display: none;
}

.transparent {
  opacity: 0.5;
}

.inline {
  display: inline-block;
}

.pad {
  padding: 0.5em;
}
.mar {
  margin: 0.5em;
}

.fade-enter {
  top: -10px;
  opacity: 0.01;
  &.fade-enter-active {
    top: 0;
    opacity: 1;
    transition: top 0.2s ease-out, opacity 0.3
        s ease;
  }
}
.fade-leave {
  opacity: 1;
  &.fade-leave-active {
    opacity: 0.01;
    transition: opacity 0.2s ease;
  }
}

.fade-in {
  animation: fade-in 0.2s ease;
}

@keyframes fade-in {
  from {
    opacity: 0;
  }
  to {
    opacity: 1;
  }
}

i.fa-spin {
  font-size: 14px;
  width: 14px;
  height: 14px;
}

@import "components";
```

*scss/_components.scss*

```
@import "spinner";
@import "checkbox";

.header {
  color: $color-text-primary-1;
  background-color: $color-primary-1;

  box-shadow: 0 0 8px 8px rgba(0,0,0,0.1),
    0 2px 4px 0 rgba(0,0,0,0.4);

  & > div {
    height: inherit;
  }

  #header-logo {
    float: left;
    display: inline-block;
    width: $sidebar-width;
    height: $header-height;

    background-color: #00ACC1;

    a {
      display: inline-block;
      width: inherit;
      height: inherit;
    }

    .logo {
      float: left;
      display: inline-block;
    }
    .logo-type {
      float: left;
      display: inline;
      width: $sidebar-width - 90px;
      line-height: $header-height;
      vertical-align: baseline;
    }
  }

  #header-content {
    display: inline-block;
    padding: 0 40px;
    font-family: "Roboto Slab", serif;
    font-weight: lighter;

    *:last-child {
      font-weight: normal;
    }
    .header-title {
      display: inline;
      line-height: $header-height;
      vertical-align: baseline;
    }
    .header-subtitle {
      display: inline;
      margin-left: 0.5em;
      line-height: $header-height;
      vertical-align: baseline;
      font-size: 100%;
    }
  }

  .session {
    float: right;
    display: inline-block;
    padding: 0 1.4em;

    input[type=text], input[type=password] {
      width: 8em;
    }

    .login-text {
      display: inline-block;
      line-height: $header-height;
      vertical-align: baseline;
    }
    .login-username {
      font-weight: bold;
    }

    button {
      margin: 0;
    }

    .button-flat {
      color: $color-text-primary-1 !important;
    }
  }
}

.logo {
  width: 90px;
  height: 75px;
  background-image: url("../img/logo-small.png
      ");
  background-position: center;
  background-repeat: no-repeat;
}
.logo-large {
  width: 300px;
  height: 130px;
  background-image: url("../img/logo-large.png
      ");
  background-position: center;
  background-repeat: no-repeat;
}
```

```
.logo-type {
  text-align: center;
  font-family: "Roboto Slab", serif;
  font-size: 14pt;
  font-weight: bold;
  color: white;
}

.hider {
  white-space: nowrap;
  .hider-handle {
    float: right;
    display: inline-block;
    height: $header-height;
    text-align: center;
  }
  .hider-handle-toggle {
    max-width: $header-height * 2;
    overflow: hidden;
    text-overflow: ellipsis;
    transition: max-width 0.4s ease-in;
  }
  &:hover .hider-handle-toggle {
    max-width: 0;
    transition: max-width 0.4s ease-out;
  }

  .hider-content {
    float: left;
    display: inline-block;
    overflow: hidden;
    max-width: 0;
    padding: 0;
    transition: max-width 0.6s ease-out,
        padding 0.6s ease-out;
  }
  &:hover .hider-content {
    max-width: 24em;
    padding: 0 8px;
    transition: max-width 0.4s ease-in;
  }
}

.body-panel {
  padding: 40px 12%;
}

.pane {
  .pane-header {
    color: $color-text;
    background-color: $color-background;
    padding: 6px 20px;
  }
  .pane-content {
    padding: 10px 20px;
  }
}

.panel-toolbar {
  .panel-toolbar-toggle {
    overflow: hidden;
    max-height: 0;
    transition: max-height 0.2s $ease-swift;
  }
  .panel-toolbar-contents {
    overflow: hidden;
    max-height: 16em;
    margin: 0 0 2em;
    transition: max-height 0.4s $ease-swift,
        margin 0.4s $ease-swift;

    &>.pure-g>div {
      padding: 30px;
      border-right: solid 1px rgba(0,0,0,0.1);
      &:last-child {
        border-right: none;
      }
    }
    h6 {
      opacity: 0.8;
    }
  }

  &.panel-toolbar-closed {
    .panel-toolbar-toggle {
      max-height: 3em;
    }
    .panel-toolbar-contents {
      max-height: 0;
      margin: 0;
    }
  }
}

.form input:not([type=file]), .form select, .
    form textarea, .input, .button {
  position: relative;
  top: 0;
  min-height: 2rem;
  margin-top: 4px;
  margin-bottom: 4px;
  padding: 0.5em 0.7em;
  border: none;
  border-radius: 2px;

  font-family: "Roboto", sans-serif;
  color: $color-text;
  background-color: white;

  &.validation-error, .validation-error & {
    border: solid 1px red;
  }

  @extend %floating-transitions;
  &:not(.button-flat):not([disabled]):not([
      readonly]) {
    top: -1px;
    @include drop-shadow(2);
    &:hover {
      top: -2px;
      @include drop-shadow(4);
    }
  }
}
.form input:not([type=button]):not([type=
    password]):not([type=submit]), .form
    select, .form textarea, .input {
  &[disabled] {
    color: rgba($color-text, 0.6);
  }
  &[readonly] {
    color: rgba($color-text, 0.7);
  }
}
form input[type=button], form input[type=
    submit], form input[type=reset], .button
    {
  cursor: pointer;
  padding: 0.8em 1.2em;
  font-family: "Roboto", sans-serif;
  font-size: 80%;
  font-weight: bold;
  text-transform: uppercase;
  text-overflow: ellipsis;

  &[disabled],&[readonly] {
    opacity: 0.4;
    background-color: transparent;
  }
  &:active {
    top: -1px;
    @include drop-shadow(2);
  }
  &.button-primary:not(x) {
    color: $color-text-primary-1;
    background-color: $color-primary-1;
  }
  &.button-complement:not(x) {
    color: $color-text-complement;
    background-color: $color-complement;
  }
  &:active:not([disabled]):not([readonly]) {
    top: -1.5px;
    @include drop-shadow(3);
  }
  &.button-flat:not(x) {
    color: $color-primary-1;
    background-color: transparent;
    &.button-complement {
      color: $color-complement;
    }
    @include drop-shadow(0);
    &:not([disabled]):hover {
      top: -1px;
      @include drop-shadow(2);
    }
  }
}
.button-close {
  opacity: 0.6;
  font-size: 140%;
  font-weight: bold;
  width: 1.5em;
  height: 1.5em;
  padding: 0;
  border-radius: 50%;

  &:hover {
    opacity: 1;
  }
}
textarea {
  width: 100%;
  height: 6em;
}

.file-wrapper {
```

```
    display: inline-block;
    cursor: pointer;
    margin: 4px 0;

    .input, .name {
      width: 6em;
    }
    .name {
      overflow: hidden;
      white-space: nowrap;
      text-align: left !important;
    }
    input[type=file] {
      display: none;
    }
  }

  .search-bar {
    @extend .input;
    input, button {
      border: none;
      background: none;
    }
    input {
      margin: -0.5em -0.7em;
      padding: 0.5em 0.7em;
    }
    button {
      margin-left: 1.4em;
    }
  }

  .data-table-header {
    @extend .data-row;
    font-weight: bold;
    color: lighten($color-text, 60%);
  }

  .data-row {
    position: relative;
    display: block;

    padding: 1em 22px;
    margin: 1px 8px 0;

    border-left: solid 0 $color-complement;
    background-color: white;

    @extend %floating-transitions;
    transition: margin-left 0.3s $ease-swift,
      margin-right 0.3s $ease-swift,
      padding 0.4s $ease-swift,
      margin-top 0.5s $ease-swift,
      margin-bottom 0.5s $ease-swift,
      border-left 0.4s $ease-swift,
      box-shadow 0.4s $ease-swift;
    @include drop-shadow(1);

    .button {
      background-color: $color-background;
    }

    .text {
      color: #606060;
    }

    &.clicky {
      cursor: pointer;
      text-align: left;
      border-right: none;
      border-top: none;
      border-bottom: none;
      &:hover {
        color: $color-text-primary-2;
        background-color: $color-primary-2;
      }
    }

    &.edit {
      padding: 28px;
      margin: 17px 0;
      border-radius: 2px;
      border-left: 4px solid $color-complement;

      @include drop-shadow(5);
    }
  }

  .field-group {
    line-height: 150%;
    margin: 0.6em 0;
  }
  .label {
    font-size: 90%;
    padding-right: 0.3em;
    opacity: 0.4;
  }
  .field {
    &.text {
      opacity: 0.9;
    }
    &.text-large {
      font-size: 120%;
    }
  }

  .token-list-input {
    @extend .clearfix;
    padding: 0 0.5em;
    width: auto;
    cursor: text;

    .token-input {
      display: inline-block;
      width: auto;
      min-width: 6em;

      .token-input-typeahead {
        padding: 0.5em;
      }
    }
  }
  .token-submit {
    display: none;
  }
  .token {
    float: left;
    display: inline-block;
    position: relative;
    top: 0;

    margin: 0.25em 3px;
    height: 1.75em;
    border-radius: 0.875em;
    font-size: 90%;

    color: $color-text-primary-2;
    background-color: $color-primary-2;

    @extend %floating-transitions;

    &:hover {
      top: -0.5px;
      @include drop-shadow(1);
    }

    span {
      padding: 0.3em 8px;
      padding-right: 0;
    }
    button {
      margin: 0;
      padding: 0.3em 8px;
      font-size: 100%;
      border: none;
      background: none;
    }
  }

  .full-button {
    background: none;
    border: none;
    width: 100%;
    height: 100%;
    text-align: inherit;
  }

  .group {
    @extend .clearfix;
    &:not(.group-no-table) {
      display: table;
      .group-component { display: table-cell; }
    }

    .group-component {
      height: 100%;
      margin-left: 0;
      margin-right: 0;
      vertical-align: middle;
      border-radius: 0;
    }
    .group-component:first-child {
      border-top-left-radius: 2px;
      border-bottom-left-radius: 2px;
    }
    .group-component:last-child {
      float: right;
      border-top-right-radius: 2px;
      border-bottom-right-radius: 2px;
    }
  }

  /* The following toggle styles are adpated
       from Materialize (materializecss.com)
   * License: The MIT License (MIT)
   * Copyright (c) 2014-2015 Materialize
   */
  .toggle label {
```

```
  color: lighten($color-text, 40%);
  font-size: 80%;
  padding: 8px;

  .toggle-lever {
    content: "";
    display: inline-block;
    position: relative;
    width: 35px;
    height: 15px;
    background-color: darken($color-background
        , 10%);
    border-radius: 15px;
    transition: background 0.3s ease-out;
    vertical-align: middle;

    &:after {
      content: "";
      position: absolute;
      display: inline-block;
      width: 21px;
      height: 21px;
      background-color: $color-background;
      border-radius: 21px;
      left: -5px;
      top: -3px;
      transition: left 0.3s $ease-swift,
          background 0.3s ease-out;
    }
  }
  input[type=checkbox] {
    opacity: 0;
    width: 0;
    height: 0;

    &:checked + .toggle-lever:after {
      left: 19px;
    }
    &:not(:disabled) {
      & + .toggle-lever {
        cursor: pointer;
      }
      & + .toggle-lever:after {
        cursor: pointer;
        @include drop-shadow(1);
      }
      &:checked + .toggle-lever {
        background-color: $color-primary-2;
      }
      &:checked + .toggle-lever:after {
        background-color: $color-primary-1;
      }
    }
    &:disabled {
      & + .toggle-lever {
        background-color: mix(desaturate(
            $color-primary-2, 60%), white,
            40%);
      }
      & + .toggle-lever:after {
        background-color: mix(desaturate(
            $color-primary-1, 60%), white,
            40%);
      }
    }
  }
}

.typeahead-nostyle {
  background: none;
  border: none;
  outline: none;
}

.modal-content {
  border-radius: 3px;
}

#notifications-container {
  pointer-events: none;

  z-index: 10;
  position: absolute;
  right: 0;
  bottom: 0;
  overflow: hidden;

  padding: 2em;

  .notification {
    pointer-events: auto;
    cursor: pointer;

    display: block;
    width: 300px;
    margin: 1em;
    padding: 1.2em;
    border-radius: 2px;

    line-height: 140%;
    font-size: 110%;

    color: white;
    background-color: rgba($color-background-
        alt, 0.9);
    @include drop-shadow(8);

    transition: color 0.3s linear, background-
        color 0.3s linear;
  }
  .notification:hover {
    opacity: 0.8;
  }

  .notification-error {
    background-color: rgba(darkred, 0.9);
  }
  .notification-success {
  }
}
```

*scss/_drop-shadow.scss*

```
@mixin drop-shadow($depth) {
  box-shadow:
    0 0 ($depth*6px) 0 rgba(0,0,0,0.01), //
        ambient
    0 ($depth*0.4px) ($depth*2px) 0 rgba
        (0,0,0,1 - 14/($depth + 16)),
    0 ($depth*0.5px) ($depth*1px) ($depth*-1px
        ) rgba(0,0,0,0.8); // direct

  z-index: $depth;
}

%floating-transitions {
  transition: all 0.1s ease-out;
  transition-property: box-shadow, top;
}
```

*scss/_fonts.scss*

```
@font-face {
  font-family: "Roboto";
  src: url("../fonts/Roboto/Roboto-Regular.ttf
      ");
  font-weight: 400;
  font-style: normal;
}
@font-face {
  font-family: "Roboto";
  src: url("../fonts/Roboto/Roboto-Bold.ttf");
  font-weight: 700;
  font-style: normal;
}
@font-face {
  font-family: "Roboto";
  src: url("../fonts/Roboto/Roboto-Italic.ttf
      ");
  font-weight: 400;
  font-style: italic;
}
@font-face {
  font-family: "Roboto";
  src: url("../fonts/Roboto/Roboto-BoldItalic.
      ttf");
  font-weight: 700;
  font-style: italic;
}

@font-face {
  font-family: "Roboto Condensed";
  src: url("../fonts/Roboto_Condensed/
      RobotoCondensed-Regular.ttf");
  font-weight: 400;
  font-style: normal;
}
@font-face {
  font-family: "Roboto Condensed";
  src: url("../fonts/Roboto_Condensed/
      RobotoCondensed-Bold.ttf");
  font-weight: 700;
  font-style: normal;
}
@font-face {
  font-family: "Roboto Condensed";
```

```
  src: url("../fonts/Roboto_Condensed/
       RobotoCondensed-Italic.ttf");
  font-weight: 400;
  font-style: italic;
}
@font-face {
  font-family: "Roboto Condensed";
  src: url("../fonts/Roboto_Condensed/
       RobotoCondensed-BoldItalic.ttf");
  font-weight: 700;
  font-style: italic;
}

@font-face {
  font-family: "Roboto Slab";
  src: url("../fonts/Roboto_Slab/RobotoSlab-
       Regular.ttf");
  font-weight: 400;
  font-style: normal;
}
@font-face {
  font-family: "Roboto Slab";
  src: url("../fonts/Roboto_Slab/RobotoSlab-
       Bold.ttf");
  font-weight: 700;
  font-style: normal;
}
@font-face {
  font-family: "Roboto Slab";
  src: url("../fonts/Roboto_Slab/RobotoSlab-
       Light.ttf");
  font-weight: 300;
  font-style: normal;
}

// @import url(http://fonts.googleapis.com/css
     ?family=Roboto+Condensed:400italic,700
     italic,400,700|Roboto+Slab:300,700,400|
     Roboto:400italic,700italic,700,400);
@import "font-awesome.min.css";
```

*scss/_reset.scss*

```
/* http://meyerweb.com/eric/tools/css/reset/
   v2.0 | 20110126
   License: none (public domain)
*/

html, body, div, span, applet, object, iframe,
      h1, h2, h3, h4, h5, h6, p, blockquote,
     pre, a, abbr, acronym, address, big, cite
     , code, del, dfn, em, img, ins, kbd, q, s
     , samp, small, strike, strong, sub, sup,
     tt, var, b, u, i, center, dl, dt, dd, ol,
      ul, li, fieldset, form, label, legend,
     table, caption, tbody, tfoot, thead, tr,
     th, td, article, aside, canvas, details,
     embed, figure, figcaption, footer, header
     , hgroup, menu, nav, output, ruby,
     section, summary, time, mark, audio,
     video {
  margin: 0;
  padding: 0;
  border: 0;
  font-size: 100%;
  font: inherit;
  vertical-align: baseline; }

/* HTML5 display-role reset for older browsers
     */

article, aside, details, figcaption, figure,
     footer, header, hgroup, menu, nav,
     section {
  display: block; }

body {
  line-height: 1; }

ol, ul {
  list-style: none; }

blockquote, q {
  quotes: none; }

blockquote {
  &:before, &:after {
    content: '';
    content: none; } }

q {
  &:before, &:after {
    content: '';
    content: none; } }

table {
  border-collapse: collapse;
  border-spacing: 0; }
```

*scss/_spinner.scss*

```
// http://codepen.io/CSS3fx/pen/EfByo
.spinner-circular {
  position:absolute;
  top:20%;
  left:50%;
  width: 70px;
  height: 70px;
  animation: spinner-rotate 2s linear infinite
       , spinner-fade-in 2s linear;
  background-color: white;
  border-radius: 50%;
  padding: 10px;
  box-shadow: 0 0 8px 0 rgba(0,0,0,0.2);

  opacity: 1;
  transition: opacity 1s linear;
}

.spinner-hide {
  opacity: 0;
}

@keyframs spinner-fade-in {
  from {opacity: 0;}
  to {opacity: 1;}
}

@keyframes spinner-rotate {
  from {transform: rotate(0deg);}
  to {transform: rotate(360deg);}
}


.spinner-path {
  stroke-dasharray: 1,200;
  stroke-dashoffset: 0;
  animation: spinner-dash 1.5s $ease-swift
       infinite,
    spinner-color 6s $ease-swift infinite;
  stroke-linecap: round;
  stroke: #3f88f8;
}

@keyframes spinner-dash {
  0%   {
    stroke-dasharray: 1,200;
    stroke-dashoffset: 0;
  }
  50%  {
    stroke-dasharray: 89,200;
    stroke-dashoffset: -35;
  }
  100% {
    stroke-dasharray: 89,200;
    stroke-dashoffset: -124;
  }
}

@keyframes spinner-color {
  0%   {stroke: $color-primary-1;}
  20%  {stroke: $color-primary-1;}
  25%  {stroke: $color-complement;}
  45%  {stroke: $color-complement;}
  50%  {stroke: $color-primary-1;}
  70%  {stroke: $color-primary-1;}
  75%  {stroke: $color-complement;}
  95%  {stroke: $color-complement;}
}
```

*scss/_typeahead.scss*

```
@import "drop-shadow";
/*
 * typehead.js-bootstrap3.less
 * @version 0.2.3
 * https://github.com/hyspace/typeahead.js-
      bootstrap3.less
 *
 * Licensed under the MIT license:
 * http://www.opensource.org/licenses/MIT
 */
// Modified by Lean Rada
.twitter-typeahead {
  width: 100%;
  float: left;
}
.group .twitter-typeahead {
  width: auto;
}
.twitter-typeahead .tt-hint {
  color: #999999;
}
.twitter-typeahead .tt-input {
  z-index: 2;
}
.twitter-typeahead .tt-input[disabled],
.twitter-typeahead .tt-input[readonly],
fieldset[disabled] .twitter-typeahead .tt-
    input {
  cursor: not-allowed;
  background-color: #eeeeee !important;
}
.tt-dropdown-menu,
.tt-menu {
  position: absolute;
  top: 100%;
  left: 0;
  z-index: 1000;
  min-width: 160px;
  width: 100%;
  padding: 5px 0;
  margin: 2px 0 0;
  list-style: none;
  font-size: 14px;
  background-color: #ffffff;
  border-radius: 2px;
  @include drop-shadow(8);
  background-clip: padding-box;
}
.tt-dropdown-menu .tt-suggestion,
.tt-menu .tt-suggestion {
  display: block;
  padding: 3px 20px;
  clear: both;
  font-weight: normal;
  line-height: 1.42857143;
  color: #333333;
}
.tt-dropdown-menu .tt-suggestion.tt-cursor,
.tt-menu .tt-suggestion.tt-cursor,
.tt-dropdown-menu .tt-suggestion:hover,
.tt-menu .tt-suggestion:hover {
  cursor: pointer;
  text-decoration: none;
  outline: 0;
  background-color: $color-primary-1;
  color: $color-text-primary-1;
}
.tt-dropdown-menu .tt-suggestion.tt-cursor a,
.tt-menu .tt-suggestion.tt-cursor a,
.tt-dropdown-menu .tt-suggestion:hover a,
.tt-menu .tt-suggestion:hover a {
  color: #262626;
}
.tt-dropdown-menu .tt-suggestion p,
.tt-menu .tt-suggestion p {
  margin: 0;
}
```

*scss/admin.scss*

```
@import "common";
$sidebar-width: 175px;

.header {
  #header-logo {
    width: $sidebar-width;
    .logo {
      width: 100%;
    }
    .logo-type {
      display: none;
    }
  }
}
.sidebar {
  width: $sidebar-width;
  color: $color-text-alt;
  background-color: $color-background-alt;
}
.body {
  left: $sidebar-width;
  background-color: $color-background;
}

.sidebar .menu {
  padding: 20px 0;

  & > li {
    width: 100%;
    font-weight: bold;
    border-left: 0px solid $color-background-
        alt;
    transition: border-left 0.3s $ease-swift,
      background-color 0.1s ease-out,
      box-shadow 0.2s $ease-swift;

    &:hover {
      color: $color-text;
      background-color: darken($color-
          background, 20%);
      border-left: 4px solid $color-complement
          ;
      box-shadow: inset -4px 0 2px -2px rgba
          (0,0,0,0.2);
    }
    &.active {
      color: $color-text;
      background-color: $color-background;
      border-left: 4px solid $color-complement
          ;
      box-shadow: inset 0 0 0 0 rgba
          (0,0,0,0.2);
    }
    &.indirect {
      background-color: darken($color-
          background, 10%);
      box-shadow: inset -4px 0 2px -2px rgba
          (0,0,0,0.2);
    }

    a {
      display: block;
      padding: 1.2em 30px;
      color: inherit;
      text-decoration: none;
    }
  }
}

a.tile-container {
  color: $color-text;
}
.tile-container {
  @extend .fade-in;

  padding: 6px;
  text-decoration: none;

  .tile {
    padding: 25px;
    color: $color-text;
    background-color: white;

    transition: all 0.1s ease-out;
    transition-property: opacity, color,
        background-color;
    @include drop-shadow(2);

    h1 {
      margin: 0px 0 30px;
    }
    .field {
      opacity: 0.6;
      margin: 4px 0;
    }
    p {
      margin-top: 0.5em;
    }
  }

  &.highlight .tile {
    color: darken($color-complement, 10%);
    background-color: white;
    @include drop-shadow(3);
    .button {
      transition: inherit;
    }
  }
  &.highlight:hover .tile {
```

```
    color: $color-text-complement;
    background-color: $color-complement;

    .button {
      color: $color-primary-1;
      background-color: white;
    }
  }
  &.faded .tile {
    opacity: 0.4;
  }
  &:hover .tile {
    opacity: 1;
    color: $color-text-primary-1;
    background-color: $color-primary-1;
  }

  &:hover {
    text-decoration: none;
  }
}

.data-row.active, .data-row.inactive {
  transition: background-color 0.6s;
}
.data-row.inactive {
  color: mix($color-background, $color-text,
      60%);
  background-color: mix(white, $color-
      background, 60%);
}

.family-member-item {
  list-style: none;
  padding: 0 0.5em;

  & > span {
    float: left;
    padding: 1em 1ch;
  }
  button {
    visibility: hidden;
  }

  .number {
    text-align: right;
    opacity: 0.2;
  }
  .surname {
    opacity: 0.6;
    font-weight: bold;
  }
  .nickname {
    opacity: 0.4;
    font-size: 90%;
    font-weight: bold;
  }

  &:hover {
    background-color: $color-background;

    button {
      visibility: visible;
    }
  }

  &:not(:first-child) {
    border-top: solid 1px $color-background;
  }
}
```

*scss/app.scss*

```
@import "common";

$infobar-height: 50%;

.header {
}
.sidebar {
  padding: 9px 0;
  background-color: white;
  box-shadow: 0 0 8px 0 rgba(0,0,0,0.4);
}
.body {
  background-color: $color-background;
}
.infobar {
  bottom: 0;
  left: $sidebar-width;
  height: 0;
  z-index: 50;

  background-color: white;
  box-shadow: 0 0 4px 0 rgba(0,0,0,0.4);
  transition: height 0.4s ease-in;
}
.infobar.show {
  height: $infobar-height;
  transition: height 0.6s $ease-swift;
}

.infobar .infobar-title {
  padding: 8px 22px;
  color: $color-text-complement;
  background-color: $color-complement;

  button {
    color: $color-text-complement;
    font-size: 160%;
  }
}
.infobar .infobar-content {
  padding: 12px 22px;
}

.map-panel {
  width: 100%;
  height: 100%;
}

.dataset-chooser-list-container {
  position: relative;
  height: 60vh;
}
.dataset-chooser-list {
  overflow-y: auto;
  height: 100%;
  padding: 5px 0;
}
.dataset-chooser-item {
  cursor: pointer;
  padding: 1.4em;

  &:hover {
    background-color: $color-background;
  }

  &.selected {
    color: $color-text-primary-2;
    background-color: $color-primary-2;

    &:not(:first-child), &+.dataset-chooser-
        item {
      border-color: transparent;
    }
  }

  &:not(:first-child) {
    border-top: solid 1px $color-background;
  }
}
.scroll-edge-fade {
  position: absolute;
  pointer-events: none;
  content: "";
  top: 0;
  bottom: 0;
  left: 0;
  right: 0;
  background: linear-gradient(0deg, white,
      transparent 40px), linear-gradient(180
      deg, white, transparent 20px);
}

.map-polygon {
  transition: all 0.4s ease-out;

  opacity: 0;
  &.visible {
    opacity: 1;
  }
}

.map-tag, .map-tag-stroke {
  text-anchor: middle;
  dominant-baseline: central;
  font-family: "Roboto", sans-serif;
  font-weight: bold;
  fill: black;
  text-rendering: optimizeSpeed;
}
.map-tag-stroke {
  stroke: white;
  stroke-width: 6px;
  stroke-linejoin: round;
}

.histogram-bar {
  fill: #BFBFBF;
  stroke: #BFBFBF;
```

```
  stroke-width: 1px;
}
.histogram-label {
  fill: black;
  font-size: 80%;
  font-weight: bold;
  writing-mode: tb;
  text-anchor: start;
  dominant-baseline: central;
  &:first-child {
    dominant-baseline: ideographic;
  }
  &:last-child {
    dominant-baseline: text-before-edge;
  }
}
.histogram-title {
  fill: black;
  font-weight: lighter;
  text-anchor: middle;
  dominant-baseline: ideographic;
}

.scatterplot-point {
  fill: #ffffff;
}
.scatterplot-point-shadow {
  fill: #000000;
  opacity: 0.3;
  transform: translate(0,0.5px);
}
.scatterplot-label {
  fill: black;
  font-size: 80%;
  font-weight: bold;
  dominant-baseline: central;
  &:first-child {
    dominant-baseline: ideographic;
  }
  &:last-child {
    dominant-baseline: text-before-edge;
  }
}
.scatterplot-label-x {
  writing-mode: tb;
  text-anchor: start;
}
.scatterplot-label-y {
  text-anchor: end;
}
.scatterplot-title {
  fill: black;
  font-weight: lighter;
  text-anchor: middle;
}
.scatterplot-title-y {
  writing-mode: tb;
  dominant-baseline: ideographic;
}

.tagcloud-legend {
  dominant-baseline: central;
}

.var-info {
  margin: 0.2em 1em;

  .var-name {
    margin: 0 0.5ch;
  }
  .var-value {
    margin: 0 0.5ch;
    font-weight: bold;
  }
}

.area-elections-list,.area-families-list {
  list-style: none;

  .title {
    // font-weight: bold;
  }
}
.area-elections-item,.area-families-item {
  margin: 0.8em 0;
  .number {
    padding-right: 0.5em;
    text-align: right;
    opacity: 0.2;
  }
}
.area-elections-item {
  .surname {
    font-weight: bold;
  }
  .nickname {
    opacity: 0.5;
  }
}
.area-families-item {
  .name {
  }
  .count {
  }
}
```

*scss/index.scss*

```
@import "common";

.splash-container {
  display: table;
  text-align: center;
}
.splash {
  display: table-cell;
  vertical-align: middle;
  width: 100vw;
  height: 80vh;

  color: white;
  text-shadow: 0 0 20px rgba(0,0,0,0.8);

  background: url("../img/header.png");
  background-attachment: fixed;
  background-position: center;
  background-size: cover;
  background-color: $color-background-alt;

  .logo-container {
    margin: 2.5em;

    .logo-large {
      margin: 0 auto;
    }
    .logo-type {
      font-size: 3rem;
    }
  }

  .splash-title {
    font-size: 1.5em;
  }

  .launch-container {
    margin: 2.5em;

    .splash-button {
      font-size: 1.5em;
      margin: 1em 0;
      padding: 1.5em 3em;
      box-shadow: 0 2px 44px 2px rgba
          (0,0,0,0.6) !important;

      &:hover {
        box-shadow: 0 3px 48px 2px rgba
            (0,0,20,0.6) !important;
      }
    }

    a, a:visited {
      color: $color-primary-2;
    }
    a:hover, a:active {
      color: white;
    }
  }
}

.content {
  color: $color-text;
  background-color: white;
  font-size: 1.5em;

  box-shadow: 0 -2px 44px 0 rgba(0,0,0,0.6);

  .text {
    padding-left: 12%;
    padding-right: 12%;
    margin: 6% 0;
  }
  .img-text {
    padding-left: 6%;
    padding-right: 6%;
    margin: 6% 0;
  }

  h1 {
    font-family: "Roboto Condensed", sans-
        serif;
```

```
  }
  p {
    margin: 1em 0;
  }

  .img-box {
    position: relative;
    overflow: hidden;
    min-height: 300px;
    transition: background-position 0.3s $ease
        -swift;

    &:before {
      content: "";
      display: block;
      width: 100%;
      min-height: 300px;
      transition: background-color 0.2s linear
          ;
    }
    &.fill-1:before {
      background-color: rgba($color-primary-1,
          0.8);
    }
    &.fill-2:before {
      background-color: rgba($color-complement
          , 0.8);
    }
    &.fill-3:before {
      background-color: rgba($color-primary-1,
          0.8);
    }

    .img-box-caption {
      position: absolute;
      padding: 1em;
      width: 100%;
      bottom: -100%;
      color: white;
      text-shadow: 0 0 0.5em rgba(0,0,0,1);
      background-color: rgba(0,0,0,0.6);
      transition: all 0.3s $ease-swift;
      transition-properties: top, bottom,
          background;
    }

    &:hover {
      background-position: center !important;
      transition: background-position 0.4s
          $ease-swift;

      &:before {
        background-color: transparent;
      }
      .img-box-caption {
        bottom: 0;
      }
    }
```

*scss/login.scss*

```
@import "common";

#body {
  // background-color: $color-background-alt;
  display: table;
  position: absolute;
  height: 100%;
  width: 100%;
  color: $color-text;
  background-color: $color-background;
}

.logo-large {
  width: 100%;
}
.logo-type {
  width: 100%;
  padding-bottom: 1em;
  font-size: 20pt;
  color: $color-text;
}
```

```
    }

    img {
      display: none;
    }
  }
}

$img-box-names: "choropleth", "bivariate", "
    upload", "upload-result", "tag-map", "tag
    -map-overlay";
@each $name in $img-box-names {
  #img-#{$name} {
    background-image: url("../img/" + $name +
        ".png");
    background-position: top;
    background-size: cover;
  }
}

.dark {
  color: white;
  background-color: $color-background-alt;

  h1 {
    color: $color-text-alt;
  }
}

.about-section {
  padding-top: 1em;
  padding-bottom: 1em;
  @extend .dark;

  .text-small {
    font-size: 60%;
  }
}

ol li {
  list-style-type: decimal;
}
ul li {
  list-style-type: square;
}

.footer-section {
  color: #909090;
  background-color: black;
  padding: 1em 10%;
}

.up-logo {
  width: 60px;
  height: 60px;
}
```

```
.login-container {
  display: table-cell;
  vertical-align: middle;
}
.login-container > * {
  margin-left: auto;
  margin-right: auto;
  width: 300px;
}

.login-box {
}

.input {
  width: 100%;
}

.login-buttons {
  margin: 1em 0;
}
```

# XI. Acknowledgement

Una sa lahat ay nagpapasalamat ako sa aking mga magulang na nag-alaga sa akin mula pa noong isinilang ako. Sila rin ang nagpaaral at lahat-lahat. Nagpapasalamat ako dahil naghirap sila para magkaroon kaming magkakapatid ng magandang buhay. Maraming salamat po sa inyo at sa inyong pagmamahal.

Ako po ay nagpapasalamat din sa aking adviser na si Prof. Sheila Magboo sa kaniyang mga payo at mga suhestiyon sa aking SP at proposal. Nagpapasalamat din ako kay Prof. Geoffrey Solano na nagbigay ng SP topic at sa kaniyang mga mungkahi kung paano ba ang magiging SP ko.

Salamat sa 2011 Block 12. Salamat sa mga kaibigan ko, sa mga nakakasama ko sa pagkain, free time, at klase, sa mga namimigay ng pagkain, sa mga napagtatanungan ng kung ano-anong mga kailangang gawin dahil makalimutin ako, sa mga nagpapaalala ng mga kailangang gawin (Marx!) lalo na na makalimutin ako, sa mga nagpapautang at mga nanlilibre, sa mga nakakalaro ko sa computer shop kahit mas gusto n'yong maglaro sa bahay ninyo (wala akong magandang computer eh), at sa mga okay lang kahit nawi-weirdo-han na sa akin. Salamat sa mga kaganapan, mga katuwaan, mga kalokohan, at mga karanasang pinagdaanan natin. Ako ay nagsisisi dahil sa huling mga buwan na lang ako naging malapit sa ilan sa inyo (at nagsisisi rin ako dahil naging malayo ako sa ilan sa inyo). Sana ay hindi pa rito nagtatapos ang ating buhay. Pagsasama, I mean. 'Yun lang. Makakatapos din tayong lahat!

Huli sa lahat, ako ay nagpapasalamat sa Unibersidad ng Pilipinas (at sa mamamayan ng Pilipinas). Tuwang-tuwa ako noong makapasok ako rito at ngayon ay mas natutuwa akong makalabas! Sa wakas!

Nagpapasalamat din ako sa 'yo, reader, dahil binasa mo ito. Wala ka sigurong magawa kaya mo ito binasa.