ProxiMap Documentation

Release 0.1

Matthew Moy de Vitry

Jan 27, 2018

Contents

Contents:

CHAPTER 1

Roadmap

1.1 Milestones

1.1.1 1. The scaffolding

- [ ] Sketch out application API
  - what data does each component consume and in what format
  - what services need to be set up to provide the data
- [ ] Create a reactive scaffolding for the app
  - [ ] decide which Angular UI to use, if any
  - [ ] create pages, menu bars, panels
  - [ ] link display to URL routes
  - [ ] make sure everything is responsive
- [ ] Implement language changing

1.1.2 2. The flesh

- [ ] Create a dummy dataset to work with
- [ ] Implement data management service
- [ ] Impement essential components
  - [ ] main menu
  - [ ] map
  - [ ] search list
  - [ ] mini detail view

1.1.3 3. Embellishments

- [ ] navigation
- [ ] full details view
- [ ] advanced search
- [ ] ...

1.2 Components

The following points refer to areas and panels in the UI. Please refer to mockup here. - Menu bar: contains logo, search bar, information button, language selection and city selection - Map: interactive map where fountains, the user's location, and navigation paths can be viewed. - List: list in which fountains are displayed and can be filtered (see 3c). The fountains are displayed in order of proximity to user (if user location available) or in alphabetical order. - Small information tab: when a fountain is selected, a summary of information about the fountain is displayed over the map. Information included in this tab depends on what is made available from the data sets, but it could include: - Construction year of fountain (e.g. 1951) - Type of water (e.g. Well water)

- Water quality (e.g. drinking water/not drinking water) - Detailed information window: provided more information about the fountain is available from Wikidata and/or Wikimedia, the detailed information window contains a detailed description and photos of the fountain. - Navigation pane: turn-by-turn instructions for how to get to a fountain. - Information pane: full-screen information about Zurich's fountains and about the application. Lots of pointers to a) open data and open source concepts b) call for action b1) on content completion/extension b2) technical efforts to make even more open

CHAPTER 2

Proximap services

The following services work in the background of the app:

2.1 App manager

2.1.1 Properties

The application's state is stored in an object. See application state variables for complete list of state properties.

2.1.2 Methods

`parse_url()` Checks and corrects url params and variables, and updates app state variables accordingly.

`update_state(variable_name, new_value)` Updates the state variable, first checking if the new value is valid. If so, the app state variables are updated and the URL is updated as well.

2.2 Data manager

2.2.1 Properties

Properties store the data. Data . . . . fountains

All fountains of selected city.

fountains_filtered filtered fountains.

fountain_selected fountain selected by user.

user_location coordinates

user_preferences user preferences, like preferred mode of transport

route_data routing data as acquired by routing service

2.2.2 Actions

- fetch_fountain_data()
- fetch_route()
- filter_data()
- set_fountain()
- geolocate_user()
- set_user_location()

CHAPTER 3

Proximap components

Note: Each component that is not always visible has a method to update its visibility based on the app state.

3.1 Map

The map is the main component of the app, and is always visible (though sometimes in the background).

3.1.1 Private Methods

`update()` This function updates the map to display desired information. It watches: `appManager.mode` to determine whether to show filtered fountains, a route, or just a single fountain `dataManager.fountains_filtered`, `fountain_selected`, and `route_data` in order to update the data displayed in the map

`update_user_location(user_location)` This function displays the user's current location on the map.

`display_route(route_data)` If the app is in `route` mode, this function displays the route on the map.

`display_fountains(fountains_filtered)` If the app is in `map` mode, this function displays the filtered fountains on the map.

`display_selected_fountain(fountain_selected)` If the app is in `details` mode, this function displays the selected fountain on the map.

3.1.2 Public Methods

Public methods can be called from the DOM or from other components/services.

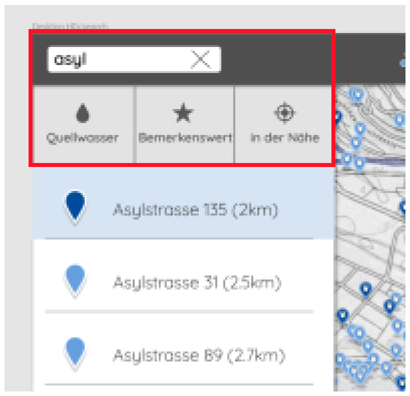
`select(fountain_id)` The user can select a fountain in the map. The function modifies the URL parameter accordingly.

`set_user_location(coordinates)` The user can set their location via the map as an alternative to automatic localization.

`refresh()` Force the map to refresh

3.2 Search

The search filter is accessed in the menu, floating over the map or at the top of the list (to be defined). It just allows easy modification of the fountain search criteria via the URL variables.



3.2.1 Private Methods

Question: are the styles directly bound to the app state variables?

3.2.2 Public Methods

`toggle_cat(category_name, value)`

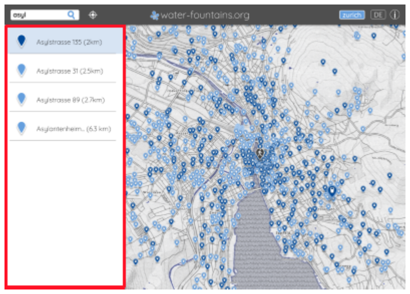
This toggles a category for the filter in the `search_cat` state variable. Examples of categories are:

- *well*: fountains with wellwater
- *historical*: fountains of historical value.

`text_filter(search_text)` Updates the full text for search in the `search_txt` state variable. Beware to sanitize the text!.

`reset_filters()` Removes all filters.

3.3 List



3.3.1 Features

Sort list elements by proximity, name, or construction date.

3.3.2 Private Methods

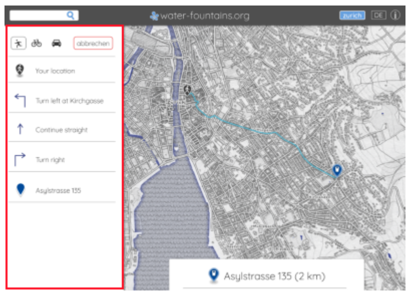
`update()` This function updates the list when the `fountains_filtered` data in the dataManager has been modified. It is triggered by a subscription to that data.

3.3.3 Public Methods

`select(fountain_id)` The user can select a fountain in the list. The function modifies the URL parameter accordingly.

3.4 Navigation pane

[low priority] This pane shows step-by-step navigation instructions.



3.4.1 Private Methods

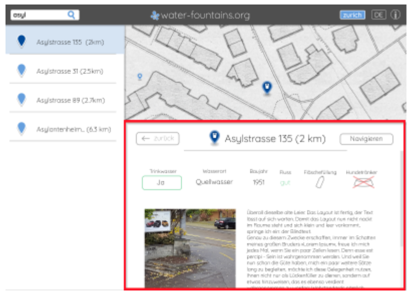
`update()` This function updates the route info when the `route_data` data in the dataManager has been modified. It is triggered by a subscription to that data.

3.4.2 Public Methods

`change_travel_mode(new_mode)` Updates travel mode.

3.5 Details pane

This pane displays information about the selected fountain. Information included in this pane depends on what is made available from the data sets, but it could include: - Construction year of fountain (e.g. 1951) - Type of water (e.g. Well water) - Water quality (e.g. drinking water/not drinking water) The pane also shows information available from Wikidata and/or Wikimedia, including a detailed description and photos of the fountain.



Überall dieselbe alte Leier. Das Layout ist fertig, der Text lässt auf sich warten. Damit das Layout nun nicht nackt im Raume steht und sich klein und leer vorkommt, springe ich ein: der Blindtext.
Genau zu diesem Zwecke erschaffen, immer im Schatten meines großen Bruders »Lorem Ipsum«, freue ich mich jedes Mal, wenn Sie ein paar Zeilen lesen. Denn esse est percipi - Sein ist wahrgenommen werden. Und weil Sie nun schon die Güte haben, mich ein paar weitere Sätze lang zu begleiten, möchte ich diese Gelegenheit nutzen, Ihnen nicht nur als Lückenfüller zu dienen, sondern auf etwas hinzuweisen, das es ebenso verdient

3.5.1 Private Methods

update() This function updates the displayed information when the `fountain_selected` data in the dataManager has been modified. It is triggered by a subscription to that data.

3.5.2 Public Methods

show_route() This function changes the mode of the app to `route` and triggers a route search between the user's current location and the selected fountain. The route search is managed in the dataManager.

3.6 Menu

The menu bar contains logo, search bar, information button, language selection and city selection. On mobile devices it is replaced with a menu button and slide-out menu on the right.

3.6.1 Public Methods

change_lang(new_lang) This function changes the language of the app.

change_city(new_city) This function changes the city of the app.

CHAPTER 4

Application state variables

The following variables define the application state and are set in the url of the web page.

state variable name	location	type	example	default	description
city	param	string	zurich	zurich	city for which data is being shown
fountain_id	var	string	q72592	none	unique identifier for fountain selected, if any
mode	var	string	map/details/route	map	application mode (determines which panels are visible)
search_txt	var	string	kluspla	none	filter text
search_cat	var	list	[well, favorite]	[]	filter categories that can be toggled on/off
hide_list	var	bool	true/false	true	in mobile mode, hide list by default
lang	var	string	en/de/fr	en	language of app

4.1 App state propagation

A service watches the url parameters and updates the internal state variables. The application data manager and view manager watch the state variables for changes and make modifications to the view and data, respectively. Individual components watch the data for changes and update independently.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search