**Heatmap Application Description and Usage**

**Description:** Heatmap application receives temperature sensors readings from *iot.op-bit.nz* server, stores them in the database used by Django powered backend in the form of current reading for the existing sensor instance and History instance associated with the existing sensor. Then the API provides it as a JSON to the React.js powered front-end. The front-end that draws a circle for each existing sensor on top of the pre-drawn .svg map of a building floor where sensors are located.

**Backend:**

1. All the logic of connecting to the IOT server and processing data from sensors is located in ‘project-root/api/mqtt.py’ file and ‘project-root/api/\_\_init\_\_.py’. While mqtt.py file holds all the logic, the \_\_init\_\_.py only starts the mqtt client which is described in the mqtt.py file when server is started.
2. Authentication utilizes Django Rest Framework token authentication. Each user receives a token upon user creation stage – look for ‘create\_auth\_token’ function in models.py file
3. In order to receive the token, the user should send the POST request according to this example:

fetch("server-address/api-auth/", {

method: 'POST',

body: JSON.stringify({ 'username': username, 'password': password }),

headers: {

'Content-Type': 'application/json'

}

})

And the response looks like this:

{

"token": "fe7d487654916b70531faf2054a636873ec43596"

}

This token should be stored and used for all POST/PUT/DELETE requests and to see the users data

1. The API is accessible at the following routes which are described in ‘project-root/api/urls.py’:
   1. ‘server-address/api/sensors/’ – returns a JSON with all the sensors:

[

{

"serialID": 2222222277771151,

"name": "Jack",

"temperature": 20.7,

"humidity": 53.1,

"x": 1071,

"y": 767,

"floor": "two"

},

...

]

* 1. ‘server-address/api/sensors/<serialID>/’ – return JSON with the requested sensor and History instances associated with it:

{

"sensor": {

"serialID": 2222222277771151,

"name": "Jack",

"temperature": 21.2,

"humidity": 63.3,

"x": 1071,

"y": 767,

"floor": "two"

},

"history": [

{

"sensor": 2222222277771151,

"temperature": 22.6,

"humidity": 53.1,

"timestamp": "2019-04-29T16:45:05.993118+12:00"

},

{

"sensor": 2222222277771151,

"temperature": 23.7,

"humidity": 48.5,

"timestamp": "2019-04-29T16:53:53.942154+12:00"

},

...

]

* 1. ‘server-address/api/history/’ – returns a JSON with all the History instances:

[

{

"sensor": 2222222277771160,

"temperature": 23.6,

"humidity": 39.5,

"timestamp": "2019-04-29T16:12:07.045149+12:00"

},

...

]

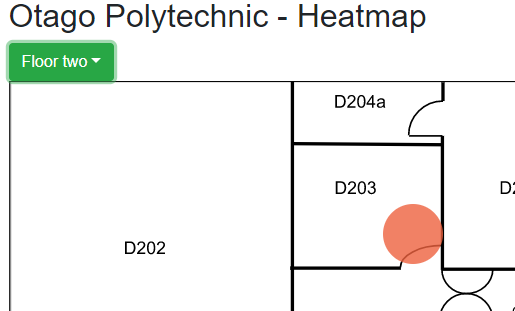
* 1. ‘server-address/api/users/’ – returns a JSON of all the registered users data, requires authentication (see Point 3)
  2. ‘server-address/api/users/<userID>/’ – returns a JSON of the user data, requires authentication (see Point 3)

**Front-end:**

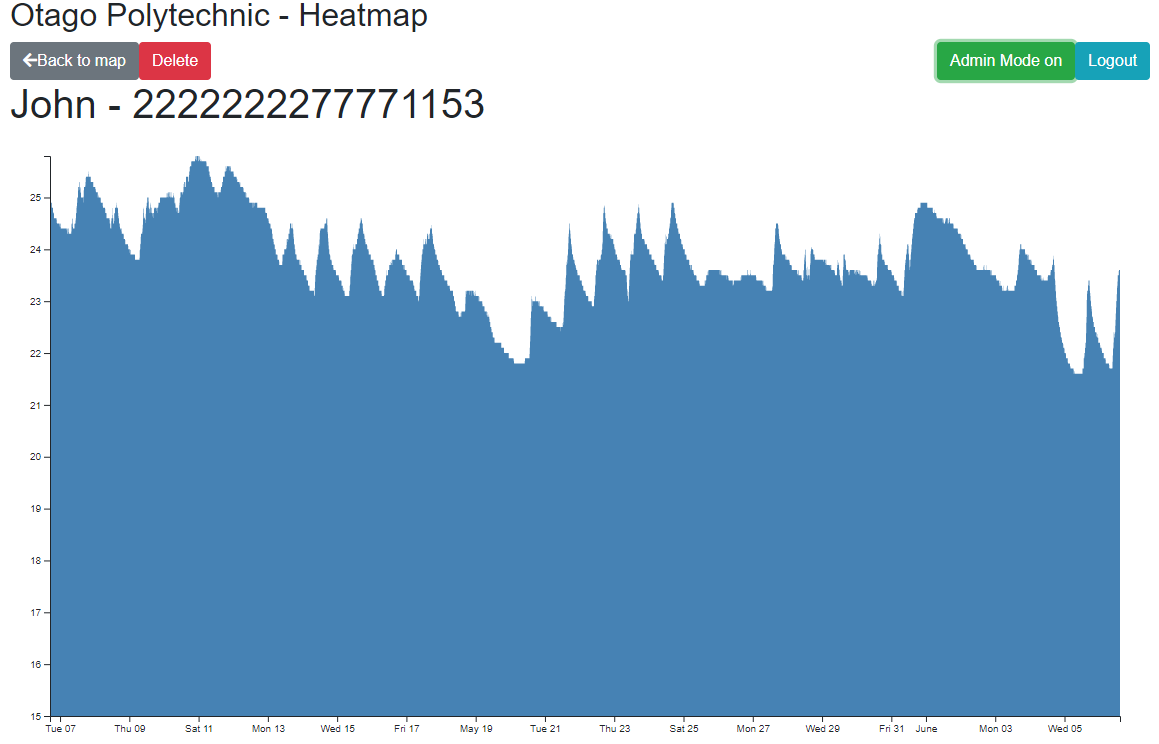
1. The front-end is built using React.js library and Redux state management. All the Redux related logic is located in the following files:
   1. ‘project-root/front-end/src/store/store.js’ – creating Redux Store object which is combined from rootReducer (sensors related storage) and loginReducer (auth token related staff) which are in turn located in ‘project-root/front-end/src/reducers’ folder.
   2. ‘project-root/front-end/src/actions’ folder contains action-types.js – string variables of Redux action types, and actions.js – contains types of actions triggering changes in the Redux state/store
   3. ‘project-root/front-end/src/index.js’ – wraps whole App component with a Provider Higher Order Component which makes sure the Redux state/store is available across the whole React app
2. All functions that interact with back-end API are located in ‘project-root/front-end/src/utils/utils.js’ file. They should be pretty easy to understand.
3. ‘project-root/front-end/src/history/history.js’ – create BrowserHistory instance in order to use it in the React Router
4. ‘project-root/front-end/src/App.js’ – adds Header and Toolbar components to be available across all the routes and Router component which specifies the navigation within the whole React app.
5. ‘project-root/front-end/src/components’ – this folder contains all the components that represent the UI of the app. There is no better way to learn the app logic than just to dig into each of the component and study it.
6. ‘project-root/front-end/build/static/media’ – this folder holds the svg files with maps for different premises. Then this files are being used as a background for the selected Premise within css. This is something that can be improved by creating a model, for example Premise, on the backend, adding FileField to it and then fetch it in the front-end and add as a style attribute to the Heatmap component based on what Premise is selected.

**Usage:**

1. Navigate to the main screen
2. Select the Premise by clicking the left top green Select to navigate between different premises



1. Login by pressing Login button in the top right corner to get access to advanced interactions like adding new sensors and delete existing ones (optional).
2. Click Circles representing heat sensors to see more details and historic data.



1. When cursor is hovering over the graph user can zoom-in/out by using a mouse scroll wheel to change the period the data is displayed.
2. Activate Admin Mode by clicking Admin Mode off button next to Logout (if you are logged in) to get access to deleting this particular sensor permanently
3. When on main screen with the map, logged in and Admin Mode on, clicking the map brings up a form to add a new sensor. Notice that X-Coordinate and Y-Coordinate are prepopulated automatically based on the position of the mouse cursor at the time of clicking:
   1. Sensor UID: this must correspond to the UID of the heat sensor given to it within IOT infrastructure
   2. Sensor Name: Arbitrary data for easier mapping between Heatmap app and IOT infrastructure.

