Firebase

This worksheet aims to explore the Google Firebase service, namely the Cloud Firestore. Cloud Firestore is a flexible, scalable NoSQL cloud database to store and sync data for client- and server-side development.

1) Firebase Configuration

Firebase has several "products", including authentication, databases, messaging, etc.. The available (Javascript) SDK makes its integration with (Web) apps easier as it hides the necessary requests and responses inside an API. There is also a command-line tool, Firebase CLI (Command Line Interface), that allows to develop and test locally the Firebase integration and then deploy the (web) application to a server.
This class only uses the Firebase Realtime Database product using requests and processing responses. The exploration of the Firebase SDKs is beyond the scope of this class.

a) Go to the Firebase console (https://console.firebase.google.com) and create a project in Europe, if you don't have already one;

b) Create a "Realtime Database" in "test mode" (the database will be publicly available and it must be secured as soon as possible). It is also possible to use the "Cloud Firestore" database, but you need to adapt the syntax of URLs and the response structure;

c) Create the collection that will store the data from your app.

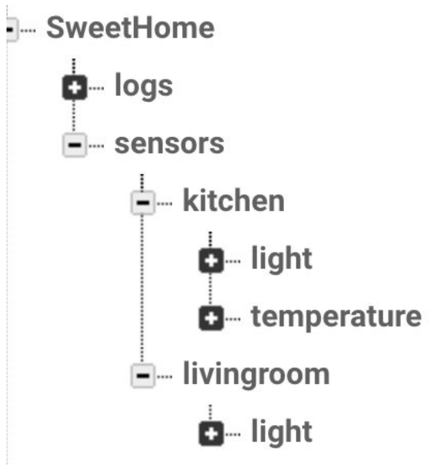
2) Cloud Database

Store the current value of all sensors associated with a top level key "sensors" and the change for each sensor with a "logs" key. Sensors states should be updated using a PUT to the ".../sensors.json" URL (replace the bold project name and top level node):

```
curl --request PUT \
  --url https://ipleiria-marcelino.firebaseio.com/SweetHome/sensors.json \
  --header 'Content-Type: application/json' \
  --data '{
        "kitchen": {
              "light": {
```

```
                        "status": false,
                        "timestamp": 1633299120
                },
                "temperature": {
                        "status": false,
                        "timestamp": 1633399120
                }
        },
        "livingroom": {
                "light": {
                        "status": true,
                        "timestamp": 1633290120
                }
        }
}'
```

This PUT request will create a structure that can be visualized in the Firebase console:



To fetch the current state of all sensors for the above database we would use the request:

```
curl --request GET \
  --url https://**ipleiria-marcelino**.firebaseio.com/**SweetHome**/sensors.json
```

All logs will be added under a "logs" key. To add a new log entry to the Firebase you can use a POST request, with an object that includes the location and name of the sensor, use the following URL:

```
curl --request POST \
  --url https://**ipleiria-marcelino**.firebaseio.com/**SweetHome**/logs.json \
  --header 'Content-Type: application/json' \
  --data '{
        "timestamp": 1633299640,
        "zone": "kitchen",
        "sensor": "light"
}'
```

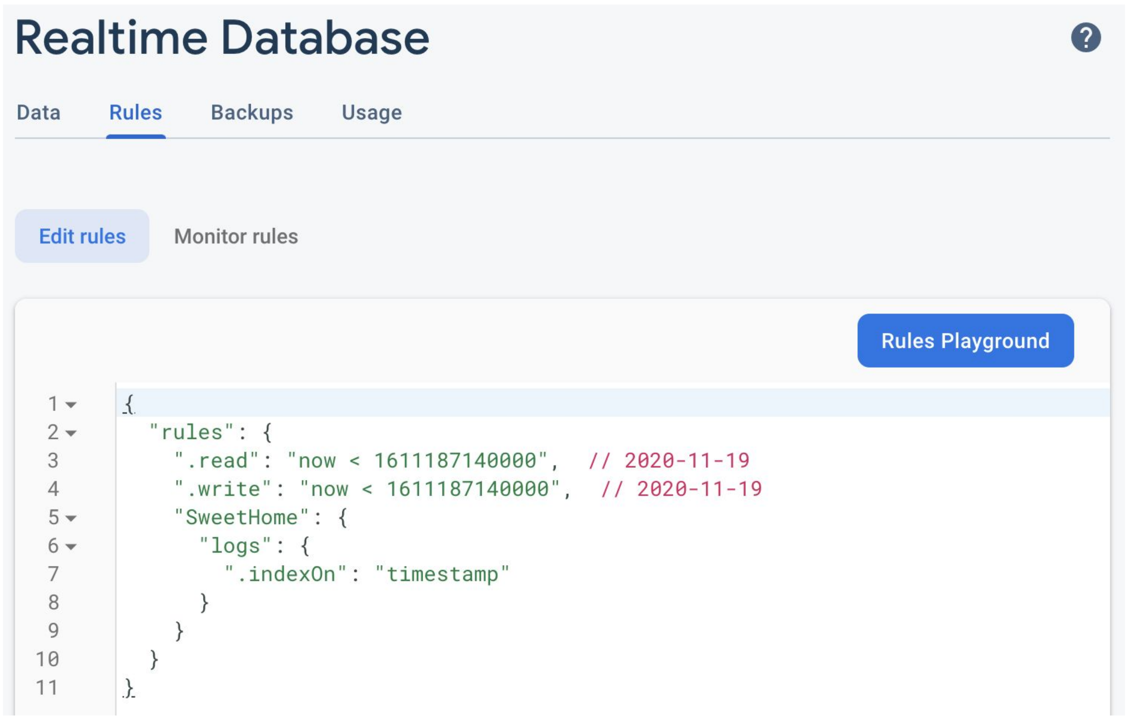
This request will return a response with the name of the key assigned to the object just added. For example:

```
{
  "name": "-MLDmsSnafe4kIeYYY9j"
}
```

When fetching the logs, we do not want all of the records, as records build up with time and will grow to a large dataset. To limit the size of the response we will use a filter that will order by the "timestamp" and only returns the last 30 records:

```
curl --request GET \
  --url
'https://ipleiria-marcelino.firebaseio.com/SweetHome/logs.json?orderBy=%22timestamp%22&limitToLast=30'
```

To be able to search by "timestamp" it is necessary to mark this property as an index. For that, it is necessary to configure the rules (change the name of top level node if required):



```
{
  "rules": {
    ".read": "now < 1611187140000",  // 2020-11-19
    ".write": "now < 1611187140000",  // 2020-11-19
    "SweetHome": {
      "logs": {
        ".indexOn": "timestamp"
      }
    }
  }
}
```

3) Home Sweet Home Firebase

Adapt the web app created in the previous classes to have the following features:

- Update (PUT) the current value of each sensor into the Firebase database and add (POST) each change to the log entries (a proper server implementation would just receive one of the communications to update the two entries)
- List the history (log) of the interactions with the sensors
- Change the pages (Dashboard and History) so that their values are updated every 10 seconds.

Reference

https://firebase.google.com/docs
https://firebase.google.com/docs/web/setup