Dashboard

Summary

1. Overview

The goal of the project is to create a dashboard to get all the information you need, in only one place. We had to create a service that offered different widgets that were customizable per user, with specific data related to the service.

A user can register with an email and a password, or with google OAuth2.

2. Technical Stack

To create this project, we needed to realize a front and a back-end, and a database to store users and users configurations.

For the front-end, we used React.js with create-react-app, and used tailwind for the styling of the different react components.

For the back-end, we used Node.js with Express.js to create the server.

For our database and everything user-related, we used Firebase with Firestore, which is a noSQL database, perfect for storing user information as documents.

3. Project Structure

a. *For the front:*

- Folders:
  - **Pages** (full pages like Auth, Home, Configure)
  - **Components** (reusable components used on multiple pages, Background, Sidebar, Navbar)
  - **Widgets** (all widgets instance are in this folder)

The front is launched with port 3000.

b. *For the back:*

- Folders:
  - **Routes** (contains every route related to a particular widget)

For all services provided by the back, you can view them with a get request on port 8080/about.json.

4. Database structure

As stated before, the firestore database is a noSQL database, so to store each user's information, we've created a collection user, that contains, for each user:

- name
- email address
- avatar

Now to store each user's configuration, we've created another collection, namely settings, that stores an object with details on each service we are offering, for every widget.

Example of such document:

```
…
nasa {
     display: true,
     refresh: 5,
     type: "apod",
}
```

5. API's used

Each service is using a different API, and so here is a list of all APIs used with each related service:

- **Youtube** API v3 => Youtube service
    - Get channel Statistics
    - Get channel last video
- **Nasa** api => Nasa service
    - Get Curiosity Mars's rover last photos
    - Get the astronomical picture of the day
- **Github** graphql api => Github service
    - Get user's contributions in the last year
    - Get user's profile
    - Get user's pinned repositories
- **Coinbase** api => Currency service
    - Get the exchange rate between two currencies
- **Reddit** api => Reddit service
    - Get the best posts from subreddit
- **Zenquotes** api => Quote api
    - Get a random quote from given category
    - Get the quote of the day from given category

- **Openweathermap** api => Weather api
    - Get the current temperature of a given city

6. Build and deployment

To build the project, you can either :

- enter this command at the root of the project : docker-compose build && docker-compose up
- Requirements: Docker and Docker-compose
- Application will be available on localhost:3000 and the server will be on port 8080

Or :

- Go in client/ directory, enter : npm install && npm start
- Go in server/ directory, enter : npm install && npm start
- Requirements : npm and node
- Application will be available on localhost:3000 and the server will be on port 8080