Area 51 Technical Documentation

Clerin Emeryck – Dudot Lucas – Guillaume Robin – Drules Maximilien – Minguet Jules – manœuvre Lorenzo

Summary

I. Project overview.

This is the technical documentation, it will detail all the technical parts and the code of the project.

If you want to know how to use the application you are invited to read the provided user's guide.

The AREA project consists in the creation of a software suite that functions similarly to IFTTT and/or Zapier.

This software suite is a divided into 3 parts:

- A server to implement all the features.
- A web application to use the app from a browser.
- A mobile application to use the app from a phone.

This project uses the following languages:

- Server → NestJS & Postgres.
- Web application → React / TypeScript.
- Mobile application → Flutter.

II. Libraries used.

The Server user jwt to generate the token for each user.

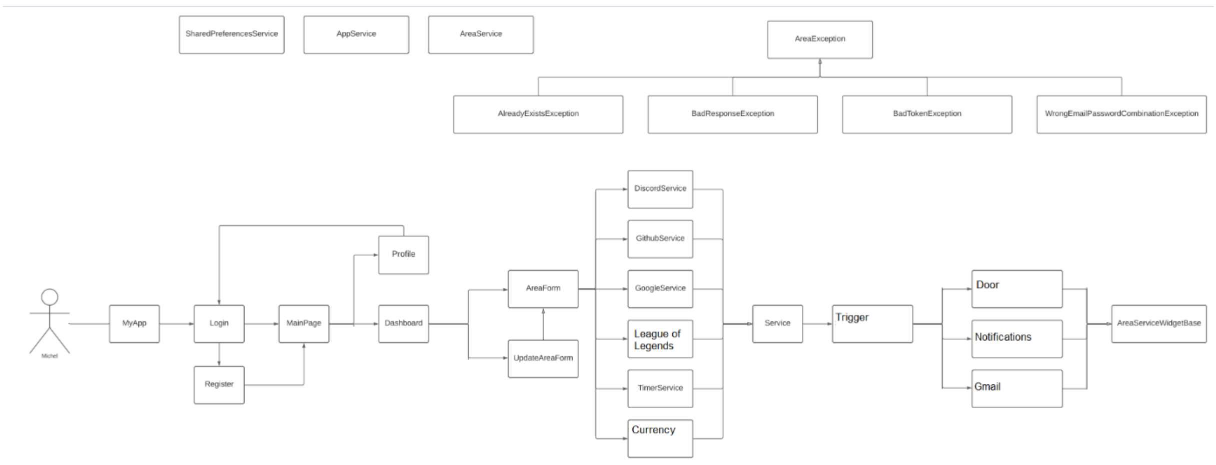
III. Project structure.

The project is structured in 3 directories each one contains the code for the corresponding part: Front, Mobile and Server.

The project is built using Docker, this will be detailed in a next section of the document.

IV. Project structure.

Mobile :



V. Apis used.

Here are the different APIs used for this project.

- Youtube API (https://developers.google.com/youtube/v3).
- Google API (https://developer.Gmail.com/en/docs).
- Github API (https://developer.github.com/v3/).

VI. Database structure.

This project uses Postgres admin as database manager.

It is integrated using Nestjs to manage the interactions between the front server and the API server.

This database contains several models, such as User,

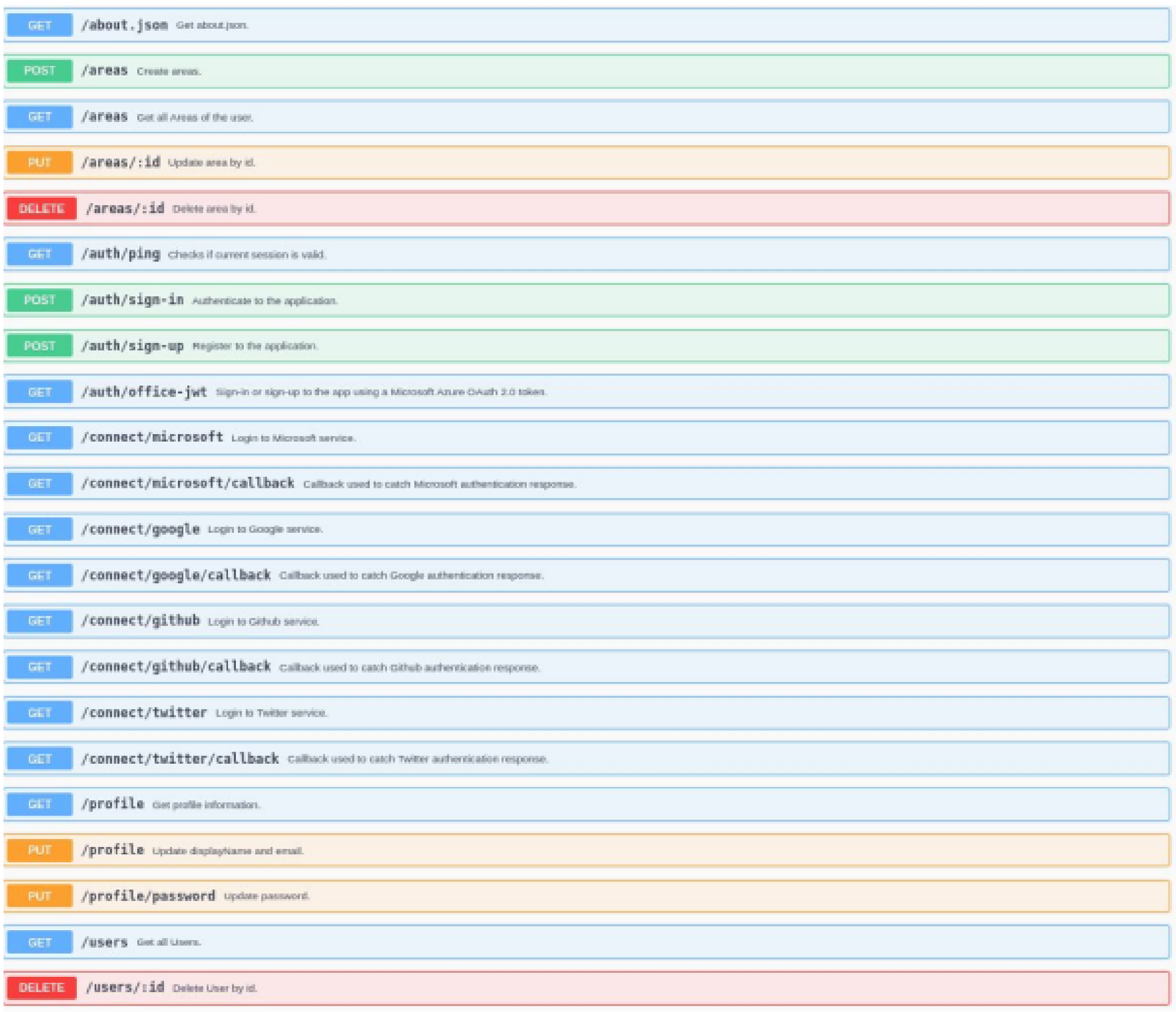
Starting with the User model, the schema contains different fields such as :

- email : email address (type: String)
- password : encrypted password (type: String)
- Description : Description (type: String)
- Name : Name (type: String)
- Image : Image(type: String)

Next, The Services model will store datas about the action

- Owner: email address (type: String)
- Action: Name of the action (type: String)
- ActionParms: Parameter needed for the action (type: String)
- Trigger: Trigger (type: String)
- TriggerParms: Parameter needed for the Trigger (type: String)
- Reaction: Name of the action (type: String)
- ReactionParms: Parameter needed for the action (type: String)

VII. Server endpoints.



```
GET     /about.json  Get about.json.
POST    /areas  Create areas.
GET     /areas  Get all Areas of the user.
PUT     /areas/:id  Update area by id.
DELETE  /areas/:id  Delete area by id.
GET     /auth/ping  Checks if current session is valid.
POST    /auth/sign-in  Authenticate to the application.
POST    /auth/sign-up  Register to the application.
GET     /auth/office-jwt  Sign-in or sign-up to the app using a Microsoft Azure OAuth 2.0 token.
GET     /connect/microsoft  Login to Microsoft service.
GET     /connect/microsoft/callback  Callback used to catch Microsoft authentication response.
GET     /connect/google  Login to Google service.
GET     /connect/google/callback  Callback used to catch Google authentication response.
GET     /connect/github  Login to Github service.
GET     /connect/github/callback  Callback used to catch Github authentication response.
GET     /connect/twitter  Login to Twitter service.
GET     /connect/twitter/callback  Callback used to catch Twitter authentication response.
GET     /profile  Get profile information.
PUT     /profile  Update displayName and email.
PUT     /profile/password  Update password.
GET     /users  Get all Users.
DELETE  /users/:id  Delete User by id.
```

VIII. <u>Project build.</u>

This project is built using Docker.

The project structure separates the back, the web and the mobile servers, therefore, there is one Dockerfile in each folder with a docker-compose.yml file at the root.

The file docker-compose.yml will call the Dockerfile in each source (/server/, /front/& /mobile/) and build the entire project as well as launching the servers.

Dockerfiles will simply build the three folders (server, front & mobile) so docker-compose.yml can launch the whole build.

Front server : localhost:8081

Back server : localhost:8080

Mobile apk download : localhost:8080/client.apk