AREA APPLICATION

Project’s goal:

The goal of this project is to allow users to trigger reactions thanks to specific actions (Action/REAction).

An action is triggered when a specific condition is met. A reaction is triggered when an action is triggered.

The project is composed of 2 parts:

- The server: the server is the core of the project. It is responsible for the management of the actions and reactions.

- The client: the client is the interface of the project. It is responsible for the management of the users and the configuration of the actions and reactions.

The client allows the user to select any action available that will trigger any reaction.

Available services and their actions/reactions:

OpenWeather service (using OpenWeather API)

- *Actions:*

    - If the indicated coordinates' current temperature reaches the indicated temperature

    - If the indicated coordinates' max temperature of the day reaches the indicated temperature

    - If the indicated coordinates' min temperature of the day reaches the indicated temperature

    - If the indicated coordinates' current humidity reaches the indicated humidity

    - If the indicated coordinates' current pressure reaches the indicated pressure

    - If the indicated coordinates' sea level is of the indicated value

  - If the indicated coordinates' ground level is of the indicated value

- *Reactions:*
  - None

Github service (using Github API)

- *Actions:*
  - If a new repository is created in the application's organization
  - If a new repository is deleted in the application's organization
- *Reactions:*
  - Create a new repository with a specified name in the application's account
  - Delete a repository with a specified name in the application's account
  - Create a new branch with a specified name in the application's repository
  - Delete a branch with a specified name in the application's repository

Chuck Norris joke service (using Chuck Norris API)

- *Actions:*
  - None
- *Reactions:*
  - None

This service is used for another one. The API is used to get a random joke, and the joke is sent to the user via email with another service.

Spring Boot Mail service (using Spring Boot Mail & Gmail API)

- *Actions:*
  - None
- *Reactions:*
  - Send an email to the user with a Chuck Norris joke

Timer service (using real time)

- *Actions:*
  - If the current minute is a multiple of 5

- *Reactions:*

    - None

Technology used:

Front-end (Client) :
The front-end uses the Angular framework (Typescript) for the web and mobile application.

Back-end (Server):
The back end uses the Java Spring Boot framework.

Database :
The database is made with MySQL.

Back-End architecture:

The back end used two kinds of entities:

- User, which stocks all the user's information like email, username, and password.
- Object, which stocks all the actions and reactions of the user.

We also use POST, DELETE and GET requests, sent by the front (the client) to register or login a user, and to add actions and reactions.

The Service folder contains all the functions and classes used to make actions and reactions with the associated services.

The Entity folder contains the two main entities (User and Object).

The Repository folder contains the Interface of the two main entities, allowing to use some useful functions.

The Controller folder contains all the classes allowing the GET and POST requests.

The Configuration folder contains the "core" of the back-end, like the "checkAction.java" file, containing the code checking the action and triggering the reactions of all users.
It retrieves the list of actions and reactions of each user, and loops between them.
The front-end also sends some information to dynamically trigger actions/reactions (by example, using user-specified weather conditions, to create a user-specified repo).

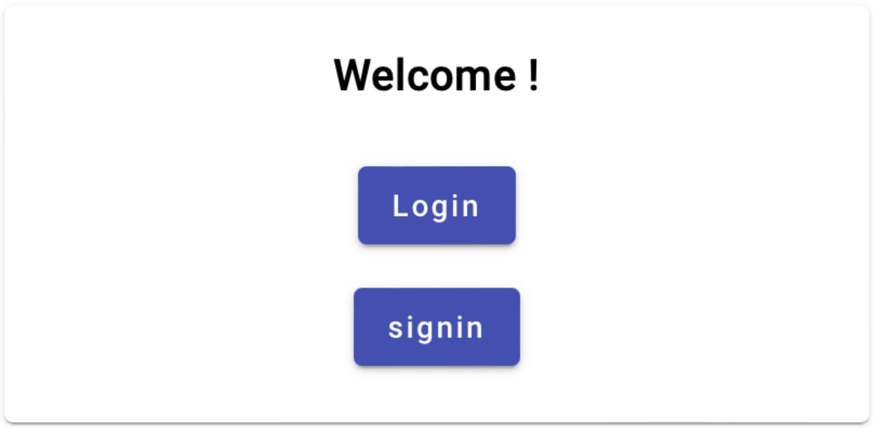
Front-End architecture:

The project uses Angular as a framework.

The objective is to create an action-reaction system based on the services that the user will choose.

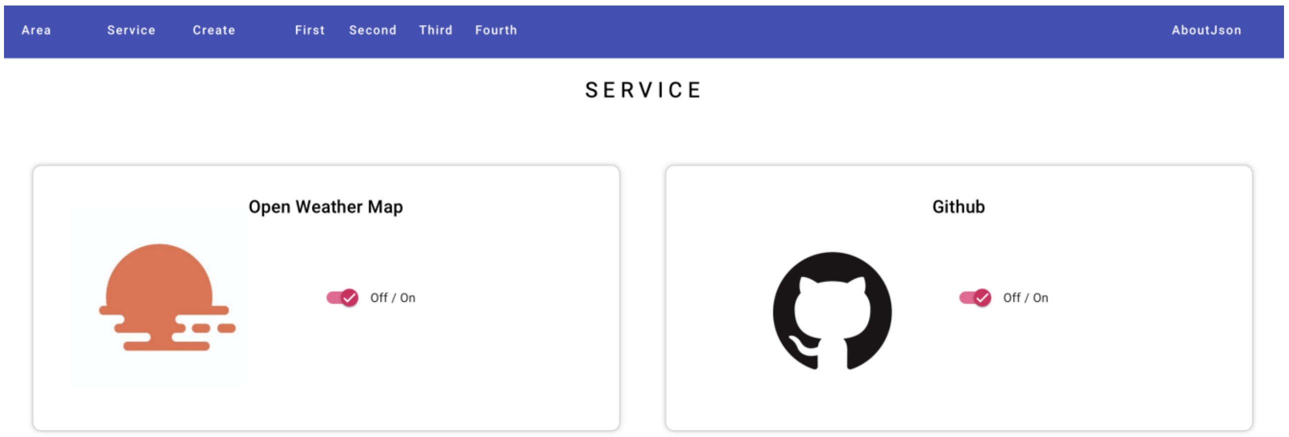
The user can register and login through the "userLogin" and "userRegister" routes on the back-end side.

The Login and register component can be find at Area/Src/app/login and Area/Scr/app/Register.



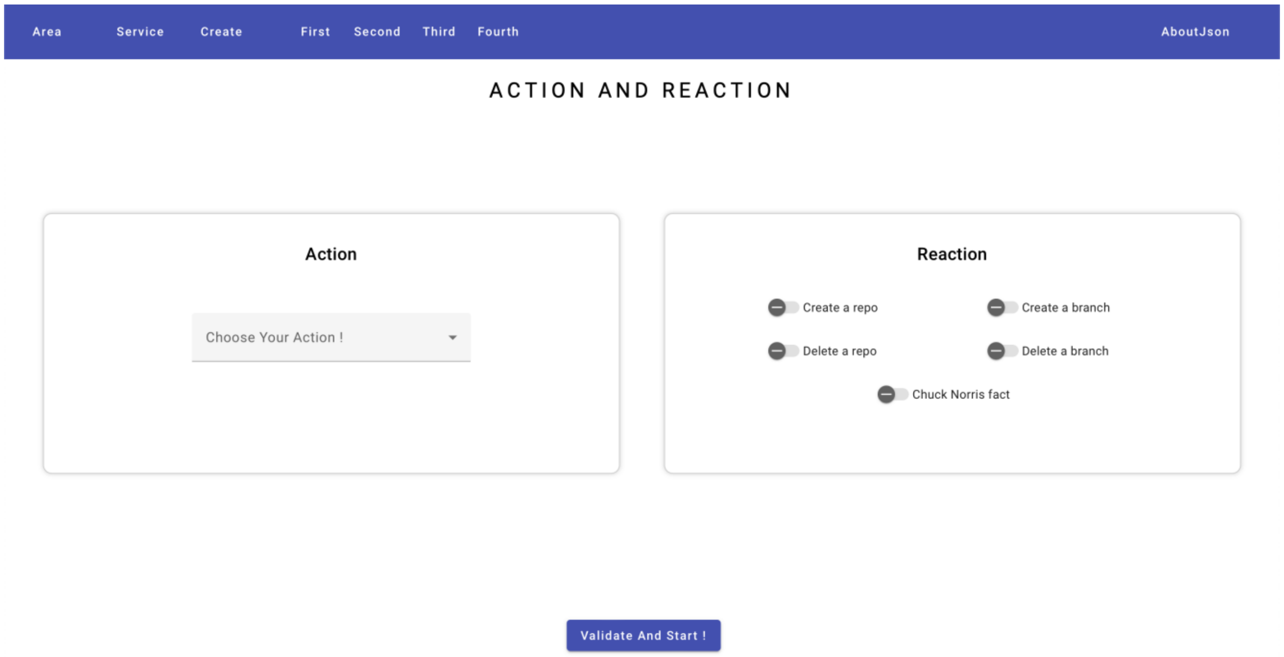
The user can choose their services and create an action page.

The Service component can be found at Area/Src/app/apiService



On this page, he can choose an action and one or more reactions.

The action component can be found at Area/Src/app/ActionX


Area
Service
Create
First
Second
Third
Fourth
AboutJson
ACTION AND REACTION
Action
Choose Your Action !
Reaction
Create a repo
Create a branch
Delete a repo
Delete a branch
Chuck Norris fact
Validate And Start !