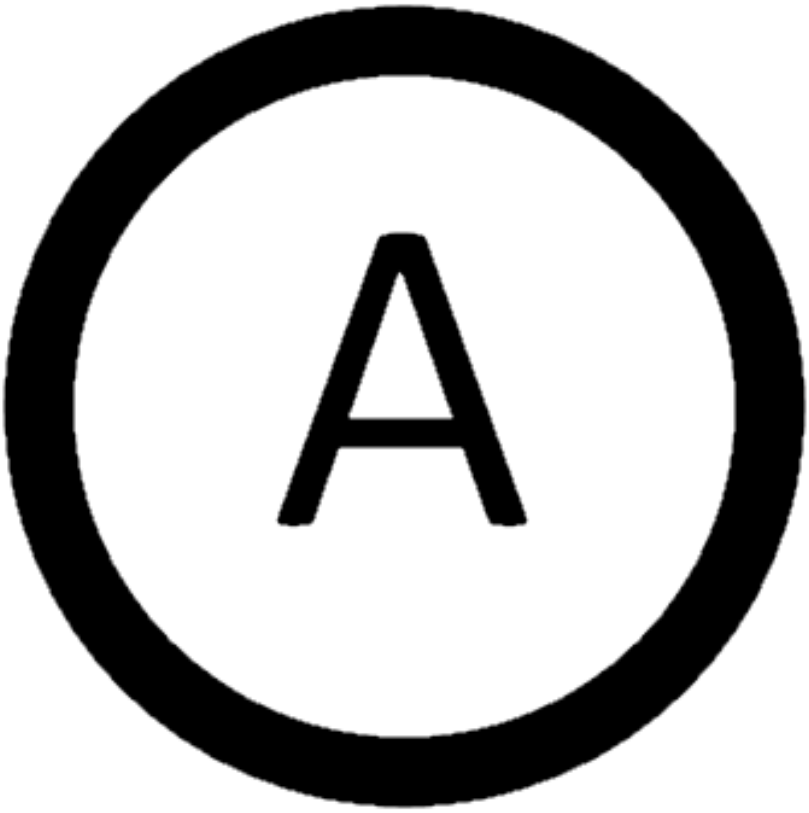
Area Documentation

Epitech 2022



Romane Bézier, Esther Bailloux, Gabin Warnier-de-wailly, Fabien Brosset, Hugo Berthomé

Index

1- Resume

Area consists of building a REST-type API in order to connect different services to each other. Like Ifttt or Zappier, the server will perform certain reactions based on certain actions performed by the services.

For example: When a message is received on a Slack conversation, it is also transmitted on a Messenger conversation.

Each connection is customizable by the user, between the services to which he has previously subscribed.

Our goal is to connect 7 services together and carry out 18 actions / reactions that the user can use.

The API server will be accompanied by two client interfaces:
- A website
- A mobile application

These interfaces have no logic but only display to better interpret the API.

2- Docker

What is "Docker" and how does it work in this project. Like "containers" and all of those things. Currently, we have 5 "containers" which each contain a part of our project:

- Db for database
- Server for the api
- Client_web for the web client
- Mobile for the mobile client
- Nginx to link the web client and the api

1- Database

On the database container there is a MySql image. That means that all data is stocked on this container. Note, this data is stocked on a volume. If the container is shutdown. All the data will be saved on a volume. When the container is started up, it loads the data of the volume.

2- Api

On the API there is the http://localhost:8080/ You can also use swagger on the page.

3- Web client

On the Web client, there is the page to connect your account, create AREA's and link your account to ours services. We got a shared space on volumes between this container and the mobile client container.

4- Mobile Client

On this container, we build an apk for our application. This container is destroyed after the end of the creation. Just before the end of the container, the apk is moved on the shared volume with the web client. You can go on this URL to download the apk: http://localhost:8081/client.apk

5- Nginx

Nginx is a quite particular container. It's a web server that link the api and the client. They create a bridge/tunnel between them to communicate. Without this container, you can communicate with api from the client and vice versa.

We talk about volumes. When you close you Docker. All containers are deleted and all data on them are destroyed and lost. But there are volumes. You can stock data on them. When containers are destroyed, data on volume aren't deleted. To delete volumes and reset the data, you have to prune them with this command: docker system prune - - volumes. With this command, you'll lose all your data.

We have also a network inside our docker. It used to have the same network configuration between all the containers. We also assign fixed up and each container. So, we know ip's and we can communicate freely between the containers. Without that, ip's are generated automatically and you have to ask to your container to your ip. When you restart, it's changes the ip and you must do it again. It's a huge waste of time and resources.

We have explained the docker-compose. But each container has a Dockerfile to describe the build phase. Besides, the docker compose is used for the uptime part. Before there is the build time. Which is made thanks to docker files. Each docker file describe the build time. For example, for the Mobile client, it describes the phases to build our apk and copy on volumes.

With docker files and the docker compose file, we open port. For example, the database port is 3306. For the web client is the 8081 and the api is 8080. If you want to change them. You must edit some files. For example, for the web client, you have to edit the docker compose, the docker file of the web client and the .env inside the folder of the web client.

You want to test our project like creation of AREA, login, actions, services etc.. You can launch a test docker compose. Be careful, if tests are launched before the end of the creation, some of them will not succeed. Relaunch it after the end off the database and all of them will succeed. We can fix that sorry ☹

3- Services

1- Github

It is a collaborative tool for working on code. It facilitates the creation of documents and their centralization with revision tools.

a- Actions

- New push : A push on a repository is detected
- New pull request : A new pull request has been created

2- Spotify

It's a music streaming platform

a- Actions

- Add new song : A new song has been had to a playlist

b- Reactions

- Add new song : Add a song to a playlist

3- Twitter

Twitter is a social media working with hashtags

a- Actions

- A user has twitted

b- Reactions

- Create a tweet

4- Outlook

Email service created by Microsoft

a- Actions

- An email has been received
- A new event has been added to the calendar

b- Reactions

- Send an email
- Create a new event inside the calendar

5- Onedrive

Online file storage created by microsoft

a- Actions

- A new file has been added
- A file has been deleted

6- Discord

It's an online message and voice chat

a- Actions

- A new message has been received
- A user joined a server

b- Reactions

- Send a message

7- Trello

It's a collaborative tool for tracking tasks in a team

a- Actions

- A new card has been had
- A deadline is approaching

b- Reactions

- Add a new card

4- Serveur

a- Architecture

The server has been created with node.js and express framework to create a REST API easily. The server is made from a MVC representation. Model – View – Controller but without views here because we only send data from a request, so it's replaced with the routes here. The data is stocked inside a mysql database *cf_ database*

- The routes received the request from client and redirect the request to the appropriate controller
- The controllers will do all the logic and getting all the data necessaries to respond at the request
- The models will connect to the extern API but also the database where the data is stored

b- Authentication

All the clients can register on the API by email but also with OAuth2 (microsoft). When a client is login, it will receive a Json Web Token (JWT) to identify himself for further request, as creating a relation between two services. The token has to be given inside the header of the appropriate requests.

For each service, the client will have to connect as OAuth2 to use the API calls provided.

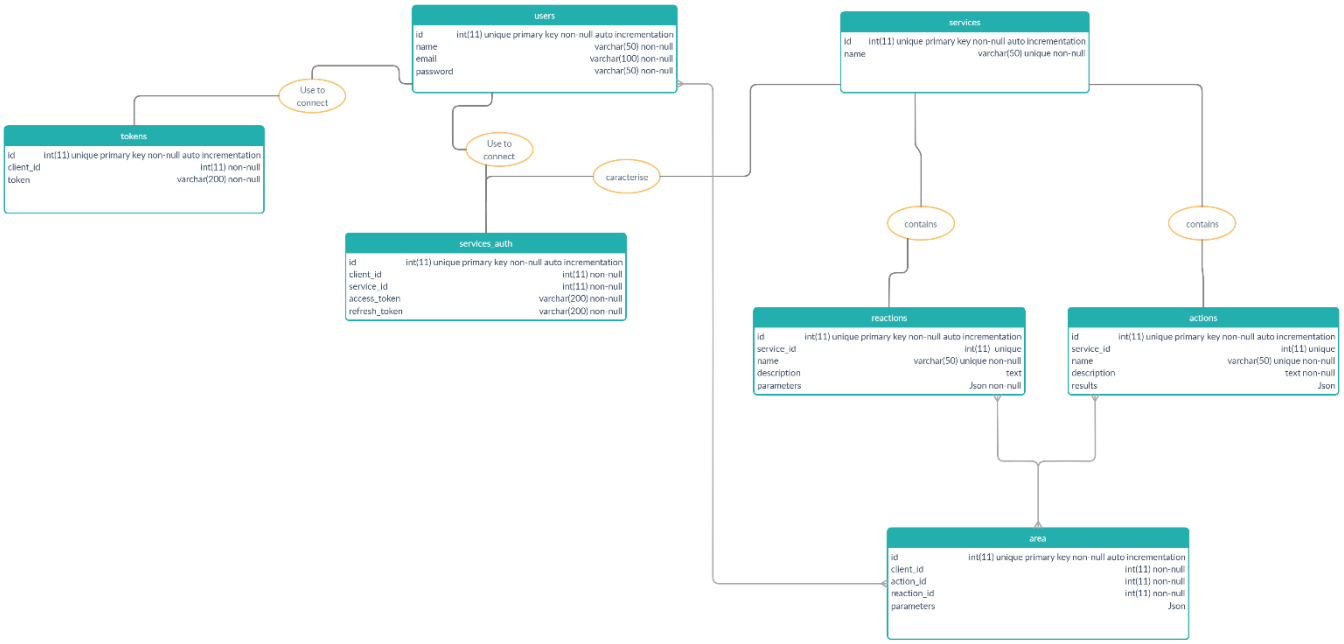
c- Swagger

A swagger documentation has been generated to see all the available routes and to test each route directly inside the browser. You can access it by going on this url : http://localhost:8081/api-docs/

You can also see all the available routes inside the **README.md**

d- Database

The database uses the MySQL technology and is named area.



e- Discord

How does our Discord API Works ?

In order to use our discord areas there is some things to setup first.
Start by updating the .env file by adding :
- a valid discord bot token
- a valid discord app token
- a valid bot permissions url

Once you are done you can get started !
Our Discord actions are :
- Receiving a new message
- Arrival of a new member in a Server

Our Discord reactions are :
- Sending a new message

ACTIONS :
Receiving a new message -->
There is some parameters available, the client can edit :
- string {ServerName} on which we'll check for messages (if empty or null it will check for receiption of private message)
- string {ChannelName} on which we'll check for messages (leave empty to check for the whole server)
- string {Username} of whom we'll check for messages (leave empty to check for any user)
- string {Content} if you which to check for a specific message content (leave empty to check for any message content)

Arrival of a new member on a Server -->
There is some parameters available, the client can edit :
- string {ServerName} on which we'll check for new members
- string {Username} if you wish to check for the arrival of a specific discord User (leave empty to check arrival of any User)

REACTIONS :
Sending a new message -->
There is some parameters available, the client can edit :
- bool {ToServer} indicates if the message is to be sent on a Server (true) or to an User (false)
- string {ServerName} on which we'll send the message if ToServer is true
- string {ChannelName} on which we'll send the message if ToServer is true
- string {Username} of whom we'll send the message if ToServer is false
- string {Content} content of the message you wish to send

This service will only work if your bot is added the server !

5- Android

a- Architecture

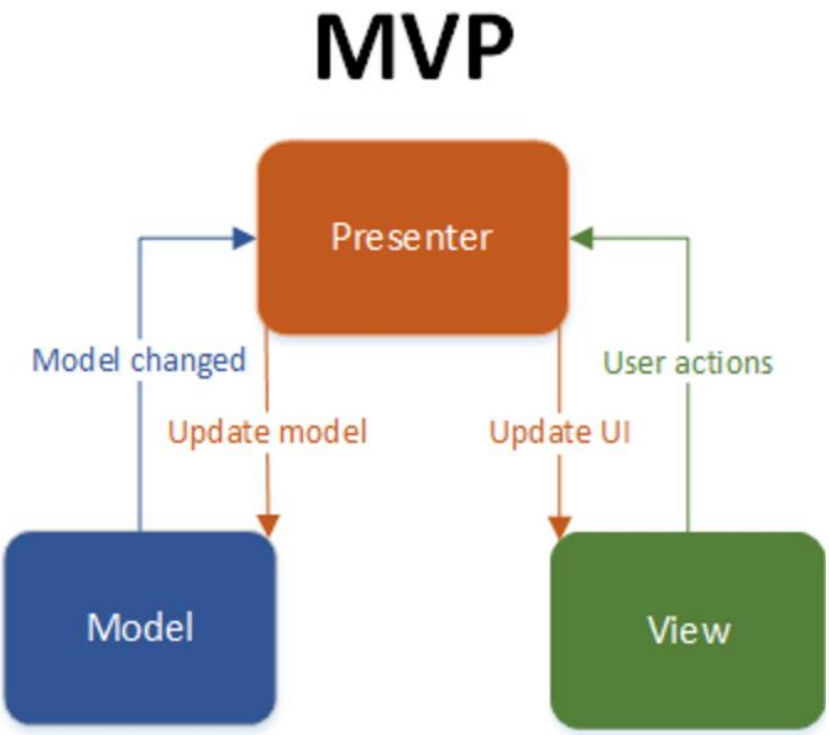
Since the Area has a mobile application part, we have therefore produced the latter with an MVP architecture (View View Model).

Source : https://riptutorial.com/fr/android/topic/4615/architecture-mvp

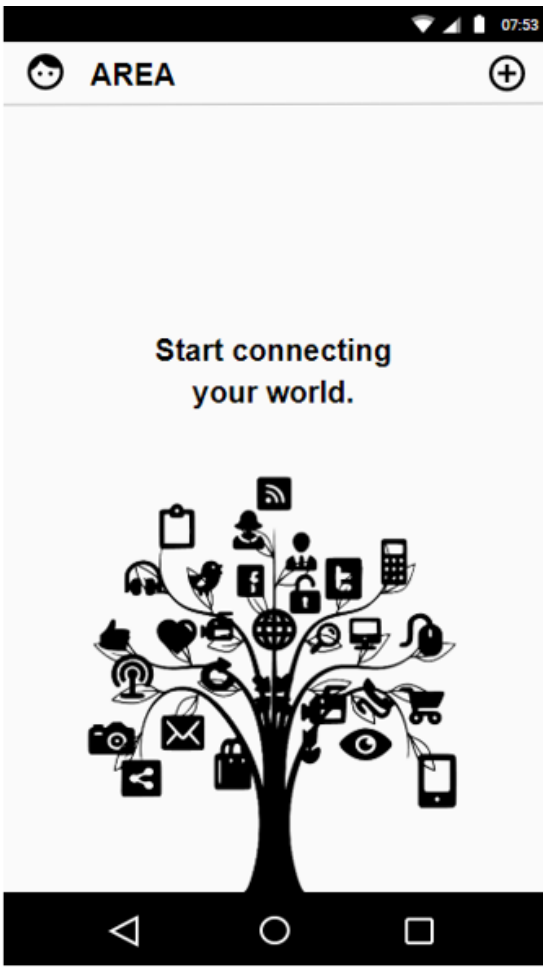
Model
In an application with a good layered architecture, this model would only be the gateway to the domain layer or business logic. See as the provider of the data we want to display in the view.

View
The view, usually implemented by an Activity or a Fragment, will contain a reference to the presenter. The only thing to do is to call a presenter method whenever there is an interface action.

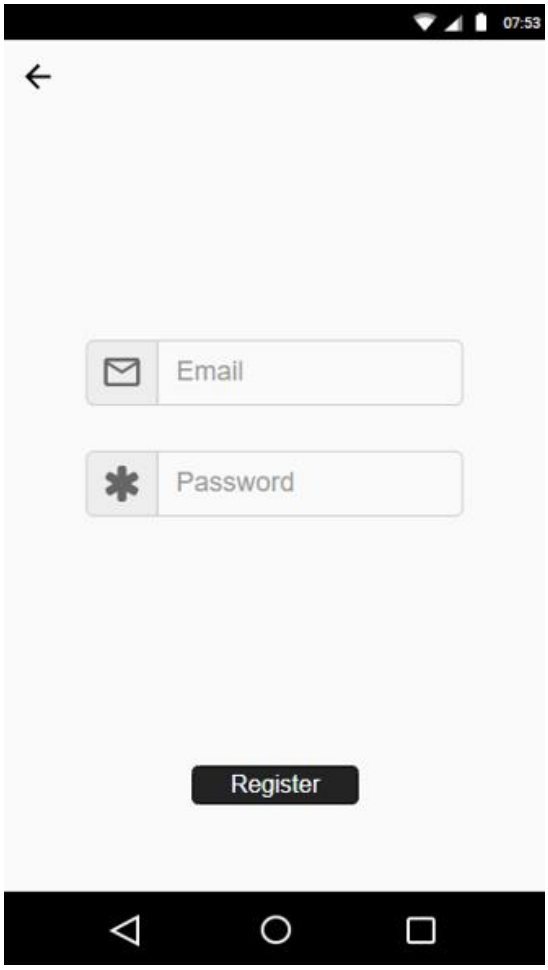
Presenter
The presenter is responsible for acting as an intermediary between View and Model. It retrieves the data from the model and returns it in view format. But unlike the typical MVC, it also decides what happens when you interact with the view.

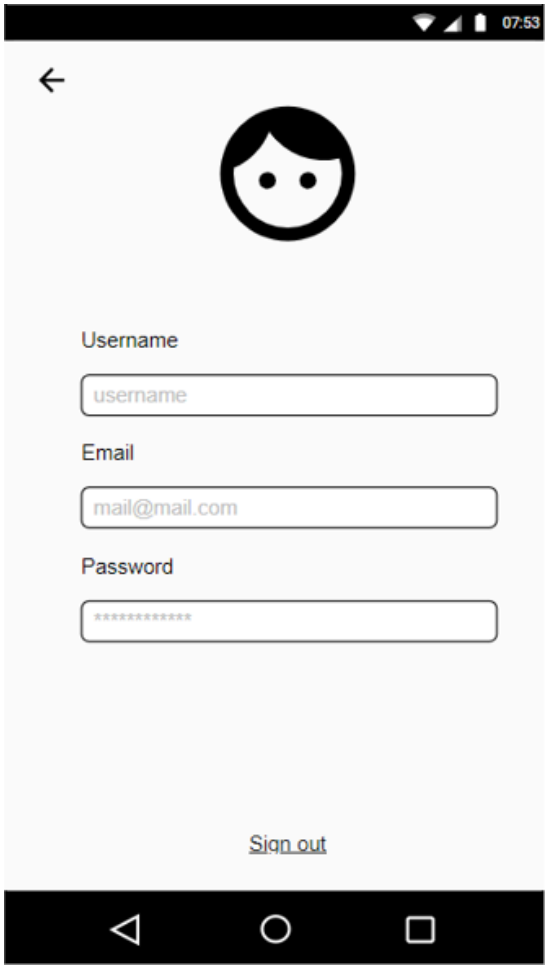


b- Mock up

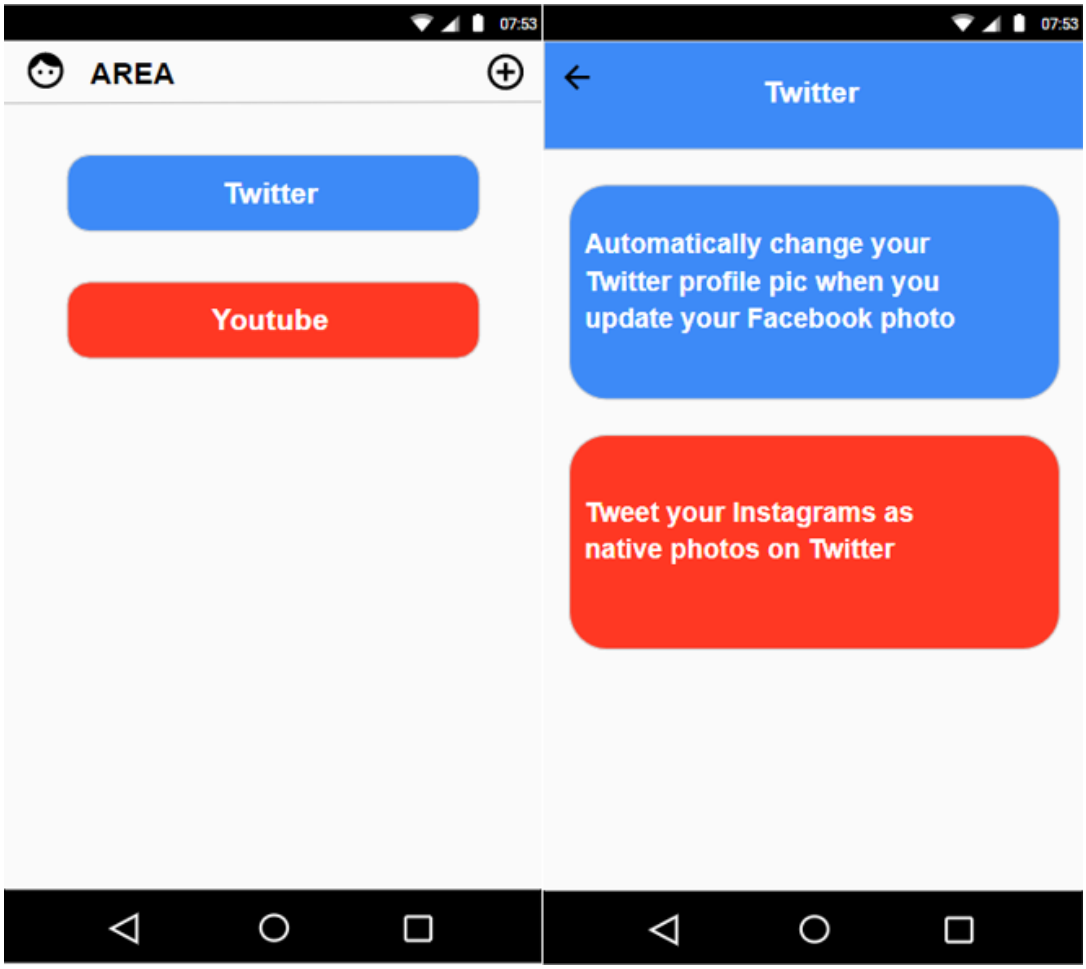


Here is the home view. The first one is when you're not connected.
The second one is when you're connected. You can add an area. And there is the list of your areas here.





Here is the login view and the register view. On the login view, you've already created an account. If not, you've to register with Username, mail and you're S3cr3t_Passw0rd !



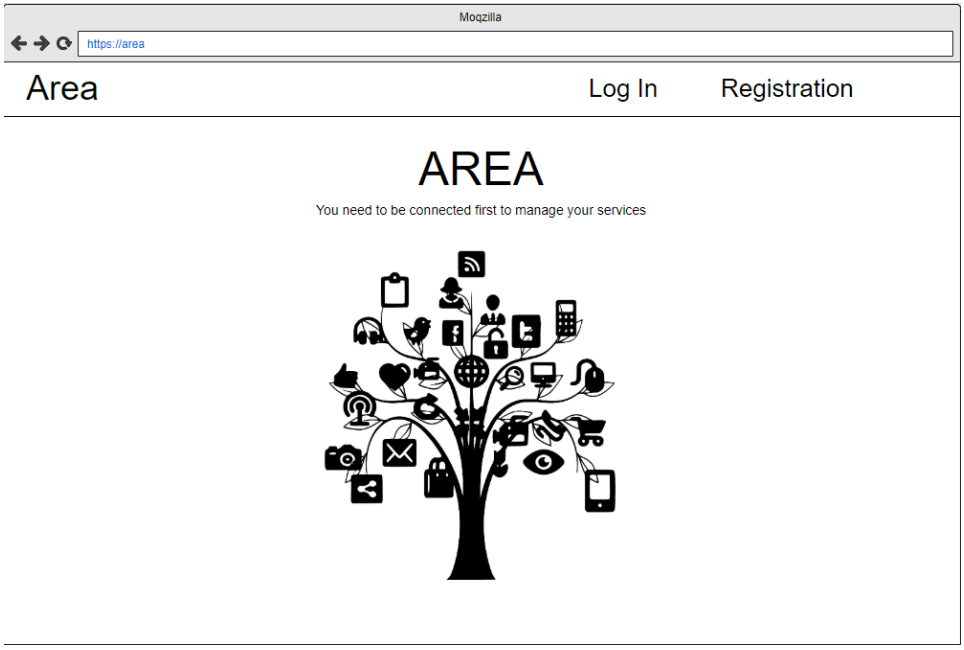
Here is the creation view. On the first one, you can choose your services. Here it's Twitter for action and Youtube for reaction. On the second page. You're customizing the Twitter parameters.

6- Web Client

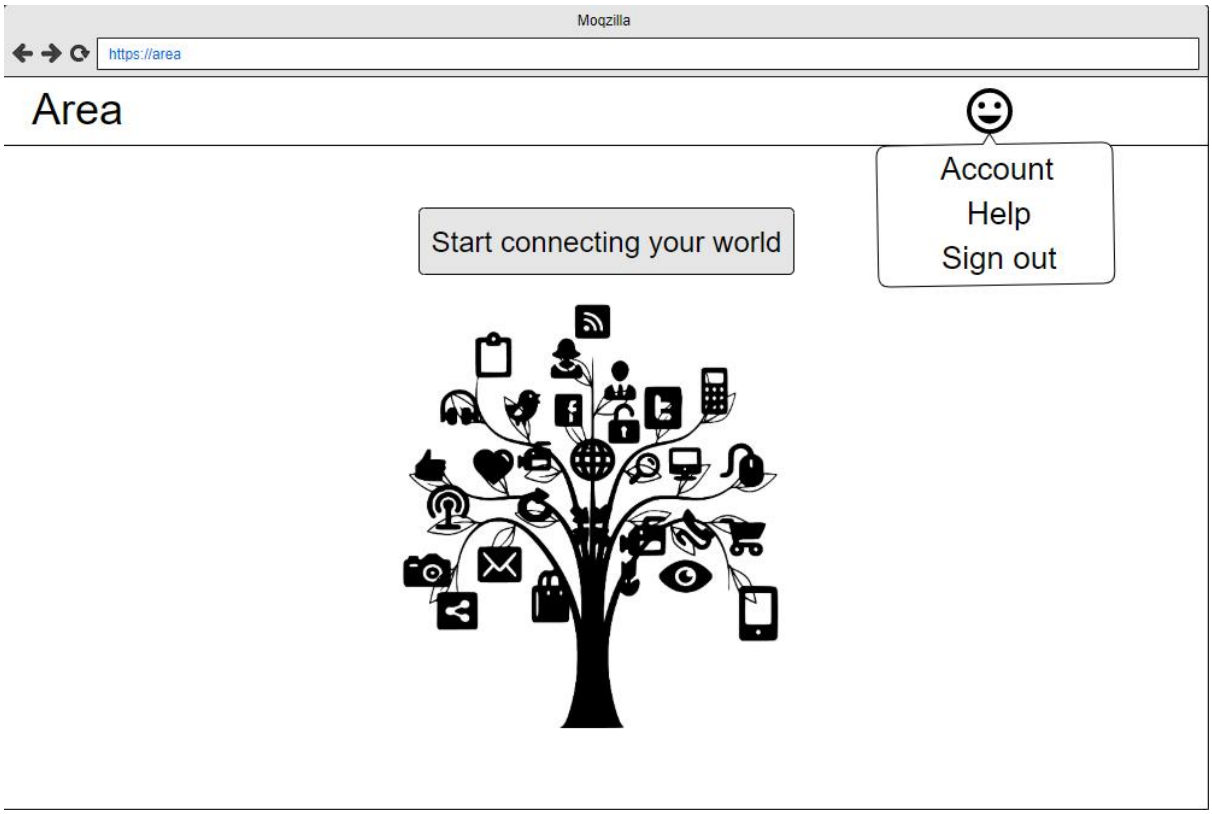
The web client is an MVP like the mobile client. But we have only the View part because the model and the presenter are on the API.
The view displays all the information and if it requires information then it calls the API before the load of the page.

1-Mockups

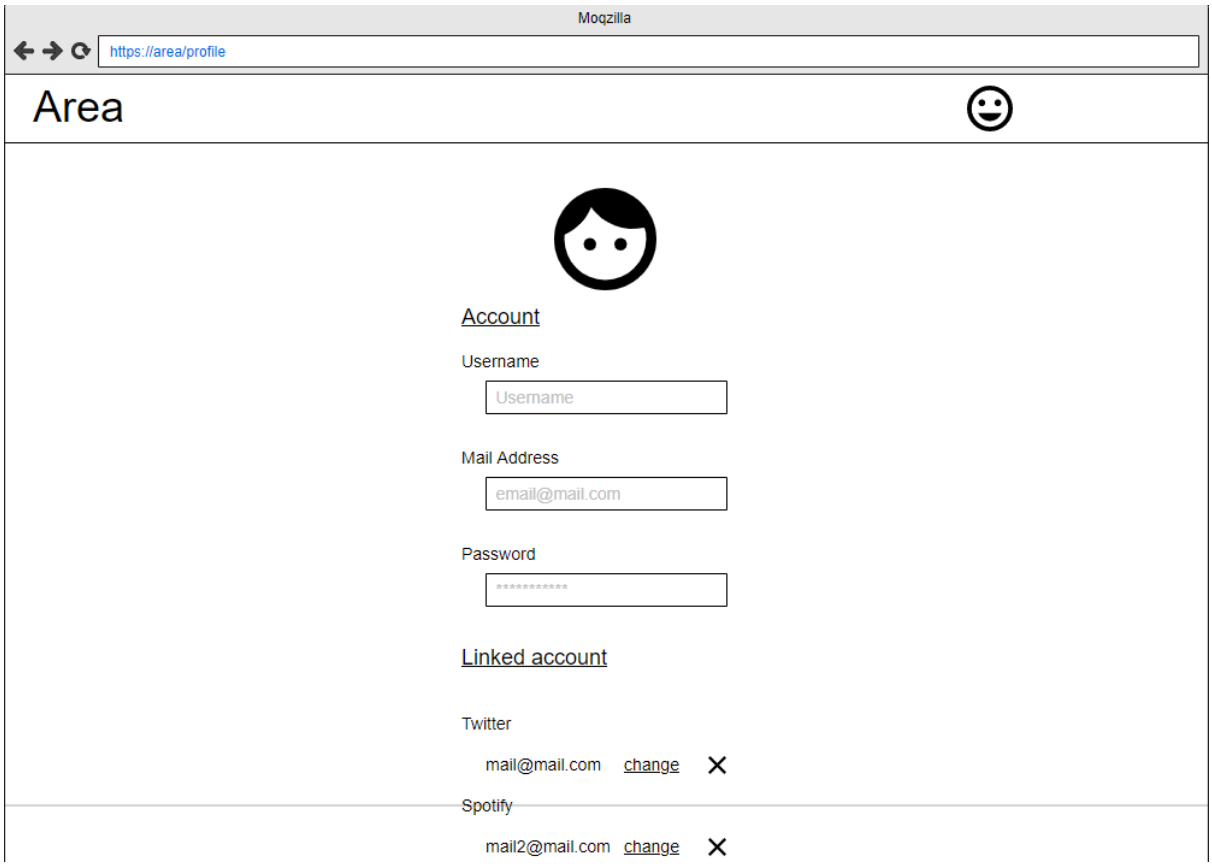
Here is the home page when you're not logged yet.



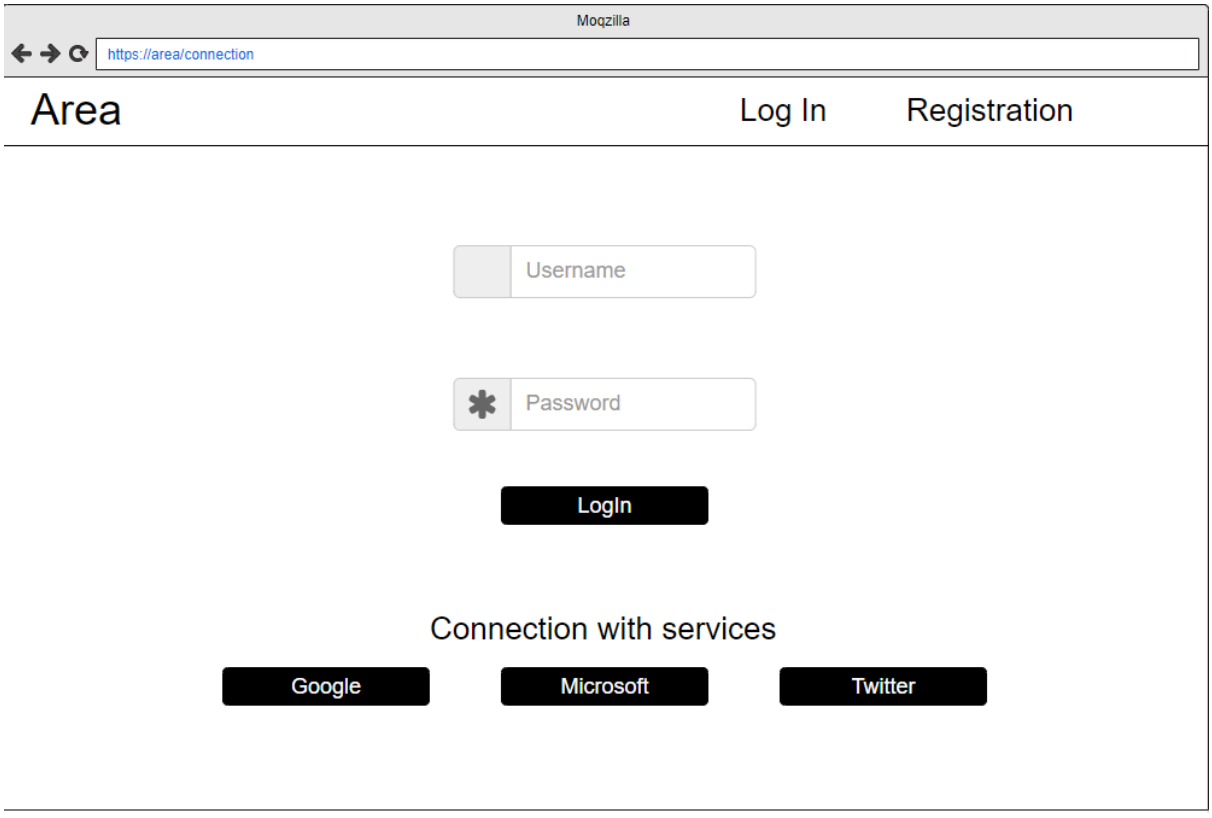
Here is the home page when you're logged.



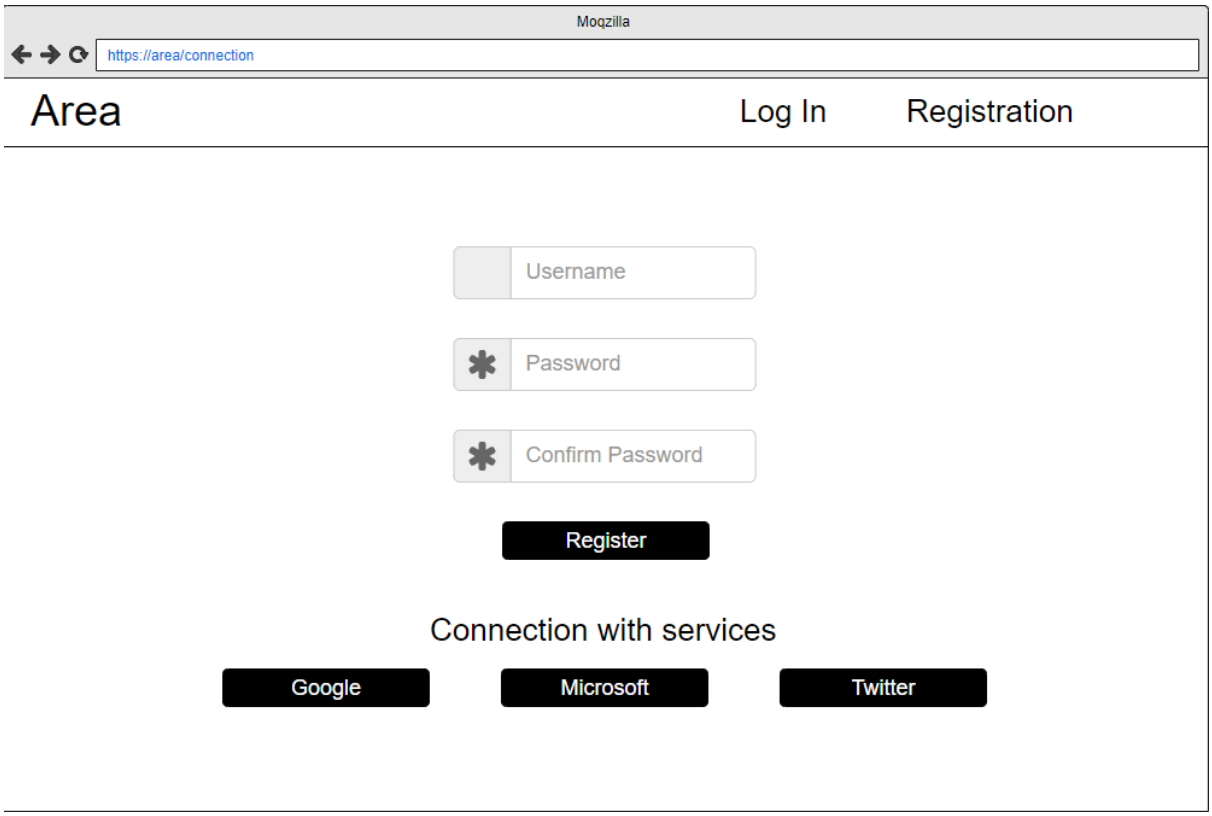
Here is the account page. You must be logged to access to this page.



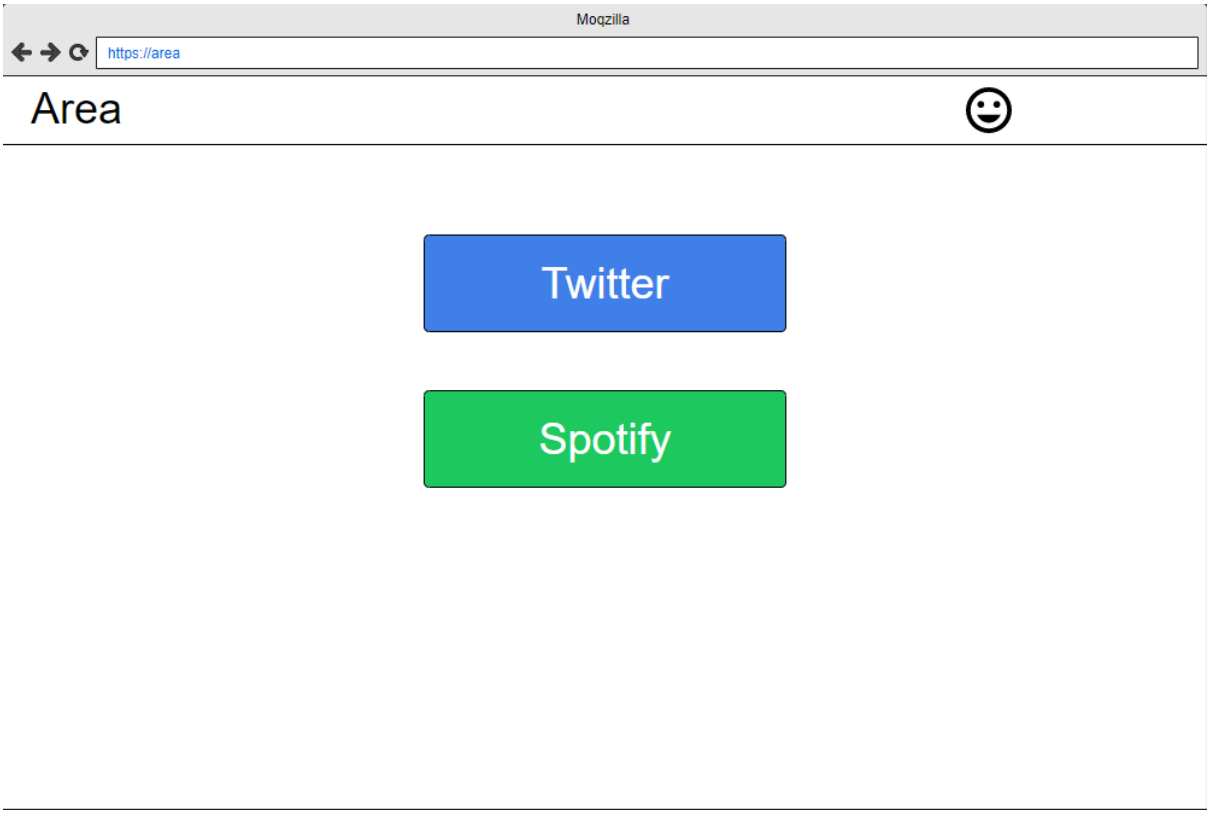
Here is the login page.



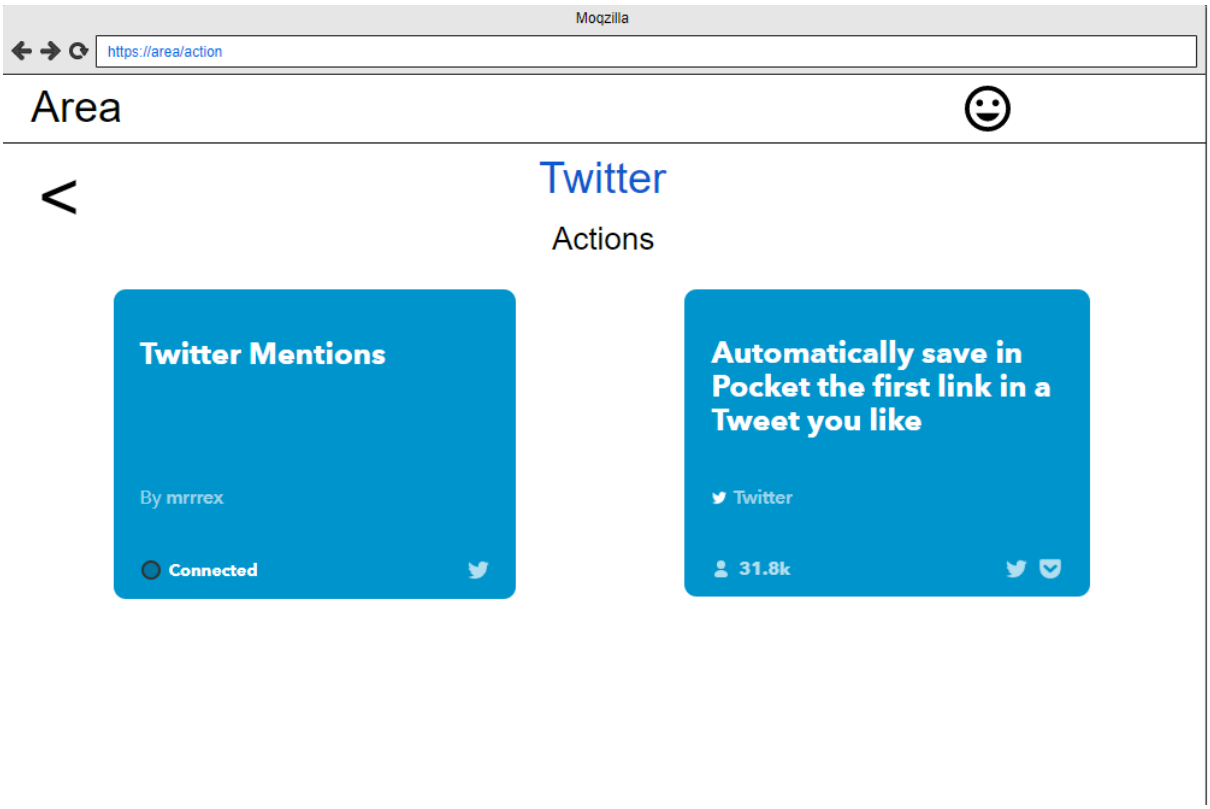
Here is the registration page.



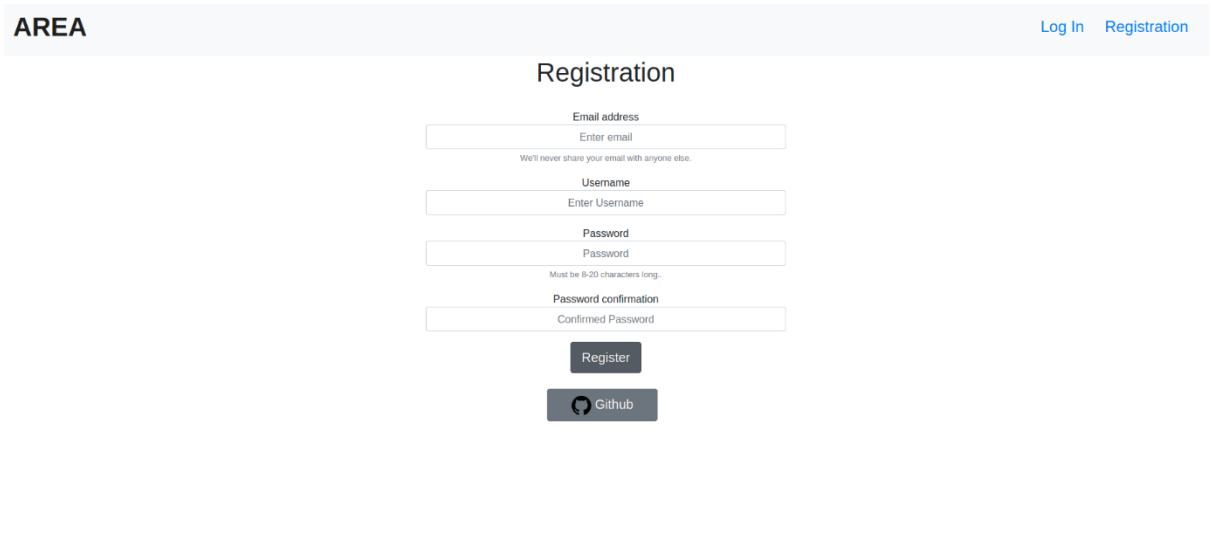
Here is the page to creation an AREA. You have chosen Twitter and Spotify.



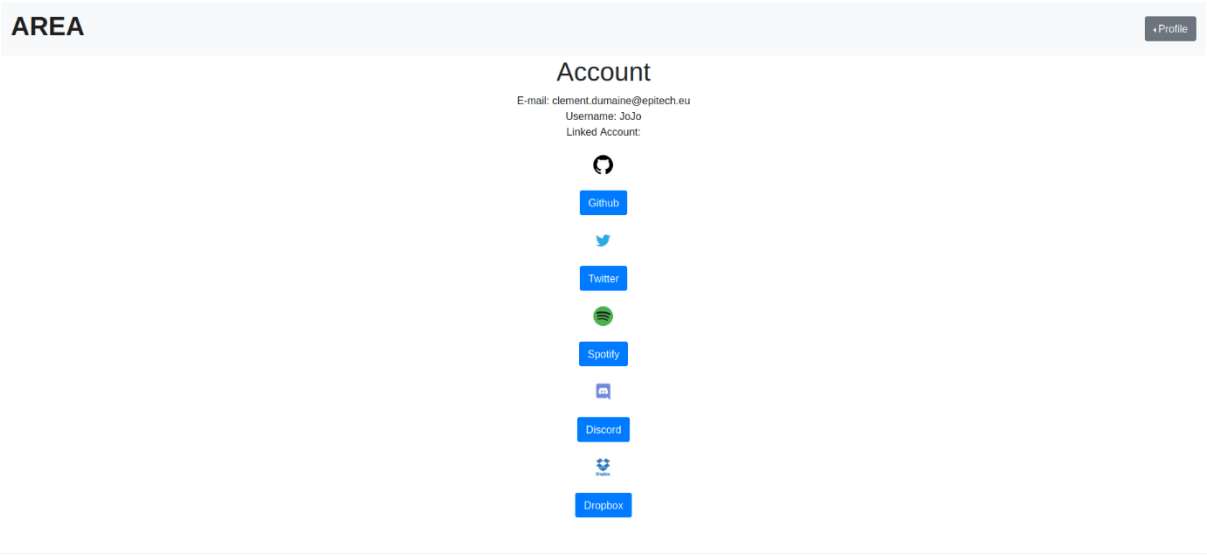
Here is the page to creation an Area. Here you custom the parameters of Twitter.



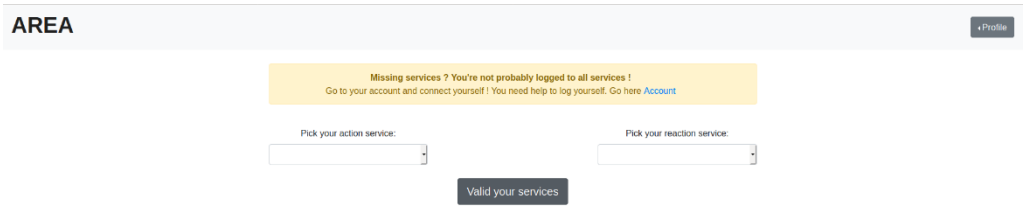
2 – In reality



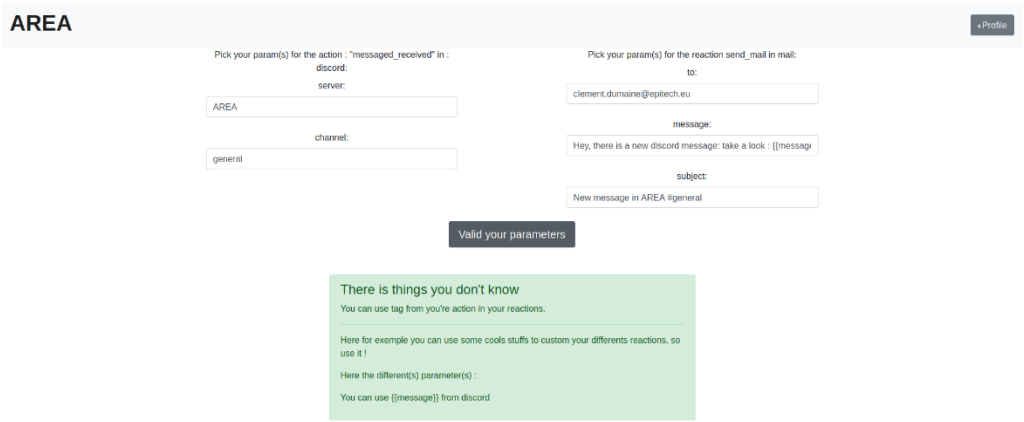
This this the actual page of registration. You must create an account for the first usage. If you don't want to create an account on our website. Just sign in with Github. We'll not get your password and neither you're mail address (Unless it is public). We only get your name.



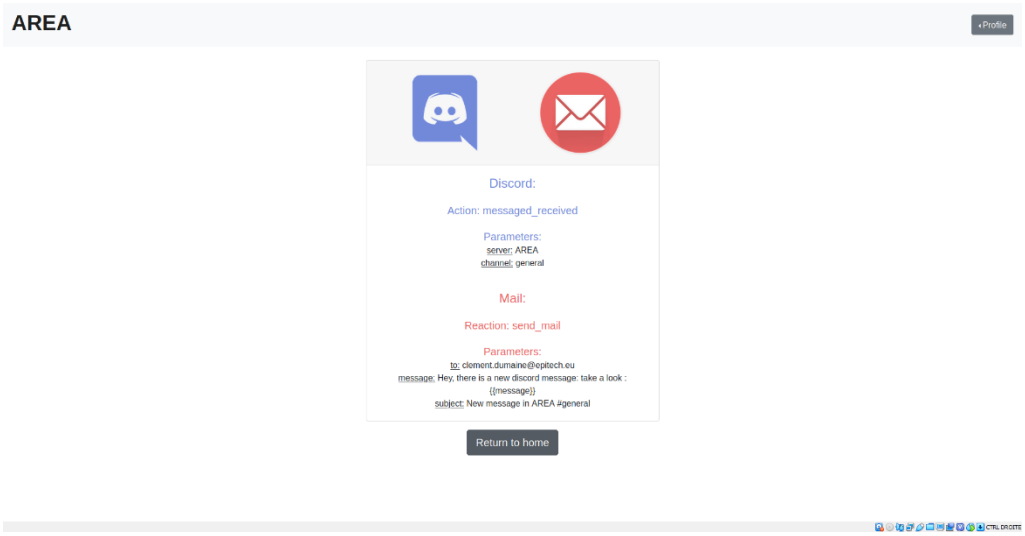
This the actual page of your account. You can see your email and your username. You can also link your different accounts to use your AREA. If you're logged with Github. You'll automatically logged on the Account page. When you link your discord account. It's just added a bot to a server. So, you can do it again to change the server.
However, if you link your Spotify account, you can unsubscribe it or delete it. Business is business.



Here is the first to create an Area.



Here is the third page to create an Area



Here is a recapitulation of your Area!