# My Authentication Platform (V1)

## Aim of this project

- Do a NodeJS project
- Use SQL database
- LAMP / WAMP / MAMP
- PhpMyAdmin
- Create a system of aunthentication

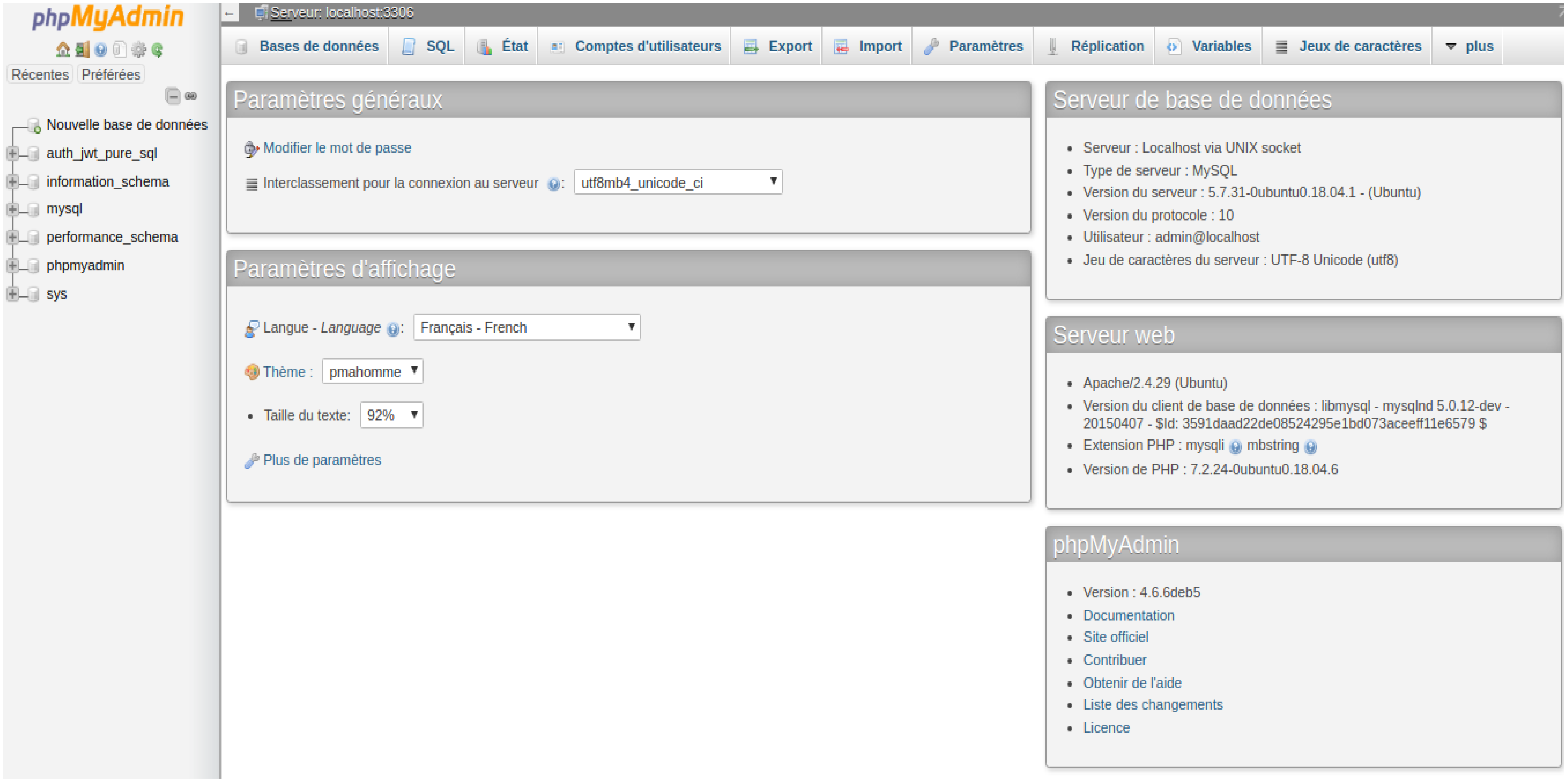
## First Step: Install LAMP / WAMP / MAMP

- Search what is LAMP / WAMP / MAMP
- Install the most appropriate for your system
- check if it's working properly

## Second Step: Install PhPMyAdmin

- As the title said, install PhPMyAdmin and test if it's work properly with your system.

-You should be able to see something as the screenshot below.



Paramètres généraux

Modifier le mot de passe

Interclassement pour la connexion au serveur: utf8mb4_unicode_ci

Paramètres d'affichage

Langue - *Language*: Français - French

Thème : pmahomme

- Taille du texte: 92%

Plus de paramètres

Serveur de base de données

- Serveur : Localhost via UNIX socket
- Type de serveur : MySQL
- Version du serveur : 5.7.31-0ubuntu0.18.04.1 - (Ubuntu)
- Version du protocole : 10
- Utilisateur : admin@localhost
- Jeu de caractères du serveur : UTF-8 Unicode (utf8)

Serveur web

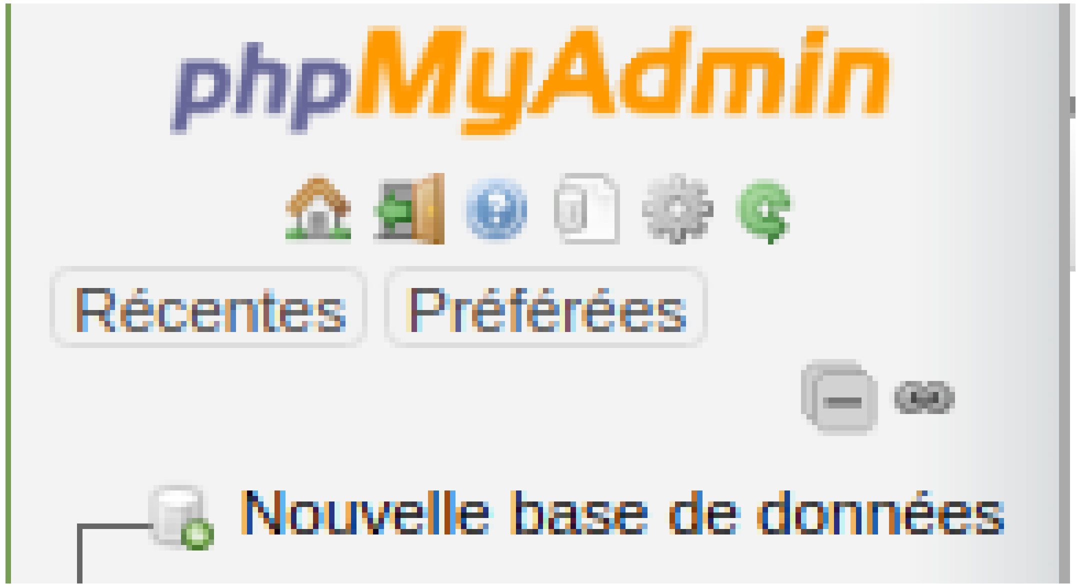
- Apache/2.4.29 (Ubuntu)
- Version du client de base de données : libmysql - mysqlnd 5.0.12-dev - 20150407 - $Id: 3591daad22de08524295e1bd073aceeff11e6579 $
- Extension PHP : mysqli mbstring
- Version de PHP : 7.2.24-0ubuntu0.18.04.6

phpMyAdmin

- Version : 4.6.6deb5
- Documentation
- Site officiel
- Contribuer
- Obtenir de l'aide
- Liste des changements
- Licence

## Third Step: Create a database

- On the PhPMyAdmin interface you will create a new bdd.

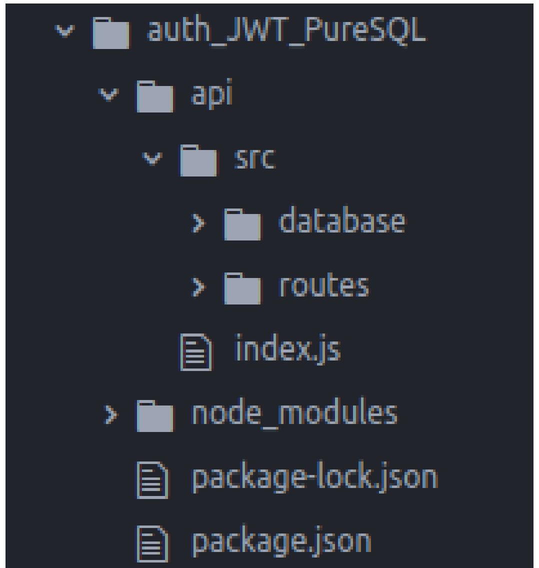


## Fourth Step: Create your api

- For this step help you of modules: express / mysql2

- We want you cut your project by modules.
That’s would mean you should have your 'api.js' file on the root of your project. But you should also have one module who contain your routes and one module who contain your implementation of mysql2

-Look all the slides of the Fourth Step !!



```
auth_JWT_PureSQL
  api
    src
      database
      routes
    index.js
  node_modules
  package-lock.json
  package.json
```

## Fourth Step: Create your api

- It's now time to create your API !
- Your API will have two routes: 'sign-up' & 'sign-in'

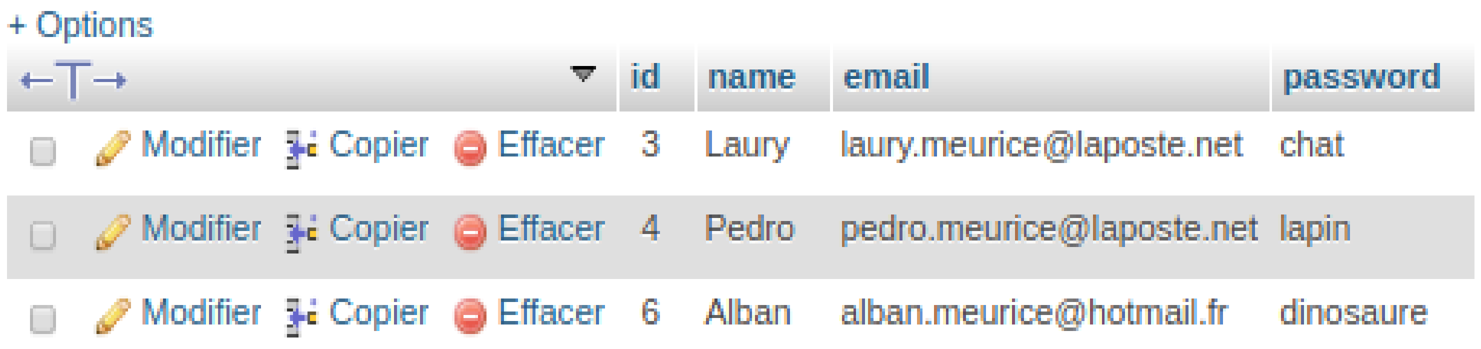
- 'sign-up' will register a new user on your database.

- 'sign-up' will have three fields: 'name', 'email' and 'password'. Use these fields for register your new user in database

- 'sign-in' will authenticate a registred user. For authenticate your user you will need the email & password of your user. If you can authenticate a user 'sign-in' will respond: "you are authenticated"
otherwise, if you can't authenticate it it will respond: "Sorry, we don't know this user"

# Fourth Step: Create your api

- Test if all works with Postman !
- In your Database we should see something like that :



+ Options

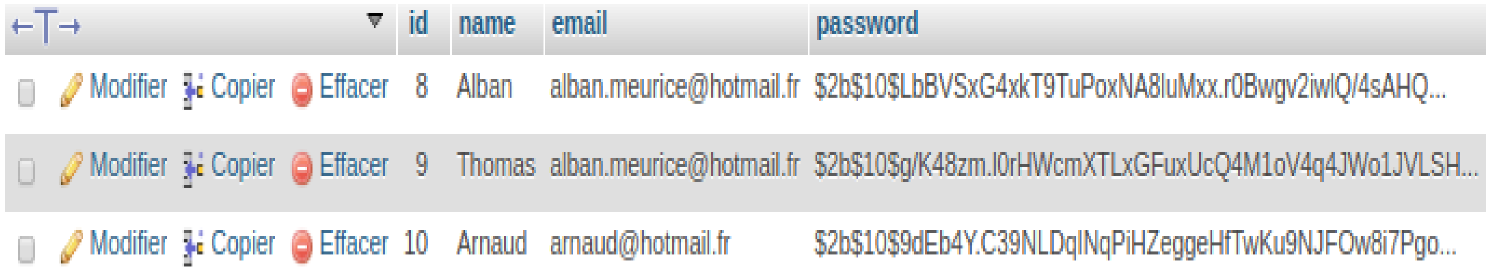
| | id | name | email | password |
|---|---|---|---|---|
| Modifier Copier Effacer | 3 | Laury | laury.meurice@laposte.net | chat |
| Modifier Copier Effacer | 4 | Pedro | pedro.meurice@laposte.net | lapin |
| Modifier Copier Effacer | 6 | Alban | alban.meurice@hotmail.fr | dinosaure |

# Fifth Step: Hash password !

- In this state we have a big security trouble ! Indeed we save the password of our user in clear on the database !

- Add bcrypt to your project.

- Save the "hash" password of your user and use also this "hash" for recognize your user when he sign-in !



| ←T→ | | | id | name | email | password |
|---|---|---|---|---|---|---|
| Modifier | Copier | Effacer | 8 | Alban | alban.meurice@hotmail.fr | $2b$10$LbBVSxG4xkT9TuPoxNA8luMxx.r0Bwgv2iwlQ/4sAHQ... |
| Modifier | Copier | Effacer | 9 | Thomas | alban.meurice@hotmail.fr | $2b$10$g/K48zm.l0rHWcmXTLxGFuxUcQ4M1oV4q4JWo1JVLSH... |
| Modifier | Copier | Effacer | 10 | Arnaud | arnaud@hotmail.fr | $2b$10$9dEb4Y.C39NLDqlNqPiHZeggeHfTwKu9NJFOw8i7Pgo... |

## My Authentication Platform (V2)

## So !

- From now, we have an API with two routes "sign-in" & "sign-up".

- It's will be time to add a front-end interface to our project!

## First Step: Add VueJS & Vue Bootstrap

- As the title said, start to create in a separate folder a new vueJS project. (You can use the vue-cli)

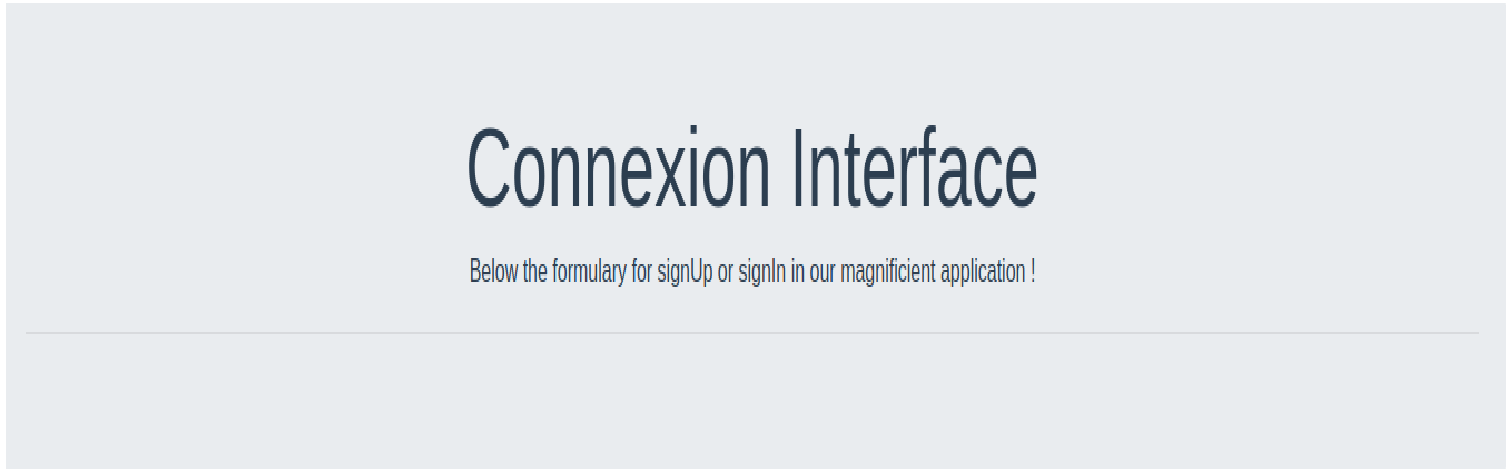
- Add Bootstrap-vue to your project.

- Add vue-router to your project

## Second Step: Home Page

- Your first view should be your 'Home' page.

- Your home page should look like this (use the jumbotron of bootstrap):


Connexion Interface

Below the formulary for signUp or signIn in our magnificient application !

## Third Step: Front - Form

- You will create a component 'FormHandler' this component will display a tabs-menu.

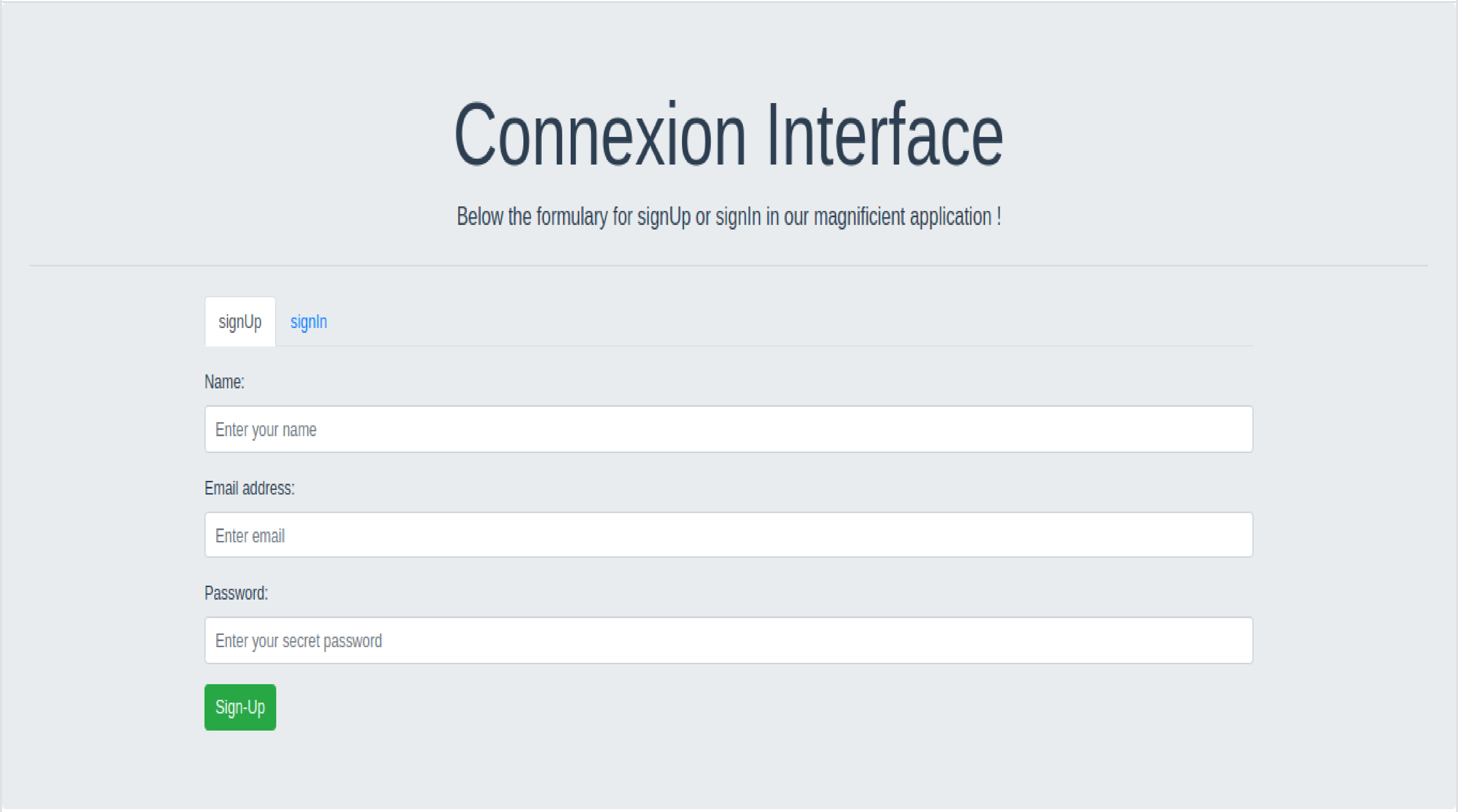
- Through this menu two form will be availaible. A "Sign-in" form & a "sign-up" form.

- These two forms will be two new components. Respectively a "SignUpForm" & a "SignInForm" component.

- The Sign-up form should have three fields [name; email; password].

-The Sign-In form should have two fields [email; password]

-Look at the two next slides, you should have something like that ...


Connexion Interface

Below the formulary for signUp or signIn in our magnificient application !

Name:

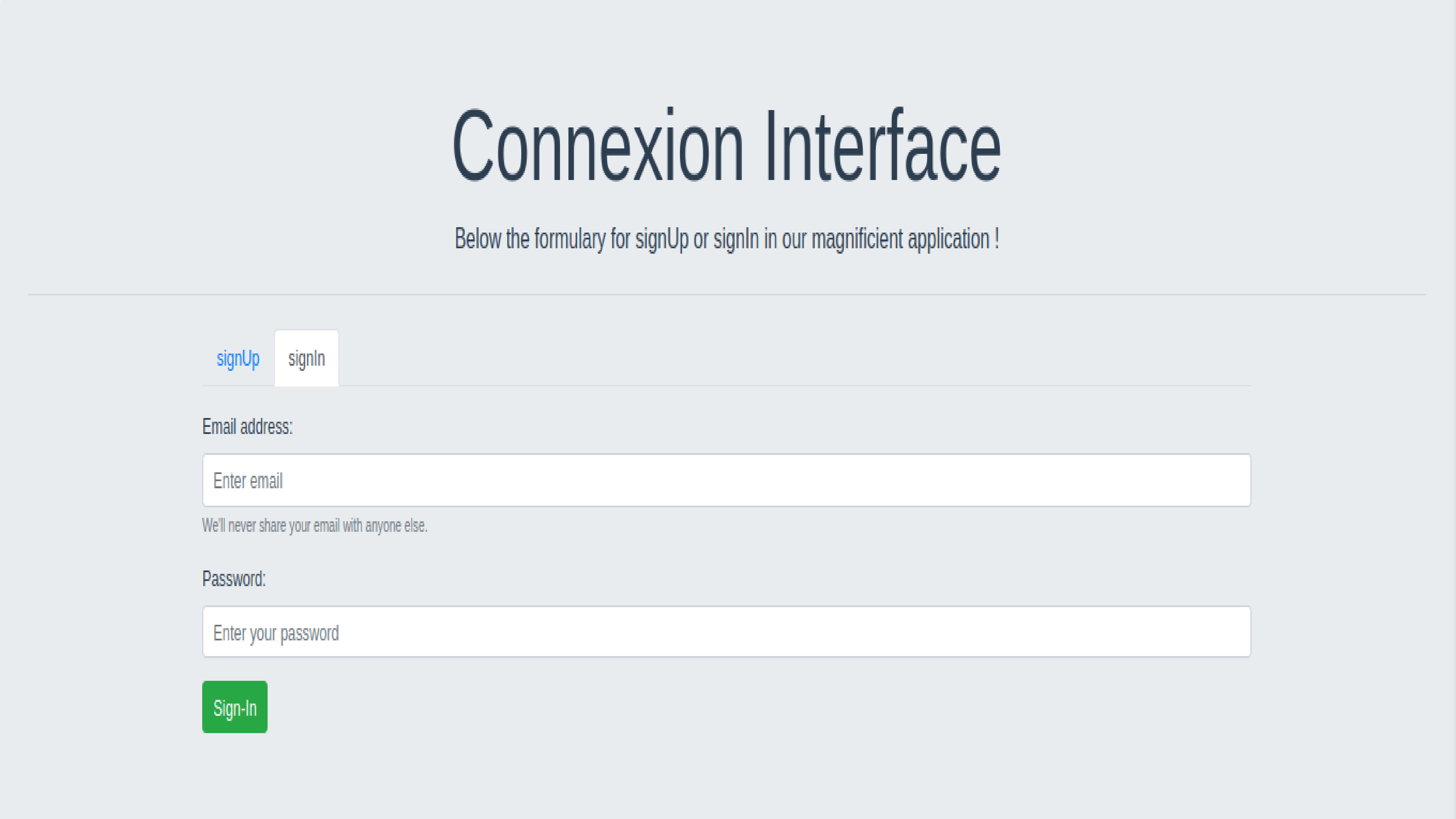
Enter your name

Email address:

Enter email

Password:

Enter your secret password



# Connexion Interface

Below the formulary for signUp or signIn in our magnificient application !

Email address:

Enter email

We'll never share your email with anyone else.

Password:

Enter your password

## Hold-on !

- As you may have noticed, our two forms have a button.

So ….. add a button to your forms.

## Fourth Step: And here comes the troubles.

And the justice-league can't help you... sorry :/

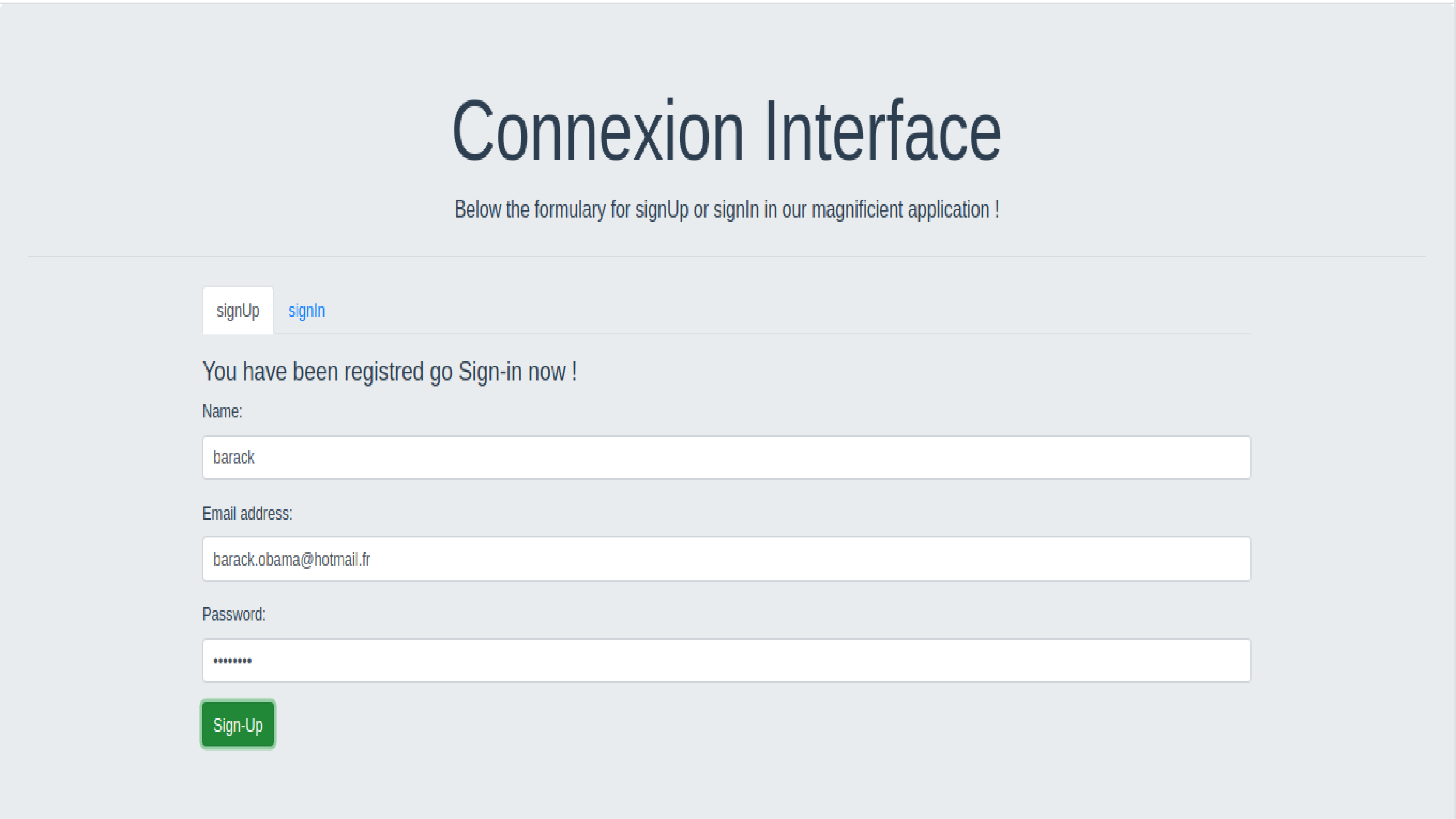
- The form you created earlier. From now the button they own will be link with a method.
These methods will use 'axios' for send a post request to your api.
(On the routes '/sign-up' for the SignUp form and on the routes '/sign-in' for the signIn form; seems obvious...)

- For the 'Sign-Up' component use your knowledge for display a message if the registration succeed or fail

- Create a new view 'Dashboard'. This view will be available on your web site at the route /dashboard

- For the 'Sign-in' component, redirect your user to the Dashboard view if the operation succeed. Otherwise display an error message.

-Look at the next slides; you should have something like that...



Connexion Interface

Below the formulary for signUp or signIn in our magnificient application !

You have been registred go Sign-in now !

Name:

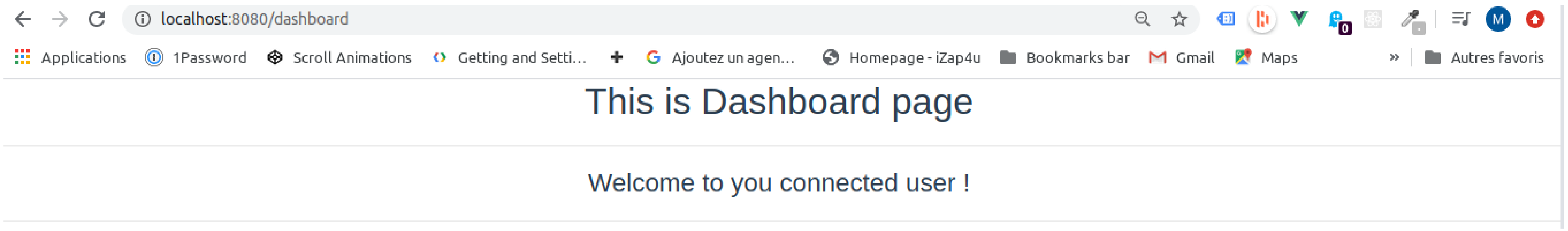
barack

Email address:

barack.obama@hotmail.fr

Password:

••••••••



This is Dashboard page

Welcome to you connected user !

## Now it's time to notice something

- Now, you have something, indeed little bit ugly BUT who could seem to be a decent authentication system.

- But in fact; No.

- Try to go to the page 'http://localhost:8080/dashboard' without been connected with one registered user.

- I guess you have seen the problem ?

## Fifth Step: More and more troubles ...

- First use 'Vuex' for create a store.

- Then do some research about "JWT token" or "Json Web Token".

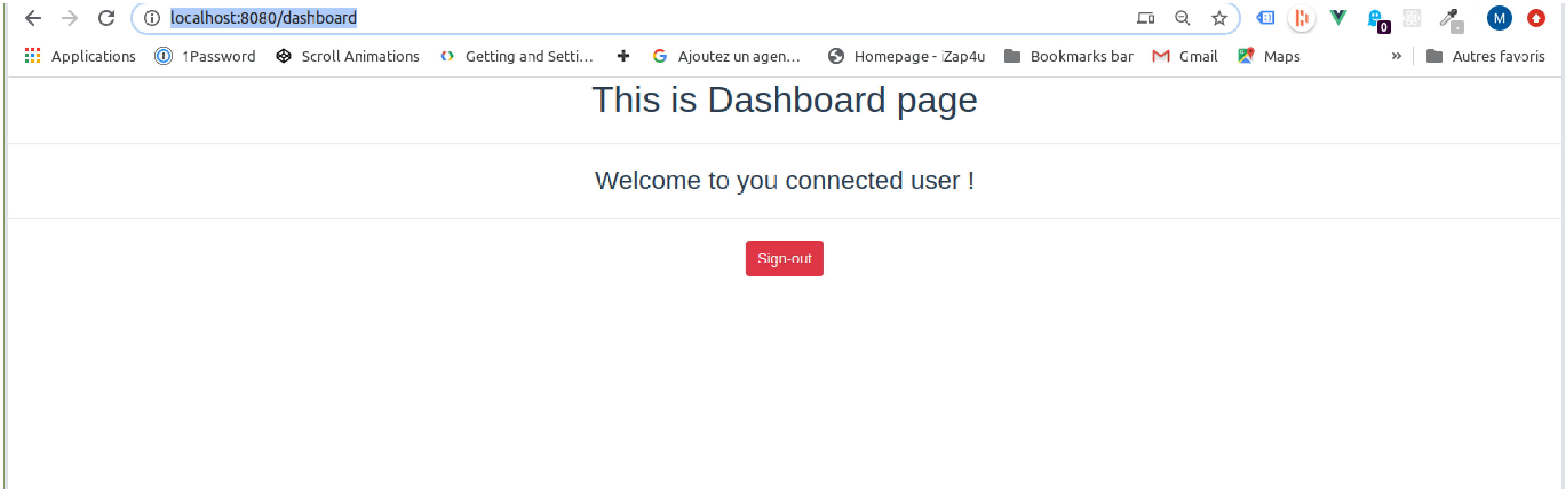
- After go to see this link: [https://www.digitalocean.com/community/tutorials/how-to-set-up-vue-js-authentication-and-route-handling-using-vue-router](https://www.digitalocean.com/community/tutorials/how-to-set-up-vue-js-authentication-and-route-handling-using-vue-router)
(Sometimes links are dead and you have to work; yeah I know... life is hard.
But keep hope & faith and do research about "protected routes vueJS")

-What we want from you is: if you are not connect and you try to go on a page where connection is required. You should be redirect to the "Home" page.

## Sixth Step:
## A (very) little one before the end.

- Add a "Sign-out" button to you /dashboard page. When your user click on it. Delete the token he own and redirect him to Home page.

- Maybe you saw. After connect when we arrive on /dashboard page. If we refresh, we are disconnected.... Well, take a look about something named: "createPersistedState"



localhost:8080/dashboard

This is Dashboard page

Welcome to you connected user !