

My Authentication Platform (V1)



Aim of this project

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



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.



phpMyAdmin

Récentes Préréfées

- Nouvelle base de données
- auth_jwt_pure_sql
- information_schema
- mysql
- performance_schema
- phpmyadmin
- sys

Serveur: localhost:3306

Bases de données SQL État Comptes d'utilisateurs Export Import Paramètres Réplication Variables Jeux de caractères plus

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

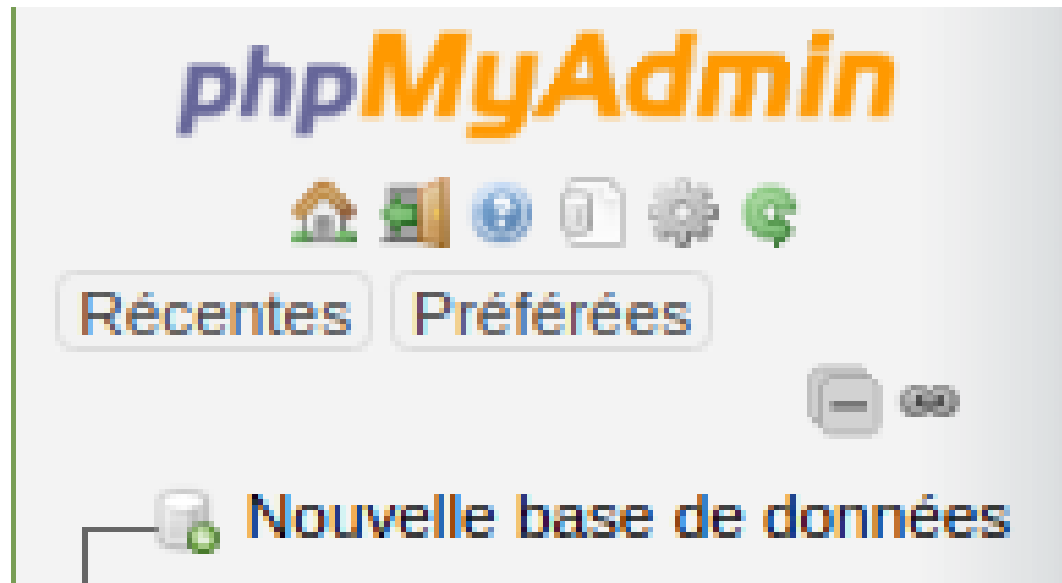
- Apache/2.4.29 (Ubuntu)
- Version du client de base de données : libmysql - mysqlnd 5.0.12-dev - 20150407 - \$Id: 3591daad22de08524295e1bd073aceeff11e6579 \$
- Extension PHP : mysqli mbstring
- Version de PHP : 7.2.24-0ubuntu0.18.04.6

phpMyAdmin

- Version : 4.6.6deb5
- [Documentation](#)
- [Site officiel](#)
- [Contribuer](#)
- [Obtenir de l'aide](#)
- [Liste des changements](#)
- [Licence](#)

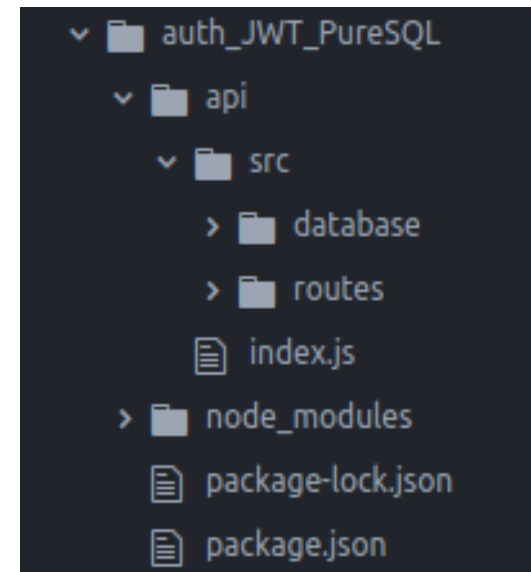
Third Step: Create a database

- 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 !!



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
<input type="checkbox"/>	 Modifier	 Copier	 Effacer	3	Laury	laury.meurice@laposte.net	chat
<input type="checkbox"/>	 Modifier	 Copier	 Effacer	4	Pedro	pedro.meurice@laposte.net	lapin
<input type="checkbox"/>	 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 !



				id	name	email	password
<input type="checkbox"/>	Modifier	Copier	Effacer	8	Alban	alban.meurice@hotmail.fr	\$2b\$10\$LbBVSxG4xkT9TuPoxNA8luMxx.r0Bwgv2iwlQ/4sAHQ...
<input type="checkbox"/>	Modifier	Copier	Effacer	9	Thomas	alban.meurice@hotmail.fr	\$2b\$10\$g/K48zm.l0rHWcmXTLxGFuxUcQ4M1oV4q4JWo1JVLSH...
<input type="checkbox"/>	Modifier	Copier	Effacer	10	Arnaud	arnaud@hotmail.fr	\$2b\$10\$9dEb4Y.C39NLDqINqPiHZeggeHfTwKu9NJFOw8i7Pgo...

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!



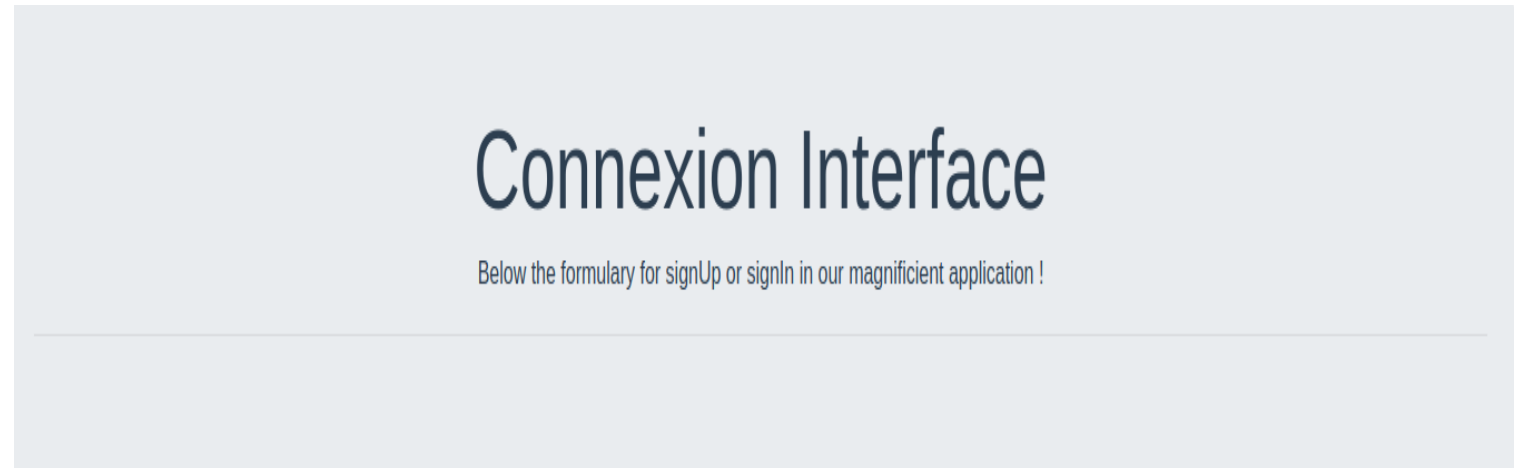
First Step: Add VueJS & Vue Bootstrap

- 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

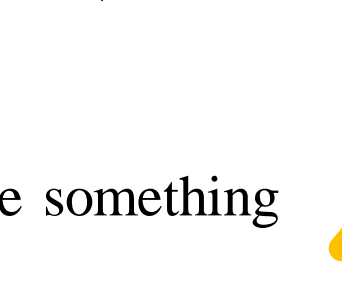


Second Step: Home Page

- Your first view should be your 'Home' page.
- Your home page should look like this (use the jumbotron of bootstrap):



Third Step: Front - Form

- You will create a component 'FormHandler' this component will display a tabs-menu.
 - Through this menu two form will be available. 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 magnificent application !

signUp

signIn

Name:

Email address:

Password:

Sign-Up

Connexion Interface

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

signUp

signIn

Email address:

We'll never share your email with anyone else.

Password:

Sign-In

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 magnificent application !

signUp

[signIn](#)

You have been registred go Sign-in now !

Name:

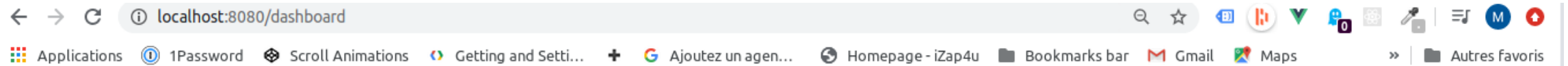
barack

Email address:

barack.obama@hotmail.fr

Password:

Sign-Up



This is Dashboard page

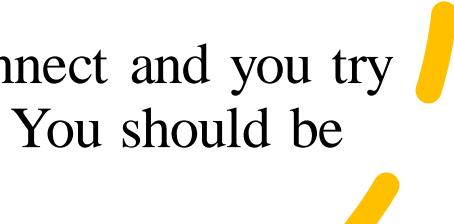
Welcome to you connected user !

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 connect 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>
(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"



This is Dashboard page

Welcome to you connected user !

Sign-out

My Authentication Platform (V3)



Vuelidate your project !

- <https://vuelidate.js.org/>
- Add validation with vuelidate on your forms (example next slide)



Connexion Interface

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

signUp

signIn

Name:

Enter your name

Veuillez renseigner votre nom.

Email address:

blabla

Vous devez fournir un email valide !

Password:

...

Votre mot de passe doit faire au moins 8 caractères.

Sign-Up

My user- friendly header

- Firstly, you will create a new component named 'ConnectedHeader'.
- This header will display the name of the connected user and the button 'sign-out'.

HELP:

- You will need to put the id and the name of your user in the payload part of your jwt token.
- In the SignIn component before redirect to /dashboard page. Decode the token you receipt and fill your store with the name and id of you user (contain by the token payload).

[signUp](#)[signIn](#)

Email address:

We'll never share your email with anyone else.

Password:

[Sign-In](#)[Dashboard](#)[Lenine](#)[Sign-Out](#)

Welcome to you connected user !

Create Contacts table

- We want each user of our website have the possibility to register some contacts from the dashboard page.
- We will save this contacts on the database in a new table name 'contacts'
- The contacts table will have fields: Id, name, email, user_affiliate



Create new route

- In your server create a route (POST) 'add-new-contact'. This route will receipt three data through the request body: name, email, id_user_affiliate

id_user_affiliate will save the user id who the contact belongs.

- All seems a bit misty ? Well, reading some documentation never killed (I think ...)
Look at that: <https://sql.sh/cours/jointures/inner-join>

- Use postman and try to register one contact.

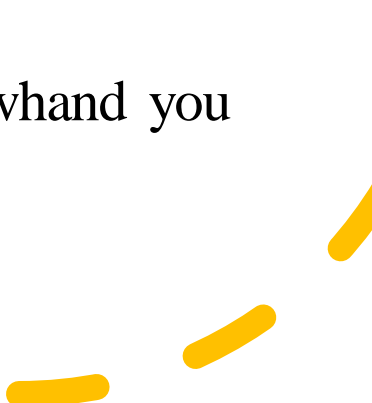


Create new route

- In your server create a route (GET) 'get-contacts'. This route will return you all the contacts registered in 'contacts' table.



Update Dashboard !

- Add a tabs to the Dashboard. We will have two tab: 'List-contacts' and 'add-contact'.
 - Create components 'AddContactForm' and 'ContactList'
 - AddContactForm should call 'add-new-contact' route and add a contact in your database.
 - ContactList should call get your contacts from the store.
 - Your store will be fill with the contacts of the authenticate user when the user succeed to 'sign-in'.
 - The contacts on your store will be update whand you successfully add a contact in the component 'AddContactForm'
 - next slide show you what you should have.
- 

Welcome to you connected user !

[List-Contacts](#)

Add-Contact

Name:

Enter the name of your contact

Email address:

Enter the email of your contact

Add

Welcome to you connected user !

List-Contacts

[Add-Contact](#)

Name: LouisLeBrocanteur

Email: louis.labrocante@yahoo.fr

Name: BruceWayne

Email: iam.batman@batsecret.us

Update route get-contacts

- From now the route 'get-contacts' will take an argument in parameter.

The route will become /get-contacts/:id

- The argument 'id' will be the id of the connected user.

- You will have to modificate your sql request for return only the contacts who belongs to the connected user.
(And not all the contacts from the 'contacts' table. As for now).

- don't forget the inner-join

! <https://sql.sh/cours/jointures/inner-join>



Middleware

- Add a middleware to your route 'sign-up'. The middleware will check if there is no user with the same name when a new-user try to be registered.
- If a user with the name your new-user want to use already exist, the route 'sign-up' should answer with an error message.

HELP:

<https://expressjs.com/fr/guide/using-middleware.html>

- You will probably have to do other research....

Protect your routes !

- You have some routes as: 'add-new-contact' or 'get-contacts/:id' you want to protect.
 - Would mean, you don't want somebody without a token can use this routes.
 - If somebody without a valid token try to use your protected routes, he should get an error message.
 - For protect your route you will have to create a middleware.
 - This middleware will be use by the routes you want to protect.
 - This middleware will check the "Headers" of your request. In the Headers he will search your jwt-token.
 - After finding the JWT token, your middleware will 'verify' if the token is valid or not.
- 