



# Bitchest

DM 14 – PROJET FINAL

L'ECOLE MULTIMÉDIA - 2018

CHBADA YOUNES & NONG ERIC

# Programme/Rubriques

- ▶ Contexte Général
- ▶ Problématique
- ▶ Solution adoptée
- ▶ Conception et Architecture générale
- ▶ Réalisation
- ▶ Perspectives d'amélioration

# Contexte Général

- ▶ L'objectif est de développer l'interface d'administration de Bitchest pour chacun des profils utilisateurs: Administrateur et Client.
- ▶ Administrateur : Gérer leur donnée personnelle , gérer les clients: création/affichage/modification et suppression ainsi que la consultation des cours des crypto monnaies.
- ▶ Client : Gérer leur donnée personnelle, gérer le portefeuille ainsi que la consultation des cours des crypto monnaies.

# Problématique

## Contraintes

- ▶ L'application web est développée uniquement par les membres du binôme.
- ▶ Elle doit utiliser le framework Laravel côté serveur.
- ▶ Elle doit répondre aux demandes décrites dans le cahier des charges.

## Libertés

- ▶ Vous êtes libre d'utiliser les bibliothèques dont vous avez besoin côté front.
- ▶ Vous êtes libre d'utiliser les API et services tiers nécessaires.

# Solution adoptée



# Solution adoptée

Frontend



Angular

Backend



Laravel

# Solution adoptée

## Pour

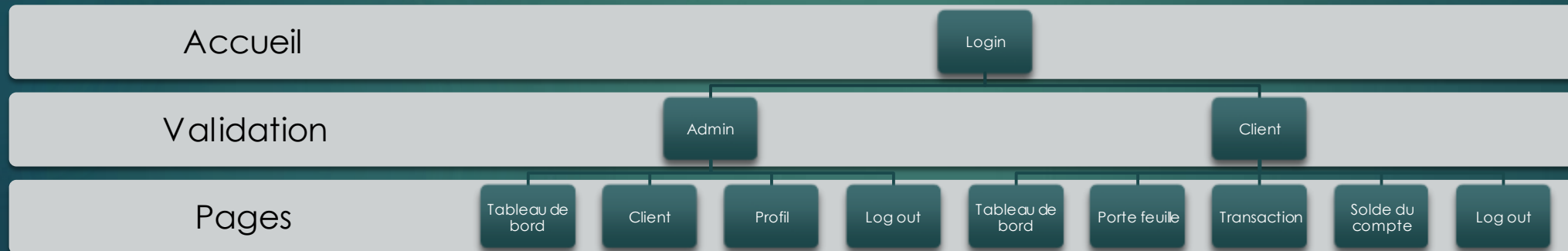
- ▶ L'application est plus simple à maintenir.
- ▶ L'absence de gestion d'état du client sur le serveur.
- ▶ Meilleure évolutivité et tolérance aux pannes.

## contre

- ▶ La nécessité pour le l'utilisateur de conserver localement les données nécessaires.

# Conception et Architecture générale

## Arborescence des écrans





# Zoning / wireframe

Connexion

A wireframe diagram of a login form titled "Connexion". The form is centered within a browser window frame. It consists of a "logo" label, an "Email" input field, a "Mot de passe" (Password) input field, and a submit button represented by an envelope icon.

```
graph TD; logo[logo] --- email[Email]; email --- password[Mot de passe]; password --- submit[Submit Button]
```

# Zoning / wireframe

Tableau de bord



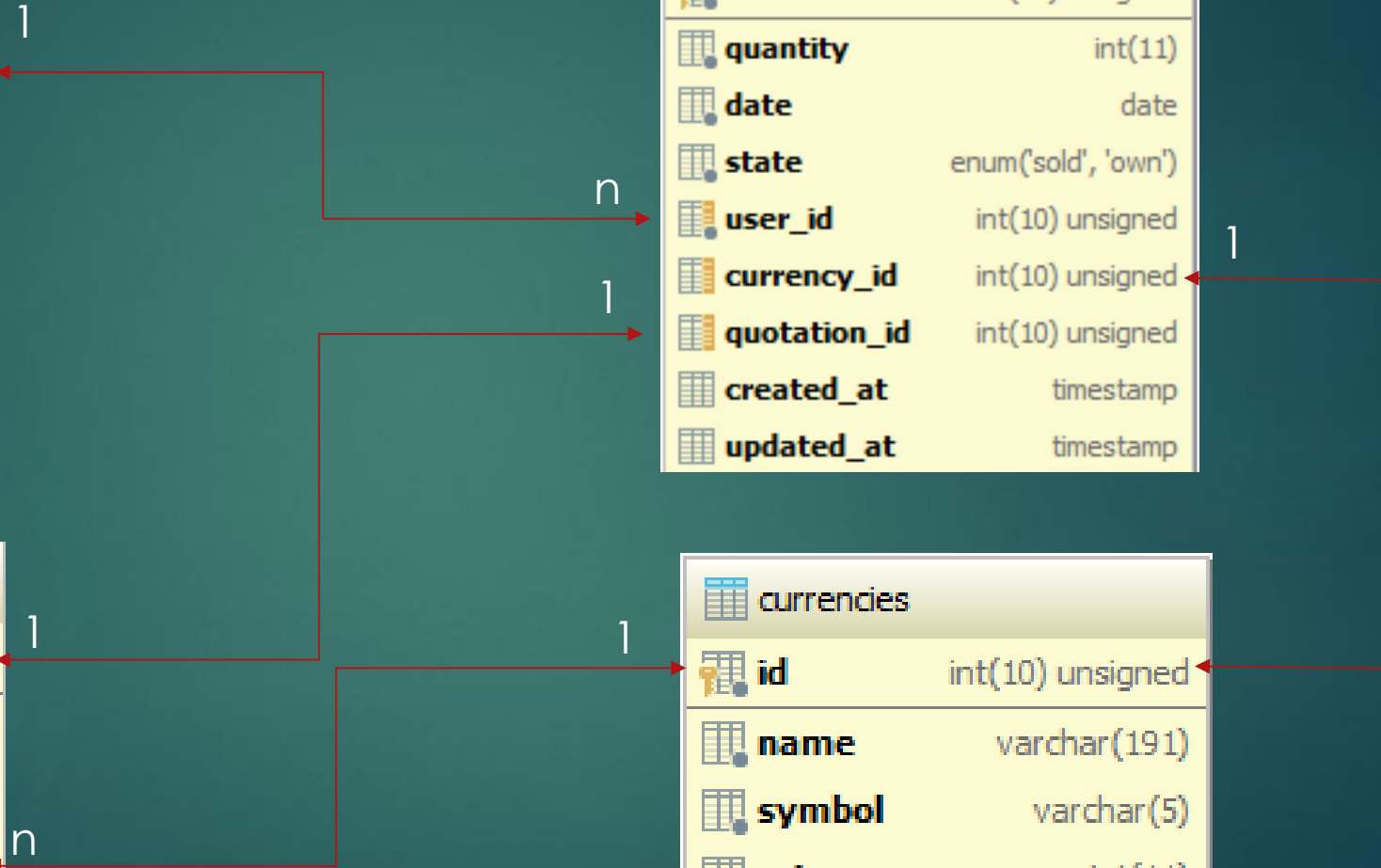
# Structure des tables

| users                 |                         |
|-----------------------|-------------------------|
| <b>id</b>             | int(10) unsigned        |
| <b>name</b>           | varchar(191)            |
| <b>email</b>          | varchar(191)            |
| <b>password</b>       | varchar(191)            |
| <b>is_verified</b>    | tinyint(1)              |
| <b>role</b>           | enum('admin', 'client') |
| <b>remember_token</b> | varchar(100)            |
| <b>created_at</b>     | timestamp               |
| <b>updated_at</b>     | timestamp               |

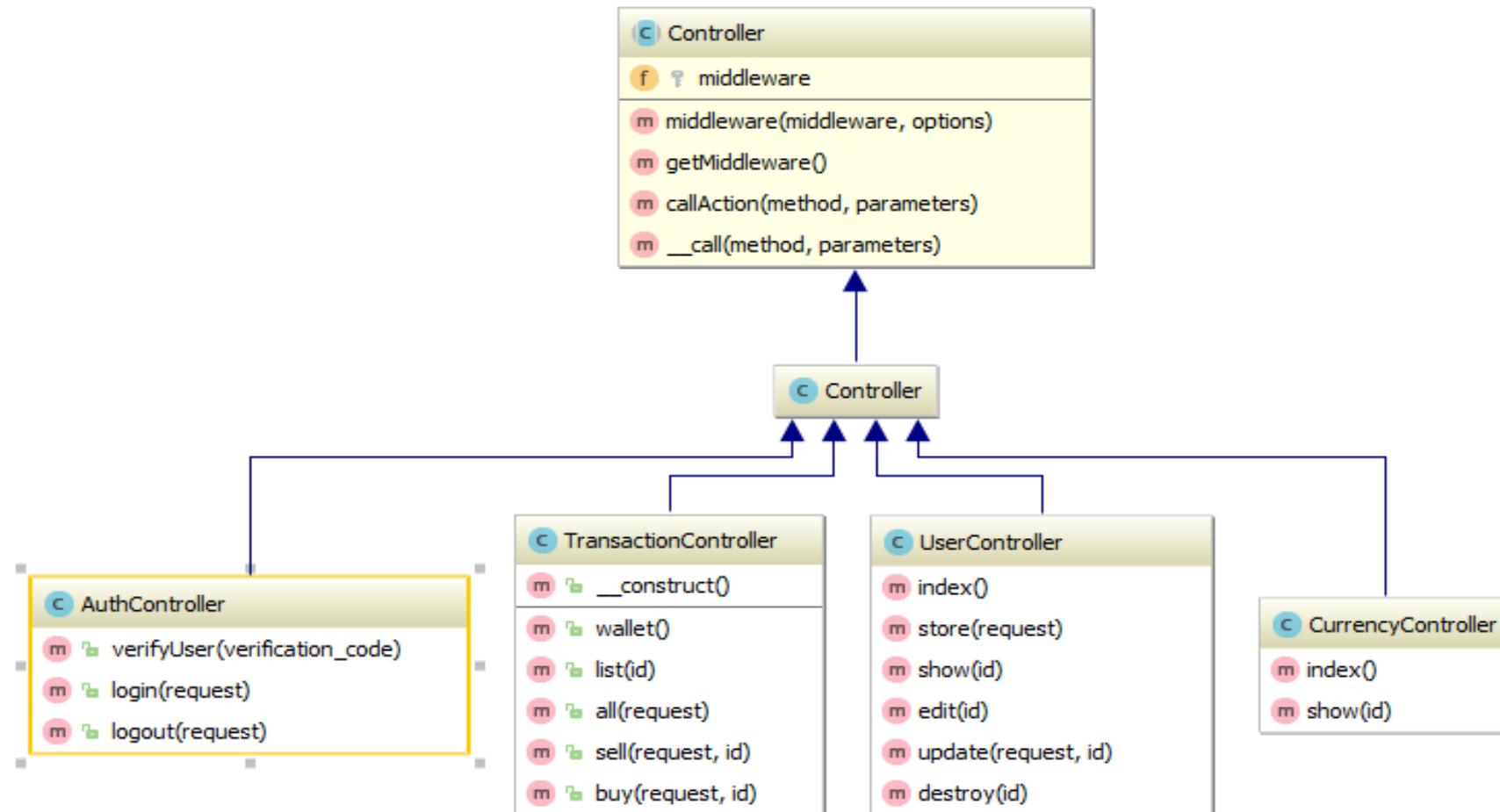
| transactions        |                     |
|---------------------|---------------------|
| <b>id</b>           | int(10) unsigned    |
| <b>quantity</b>     | int(11)             |
| <b>date</b>         | date                |
| <b>state</b>        | enum('sold', 'own') |
| <b>user_id</b>      | int(10) unsigned    |
| <b>currency_id</b>  | int(10) unsigned    |
| <b>quotation_id</b> | int(10) unsigned    |
| <b>created_at</b>   | timestamp           |
| <b>updated_at</b>   | timestamp           |

| quotations         |                  |
|--------------------|------------------|
| <b>id</b>          | int(10) unsigned |
| <b>date</b>        | date             |
| <b>rate</b>        | double(8,2)      |
| <b>currency_id</b> | int(10) unsigned |
| <b>created_at</b>  | timestamp        |
| <b>updated_at</b>  | timestamp        |

| currencies        |                  |
|-------------------|------------------|
| <b>id</b>         | int(10) unsigned |
| <b>name</b>       | varchar(191)     |
| <b>symbol</b>     | varchar(5)       |
| <b>price</b>      | int(11)          |
| <b>created_at</b> | timestamp        |
| <b>updated_at</b> | timestamp        |



# Diagramme des controllers



# Liste des endpoints

<http://localhost/BitChest/public/docs/index.html>

# Présentation de l'application

Bitchest

# Point de développement important

- Problématique :

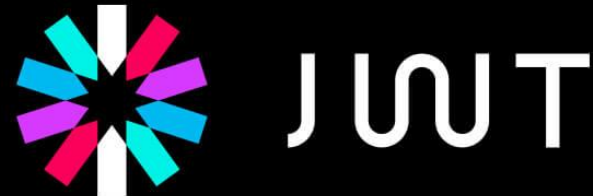
La gestion de la sécurité du dialogue entre le client et l'api



## Point de développement important

Solution adoptée :

Authentification de l'utilisateur via un  
Json Web Token unique.



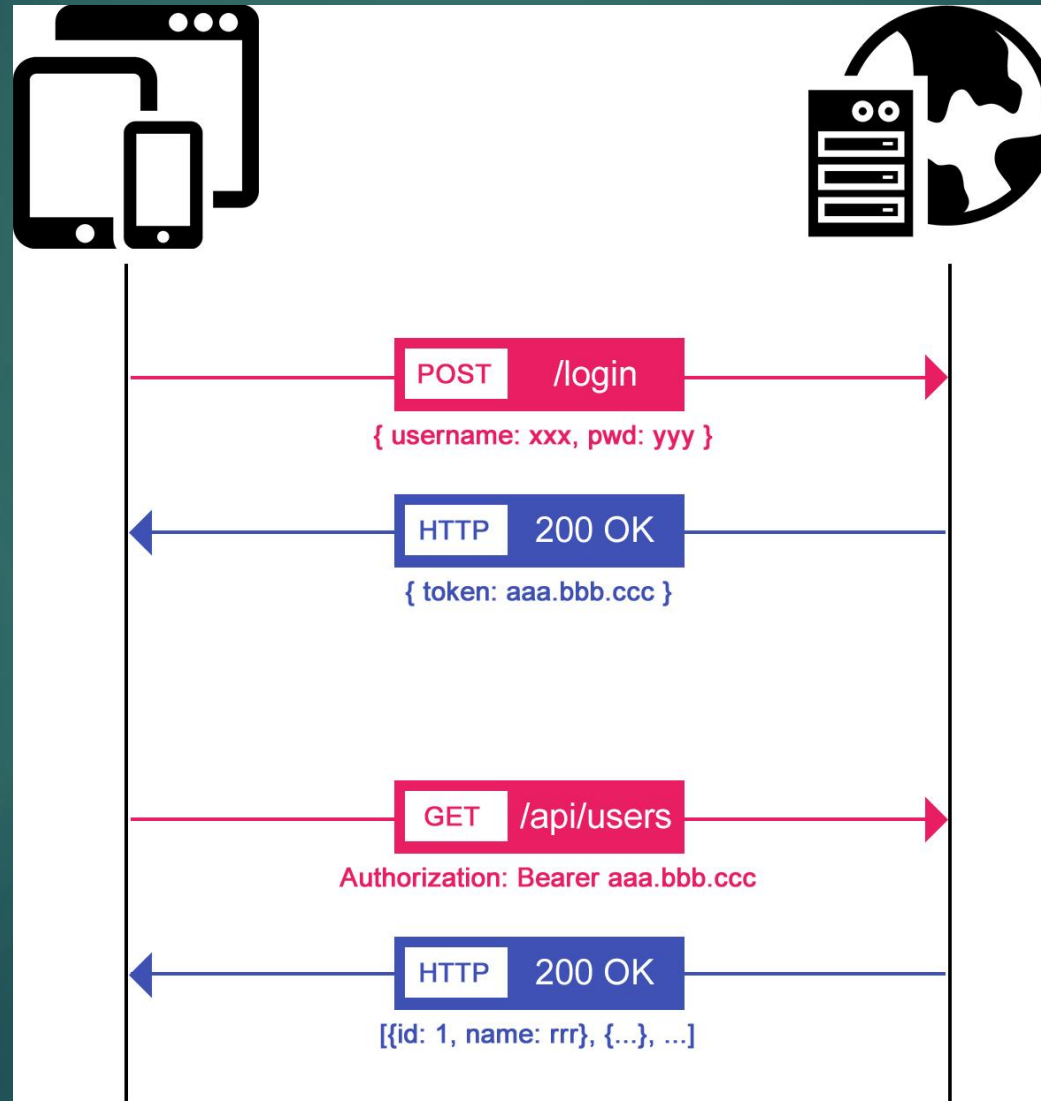


## Informations contenues dans le Token

# Clé de cryptage des données

| HEADER: ALGORITHM & TOKEN TYPE  |
|---|
| <pre>{   "alg": "HS256",   "typ": "JWT" }</pre>   |
| PAYLOAD: DATA   |
| <pre>{   "email": "1234567890",   "name": "John Doe",   "password": "root",   "role": "Admin" }</pre>   |
| VERIFY SIGNATURE  |
| <pre>HMACSHA256(   base64UrlEncode(header) + "." +   base64UrlEncode(payload),   <input type="text" value="secret"/> ) <input type="checkbox"/> secret base64 encoded</pre> |

# Utilisation du JWT



# Perspectives et propositions d'amélioration

- ▶ Personnalisations graphique de l'interface utilisateur
- ▶ Possibilités de développement de client mobile
- ▶ Possibilités d'achat de crypto-monnaies en utilisant des crypto-monnaies du porte-feuille.
- ▶ Possibilités pour l'utilisateur de crédit son compte par carte bancaire.



Avez-vous des questions ?