# Image result for intech png

# ITI.CryptosData



# 27/01/2020

Membres :

Sandya MADONNE

Alexandre PICARD

Dylan HERVE

Sommaire

[Présentation : 3](#_Toc31013785)

[Architecture du projet : 3](#_Toc31013786)

[Exercices : 5](#_Toc31013787)

[Users : 5](#_Toc31013788)

[Wallets : 5](#_Toc31013789)

[Transactions : 5](#_Toc31013790)

## Présentation :

Le projet a pour but d’être un portefeuille électronique pour de la crypto monnaie. Pour ceux qui connaisse Coinbase(<https://www.coinbase.com/>), c’est exactement la même chose.

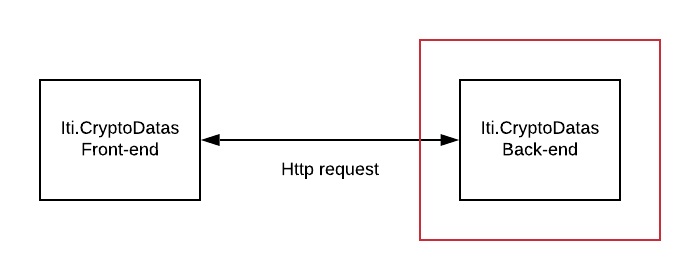
Ici, il s’agit de la partie back-end de l’application.

Vous pouvez :

* Créer un compte et vous enregistrer
* Ouvrir des portefeuilles pour chaques Cryptos
* Faire des transferts avec d’autres utilisateurs

## Architecture du projet :

Nous nous occuperons uniquement de la partie back-end, mais dans une architecture plus ou moins complète cela ressemblerait à cela :



Le front-end affichera seulement les éléments que le back lui enverra et le back-end récupéra les données et fera du traitement d’information.

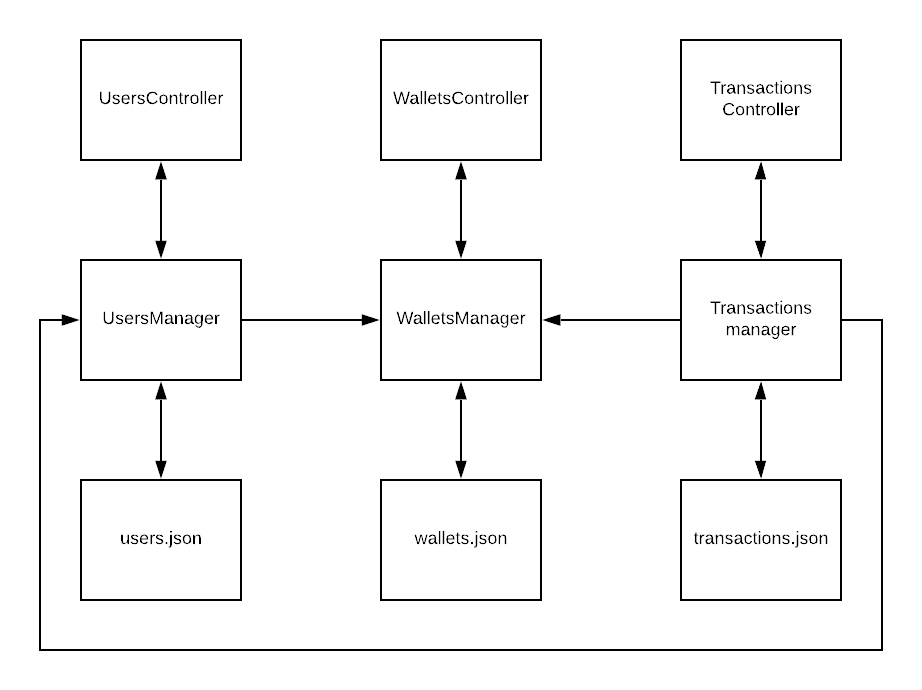
Pour que cela soit plus rapide, la base de données a été faite en JSON.

En ce qui concerne l’architecture même du back-end, nous avons 3 controllers qui vont recevoir les requetés et les envoyer après l’analyse des données aux managers.

Les managers quant à eux, vont s’occuper de tout ce qui est logique métier ainsi que de saisir les données dans la base de données :

* Création d’un utilisateur
* Création des portefeuilles
* Réapprovisionnement de fond
* Transfert de fond vars un autre utilisateur

Ci-dessous un diagramme avec les flèches indiquant les interactions entre les classes.



## Exercices :

### Users :

#### Création d’un utilisateur

Un utilisateur doit pouvoir créer son compte pour commencer le transfert d’argent.

Son compte sera ainsi enregistré dans la base de données.

L’username devra être unique, un utilisateur ne peut créer un compte avec 2 fois le même username.

Tests unitaires associés :

* can\_create\_an\_user\_and\_save\_in\_database
* return\_unproccessable\_entity\_for\_user\_already\_registered

#### Login de l’utilisateur

L’utilisateur doit être capable de se connecter à son compte via son username et mot de passe en regardant.

(Le mot de passe ne doit pas être enregistré en clair)

Tests unitaires associés :

* user\_can\_login

Dans un second temps nous allons utiliser les token qui serviront à identifier l’utilisateur lors de chaque requête.

A faire à la fois pour le login et register.

Tests unitaires associés :

* can\_get\_jwt\_token\_when\_login\_and\_register

#### Modification, suppression et récupération d’un utilisateur de l’utilisateur

Maintenant l’utilisateur doit pouvoir modifier son compte (firstname, lastname) et supprimer son compte de la base de données.

Pour des moyens de vérification nous utiliserons une méthode qui nous permettra de récupérer un compte. Cette méthode sera aussi à remplir.

Tests unitaires associés :

* can\_edit\_and\_remove\_an\_user

### Wallets :

#### Créer les wallets lorsqu’un utilisateur est créé

Lorsqu’un utilisateur créer son compte, des portefeuilles avec 0 comme fond sont créé pour chaque crypto que notre site peut offrir.

Tests unitaires associés :

* initialize\_wallet\_when\_a\_user\_is\_created

#### Pouvoir ajouter ou retirer de la crypto sur les portefeuilles

Ensuite nous devons pouvoir ajouter des fonds à ces portefeuilles.

Tests unitaires associés :

* add\_and\_remove\_fund

#### Supprimer les wallets d’un utilisateur

Pour finir, nous voulons supprimer tous les portefeuilles lorsqu’un utilisateur décide de supprimer son compte.

Tests unitaires associés :

* remove\_all\_wallets\_when\_an\_user\_is\_deleted

### Transactions :

#### Faire le transfert de crypto

Pour finir, l’étape bonus qui est un peu plus longue que les autres.

Nous voulons que l’utilisateur A puisse envoyer des fonds à l’utilisateur B.

Regardez bien le test unitaire pour savoir quelles méthodes doivent être remplie.

Tests unitaires associés :

* can\_do\_transaction\_to\_another\_user