AMT Projet - Plateforme De Trading

Bijelic Alen & Bogale Tegest

Professeur : Chapuis Bertil
Assistant : Gambin Dorian

13.12.2023

Table des matières

Plateforme de trading

Introduction

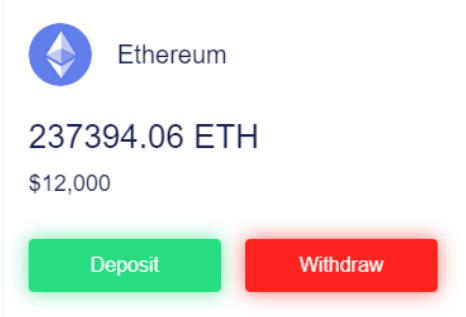
Pour ce dernier laboratoire, nous allons développer une solution d'échange de crypto-monnaies fictive.
La nouvelle technologie que nous allons utiliser est Hilla, qui nous permet de créer un Frontend en React avec les composants qu'elle propose et un Backend en Spring Boot, un framework que nous voulons explorer, au vu du grand nombre de postes dans l'industrie utilisant Spring Boot.
Nous allons aussi devoir implémenter une communication par message en utilisant JMS, notamment pour récupérer les prix des crypto-monnaies.

Description

En tant qu'utilisateur, je peux
- Accéder à la liste de crypto monnaies listées sur le site
- Créer un compte et me connecter
- Acheter et vendre des crypto monnaies
- Ajouter et retirer des fonds

En tant qu'admin, je peux
- Accéder à toutes les pages mais je ne peux pas acheter et vendre des crypto monnaies, ni ajouter et retirer des fonds
- Lister de nouvelles crypto monnaies sur le site

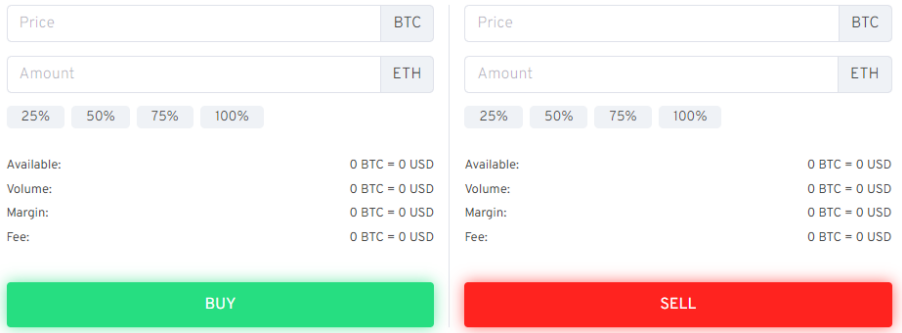
Le site disposera des pages suivantes:
- Page d'accueil
  - avec une liste des crypto monnaies listés sur notre site, leur prix actuel et un bouton permettant de rejoindre la page de trading correspondant à la crypto monnaie choisie pour acheter ou vendre
  - Exemple du composant affichant le logo, le nom, le prix, et un bouton pour acheter/vendre



- Page de login et register
- Page admin
  - Lister une nouvelle crypto-monnaie sur notre site. On aura accès à une liste de 17 crypto-monnaies et on pourra ajouter un certains nombre de ces 17 sur

notre site, Le but c'est d'en avoir au minimum 2 mais pas énormément, car l'accès aux API sont limités à 100'000 requêtes

- Page de trading
  - On peut acheter ou vendre des crypto-monnaies selon l'état du compte.
  - On peut vendre si on dispose d'un certain nombre de crypto monnaies
  - On peut acheter si on dispose d'assez de fond
  - Exemple du composant d'échange avec la possibilité d'entrer la quantité de cryptos à acheter ou le montant souhaité en dollar.



- Page de compte
  - permettant à un utilisateur d'ajouter ou de retirer ses fonds
  - ces fonds sont fictifs, on "retire" autant qu'on peut et on "ajoute" autant qu'on veut (1'000'000$ maximum)
  - les fonds seront en dollar
  - un graphique en fromage de l'état actuel de ses possessions, on utilisera chartjs (https://www.chartjs.org/)
  - Une liste simple des trades effectués (avec un filtre sur les cryptos - filtre facultatif si on a le temps)

Messaging

JMS sera utilisé pour récupérer les prix des crypto-monnaies à temps régulier (toutes les minutes, pour ne pas épuiser le plan gratuit de l'API). Les prix sont récupérés sur l'API de https://blockchair.com/ où nous avons un plan à 100'000 requêtes pendant un an grâce au GitHub Students Pack.

Technologies

Utilisation de **Hilla** (https://hilla.dev/) Frontend **React** et Backend **SpringBoot**
Utilisation de l'API blockchair (https://blockchair.com/) pour récupérer la liste de toutes les crypto-monnaies, ainsi que le prix de celles listées sur notre site.
Base de données Postgres et utilisation de Spring Data JPA.

Data model

Prix

- Id : *Int*
- Prix du marché en USD pour 1 unité de crypto : *Double*
- Date/Heure: *Date*

Crypto

- Id : *Int*
- Acronyme (pour bitcoin c'est BTC par exemple) : *String*
- Nom: *String*
- **Liste des prix : *ArrayList<Prix>***

User

- UUID: *UUID*
- Prénom: *String*
- Nom: *String*
- Email: *String*
- Mot de passe: *String*
- Fond: *Double*
- Admin: Boolean

Trade

- **User**
- **Prix**
- Date/Heure: *Date*

UML

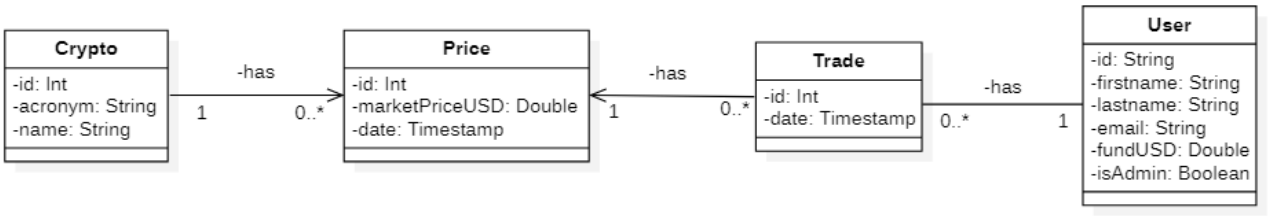
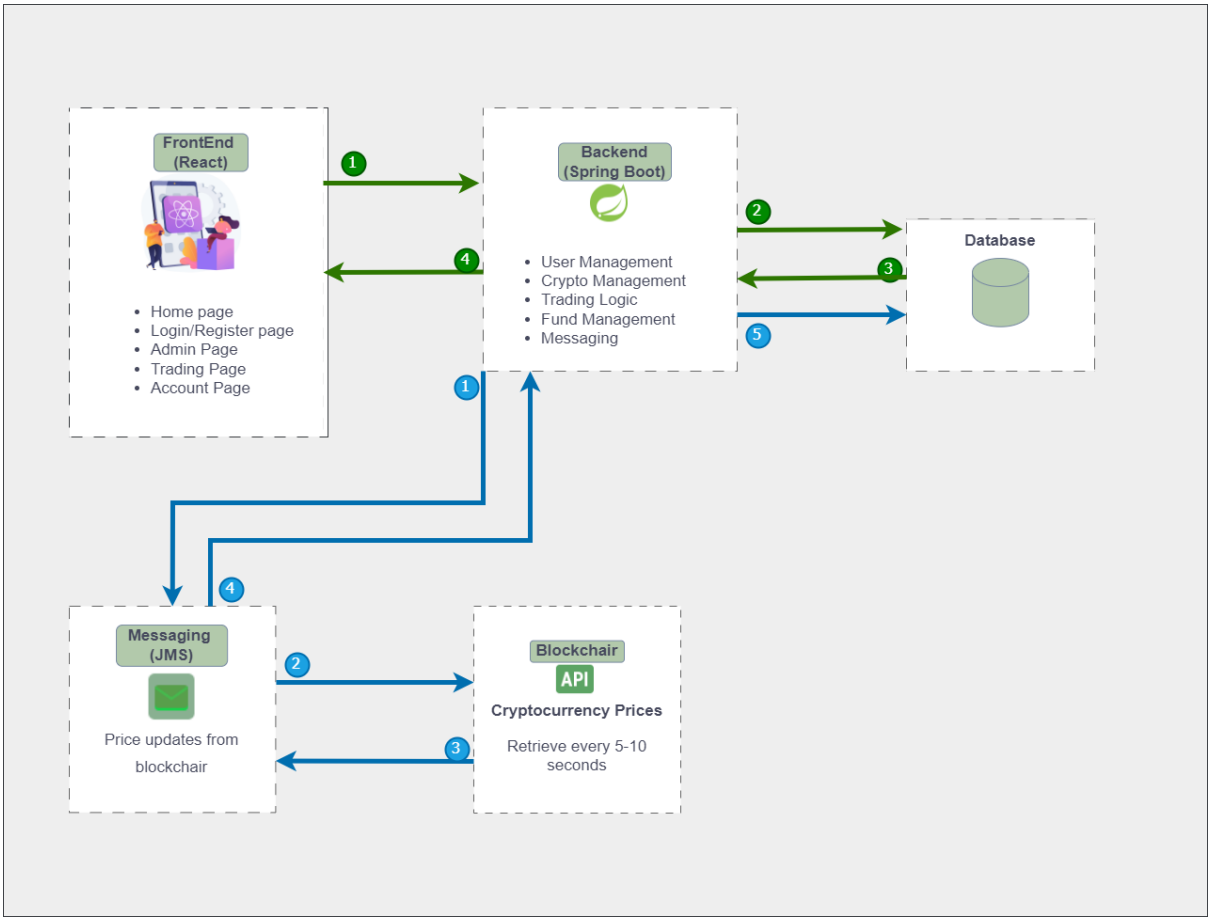


Diagramme d'architecture



Dans ce diagramme, nous avons illustrer deux choses:

- Les communications clients, en vert, par exemple pour récupérer les derniers prix des cryptos (mais convient pour toutes autres actions)
    - Le Frontend effectue une requête GET au Backend (1)
    - Le Backend récupère le dernier prix de la crypto dans la DB (2 et 3)
    - Cette valeur est retournée au Frontend (4)
- Les communications avec l'API Blockchair, en bleu
    - Le Backend va demander le dernier prix (1)
    - Le PriceProducer va aller récupérer le prix grâce à l'API Blockchair (2 et 3)
    - Le PriceConsumer va sauvegarder le prix dans la DB (4 et 5)