**Project description: Prykhodko-telegram-bot**

Task: Create a backend for the Telegram Bot project using Spring JPA.

The bot allows users to subscribe to the rate of one or more cryptocurrencies and manage their subscriptions.

Technologies used:

- Java version: 11+;

- Maven version: 4.0.0;

- Spring Boot version: 3.3.5;

- GitHub;

- MySQL database;

- JPA;

- Postman.

**Functional**

After connecting to the bot through a set of commands, the user can subscribe to receive daily cryptocurrency rates. He can subscribe to several cryptocurrencies at the same time.

**Commands for interacting with the bot:**

/start - connect to the bot;

/stop - unsubscribe from the bot;

**Subscription interaction commands:**

/addSubscription - start receiving the rate of the selected cryptocurrency;

/ removeSubscription - no longer receive the rate of the selected cryptocurrency;

/getAllSubscriptions - list of subscriptions.

**The administrator role** allows to access a list of users and their subscriptions**,** and also process statistics.

**Architecture**

**Repository Layer**

The Repository layer is responsible for interacting with the database. It uses Spring Data JPA to keep entities related to users-role, subscriptions and cryptocurrencies.

**Service Layer**

The Service layer display business logic and relationship between repositories and other applications components (controllers, external APIs).

**Controllers Layer**

The role of Controller plays TelegramBot.

**APIs**

**Telegram API** allows to connect bots to our system. Telegram bots are special accounts that do not require an additional phone number to set up. The Bot API makes it easy to create programs that use Telegram messages as an interface.

**Coincap API** allows you to access the current rate of more than 1000 cryptocurrencies.

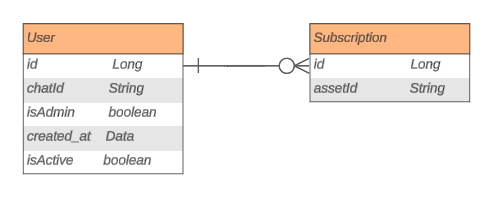
|  |  |  |
| --- | --- | --- |
| **Request** | **Method** | **Endpoint** |
| Get the current cryptocurrency rate | GET | api.coincap.io/v2/assets |

Expected result:



**Data model**

The data model includes two tables: users and subscriptions, related to each other on a one-to-many basis.



**Errors**

**Invalid cryptocurrency:** if the user incorrectly enters the name or ticket of the cryptocurrency.

**Service unavailable:** when there is no connection to the server.

**Security**

**Admin Authentication** assumes that only a user with the administrator role has access to information about bot users and their subscriptions.

**Token-based** **Authentication** –the bot uses Telegram Bot Token for authentication with the Telegram API.