PROJECT REPORT PT04

A crypto-currency exchange application: CryptoJoint

Ferran van der Have & Laurens Kubat (CP) & Reinier Sanders & Wout van den Berg & Wouter van Battum

Radboud University | Comeniuslaan 4, 6525 HP Nijmegen

Table of Contents

Introduction

General description

Our application let the users practice with the trading of cryptocurrency. Users are able to trade on different exchanges, with crypto coins. When downloaded, the users get 10000 fake dollars to trade with on the exchanges.

Goals of the system

The goal of our application is to let the customers get to know the ins and outs of cryptocurrency trading and practice with the trading.

Users of the system

We assume that the users of our application are familiar with cryptocurrency but not quite familiar with trading.
The users of our application are interested in our application, so they can practice trading with fake money and if they think they have mastered the skill of trading with cryptocurrency they can use real money.

Description

Focus on properties

Product justification

Our application is worth downloading because it gives someone the opportunity to learn how to trade with cryptocurrency. Also the usage and differences of exchanges.
A product that is similar to ours, is BUX. BUX is an application for the 'normal' stock market. However, BUX is only one example, but there are more of these apps. But our application is only one for crypto exchanges that we know of.
The new and innovative contributions of our application are:

- Cryptocurrency trading
- Different crypto exchanges
- …

Specifications

Design

Global design

Front-end:

Back-end:

We have four components in the back-end:

1. Client
2. Trader
3. API fetcher
4. Updater

Client

The Client component does all the communication with the front-end. The component keeps track of a wallet, in that wallet are all currencies the user has. It will be expressed in dollars or other currencies. The Client component has internally a map where all currencies and values are stored. The component requests from the Trader a trade-id and sends along a currency, if the combination checks out it gets a confirmation from the Trader component.

Trader

The Trader component requests an update of the values of the currencies every five seconds. It also does all the trading proceedings. The component makes sure the completion of the trades goes well. When the Trader gets trade request from the Client component, then the Trader returns a conformation and …

API Fetcher

The API Fetcher translates the packages received from the exchanges to a Map.

API Updater

The API Updater contains a list (Map) with previous requested currencies. From the Updater Maps of type <currency (String), prijs (double), tradeid> are send to the Client component.

Detailed design

Classes

Currency

Currency is a very important class. It gets the name of the currency the user wants to purchase and the value of the currency. Using the functions: getName(), getValue() and setValue(double value).

APIFetcher

The APIFetcher class returns API's. At first it checks using a public API (GetFormat(PairTradeType PairEndpoint, String Exchange)) which contains a switch function, on which exchange the user wants to trade and then returns through another public API (MakeBinanceAPI(PairTradeType PairEndpoint)) the wanted API.

/*
PairTradeType
The PairTradeType class returns a string of the pair with the … and returns a string of the en
d point, using the functions: getPair() and getEndPoint().
*/

API

The API class makes a http get request using a URLConnection method and with InputStream and ByteArrayOutputStream methods. The function is called makeCall().

Updater

The Updater creates an Updater using the constructor Updater(), in this constructor it sets the values of CurrentPairs and fetcher. CurrenctPairs is a private Map<PairTradeType,API> and fetcher is a private APIFetcher.

//Trade

Trader

This class has a private ArrayList<Currency> currencies, private ArrayList<Trade> trades and private Map wallet that matter. This class allows the user to make trades and see their wallet. In their wallet they can see which currencies they have and also the amount of that currency. The function showWallet() makes this happen. With the removeTrade(Trade trade) function a trade can be removed. The function deTrades() checks if the trade is successful. The function addWallet(Currency currency, double amount), adds currencies and the amount to the wallet.

Design justification

Our design is

Project management

Division of tasks

- Front-end: Reinier Sanders (chief)
- Back-end: Laurens Kubat (chief), Wout van den Berg, Ferran van der Have
- Project report: Wouter van Battum (chief)

Progression

Week 1:

In the first week we just planned the project and everyone suggested some idea's. Also the meeting with the student assistant was in the first week.

Week 2:

in week two we started on the report and made appointments for meetings about the stages of the application.

Week 3:

In week three we started implementing the code and interface

Week 4:

In week four we working on implementing the code and the interface, and during the meeting discuss the interface and make sure the same methods and attributes are used.

Week 5:

Organization

In the project meetings we all collaborated well and listened to everyone's opinion. If any uncertainties occurred, for example if Map needed to be used or a List, no one hesitated to ask. Also if someone struggled with a part someone who knew how to solve it or something would help always helped. The members that worked on components that cohered, communicated well to attune the classes and components to each other.

During the meetings we did not actually said at that time this has to be ready, it was more like who is going to do what and when that was decided everyone just started working on his part and occasionally someone in the WhatsApp group asked how everyone was doing.

Evaluation