CZECH TECHNICAL UNIVERSITY IN PRAGUE

FACULTY OF MECHANICAL ENGINEERING

DEPARTMENT OF INSTRUMENTATION AND CONTROL ENGINEERING

MASTER THESIS

CONTROL STRATEGIES FOR ARBITRATION IN CRYPTOCURRENCIES

TAMAROVSKIY SERGEY

2019

SUPERVISOR: DOC.ING.JOSEF KOKES

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Tamarovskiy Sergey** Personal ID number: **473089**

Faculty / Institute: **Faculty of Mechanical Engineering**

Department / Institute: **Department of Instrumentation and Control Engineering**

Study program: **Mechanical Engineering**

Branch of study: **Instrumentation and Control Engineering**

II. Master's thesis details

Master's thesis title in English:

Control strategies for arbitration in cryptocurrencies

Master's thesis title in Czech:

Řídicí strategie pro arbitrážní obchody v kryptoměnách

Guidelines:

1. Study the 'blockchain' technology and its application to artificial currencies (cryptocurrencies).
2. Verify whether a system, that makes arbitrage tradings, can profitably exist or not.
2.1. Create library of connections to APIs of various exchangers
2.2 Write the program that collects and stores the rates of cryptocurrencies(to verify if it is possible to make a profit from arbitrage trading)
2.3 Write the program to analyse the data.
3. Create a website to display the received information (tables, graphs and etc)

Bibliography / sources:

Debajani, Mohanty: BlockChain, From Concept to Execution. BSB Publications, New Delhi, 2018, ISBN 978-93-87284-18-0.
Bennett, Sean: A Guide to Understanding Blockchain. CreateSpace Independent Publishing Platform, 2017, ISBN 9781981954582.
Caetano, Richard: Learning Bitcoin. Packt Publishing, 2015, ISBN 978-1785287305.

Name and workplace of master's thesis supervisor:

doc. Ing. Josef Kokeš, CSc., U12110.3

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **26.04.2019** Deadline for master's thesis submission: **12.06.2019**

Assignment valid until: _____

doc. Ing. Josef Kokeš, CSc. Supervisor's signature	Head of department's signature	prof. Ing. Michael Valášek, DrSc. Dean's signature

III. Assignment receipt

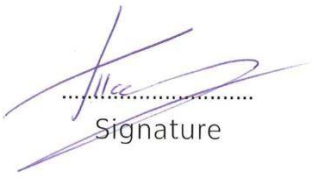
The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt	Student's signature

STATEMENT

I declare that I have worked out this thesis independently assuming that the result of the thesis can also be used at the discretion of the supervisor of the thesis as its co-autor. I also agree with the potential publication of the results of the thesis or of its substantial part, provided I will be listed as the co-autor.

Prague, 11.06.2019



.....
Signature

ABSTRACT

This work is, mainly, dedicated to trading cryptocyrrencies between different exhnangers that provide high safety, reliability and so-called API.

Nowadays, there are situations where at the same instance the same cryptocyrrency is traded at significantly different prices at different exchanges. Concurrent purchase/sales at various locations would therefore generate profits, absolutely without risk. This kind of trading is called arbitration. It can be summed up by saying that the main aim of the thesis is to verify whether such a system can exist or not and also to write a website to display the obtained results.

Keywords: API, cryptocyrrency, blockchain, exchanger, bitcoin, bitcoin cash, ethereum, litecoin, ripple.

Tato práce je zaměřena především na obchodování kryptocyrrence mezi různými výměník, které poskytují vysokou bezpečnost, spolehlivost a tzv. API.

V současné době existují situace, kdy se ve stejné instanci obchoduje stejná kryptocyrrence za výrazně odlišné ceny na různých burzách. Současný nákup / prodej na různých místech by proto generoval zisk, absolutně bez rizika. Tento druh obchodování se nazývá arbitráž. Lze to shrnout tak, že hlavním cílem práce je ověřit, zda takový systém může existovat či nikoliv, a také napsat webovou stránku pro zobrazení získaných výsledků.

Klíčová slova: API, kryptocyrrence, blockchain, výměník, bitcoin, bitcoin cash, ethereum, litecoin, ripple

ACKNOWLEDGMENT

I would like to express my honest gratitude to doc.Ing.Josef Kokes for his guidance and valuable comments on my master theis. Huge thanks are in order to my advisor, Ing.Matous Cejnek for his patience and advices. I would also like to thank my family for always being there for me.

ABBREVIATIONS

API	Application Programming Interface
BTC	Bitcoin
BCH	Bitcoin cash
ETH	Ethereum
LTC	Litecoin
XRP	Ripple
HTTP	HyperText Transfer Protocol
JSON	Javascript Object Nation
OOP	Object-oriented programming
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets
SQL	Structeured Query Language
CSV	Comma Separated Values

TABLE OF CONTENTS

INRODUCTION

Motivation

Nowadays, the use of cryptocurrency has increased multifold. The first cryptocyrrency bitcoin(it will be called as *BTC* in the text) was founded in 2009 by a man named Satoshi Nakamoto [1]. The growth of Bitcoin capital can be seen in Fig.1. It explains why the amount of crypto-exchangers has taken off. The People trade, pay, exchange and do other operations with this because of high level of safety, achieved by use of blockchain technology, and convenience.

Fig.1 Bitcoin capital growth chart [2]

To provide statictics and other information about cryptocurrencies, there are a lot of websites which have come up. However there are no websites that would provide possibilities of trading between different exhangers. That is the main reason whe i deided to figure out if such a system can exist and implement it. The system will:

- Gather statistic of selected cryptocyrrencies
- Save them in a proper format
- Analyze the data
- Provide an interface for users with ability to search information and change settings manually

Goals

The main goals of the thesis are :

1) To study the 'blockchain' technology and its application to artificial currencies (cryptocurrencies).
2) To verify whether a system, that makes arbitrage tradings, can profitably exist or not.
   2.1 Create library of connections to Application Programming Interface (*It will be called API in the text*) of various exchangers.
   2.2 Write the program that collects and stores the rates of cryptocurrencies(to verify if it is possible to make a profit from arbitrage trading)
   2.3 Write the program to analyse the data.
3) To create a website to display the received information (tables, graphs and etc).

For these purposes, I should define requirements and use cases for the project. The next step is to study blockchain technology, select exchangers and cryptocurrencies. Then, I need to research existing approaches and resources suitable for data gathering. Finally, I have to implement a Web application that will display all gathered data, generated analysis results and provide the tool for searching specific information.

Chapter 1

1 Analysis and research

The analysis and research part is the initial step in software development. In this chapter, I have written requirements, that the system has to have, such as fuctional and non-functional ones. At the next step I explained the use cases. Futhermore, I figured out how blockchain technology works. Afterwards I made a decision what cryptocurrencies and exchangers I would use for my thesis. The last part of this chapter is dedicated to the existing solutions of the problem.

The whole system consists of several separate components. Firstly, the program that takes and saves crypto-rates from selected exchangers. Secondly, the analyser of the received data and lastly, the GUI which is the website.

1.1 System functional requirements

The system functional requirements can be split into 3 groups which correspond to the system components.

Program that collects and stores the crypto-rates

F1. The system has to take the following data from selected exchangers:

- Couple of trading pairs(as an example – BTC/USD)
- Date
- Time
- Price for selling
- Price for buying

F2. The system has to have a scheduler, that would run the program in the particular intervals:

- Every 15 mins
- Every hour

F3. The system has to store data in the following manner:

- Create proper folders with appropriate naming
- Create files with appropriate names
- Check if mentioned folders and files already exist and if so, add gathered data

Analyser of received data

F4. The system has to analyse all saved files.

F5. The system has to separetly provide statistical analysis of crypto-rates:

- analysis with respect to exhangers
- analysis with respect to cryptocurrencies
- analysis with the respect to running time(15 mins or 1 hour)
-

F6. The system must provide a comparing algorithm based on collected data.

Web-based GUI

F7. The system must provide web-based GUI for displaying collected and analysed data in a user-friendly view.

F8. The system must provide communication between server and GUI.

F9. The GUI has to implement a specific interface for users queries. These queries have to contain the following parameters:

- user can set the parameters for fees:
  - transaction fee.
  - fee of 1st exhanger.
  - fee of second exhanger.
- user can see graph of changing rate in time by setting:
  - exhanger.
  - cryptocurrency.
  - buy or sell price.
- user can seek for specific results:
  - pairs of currencies.
  - pairs of exchangers.
  - date and time.
  - percent of fees.

1.2 System non-functional requirements

In this part all non-fuctional requirements can be found.

NF1. Legal resources – all websites, books and ect have to be free to use, or data usage permission must be received.

NF2. The entire project has to be implemented using following technologies:

- Python3 (for web development Django framework is used)
- JavaScript
- Hyper Text Markup Language (*It will be called as HTML in the text*);
- Cascading Style Sheets (*It will be called as CSS in the text*);
- Structeured Query Language (*It will be called as SQL in the text*);

NF3. The collected data has to be stored in the local machine.

NF4. The time length of stored data – for better analysing it should be at least 3 months of crypto-rates.

NF5. Handling of failures – all bags or problems with collecting of rates should be logged.

NF6. The web-based GUI has to be private.

NF7. Extensibility – the programs of this project should be done by using flexible technologies in order to be able to alter or create new features of the program easily in the future.

NF8. Reusability – implementation and desigh of programs have to be done in the way that all possible components could be reused for extending functionality.

1.3 System use cases

I have decided to dedicate this part of the first chapter to the system uses cases for the user. They also can be split into several points.

User logs in to the system

1)a user opens the web application.

2)the user can enter his user name and password and click submit.

3)in case the user does not have an access to the website, he has to proceed through the procedure to get one.

4)after successfull logging in, the user can see the main page with analysed results.

User views list of possible trading opportunities

1)the user has to login sucessfully.

2)afterwards he can see a table with all possible trading opportunities.

User seeks for particular trading opportunities

1)after logging in the user looks at the table with results.

2)the users types in the table's window what he wants to seek for.

3)now, after he has written condidtions, he can see new table of results.

User views graph of rates history

1)the user has to be logged in. In the main page he has to select exchanger, cryptocurrency and sell or buy price.

2)the user clicks on submit button.

3)now, the new web page opens and the user can see graph with history of rate.

User sets fees for exhangers and transaction

1)the user is logged in.

2)the user clicks the button, that is responsible for fees.

3)new web-page is opened.

4)the user can see 3 text box to fill in (fees for charges and transaction).

5) after he has clicked on submit button , it returns to the previous page with new results.

Admin adds new users

1)the admin has to open web page.

2)the admin has to use special user name and password to log in under admin rights.

3)after logging in, an admin web-page opens.

4)the admin can see the form for adding new users. He fills in the information with user name and his password, press the submit button.

5)new user is to be added to database.

Admin deletes users

1)after the admin has logged in correctly, he can see the the form for deleting users from database.

2)he fills in the form, only user name is required,presses submit button.

3)a user is deleted from the database.

User or admin log out

1)after successfull logging in at any page there is a logout button.

2)he presses this button, it returns him to the first web-page.

1.4 Blockchain technology

Since I have finished analyzing of how the whole system should be working, taking into consideration all requirements and use cases, research parts are left to be explained.

The first and basic part is what blockchain technology is and how it is used in trading of cryptocurrencies.

As it was said in introduction part, the popularity of cryptocurrency has taken off recently. The reason why is because of a few factors. This includes special design of data storage structure, transactions can happen without third sides and core of technology is blockchain.

It is known that blockchain technology was put forward in 2008 and implemented in 2009. In general, this technology can be regarded as a public legder and all committed transactions are stored in list of blocks. This chain gets bigger as new blocks are to be added to it continuously.

It should be noticed that to keep high level of user protection and ledger consistency, cryptography has been used. The blockchain technology generally has key characteristic of decentralization, persistency, anonymity and auditability [3].

The simple illustration of blockchain structure can be seen in Fig.2

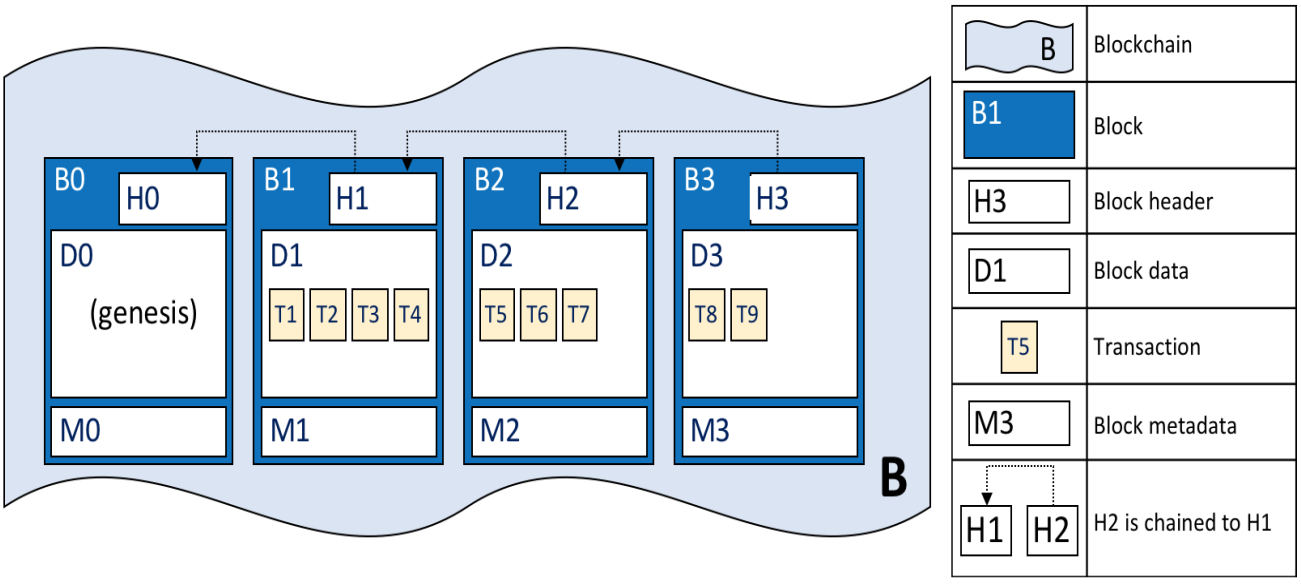


Fig.2 Structure of blockchain

Blockchain is a sequence of blocks, which holds a complete list of transaction records like a ledger [4]. Every previous block hash located in the block header, has only one ,so called parent block that is 256 bit hash value with pointer to the previous one. The first block of blockchain is called genesis block and it has no parent block.

In general Block consists of header and body. Each header has following parts:

- Version of block with indication of what set of block validation rules to follow;
- Merkle tree root hash – value of all trasactions in the block;
- Timestamp – current time as seconds;
- nBits – targer treshold of valid block hash;
- nonce – an 4 byte field, that begins from 0 and get begger by one for every hash calculations;
- parent block.

Each body contains:

- transaction counter – number of transactions depends on the size of block;
- transactions.

To validate the authentication, cryptography mechanism has been used. This is knows as digital signature.

All users have a pair of private key and public key. The private one has to be kept in confidentaly, because it is used to sign the transaction. All transactions are broadcasted via a network. The digital signature has two parts or so called phases : singing phase and verification phase.

In the first phase user 1 encrypts his data with his private key and sends it to the user 2, the encrypted result and original data. Now the second phase is to begin, the user 2 validates the value with public key of user 1. In that way, the second user can easily check if the data has been changed or touched by third person. The typical algorithm for digital signature that used is the ellipic curve digital signature algorithm(ECDSA) [3].

In the Figure 3 the typical interaction between two users(Bob and Alice) is illustrated.

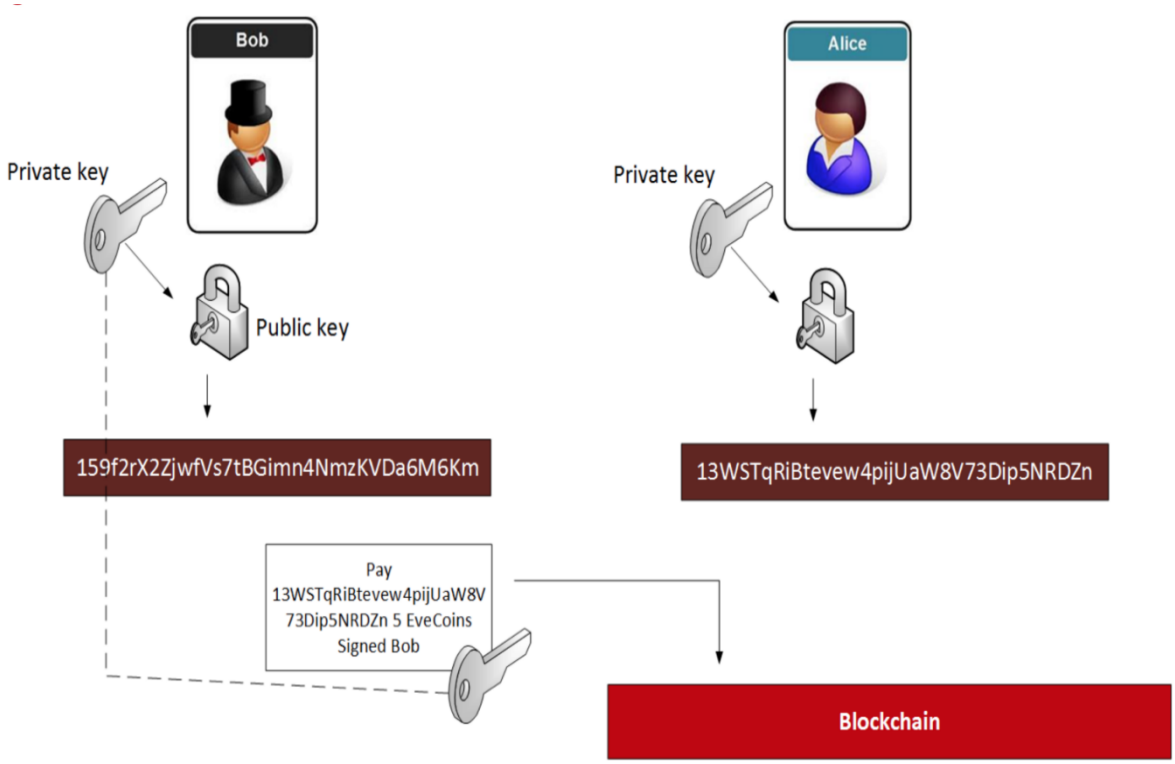


Fig.3 Two users exchange the data by blockchain

This part can be summed up by underlying the main characteristic of blockchain technology:

- Decentralzation system. In such a system, each transaction has to be validated through the central system. It basically means that there is no need ot third person to take part in it.
- Persistency. Validation happens really fast and invalid transactions would be accepted by miners. Moreover it is really close to impossible to delete or change transaction once it is inserted in blockchain. It leads us to the fact that if some block has an invalid transaction it can be recognised immediately.
- Anonymity. A user can communecate with blockchain with a generated address, that has no real identity of the user.
- Auditability. Usually the cryptocurrencies use the unspent Transaction output model for storing data about user balance [5]. All transactions have to refer to some previous unspent trasactions. Once the transaction gets to blockchain, the state of unspent transactions changes from unspent to spent, so all transactions can be easily verified and tracked.

1.5 API

Defining API

I can define the functionality of API in the following way: an API is a special interface that that allows interacting of computer systems with each other. There are many types of APIs. AS an example when we are talking about mobile phones, they provide APIs to provide access to location or other data that is taken from sensor (GPS (Global Positioning System) location or the orientation of mobile phone), the other sufficient example can be operating system, it also has API that is used by programs to open files, to get the access memory, and draw text on the screen. The programming language that I use to get done this task and so communicate with exchangers is PHYTON and it has a built-in API.

In general, it can be said that there are too many situations when it is really convenient to use API and also there is one thing all APIs have in common that is providing functionality for use by another program.

WEB API

When we are talking about WEB API we can define it as a framework that helps to build HyperText Transfer Protocol(*It will be called HTTP in the text*) Services that reach a broad range of clients or An API is the tool that makes a website's data digestible for a computer. Through it, a computer can view and edit data, just like a person can by loading pages and submitting forms. Making data easier to work with is good because it means people can write software to automate tedious and labor-intensive tasks. What might take a human hours to accomplish can take a computer seconds through an API [2].

When two systems (websites, desktops, smartphones) link up through an API, we say they are "integrated." In integration, you have two sides, each with a special name. One side we have already talked about: the server. This is the side that actually provides the API. It helps to remember that the API is simply another program running on the server. It may be part of the same program that handles web traffic, or it can be a completely separate one. In either case, it is sitting, waiting for others to ask it for data.

The other side is the "client." This is a separate program that knows what data is available through the API and can manipulate it, typically at the request of a user. A great example is a smartphone app that syncs with a website. When you push the refresh button in your app, it talks to a server via an API and fetches the newest info [6].

It the Fig.4 below, you can see the graphical representation of the described process

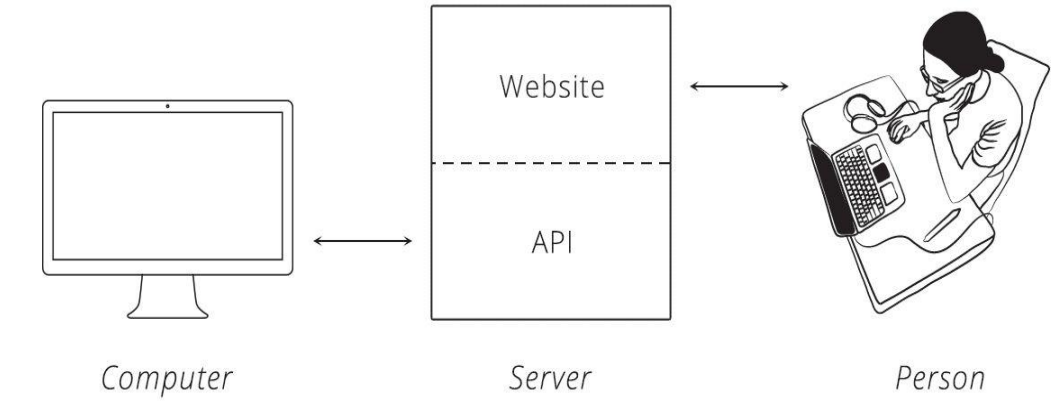


Fig.4 Communication of client and server throught API

When we send the request to the web server, that provides API, we get the response message and the structure of the data of message can be done in XML (Extensible Markup Language) or Javascript Object Nation (*It will be called JSON in the text*) format. The Fig.5 displays an example of data in JSON format.

Fig.5 JSON format

1.6 Selecting of cryptocurrencies

In this section I will explain what cryptocurrencies I have chosen and why.

Cryptocurrency is a digital currency that uses a blockchain technology to record the transactions. Nowadays there are more than 2000 different cryptocyrrencies that exist [2].

The crucial step for getting the task done was to select cryptocurrencies to work with. At the beginning I had to define the most important criteria for cryptocyrrencies, that can be found below:

1. Possibility of practical usage – if it is possible to buy things paying this currency or if it is easy to exchange it to another one. It is pointless to work with currency that you can not use later;
2. Stability of it – if the currency is new one, it can be not stable and get vanished soon. For my purposes I needed the one that has already been in market for longer than 1-2 years;
3. Market cup – it is important criteria to know how many tokens of currency exist and for what price;
4. Average investments per day – this criteria shows if popularity grows or decreases in time;
5. Availability in exchangers – one of the most important thing is to have chosen currency in different exchangers to be able to trade and in my case to get the price for sell and buy for later analyzing.

List of cryptocurrencies

1.Bitcoin(BTC)

This is the oldest and the biggest cryptocurrency nowadays. The defined criteria are applied:

1. BTC is easy to exchange and use for payments;
2. It's been more than 10 years of its existence;
3. It has the biggest market cup amoung all currencies, more than 100 000 000 000 USD [2];
4. Avearge investments per day is around 16 000 000 000 USD, which is really high [2];
5. It is available almost in all exchangers.

Result: perfect choice for the project.

2.Ethereum(it will be called as *ETH* in the text)

This currency was launched in 2015 [7]. One of the founder is Vitalic Buterin. The defined criteria are applied:

1. ETH is easy to exchange and use for payments;
2. It's been more than 4 years of its existence;
3. It has pretty big market cup amoung all currencies, more than 17 000 000 000 USD [2];

4. Avearge investments per day is around 6 000 000 000 USD, which is really high [2];
5. It is available in many exchangers.

Result: A good choice for the project.

3.Binance Coin

This currency is respectively new one, about 2 years old. The defined criteria are applied:

1. It is easy to exchange and use for payments in some platforms;
2. It's been about 2 years of its existence;
3. It has a market cup with more than 2 000 000 000 USD [2];
4. Avearge investments per day is around 170 000 000 USD [2];
5. It is available in some exchangers.

Result: Depending on exchangers it might be one of choices for the project.

4.Bitcoin Cash(it will be called as *BCH* in the text)

In 2017 BTC project and its community split in 2, one of them is BCH [8]. The defined criteria are applied:

1. BCH is easy to exchange and use for payments;
2. It's been more than 2 years of its existence;
3. It has big market cup amoung all currencies, more than 5 000 000 000 USD [2];
4. Avearge investments per day is around 1 000 000 000 USD, which is high [2];
5. It is available in many exchangers.

Result: A good choice for the project.

5.Monero

Monero is a growing cryptocurrency, that was started in 2014 [9]. The defined criteria are applied:

1. Monero is easy to exchange, but usage for payments is difficult;
2. It's been more than 4 years of its existence;
3. It has a market cup more than 1 000 000 000 USD [2];
4. Avearge investments per day is around 40 000 000 USD, which is pretty low [2];
5. It is available in some exchangers.

Result: A bad choice for the project.

6.Litecoin(it will be called as *LTC* in the text)

Litecoin was founded in 2012 [10]. The defined criteria are applied:

1. LTC is easy to exchange and use for payments;
2. It's been more than 6 years of its existence;
3. It has big market cup amoung all currencies, more than 4 000 000 000 USD [2];
4. Avearge investions per day is around 2 500 000 000 USD, which is comparativly high [2];
5. It is available in many exchangers.

Result: One of good possible choises for the project.

7.Stellar

Stellar was launched in 2015 [11]. The defined criteria are applied:

1. Stellar is easy to exchange, but usage for payments is difficult;
2. It's been more than 3 years of its existence;
3. It has a market cup more than 1 500 000 000 USD [2];
4. Avearge investments per day is around 200 000 000 USD, which is low compare to others [2];
5. It is available in some exchangers.

Result: A possible choice for the project.

8.Ripple(it will be called as *XRP* in the text)

XRP was founded in 2012 [12]. The defined criteria are applied:

1. XRP is easy to exchange and use for payments;
2. It's been more than 6 years of its existence;
3. It has big market cup amoung all currencies, more than 12 500 000 000 USD [2];
4. Avearge investments per day is around 895 000 000 USD, which is middle amoung the biggest ones [2];
5. It is available in many exchangers.

Result: One of good possible choices for the project.

9.EOS

The defined criteria are applied:

1. EOS is easy to exchange, but usage for payments is difficult;
2. It's been more than 2 years of its existence;
3. It has a market cup more than 4 000 000 000 USD [2];
4. Avearge investments per day is around 1 000 000 000 USD, which is high [2];
5. It is available in some exchangers.

Result: A possible choice for the project, it has strong creteria, except the fact that it is not too many exhabgers that use it.

Summary

After analysing more than 20 cryptocyrrencies(only 9 most competitive were mentioned) I figured out that too all of them, except few ones, have similar indicators. However, some of then can have really high market cup, but barely found in eschangers. In the table 1 it can be seen the most competitive crypticurrencies and how they fit to the given creteria.

Currency	Criteria 1	Criteria 2	Criteria 3	Criteria 4	Criteria 5	Result
BTC	✓	✓	✓	✓	✓	To use
ETH	✓	✓	✓	✓	✓	To use
Binance coin	+/-	+/-	✓	+/-	+/-	Not to use
BCH	✓	+/-	✓	✓	✓	To use
Monero	+/-	✓	+/-	+/-	+/-	Not to use
LTC	✓	✓	✓	✓	✓	To use
Stellar	+/-	✓	✓	+/-	+/-	Not to use
XRP	✓	✓	✓	+/-	✓	To use
EOS	+/-	+/-	✓	✓	+/-	Not to use

Tab.1 Suitable of currencies to criteria

After analysing the results, I have selected 5 cryptocurencies to work with, they are: BTC, ETH, BCH, LTC, XRP.

1.7 Selecting of exchangers

In this subsection I will explain what exchangers I have chosen and why.

It is really important to use proper exchanger, because it can lead people to lose money.

Nowadays there are hundreds of platforms that allow you to open a wallet and start buying or selling currencies, however, not all of them are suitable for the task.

That's why , I would say, selecting of exchangers is the most important thing. I have defined criteria for exchangers:

1. Legality of usage – if this exchanger is legal to use and work with;
2. Reputation – if this exchanger is used by many people around the world;
3. Availability of selected cryptocurrencies;
4. Methods of transferring money;
5. Public API– if the exchanger provides public API for free and without any need to open account in it.

List of exhangers

1.OKCOIN

This exchanger seems to be really famous among people. It was founded in 2013 [13]. It has low margin for transactions and low fees for operations. The defined criteria are applied:

1. OKCOIN is legal to use;
2. It has a nice reputation amoung users, also it has been in the market for more than 6 years;
3. All selected currencies are available there to trade;
4. It supports main methods of transferring money;
5. It provides public API for free and without a need to register in it.

Result: Perfect choice for the project.

2.COINBASE

COINBASE is the one of the biggest exchangers around the world. It was founded in 2012 [14]. The defined criteria are applied:

1. COINBASE is legal to use;
2. Being one of the oldest exhanger, more than 7 years, It has a perfect reputation;
3. BTC, BCH, LTC, ETH are available in it, however XRP is not used there;
4. It supports main methods of transferring money;
5. It provides public API only after registration and using key authentification, that makes use of ot high protective.

Result: A good choice for the project, but does not support public API without registration and does not work with XRP.

3.Kraken

Kraken was launched in 2011 [15]. The defined criteria are applied:

1. Kraken is legal to use;
2. It has high reputation;
3. All selected currencies are available there;
4. It supports main methods of transferring money;
5. It provides public API.

Result: A perfect choice for the project.

4.BITTREX

BITTREX was founded in 2014 [16]. It is well known exchanger. The defined criteria are applied:

1. It is legal to use;
2. BITTREX has a high reputation;
3. All chosen currencies are available there;
4. It supports main methods of transferring money;
5. It provides public API without a need of registration.

Result: A perfect choice for the project.

5.BITSTAMP

BITSTAMP was founded in 2011 [17]. It is well known exchanger. The defined criteria are applied:

1. BITSTAMP is legal to use;
2. It has a high reputation;
3. All chosen currencies are available there;
4. It supports main methods of transferring money;
5. It provides public API without a need of registration.

Result: A perfect choice for the project.

6.BITFINEX

BITFINEX was lauched in 2012 [18]. The defined criteria are applied:

1. It is legal to use;
2. The exchanger has a high reputation;
3. All selected currencies are available there;
4. It supports main methods of transferring money;
5. It provides public API only after registration and using key authentification, that makes use of ot high protective.

Result: A good choice for the project, but does not support public API without registration.

Summary

I have analysed 6 most famous and reliable exhangers.All of them could be nice resources for this project, because these days all exchangers try to satisfy all users demands. In the table 2 it is shown, the analysed exchangers and how they satisfy the given criteria.

Exchanger	Criteria 1	Criteria 2	Criteria 3	Criteria 4	Criteria 5	Result
OKCOIN	✓	✓	✓	✓	✓	To use
COINBASE	✓	✓	+/-	✓	+/-	Not to use
Kraken	✓	✓	✓	✓	✓	Alternative
Bittrex	✓	✓	✓	✓	✓	To use
Bitstamp	✓	✓	✓	✓	✓	To use
Bitfinex	✓	✓	✓	✓	+/-	Not to use

Tab.2 Suitable of exhangers to creteria

After analysing the results, I have chosen 3 exhangers to work with, they are: OKCOIN, BITTREX and BITSTAMP. The Kraken has remained as alternative variant, since it perfectly fits to the criteria.

1.8 Existing solutions

In this section I will briefly tell what I have found about already existing solutions of the task.

My final step in research was to find any other solutions of the task. After spending a lot of time researching and seeking for such solutions, I concluded with the following results:

1) In the internet there are a lot of websites that can provide information about rates of cryptocurrencies, however, if there is a need to have, as an example, rate of BCH for every hour in one month, for this information you have to pay. What I found for free was just maximum price and minimum price per day. That is why I needed to write a program for collecting rates of currencies at some particular time;
2) There are some providers who can give some advises how to trade currency and possibly get a benefit from that, however there is no guarantee that user will not lose money;
3) Some trading programs have been found as well. Algorithms of them are based on historical data and some dependencies. I classified these programs into two groups: programs that use algorithms that were created by people and algorithms based on artificial intelligence base.

4) The other possibility to make profit from cryptocurrency is the so-called mining. It is machines that solves some mathematical task and get some cryptos for that. But this is not really the point of my task.

Based on these results, I summed up by saying that there is not an existing program that would implement the idea of the proposed thesis. None of existing solution can guarantee the user a profit and the results if such a system can exist, I will explain in results of the project.

2 Implementation

Guided by the tasks of this thesis, I had to implement the appllication for collecting and saving cryptocurrencies rates, analyze them and display results. In previous chapter the analysis and research of application were provided. Based on the results I've decided to devide it into 4 most importante parts:

- Data acquisition
- Data storage
- Analyzing of received data
- Data displaying

2.1 Data acquisition

In this section I will describe implementation of data acquisition process.

First thing I was supposed to do was to create a library for API connections. Since I had already chosen exchangers and currencies I needed to find the APIs for particular cryptocurrencies. I would like to notice that sources I used are free accessed and can be found in the internet.

The APIs of selected exchangers provide an information in JSON format. In the figures 6-8 there can be seen examples how the data comes from chosen exchangers. In all figures there will be rates of BTC to USD.

In figure 6 we can see result of OKCOIN public API.

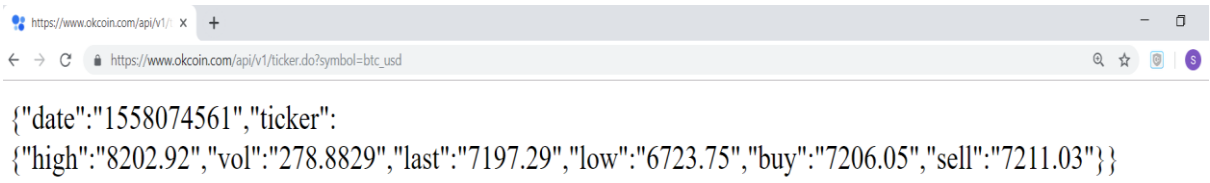


Fig.6 Result of OKCOIN public API

As it could be seen from example over the result OKCOIN API completely satisfies to my needs. From this data I needed to take "buy", the price that exchanger can pay for buying cryptocurrency, and "sell", the price that exchanger is ready to sell for one of exchanged currency. Also "date" is importance thing, it corresponds to the time when request is sent. "high", "vol", "last" and "low" could be also interesting information to analyze, but

for my goals they are redundant, so they have not been saved or used somewhere in the project.

Figure 7 consists of information that BITTREX public API provides.

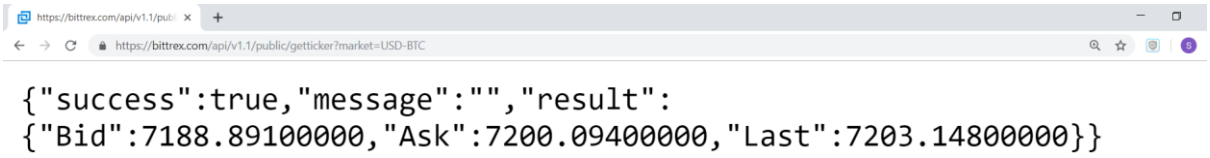


Fig.7 Result of BITTREX public API

We can absorb from the example above, the result BITTREX API also fits to my requirements, however the "date" information is missing. From this data I needed to take "Bid", that also can mean the price that exchanger can pay for buying cruptocurrency, and "Ask", which means the price that exchanger is ready to sell for one of exchanged currency.

One more example of request can be found in Figure 8. This one is from BITSTAMP API.

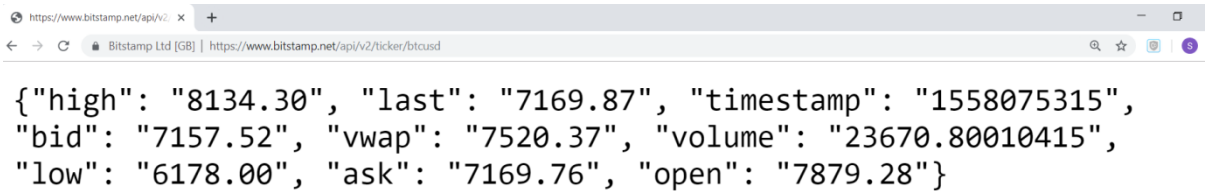


Fig.8 Result of BITSTAMP public API

The result from BITSTAMP totally satisfies to my needs. It contains of crucial information for me: "timestamp", "bid" and "ask".

After few example I got from API, I made a table that consists of links of all exchangers that I had to use. In Table 3 I inserted the links of APIs provided by selected exchangers.

Exchanger	Link for API
OKCOIN	https://bittrex.com/api/v1.1/public/getticker?market=USD-BTC
	https://bittrex.com/api/v1.1/public/getticker?market=USD-ETH
	https://bittrex.com/api/v1.1/public/getticker?market=USD-LTC
	https://bittrex.com/api/v1.1/public/getticker?market=USD-BCH
	https://bittrex.com/api/v1.1/public/getticker?market=USD-XRP
BITTREX	https://www.okcoin.com/api/v1/ticker.do?symbol=btc_usd
	https://www.okcoin.com/api/v1/ticker.do?symbol=eth_usd
	https://www.okcoin.com/api/v1/ticker.do?symbol=ltc_usd
	https://www.okcoin.com/api/v1/ticker.do?symbol=bch_usd
	https://www.okcoin.com/api/v1/ticker.do?symbol=xrp_usd
BITSTAMP	https://www.bitstamp.net/api/v2/ticker/btcusd
	https://www.bitstamp.net/api/v2/ticker/ethusd
	https://www.bitstamp.net/api/v2/ticker/ltcusd
	https://www.bitstamp.net/api/v2/ticker/bchusd
	https://www.bitstamp.net/api/v2/ticker/xrpusd

Tab.3 Links of connections for APIs

Based on these links I created library of connections and wrote a program to work with them that will be explained later.

Since Python language completely supports Object-oriented programming (it will be called as *OOP* in the text) I decided to relate each of exchager to the class.

Few words about OOP and Python:

- Object-oriented – almost everything is an object;
- Main features of OOP are encapsulation, data hiding, inheritance and polymorphismand;
- Class – new object type, which holds its own data members and member functions;
- Python code is automatically compiled to byte code and executed;
- Python can be extended in C anf C++, so it can provide high speed for operations [19];

The requirements for the first program that collects and stores the data were described in Chapter one. Since I've decided to use for each exchanger a class, I designed what each class has to consist of. In listing 2.1.1 there is main part of the program:

Listing 2.1.1 The main part of the first script

```
import okcoin
import bitrex
import bitstamp
import saving_writing



ex_okcoin = okcoin.Okcoin()
ex_bitstamp = bitstamp.Bisstamp()
ex_bitrex = bitrex.Bittrex()
saving_writing.insertInfo(ex_okcoin.data, ex_bitstamp.data, ex_bitrex.data)
```

As it can be seen in Python it is really easy to implement things. At the beginning I am importing scripts okcoin, bitrex, bitstamp and saving_writing. I named these scripts with respect to the functions they provide, so okcoin, bitrex and bitstamp are basically classes for exchangers. The saving_writing script is responsible for storing received data and will be explained in the section 2.2.

After importing these scripts, three objects are to be created. During creation of objects the particular request is sent to the API, so the last string of code is to save the data.

Now I will explain how each class works, however I will describe only one in details, because the other 2 is pretty similar, only few things might be different, such as different link for requesting, some names and structure.

As an example of structure of class I took the class which is dedicated to OKCOIN exchanger.

The main requirements for such a class were to have link for API, list of cryptocurrencies to form a proper request and list to save the received data in a way to execute it and work with it easily. Also the class has to contain functions such as setting and displaying data of created object.

Listing 2.1.2 depicts the Okcoin class.

To make a proper request and work with them easily, there are libraries in Python that already exist: request, urllib and json libraries. All of them are imported at the beginning of the script.

I've designed each class with respect to the requirements I said before. So as it can be seen in Listing 2.1.2 each object of this class will have its own properties:

- Main_api - the link for API of selected exchanger;
- cruptoC – list of currencies , also names are done in the way to add them to the end of main_api link to send a right request;
- data – dictionary that consists of keys and values. In Python dictionary is a collection, whose values are accessible by key [19]. In my case the dictionary has next keys: "BTC_USD", "ETH_USD", "LTC_USD", "BCH_USD", "XRP_USD", each of these keys has a value, that is another dictionary, that consists of 2 keys : "buy" with value of type float and "sell" with value of type float. Such a structure makes the searching or implementation in easy and understandable way. At the point of beginning if creating the object values for "buy" and "sell" are set to be zeros, in order to not have an computation error if one of chosen exchangers stop trading of some currency;

Few notes about used Python libraries:

- The request library is used to make HHTP request in Python. It is suitable for all HTTP requests, also easy to use;

- The urllib library is used to work with URLs, it provides functions such as opening, reading, parsing and other functions;
- The json library is used to work with JSON data. It provides encoding and decoding of data. In my case, as it was already mentioned, all replies come in this format.

Listing 2.1.2 Class Okcoin

```
import requests
import json
import urllib.parse

class Okcoin:
    main_api = 'https://www.okcoin.com/api/v1/ticker.do?'
    cryptoC = ['btc_usd', 'eth_usd', 'ltc_usd', 'bch_usd', 'xrp_usd']
    data = {'BTC_USD': {'buy': 0.0, 'sell': 0.0}, 'ETH_USD': {'buy': 0.0,
'sell': 0.0},'LTC_USD': {'buy': 0.0, 'sell': 0.0},'BCH_USD': {'buy': 0.0,
'sell': 0.0},'XRP_USD': {'buy': 0.0, 'sell': 0.0}}

    def __init__(self):
        count=0
        for i in self.cryptoC:
            url = self.main_api + urllib.parse.urlencode({'symbol': i})
            datatoprint = requests.get(url).json()
            try:
                if count==0:
                    self.data['BTC_USD']['sell'] =
float(datatoprint['ticker']['sell'])
                    self.data['BTC_USD']['buy'] =
float(datatoprint['ticker']['buy'])
                elif count==1:
                    self.data['ETH_USD']['sell'] =
float(datatoprint['ticker']['sell'])
                    self.data['ETH_USD']['buy'] =
float(datatoprint['ticker']['buy'])
                elif count == 2:
                    self.data['LTC_USD']['sell'] =
float(datatoprint['ticker']['sell'])
                    self.data['LTC_USD']['buy'] =
float(datatoprint['ticker']['buy'])
                elif count == 3:
                    self.data['BCH_USD']['sell'] =
float(datatoprint['ticker']['sell'])
                    self.data['BCH_USD']['buy'] =
float(datatoprint['ticker']['buy'])
                elif count == 4:
                    self.data['XRP_USD']['sell'] =
float(datatoprint['ticker']['sell'])
                    self.data['XRP_USD']['buy'] =
float(datatoprint['ticker']['buy'])
            except Exception:
                print("IS NOT WORKING", count)
            count+=1

    def set(self, main_api, cryptoC ):
        self.main_api = main_api
        self.cryptoC = cryptoC

    def get_data(self):
        p = 0
        print('Cycle for okkoin')
        for i in self.cryptoC:
            url = self.main_api + urllib.parse.urlencode({'symbol': i})
            datatoprint = requests.get(url).json()
            print('For ' + i + ': Sell is ' +
str(datatoprint['ticker']['sell']) + '. Buy is ' +
str(datatoprint['ticker']['buy']))
            p = p + 1
            if p == 5:
                print('End of cycle for okcoin')
```

In short, after this script is run, firstly it creates three objects. Each object consists of selected cryptocurrencies and their rates. While creating the new object, there 3 properties to be created as well. Next step of this class is that function def init__(self) is to be run. At this point the program makes five different requests to the API. It receives the reply from API, by using json library it takes needed information and saves it to the already created dictionary. This whole process is done by using 'for' loop, for each cycle it takes objects from list cryproC then it is to be added to main__api , that is how link for the request is done. Since I knew the positions of currencies in cryproC I decided to use counter to recognize and insert received data properly in the dictionary data.

Each class has 2 more functions: 'def set' and 'def get__data'. These 2 functions are used mainly for maintaining or testing the scripts. Function 'def set' sets the properties of object, while 'def get__data' prints out what dictionary data consists of.

In this subsection I explained in detail what program consists of and how it works. As an example I took Okcoin class and described it. The oter two classes Bitstam and Bittrex are really similar. They have the same properties and fuctions, the main diferences are that main__api changes and cryptoC list is different as well. Also I would like to point out that from all of properties, two are static, which are main__api and cryptoC, and one is dynamic – data dictionary.

This program has one more script which is called saving__writing, it is dedicated to saving received data and I will explain it in details in the next part.

2.2 Data storage

In this part I will explain how I implemented the storage of received data.

When I finished writing code for collecting data I designed the new one and as I said before the script that is responsible for data storage is a part of program that collects and stores the data. Being part of this program It was done in a separated script. As it can be seen in Listing 2.1.1 this script is called saving__wrinig. After the program creates three objects with rates, then the saving__writing script runs at the last function.

The crucial demandings for this script were to save rates in a way to have an access easily by other programmes and to be more or less universal to use.

In listing 2.2.1 there can be seen implementation of script that is supposed to store obtained data.

Listing 2.2.1 Program for storing data

```
import datetime


def insertInfo(okcoinData, bitstampData, bitrexData, argfromsys, path):
    #current_directory = os.getcwd()
    directory_okcoin = os.path.join(path, r'data_from_exhangers', 'okcoin')
    directory_bitstamp = os.path.join(path, r'data_from_exhangers',
'bitstamp')
    directory_bittrex = os.path.join(path, r'data_from_exhangers',
'bittrex')
    directory_names = {directory_okcoin: okcoinData, directory_bitstamp:
bitstampData, directory_bittrex: bitrexData}
    file_names = []
    now = datetime.datetime.now()
    if not os.path.exists(directory_okcoin):
        os.makedirs(directory_okcoin)
    if not os.path.exists(directory_bittrex):
        os.makedirs(directory_bittrex)
    if not os.path.exists(directory_bitstamp):
        os.makedirs(directory_bitstamp)
    for key in okcoinData:
        file_names.append(key)
    for temp_dir in directory_names:
        for temp_name in file_names:
            temp_file=os.path.join(temp_dir, temp_name)
            temp_file=temp_file+ '_'+ argfromsys+ '.txt'
            temp_file = open(temp_file, 'a')
            temp_file.write(str(now) + '    ' +
str(directory_names[temp_dir][temp_name]['buy']) +'   '+
str(directory_names[temp_dir][temp_name]['sell']) +'\n')
            temp_file.close()
```

To register the time when rates are valid I used python library datetime. It has a fuction in it that returns you current time, later this time goes to the file with rates.

Implementation of this script was done with help of inbuild Python libraries and fuctions, such as os(operating system) or fucntions for creating, opening, writing and closing files. It could show one more time how comfortable to use python language for developing software.

The function ‚def insertinfo' takes five arguments, three of them are objects that contain information about rates in exchangers, also the other argument is called ‚argfromsys', the purpose of it will be explained later,

and the last argument is a path, it is route where in the machine created files have to be stored.

The other question to think over was in what interval of time I need rates of cryptos and taking into consideration that the program has to be universal, I decided to write it in the way that it can run any time the user wants, however the problem was how to distinguish the collected data if there are two or more time intervals of sampling. That is when the argument ‚argfromsys' is coming. It is supposed to be type of string or integer.

Since I described the reasons of all arguments, now I will explain how the script itself works. After the function is run and all arguments have come as well, it creates folders with names of exchangers, with respect to the path I would give. Then in each folder new files are to be created where the rates are to be stored. These files have extension Comma Separated Values (*It will be called as CSV in the text*), every file has its own name that corresponds to the name of cryptocurrence plus the name of argument argfromsys. However if the folders and files already exist, the program writes new rates in the end of existing files.

In my case I decided to use as a path on my machine this address: D:/data_collection/... and figure 9 illustrates the schematic example how folders and files could look like.

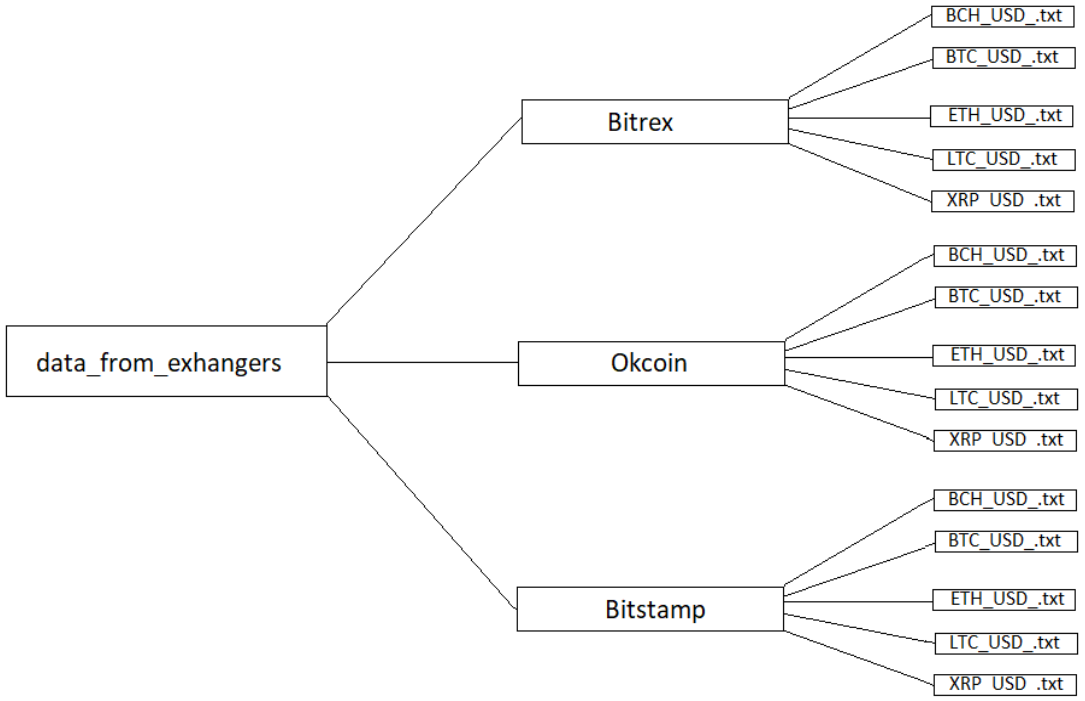


Fig.9 Structure of folders

To satisfy the requirement of errors or logs(Chapter 1, paragraph 1.2, NF5) I have used library logging. In my case it creates fale with name error.log then saves there logs and errors system can have during working.

After changes I have made, the main part of the first script changed as well, Listing 2.2.2 depicts it.

Listing 2.2.2 Final main part of the first script

```
import os
import sys
import socket
import logging


if socket.gethostname()=='aspicc-P85-D3':
    path=os.path.join('home', 'sergey')
else:
    path=os.path.join('D:','data_collection')


logging.basicConfig(level=logging.DEBUG,
                    format='%(asctime)s : %(levelname)s : %(message)s',
                    filename='error.log')


argfromsys=''
if len(sys.argv) > 1:
    argfromsys = sys.argv[1]
ex_okcoin = Okcoin()
ex_bitstamp = Bisstamp()
ex_bitrex = Bittrex()
insertInfo(ex_okcoin.data, ex_bitstamp.data, ex_bitrex.data, argfromsys,
path)
```

Afterwards I have decided to have my sampling period in every 15 and 60 minutes. To be able to run the program at this particular time automatically I used the software for scheduling that is called as CRONTAB. CRONTAB is a multi-user operating system that creates a list or table of different commands that are to be executed at particular time by operating system. These commands are set by user. So I used the fuctionality of CRONTAB to run my script everyday every 15 minutes and every hour automatically.

At the same time I needed server or machine that has a connection to the internet 24/7, since It was impossible to do in my own computer, I got a permittion to use server that belongs to CVUT. As a consequence I added to the script ‚socket library'. This library was needed to check from what machine the script was called and it is done by using gethostname() fuction.

Summary

I have written the program that satisfies all requirements that I defined in chapter 1. I divided this program in separated scripts. I created classes and objects to collect the data correctly and in a easy way. Then I wrote a script for storing the obtained data.

Afterwards I set the CRONTAB software to run the script at particular time.

The next step I did was to write a program for analysing rates. The work and fuctionality of it will be explained in the next section.

2.3 Analyzing of received data

This part is dedicated to explanations of implementing analyzer of received data.

To satisfy functional requirements (Chapter 1, paragraph 1.1, NF4-6) I have decided to write analyzer that would work separately from already existing programs.

The basic principle of operation of analyzer is explained below:

- at first, analyzer goes to the created folders with crypto rates, opens each file, takes each row of information from there and saves it in a memory (already prepared list or dictionary in the script);
- because all rates are saved as float type, analyzer can compare each sample of crypto rate with respect to time and exhanger;
- after this, if the analyzer finds that at some particular time the price to buy currency in one exchanger is higher than the price another exchanger can sell with the same currency, then analyser has to save this possible trading opportunity to a list. At the first round analyzer does not take into consideration the fees that the exchangers can charge the users for actions;
- finally, analyser can display all found trading opportunities;

Listing 2.3.1 depicts the part of script, where it is seen how analyzer compares all rates and saves them in lists.

Listing 2.3.1 Analyser compares rates of cryptos

```
for w in range(len(wholeData)):
    #print(wholeData[w])
    for w1 in wholeData[w]:
        for e in exhangers:
            if e!=w1:
                if (float(wholeData[w][e]['sell'])!=0) and
                 (float(wholeData[w][w1]['buy'])!=0):
                    if float(wholeData[w][e]['sell']) >
                     float(wholeData[w][w1]['buy']):
                        currenciesforweb.append(wholeData[w][w1]['currency'])
                        exhangersforweb.append(str(e + "/" + w1))
                        benefirforweb.append(float(wholeData[w][e]['sell']) -
                        float(wholeData[w][w1]['buy']))
                        feesforweb.append(0)
                        timeforweb.append(wholeData[w][w1]['time'])
                        selltoforweb.append(float(wholeData[w][e]['sell']))
                        buyfromforweb.append(float(wholeData[w][w1]['buy']))
```

As it could be seen from Listing 2.3.1 there is an algorithm for comparing rates. It is done with 3 'for' loops. First loop goes through all elements of "wholedata" list that consists of already saved rates from cryptos. So the length of it corresponds to the all samples taken from the beginning when the program runs. Then the next loop of wholedata[w] consists of all exchangers, so basically the program will go through all exchangers, all currencies and all saved samples of rates.

The list with name "exchangers" has only the selected exchangers, so next loop is to take all exchangers from this list. It is done in order to compare all rates, but not to compare rates from the same exchanger, that is why the next line is statement 'if' variables do not have the same values (same exchangers).

Next check point to be done is statement 'if' to check if some of rates are not zero. After observations of rates I have noticed that at some time exchanger can stop selling or buying currencies, but meanwhile it still provides API with rates, but they are equal to zero. To avoid the situation when at the same time one exchanger trades the cryptos and other has stopped but gives rates as 0, I added the statement to check if both of variables are not equal to zero.

After check is done, the comparison of two rates is done. If the "sell" price is higher than the "buy" price, the program saves to pre-prepared lists information about current sample.

This comparison is done without taking into account the fees. To solve it I have written the function that takes six variables as arguments, they are: two already compared rates, 3 arguments represent the fees and the last one is amount of money to trade. This function is called "benefitWithPercent" and can be seen in Listing 2.3.2.

As Listing 2.3.2 depicts the function "benefitWithPercent" returns list that contains 2 variables: 'flag', type of Boolean, and 'temp', type of float. Every time the function is called it returns list with these 2 variables, 'temp' variable is equal to a possible benefit taking into consideration fees. I used variable 'flag' to make clear if these couple of rates are beneficial to trade, so later if flag's state is true it would mean there is a benefit to trade with given fees.

Listing 2.3.2 Function "benefitWithPercent"

```
def benefitWithPercen(sellto, buyfrom, perfee1=0, perfee2=0, perfee3=0,
amoutn=100):
    #print(sellto, buyfrom, perfee1, perfee2, perfee3, amoutn)
    temp=0.0
    toreturn=[]
    flag = False
    if sellto!=0 and buyfrom!=0:
        if perfee1!=0:
            #print('if1')
            temp = (amoutn -(amoutn/100)*perfee1) / buyfrom
        else:
            #print('els1')
            temp = amoutn / buyfrom
        if perfee2!=0:
            #print('if2')
            temp = temp - (temp/100) * perfee2
        if perfee3!=0:
            #print('if3')
            temp = temp*sellto - ((temp*sellto)/100)*perfee3
        else:
            #print('els3')
            temp = temp*sellto
        if temp>amoutn:
            flag=True
    toreturn.append(flag)
    toreturn.append(temp)

    return toreturn
```

To sum up this part, I would like to say that analyzing of received data is done by a separate program. This program satisfies the given requirements and also it is done in a way to add or change it easily.

I have written a few tests to check its functionality in general and each functions. The tests were done and it showed that the program works correctly.

Since the analyzer will be used in displaying part, I will add some functions and also talk about it in the next part.

2.4 Data displaying

In this section I will explain and show how the received information is displayed.

As it was already mentioned before I had to display data in web-based GUI. In order to do that, I have decided to write a private website.

Nowadays internet is a rapidly developed network of computers with the goal to enable communication among them. The physical communication is done by connecting computer through wires, however on the other level of communication it is done by TCP/IP protocol.

To plot any information in the network(internet) we need to have it in some files, these files are created by using HTML language and called HTML-files.

Web-page is a document that user can get by reading HTML file with use of Web browser. Web-page can consists of text, graphics, links and other information.

Web-browser Is a program that is used to read HTML files. There are too many of web-browsers and the most popular ones are Opera, Google Chrome, Internet Explorer and ect. It also can be used to go from one web-page to another by links, to download media files and to play them as well.

Web-server basically is a computer that is connected to the internet, who supports HTTP protocol. HTTP is used to send a data, to download and play media files.

In my project for web development I have made a decision to use the idea of MVC pattern. Figure 10 depicts typical interaction with MVC pattern.

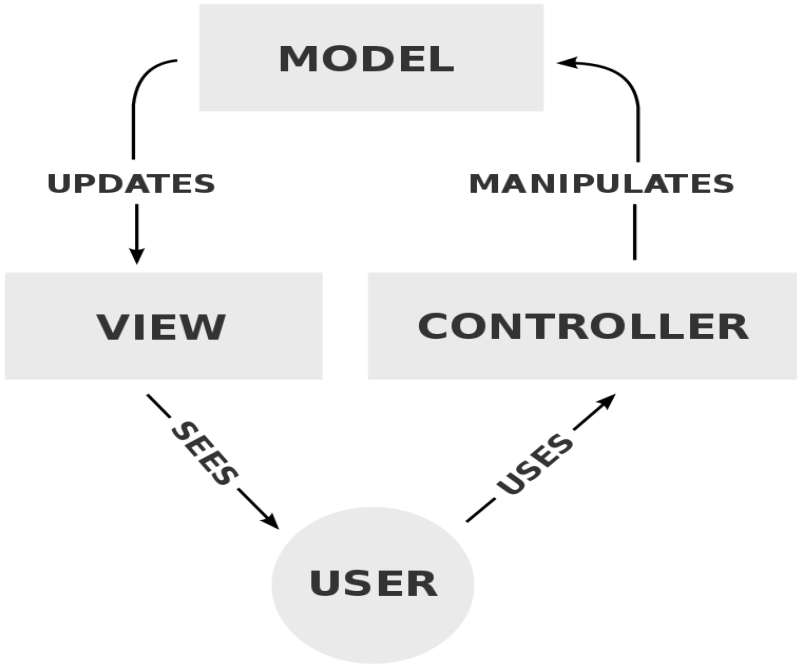


Fig.10 Diagram of interaction with MVC pattern

The MVC pattern is the most used pattern for web applications. It has been used for the first time in Smalltalk and then adopted and popularized by Java. It separates as application into 3 modules: Model, View and Controller.

The model is responsible to manage the data. It stores and retrieves entities used by an application, usually from a database, and contains the logic implemented by the application.

The View is used to display the data coming from the Model in a specific format.

The Controller handles the Model and View layers to work together. The Controller gets a request from the client, invokes the Model to perform the requested operations and sends the data to the View. The View formats the data to be presented to the user, in my application as a HTML output [20].

The main reason why I decided to use Django web framework in my project is because it supports the MVC pattern. The simple example of it can be seen in figure 11.

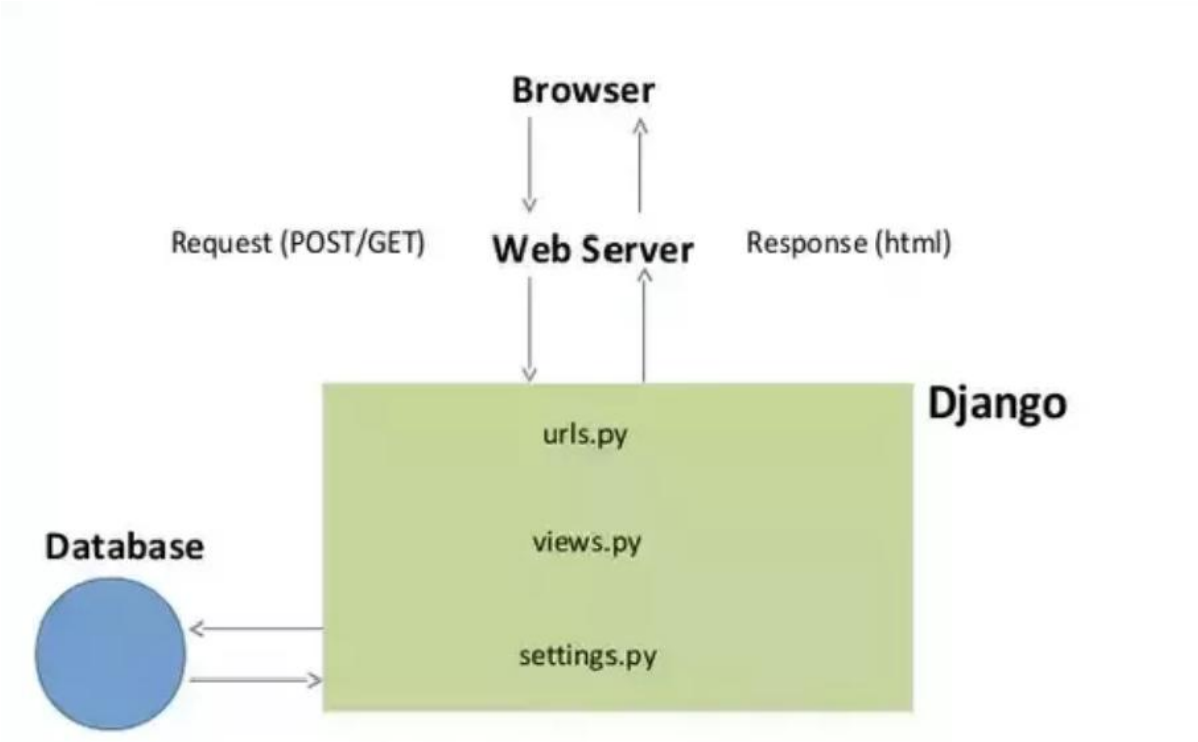


Fig.11 Django framework

The next explantion will be divided into 3 subsections:

- front-end development;
- back-end development;
- working with data;

Front-end development

For developing front-end, I have used the following technologies:

- HTML;
- Javascript;
- CSS;
- Jinja – a web template engine for the Python language;

Front-end web development is a client side of web application. It is graphical interface for user to work and interact with information using technologies, such as HTML, CSS and Javascript.

HTML is the base of web page. HTML code provides an general framework of how the web application will look like.

CSS is used for controlling presentation of web page, it allowes a webpage to have its own special design.

Javascript language usualy is used for transformation static HTML page into a dynamic interface.

For needs of my project I created 6 files with HTML extension. 4 of them are pages of my web application, the other 2 are used as separated fuctions, as an example button HTML, contains log out button that is inserted to all web pages. The list below is the list of all created HTML pages:

1.home.html – is the first web page to be opened. It has a view of typical private website, where you can find windows to fill in with information, such as password, user name and so on;

2.mainpage.html – after logging in the mainpage.html is to open. It has a main table of all posibilities of tradings, also links to open graphs of rate history and link for settings fees. The table of results contains the results of my diploma thesis and will be shown later in results chapter;

3.settings.html – this page is dedicated to set the values of fees;

4.adminpage.html – this page is for admin only. Deleting or adding users to database can be done in there;

5.button.html – consists of only code for log out button;

6.table.html – it is HTML page which is responsible to display the graph of history rate of selected currency;

The front-end part also has 3 created CSS files. I have written them to have an user friendly view of web pages.

The Following files represent them:

1.style.css – creates style for home.html page;

2.admin_style.css - creates style for adminpage.html page;

3.mainpage_style.css - creates style for mainpage.html page.

Listing 2.4.1 depicts the code of home.html page. There can be seen how I used Javascript for dynamic changes in web page, HTML code and use of CSS file for styles.

Listing 2.4.1 home.html page

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Logging in</title>

    <script type="text/javascript" href="jss.js">
        function write_to_creator(button){
         alert("If you want to have an access to this website you have to
proceed identification procedure. " +
                "Write to this e-mail address to get next instructions" +
"\nopa5@mail.ru");
      }
   </script>
    <link rel="stylesheet" type="text/css" href="../static/style.css"/>
</head>
<body>
<form action="/log/" method="post">
    {%csrf_token%}
    <div class="form-container">
    <div class="user-img"></div>
    <ul class="list">
        <li><h2>Member login</h2></li>
        <li><input type="text" name="user_name" placeholder="User
Name"></li>
        <li><input type="password" name="password"
placeholder="......"></li>
        <li><button type="submit">Submit</button></li>
        <li>If you forgot the password click below</li>
        <li><input type="button" name = 'write', value = 'Write to the
admin' onclick="write_to_creator(this)" /></li>
    </ul>
    </div>
</form>
</body>
</html>
```

Back-end development

As it was said previously for back-end I used Phyton language.

While front-end is a client side, the back-end is a server side of applications.

When user makes some requests or actions on a web pages, all of them are sent to the server part to be processed.

My web application has the following main scripts in the server side:

1.settings.py – consists of main settings of the application;

2.urls.py – consists of ursl addresses and fuctions for them respectively;

3.views.py – this script has fuctions for processing requests from client part;

4.database.py – has fuctions to work with database;

5.analysingofdata.py – this already created script for making an interaction between web site and received data of cryptocurrencies;

In the Listing 2.4.2 can be seen urls and fucntion to be called from view.py.

Listing 2.4.2 Urls in server side

```
from django.conf.urls import url
from . import views


urlpatterns = [
    url(r'^$', views.show),
    url(r'adminsave/$', views.admin_save),
    url(r'admindel/$', views.admin_del),
    url(r'log/$', views.login),
    url(r'logout/$', views.log_out),
    url(r'settings/$', views.settings),
    url(r'set/$', views.set),
    url(r'showtable/$', views.show_table),
]
```

For each of urls adresses there is a special fuction to be called. As an example when a user opens home page , after he has filled in his name and password and pressed the button „submit“, it sends the requests to server side and the „login“ fuction is to be called. This function is located in views.py script and can be seen in Listing 2.4.3:

Listing 2.4.3 „login“ function

```
def login(request):
    user_name = request.POST['user_name']
    password = request.POST['password']
    flags = db.find_user(user_name, password)
    if flags[0] and flags[1]:
        return render(request, 'adminpage.html')
    elif flags[0]:
        return display_main_table(request)
    else:
        return render(request, 'home.html')
```

As it is depicted in Listing 2.4.3 the function “login” has an argument – request, it takes values (user name and password) from the HTML page by method POST, and checks if they are in the database. Depending on the result user can get a reply and see the main page of website, if he has an access, or the same home page is to be returned if authentication is failed.

The whole website works by the logic of example above.

The next layer of application is working with data that will be described in the next subsection.

Working with data

The web application works with 2 different data. From the first side it works with collected rates that are stored in files with CSV format, at the second side the application has user information, which is stored in database.

As it was explained in Chapter 2, paragraph 2.2 all rates are saved in files with CSV format, hence to dispay them web application has to have an communication. In project architecture there is separated script with name analysingofdata.py that was already written to get an information from mentioned files. The script was changed after its origin to be able to communicate and satisfy fucntions of web application. Some new fuctions were added and changed. The use and work also were explained earlier.

For saving information about users I used a SQLite database. This database is free to use and also really convenient, however it was created for small project usage. Since I needed to store there only user information it perfectly fits to my needs.

The python language has a library to work with SQLite database – sqlite3. This module provides a SQL interface and it does not require a separate server process and allows accessing the database using nonstandard variant of the SQL query language.

By usage of the sqlite3 library I have created a database users.db. It has 3 columns :

- Id number – id number of user, type of integer;
- Login – the user name, type of text;
- Password – password of user – tepr of text;

At the very begining I added there 3 users with their names and passwords. Later the amount of them can be changed, only by admin rights.

In the web application structure there is separated script to work with this database. As an example of how the connection and work with it is shown in Listing 2.4.4. Listing 2.4.4 shows fuction for adding users to it.

Listing 2.4.4 Function for adding users to database

```
def add_user(name, password):
    conn = sqlite3.connect('./users.db')
    c = conn.cursor()
    position_to_insert=len(c.execute('SELECT * FROM users').fetchall())
    to_insert=[position_to_insert+1, name, password]
    # Insert a row of data
    c.execute('INSERT INTO users VALUES (?, ?, ?)', to_insert)
    # Save (commit) the changes\
    conn.commit()
    for row in c.execute('SELECT * FROM users'):
        print(row)
    conn.close()
```

The function "add_user" takes 2 arguments – name and password. First thing to be done is to create an object for connection with database. For this purpose sqllite3 library has a function "connect". The next step after connection is to create a "cursor" object. By calling "execute" function, commands are performed. In this example I use "INSERT INTO" command to insert new user to the database.

The script "database.py" consists of 4 functions that satisfy all needs of web application, such as adding user, deleting user, find user and function to create database

Summary

I have created the web application that satisfies all requirements that I defined in chapter 1. I have used the idea of MVC pattern to separate the application in 3 parts.

The work of front-end, back-end and data were explained.

The important point I would like to emphasize is that I have accomodated previously, the written programs with my web-based GUI. So it now works in a way for collecting the rates and website work separately and independently. The link or communication between them is done by program that analyses the data of rates. In order to make it work correctly I changed the analyser from its origin.

After run of whole project I got really important results that will be demonsrated in the next chapter.

3 Results

This chapter shows the results I obtained after implementing all required programs.

For the results I will show the main parts of web-based GUI and all possible opportunities that have happened during collection of rates.

First of all I would like to show how the first page of website looks. Figure 12 depicts it.

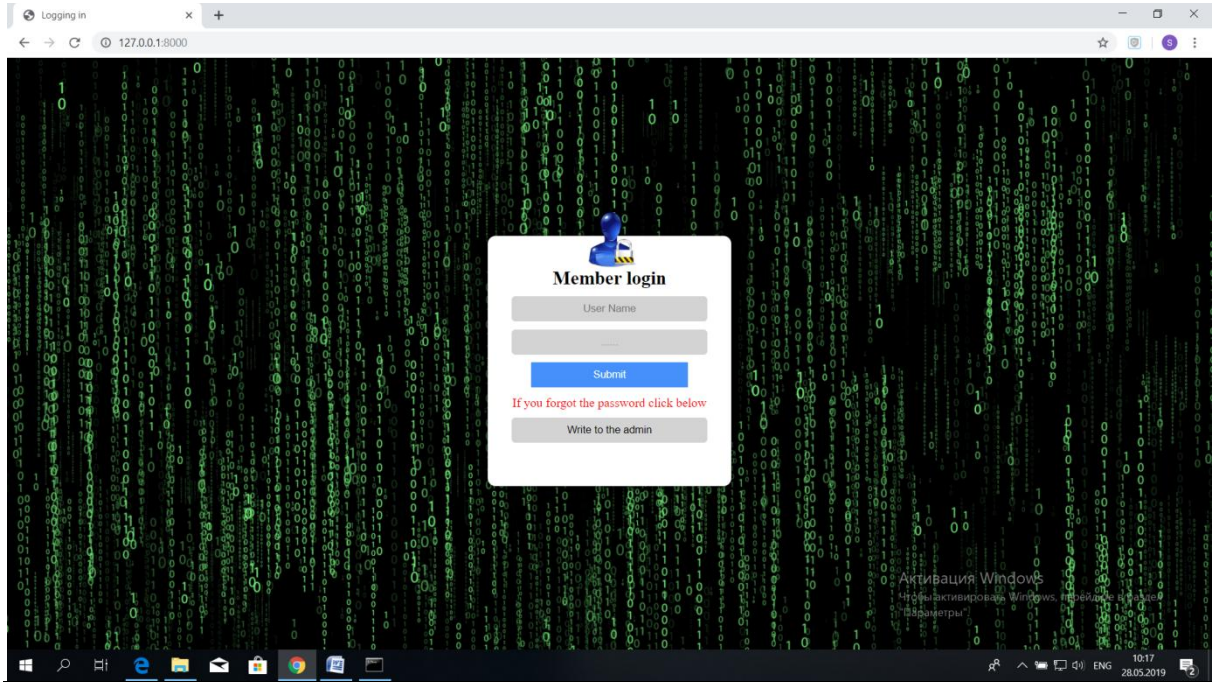


Fig.12 First page of created website

As it could be seen I run the website at the local machine, so the address of website is 127.0.0.1 port. The design and look of the first page is really user-friendly and modern, it is done by using CSS style.

It was said before that the website has to be private. To get access to the website the user is supposed to click „Write to the admin" button, then the window with information is to pop up. The Information says that in order to get an access you need to send an email to the admin's address with explanation why you need the access to the website and how you would use a information from website.

The other important view to be displayed is the admin page. Figure 13 illustrates how the admin view looks.

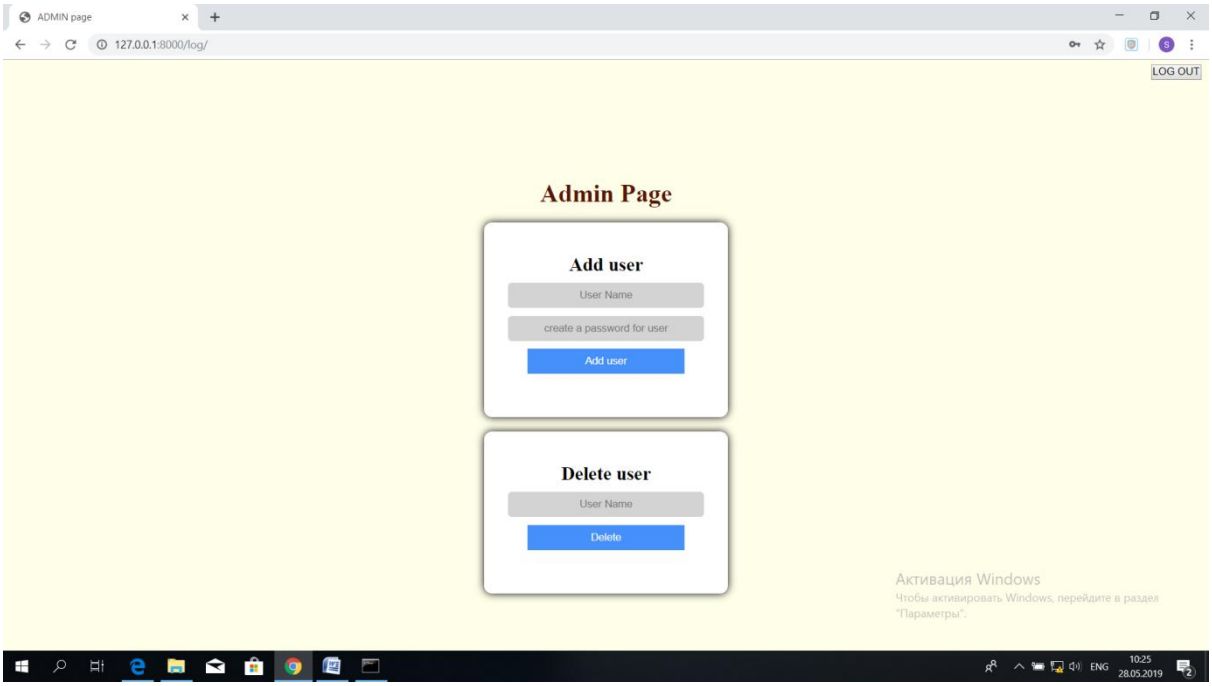


Fig.13 Admin page of created website

According to the requirements for the admin , there has to be a possibility to add or delete users. In the figure 13 it can be seen that the admin has the possibility to do that.

Now the Figure 14 shows the main page after logging in.

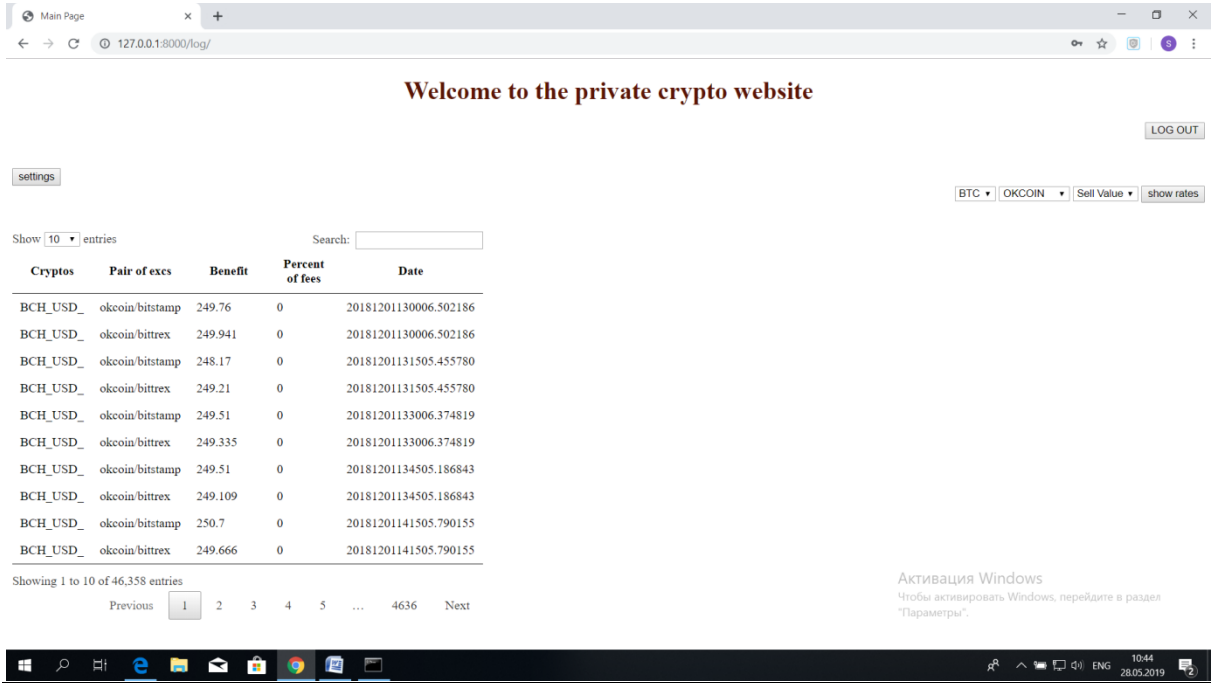


Fig.14 The main page of created website

Firstly I will explain the functionality, then the table of results. At the left upper corner there is „settings" button. User would use it to set the fees for

transactions. After pressing it, simple HTML page will come with text boxes to fill in, it will not be illustrated ,because of simplicity.

At the right upper corner there is a „log out" button for logging out and also selecting boxes for displaying rates. The user can set the currency, exchanger and sell or buy price of currency. After pressing „show rates" button, the graph will come. The example of such a graph is shown in Figure 15.

Fig.15 Example of graph with crypto rates

In the central part of Figure 14 is the table with results (possible opportunity to trade). The table has 6 columns with following information:

- Cryptos – crypto currency that you can trade to make a profit;
- Pair of exch – pair of exchangers where this opportunite happened;
- Benefit – the amount of profit from one trading;
- Percent of fees – set fees of exchangers for actions. At the beginning set as zeros;
- Date – time when the opportunity happened. It is float number, 4 first number represent year, next two – month, next two – day, other numbers is the exact time of sampling;

At the beginning the table displays only 10 results, however at the left up corner of the table there is a selecting box. User can set there how many results the table will display.

In order to make a convenient search at the right upper corner of the table is a search box. The search is universal, it searches in all columns

simultaneously, moreover this function is done in the Front-end part, so user does not have to wait from the reply from server and search can be done even without internet connection.

Now we came to the main point of the result part and my diploma thesis. After analyzing received data of rates the table consists of 46 358 possible trading opportunities. These opportunities have happened in the following interval of time:

- The first sample was taken 2018-12-01 at 13 00;
- The last one was taken 2019-05-20 at 12 00.

The number of 46 358 proofs that during last almost 6 months there have been so many possible tradings and also that the main idea of the thesis is real one.

Taking into consideration that the program that collects the rates is still working, the number 46 358 is to be increased.

However one important thing is to be noticed. All of these results are valid only with zero percent of fee. The more detailed analysis of it and risks will be considered in the last chapter of the thesis – Conclusion.

Conclusion

The first task of the thesis was to study „blockchain" technology. Secondly, to verify if a system that makes arbitrage trading with benefit can exist or not. The second task included three parts in it. The last step was to provide collected data and analysis results to the user by web-based GUI. All tasks were worked out and implemented. The implementation has strong sides and some flaws that have occurred.

In the first chapter, I defined system functional and non-functional requirements and use cases. Also how the „blockchain" technology works, what is API, how I selected cryptocurrencies and exchangers, were explained. Last section of the first chapter was dedicated to existing solutions of the task.

Based on the analysis and research, I have implemented the target applications. I divided this chapter into 4 main parts: data acquisition, data storage, analyzing of received data and data displaying. Each of part has some subparts with deeper explanations. As a result, the functional system was created. Both functional and non-functional requirements were fulfilled. However, because of lack of time and difficulty of the project, the quality of some implemented components can be improved in the future development.

As the final result, after collecting data during almost six months and analyzing it, I got the big number – 46 358 – that is how many trading opportunities have happened. However, after setting the fees for actions the number of them decreases. Also usually the percent of fee depends on how much money the user wants to spend for trading. So number of trading opportunities is varying with the amount of money. That it the advantage of my implementation that the user can set fees himself to see how many opportunities have happened recently. This proves the point that the system for making arbitrage trading can exist. However, there are still some risks to lose profit. The following points describe the main of them:

1) To make a trading profitable the user has to trade with big amount of money;
2) Time required for finishing transaction might be more than a few hours;
3) Because of some reasons one transaction might be declined while the other one is not;
4) Risks of cyber attacks.

Apart from the risks above, I would say that the system has a big chance to work correctly and make a profit without risks.

To conclude I would like to say that this thesis was a great benefit for me in which I have learned and tried in practice several technologies, that are new nowadays. Personally, I see a big potential of this project. The future possible changes I will describe in the following section.

Future changes

With high popularity and interest in cryptocurrencies the current thesis and created system has a lot to do in the future. The most important of them I will describe below.

- Data storage – since my in application all rates are stored in CSV format, in the future it should be restored in database in order to not occupy a big space in the memory;
- An attractive GUI – for some web pages there was a minimal amount of work with styles and Javascript. Therefore, I would say that some web pages could look more attractive.
- Mobile application – in the future some users might want to use the website through mobile phones, so the mobile version of web application should be implemented;
- Analysis of received data – nowadays the implementations of neural networks got increased. So the obtained data of rates could be used in a neural network to find some dependencies.

LIST OF FIGURES

LIST OF LISTINGS

LIST OF TABLES

References

[1] Katherine Sagona-Stophel – 'Bitcoin 101 white paper'
Available:
https://web.archive.org/web/20160813163512/http://www.trssllc.com/wp-content/uploads/2013/05/White_Paper_Bitcoin_101.pdf

[2] Coinmarketcap (analytical recourse)
Available:https://coinmarketcap.com/currencies/bitcoin/

[3] Zibin Zheng, Shaoan Xie, Honging Dai, Xiangping Chen, Huaimin Wang – ,An owerview of Blockchain Technology'
Available:
https://www.researchgate.net/publication/318131748_An_Overview_of_Blockchain_Technology_Architecture_Consensus_and_Future_Trends.

[4] D.lee Kuo Chen – "Handbook of Digital currency",
Available: https://www.sciencedirect.com/book/9780128021170/handbook-of-digital-currency

[5] Satoshi Nakamoto – "Bitcoin: A peer-to-peer electronic cash system",
Available: https://bitcoin.org/bitcoin.pdf

[6] Brian Cooksey - 'An introduction to APIs'.
Available: https://zapier.com/learn/apis/

[7] Website for ETH cryptocurrency.
Available: https://www.ethereum.org/

[8] Website for BCH cryptocurrency.
Available: https://www.bitcoincash.org/

[9] Website for Monero cryptocurrency.
Available:https://web.getmonero.org/resources/about/

[10] Website for LTC cryptocurrency.
Available:https://litecoin.com

[11] Website for LTC cryptocurrency.
Available:https://www.stellar.org/about/

[12] Website for LTC cryptocurrency.
Available: https://ripple.com

[13] Website for OKCOIN exchanger.
Available: https://www.okcoin.com

[14] Website for COINBASE exchanger.
Available:https://www.coinbase.com/about

[15] Website for Kraken exchanger.
Available:https://www.kraken.com/

[16] Website for BITTREX exchanger.
Available: https://international.bittrex.com

[17] Website for BITSTAMP exchanger.
Available: https://www.bitstamp.net

[18] Website for BITFINEX exchanger. Available: https://www.bitfinex.com/

[19] Dave Kuhlman – A Python Book: Beginning Python, Advanced Python, and Python Exercises.
Available: https://www.davekuhlman.org/python_book_01.pdf

[20] Xiaohong Li, Na Liu – Research on L-MVC framework.
Available:https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7943348