
**BRNO UNIVERSITY OF TECHNOLOGY**
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**

**FACULTY OF INFORMATION TECHNOLOGY**
**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**

**DEPARTMENT OF INFORMATION SYSTEMS**
**ÚSTAV INFORMAČNÍCH SYSTÉMŮ**


# THE SYSTEM FOR COLLECTION AND ANALYSIS OF CRYPTOCURRENCY EXCHANGE RATES

**SYSTÉM PRO SBĚR A ANALÝZU KURZŮ KRYPTOMĚN**

**BACHELOR'S THESIS**
**BAKALÁŘSKÁ PRÁCE**


**AUTHOR** **FILIP ČALÁDI**
**AUTOR PRÁCE**

**SUPERVISOR** **Ing. VLADIMÍR VESELÝ, Ph.D.**
**VEDOUCÍ PRÁCE**



**BRNO 2020**

**Vysoké učení technické v Brně**
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2019/2020

# Zadání bakalářské práce


Student: **Čaládi Filip**
Program: Informační technologie
Název: **Systém pro sběr a analýzu kurzů kryptoměn**
**The System for Collection and Analysis of Cryptocurrency Exchange Rates**
Kategorie: Počítačové sítě
Zadání:

1. Nastudujte si teorii za nejdůležitějšími kryptoměnami (Bitcoin, Ethereum, Ripple, EOS, Stellar, Cardano) a seznamte se s principy obchodování s nimi, zaměřte se zejména na kryptoměnové směnárny a cenotvorbu na nich.
2. Identifikujte vhodná dostupná API, pomocí nichž lze získat aktuální i historické údaje o vývoji směnných kurzů (jak pro kryptoměny, tak pro reálné měny). Při výběru zohledněte výkonnost (s ohledem na vestavěná omezení) a množství/přesnost dostupných údajů.
3. Navrhněte systém, který by využíval zdroje z bodu 2) a v pravidelných intervalech sbíral a transformoval směnné informace do databáze.
4. Dle doporučení vedoucího implementujte výše uvedený systém jako webovou aplikaci poskytující i REST API k ovládání.
5. Proveďte testování vašeho systému, zaměřte se na: a) zhodnocení výkonnosti subsystémů pro dotazování se ze zdrojů; b) sepsání vhodného množství jednotkových testů dílčích částí.
6. Výsledné řešení připravte pro nasazení v kontejneru. Diskutujte jeho možná rozšíření.

Literatura:

- Narayanan, A., Bonneau, J., Felten, E., Miller, A., & Goldfeder, S. (2016). *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton University Press.
- HILEMAN, Garrick; RAUCHS, Michel. Global cryptocurrency benchmarking study. *Cambridge Centre for Alternative Finance*, 2017, 33.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/
Vedoucí práce: **Veselý Vladimír, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 21. října 2019

# Abstract

This Bachelor's thesis is focused on cryptocurrencies, cryptocurrency exchanges and pricing principles on each of them. The goal of this project is to collect actual and historical data about cryptocurrency and Fiat money exchange rates from available resources. The data are collected and transformed into a structured database at regular intervals with a focus on effectivity and memory management. The proposed system which is capable of gathering such data is providing all collected information in the form of REST API or web application and deployed in Docker container image. The system validity is tested with performance measurement of the data-collecting subsystem and implementing several unit-tests for each part of the system.



# Abstrakt

Táto bakalárska práca je zameraná na kryptomeny, kryptomenové zmenárne a spôsoby cenotvorby na nich. Cieľom tohoto projektu je zber aktuálnych a historických dát o zmenárenských kurzoch, z dostupných zdrojov, zameraných na kryptomeny a Fiat peniaze. Zozbierané dáta su uložené v štruktúrovanej databáze s ohľadom na efektivitu a správu pamate. Navrhovaný systém, ktorý je schopný popísané dáta zozbierať, poskytuje všetky dostupné informácie vo forme REST API alebo webovej aplikácie a je nasadený v docker kontajneri. Validitu takéhoto systému som odtestoval výkonnostným testovaním modulu zodpovedného za zbieranie dát a implementovaním jednotkových testov pre každú časť systému.



# Keywords

cryptocurrencies, FIAT, exchanges, exchange rates, historical data, REST API

# Klíčová slova

kryptomeny, FIAT, zmenárne, zmenárenské kurzy, historické dáta, REST API


# Reference


ČALÁDI, Filip. *The System for Collection and Analysis of Cryptocurrency Exchange Rates.* Brno, 2020. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Vladimír Veselý, Ph.D.

# Rozšířený abstrakt

Cieľom tejto bakalárskej práce je pravidelné zbieranie aktuálnych a historických dát o zmenárenských kurzoch kryptomien a fiat peniazoch. Tieto dáta sú ďalej poskytované vo forme rôznych grafov a špeciálne vypočítaných hodnôt ako aj vo forme REST API.

Ako prvé, práca poskytuje základné informácie o kryptomenách a Fiat peniazoch, rôznych obchodovacích princípoch a kryptomenových zmenárňach. Práve kryptomenové zmenárne sú základným stavebným kameňom tejto bakalárskej práce a práve preto sú tu vysvetlené niektoré informačné komponenty, ktoré sú dostupné takmer na každej z nich. Ďalej sa práca zaoberá typmi kryptomenových zmenární a nakoniec poskytuje základné informácie o spôsobe ich zabezpečenia, čo je v dnešnej dobe celkom diskutovaná téma.

Aby bolo vôbec možné navrhnúť nejaky systém a začať implementovať, tak bolo potrebné zozbierať informácie o dostupných zdrojoch, ktoré poskytujú dáta opísané vyššie. Preto sa ďalej práca zaoberá dostupnými zdrojmi, opisuje kritéria na základe ktorých boli tieto zdroje vybrané, opisuje ich výhody a nevýhody a spôsoby použitia.

V ďalšej časti je popísaný navrhnutý systém vo forme ERD diagramu a flow diagramu. Ďalej práca popisuje technológie, ktoré boli použité pri implementáci ako napríklad Vue a Javascript pre frontend a PHP a Laravel pre backend.

V implementačnej časti tohto dokumentu sú popísané techniky akými boli jednotlivé časti systému implementované. Základným bodom implementácie je periodické zbieranie dát každú minútu (pre Fiat raz za deň), tak, aby bolo možno jednoznačne určiť, do ktorej kryptomenovej zmenárne tieto dáta patria. Ďalej bolo potreba vymyslieť ako budú tieto surové dáta poskytnuté koncovým užívateľom. Preto práca predstavuje rôzne vzorce a spôsoby prevodu, ktoré boli následne vo velkej miere transformované do databázových dotazov v PostgreSQL. Pre pohodlnú dostupnosť takto vypočítaných dát bolo implementované webové rozhranie pomocou Javascriptu a Vue frameworku, a ďalej REST API k ovládaniu pomocou Laravel frameworku a PHP.

Výsledky tejto práce, teda hlavne dostupné ceny, ktoré systém poskytuje boli zvalidované na základe porovnania s ostatnými platformami, ktoré takéto dáta poskytujú. Validácia dopadla úspešne, nakolko bolo zistené, že poskytované ceny približne korelujú s cenami na ostatných platformách.

Nakoľko tento systém z veľkej časti využíva dotazovanie na databázu, bolo potrebné odtestovať tieto dotazy hlavne v zmysle výkonnosti. Toto testovanie prebiehalo za použitia špeciálne príkazu **explain** v PostgreSQL. Výsledky aj s jednotlivými príkladmi su popísane v časti Testovanie, ktorá ešte nakoniec popisuje spôsoby testovania dostupného API. Toto testovanie bolo taktiež zamerané hlavne na validitu vrátených dát a výkonnosť.

# The System for Collection and Analysis of Cryptocurrency Exchange Rates

## Declaration

Hereby I declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Mr. Ing. Vladimír Veselý, Ph.D. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Filip Čaládi
May 28, 2020


## Acknowledgements

I would like to thank my supervisor Mr. Ing. Vladimír Veselý, Ph.D. for his guidance and valuable suggestions. As it is said, that people in information technology industry are food lovers, I would like to share recipe for my favourite food. The food is called Plum Dumplings and has been served in our family since I was born. The ingredients are: 5 medium potatoes (peeled, boiled, mashed and cooled), 2 large eggs, 1 teaspoon salt , 2 1/2 cups all-purpose flour, 18 damson or Italian prune plums (washed and pitted), 4 tablespoons (1/2 stick) butter, 1 1/2 cups breadcrumbs (very fine, 1/4 cup sugar, 1 tablespoon cinnamon.

And steps to make: peel the potatoes and boil for 20 to 25 minutes until they are soft enough to mash. Remove them from the water, mash, and cool them. They must be cool before using in the next step. In a large bowl, combine the cooled mashed potatoes, eggs, and salt. When well combined, add flour and mix until a soft dough forms. Cover the dough with plastic wrap and let it rest for 30 minutes. Place a large pot of salted water on to boil. On a lightly floured surface, roll the dough to 1/3 inch. Cut into 2-inch squares. Place a plum in the center of each square and fold in half, pressing out all air and sealing the edges. Moisten the edges before crimping, if necessary to seal. Carefully drop the filled dumplings individually into boiling water. Repeat until all plums are in the water. Cook for 30 minutes. Meanwhile, melt the butter in a large skillet, add breadcrumbs, and brown. Remove from heat and set aside. Using a slotted spoon, remove the dumplings to a colander to drain. Place the skillet back on the heat and add the dumplings, coating them with the buttered crumbs. Mix the sugar and cinnamon. Transfer the dumplings to a serving platter and sprinkle them with the cinnamon sugar. Serve the dumplings warm and enjoy.

# Contents

# Chapter 1

# Introduction

Cryptocurrencies are digital assets designed to replace centralized digital currency and central banking systems. They provide decentralized control of all transactions which are registered in a public distributed ledger. These assets can then be exchanged for different types of goods and services. Further, the consumers are able to hold the assets for various amounts of time and then sell them for either another cryptocurrency or Fiat money. Such tradings are mainly possible via cryptocurrency exchanges.

The goal of this thesis is to create a tool that is able to retrieve a large amount of actual and historical data about cryptocurrency and Fiat money exchange rates. Gathered data are then stored to the database and provided to the user in the form of the web application with different types of charts and statistics as well as in the form of REST API, which should serve as a middle interface between potential developers interested in this project and database. The unique aspect of this tool is that it can provide historical data in a specified range of time with reference to the concrete cryptocurrency exchange from which the data was retrieved.

The diploma thesis is divided into seven chapters. Chapter 2 outlines basic theory about the cryptocurrency market and highlights differences between cryptocurrency and Fiat money. It then explores cryptocurrency exchanges, their types, standard information components that can be found in every exchange platform, and different security measurements needed to consider specific exchange safe to store valuable assets in it. Chapter 3 describes various types of resources where the major one are compared with an emphasis placed on the rate between price and available pieces of information. Chapter 4 is dedicated to the design of web application and Rest API, as well as the application's database. Chapter 5 is focused on practices that were used during implementation. Chapter 6 describes techniques and tools which were used during performance testing and unit-tests implementation. Finally, Chapter 7 summarizes the work on this diploma, mentions the successes, and highlights possible improvements.

# Chapter 2

# Cryptocurrency Market

This chapter serves as a basic inside into the differences between Cryptocurrency coins and Fiat money 2.1. It then provides information about basic cryptocurrency trading principles 2.2. And further, the section describes cryptocurrency exchanges 2.3, their main information components, types, and security measurements.

## 2.1 Cryptocurrencies and Fiat Money

Money is any item that is generally accepted as payment for goods and services. The main functions of money are, for example, the medium of exchange or store of value. Any item that fulfills these functions can be considered as money. This section introduces two types of payment, such as Fiat money 2.1.1 and Digital money, especially their cryptocurrency sector 2.1.2.

### 2.1.1 Fiat Money

Fiat currency is any legal tender designated and issued by a central authority that people are willing to accept in exchange for goods and services because it is backed by regulation and because they trust this central authority[12]. Fiat money is generally divided into four groups:

- **Paper Currency**: consists of pieces of paper, for example in United States there are $1, $5, $10, $20, $50, $100 bills
- **Metal Coins**: a piece of hard material used as a form of money
- **Checking accounts**: allow its customer easy access to the fund by writing a check and use this fund to pay bills and executing other financial transactions
- **Electronic money**: modern use of electronic bank account balances without the use of a paper check, access directly through ATM machines and debit card

### 2.1.2 Cryptocurrencies

As I already mentioned in Chapter 1, cryptocurrencies are digital assets created to replace centralized banking systems. The core aspect of cryptocurrency is blockchain technology which is distributed, autonomous peer to peer network. This technology operates as a private payment gateway that uses cryptography to secure its transactions and control the

creation of new digital units [8]. However, Bitcoin is undoubtedly the true worldwide leader of the cryptocurrency market. There are more than one thousand cryptocurrencies in circulation, according to the authoritative website CoinMarketCap.com. These cryptocurrencies can be further divided into two groups, Bitcoin and Altcoins.

### Bitcoin

Bitcoin is the first developed decentralized digital currency created by an unknown group of people or one person named Satoshi Nakamoto. In the time of writing, it is the most used cryptocurrency in the world, with a total market of around 150 trillion U.S dollars[1]. The smallest amount that can people trade is one hundred millionth of a single Bitcoin, also called a satoshi. The total amount of Bitcoins ever created is estimated at 21 million units. This limitation must exist for the currency to have any value. Transactions can be made from user to user on peer to peer Bitcoin network without any third parties involved. They are then verified by network nodes by cryptography and recorded in public distributed ledger. This verification process is often called mining, which prevents from re-spending coins that were already spent in the past and generates new Bitcoin units that serve as a reward for nodes that successfully verified the transaction.

### Altcoins

Altcoin can be also defined as alternative coin. It basically means that it is any other cryptocurrency coin except Bitcoin. Many of the altcoins are built on top of Bitcoin blockchain with small differences in emission speed or encryption algorithms, these altcoins are also called forks. This approach is relatively easy to execute because Bitcoin is free and open source project. The process of creating new fork can be imagined as creating new HTTP web page. When creating a new web page, one does not have to create HTTP all over again [2]. The most famous altcoin forks includes for example Litecoin, Bitcoin cash, Dash and many others 2.1.

However, there also exists altcoins that are not directly forked from Bitcoin protocol. Such coins are adding new feautres to cryptocurrency family for example **Ethereum** is first platform that created blockchain based smart contracts. Another cryptocurrency **Dash** introduced new features such as instantly settled payments or transactions with near-zero fees. As you can guess there are more than enough of these coins and every one of them is trying to bring some new feature that would be different from the others and useful to the cryptocurrency community.

[1] https://coinmarketcap.com/currencies/bitcoin/

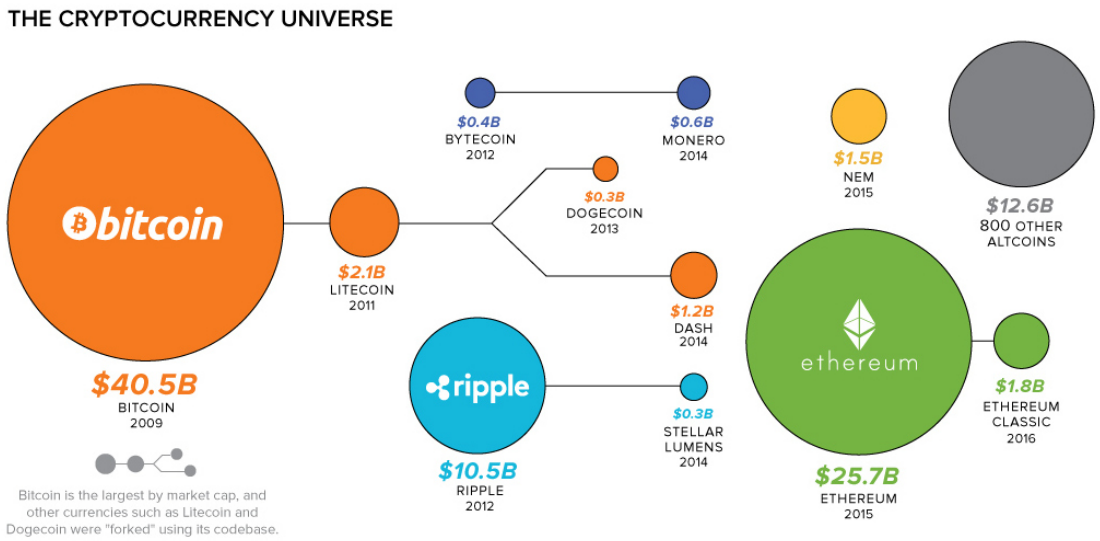



Figure 2.1: The cryptocurrency universe [2]

## 2.2 Trading Principles

This section describes different types of trading principles such as long selling 2.2.1, short selling 2.2.2, margin trading 2.2.3 and futures trading 2.2.4. It also provides examples of some principles, their cons, and pros.

### 2.2.1 Long Selling

Long selling can also be reproduced as buy low sell high strategy. Actually, this is a basic idea behind this approach to cryptocurrency trading. It means that the trader is investing in currency with belief that the price will go up in a reasonable amount of time, where he would be able to sell it for a higher price than he previously bought it for.

### 2.2.2 Short Selling

Short selling, shorting, or going short is one of the most challenging trading principles in the cryptocurrency market because the trader is required to have strong knowledge of the market and an excellent estimate when could the price of the market drop increasingly. A typical motivation for a short sale is the hope that the asset's market value will decline. The core concept of short selling can be described as follows. Suppose some trader believes that Bitcoin's price will drop significantly in a short period of time, where the current price is at 10 000 USD. Also, at this time, he does not possess any amount of Bitcoin at all. So, he decides to borrow one Bitcoin from a friend with the condition that he will return precisely one Bitcoin to him in a short period of time. Another step is that he sells that one Bitcoin for 10 000 USD, with the remainder, that he still owns one Bitcoin to a friend. After a few days, the price of Bitcoin drops significantly to 2 000 USD. For the trader, this is exactly the perfect time to buy back that one Bitcoin and return it to his friend. The result is that he completed the debt and earned 8 000 USD.

It is important to say that this method of trading is hazardous, as well. Continuing from above example, the price could not drop but go up, and in the end, the trader would not earn but lost a lot of money. Such trading is possible mainly by margin trading 2.2.3 or futures trading 2.2.4, which are described in the next sections.

### 2.2.3 Margin Trading

Margin or leverage trading is another commonly used strategy in cryptocurrency trading. The main principle of this strategy is that traders can increase their buying power against their current prices. Margin trading can be used to go short (where the traders speculate that price will go down) 2.2.2 or to go long (price will most likely go up) 2.2.1. In most of the cryptocurrency exchanges, we can margin trade at some given ratios, such as 1:5, 1:10, or 1:30 (at some exchanges, it can be described as 5x, 10x, and 30x). For example, a 1:5 ratio means that for every 5 dollars, the trader has to pay 1 dollar from his current funds. In the end, the trader would have to return the borrowed money plus some fees to the exchange, and the rest remains to him (if the trade is profitable).

This strategy is beneficial when the trader wants to buy some assets he can not currently afford. For example, if he wants to buy one Bitcoin at the price of 10 000 USD with the belief that price will most likely go up, but he currently owns just 10 dollars, he would need to margin trade at ratio 1:1000. Finally, when the price rises to the desired amount, he will sell back that one Bitcoin, return the borrowed money plus some fees to the exchange, and the rest remains to him. On the other hand, it can be very dangerous as well. From the above example, imagine that price will not go up, but instead down. That means that the trader would lose 9990 USD plus fees, which is a huge number.

### 2.2.4 Futures Trading

A futures contract is an agreement to buy or sell a specific asset at a given price and date in the future. It is bet between two or more parties on asset price to go down or up. The traders can either go long, betting on price to go up, or go short, assuming the price to go down. Traders going long agree to buy an asset in a specified date for a specified price and vice-versa for traders going short. Upon the arrival of the contract's expiration date, the parties settle, and the contract is closed.

## 2.3 Cryptocurrency Exchanges

Cryptocurrency exchanges provide services to buy and sell cryptocurrencies and other digital assets for national currencies and other cryptocurrencies. They were one of the first services to emerge in the cryptocurrency industry, and (at the time of writing), there are more than 300 active exchanges[2]. This section discusses different types of cryptocurrency exchanges, their main information components, and security, which is one of the most discussed topics in recent years after the Mt.Gox hack in 2014.

### 2.3.1 Information Components

This subsection discusses the essential information components on all cryptocurrency exchanges, which everyone should know when they want to start trading.

[2] https://coinmarketcap.com/rankings/exchanges/4/

### Chart

The chart shows changes in cryptocurrency rates over some time. It plots spread changes, which is a difference between prices of buy and sell orders at the same period. The chart is usually presented in the form of `Japanese candlesticks`.


Figure 2.2: Binance exchange chart

One candlestick represents price movement in a certain period. It is usually 10, 15, or 30 minutes (can be adjusted in some exchanges). The filled part of the candlestick is called the body, and it describes the opening and closing price of the specific period. If the body is red, that means that the closing price is lower than the opening price. On the other hand, when the body is green, the closing price is higher than the opening price. The lines over and below the candle's body represent the highest and lowest prices. That means when a candle does not have the upper line, the closing price is the highest price as well. The same principle is applicable to the candles below the line.

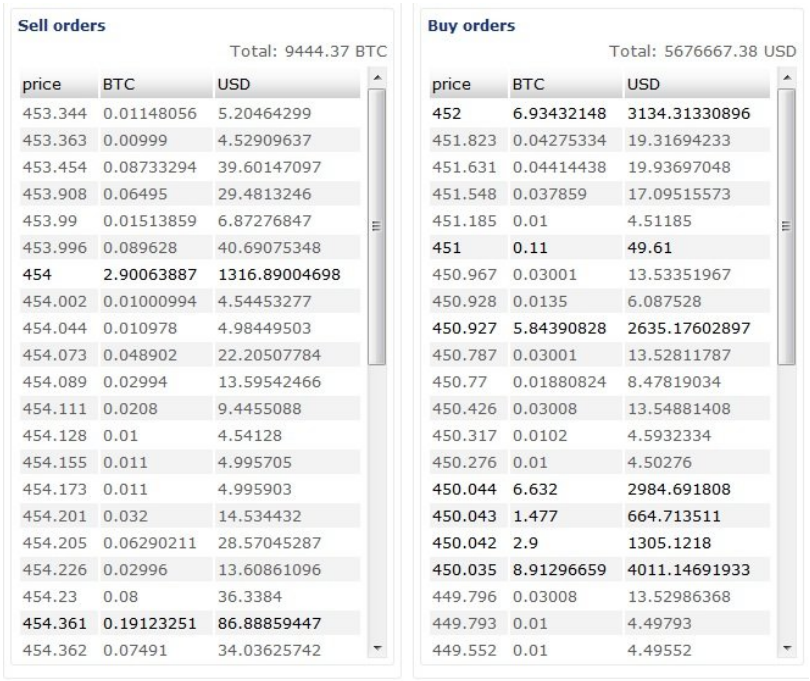

Figure 2.3: Japanese candlesticks

### Buy/Sell orders

Buy and sell orders are tables that show buyers and sellers orders that have the biggest chance to be executed. The tables are called order books. An important feature of order books is that sell orders are ordered from lowest to higher prices. Buy orders are exactly the opposite. Order books are divided into three sections:

- **price**: describes for how much is trader willing to buy or sell
- **asset's amount**: describes the amount of asset to be traded

- **final price**: sometimes even cost or value, typically it is product of **price** and **asset's amount**.



**Sell orders**

Total: 9444.37 BTC

| price | BTC | USD |
|---|---|---|
| 453.344 | 0.01148056 | 5.20464299 |
| 453.363 | 0.00999 | 4.52909637 |
| 453.454 | 0.08733294 | 39.60147097 |
| 453.908 | 0.06495 | 29.4813246 |
| 453.99 | 0.01513859 | 6.87276847 |
| 453.996 | 0.089628 | 40.69075348 |
| 454 | 2.90063887 | 1316.89004698 |
| 454.002 | 0.01000994 | 4.54453277 |
| 454.044 | 0.010978 | 4.98449503 |
| 454.073 | 0.048902 | 22.20507784 |
| 454.089 | 0.02994 | 13.59542466 |
| 454.111 | 0.0208 | 9.4455088 |
| 454.128 | 0.01 | 4.54128 |
| 454.155 | 0.011 | 4.995705 |
| 454.173 | 0.011 | 4.995903 |
| 454.201 | 0.032 | 14.534432 |
| 454.205 | 0.06290211 | 28.57045287 |
| 454.226 | 0.02996 | 13.60861096 |
| 454.23 | 0.08 | 36.3384 |
| 454.361 | 0.19123251 | 86.88859447 |
| 454.362 | 0.07491 | 34.03625742 |

**Buy orders**

Total: 5676667.38 USD

| price | BTC | USD |
|---|---|---|
| 452 | 6.93432148 | 3134.31330896 |
| 451.823 | 0.04275334 | 19.31694233 |
| 451.631 | 0.04414438 | 19.93697048 |
| 451.548 | 0.037859 | 17.09515573 |
| 451.185 | 0.01 | 4.51185 |
| 451 | 0.11 | 49.61 |
| 450.967 | 0.03001 | 13.53351967 |
| 450.928 | 0.0135 | 6.087528 |
| 450.927 | 5.84390828 | 2635.17602897 |
| 450.787 | 0.03001 | 13.52811787 |
| 450.77 | 0.01880824 | 8.47819034 |
| 450.426 | 0.03008 | 13.54881408 |
| 450.317 | 0.0102 | 4.5932334 |
| 450.276 | 0.01 | 4.50276 |
| 450.044 | 6.632 | 2984.691808 |
| 450.043 | 1.477 | 664.713511 |
| 450.042 | 2.9 | 1305.1218 |
| 450.035 | 8.91296659 | 4011.14691933 |
| 449.796 | 0.03008 | 13.52986368 |
| 449.793 | 0.01 | 4.49793 |
| 449.552 | 0.01 | 4.49552 |

Figure 2.4: Buy/Sell order book

### Resistance and Support

Resistance consists of a level where the price of an asset fails to break trough either due to intense selling pressure or so-called sell wall[3]. This level can be named 'ceiling', which is caused by a large supply of sellers operating in the same price area. This means that it can be surpassed just by a significant amount of buying orders.

Support refers to a level that usually holds the price of an asset on given value due to strong buying pressure or buy wall[4]. This level is expected to act as a 'floor' caused by a large supply of buyers in a certain price region. So it is clear that the price of an asset could be broken only by strong selling pressure at this level.

[3] https://www.binance.vision/glossary/sell-wall
[4] https://www.binance.vision/glossary/buy-wall


Figure 2.5: Resistance and Support levels

#### History of Transactions

This section show buyers/sellers prices and amounts they are trading at. History of transactions can be used to derive trading volume.

### 2.3.2 Types

This subsection describes two types of cryptocurrency exchanges such as Order-book exchange and Over the counter market.

#### Order-Book

Order-book is a real-time list an exchange uses for recording outstanding orders placed on the marketplace. The process is simple, when the seller places an order into the book, it is called *bid*, and when the buyer places an order into the book, it is called *ask*. If the exchange finds counterparty to order, it will manage the order and close both bid and ask.

There are two main types of orders:

- **Passive orders**

  Orders to buy or sell at a specified price or even better. They are typically placed into the order book until either they are canceled by the trader or processed by the exchange.

- **Active orders**

  Orders to buy or sell at the best available price. In highly liquid[5] market, the orders are processed almost immediately. They are not shown in the order book. Instead, they are placed into the trade history window to indicate market activity.

[5] https://en.wikipedia.org/wiki/Market_liquidity

### OTC

**OTC** stands for Over the counter market, which means that trading is done directly between two parties, without exchange as a middleman. Such trades are executed, in a decentralized place with no physical location, in dealer networks. But how does it work? Firstly, the trader has to find some institution which provides OTC trading services. After creating an account, the trader can show interest in either buying or selling cryptocurrency at a given price. Then, three things can possibly happen:

- On the platform, there may already be open orders from which the trader can choose from.
- Broker give a price quote for this trade.
- Broker can look for the interested seller in the client base or ask other brokers for referrals.

If both parties agree, the broker usually offers escrow service to ease the deal.

### Exchange vs. OTC

Imagine the trader wants to buy 1 Bitcoin on cryptocurrency exchange for current market price (active order). Firstly, he needs to create buy order which would be then placed into the order book. Further, let's say that in time of creating buy order, the sell orders look exactly as in Figure 2.6. Since active order is executed right away, this buy order will match the cheapest sell order available in order book, in this case 1 Bitcoin for 7097.41 USDT. But, what if the trader wants to buy 5 Bitcoins? The cheapest sell order available will not be enough to fulfill his order, so the exchange engine will match another cheapest sell orders available until the order is completed. So the trader will buy his 5 Bitcoins for 7097.51 USDT instead of 7097.41 USDT. This effect is called slippage[6].

If we compare trading same amount of Bitcoins on cryptocurrency broker, there is possibility that trader will be able to trade at price that is suitable for him thanks to the fact that price quote will be very close to the actual market price.

In conclusion, the result of this benchmark is that cryptocurrency exchanges are suitable for traders who are willing to trade small amount of assets, so in that case slippage should not happen so often. If they would still decide to exchange large amount of digital assets on cryptocurrency exchange they would, most probably, get them for much higher than actual market price or they could decide to trade them on more exchanges which is not ideal as well. Just because of this, OTC market is better place for traders, so called whales[7], who are willing to trade large amounts of digital assets. This can be done in one place with usually no fees at all.

[6] https://en.wikipedia.org/wiki/Slippage_(finance)
[7] https://en.bitcoinwiki.org/wiki/Whales

| Price(USDT) | Amount(BTC) | Total(USDT) |
|---|---|---|
| 7098.96 | 0.811429 | 5,760.30201384 |
| 7098.46 | 0.050000 | 354.92300000 |
| 7098.41 | 0.003995 | 28.35814795 |
| 7098.23 | 0.300000 | 2,129.46900000 |
| 7098.05 | 0.500000 | 3,549.02500000 |
| 7098.02 | 0.500000 | 3,549.01000000 |
| 7098.00 | 0.014331 | 101.72143800 |
| 7097.98 | **0.309130** | 2,194.19855740 |
| 7097.97 | 2.200000 | 15,615.53400000 |
| 7097.82 | 2.200000 | 15,615.20400000 |
| 7097.63 | 2.200000 | 15,614.78600000 |
| 7097.51 | 0.166447 | 1,181.35924697 |
| 7097.44 | 0.255895 | 1,816.19940880 |
| 7097.43 | 1.925255 | 13,664.36259465 |
| 7097.42 | 0.158465 | 1,124.69266030 |
| 7097.41 | 1.995972 | 14,166.23163252 |

Figure 2.6: Sell section of order-book

### 2.3.3 Security

The growing public interest in cryptocurrencies has made exchanges a big target for criminals and thefts as they handle great amount of cryptocurrency assets. There are a lot of cases that led to the complete loss of customer's funds, which were never retrieved back. In some cases where such events occurred, the exchanges had to be closed. This section describes one of the most famous cases wherein a particular exchange has lost a lot of funds and had to be closed in the end. It then highlights some of the security techniques which cryptocurrency exchanges use or should be using.

#### Mt. Gox

Attack on Mt. Gox, the leading Bitcoin currency exchange in 2013, is one of the biggest attacks of it's kind. In June 2011, the unknown hacker used credentials from a Mt. Gox auditor's tampered to transfer a large number of Bitcoins to his account. Right after, he manipulated exchange software to create massive sell orders and sold all the Bitcoins at any price. After this attack, accounts with more than $USD8,750,000 were impacted [3]. After a few other similar breaches, the exchange was delaying or refusing withdrawals and suspended the trading in the next years. In February 2014, alleged internal information published that the exchange lost 744,408 client's Bitcoins and 100,000 of its own in a theft undetected for years [1]. After that, the exchange bankrupt and faced lawsuits from its clients.

#### Two-Factor Authentication (2FA)

2FA is the most widely used advanced authentication method in everyday life. In general, multi-factor authentication is authentication where user, trying to log in, has to provide more logging factors (e.g., password and unique one-time generated token). Cryptocurrency exchanges usually provide 2FA logging (75%), withdrawing (77%), and trading (51%) services [7].

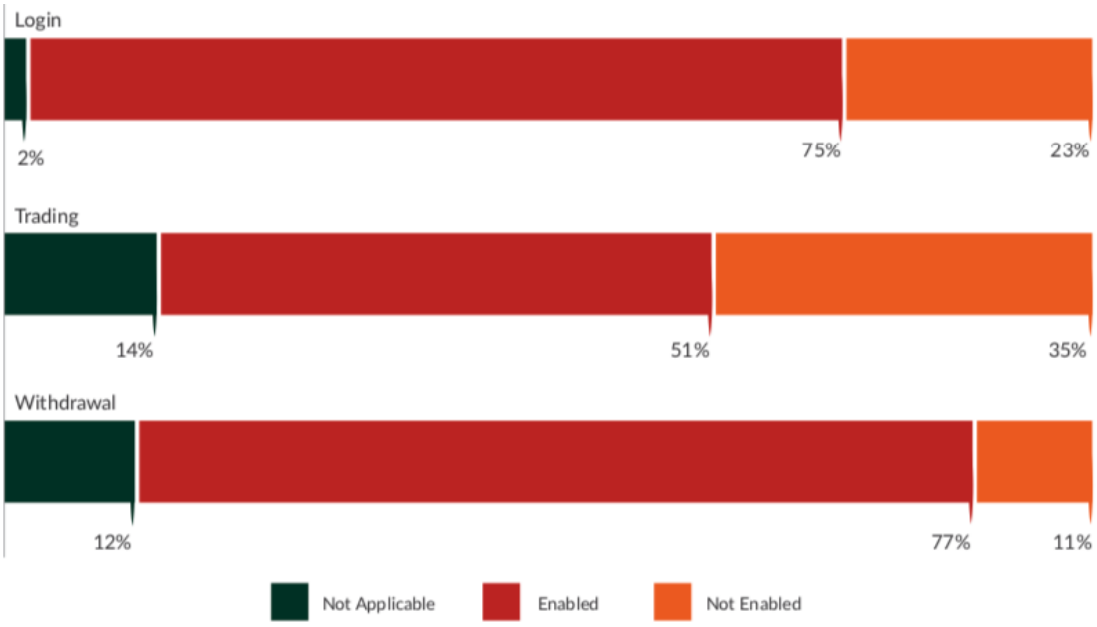



Figure 2.7: Customers - optional 2FA [7]

### Cold Storage

Cold storage in the context of cryptocurrency refers to storing cryptocurrency assets offline. For example, a cryptocurrency exchange that offers withdrawals feature keeps the majority of the funds in cold storage (not in the webserver or any other online computer), and the amount kept on the server is the amount needed to cover expected withdrawals for one day. 92% of exchanges indicate that they are using some cold storage system to secure a portion of both customers and their own funds [7].

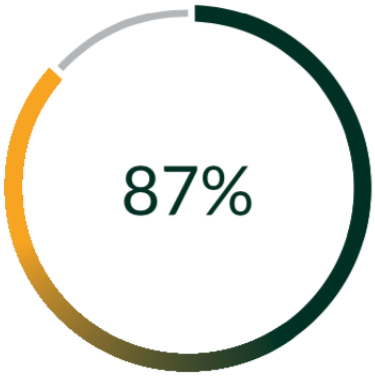



Figure 2.8: Average % of Funds Held in Cold Storage [7]

### Proof of Reserves

Proof of reserve is a cryptographic trick that gives exchange's customers some comfort about money that they deposited. It means that exchanges try to prove that they have some fractional reserve (e.g., 25% or even 100%) of deposits that customers made. Firstly,

the exchange publishes a valid payment-to-self transaction of claimed reserve amount. Then, it signs a challenge string (a random string of bits generated by some impartial party), with the same private key that was used to sign the transaction. This proves the participation of someone who knew that private key in proof of reserve [9].

# Chapter 3

# Data Resources

One of the biggest challenges of the thesis was to pick the best data resources suitable for this project. Therefore, this section describes different types of data resources, such as APIs and others, highlights the selected ones, and explains why they were selected.

As was mentioned above, the resources can be divided into two groups, API based resources, and others.[1]

## 3.1 API

The biggest drawback of this type of resource is that it is usually not free, and in order to use such resources fully, the user would have to pay a considerable amount of money, which in most cases is not an option. For all that, the available APIs were rated based on the following criteria:

- pricing
- rate limit
- amount of available data

### Pricing

It means that resources were taken into consideration only when they are for free or provides some free plan.

### Rate Limit

Usually, cryptocurrency API providers specify the available amount of requests for a given time interval. In general, the higher number of requests and shorter time interval, the more suitable the resource is.

### Available Data

The biggest challenge was to find a resource that provides current and historical data about cryptocurrency exchange rates for free and with connection to specific cryptocurrency exchange.

[1] They do not provide any API interface, which means that the data retrieval is completely up to the programmer.

### CoinMarketCap

CoinMarketCap is one of the most trusted cryptocurrency data authority, and it provides API for its customers. It provides data such as market cap, volumes, the latest crypto and fiat exchange rates, and many more. Moreover, the platform provides a free plan, but unfortunately, this plan does not include historical data and has a 10000/month rate limit. Because of this, this API is not suitable for the needs of this project.

### CoinGecko API

CoinGecko provides free crypto API with over 5000 coins and more than 350 exchanges [4]. This platform returns data such as live pricing, trading volume, historical data, information about crypto exchanges, and much more. But, the most important future of this API is that the rate limit is configured on 100 requests for one minute. The user can choose from more than 30 endpoints [5]. The most important ones for this project are:

- **/exchanges/{id}**

  Returns information (name, id, URL, etc.) about given exchange and different present values of exchange pairs provided by the exchange.

- **/exchanges/{id}/tickers**

  Returns the same data as the endpoint above, but without information about exchange. The user can also filter out the coins he is not interested in.

- **/coins/{id}**

  Returns current data such as price, market and many more for specified coin.

The API also provides historical cryptocurrency data with connection to the specific exchange, but unfortunately, this data are returned in *timestamp, volume* format, which is not enough.

### Cryptowatch

This platform provides REST API with real-time market data on 23 exchanges, which can be used to fetch 24-hour market statistics or historical data in a given time interval. The rate limit is measured by CPU allowance in nanoseconds where the allowance is reset every hour on the hour [6]. Even though the API is paid, it is accessible by the free plan as well. The free plan provides four seconds of CPU allowance, which is quite enough for the needs of this project.

The format of URL for historic data is **/markets/{exchange}/{pair}/ohlc**. *Exchange* is unique identifier of targeted exchange and *pair* is requested coin pair in format **BTCUSD** (BTC - Bitcoin, USD - american dolar). The user can also specify query parameters such as *before* (returns data before this time), *after* (returns data after this time), *periods* (time period for returned items e.g each day).

The platform also provides current cryptocurrency prices from the specific exchange. The format or URL for all market prices is **/markets/prices**. If the user wanted to specify the concrete exchange and pair symbol, he would use endpoint with the format **/market/{exchange}/{pair}/price**.

### Exchange APIs

Almost all cryptocurrency exchanges provides their own APIs with specific conditions and endpoints. However, most of them are free in some way or another, I did not choose to use all of them. Reasons for this decision are simple, some of the APIs do not provide their data in form of the system needs, their endpoints are quite uncomfortable to use or do not match criteria described in Section 3.1.

In time of writing, the system is collecting data from following exchange APIs:

- **Kraken**
- **Binance**
- **Bitfinex**
- **CoinbasePro**
- **HitBtc**
- **Okcoin**

### Foreign Exchange Rates

Exchange rates API is a free service for current and historical foreign exchange rates published by the European Central Bank [11]. The API web page does not provide any pieces of information about pricing or rate limit, so it can be said that it is completely for free and without any limit on request amount. It provides two endpoints **/latest** and **/history**.

- **/latest** This endpoint returns latest foreign exchange rates.
- **/history** Endpoint which provides historic data dated more than 15 years ago. The user needs to specify query parameters such as **start_date** and **end_date**.

The user can specify additional query parameter **base** (base currency) and **symbols** (returns specified rates separated by comma) in both endpoints.

## 3.2 Other Resources

Other resource means that specific platform provides data in the form of CSV or XML files or another type of data formatting. These resources are usually provided for free, and it is generally up to the user how he gets and handles the data.

### Cryptodatadownload

This platform provides free volume and historical data organized by cryptocurrency exchanges. Time intervals differ from exchange to exchange, but the most common ones, daily and hourly, are provided for all exchanges. The data are provided in the form of CSV files, easy to download and updated every day. The URL query parameters for specific coin pair and exchange are as follows **cdd/{exchange_name}_{coins_pair}_d.csv**. Thanks to this simplicity of URL query, the data can be easily downloaded e.g., with curl[2] tool or any other transfer data tool.

[2] https://curl.haxx.se/

| Date | Symbol | Open | High | Low | Close | Volume BTC | Volume USD |
|---|---|---|---|---|---|---|---|
| 2020-01-03 | BTCUSD | 6942.3 | 7216.2 | 6860 | 7216.2 | 1039.95 | 7213839.92 |

Table 3.1: Example of Kraken BTC/USD pair

### European Central Bank

The European Central Bank (ECB) is the central bank of the nineteen European Union countries that adopted the Euro. The bank provides current and historical euro foreign exchange reference rates for 32 currencies. The data are updated every working day around 16:00 CET except on target closing days (New Year's Day, Christmas Day, etc.). Exchange rates can be downloaded in forms of CSV, XML, or PDF files. The bank even provides examples for developers on how to download their data, so it should not be hard to integrate this platform into the project.

### Conclusion

As you can see, there are many resources of different types and rate limits. All of them have some free plan included (some of them are free at all). That is because of the pricing condition described in Section 3.1. What should be clear from this section is that not all free plans are convenient for projects where many requests are required for a short period of time. That means that not all of the highlighted resources are actually used in the implemented system. The usage of individual resources is described in Section 4.

## 3.3 Possible Improvement

One of the possible improvements is that the data would not be gathered from third-party APIs or resources in general. Instead, there could be a single module, for every cryptocurrency exchange, that would be responsible for collecting current exchange rates and historical data as well. The module would be implemented in the form of an automated web scraper. Automated web scraping is a method for extracting data from websites using a bot or web crawler. Using this method, the proposed system would not be dependent on third party data resources, which could prevent multiple problems in the future.

# Chapter 4

# Design

This chapter focuses on describing the design of a tool that periodically collects current and historical data about cryptocurrency and foreign exchange rates. It also highlights used tools such as PHP, Javascript, Laravel, Vue framework, and PostgreSQL.

## 4.1 Database

The application database can be divided into two main parts. The first part is responsible for storing data about current and historical cryptocurrency exchange rates, while the second part focuses on Fiat money exchange rates from the present and past as well.

The database, which is represented by the ER diagram in Figure 4.1, is written in PostgreSQL language. It is an open-source object-relational database system that uses and extends the SQL language. Postgre is ACID-compliant and runs on all operating systems. The most popular features of this database system are that the user can define his own datatypes, build custom functions, or write code from different programming languages without recompiling the database.

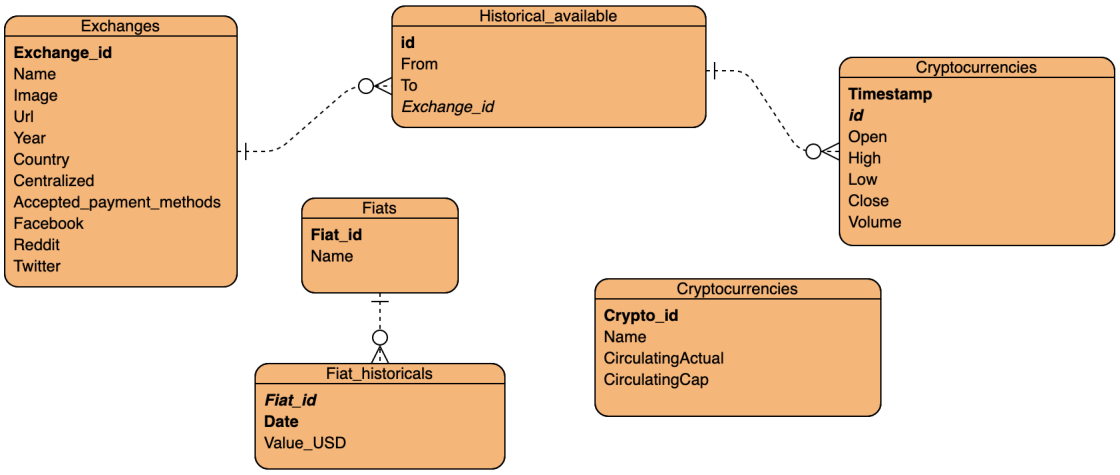



Figure 4.1: Application's ER diagram

The entity-relationship diagram consists of six entities. It was designed with respect to the so-called database normal forms, which reduce data redundancy and improve data integrity.

- **Exchanges**: Represents all cryptocurrency exchanges in the system.
- **Cryptocurrencies**: Represents all supported cryptocurrencies in the system.
- **Historical available**: Contains information on whether or not the system holds historical data from the specific exchange with specific currency pair.
- **Cryptocurrencies**: Stores Open, High, Low, Close (OHLC), and Volume data for specific market pair and cryptocurrency exchange.
- **Fiats**: Stores all information about supported fiat coins.
- **Fiat Historicals**: Data about historical fiat money exchange rates on a specific day with USD as the base currency.

## 4.2 Available Data

This section describes the availability of different types of cryptocurrency exchanges, coins, and Fiat money. I have had to analyze available data from resources described in Chapter 3. These resources were then intersected, so the available data in resource A would be available in resource B.

### Crypto Exchanges

Selection of supported exchanges was based on intersection of all available exchanges from resources described in Chapter 3 and finally, thirteen top exchanges, from my point of view, were selected. The overview of all available exchanges is provided in Figure 4.1.

| Gemini | CoinBase | Kraken | Bitstamp | Bitfinex | Cexio | Poloniex |
|---|---|---|---|---|---|---|
| Binance | Bittrex | HITBTC | Okex | BitBay | OKCoin | |

Table 4.1: Available crypto exchanges in the system

Proposed exchanges are further available in all other data resources dealing with cryptocurrencies from Chapter 3.

### Crypto Coins and Fiat Currencies

Cryptocurrency coins that are supported by the proposed system were selected as follows. Firstly, I have analyzed the top fifty cryptocurrencies by market capitalization from the CoinMarketCap platform. Then, I have proceeded in descending order and gradually verified if the specific cryptocurrency coin was available in all data resources (Chapter 3). If yes, the coin was inserted into the resulting list of 20 supported cryptocurrencies. The full overview of all cryptocurrencies is provided in Figure 4.2.

Support of fiat currencies is designed to cover all main currencies from around the world starting with USD (U.S. dollars) and ending with JPY (Japanese Yen).

| | | | | |
|---|---|---|---|---|
| Bitcoin | Ethereum | XRP | Bitcoin Cash | LiteCoin |
| EOS | Tron | Monero | Cardano | Stellar |
| Tezos | Neo | Dash | Ethereum Classic | Zet Cash |
| NEM | Dogecoin | Qtum | Bitcoin Gold | 0X |

Table 4.2: Available crypto currencies in the system

## 4.3 Data Collection and Storing Flow

Collection and storing flow of acquired data is a crucial aspect of this Bachelor's thesis. I have had to come up with a design that would collect and store a large amount of data in a database at regular time intervals.

The biggest obstacle was to come up with the design so that the system will not overstep all data resources rate limits. The proposed system's flow is represented by the flow diagram in Figure 4.2.

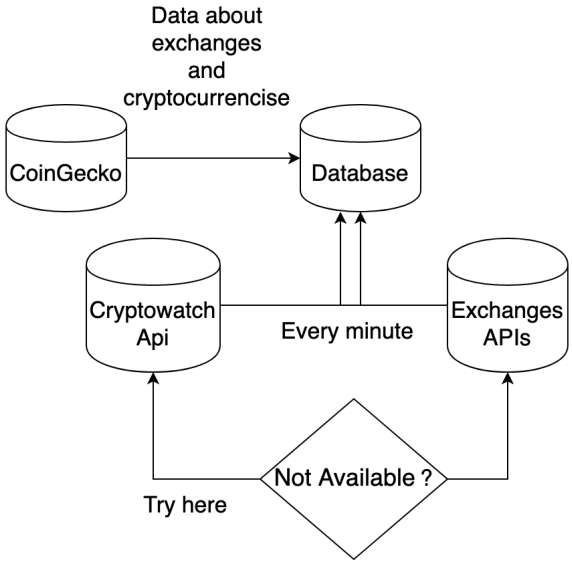



Figure 4.2: The system's flow diagram

As you can see, the core point of the flow diagram is Cryptowatch API and other Exchange APIs. The system will be able to query for OHLC and volume data every minute, which will serve as raw data for additional computed information (Price, Volume market pair shares, and more). If some of the Exchange API would not be available or not responding, there is no problem with fallback for Cryptowatch data. In the worst case, the system could use raw data resources, for example, from the Cryptodatadownload platform. This is not an ideal option because of their data time range sampling (an only hour or daily data, in some cases 5 or 30 minutes data).

Coingecko API serves as a resource for fetching useful pieces of information about cryptocurrency exchanges and cryptocurrencies. However, they are adding more and more endpoints to their base, so more significant usage of this API is not excludable in the future.

## 4.4 Used Tools

### PHP

PHP or its acronym Hypertext Preprocessor is an open-source general-purpose scripting language. It is especially suited for web development and can be embedded into HTML. This language is included in the group of so-called server-side languages. It means that when the client requests information from the server, the server executes the code and sends the result back to the client. One of the core and most significant features of PHP is its support for a wide range of databases. This feature is possible, thanks to one of the database extensions or using abstraction layer PDO (PHP Data Objects).

In conclusion, PHP provides a large number of features, but the best thing about it is that it is relatively easy to learn and is well suited for web development. Therefore, this language is selected for the implementation of the proposed web application backend.

### Laravel

Laravel is a PHP web application framework that attempts to make the development of web applications easier. It follows the structure of the MVC (Model View Controller), which is a software design pattern commonly used in developing user interfaces. Laravel comes with a large number of features, some of which are as follows:

- **Dependency management**: functionality provided by IoC (Inversion of control) or Service Container, which is a tool for managing class dependencies.
- **Modularity**: ability to split business logic to separate modules that all work together to make a web application functional.
- **Routing**: can be used to create a restful application with the ability to group routes, apply filters, or bind data model to them.
- **Restful controllers**: enable to separate the logic behind serving GET or POST requests and create resource controllers that can be used to develop CRUD quickly.
- **Testing and debugging**: support for testing with PHPUnit is included out of the box.
- **Database query builder**: provides a convenient way to create database queries with helper functions which can be used to filter down the data.
- **Eloquent object-relational mapping(ORM)**: provides support for almost all database engines. It means that each database table has its own 'Model', which is used to interact with that table.

The above features are far from everything that Laravel offers, but they are the most important ones for the development of this application. Another features that one can be interested in are for example **Authentication**, **Caching** or **Security**.

### Javascript

Javascript is a scripting language that helps to create interactive web pages. In contrast with PHP, this language belongs to the group of client-side languages. It means that it runs

in the user's web browser without the need for any resources from a web server. Javascript is also a prototype-base, dynamic language with the support of object-oriented programming styles.

### Vue

Vue is a progressive framework for building user interfaces which is perfectly capable of building single page applications. Among its main features belongs declarative rendering, conditionals, comfortable handling of user input and component system.

### Cron

Cron serves as a time-based job scheduler in Unix-like computer operating systems. This software utility is used for running periodically scheduled jobs in fixed times, dates, or intervals and is most suitable for repetitive tasks.

Cron is driven by crontab, which is a file used to schedule the execution of programs. It contains instructions for the cron daemon. It is important to say that commands defined in any given crontab are executed under the user who owns that particular crontab. The format of time and date fields is {**minute/hour/day of month/month/day of week**}. There also exist some special time specifications, which replace the five initial time and date fields (once a year, once a month, and so on) [10]. Here are some examples of scheduled jobs for specific time and date:

- **00 00 * * ***: every day at 00:00.
- **5 4 * * sun**: at 04:05 on Sunday.
- **@weekly**: at 00:00 on Sunday.

# Chapter 5

# Implementation

The implementation of the proposed system from Chapter 4 can be divided into four main parts. Data acquisition, API implementation, database setup, and user interface implementation.

This section explains techniques and procedures which have been used during the implementation process in each of these parts.

## 5.1 Data Acquisition

### Cryptocurrencies

As was mentioned in Chapter 4 the system supports certain number of cryptocurrency exchanges and cryptocurrency coins. The data are collected in the form of OHLC candlestick and volume amount and stored to the database with a link to specific exchange, timestamp, and market pair from Cryptowatch or specific exchange API.

Parsing of the returned data and sending this data to the internal API handles module that exists for every exchange separately. For the Cryptowatch use case, the process of collecting and storing is pretty much the same. It is just aggregated to one module for every exchange. Every module responsible for this kind of task is located in **app/Modules/** directory.

The data collection process is triggered every minute, which means that the system is working with one-minute data for every computed property it provides. This periodicity is handled by a time-based job scheduler utility cron and Laravel scheduler. The process can be expressed as follows. Every minute, only one cron job is executed. Then, this cron job will call the Laravel command scheduler, and when the scheduler is executed, Laravel will perform every scheduled task defined in **App/Console/Kernel** class that is due.

Why not define all tasks in one cron file? This has two simple reasons. First, the developer has source control for every scheduled task set in Laravel, so he does not have to access the server at all if he wants to edit something. Second, one can execute some additional functions on top of the scheduled command, for example, where to send generated output, task hooks (code executed before, after, onSuccess or onFailure), and much more.

### Fiat

The purpose of this Bachelor's thesis is to collect data about different cryptocurrency market pairs and their prices and to be able to convert them to any other national currency

if possible. Therefore, the system can collect data about current and historical foreign exchange rates published by the European Central Bank. These data are collected every day at 14:05 UTC (official time of updating at 16:00 CET) from Foreign exchange rates API.

Cases where the rates are not updated (special days like Christmas, etc.) are handled by checking timestamp in response, and if the timestamp does not match the current date timestamp, then the returned data are stored for the present day regardless. On the one hand, this causes the same data are stored in the database for more timestamps. On the other hand, it is a crucial feature to have these data stored for every day, as you will see in Subsection 5.3. Scheduling of storing process of such data is implemented the same as for cryptocurrencies data explained above.

## 5.2 API

An application programming interface (API) is often used as a middleman between user and database interactions. It is not otherwise in this thesis, as well. The resulting system provides REST API[1] with different endpoints to interact with collected data described in Section 5.1. All endpoints are defined in **routes/api.php**. For every endpoint exists a single public method, all implemented in **app/Http/Controllers/ApiController.php** file. In Laravel, controllers are classes that can group related requests logic into one class. Individual endpoints can be divided into four main groups:

- **Server**: informations about server, just **ping** for now;
- **Asset**: a group of endpoints which provide data about individual supported; cryptocurrency assets such as global market value, circulating or total supply and more
- **Exchange**: provide data about cryptocurrency exchanges for example progress of specific cryptocurrency asset on their market, OHLC and volume data and some basic information about given exchange;
- **Fiat**: informations about foreign exchange rates.

### 5.2.1 Methodology

In this subsection, I would like to summarize how the system calculates and evaluates different metrics available in this API.

#### Cryptocurrency Price

Price for a specific crypto asset is calculated based on the pairings available and collected by the system. The price returned by the API is calculated using a global volume-weighted average price (VWAP) formula and sampled by specified time range. That means that if you want to get most recent price for specific crypto asset you would have to specify current date timestamps (start and end) and one minute time range (note that one minute time range is smallest time unit supported by the system) and pick the last value.

**VWAP** is calculated by multiplying exchange average price (Average of Open, High, Low divided by 3) and volume divided by total asset volume for specified time range.

[1] https://app.swaggerhub.com/apis-docs/xcalad01/Bakalarka/0.1

$$VWAP = \frac{\sum Price * Volume\_exchange}{\sum Volume}$$

**Example: BTC to USD**

Suppose that the system tracks only 2 exchanges A and B. Exchange A trades at BTC vs. USD market pair and exchange B trades at BTC vs. EUR market pair.

**Exchange A**: BTC/USD = 7000 USD & 1000 BTC Trading Volume (in specified time range)
**Exchange B**: BTC/EUR = 6400 EUR & 500 BTC Trading Volume (in specified time range)
**Exchange B**: EUR/USD = 0.9 & BTC/EUR = 6400 EUR & BTC/USD = 7111.11 USD

**BTC price in USD**:

$$VWAP = \frac{(7000 * 1000) + (7111.11 * 500)}{1000 + 500} = 7037.03666667$$

This kind of crypto asset price can be acquired from **/crypto/historical/asset/value/{from}/{to}/{start}/{end}/{range}** endpoint. **Range** query parameter is optional and if not provided the price is computed and sampled in **end** - **start** time range.

**Volume by Currency**

This metric defines exchange volume shares of all available currencies in which individual crypto assets are traded. Calculation of this metric can be divided into following points:

- Make volume sum of all individual market pairs traded on given exchange grouped by currency to which they are traded and convert the sum to unified currency (in this case to USD).
- Calculate total volume of all market pairs on given exchange, converted to unified currency as well.
- Calculate the exchange volume share of individual grouped volume sums by currency divided by total volume sum.

**Example: Exchange A**

Suppose that Exchange A trades in only two pairs, BTC vs. USD and ETH vs. EUR.

**BTC/USD volume**: 1000 & BTC/USD = 7000 USD & 7000 000 USD volume.

**ETH/EUR volume**: 1000 & ETH/EUR = 250 & USD/EUR = 0.9 & ETH/USD = 225 & 225 000 USD volume.

**Total volume**: 925000 USD.

And the final share is 75.67 % for USD and 24.32 % for EUR. This cryptocurrency exchange information can be found on **exchange/stats/{exchange}** endpoint and also seen on exchange's quick stats dashboard of system user interface (UI) (example on Figure 5.1).

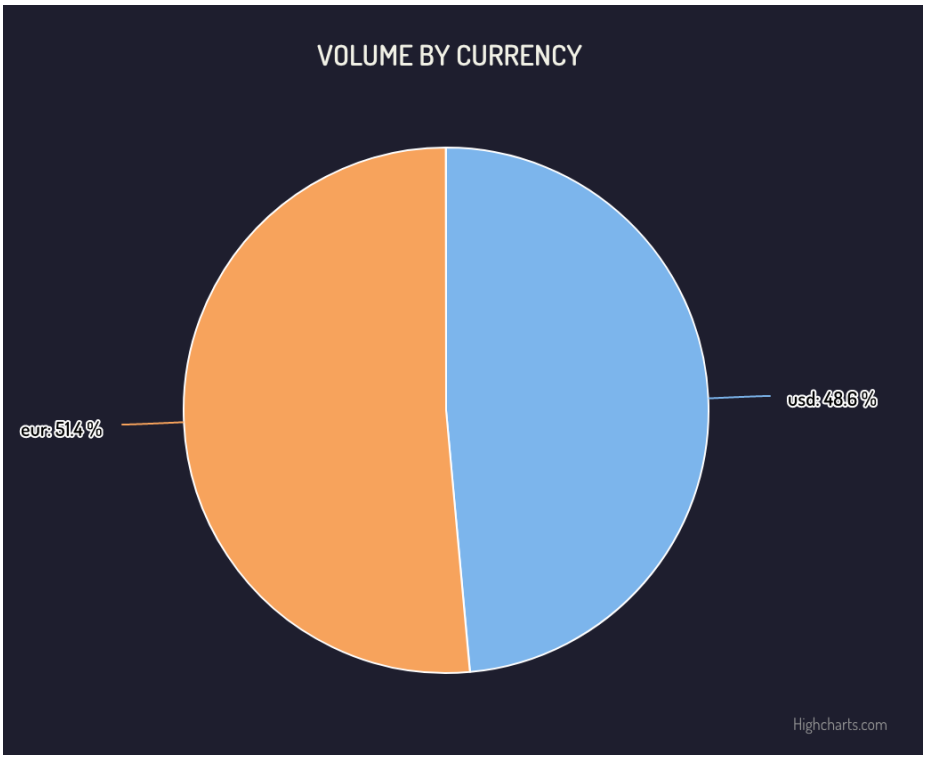



Figure 5.1: Volume by currency on Kraken exchange

### Volume by Market Pair

Metric, which provides exchange volume shares by supported market pairs on that specific exchange. Calculation of this information can be divided into the following groups as well:

- Make volume sum grouped by exchange market pairs and convert it to USD.
- Calculate the total volume of all market pairs on the given exchange and convert it to USD.
- Calculate the exchange volume share of individual grouped volume sums by market pair divided by total volume sum.

Following from previous subsection example for **Exchange A**:

**BTC/USD volume share**: BTC/USD volume = 700000 USD & BTC/USD volume % Total Volume & 700000 % 925000 * 100 = 75.67 %

**ETH/EUR volume share**: ETH/EUR volume = 225000 USD & BTC/USD volume % Total Volume & 225000 % 925000 * 100 = 24.32 %

Volume by market pair for given exchange can be found, again, on **exchange/stats/{exchange}** endpoint and seen on exchange's quick stats dashboard of system UI (example on Figure 5.2).

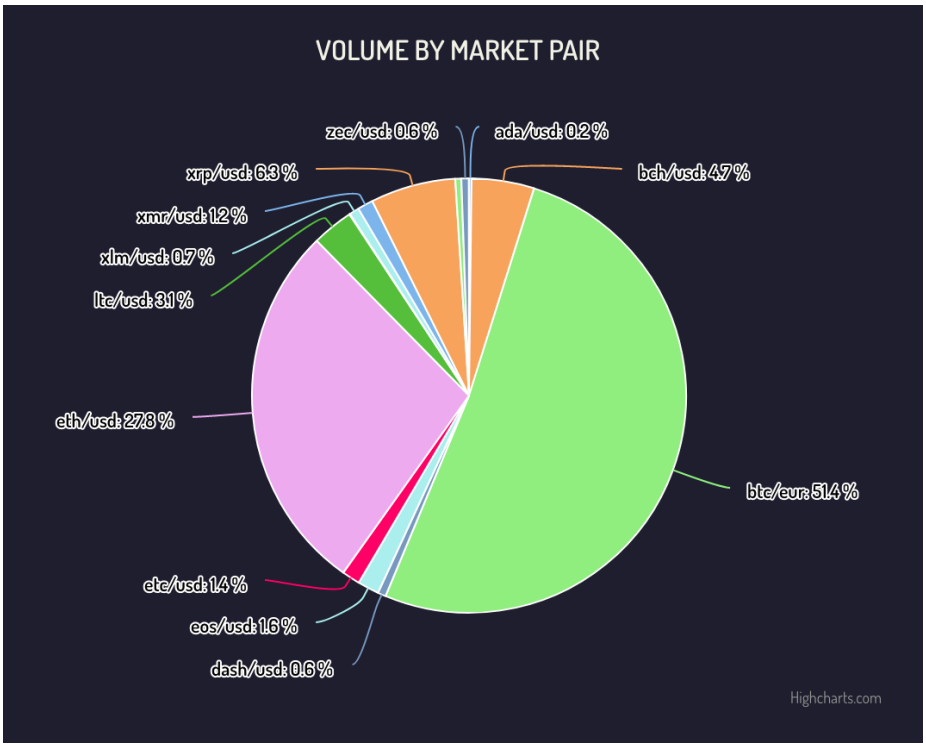



Figure 5.2: Volume by market pair on Kraken exchange

### Others

I do not find other metrics, such as average cryptocurrency value or OHLC data calculated and sampled in a specified time range, interesting topics enough to include them in this methodology section. I think that their calculation is pretty straightforward. Basically, it is just average of different computation methodologies (e.g for average value: (Open + High + Close) / 3) sampled into specific time range.

## 5.3 Database

Database setup and configuration is another crucial aspect of this Bachelor's thesis. To store the data described in a few sections above, I have had to come up with a schema containing more than one table. This indicates that to extract meaningful information from raw OHLC and volume data, more advanced methods than just simple select needed to be used.

But, before any database can be used, the database schema needs to be created. This was mainly achieved by Laravel migrations, which is a type of version control for databases. To create a new database table, one may use the **make:migration** command defined on

Artisan CLI[2], where additional parameters can be specified such as to which table the migration belongs or if the migration should create a new table. So, for example, to create a new and single table that would be storing data about cryptocurrency exchanges (Id, Name, Image url, and Url), I would use the following command:

```
php artisan make:migration add_exchanges_table --table=exchanges
```

This command creates single file in **database/migrations** folder, which contains two functions **up** and **down** and timestamp for artisan to know the order of individual migrations. The contents of this file for example above is as following:

```
public function up()
{
    Schema::create('exchanges', function (Blueprint $table) {
        $table->string('Exchange_id');
        $table->string('Name');
        $table->string('Image');
        $table->string('Url');

        $table->primary('Exchange_id');
    });
}

public function down()
{
    Schema::dropIfExists('exchanges');
}
```

The **up** function creates individual table columns and specifies the primary key. The **down** function serves for situations when migration rollback is required. After creating similar migration files for all database tables, simple artisan command **make:migration** can be used to execute all defined migrations. After the execution of this command, the database schema with all its tables is created directly on the database server and ready to be used.

In conclusion, I need to say that the migration concept, not in just Laravel but any other framework, is a very powerful way of handling databases due to its abstraction and simplicity of usage.

### Data Extraction Methodology

To extract meaningful information from big OHLC and volume dataset, the queries need to be as efficient as possible. When creating these queries, the basic idea was to calculate everything on the database side so that the client can count only with final result. This includes sampling data to different time ranges or calculating the average OHLC and volume data for these time ranges. Another important feature, conversion to another foreign currency, is also included in these queries.

[2] https://laravel.com/docs/5.0/artisan

As you will see, lot of queries use Postgres so called common table expressions[3] (CTEs or WITH statements). They can be thought as temporary tables which exist just for the query where they are defined. Following example demonstrates usage of such statement:

```
 WITH offset_value (
    offset_val
) AS (
    values('{$start}' - floor((extract('epoch' FROM to_timestamp(
    '{$start}')) /
    $ RANGE)) * $ RANGE)
)
```

The example above is a real statement, which is used in almost every query of the system. It basically calculates time range offset in seconds so that the resulting time ranges will be accordingly aligned.

To demonstrate the most interesting queries, from my point of view, I need to define 3 small example tables:

- **historical available**:

| id | From | To | Exchange_id |
|---|---|---|---|
| 4 | btc | usd | bitfinex |

Table 5.1: Example of btc vs. usd market pair on bitfinex exchange

- **fiat historical**:

| Date | Fiat_id | Value_USD |
|---|---|---|
| 1589558700 | usd | 1 |
| 1589558700 | eur | 0.9173470324 |

Table 5.2: Example of fiat historical table, all values in USD

- **crypto historical**:

| id | Timestamp | Open | High | Low | Close | Volume |
|---|---|---|---|---|---|---|
| 4 | 1589640780 | 9415 | 9415 | 9407.6 | 9415 | 2.3848 |
| 4 | 1589640720 | 9415 | 9415 | 9413.4 | 9415 | 5.4554 |
| 4 | 1589640660 | 9410 | 9415 | 9409.9 | 9415 | 5.3035 |

Table 5.3: Example of crypto historical table, OHLC and volume data

### Open, High, Close Average Price

Full open, high, close (OHC) average price query is available in Appendix C. The core of this query is three-table JOIN so that after this operation it will be practically pretty easy to calculate the price and convert it to desired currency. Query flow can be expressed as follows:

[3] https://www.postgresql.org/docs/9.1/queries-with.html

- Join **crypto historical** and **historical available** tables by **id** (Table **A**).
- Join **fiat historical** and table **A** by **Fiat_id** and **To** column (Table **B**).
- Join table **B** with **fiat historical** where **Fiat_id** is equal to some other currency to which should be final price converted (Table **C**).
- Filter table **C** by conditions defined in where clause.
- Group the filtered result by specified time range, calculate the average of open, high, close summary columns divided by three and convert to specified currency.

Similar approach is applied in OHLC and Volume queries, the values are taken, averaged and aggregated to groups by timestamps. The result of this queries can be single pair or set of pairs of **Timestamp**, **Value** (Open, High, Low, Close for OHLC query) format.

#### VWAP Price

This query is basically rewritten formula for **vwap** from Subsection 5.2.1 to SQL notation. For the most part, it uses already mentioned CTEs, which create three more additional tables **main_table**, **sum_table_volume** and **sum_table_volume_price**.

The **main_table** query is very similar to already discussed **ohc** query and just minimal adjustments were made in there. Another table, **sum_table_volume**, represents denominator part, whereas **sum_table_volume_price** represents dividend part of the formula. In the end, these two tables are grouped by a specified time range, and the resulting value is calculated by the division of values from these tables. Note that without using so-called common table expressions, the query would probably be using sub-selects commands, which are, in my opinion, much harder to read than the CTE approach.

As it was in **ohc** query, the result of this query is pair or set of pairs in **Timestamp**, **Value** format, it depends on specified time range and interval. The full SQL code for this query can be found in Appendix C.

### Conclusion

As you can see, to extract desired pieces of information from the raw dataset, little more advanced queries needed to be used. The strong feature of these queries is that they can do more things than extract and aggregate data based on some conditions. The performance evaluation can be found in Section 6.

## 5.4 User Interface

To visualize the acquired data described above, the part of this Bachelor's thesis is to implement a user interface that would provide information about cryptocurrencies and cryptocurrency exchanges in the form of charts and other visualizing resources. The leading technologies used to achieve this goal are, for the most part, the Vue framework and Javascript.

The user interface is implemented in the form of dashboards, where the first dashboard displays information about exchanges and the second dashboard about cryptocurrency assets. Further, each dashboard is divided into individual components. In Vue, the component is reusable Vue instance with its own name, variables, and methods. These instances are

handy for building single-page web applications, where you can pretty quickly lay out the page with them. You can see the example dashboard for Bitcoin asset and the following components deployment on Figure 5.3.

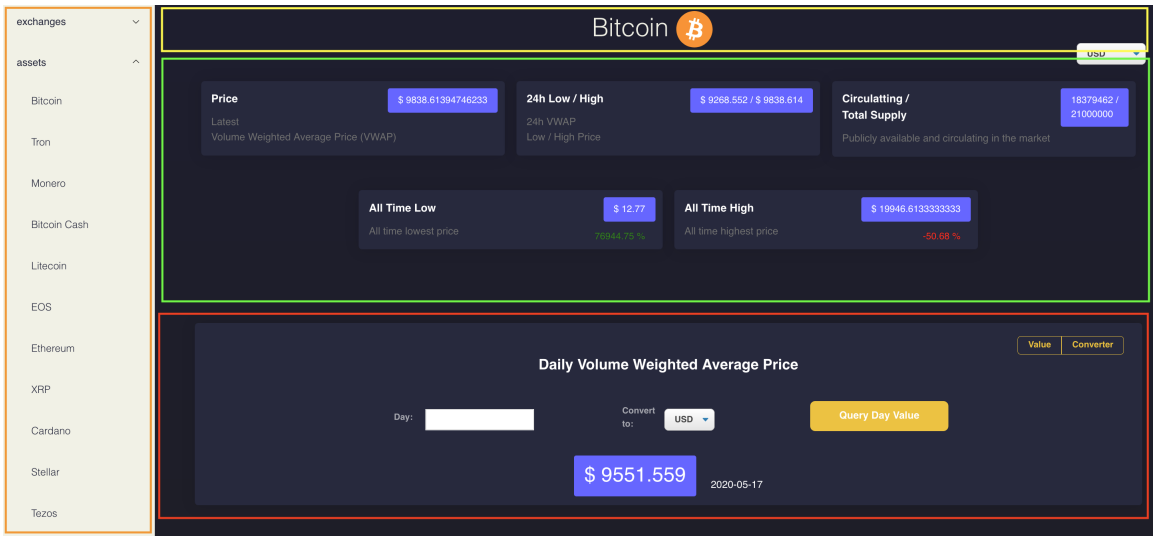



Figure 5.3: Bitcoin dashboard example

# Chapter 6

# Testing

This chapter explains different testing experiments that were done against the resulting system. These experiments provide insight into system features, for example, the validity of computed crypto asset prices against other platform's prices. Further, the chapter is focused on the performance evaluation of the implemented API and database.

## 6.1 Prices

The main functionality of the implemented system is to provide prices about different cryptocurrency assets. Therefore, they needed to be somehow tested and validated, to make sure that the prices for specific asset correlate with prices on other platforms which provide such data. It follows that if they did not correlate, the system would be practically unusable.

### Realtime

Validation of realtime price computed property was executed as follows. I have created a job that would collect fresh prices from different cryptocurrency platforms at regular intervals and compare them to the price computed by the system. The development of the individual price values grouped by platforms from which the data were collected is provided in Figure 6.1. As you can see, the prices approximately correlate with each other, including our price, which is a good sign. But, little more interesting data can be extracted from this kind of dataset.

Figure 6.2 describes the average percentage difference between our price and prices from other platforms. And the results are, again, very optimistic. The percentage difference is moving somewhere between **0%** and **0.4%**. Even though these values are not great, note that the difference is computed for BTC vs. USD market pair. This means that if the difference is **0.2%**, and the price is equal to **9000$**, the difference value would be **18$**. At first glance, this value does not look very promising. But, as you can see in Figure 6.3, the actual price differences are sometimes much higher. This figure also reveals that the average price difference between our price and other platforms prices is often smaller than the average price difference between individual platforms. This could mean that our price has a very good correlation with each market from the experiment dataset.


Figure 6.1: BTC vs. USD price development on different platforms

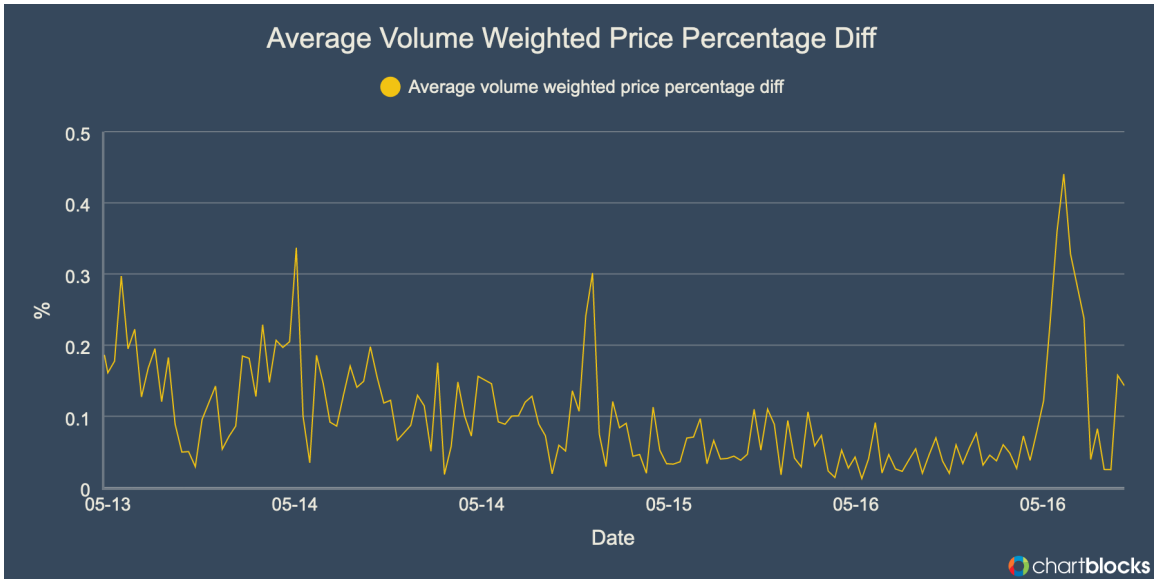



Figure 6.2: BTC vs. USD average percentage difference

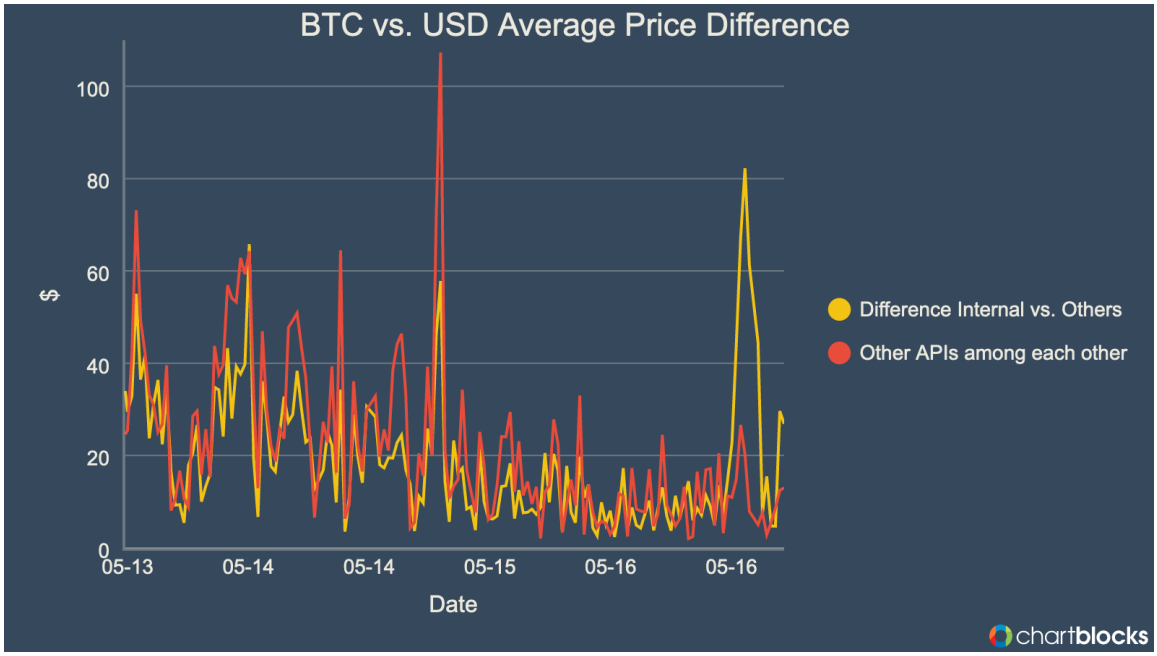



Figure 6.3: BTC vs. USD average price difference

If I combine all the arguments provided above, I can declare that the prices, which are computed and displayed on the system UI when the page is loaded or refreshed (latest available price for the current timestamp) are valid.

### History

Providing history data about cryptocurrency prices is another important feature of the implemented system. The validity of the historical crypto-asset price for a random day was tested by collecting data from internal, CoinGecko, and CoinBase APIs and comparing them with each other. You can see an example of such data in Table 6.1.

| | **2018-12-11** | **2014-09-24** | **2017-12-11** | **2019-10-26** | **2020-04-05** |
|---|---|---|---|---|---|
| **InternalPrice** | 3416.96 $ | 429.87 $ | 16467.00 $ | 9324.87 $ | 6794.40 $ |
| **CoinGeckoPrice** | 3433.81 $ | 420.92 $ | 17106.51 $ | 8652.03 $ | 6859.42 $ |
| **CoinGeckoPricePercDiff** | 0.25 % | 1.05 % | 1.90 % | 3.74 % | 0.48 % |
| **Coinbase** | 3365.79 $ | 429.58 $ | 16697.32 $ | 9322.00 $ | 6798.43 $ |
| **CoinbasePricePercDiff** | 0.75 % | 0.03 % | 0.69 % | 0.02 % | 0.02 % |

Table 6.1: Internal, CoinGecko and CoinBase prices in BTC/USD market pair

In some cases, the data differ a lot. As all prices are computed based on the global average volume-weighted price formula, and the price mostly depends on the volume, this can be caused by the fact that individual platforms collect different data from different sources. This means that if platform A collects data from source X and platform B does not, and source X has a big volume share for specific market pair and day, the price could be potentially affected more on platform A than on platform B. The average price difference between internal API price and other APIs from this experiment dataset is **63.5$**. If we compare this value to Figure 6.3 from the previous experiment, we can see that this value belongs somewhere in the middle.

In general, I would say that historical prices provided by the system can be considered valid and that the comparison result depends on platforms to which you compare the prices.

## 6.2 Database

Database testing was performed continuously with the development of the project with tools such as PostgreSQL **explain**[1] command or Datadog **APM**[2] service.

PostgreSQL **explain** command displays execution plan that the planner generates for a given query. You can specify different additional parameters, but mostly, I was using just the **analyze** parameter. Analyze additional functionality is that the explain command does not just generate the plan, but it actually executes the query and prints out the actual number of result rows, execution costs, and much more useful information.

Datadog APM is a service that collects information about your application in the form of traces. It basically traces all your application services (or those that you specify) and logs the information such as average latency, error rate, number of requests, and others, which could be useful when you want to understand your application bottlenecks.

### Raw vs. Eloquent

This experiment's goal is to determine which technique is better in terms of performance for the database select operation. Insert and update operations were omitted from the reason that the implemented system does not update almost anything, and insert operations are executed on a small group of data per time unit, so the results would not be so interesting.

The experiment was executed as follows. First, I have evaluated the query with the biggest average latency value on database service with the help of already mentioned Datadog **APM**. Then I connected this query (Average volume-weighted price C) with a specific application endpoint where it is used the most. After that, I started shooting requests to the API (for different time intervals and market pairs) and collecting information about the query duration times. The result of this benchmark can be seen in Figure 6.4.

[1] https://www.postgresql.org/docs/9.1/sql-explain.html
[2] https://www.datadoghq.com/apm/

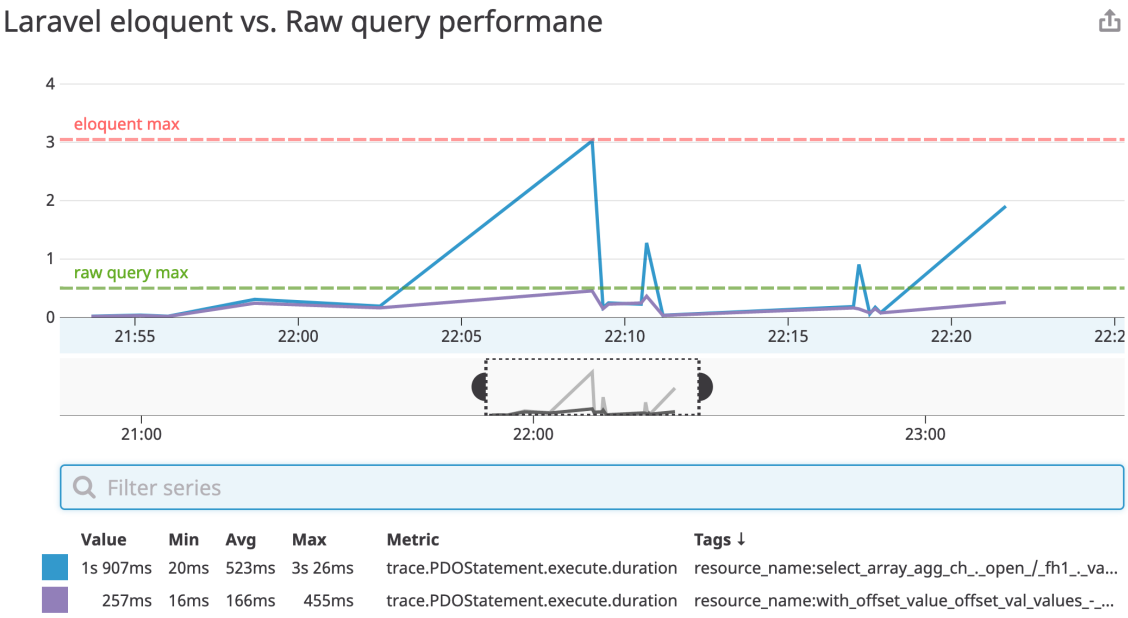



Figure 6.4: Laravel eloquent vs. Raw query benchmark

The above picture shows the comparison result of query durations for Laravel Eloquent (blue) and Raw SQL (purple). As you can see, Raw SQL commands are much faster than Eloquent. The duration value extreme (min, max) and average values can be found in the bottom left corner of the picture. From these values, you can get a pretty good understanding of the benchmark result. The minimum values are almost identical, but the difference between maximum values is **2805** milliseconds. Although the most interesting are average values, it can be understood from these values that in this example, the Raw SQL queries were four times faster on average than the Laravel Eloquent queries.

The above experiment was performed on a large OHLC and volume dataset with more than seven years of historical data. So let's compare these two approaches with a significantly smaller dataset and less complex query. The use case is simple, select everything from **exchanges** table where **Exchange_id** is equal to some value. As you can see in Figure 6.5, the result is very similar to the experiment above. The Raw SQL query (purple) is, again, almost four times faster on average than the Laravel Eloquent query (blue). But in this case, Eloquent queries are usable as well because the average duration time for this approach is only **45** milliseconds.

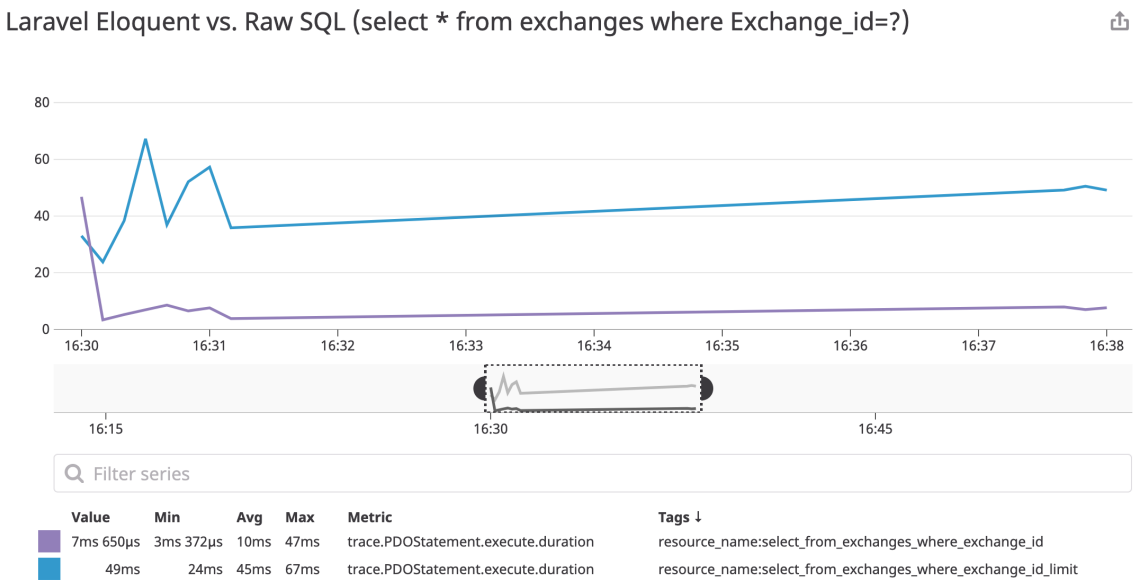



Figure 6.5: Laravel eloquent vs. Raw SQL benchmark

In conclusion, I would say that I proved the fact that in terms of performance of SQL select operations, the Raw SQL queries does not have any concurrency. It became evident that Laravel Eloquent is not suitable for large applications with a massive amount of data and where many requests to the database at a time are involved. We can conclude that Eloquent ORM is suitable for building applications with small to medium-sized database, where it would be still slower than Raw SQL but at least usable.

### Explain

The usage of **explain** or **explain analyze** command is very simple. You just have to prepend it to the query you want to analyze. So the command would look something like this: **explain / explain analyze {query}** and the resulting output would be in **text** format.

Following is an example of such command.

```
explain analyze select * from exchanges;
```

Listing 6.1: Example of explain analyze command

And the result is beneath.

```
Seq Scan on exchanges (cost=0.00..2.13 rows=13 width=240)
(actual time=0.025..0.032 rows=13 loops=1)
Planning time: 0.054 ms
Execution time: 0.051 ms
```

Listing 6.2: Example of result of explain analyze command

In the example above, the PostgreSQL planner generated a plan with a sequential scan on the whole **exchanges** table with an estimated cost between 0 and 2.13 units with thirteen resulting rows. The actual sequential scan time was somewhere between 0.025

and 0.032 milliseconds with thirteen resulting rows, and the scan was executed exactly once (loops=1). The execution of the whole command took 0.051 milliseconds, whereas planning time is reported to 0.054 milliseconds.

### Query Tuning

One of the most slowest part in the system queries in terms of average latency is place where the two tables **crypto_historical** and **historical_available** would be joined based on some filtering conditions. Because of that, I have decided to highlight several features which I found out with help of PostgreSQL explain command. The tuning was run in following steps:

- **default PostgreSQL config with no indexes**:

  With default config and no indexes used the part of an explain analyze output is looking like this:

```
Workers Planned: 2
Workers Launched: 2
...
Parallel Seq Scan on crypto_historical ch
(cost=0.00..330816.44 rows=122871 width=28)
(actual time=76.002..1017.697 rows=96730 loops=3)
"Filter: (((""Timestamp"" >= 1587411420) AND
((""Timestamp"" <= 1587670619))"
...
Planning time: 1.116 ms
Execution time: 1551.646 ms
```

Listing 6.3: Example of explain analyze result 2 workers

  In the example above, the planner used **parallel sequential scan** on **crypto_historical** table with 2 workers. In PostgreSQL, a parallel sequential scan means that the table will be scanned row by row and filtered by **Filter** condition. You can imagine that if the table contains millions of rows, this approach is not very effective. So, can we somehow tune this process to be more efficient (faster)?

- **max_parallel_workers_per_gather**[3]:

  From version **9.6**, PostgreSQL provides special **max_parallel_workers_per_gather** variable. This variable defines maximum number of workers that can be started in single gather. The variable is only limited by another variable **max_parallel_workers** that defines maximum number of workers per all parallel processes combined. The default variable value is set to **2** on system startup. So, I tried to double it and the result was as follows:

[3] https://www.postgresql.org/docs/10/runtime-config-resource.html#GUC-MAX-PARALLEL-WORKERS

```
Workers Planned: 4
Workers Launched: 4
...
Parallel Seq Scan on crypto_historical ch
(cost=0.00..281036.36 rows=73726 width=28)
(actual time=62.795..735.941 rows=58038 loops=5)
"Filter: (((""Timestamp"" >= 1587411420) AND
((""Timestamp"" <= 1587670619))"
...
Planning time: 2.331 ms
Execution time: 1301.469 ms
```

Listing 6.4: Example of explain analyze result 4 workers

On average, the queries were just as fast or a little faster than the queries with the default config. However, I did not manage to tweak the PostgreSQL config to the state where these parallel queries would be markedly faster.

This could be caused by the fact that in PostgreSQL parallel joins (and the joins are used in many system queries), the planner would rather choose a method that does not contain parallelism when too large tables or sub-tables are joined[4]. Following is an example of a one-day time range when querying for price value in two months' time interval.

```
Seq Scan on crypto_historical ch
(cost=0.00..506565.79 rows=2673839 width=28)
(actual time=0.076..2940.549 rows=2578679 loops=1)
"Filter: (((""Timestamp"" >= 1584662400) AND
((""Timestamp"" <= 1587670619))"
Rows Removed by Filter: 17392446
...
Planning time: 2.244 ms
Execution time: 10120.997 ms
```

Listing 6.5: Example of explain analyze result Sequential scan on big tables join

And as you can see, the planner did indeed use the normal sequential scan rather than the parallel one.

- **adding index**

  The last thing I tried was to use indexes on these tables. The index for the **crypto_historical** table is composed from **id** and **Timestamp** columns, in that particular order. It is important to understand that if this index was in reversed order than it would most probably not work because, first, the tables are joined by **id** and than filtered by **Timestamp**, and the result would be that the planner would most probably use sequential scan on **crypto_historical** table.

[4] https://www.postgresql.org/docs/10/parallel-plans.html

Beneath is part of the **explain** command, when using the index, for the same query as in the experiments above.

```
...
Bitmap Heap Scan on crypto_historical ch
(cost=1118.23..75255.68 rows=26484 width=28)
(actual time=0.548..2.265 rows=9619 loops=1)
"Recheck Cond: ((id = ha.id) AND
(""Timestamp"" >= 1584662400) AND
(""Timestamp"" <= 1587670619))"
Heap Blocks: exact=139
Bitmap Index Scan on crypto_historical_pkey
(cost=0.00..1111.61 rows=26484 width=0)
(actual time=0.530..0.530 rows=9619 loops=1)
"Index Cond: ((id = ha.id) AND (
""Timestamp"" >= 1584662400) AND
(""Timestamp"" <= 1587670619))"
...
Planning time: 1.311 ms
Execution time: 84.357 ms
```

Listing 6.6: Example of explain analyze result when using index

As you can see, the execution time is almost eighteen times shorter by adding a simple multicolumn index. The result is that in cases where it is suitable, trying to force the query to use an index scan is probably the best solution you can find. But, you have to be careful when adding new indexes to your database schema because each index has a cost to create and maintain (on writes) and to use (on reads). This is also a reason why the planner could choose sequential scan against the index scan in some cases.[5]

## 6.3 API

The implemented API was mainly tested in terms of the validity of the returned data and the handling of unexpected situations. That means that if the user requests data from exchange X in given timestamp intervals and time range, the returned data will be correctly sampled and timed. On the other hand, if the user requests data from the non-existing resource (unsupported coin or exchange) or the time parameters are incorrect, he will be noticed about it.

The second part of the API testing is focused on API performance and if it can handle a bigger load. First, the API was tested with two thousand requests on a relatively fast endpoint with different timestamps, time ranges, currencies to convert to, and so. All requests were resolved in around seven minutes. That means that one request was resolved in 210 milliseconds on average, which is not bad at all.

The second benchmark was executed on various endpoints (relatively fast and slower as well) with the same configuration as in the benchmark above. Just 418 requests were

[5] https://www.postgresql.org/docs/10/using-explain.html

resolved in fifteen minutes, which makes 2153 milliseconds for each request on average. This can be caused by two things. Unfortunately, the API runs at synchronous mode, so no asynchronous framework is used. That could mean that tasks that are expected to run longer than other tasks could block those tasks from executing, which can escalate to smaller throughput or bigger latency. The second problem could be a lack of hardware resources as this project currently runs on a custom virtual private server. It would be really interesting to see how it would act on more powerful servers.

# Chapter 7

# Conclusion

The goal of this Bachelor's thesis was to create a web application that has the ability to provide actual and historical information about cryptocurrency to Fiat money or cryptocurrency to cryptocurrency exchange rates with link to the specific exchange platform. To achieve this, I have had to analyze available resources providing such data and start to gather them at regular time intervals. These data are then transformed into a database and provided to the user in the form of charts and different types of statistics.

The first step when solving this thesis was to gather the literature about underlying principles of cryptocurrencies and their differences with Fiat money. Then, I collected different pieces of information about cryptocurrency exchanges, their types, main information components that can be found on every exchange platform, and their security, which is one of the most discussed topics in the cryptocurrency community. Furthermore, I have had to collect information about the basics of trading with cryptocurrencies and different ways of pricing on them. All of the collected knowledge is provided in Chapter 2.

The next step was to analyze available resources such as cryptocurrency APIs or any other platform which is able to provide actual or historical data about different markets. The main focus had to be aimed at the ratio between the provided amount of information and price. The best of them are described and benchmarked in Chapter 3.

After gathering all the necessary information, I have proposed the design of the application, described in Chapter 4. The scheme consists of an ERD diagram, flow diagram, and tools which were used during implementation.

In Chapter 5 is described how were all the proposed ideas implemented, and in Chapter 6, individual parts of the system are evaluated, mostly in terms of performance and validity of the data.

In the end, it is essential to say that the work on this system does not end with the submission of this thesis. My supervisor and I have already discussed some upgrades that could be and will be done in the future. For example, the API needs to be rewritten to asynchronous mode to handle a more massive load. Or, I think it would be proper not to rely on third-party APIs in the future. Instead, web scraping methods for every supported exchange could be used so that if some third-party API were not available, the system would not be affected by that. Other improvements certainly include adding more and more exchanges and assets, so that the system can provide more accurate prices for more assets. I think that the true beauty of this system is that one can never say that it is finished and that there will always be features that we could add or upgrade.

This thesis is a part of a grant for an Integrated platform for the analysis of digital data from security incidents under the Department of Information Systems on the Faculty of Information Technology at Brno, University of Technology[1].

[1] https://www.fit.vut.cz/research/project/1063/

# Bibliography


[1] Bhaskar, N. D. and Chuen, D. L. K. Bitcoin exchanges. In: *Handbook of Digital Currency*. Elsevier, 2015, p. 559–573.

[2] BitcoinWiki contributors. *Altcoin* [online]. 2017 [cit. 2019-12-28]. Available at: https://en.bitcoinwiki.org/wiki/Altcoin.

[3] Chohan, U. W. The Problems of Cryptocurrency Thefts and Exchange Shutdowns. *Available at SSRN 3131702*. 2018.

[4] CoinGecko. *CoinGecko* [online]. 2020 [cit. 2020-01-05]. Available at: https://www.coingecko.com/en.

[5] CoinGecko. *CoinGecko API V3* [online]. 2020 [cit. 2020-01-05]. Available at: https://www.coingecko.com/api/documentations/v3.

[6] CryptoWatch. *Market Data REST API* [online]. 2020 [cit. 2020-01-03]. Available at: https://docs.cryptowat.ch/rest-api/.

[7] Hileman, G. and Rauchs, M. Global cryptocurrency benchmarking study. *Cambridge Centre for Alternative Finance*. 2017, vol. 33.

[8] Islam, M. R., Nor, R. M., Al-Shaikhli, I. F. and Mohammad, K. S. Cryptocurrency vs. Fiat Currency: Architecture, Algorithm, Cashflow Ledger Technology on Emerging Economy: The Influential Facts of Cryptocurrency and Fiat Currency. In: *2018 International Conference on Information and Communication Technology for the Muslim World (ICT4M)*. 2018, p. 69–73.

[9] Narayanan, A., Bonneau, J., Felten, E., Miller, A. and Goldfeder, S. *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton University Press, 2016.

[10] Vixie, P. *Crontab Guru* [online]. 2012-11-22 [cit. 2019-12-31]. Available at: https://crontab.guru/crontab.5.html.

[11] Väin, M. *Foreign exchange rates API with currency conversion* [online]. N.d. [cit. 2019-12-30]. Available at: https://exchangeratesapi.io/.

[12] White, L. H. The market for cryptocurrencies. *Cato J.* HeinOnline. 2015, vol. 35, p. 383.

# Appendix A

# Installation Instructions

For successful instalation, you need to have **docker**[1] and **docker-compose**[2] installed on your machine. The installation consists of the following steps:

1. In root directory run **cp .env.example .env** and edit this file for your needs.
2. Run **docker-compose up -d** command and wait for it to be finished.
3. Connect to **workspace** docker image shell with **docker exec -it workspace bash**.
4. Run the initialize script **chmod +x entrypoint.sh && ./entrypoint.sh**. This script will start the API and cron service, migrate and initialize the database and deploy the UI.

After the series of above commands the API should be available at:

- **http://{MIX_API_URL}:{MIX_API_PORT}**.

And the UI at:

- **http://{MIX_API_URL}/**.

[1] https://www.docker.com/
[2] https://docs.docker.com/compose/

# Appendix B

# CD Contents

The CD **src** file refer to the same content as available in the git[1] repository. The CD consists of this structure:

- **src**: all source files + docker-compose infrastructure
- **xcalad01.pdf**: documentation
- **2020-05-27_dump**: latest database snapshot before submission, created with pg_dump's custom dump format[2].

[1] https://github.com/xcalad01/CryptoExchangeRatesTracker
[2] https://www.postgresql.org/docs/8.3/backup-dump.html

# Appendix C

# Database Queries Examples

### OHC average value on specific exchange

```
 WITH offset_value (
       offset_val
) AS (
       values('{$start}' - floor((extract('epoch' FROM to_timestamp
       ('{$start}')) /
       $ RANGE)) * $ RANGE)
)
SELECT
       (array_agg("ch"."Open" / "fh1"."Value_USD" * "fh2"."Value_USD"
       ORDER BY "ch"."Timestamp" ASC)) [1] AS "Open",
       MAX("ch"."High" / "fh1"."Value_USD" * "fh2"."Value_USD") AS "High",
       MIN("ch"."Low" / "fh1"."Value_USD" * "fh2"."Value_USD") AS "Low",
       (array_agg("ch"."Close" / "fh1"."Value_USD" * "fh2"."Value_USD"
       ORDER BY "Timestamp" DESC)) [1] AS "Close",
       to_timestamp(floor((extract('epoch' FROM

       to_timestamp("ch"."Timestamp")) /
       $ RANGE)) * $ RANGE + offset_val) AT TIME ZONE 'UTC'
       AS "interval_alias"
FROM
       offset_value,
       "historical_available" AS "ha"
       JOIN "crypto_historical" AS "ch" ON "ha"."id" = "ch"."id"
       JOIN "fiat_historicals" AS "fh1" ON "ha"."To" = "fh1"."Fiat_id"
       JOIN "fiat_historicals" AS "fh2" ON '{$to}' = "fh2"."Fiat_id"
WHERE
       "ha"."Exchange_id" = '{$exchange}'
       AND "ha"."From" = '{$historical_available->From}'
       AND "ha"."To" = '{$historical_available->To}'
       AND "ch"."Timestamp" BETWEEN '{$start}'
       AND '{$end}'
       AND "fh1"."Date" BETWEEN "ch"."Timestamp" - 86400
```

```
        AND "ch"."Timestamp"
        AND "fh2"."Date" BETWEEN "ch"."Timestamp" - 86400
        AND "ch"."Timestamp"
GROUP BY
        "interval_alias"
ORDER BY
        "interval_alias";
```

## AVWP Price

```
WITH offset_value (
        offset_val
) AS (
        values('{$start}' - floor((extract('epoch' FROM
        to_timestamp('{$start}')) / $ RANGE)) * $ RANGE)
),
main_table AS (
        SELECT
                SUM("ch"."Volume" * (("Open" + "High" + "Low") / 3))
                AS "Volume_Exchange",
                AVG(("ch"."Open" + "ch"."Close" + "ch"."High") /
                3 / "fh1"."Value_USD" * "fh2"."Value_USD") AS "Price",
                to_timestamp(floor((extract('epoch' FROM
                to_timestamp("ch"."Timestamp")) / $ RANGE)) *
                $ RANGE + "ov".offset_val)
                AT TIME ZONE 'UTC' AS "interval_alias"
        FROM
                "offset_value" AS "ov",
                "crypto_historical" AS "ch"
                JOIN "historical_available" AS "ha" ON "ch"."id" = "ha"."id"
                JOIN "fiat_historicals" AS "fh1" ON
                "ha"."To" = "fh1"."Fiat_id"
                JOIN "fiat_historicals" AS "fh2" ON
                '{$convert_to}' = "fh2"."Fiat_id"
        WHERE
                "ha"."From" = '{$crypto_id}'
                AND "ha"."To" IN('usd',
                        'eur')
                AND "ch"."Timestamp" BETWEEN '{$start}'
                AND '{$end}' - 1
                AND "fh1"."Date" BETWEEN "ch"."Timestamp" - 86400
                AND "ch"."Timestamp"
                AND "fh2"."Date" BETWEEN "ch"."Timestamp" - 86400
                AND "ch"."Timestamp"
        GROUP BY
                "interval_alias",
                "Exchange_id"
```

```
),
sum_table_volume AS (
	SELECT
		"interval_alias",
		SUM("Volume_Exchange") AS "Volume_Sum"
	FROM
		main_table
	GROUP BY
		"interval_alias"
),
sum_table_volume_price AS (
	SELECT
		SUM("Volume_Exchange" * "Price") AS "Price",
		"mt"."interval_alias"
	FROM
		main_table AS "mt"
	GROUP BY
		"mt"."interval_alias"
	ORDER BY
		"mt"."interval_alias"
)
SELECT
	"stvp"."Price" / "stv"."Volume_Sum" AS "sum",
	"stv"."interval_alias"
FROM
	sum_table_volume_price AS "stvp",
	sum_table_volume AS "stv"
WHERE
	"stv"."Volume_Sum" != 0
	AND "stv"."interval_alias" = "stvp"."interval_alias"
ORDER BY
	"stv"."interval_alias";
```

# Appendix D

# UI Examples


Figure D.1: Exchange dashboard with realtime charts


Figure D.2: Exchange dashboard with info charts

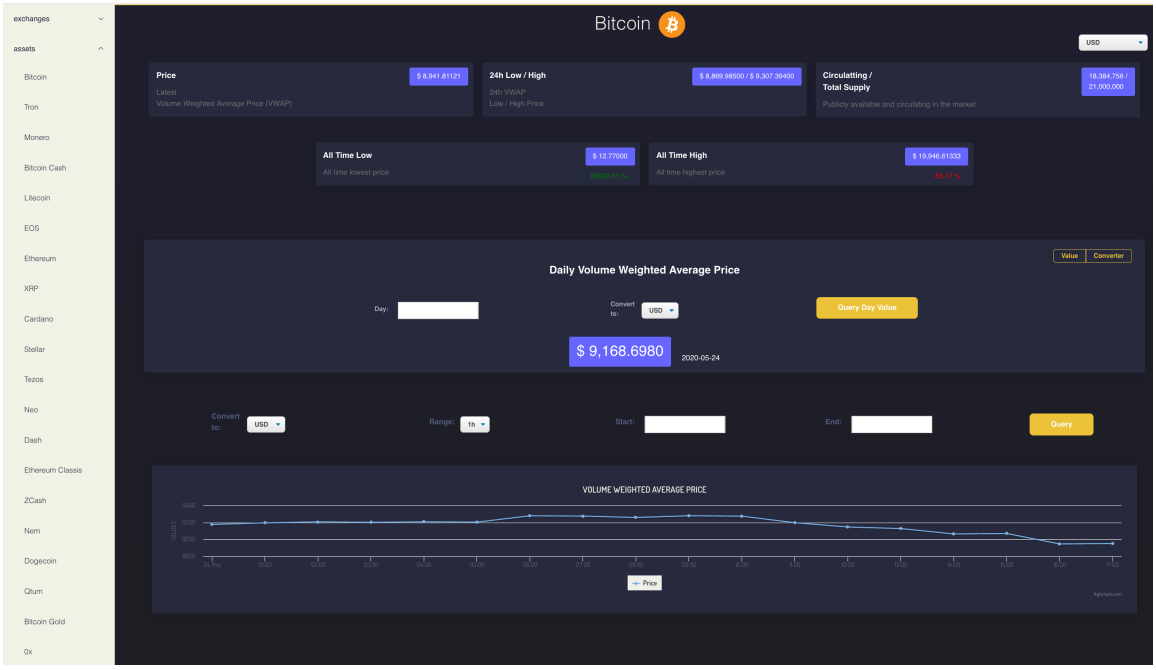



Figure D.3: Asset dashboard with value component

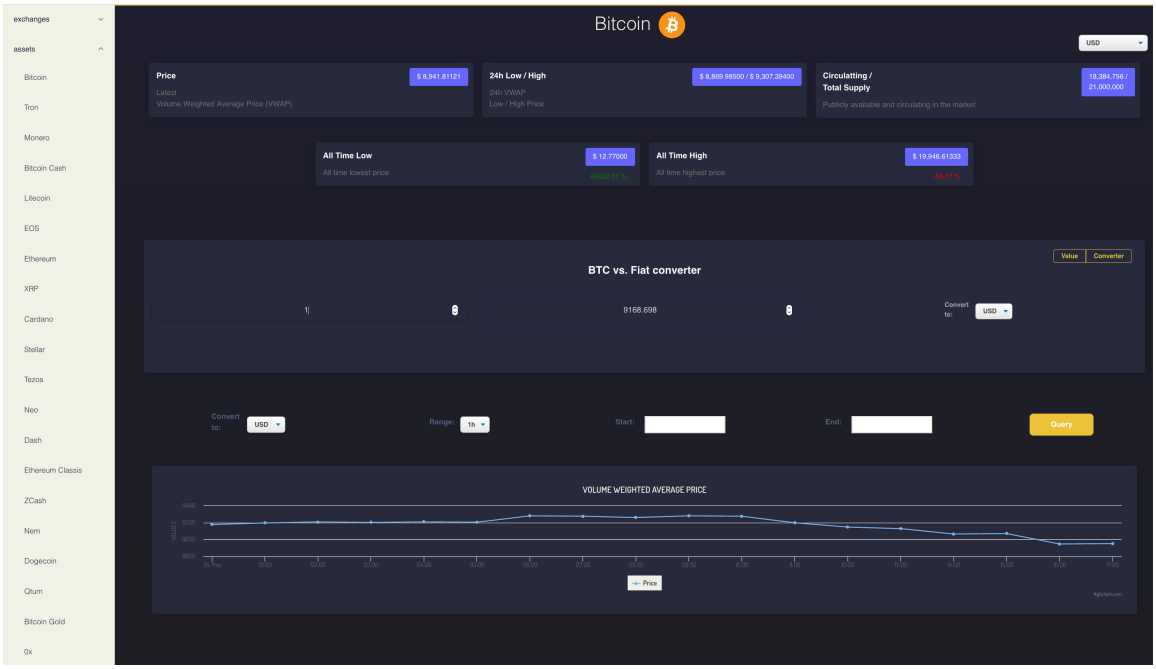



Figure D.4: Asset dashboard with converter component