SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Aplikacija za analizu velikih količina podataka (NoSQL, map/reduce) - polustrukturirane baze podataka - MongoDB - sa sučeljem na LibreOffice Calc

Projektni rad

Tomislav Pertinač
br. indeksa: 45274/16-R

Mentor:
Doc. dr. sc. Markus Schatten

Varaždin, 29. kolovoza 2017.

Sadržaj

Poglavlje 1

Uvod

U ovome projektu je opisan i prikazan princip rada polustrukturiranih baza podataka, te je prikazano na koji način možemo napraviti i prikazati analizu velikih količina podataka. Aplikacija za analizu velikih količina podataka je napravljena pomoću NoSQL baze, odnosno korištena je MongoDB baza podataka. Za programski logiku je korišten programski jezik python, a samo sučelje, odnosno prikaz podataka i analiza je prikazana na LibreOffice tablični kalkulator.
Aplikacija je funkcionalna na Linux platformi, makar uz malu nadogradnju bi lako bila i funkcionalna na Windows platformi. Konretna operacijski sustav koji se koristio za razvijanje aplikacije i na kojem aplikacija radi je Ubuntu 16.04 lts. U ovom projektnom radu upoznat ćemo se što je to big data i što su polustrukturirane baze podataka. Vidjet ćemo što je to NoSQL po čemu je on različit ili sličan sa SQL-om, bit će opisan MongoDB te će bit prikazana arhitektura, implementacija te primjer korištenja same aplikacije.

Poglavlje 2

Big Data

Big Data opisuje ogromne količine strukturiranih ili nestrukturiranih podataka s kojima je vrlo teško raditi na klasičan način i korištenjem standardnih alata ili relacijskih baza podataka. Kod big data se često susrećemo sa nazivom V3 odnosno to označava *volume, velocity i variety*. Ponegdje se može još i susreti sa još dvije karakteristike a to su *varijabilnost i vjerodostojnost*.
Što se tiče volumena sam naziv big data sve objašnjava. On opisuje ogromnu količinu odnosno volumen podataka koji se tu javljaju.
Raznolikost govori o tome da podaci nisu dovoljno strukturirani, da su često neuredni i nabacani iz raznih izvora i na razne načine.
Iako se ovdje govori o velikoj količini podataka svejedno nam je važna brzina da dovoljno brzo i u nekom konačnom, realnom i prihvatljivom vremenu izvučemo za korisnika važne mu podatke koje se kasnije i koriste pri kreiranju izvještaja i za analizu podataka.
Pošto se u big data prikupljaju mnogi podaci te se u toj gomili podataka mogu vidjeti realna stanja stvari a to znači i kvalitetu podataka pa zbog toga se negdje i spominje termin vjerodostojnosti u big data.
Osim tog termina spominje se još i termin varijabilnosti koji označava da se značenje podataka stalno mijenja. U nekim slučajevima čak se zna i naići na još dvije karakteristike. Jedna od njih je vizualizacija gdje je jasno da iz te velike količine podataka se mogu izvući oni bitni i ključni za nekog korisnika, i na temelju njih se mogu raditi analiz i grafovi iz kojih možemo iščitati neke važne inforacije. Te drugi termin je vrijednost, a tu se smatra upravo poslovna vrijednost ili trošak pri korištenju big data tehnologija gdje se važe koliko zapravo korištenje big data tehnologija ima benefita za samo poslovanje.(Wik, 2017)

Poglavlje 3

Polustrukturirane baze podataka

U današnje vrijeme gdje večina svijeta ima pristup internetu, ne čudi da se javila potreba za nekom drugim načinom implementacije baza podataka osim za relacijskim modelom. Tako se zbog sve više korisnika povećava i konstatno obujam podataka koji kruži preko mreže. Tu u igru dolaze polustrukturirani podaci u kojima su informacije sadržane unutar podataka pa se i još znaju zvati "self-describing" ili samo opisne. Polustrukturirani modeli podataka se najčešće prikazuju preko staba. Kod stabla postoji korijen koji predstavlja neki objekt, a vrijednosti se nalaze u svim ostalim birdovima tog stabla.
Polustrukturirani podaci su najšeće prikazani pomoću OEM-a odnosno Object exchange modela. Isto tako polustrukturirani podaci su često oblikovani pomoću JSONA ili XML-a. Dok je XML vrlo fleksibilan te ima veliku ulogu u razmjeni raznih podataka na webu, JSON ima prednost što je lakše čitljiv ljudima i jednostavniji računalima za rad sa njim.
Kod polustrukturiranih tipova podataka je prednost što objekti ne moraju imati iste atribute, te atributi ne moraju biti isti tip podataka, a samim polustrukturiranim modelom bez problema se mogu prikazati i strukturirani podaci.
Neki prednosti i nedostatci su dolje navedeni

Prednosti:

- Programeri koji rade s objektima ne moraju brinuti o neslaganjima koje prouzročuju objekti već se s objektima lako manipulira sa light-weight libraryjem.
- Podrška za ugniježđenim ili hijerarhijskim podacima pojednostavljuje model podataka koji inače predstavlja složene odnose izmedu entiteta.
- Podrška za listama objekata pojednostavljuje model podataka pri čemu se izbjegava nered konverzije lista u relacijski model podataka.

Nedostaci:

- Tradicionalni model relacijskih podataka ima popularan i gotov Query Language - SQL.

- Uklanjanjem ograničenja iz modela podataka sve se manje promišlja da je potrebno razraditi unos podataka.

(SSD, 2016)

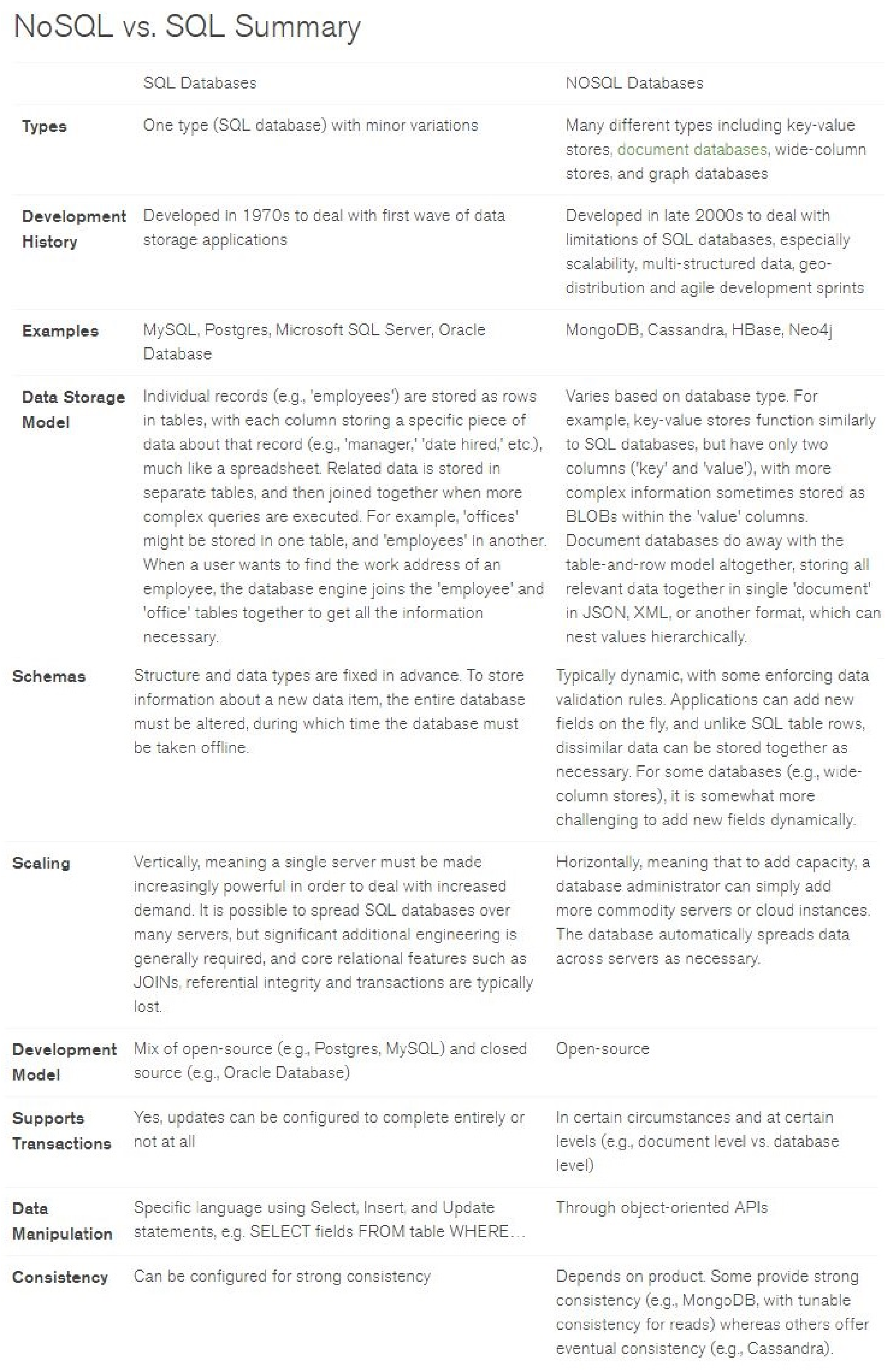
Poglavlje 4

NoSQL

NoSQL baze podataak su baze za pohranu i dohvaćanje podataka koje koriste podosta drugačije principe spram klasičnih relacijskih baza podataka koji koriste tablasti model. NoSQL se još i nazivaju Not Only SQL a se da zaključiti da zapravo podržavaju standradne SQL upite ali i više od toga. Najbitnija razlika je ta da ne koriste Join upite koji su veoma specifični za klasični SQL. Dok spomenemo NoSQL baze podataka tada to povezujemo i sa pojmovima da nisu relacijske, da su distribuirane, otvorenog koda i horizontalno skalabilne. Što se tiče tipova NoSQL baza podataka, postoji ih nekoliko a to su:

- Dokumentne baze u kojima se uparuje svaki ključ s kompleksnom podatkovnom strukturom poznatom kao dokument. Dokumenti mogu sadržavati razne parove ključ-vrijednost, parove ključ-polje ili čak i ugniježene dokumente.
- Grafičke baze se koriste da sačuvaju informacije o nekoj mreži podataka, kao što su na primjer socijalne mreže.
- Ključ-vrijednost je najjednostavnija NoSQL baza podataka. Svaka pojedina stavka ima ime atributa ili ključ povezanu sa vrijednost.
- Široko-stupaste baze kao što je npr. Cassandra i HBase su optimizirane za upite nad velikim skupovima podataka. te spremaju stupce podataka zajedno umjesto u redove.

NoSQL je odgovor na relacijske baze podataka koje imaju čvrstu strukturu. U današnje vrijeme se sve više koriste agilne metodologije razvoja programskog proizvoda te zbog toga se i često mijenja struktura podataka te tu NoSQL ima prednost jer jednostavnije se nosi sa tim izazovim i zahtjevima. Na slici(4.1). možemo vidjeti ukratko koje su razlike između SQL baza podataka i NoSQL baza podataka. (NoS, 2017)



NoSQL vs. SQL Summary

	SQL Databases	NOSQL Databases
Types	One type (SQL database) with minor variations	Many different types including key-value stores, document databases, wide-column stores, and graph databases
Development History	Developed in 1970s to deal with first wave of data storage applications	Developed in late 2000s to deal with limitations of SQL databases, especially scalability, multi-structured data, geo-distribution and agile development sprints
Examples	MySQL, Postgres, Microsoft SQL Server, Oracle Database	MongoDB, Cassandra, HBase, Neo4j
Data Storage Model	Individual records (e.g., 'employees') are stored as rows in tables, with each column storing a specific piece of data about that record (e.g., 'manager,' 'date hired,' etc.), much like a spreadsheet. Related data is stored in separate tables, and then joined together when more complex queries are executed. For example, 'offices' might be stored in one table, and 'employees' in another. When a user wants to find the work address of an employee, the database engine joins the 'employee' and 'office' tables together to get all the information necessary.	Varies based on database type. For example, key-value stores function similarly to SQL databases, but have only two columns ('key' and 'value'), with more complex information sometimes stored as BLOBs within the 'value' columns. Document databases do away with the table-and-row model altogether, storing all relevant data together in single 'document' in JSON, XML, or another format, which can nest values hierarchically.
Schemas	Structure and data types are fixed in advance. To store information about a new data item, the entire database must be altered, during which time the database must be taken offline.	Typically dynamic, with some enforcing data validation rules. Applications can add new fields on the fly, and unlike SQL table rows, dissimilar data can be stored together as necessary. For some databases (e.g., wide-column stores), it is somewhat more challenging to add new fields dynamically.
Scaling	Vertically, meaning a single server must be made increasingly powerful in order to deal with increased demand. It is possible to spread SQL databases over many servers, but significant additional engineering is generally required, and core relational features such as JOINs, referential integrity and transactions are typically lost.	Horizontally, meaning that to add capacity, a database administrator can simply add more commodity servers or cloud instances. The database automatically spreads data across servers as necessary.
Development Model	Mix of open-source (e.g., Postgres, MySQL) and closed source (e.g., Oracle Database)	Open-source
Supports Transactions	Yes, updates can be configured to complete entirely or not at all	In certain circumstances and at certain levels (e.g., document level vs. database level)
Data Manipulation	Specific language using Select, Insert, and Update statements, e.g. SELECT fields FROM table WHERE…	Through object-oriented APIs
Consistency	Can be configured for strong consistency	Depends on product. Some provide strong consistency (e.g., MongoDB, with tunable consistency for reads) whereas others offer eventual consistency (e.g., Cassandra).

Slika 4.1: SQL vs NoSQL ()

Poglavlje 5

MongoDB

MongoDB je open-source dokument baza podataka a zapis u MongoDB bazi zove se dokument, koji je podatkovna struktura sastavljena od parova ključeva i njihovih vrijednosti. MongoDB dokumenti su strukturom slični JSON objektima. Vrijednosti polja mogu uključivati i druge dokumente, polja te listu polja.
MongoDB sprema BSON dokumente u kolekcije. Što se tiče same razlike u terminologiji kod klasičnih relacijskih baza imamo tablice a to se kod MongoDB-a zovu kolekcije, isto tako redak se zove dokument a stupa je polje. Rad sa MongoDB-om može ići preko terminala odnosno komadnom linijom ili preko nekih programa za vizualizaciju i olakšano korištenje MongoDB-a kao što je Robomongo. MongoDb ne koristi baš klasične sql upite već pruža neke metode. Tako npr. imamo metode za unos dokumenata u kolekciju(db.collection.insert()), pa za klasičan selekt upit postoji metoda find() nad kojom možemo još i definirat neke dodatne uvjete. Isto tako imamo i metodu update(), delete() te remove(). Postoje i malo specifičnije nabrojene CRUD metode koje se mogu odnosi na samo jedan zapis u kolekciji ili više njih itd.
Operacije agregiranja procesiraju zapise podataka i vračaju izračunate rezultate. Takve funkcije obrađuju podatke i prikazuju rezultat u potrebnom obliku. Agregirajuče operacije koriste kolekcije i dokumente kao ulaz i izlaz, poput upita. MongoDB pruža tri načina za izvršavanje agregacije podataka a to su agregacijski cjevovod, map-reduce te agregirajuče funkcije jednostavne svrhe. (Mon, 2016)

Poglavlje 6

Izrada baze podataka u MongoDB

Za početak trebamo skinuti i instalirati MongoDB bazu podataka. Pošto se za operacijski sustav koristi Ubuntu, to ćemo obaviti preko naredba u terminalu i apt sustava za upravljanje paketima.
Prvo utipkamo naredbu za uvođenje javnog ključa

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80
--recv 0C49F3730359A14518585931BC711F9BA15703C6
```

Zatim kreiramo listu datoteka za Ubuntu verziju koju koristimo, u ovom slučaju je to 16.04

```
echo "deb [ arch=amd64,arm64 ]
http://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/3.4 multiverse"
| sudo tee /etc/apt/sources.list.d/mongodb-org-3.4.list
```

Nakon toga učitamo lokalnu bazu podataka paketa

```
sudo apt-get update
```

Sada konačno možemo skinuti i instalirati MongoDB

```
sudo apt-get install -y mongodb-org
```

Nakon što smo to odradili možemo pokrenuti lokalni server sa naredbom

```
sudo service mongod start
```

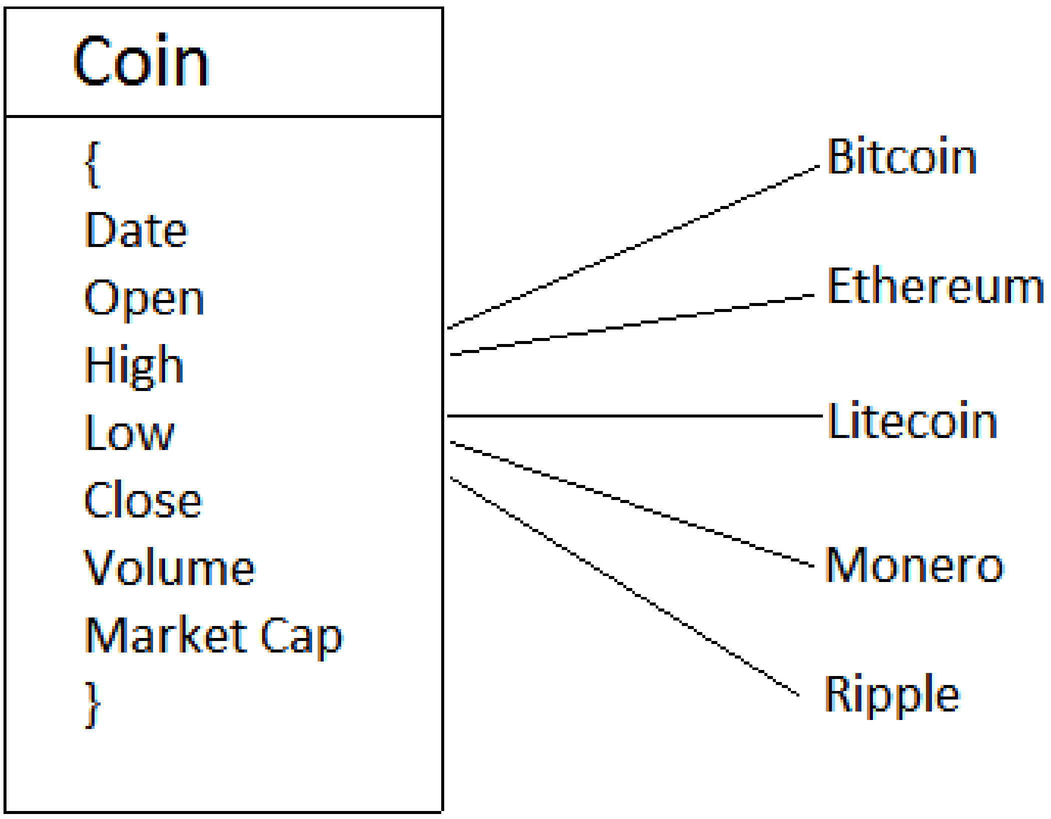
Naredbe za provjeru statusa servera, resetiranje i zaustavljanje su sljedeće:

```
sudo service mongod {status|restart|stop}
```

Kada bi htjeli nešto testirati i odraditi neke administratorske stvari onda bi utipkali naredbu *mongo* bi nam se tada otvorilo JavaScript shell sučelje za MongoDB. Za ovu aplikaciju koristimo gotove skupove podataka koji su vezani za rudarenja virtualnih valuta. Tako na slici(6.1) vidimo otprilike kako izgleda model baze u ovoj aplikaciji.

Dolje možemo vidjeti dio koda, odnosno JSON-a jednog od kriptovaluta:

```
{
    "Date": "Aug 15, 2017",
    "Open": 299.95,
```



Slika 6.1: Model baze

```
    "High": 300.41,
    "Low": 279.33,
    "Close": 289.82,
    "Volume": "1,051,800,000",
    "Market␣Cap": "28,195,800,000"
}
```

Da konkretnije pojasnimo:

- Date : datu promatranja
- Open : Cijena na početku dana
- High : Najviša cijena u danu
- Low : Najniža cijena u danu
- Close : Cijena na kraju danu
- Volume : Volumen transakcija tog dana
- Market Cap : Tržišna kapitalizacija u USD

Za ovaj projekt smo kreirali novu bazu pod nazivom tbp:

```
# mongo
> use tbp
```

Zatim smo svaki od skupa podataka dodali u bazu:

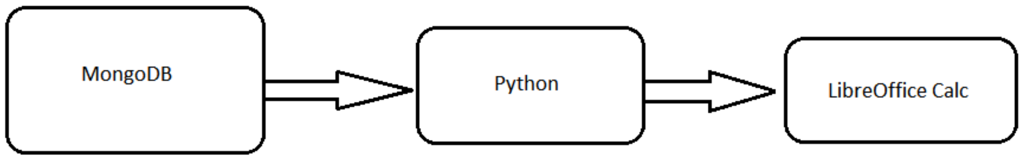
```
mongoimport --db tbp --collection bitcoin --file bitcoin.json
--jsonArray
mongoimport --db tbp --collection ethereum --file ethereum.json
--jsonArray
mongoimport --db tbp --collection litecoin --file litecoin.json
--jsonArray
mongoimport --db tbp --collection monero --file monero.json
--jsonArray
mongoimport --db tbp --collection ripple --file ripple.json
--jsonArray
```

Nakon svih ovih radnji, naša baza je spremna za rad s aplikacijom.

Poglavlje 7

Aplikacija

Aplikacija u ovom projektu za logiku koristi python programski jezik. Konkretno koristio se PyCharm community ide za lakši razvoj aplikacije. Tako se u pythonu nalazi isprogramirana logika koja se povezuje na MongoDB bazu, radi određene radnji tako da dohvaća određene podake ovisno o upitu te ih zapisuje u LibreOffice Calcu te ukoliko odaberemo određene radnje stvara i gafove kroz koje možemo vidjeti neku analizu stanja kriptovaluta(7.1).



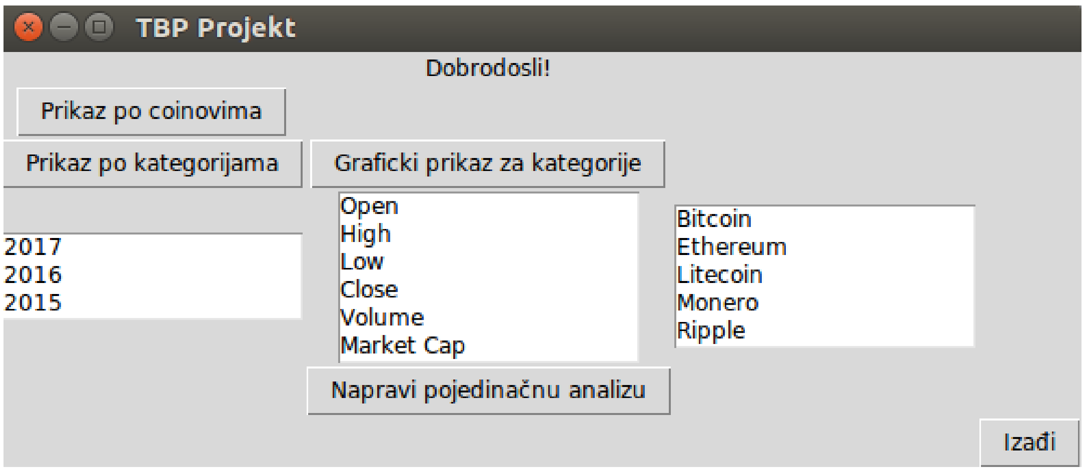
Slika 7.1: Model aplikacije

Što se tiče same logike i koda, za početak imamo biblioteke koje koristimo. Tako nam subprocess služi da možemo pozivati programe van pythona kao npr terminal ili libreoffice calc. Nakon toga imamo biblioteku time koja nam koristi za pozivanje sleep metode. Biblioteke xlwt i pyoo nam služe za rad sa LibreOffice calcom, odnosno jedna je korištena za samo upisivanje a druga za izradu grafova. Biblioteka os nam služi za dohvaćanje putanje projekta. Biblioteka tkinter nam je osnovna biblioteka za prikaz objekata na ekran(prozor, gumbi, popisi). Te za kraj imamo pymongo biblioteku koja nam služi za spajanje na MongoDB bazu. (Pojman, 2017), (Machin, 2017)

```
import subprocess
import time
import xlwt
import pyoo
import os
import tkinter.messagebox
from tkinter import *
from pymongo import MongoClient
```

Zatim na početku prvo pozivamo konstruktor klase te se spajamo na bazu i spremamo našu tbp bazu u db varijablu koju dalje prosljeđujemo metodama koje će je koristit kako se nebi trebali svaki puta ponovno spajati na bazu. Osim toga stvaramo početni prozor, gubove, listu, njihove funkcionalnosti i položaj u prozoru.

```
class MiningCoins(Frame):

    def __init__(self, master):
        try:
            #subprocess.Popen(['service mongod start'], shell=True)
            #time.sleep(5)
            client = MongoClient('mongodb://localhost:27017/')
            db = client['tbp']
        except ConnectionRefusedError:
            tkinter.messageBox.showinfo('MongoDB␣connection␣error',
            'Nije␣moguce␣se␣spojiti␣na␣MongoDB!')
        frame = Frame(master)
        frame.grid()
        label = Label(frame, text = "Dobrodosli!")
        label.grid(row=0, column=1)
        btnCoin = Button(frame, text = "Prikaz␣po␣coinovima",
        command=lambda: self.coin(db))
        btnCoin.grid(row=1)
        btnKat = Button(frame, text = "Prikaz␣po␣kategorijama",
        command=lambda: self.category(db))
        btnKat.grid(row=2)
        btnGKat = Button(frame, text="Graficki␣prikaz␣za␣kategorije",
        command=self.grafKat)
        btnGKat.grid(row=2, column=1)
        lbGod = Listbox(frame, height=3, exportselection=0)
        lbGod.insert(0, "2017")
        lbGod.insert(1, "2016")
        lbGod.insert(2, "2015")
        lbGod.grid(row = 3)
        lbKat = Listbox(frame, height=6, exportselection=0)
        lbKat.insert(0, "Open")
        lbKat.insert(1, "High")
        lbKat.insert(2, "Low")
        lbKat.insert(3, "Close")
        lbKat.insert(4, "Volume")
        lbKat.insert(5, "Market␣Cap")
        lbKat.grid(row=3, column=1)
        lbCoi = Listbox(frame, height=5, exportselection=0)
        lbCoi.insert(0, "Bitcoin")
        lbCoi.insert(1, "Ethereum")
        lbCoi.insert(2, "Litecoin")
        lbCoi.insert(3, "Monero")
        lbCoi.insert(4, "Ripple")
        lbCoi.grid(row=3, column=2)
        btnP = Button(frame, text="Napravi␣pojedinacnu␣analizu",
        command=lambda: self.analiza(db, lbGod.get(ACTIVE),
        lbKat.get(ACTIVE), lbCoi.get(ACTIVE)))
        btnP.grid(row=4, column=1)
        btnQuit = Button(frame, text="Izadi", command=frame.quit)
        btnQuit.grid(row=5, column=4)
```



Slika 7.2: Aplikacija

Prva metoda koju koristimo je coin(db) koja jednostano dohvaća sve kolekcije iz baze i sve dokumente iz svake kolekcije i upisuje u LibreOffice Calc dokument na način da svaka kolekcija se nalazi u vlastitom sheetu.

```
def coin(self, db):
        bit = db.bitcoin.find()
        eth = db.ethereum.find()
        lit = db.litecoin.find()
        mon = db.monero.find()
        rip = db.ripple.find()

        cursors = list([bit, eth, lit, mon, rip])

        wb = xlwt.Workbook()

        sheetB = wb.add_sheet("Bitcoin")
        sheetE = wb.add_sheet("Etherium")
        sheetL = wb.add_sheet("Litecoin")
        sheetM = wb.add_sheet("Monero")
        sheetR = wb.add_sheet("Ripple")

        sheet = list([sheetB, sheetE, sheetL, sheetM, sheetR])

        for s in sheet:
            s.write(0, 0, 'Datum')
            s.write(0, 1, 'Open')
            s.write(0, 2, 'High')
            s.write(0, 3, 'Low')
            s.write(0, 4, 'Close')
            s.write(0, 5, 'Volume')
            s.write(0, 6, 'Market␣Cap')



        i = 0
        for c in cursors:
            j = 1
            for row in c:
                sheet[i].write(j, 0, str(row['Date']))
                sheet[i].write(j, 1, float(row['Open']))
                sheet[i].write(j, 2, float(row['High']))
                sheet[i].write(j, 3, float(row['Low']))
                sheet[i].write(j, 4, float(row['Close']))
                sheet[i].write(j, 5, str(row['Volume']))
                sheet[i].write(j, 6, str(row['Market␣Cap']))
                j = j + 1
            i = i + 1

        reportDir = os.path.dirname(os.path.abspath('reports'))
                                + "/reports/"
        wb.save(reportDir + "coinovi.ods")
        subprocess.call(['/usr/bin/localc', reportDir
                                        + 'coinovi.ods'])
```

Druga metoda category(db) također radi istu stvar da povlači sve kolekcije i sve dokumente svake od kolekcije ali ovoga puta spremamo te zapise na način da svaki sheet označava jednu kategoriju odnosno atribut koji je zajednički svim tim kolekcijama.

```
def category(self, db):
        lim = db.ethereum.count()

        eth = db.ethereum.find()
        bit = db.bitcoin.find().limit(lim)
        lit = db.litecoin.find().limit(lim)
        mon = db.monero.find().limit(lim)
        rip = db.ripple.find().limit(lim)

        cursors = list([bit, eth, lit, mon, rip])

        wb = xlwt.Workbook()
```

```
    sheetO = wb.add_sheet("Open")
    sheetH = wb.add_sheet("High")
    sheetL = wb.add_sheet("Low")
    sheetC = wb.add_sheet("Close")
    sheetV = wb.add_sheet("Volume")
    sheetMC = wb.add_sheet("Market Cap")

    sheet = list([sheetO, sheetH, sheetL,
                                        sheetC, sheetV, sheetMC])

    for s in sheet:
        s.write(0, 0, 'Datum')
        s.write(0, 1, 'Bitcoin')
        s.write(0, 2, 'Ethereum')
        s.write(0, 3, 'Litecoin')
        s.write(0, 4, 'Monero')
        s.write(0, 5, 'Ripple')

    i = 1
    for row in bit:
        for s in sheet:
            s.write(i, 0, str(row['Date']))
        sheet[0].write(i, 1, float(row['Open']))
        sheet[1].write(i, 1, float(row['High']))
        sheet[2].write(i, 1, float(row['Low']))
        sheet[3].write(i, 1, float(row['Close']))
        sheet[4].write(i, 1, int(str(row['Volume'])
                                        .replace(',', '')))
        sheet[5].write(i, 1, int(str(row['Market Cap'])
                                        .replace(',', '')))
        i = i + 1

    j = 1
    for c in cursors:
        i = 1
        for row in c:
            sheet[0].write(i, j, float(row['Open']))
            sheet[1].write(i, j, float(row['High']))
            sheet[2].write(i, j, float(row['Low']))
            sheet[3].write(i, j, float(row['Close']))
            sheet[4].write(i, j, int(str(row['Volume'])
                                              .replace(',','')))
            sheet[5].write(i, j, int(str(row['Market Cap'])
                                              .replace(',','')))
            i = i + 1
```

```
                j = j + 1

            reportDir = os.path.dirname(os.path.abspath('reports'))
                                    + "/reports/"
            wb.save(reportDir + "kategorije.ods")
            subprocess.call(['/usr/bin/localc', reportDir
                                    + 'kategorije.ods'])
```

Treća metoda grafKat() prvo poziva terminal kako bi pokrenula dana naredba i kako bi se time napravio soffice socket kako bi uopće mogli koristiti funkcije pyoo biblioteke. Tako otvaramo dokument kategorije kojeg smo gore kreirali, ili vraćamo grešku ukoliko još nije kreiran. Zatim na temelju zapisa u Calc datoteci za svaki sheet radimo graf koji prikazuje usporedbu fluktuacija za određenu kategoriju po svakoj od kriptovaluta.

```
def grafKat(self):
    try:
        subprocess.Popen(['soffice
␣␣␣␣␣␣␣␣accept="socket,host=localhost,port=2002;urp;"
␣␣␣␣␣␣␣␣--norestore␣--nologo␣--nodefault␣#␣--headless'],
        shell=True)
        time.sleep(2)
        desktop = pyoo.Desktop('localhost', 2002)
        reportKat = os.path.dirname(os.path.abspath('reports'))
        + "/reports/kategorije.ods"
        doc = desktop.open_spreadsheet(reportKat)

        sheet1 = doc.sheets[0]
        sheet2 = doc.sheets[1]
        sheet3 = doc.sheets[2]
        sheet4 = doc.sheets[3]
        sheet5 = doc.sheets[4]
        sheet6 = doc.sheets[5]

        chart = sheet1.charts.create('Open', sheet1[2:50, 7:18],
        sheet1[0:740, 0:6], row_header=TRUE, col_header=TRUE)
        diagram = chart.change_type(pyoo.LineDiagram)
        diagram.y_axis.title = "USD"
        diagram.y_axis.logarithmic = True

        chart = sheet2.charts.create('High', sheet2[2:50, 7:18], sheet2[0:740, 0:6],
        row_header=TRUE, col_header=TRUE)
        diagram = chart.change_type(pyoo.LineDiagram)
        diagram.y_axis.title = "USD"
        diagram.y_axis.logarithmic = True

        chart = sheet3.charts.create('Low', sheet3[2:50, 7:18],
        sheet3[0:740, 0:6], row_header=TRUE, col_header=TRUE)
        diagram = chart.change_type(pyoo.LineDiagram)
        diagram.y_axis.title = "USD"
        diagram.y_axis.logarithmic = True

        chart = sheet4.charts.create('Close', sheet4[2:50, 7:18],
        sheet4[0:740, 0:6], row_header=TRUE, col_header=TRUE)
        diagram = chart.change_type(pyoo.LineDiagram)
        diagram.y_axis.title = "USD"
        diagram.y_axis.logarithmic = True

        chart = sheet5.charts.create('Volume',
        sheet5[2:28, 7:18], sheet5[0:740, 0:6],
        row_header=TRUE, col_header=TRUE)
        diagram = chart.change_type(pyoo.LineDiagram)
        diagram.y_axis.title = "USD"
        diagram.y_axis.logarithmic = True

        chart = sheet6.charts.create('Market␣Cap',
        sheet6[2:28, 7:18], sheet6[0:740, 0:6],
        row_header=TRUE, col_header=TRUE)
        diagram = chart.change_type(pyoo.LineDiagram)
        diagram.y_axis.title = "USD"
        diagram.y_axis.logarithmic = True

        reportDir = os.path.dirname(os.path.abspath('reports'))
                                + "/reports/"
        doc.save(reportDir + 'grafovi.ods')
    except:
        tkinter.messagebox.showinfo('Dokument␣ne␣postoji',
        'Potrebno␣je␣prvo␣generirati␣prikaz␣kategorija!')
```

Četvrta metoda analiza(db, god, cat, coi) na temelju primljenih argumenata specificira upit nad kojom kriptovalutom će se napraviti, za koju godinu i koju kategoriju te na temelju dobivenog Calc dokumenta radi graf iz kojeg opet možemo radit neku analizu kretanja vrijednosti kriptovalute ovisno o odabranim parametrima.

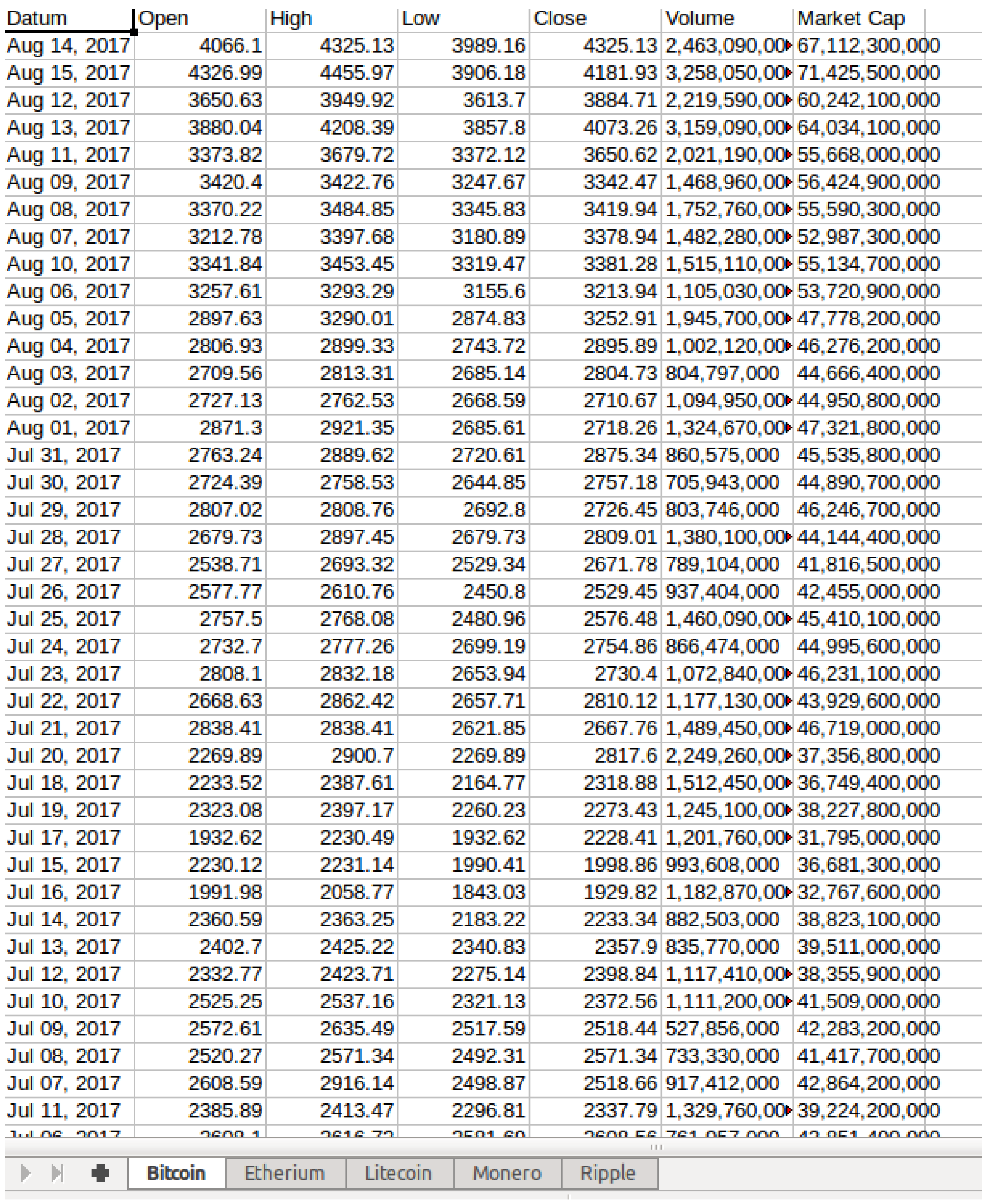
```
def analiza(self, db, god, cat, coi):

        if coi == 'Bitcoin':
            cursor = db.bitcoin.find({'Date': {'$regex': god}},
            {'Date' : 1, cat: 1})

        elif coi == 'Ethereum':
            cursor = db.ethereum.find({'Date': {'$regex': god}},
            {'Date' : 1, cat: 1})
```

```
    elif coi == 'Litecoin':
        cursor = db.litecoin.find({'Date': {'$regex': god}},
        {'Date' : 1, cat: 1})

    elif coi == 'Monero':
        cursor = db.monero.find({'Date': {'$regex': god}},
        {'Date' : 1, cat: 1})

    elif coi == 'Ripple':
        cursor = db.ripple.find({'Date': {'$regex': god}},
        {'Date' : 1, cat: 1})

    wb = xlwt.Workbook()

    sheet = wb.add_sheet(coi)

    sheet.write(0, 0, 'Datum')
    sheet.write(0, 1, cat)


    i = 1
    for row in cursor:
        sheet.write(i, 0, str(row['Date']))
        if (cat == 'Volume' or cat == 'Market␣Cap'):
            sheet.write(i, 1, int(str(row[cat]).replace(',', '')))
        else:
            sheet.write(i, 1, float(row[cat]))
        i = i + 1

    reportDir = os.path.dirname(os.path.abspath('reports'))
                            + "/reports/"
    wb.save(reportDir + "analiza.ods")

    subprocess.Popen(['soffice␣--accept="socket,host=localhost,
␣␣␣␣␣␣␣␣port=2002;urp;"␣--norestore␣--nologo␣--nodefault␣#␣--headless'],
                    shell=True)
    time.sleep(2)
    desktop = pyoo.Desktop('localhost', 2002)
    reportKat = os.path.dirname(os.path.abspath('reports'))
                            + "/reports/analiza.ods"
    doc = desktop.open_spreadsheet(reportKat)

    sheet1 = doc.sheets[0]
```

```
        chart = sheet1.charts.create(cat, sheet1[2:20, 4:20],
        sheet1[0:cursor.count()+1, 0:2], row_header=TRUE,
        col_header=TRUE)
        diagram = chart.change_type(pyoo.LineDiagram)
        diagram.y_axis.title = "USD"
        diagram.y_axis.logarithmic = True

        reportDir = os.path.dirname(os.path.abspath('reports'))
                                + "/reports/"
        doc.save(reportDir + 'analiza.ods')
```

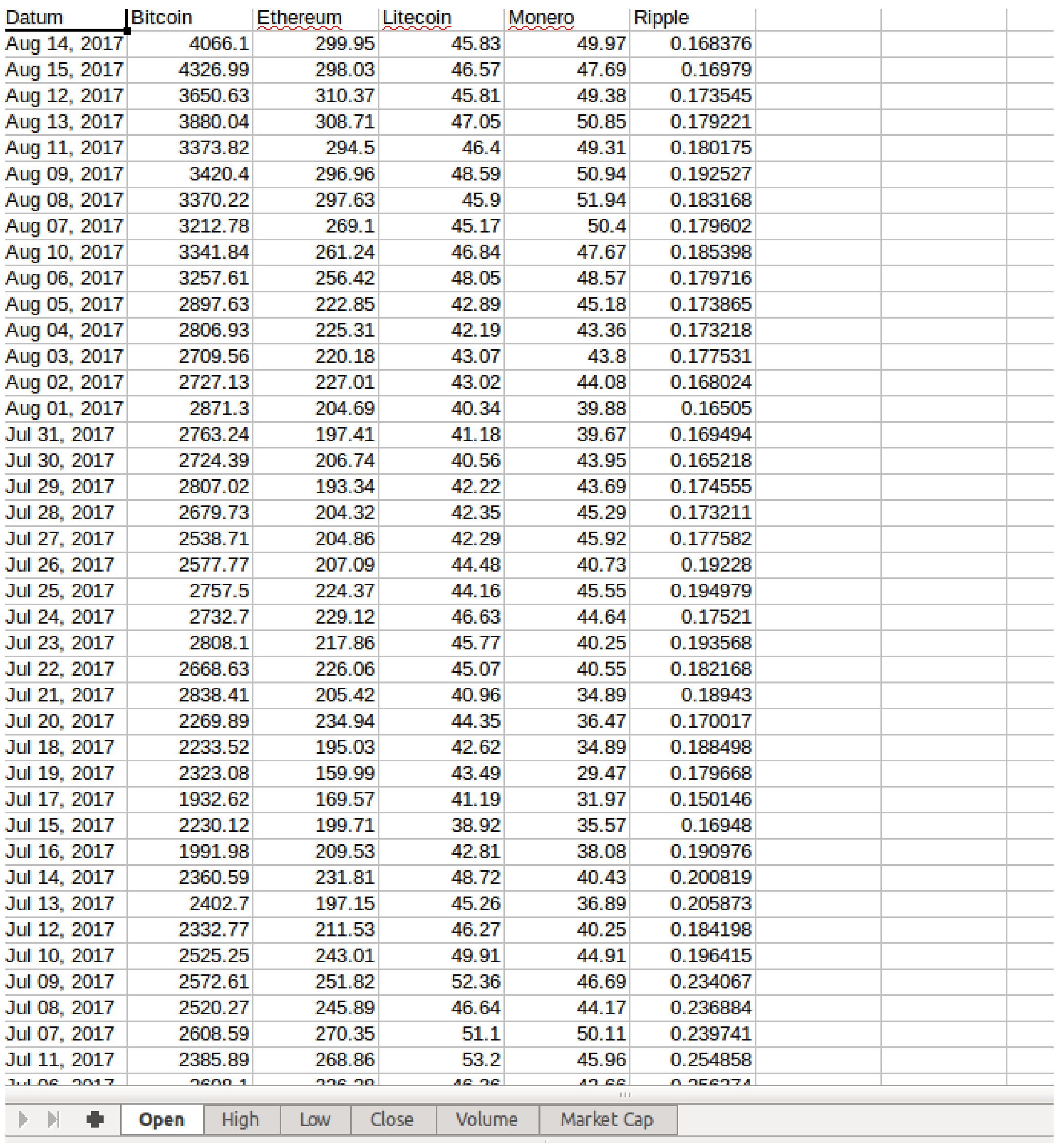
Za kraj imamo linije koda koje instaciraju objekat klase Tk() koji nam služi dza definiranje prozora i njegovih objekata, te pozivamo sami kontruktor klase kako bi se pokrenuo program.

```
root = Tk()
root.title("TBP␣Projekt")
mc = MiningCoins(root)
root.mainloop()
```



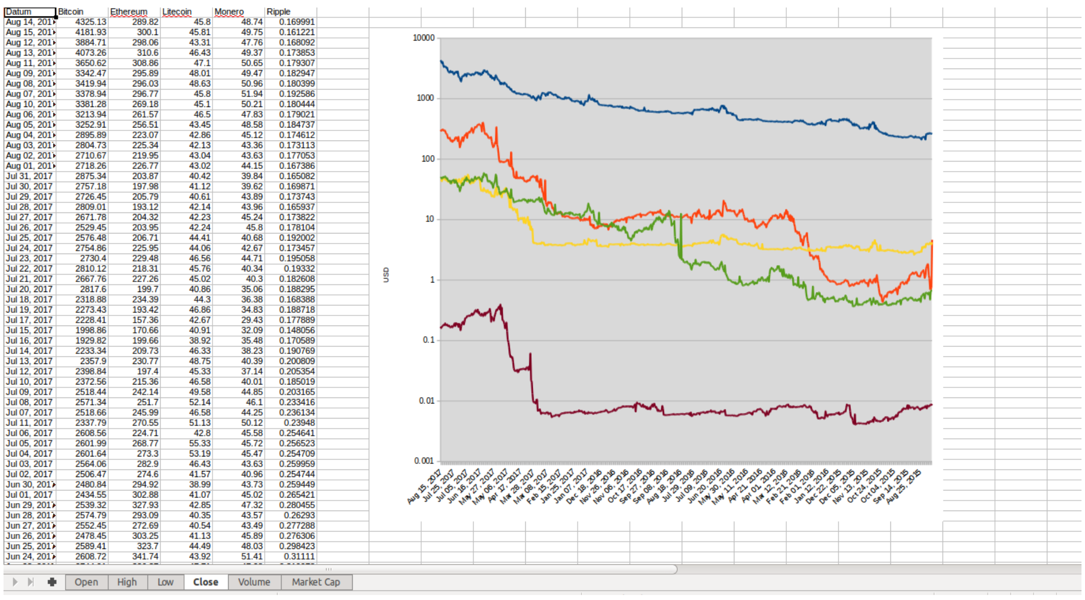
Datum	Open	High	Low	Close	Volume	Market Cap
Aug 14, 2017	4066.1	4325.13	3989.16	4325.13	2,463,090,00	67,112,300,000
Aug 15, 2017	4326.99	4455.97	3906.18	4181.93	3,258,050,00	71,425,500,000
Aug 12, 2017	3650.63	3949.92	3613.7	3884.71	2,219,590,00	60,242,100,000
Aug 13, 2017	3880.04	4208.39	3857.8	4073.26	3,159,090,00	64,034,100,000
Aug 11, 2017	3373.82	3679.72	3372.12	3650.62	2,021,190,00	55,668,000,000
Aug 09, 2017	3420.4	3422.76	3247.67	3342.47	1,468,960,00	56,424,900,000
Aug 08, 2017	3370.22	3484.85	3345.83	3419.94	1,752,760,00	55,590,300,000
Aug 07, 2017	3212.78	3397.68	3180.89	3378.94	1,482,280,00	52,987,300,000
Aug 10, 2017	3341.84	3453.45	3319.47	3381.28	1,515,110,00	55,134,700,000
Aug 06, 2017	3257.61	3293.29	3155.6	3213.94	1,105,030,00	53,720,900,000
Aug 05, 2017	2897.63	3290.01	2874.83	3252.91	1,945,700,00	47,778,200,000
Aug 04, 2017	2806.93	2899.33	2743.72	2895.89	1,002,120,00	46,276,200,000
Aug 03, 2017	2709.56	2813.31	2685.14	2804.73	804,797,000	44,666,400,000
Aug 02, 2017	2727.13	2762.53	2668.59	2710.67	1,094,950,00	44,950,800,000
Aug 01, 2017	2871.3	2921.35	2685.61	2718.26	1,324,670,00	47,321,800,000
Jul 31, 2017	2763.24	2889.62	2720.61	2875.34	860,575,000	45,535,800,000
Jul 30, 2017	2724.39	2758.53	2644.85	2757.18	705,943,000	44,890,700,000
Jul 29, 2017	2807.02	2808.76	2692.8	2726.45	803,746,000	46,246,700,000
Jul 28, 2017	2679.73	2897.45	2679.73	2809.01	1,380,100,00	44,144,400,000
Jul 27, 2017	2538.71	2693.32	2529.34	2671.78	789,104,000	41,816,500,000
Jul 26, 2017	2577.77	2610.76	2450.8	2529.45	937,404,000	42,455,000,000
Jul 25, 2017	2757.5	2768.08	2480.96	2576.48	1,460,090,00	45,410,100,000
Jul 24, 2017	2732.7	2777.26	2699.19	2754.86	866,474,000	44,995,600,000
Jul 23, 2017	2808.1	2832.18	2653.94	2730.4	1,072,840,00	46,231,100,000
Jul 22, 2017	2668.63	2862.42	2657.71	2810.12	1,177,130,00	43,929,600,000
Jul 21, 2017	2838.41	2838.41	2621.85	2667.76	1,489,450,00	46,719,000,000
Jul 20, 2017	2269.89	2900.7	2269.89	2817.6	2,249,260,00	37,356,800,000
Jul 18, 2017	2233.52	2387.61	2164.77	2318.88	1,512,450,00	36,749,400,000
Jul 19, 2017	2323.08	2397.17	2260.23	2273.43	1,245,100,00	38,227,800,000
Jul 17, 2017	1932.62	2230.49	1932.62	2228.41	1,201,760,00	31,795,000,000
Jul 15, 2017	2230.12	2231.14	1990.41	1998.86	993,608,000	36,681,300,000
Jul 16, 2017	1991.98	2058.77	1843.03	1929.82	1,182,870,00	32,767,600,000
Jul 14, 2017	2360.59	2363.25	2183.22	2233.34	882,503,000	38,823,100,000
Jul 13, 2017	2402.7	2425.22	2340.83	2357.9	835,770,000	39,511,000,000
Jul 12, 2017	2332.77	2423.71	2275.14	2398.84	1,117,410,00	38,355,900,000
Jul 10, 2017	2525.25	2537.16	2321.13	2372.56	1,111,200,00	41,509,000,000
Jul 09, 2017	2572.61	2635.49	2517.59	2518.44	527,856,000	42,283,200,000
Jul 08, 2017	2520.27	2571.34	2492.31	2571.34	733,330,000	41,417,700,000
Jul 07, 2017	2608.59	2916.14	2498.87	2518.66	917,412,000	42,864,200,000
Jul 11, 2017	2385.89	2413.47	2296.81	2337.79	1,329,760,00	39,224,200,000
Jul 06, 2017	2608.1	2616.72	2581.69	2608.56	761,957,000	42,851,400,000

Slika 7.3: Prkaz svih kriptovaluta i vrijednosti



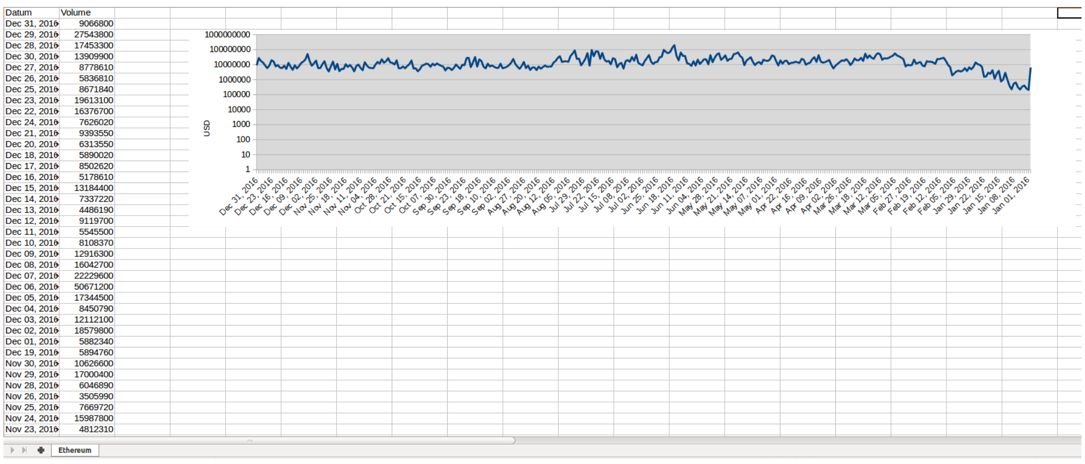
Datum	Bitcoin	Ethereum	Litecoin	Monero	Ripple
Aug 14, 2017	4066.1	299.95	45.83	49.97	0.168376
Aug 15, 2017	4326.99	298.03	46.57	47.69	0.16979
Aug 12, 2017	3650.63	310.37	45.81	49.38	0.173545
Aug 13, 2017	3880.04	308.71	47.05	50.85	0.179221
Aug 11, 2017	3373.82	294.5	46.4	49.31	0.180175
Aug 09, 2017	3420.4	296.96	48.59	50.94	0.192527
Aug 08, 2017	3370.22	297.63	45.9	51.94	0.183168
Aug 07, 2017	3212.78	269.1	45.17	50.4	0.179602
Aug 10, 2017	3341.84	261.24	46.84	47.67	0.185398
Aug 06, 2017	3257.61	256.42	48.05	48.57	0.179716
Aug 05, 2017	2897.63	222.85	42.89	45.18	0.173865
Aug 04, 2017	2806.93	225.31	42.19	43.36	0.173218
Aug 03, 2017	2709.56	220.18	43.07	43.8	0.177531
Aug 02, 2017	2727.13	227.01	43.02	44.08	0.168024
Aug 01, 2017	2871.3	204.69	40.34	39.88	0.16505
Jul 31, 2017	2763.24	197.41	41.18	39.67	0.169494
Jul 30, 2017	2724.39	206.74	40.56	43.95	0.165218
Jul 29, 2017	2807.02	193.34	42.22	43.69	0.174555
Jul 28, 2017	2679.73	204.32	42.35	45.29	0.173211
Jul 27, 2017	2538.71	204.86	42.29	45.92	0.177582
Jul 26, 2017	2577.77	207.09	44.48	40.73	0.19228
Jul 25, 2017	2757.5	224.37	44.16	45.55	0.194979
Jul 24, 2017	2732.7	229.12	46.63	44.64	0.17521
Jul 23, 2017	2808.1	217.86	45.77	40.25	0.193568
Jul 22, 2017	2668.63	226.06	45.07	40.55	0.182168
Jul 21, 2017	2838.41	205.42	40.96	34.89	0.18943
Jul 20, 2017	2269.89	234.94	44.35	36.47	0.170017
Jul 18, 2017	2233.52	195.03	42.62	34.89	0.188498
Jul 19, 2017	2323.08	159.99	43.49	29.47	0.179668
Jul 17, 2017	1932.62	169.57	41.19	31.97	0.150146
Jul 15, 2017	2230.12	199.71	38.92	35.57	0.16948
Jul 16, 2017	1991.98	209.53	42.81	38.08	0.190976
Jul 14, 2017	2360.59	231.81	48.72	40.43	0.200819
Jul 13, 2017	2402.7	197.15	45.26	36.89	0.205873
Jul 12, 2017	2332.77	211.53	46.27	40.25	0.184198
Jul 10, 2017	2525.25	243.01	49.91	44.91	0.196415
Jul 09, 2017	2572.61	251.82	52.36	46.69	0.234067
Jul 08, 2017	2520.27	245.89	46.64	44.17	0.236884
Jul 07, 2017	2608.59	270.35	51.1	50.11	0.239741
Jul 11, 2017	2385.89	268.86	53.2	45.96	0.254858

Slika 7.4: Kategorijzacija



Datum	Bitcoin	Ethereum	Litecoin	Monero	Ripple
Aug 14, 201	4325.13	289.82	45.8	48.74	0.169991
Aug 15, 201	4181.93	300.1	45.81	49.75	0.161221
Aug 12, 201	3884.71	298.06	43.31	47.76	0.168092
Aug 13, 201	4073.26	310.6	46.43	49.37	0.173853
Aug 11, 201	3650.62	308.86	47.1	50.65	0.179307
Aug 09, 201	3342.47	295.89	48.01	49.47	0.182947
Aug 08, 201	3419.94	296.03	48.63	50.96	0.180399
Aug 07, 201	3378.94	296.77	45.8	51.94	0.192586
Aug 10, 201	3381.28	269.18	45.1	50.21	0.180444
Aug 06, 201	3213.94	261.57	46.5	47.83	0.179021
Aug 05, 201	3252.91	256.51	43.45	48.58	0.184737
Aug 04, 201	2895.89	223.07	42.86	45.12	0.174612
Aug 03, 201	2804.73	225.34	42.13	43.36	0.173113
Aug 02, 201	2710.67	219.95	43.04	43.63	0.177053
Aug 01, 201	2718.26	226.77	43.02	44.15	0.167386
Jul 31, 2017	2875.34	203.87	40.42	39.84	0.165082
Jul 30, 2017	2757.18	197.98	41.12	39.62	0.169871
Jul 29, 2017	2726.45	205.79	40.61	43.89	0.173743
Jul 28, 2017	2809.01	193.12	42.14	43.96	0.165937
Jul 27, 2017	2671.78	204.32	42.23	45.24	0.173822
Jul 26, 2017	2529.45	203.95	42.24	45.8	0.178104
Jul 25, 2017	2576.48	206.71	44.41	40.68	0.192002
Jul 24, 2017	2754.86	225.95	44.06	42.67	0.173457
Jul 23, 2017	2730.4	229.48	46.56	44.71	0.195058
Jul 22, 2017	2810.12	218.31	45.76	40.34	0.19332
Jul 21, 2017	2667.76	227.26	45.02	40.3	0.182608
Jul 20, 2017	2817.6	199.7	40.86	35.06	0.188295
Jul 18, 2017	2318.88	234.39	44.3	36.38	0.168388
Jul 19, 2017	2273.43	193.42	46.86	34.83	0.188718
Jul 17, 2017	2228.41	157.36	42.67	29.43	0.177889
Jul 15, 2017	1998.86	170.66	40.91	32.09	0.148056
Jul 16, 2017	1929.82	199.66	38.92	35.48	0.170589
Jul 14, 2017	2233.34	209.73	46.33	38.23	0.190769
Jul 13, 2017	2357.9	230.77	48.75	40.39	0.200809
Jul 12, 2017	2398.84	197.4	45.33	37.14	0.205354
Jul 10, 2017	2372.56	215.36	46.58	40.01	0.185019
Jul 09, 2017	2518.44	242.14	49.58	44.85	0.203165
Jul 08, 2017	2571.34	251.7	52.14	46.1	0.233416
Jul 07, 2017	2518.66	245.99	46.58	44.25	0.236134
Jul 11, 2017	2337.79	270.55	51.13	50.12	0.23948
Jul 06, 2017	2608.56	224.71	42.8	45.58	0.254641
Jul 05, 2017	2601.99	268.77	55.33	45.72	0.256523
Jul 04, 2017	2601.64	273.3	53.19	45.47	0.254709
Jul 03, 2017	2564.06	282.9	46.43	43.63	0.259959
Jul 02, 2017	2506.47	274.6	41.57	40.96	0.254744
Jun 30, 201	2480.84	294.92	38.99	43.73	0.259449
Jul 01, 2017	2434.55	302.88	41.07	45.02	0.265421
Jun 29, 201	2539.32	327.93	42.85	47.32	0.280455
Jun 28, 201	2574.79	293.09	40.35	43.57	0.26293
Jun 27, 201	2552.45	272.69	40.54	43.49	0.277288
Jun 26, 201	2478.45	303.25	41.13	45.89	0.276306
Jun 25, 201	2589.41	323.7	44.49	48.03	0.298423
Jun 24, 201	2608.72	341.74	43.92	51.41	0.31111

Slika 7.5: Grafovi po kategorijama



Datum	Volume
Dec 31, 2016	9066800
Dec 29, 2016	27543800
Dec 28, 2016	17453300
Dec 30, 2016	13909900
Dec 27, 2016	8778610
Dec 26, 2016	5836810
Dec 25, 2016	8671840
Dec 23, 2016	19613100
Dec 22, 2016	16376700
Dec 24, 2016	7626020
Dec 21, 2016	9393550
Dec 20, 2016	6313550
Dec 18, 2016	5890020
Dec 17, 2016	8502620
Dec 16, 2016	5178610
Dec 15, 2016	13184400
Dec 14, 2016	7337220
Dec 13, 2016	4486190
Dec 12, 2016	9119700
Dec 11, 2016	5545500
Dec 10, 2016	8108370
Dec 09, 2016	12916300
Dec 08, 2016	16042700
Dec 07, 2016	22229600
Dec 06, 2016	50671200
Dec 05, 2016	17344500
Dec 04, 2016	8450790
Dec 03, 2016	12112100
Dec 02, 2016	18579800
Dec 01, 2016	5882340
Dec 19, 2016	5894760
Nov 30, 2016	10626600
Nov 29, 2016	17000400
Nov 28, 2016	6046890
Nov 26, 2016	3505990
Nov 25, 2016	7669720
Nov 24, 2016	15987800
Nov 23, 2016	4812310

Slika 7.6: Prikaz grafa za korisničke određene paremetre

Poglavlje 8

Zaključak

U ovom projektu vidjeli smo što je to pojam big data, kako je on povezan sa polustrukturiranim podacima, kakvu oni strukturu imaju i zašto su korisni. Isto tako smo se upoznali sa NoSQL-om te uvidjeli razlike sa klasičnim SQL-om odnosno relacijskim modelom. Vidjeli smo i što je to MongoDB, čemu služi, kako se koristi. Kroz rad je bilo prikazano kako instalirati MongoDB na Ubuntu 16.04 kako se koristi, kako smo kreirali bazu i dodali skupove podataka, na koji način se python može povezati na MongoDB bazu te kako smo pomoću pythona te podatke prikazali na sučelje LibreOffice Calca. S pythonom se to dalo lijepo odraditi ali nedostaje još malo nekih dodatnih mogućnosti sa bibliotekama te je teško pronaći dobre primjere i dokumentaciju za korištenje biblioteka koje mogu radit sa LibreOffice Calcom. Međutim, na kraju se može uvidjeti da su se uspješno napravili određeni grafovi iz kojih se može izvući neka analiza podataka iz baze, te bi se lako aplikacija mogla nadograditi da ima još neke dodatne grafove za recimo još nižu razinu po mjesecu ili da se uzmu samo određene kriptovalute za određene parametre.

Bibliografija

2016. *MongoDB Docs*.

2016. *Semi-structured data*.

2017. *Big Data*.

2017. *NoSQL*.

Machin, J., 2017. *xlwt 1.3.0*.

Pojman, M., 2017. *pyoo 1.2*.