Izrada programske aplikacije za interakciju i trgovanje na kripto burzi

Novak, Leon

VELEUČILIŠTE U BJELOVARU
PREDDIPLOMSKI STRUČNI STUDIJ RAČUNARSTVO

IZRADA PROGRAMSKE APLIKACIJE ZA INTERAKCIJU I TRGOVANJE NA KRIPTO BURZI

Završni rad br. 11/RAČ/2021

Leon Novak

Bjelovar, listopad 2022.

obrazac ZR - 001

Veleučilište u Bjelovaru

Trg E. Kvaternika 4, Bjelovar

1. DEFINIRANJE TEME ZAVRŠNOG RADA I POVJERENSTVA

Kandidat: **Novak Leon** Datum: 31.08.2021. Matični broj: 001981

JMBAG: 0314019527

Kolegij: **UVOD U PROGRAMIRANJE**

Naslov rada (tema): **Izrada programske aplikacije za interakciju i trgovanje na kripto burzi**

Područje: **Tehničke znanosti** Polje: **Računarstvo**

Grana: **Programsko inženjerstvo**

Mentor: **Ivan Sekovanić, mag.ing.inf.et comm.techn.** zvanje: **predavač**

Članovi Povjerenstva za ocjenjivanje i obranu završnog rada:

1. **dr.sc. Zoran Vrhovski, predsjednik**
2. **Ivan Sekovanić, mag.ing.inf.et comm.techn., mentor**
3. **Ante Javor, struč.spec.ing.comp., član**

2. ZADATAK ZAVRŠNOG RADA BROJ: 11/RAČ/2021

U radu je potrebno razviti vlastitu aplikaciju koja je sposobna samostalno, kroz interakciju s programskim sučeljem kripto burze, izvršavati trgovinu kriptovalutom. Aplikacija mora podržavati konfiguraciju više različitih parametara koji određuju rad algoritma za trgovinu, bilježiti osnovnu statistiku o svom radu te po potrebi izvještavati korisnika o unaprijed definiranim događajima. U radu treba prezentirati primjer rada navedene aplikacije.

Zadatak uručen: 31.08.2021. Mentor: **Ivan Sekovanić, mag.ing.inf.et comm.techn.**

REPUBLIKA HRVATSKA 2 VELEUČILIŠTE U BJELOVARU

Sadržaj

1. UVOD 1

2. ŠTO SU KRIPTOVALUTE? 2

2.1. Princip rada kriptovaluta 2

3. RAZVOJNO OKRUŽENJE 4

3.1. Python 4
3.1.1. Korištene Python biblioteke 4

3.2. Visual Studio Code 5

4. IZRADA APLIKACIJE ZA TRGOVANJE 6

4.1. Glavni dio programa 6

4.2. Prijava 8

4.3. Prikaz stanja 10

4.4. Prikaz cijena 10

4.5. Opcije algoritma 11
4.5.1. Unos parametara algoritma 12

4.6. Algoritmi 15
4.6.1. Funkcija ema 17
4.6.2. Funkcija macd 20

4.7. Klase 21
4.7.1. Order 21
4.7.2. Session 22
4.7.3. Statistics 24

4.8. Prikaz statistika 24
4.8.1. Primjeri statistika 26

5. ZAKLJUČAK 29

6. LITERATURA 30

7. OZNAKE I KRATICE 33

8. SAŽETAK 34

9. ABSTRACT 35

1. UVOD

U zadnjih nekoliko godina kriptovalute su eksplodirale na tržištu te su ubrzo postale jedna od najpopularnijih metoda za ulaganje kod mladih. S obzirom na to da je kripto tržište jako promjenjivo te cijena određenih kriptovaluta može više puta varirati svakodnevno, pojavila se potražnja za programskim rješenjima koja automatiziraju procese praćenja, kupovanja, prodavanja te analize kriptovaluta.

U ovom završnom radu popraćen je proces kreiranja programa koji komunicira s Binance kripto burzom. Preko komunikacije sa aplikacijskim programskim sučeljem program dohvaća podatke o računu korisnika, općenitom stanju tržišta te po korisnikovim unesenim parametrima izvršava kupnju i prodaju kriptovaluta s ciljem povećanja portfolija i bilježenja statistika.

U drugom poglavlju opisani su nastanak kriptovaluta, općeniti princip rada, različiti tipovi te njihova sigurnost. U trećem poglavlju navedene su tehnologije korištene za izradu ovoga rada te inačice svake tehnologije kako bi korisnik mogao podesiti svoje okruženje za korištenje programa. U četvrtom poglavlju opisati će se datotečna struktura rada, pokretanje istoga te objasniti cijeli programski kod, tako da pratimo korisnikovo kronološko korištenje programa, uz sve njegove opcije za interakciju s programom. Na kraju rada dan je zaključak.

2. ŠTO SU KRIPTOVALUTE?

Prema [1] kriptovalute su digitalni oblik novca kojim se fokus stavlja na decentralizaciju fiskalnog sustava. Mreža kojom se provode transakcije, naziva se *blockchain* što dolazi od doslovnog prijevoda blok (engl. *block*) i lanac (engl. *chain*) iz razloga što se svaka nova transakcija veže uz prethodni blok i nastavlja na njega, tvoreći tako novi jedinstveni blok s unikatnom vrijednošću. Tvorac ili tvorci *blockchain* tehnologije poznat/i su pod sinonimom Satoshi Nakamoto. Oni su zaslužni za razvoj Bitcoina te bijelu knjigu (engl. *Whitepaper*) na kojem je naveden kratki sažetak implementacije *blockchain* tehnologije te samog Bitcoina kao predvodnika svih kriptovaluta.

2.1. Princip rada kriptovaluta

Kriptovalute funkcioniraju koristeći različite metode, odnosno algoritme. Primjeri metoda su: *Proof of Work* (PoW), *Proof of Stake* (PoS), *Proof of History* (PoH) itd. PoS je metoda na kojoj je baziran Bitcoin. Ta metoda funkcionira na način da ljudi preko svojih osobnih računala „rudare“ na *blockchain* platformi, odnosno daju energiju osobnih računala kojom programi rješavaju kompleksne zadatke te zauzvrat dobivaju nagradu. Važno je napomenuti kako računalo koje riješi zadatak, dobiva nagradu, dok ostala računala ne dobiju ništa. Iz tog razloga, ljudi se grupiraju u skupine, takozvane bazene (engl. *pool*) kako bi šansa za dobivanje nagrade bila veća. Uzročno posljedično nagrada se dijeli svakome u bazenu. Druga metoda je PoS. Ona funkcionira po principu, što je veći udio kriptovatue založen, veće su šanse da validator bude odabran za potvrdu novog bloka u *blockchainu*. Validator je čvor *blockchain* mreže koji je založio kriptovalute. Također, validatori koji ulože u takozvani ulog (engl. *stake*), zauzvrat dobivaju udio od svake transakcije te što duže drže svoj novac unutra veći je povrat. Najpoznatija valuta po metodi PoS jest Ethereum. Glavne razlike u PoW i PoS jesu da PoW kapacitet rudarenja ovisi o resursima računala. Rudari su plaćeni prema broju riješenih kompleksnih zadataka na računalu. Transakcije traju ovisno o naknadi koju je korisnik spreman platiti te je ovo najsporija metoda. Da bi hakeri hakirali mrežu potrebno je imati pod kontrolom 51% svih umreženih računala. PoS ovisi o količini uloženog novca koji održava mrežu, plaćeni su od naknada koje su dodijeljene transakcijama te su transakcije brže no ne i jeftinije. Da bi se hakirala PoS mreža potrebno je posjedovanje 51% svih novčića određene kriptovalute na tržištu. Zadnja metoda je PoH

koja je kombinacija PoW-a i PoS-a. Napravljena je kao naprednija, bolja i učinkovitija inačica ovih metoda, a najpoznatija kriptovaluta čiji je rad zasnovan na PoH metodi jest Solana. Transakcije se izvršavaju u petnaestak sekundi te su naknade niske. Istovremeno, rudare se novčići i zalažu na mreži. PoH je optimalan za industriju igara te se sve moderne „igraj da zaradiš“ (engl. *play to earn)* igrice baziraju baš na istom modelu. Svaki od tih modela dijeli se na glavnu mrežu i mrežu za razvoj (engl. *testnet*). Glavne mreže su Bitcoin, Ethereum, Solana itd. Preko njih se odvijaju sve transakcije. Mreže za razvoj služe kao potpora glavnoj te imaju zasebne funkcije u sklopu istog modela.

3. RAZVOJNO OKRUŽENJE

3.1. Python

Python je objektno orijentirani programski jezik koji koristi interpreter za njegovo izvršavanje. Zbog njegove jednostavnosti i širokih mogućnosti korištenja nazivamo ga programskim jezikom opće namjene. Neka od područja primjene su: analitika, strojno učenje, automatsko preuzimanje i spremanje sadržaja s interneta (engl. *web scraping*), web aplikacije, digitalna obrada slika, automatizacija raznih poslova, razvoj igara itd. Široku primjenu Pythona omogućuje njegova velika zajednica koja je objavila mnoštvo biblioteka, koje omogućuju i neiskusnom programeru pisanje kompleksnih programa u minimalnom vremenu [2].

3.1.1. Korištene Python biblioteke

U ovom segmentu će se navesti Python (inačica 3.9.1) biblioteke korištene u projektu, većinom preuzete preko Pythonovog programa za instalaciju biblioteka pip (inačica 22.0.4) i ukratko ih opisati:

- os - Pythonova integrirana biblioteka za komunikaciju s operacijskim sustavom.
- json - Pythonova integrirana biblioteka za rad podacima tipa JSON.
- time - Pythonova integrirana biblioteka s funkcijama vezane uz vrijeme kao npr. dohvaćanje trenutnog vremena i pauziranje programa.
- pickle - Pythonova integrirana biblioteka za serijalizaciju objekta
- pathlib - Pythonove integrirane klase koje predstavljaju staze datotečnog sustava prikladne za različite operativne sustave, s podrškom za ulazno/izlazne operacije.
- datetime - Pythonove integrirane klase za manipulaciju datumima i vremenima.
- python-binance (inačica 1.0.12) - Aplikacijsko programsko sučelje za komunikaciju s Binance kripto mjenjačnicom [3].
- cryptography (inačica 3.4.7) - Biblioteka za kriptografske algoritme kao što su simetrične šifre, sažetci poruka i funkcije za derivaciju ključeva [4].
- pandas (inačica 1.2.2) - Biblioteka za analizu, manipulaciju i prikaz podataka [5].

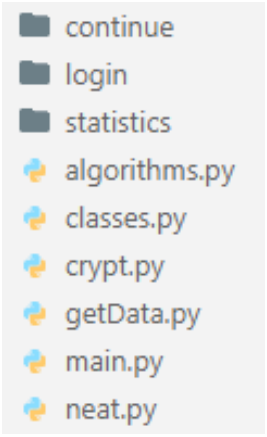
- bta-lib (inačica 1.0.0) - Na pandasu bazirana biblioteka za izračun tehničkih indikatora.

3.2. Visual Studio Code

Visual Studio Code (skraćeno VS Code) je Microsoftov uređivač koda izrađen s Electron razvojnim okvirom (engl. *Framework*), za Windows, Linux i macOS. On je pojednostavljeni uređivač koda s podrškom za otklanjanje pogrešaka (engl. *Debugging*), pokretanja programa i verzioniranje pomoću integriranog Git sustava. Cilj mu je pružiti upravo alate koji su razvojnom programeru potrebni za brzu izradu koda i otklanjanje pogrešaka. Pomoću isticanja sintakse, inteligentnog dovršavanja koda, refaktoriranja koda i gotovih predloška, postiže se puno brži i lakši razvoj programa i web aplikacija. Također mu je kod popularnosti pomogla zajednica korisnika razvojem proširenja, kako bi dodatno olakšala i pojednostavila posao programera. U ovom projektu korištena su Microsoftova proširenja Python i Pylance, za lakše pisanje, izvršavanje i otklanjanje pogrešaka Python koda [6].

4. IZRADA APLIKACIJE ZA TRGOVANJE

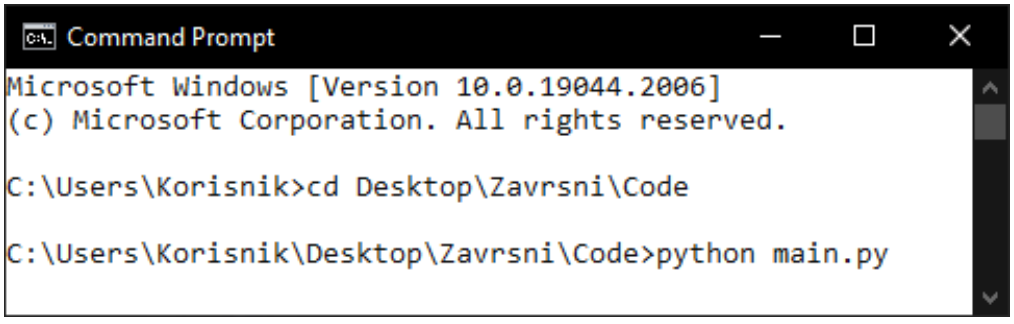
Programski kod ovoga projekta smješten je u mapu Code (slika 4.1) i grupiran je prema svojim funkcionalnostima u zasebne Python datoteke.



Slika 4.1: Prikaz direktorija Code u programu VS Code

4.1. Glavni dio programa

Prije prvoga pokretanja programa potrebno je na Binanceu izraditi korisnički račun te zatražiti javni i privatni ključ, kako bi mogli komunicirati s API servisom. Treba se napomenuti da se koristi različiti par ključeva za *mainnet* (produkcija) i *testnet* (simulirano okruženje za potrebe testiranja). Program se pokreće preko naredbenog retka, pozicioniranjem u direktorij gdje se nalazi datoteka *main.py* te pokretanje iste pomoću Pythonovog interpretera, što je vidljivo na slici 4.2.



Slika 4.2: Pokretanje programa pomoću naredbenog retka

U datoteci main.py (programski kod 4.1) uvezene su funkcije iz vlastitih biblioteka, definirana je funkcija *main* i prema Python standardima poziv funkcije *main*, ako je datoteka pokrenuta kao izvršna datoteka, a ne uvezena kao biblioteka.

Programski kod 4.1: Datoteka main.py

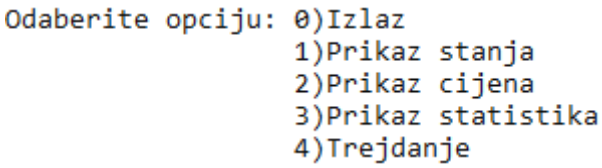
```
#vlastite biblioteke
from crypt import login
from getData import getBalance, getPrice
from neat import clr, mainOptions, leave,
    algorithOptions, statisticOptions

def main():
    clr()
    client = login()

    while True:
        mainOptions()
        option = input()
        if(option == "0"):
            leave()
        elif(option == "1"):
            getBalance(client)
        elif(option == "2"):
            getPrices(client)
        elif(option == "3"):
            statisticOptions(client)
        elif(option == "4"):
            algorithOptions(client)
        else:
            clr()
            print("Pogrešan unos")

if __name__ == "__main__":
    main()
```

Pokretanjem funkcije *main* radi preglednosti prvo se očisti naredbeni redak pomoću funkcije *clr* (programski kod 4.2) koja pomoću biblioteke os provjeri o kojem se operativnome sustavu radi te prema tome mu prosljeđuje odgovarajuću naredbu.

Programski kod 4.2: Funkcija clr

```
def clr():
    command = "clear"
    if os.name in ("nt", "dos"):
        command = "cls"
    os.system(command)
```

Poslije brisanja ekrana instancira se objekt client koji se koristi za svu komunikaciju s Binance servisom, pomoću vlastite funkcije *login* koja će se detaljnije objasniti u sljedećem segmentu. Beskonačnom se petljom prikazuje glavni izbornik pomoću funkcije *mainOptions* (programski kod 4.3) te se od korisnika čeka odabir jedne od opcija na slici 4.3.



Slika 4.3: Glavni izbornik

Programski kod 4.3: Funkcija mainOptions

```
def mainOptions():
    print(
    '''Odaberite opciju: 0)Izlaz
                  1)Prikaz stanja
                  2)Prikaz cijena
                  3)Prikaz statistika
                  4)Trejdanje'''
    )
```

Pomoću korištenja 3 jednostruka navodnika na početku i kraju teksta u *print* funkciji dobije se veća kontrola nad formatiranjem teksta te prijelaz u sljedeći red, bez korištenja odgovarajućih znakova.

4.2. Prijava

Funkcija *login* (programski kod 4.4) provjerava postoji li u mapi login datoteka s ključem koji se koristi za dešifriranje pristupnih podataka za Binance API, ako je nema traži se unos API ključa i API tajne te se pomoću njih inicijalizira objekt tipa klijent i poziva funkcija za dobivanje računa kako bi provjerili ispravnost podataka. Ako su podaci neispravni dobiva se poruka greške te se ponavlja procedura. Pomoću Fernet kriptografskog algoritma generira se ključ koji se binarno zapisuje u datoteku kako bi se kasnije koristio [4]. Pristupni podaci zapisuju se u JSON formatu kao tekst u datoteku. Datoteka se poslije toga binarno pročita i šifrira, kako bi se podaci u tom obliku opet binarno zapisali u datoteku i na kraju se vraća objekt *client*. Ako postoji datoteka s ključem binarno se pročitaju ključ i šifrirani podaci, kako bi se dešifrirali i prebacili u oblik prigodan za kreiranje klijenta koji se vraća za daljnje korištenje.

Programski kod 4.4: Funkcija login

```
def login():
    #provjera prve prijave
    if not exists("login/" + keyFile):
        #provjera ispravnosti podataka za prijavu
        while True:
            try:
                apiKey = input("Unesite api key: ")
                clr()
                apiSecret = input("Unesite api secret: ")
                clr()
                client = Client(apiKey, apiSecret, testnet = True)
                #greše se javljaju tek kod poziva funcija
                client.get_account()
                break
            except:
                print("Dogodila se greška, pokušajte ponovo!")

        #priprema za enkripciju
        key = Fernet.generate_key()
        with open("login/" +keyFile, "wb") as fileKey:
            fileKey.write(key)

        #zapis u JSON formatu
        dict={
            "apiKey" : apiKey,
            "apiSecret" : apiSecret
        }
        with open("login/" +loginFile, "w") as file:
            json.dump(dict, file)

        #čitanje binarnog zapisa
        with open("login/" +loginFile, "rb") as file:
            original = file.read()

        #enkripcija
        encrypted = Fernet(key).encrypt(original)

        #binarni zapis kriptiranih podataka
        with open("login/" +loginFile, "wb") as file:
            file.write(encrypted)
        return client

    #čitanje kluča i dekripcija podataka za login
    with open("login/" +keyFile, "rb") as filekey:
        key = filekey.read()
    with open("login/" +loginFile, "rb") as enc_file:
        encrypted = enc_file.read()
    decrypted = Fernet(key).decrypt(encrypted)
    decrypted = decrypted.decode("utf8").replace("'", '"')
    decrypted = json.loads(decrypted)

    client = Client(decrypted["apiKey"], decrypted["apiSecret"],
                    testnet = True)
    return client
```

4.3. Prikaz stanja

Za prikaz stanja koristi se funkcija *getBalance* (programski kod 4.5) u koju se šalje client kao ulazni parametar. Preko njega šalje se zahtjev Binance mreži za informacije o korisničkome računu, gdje se pod ključem *balances* nalazi polje svih kriptovaluta koje korisnik posjeduje. Pomoću simbola i slobodnog stanja kriptovalute informira se korisnika o trenutnom stanju. Za korištenje ove automatizirane aplikacije (engl. *bot*), bitno je da se posjeduje dovoljno USDT-a (Tethera), pošto se ona koristi za kupovanje ostalih kriptovaluta i kod prodaje se opet sve prebacuje u USDT. Nakon ispisa čeka se da korisnik unese nulu, kako bi se obrisao ekran i program nastavio s radom.

Programski kod 4.5: Funkcija getBalance

```
def getBalance(client):
    balance=client.get_account()

    while True:
        clr()
        for token in balance['balances']:
            print(token['asset'] + " : " + token['free'])

        print('0)Povratak')
        if input() == '0':
            break

    clr()
```

4.4. Prikaz cijena

Funkcija *getPrices* (programski kod 4.6) pomoću ulaznog parametra *client* s Binance mreže povuče sve tečajeve razmjena svih kriptovaluta koje su dostupne na burzi. Pomoću zadnjih 3 znamenaka simbola razmjene filtriraju se samo one kojima je odredišna valuta američki dolar, pošto je središnja valuta programa USDT, koji je jednak američkom dolaru. Nakon toga podaci se formatiraju u oblik prikladan za čitanje i čeka se korisnikov unos nule, kako bih se ekran očistio i program nastavio izvršavati.

Programski kod 4.6: Funkcija getPrices

```
def getPrices(client):
    prices = client.get_all_tickers()

    while True:
        clr()
        for token in prices:
            if token['symbol'][-3:] == 'USD':
                print(token['symbol'][:-4] + "-" +
                      token['symbol'][-3:] + " : " + token['price'])

        print('0)Povratak')
        if input() == '0':
            break

    clr()
```

4.5. Opcije algoritma

Za početak trgovanja poziva se funkcija *algorithOptions* (programski kod 4.7) pomoću koje korisnik odabire algoritam koji će se koristiti za sesiju. Ovisno o tome koji se algoritam odabrao slijedi unos simbola kriptovalute koja se želi kupovati odnosno prodavati i parametra koji opisuju ponašanje algoritma. Funkciji *algorithParameters* prosljeđuje se klijent i naziv algoritma. Ako je vrijednost varijable *parameters* nula, ponovno se bira algoritam. No, ako je vrijednost varijable *parameters* jednaka „continue“, nastavlja se prošla nezavršena sesija, a u suprotnom se pokreće nova sesija s unesenim parametrima.

Programski kod 4.7: Funkcija algorithOptions

```
def algorithOptions(client):
    clr()
    while True:
        print(
    '''Odaberite algoritam: 0)Povratak
                        1)MACD
                        2)EMA'''
        )
        inp = input()
        clr()

        #izlaz iz trejdanja
        if(inp == "0"):
            break

        elif(inp == "1"):
            #unos parametra
            symbol, parameters = algorithParameters(client, "macd")
```

```
            #ponovno biranje algoritma
            if parameters == 0:
                clr()
                continue

            #nastavak sesije
            if parameters == "continue":
                macd(client, symbol, [], True)
            else:
                #ulaz u novu sesiju
                macd(client, symbol, parameters, False)

        elif(inp == "2"):
            symbol, parameters = algorithParameters(client, "ema")
            if parameters == 0:
    clr()
                continue
            if parameters == "continue":
                ema(client, symbol, [], True)
            else: ema(client, symbol, parameters, False)

        else:
            print("Pogrešan unos")
```

4.5.1. Unos parametara algoritma

Funkcija *algorithParameters* (programski kod 4.8) od korisnika pomoću pomoćne funkcije *symbolInput* korisniku daje odabir kriptovalute koja će se koristiti za razmjenjivanje u sesiji. Simbol se prema programskom kodu 4.9 postavlja tako da se odabere vrijednost enumeriranog polja svih dostupnih kriptovaluta, izuzev *stables* kojima vrijednost niti ne raste, niti ne pada. Nakon unosa program provjerava postoji li u mapi *continue* datoteka, koja u nazivu sadrži simbol odabrane kriptovalute i naziv prethodno odabranog algoritma. Ukoliko postoji vraća se *symbol* zajedno sa tekstom „continue“ za nastavak nezavršene sesije, a u suprotnom pozivaju se pomoćne funkcije za unos pojedinih parametara koji se kao polje, uz simbol vrate za pokretanje nove sesije. Korisnik u bilo kojem koraku može pritisnuti 0 kako bi se vratio na ponovni odabir algoritma. U tom slučaju se iz pomoćnih funkcija izlazi bez prosljeđivanja vrijednosti te u tom slučaju funkcija *algorithParameters* vraća polje sa dvije nule.

Programski kod 4.8: Funkcija algorithParameters

```
def algorithParameters(client, algorithm):
    symbol = symbolInput(client)
    if symbol == None: return []
    symbol += 'USDT'
    #provjera datoteke za nastavak
    path = "./continue/" + symbol + "_" + algorithm
    if os.path.exists(path):
        return symbol, "continue"
    clr()
    money = moneyInput(client)
    if money == None: return [0, 0]
    clr()
    takeMoney = takeMoneyInput(money)
    if takeMoney == None: return [0, 0]
    clr()
    leaveMoney = leaveMoneyInput(money)
    if leaveMoney == None: return [0, 0]
    clr()
    amount = amountInput(money)
    if amount == None: return [0, 0]
    clr()
    takeProfit = takeProfitInput()
    if takeProfit == None: return [0, 0]
    clr()
    stopLoss = stopLossInput()
    if stopLoss == None: return [0, 0]
    clr()

    return symbol, [money, takeMoney, leaveMoney, amount,
                    takeProfit, stopLoss]
```

Programski kod 4.9: Funkcija symbolInput

```
def symbolInput(client):
    balance = client.get_account()
    balance = [x for x in balance['balances'] if x['asset'] not in
stables]
    clr()

    while True:
        print("Odaberite željenu kriptovalutu: 0)Povratak")
        for i, token in enumerate(balance, start=1):
            print("                                    "+str(i)
                  + ")" + token['asset'])
        inp = input()
        if inp == '0':
            break
        try:
            return balance[int(inp)-1]['asset']
        except:
            print("Pogrešan unos")
```

U nastavku će se ukratko opisati parametri koji ulaze u sesiju:

- money - Novčani ulog u USDT-u
- takeMoney - Granica za izlaz iz sesije kod profita
- leaveMoney - Granica za izlaz iz sesije kod gubitka
- amount - Iznos koji se koristi za svaku pojedinu razmjenu
- takeProfit - Granica za prodaju kupljene kriptovalute s profitom
- stopLoss - Granica za prodaju kupljene kriptovalute s gubitkom

Ostale pomoćne funkcije postavljaju pojedine varijable na korisnikove unesene brojčane vrijednosti, nakon određenih provjera. Funkcija *moneyInput* (programski kod 4.10) provjerava ima li korisnik dovoljno slobodnih sredstava na računu. Funkcije *takeMoneyInput* i *leaveMoneyInput* provjeravaju je li unos veći, odnosno manji od parametra money. Funkcije *amountInput*, *takeProfitInput* i *stopLossInput* koriste se za unos cjelobrojnih vrijednosti koje označavaju postotke.

Programski kod 4.10: Funkcija moneyInput

```
def moneyInput(client):
    balance = float(client.get_asset_balance(asset="USDT")["free"])
    while True:
        inp=input("Unesite ulog za odabranu kriptovalutu u USDT-u ili
0 za povratak: ")
        if inp == "0":
            break
        try:
            inp = float(inp)
            if inp < 0:
                raise Exception
        except:
            clr()
            print("Pogrešan unos")
            continue
        if inp > balance:
            clr()
            print("Nedovoljna sredstva na računu:", balance)
            continue
        return inp
```

4.6. Algoritmi

Kako bi „uspješno“ razmjenjivali kriptovalute potrebna je velika količina povijesnih podataka radi analize i izračuna tehničkih indikatora, prema kojima se kupuje, odnosno prodaje. Za navedeno koristi se funkcija *getKlines* (programski kod 4.11) kojoj se zadaje: simbol kriptovalute o kojoj se zatražuju podaci, vremenski interval između podatkovnih zapisa i vrijeme koliko daleko iz prošlosti se vuku podaci. Navedeni parametri šalju se s klijentovom *get_historical_klines* funkcijom u *try*/*except* bloku prema Binance mreži, kako bi u slučaju greške mogli nastaviti s prošlim podacima te kasnije pokušati opet. Iz dobivenih podataka za svaki zapis brišu se nepotrebni stupci te se smještaju u pandas podatkovni okvir. Pandas je korišten zbog svog otvorenog koda te svoje brzine, fleksibilnosti i jednostavnosti kod analize i manipulacije velikih količina podataka. Potrebni podaci su: datum/vrijeme zapisa u milisekundama, cijena na početku intervalu, cijena na kraju intervalu, najviša cijena u intervalu, najniža cijena u intervalu i volumen koji označava promet u intervalu. Kao indeks podatkovnog okvira uzimaju se stupac datum, pošto je on jedinstven i pretvara se vrijednost iz milisekunda u zapis oblika datum/vrijeme, nakon čega se podaci vraćaju.

Programski kod 4.11: Funkcija getKlines

```
def getKlines(client, symbol, interval, lookback):
    try:
        bars = client.get_historical_klines(symbol, interval,
lookback)
    except Exception as e:
        print("Greška kod dohvata podataka: ", e)

    for line in bars:
        del line[6:]
    klines = pd.DataFrame(bars, columns=['date', 'open', 'high',
'low', 'close', 'volume'])
    klines.set_index('date', inplace=True)
    klines.index = pd.to_datetime(klines.index, unit='ms')
    return klines
```

Nakon što je korisnik odabrao algoritam za trgovanje i željenu kriptovalutu te unio parametre za sesiju, program ulazi u način rada bez potrebe za korisnikovom interakcijom. Kako bi se korisniku prikazalo pravilno izvršavanje algoritma, program svake minute nakon

dohvata povijesnih podataka ispiše trenutačno stanje sesije. U svakome zapisu prikazani su osnovni podaci korišteni u algoritmu koji će se kasnije detaljnije opisati. Na slici 4.4 prikazani su ispisi trenutačnih stanja, odvojeni s podvlakama.

```
------------------
17:32:00
Price: 53.65
Buy price: 53.62
Take price: 55.7648
Stop price: 52.547599999999996
Profit : 0
Open position
------------------
17:33:00
Price: 53.63
Buy price: 53.62
Take price: 55.7648
Stop price: 52.547599999999996
Profit : 0
Open position
------------------
17:34:00
Price: 53.63
Buy price: 53.62
Take price: 55.7648
Stop price: 52.547599999999996
Profit : 0
Open position
------------------
17:35:00
Price: 53.61
Buy price: 53.62
Take price: 55.7648
Stop price: 52.547599999999996
Profit : 0
Open position
------------------
17:36:00
Price: 53.61
Buy price: 53.62
Take price: 55.7648
Stop price: 52.547599999999996
Profit : 0
Open position
------------------
17:37:00
Price: 53.61
Buy price: 53.62
Take price: 55.7648
Stop price: 52.547599999999996
Profit : 0
Open position
```

Slika 4.4: Primjer rada algoritma

4.6.1. Funkcija ema

Funkcija *ema* (programski kod 4.12) prima varijable: *client, symbol, parameters* i *continued* koje će se naknadno objasniti. Na početku funkcije inicijaliziraju se varijable: *start, orders, symbol, ema8, ema21, prev_ema8, prev_ema21, buyPrice, sellPrice, profit* i *bought*. U varijablu *start* zapisuje se trenutačno vrijeme tj. početak sesije, u formatu „dan.mjesec.godina sati:minute:sekunde“. Varijabla *orders* označava prazno polje narudžbi, koje će se kasnije puniti. Varijable *ema8, ema21, prev_ema8* i *prev_ema21* na početku nemaju vrijednost. *BuyPrice, sellPrice* i *profit* postavlja se na nulu te se varijabla *bought* postavlja na laž (engl. *False*). Ovisno o tome dali je prethodno spomenuta varijabla *continued* istinita (engl. *True*) program prebriše vrijednosti varijabli *parameters, start* i *orders* s vrijednostima koje se dobiju iz prethodne sesije, pomoću objekta *Statistics* koji će se kasnije objasniti. Pomoću polja *parameters* postavljaju se varijable: *money, takeMoney, leaveMoney, amount, takeProfit* i *stopLoss*. Svaka kriptovaluta ima svoju minimalnu vrijednost koja se može kupovati, odnosno prodavati te se ta vrijednost pomoću funkcije *get_symbol_info* sprema u varijablu *info*. Pomoću *for* petlje se navedenoj varijabli prebroji broj decimalna mjesta, koji se zapisuje u varijablu *precision*.

U *try* bloku smještena je beskonačna petlja koja formatirano ispisuje trenutačno vrijeme i pomoću prije spomenute metode *getKlines* dohvaća trenutačne informacije o tržištu. S navedenim podacima izračunavaju se tehnički indikatori (signali za kupovanje/prodavanje) *ema8* i *ema21,* koje se ovisno o njihovom periodu zapisuju u odgovarajuće varijable te se trenutačna cijena kriptovalute zapisuje u varijablu *price*. Ispisuju se trenutačna cijena, cijena po kojoj je kupljena kriptovaluta, donja i gornja granica za prodaju, trenutačni profit koji može biti pozitivan ili negativan te se obavještava korisnika ako postoji trenutačno otvorena pozicija.

U programskom kodu slijedi uvjet kupnje koji provjerava je li *ema8* prerasla *ema21,* ako je prošla barem jedna iteracija petlje i nema otvorene pozicije. U slučaju da je uvjet istinit računa se količina kriptovalute koja će se kupiti, tako da se prije određena varijabla *amount* podjeli s trenutačnom cijenom i pomoću varijable *precision* zaokruži na najveći dopušteni broj decimala. Funkcijom *order_market_buy* pomoću simbola i prije spomenute količine kupuje se kriptovaluta, nakon čega se varijabla *bought* promijeni u istinu. Za završetak kupovine zapisuju se u varijable kupovna cijena i vrijeme kupnje, koje se uz količinu zapisuju u objekt tipa *Order* kako bi bio ubačen u polje *orders*. Odmah poslije u kodu smješten je uvjet prodaje koji ukoliko ima otvorene pozicije, provjerava je li cijena

veća, odnosno manja od prije definiranih cijena za prodaju s profitom, odnosno s gubitkom. U slučaju da je uvjet istinit pomoću funkcije *order_market_sell* i kod kupnje definirane količine, nakon čega se prodaje kriptovaluta i postavlja varijablu *bought* u laž. U određene varijable zapisuju se podaci o prodajnoj cijeni i vremenu prodaje, koje se pomoću metode *sell* spremaju u kod kupnje kreirani objekt *Order*. Razlika prodajne i kupovne cijene se pomnoži s količinom kupljene kriptovalute i vrijednost se nadodaje na profit, nakon čega se varijable *buyPrice* i *sellPrice* ponovno postavljaju na nulu. Za završetak prodaje provjerava se je li ulog sesije uvećan za profit pao ispod ili nadmašio uvjete za izlaz iz sesije te ako je izlazi se iz beskonačne petlje, tj. završava se sesija. Ukoliko sesija nije završila, vrijednosti varijabli *ema8* i *ema21* spremaju u pomoćne varijable za korištenje u sljedećoj iteraciji. Program čeka početak sljedeće minute kako bi nastavio s petljom.

Na kraju sesije zabilježi se vrijeme završetka koje se uz sve ostale informacije o sesiji koristi za kreiranje objekta tipa *Session*, koji se sprema pomoću njegove metode *save*. Ukoliko je sesija završila s otvorenom pozicijom, u mapi continue kreira se datoteka čije postojanje označuje da se kod sljedećeg poziva algoritma s istom kriptovalutom nastavlja trenutačna sesija. Ime datoteke kombinacija je simbola kriptovalute, podvlake i naziva algoritma.

Programski kod 4.12: Funkcija ema

```
def ema(client, symbol, parameters, continued):
    start = datetime.now().strftime("%d.%m.%Y. %H:%M:%S")
    orders = []
    symbol = symbol
    ema8 = ema21 = prev_ema8 = prev_ema21 = None
    buyPrice = sellPrice = profit = 0
    bought = False

    if(continued):
        parameters, start, orders = Statistics("ema",
symbol).continueSession()
        buyPrice = orders[-1].buyPrice
        quantity = orders[-1].quantity
        bought = True

    money, takeMoney, leaveMoney, amount, takeProfit, stopLoss =
parameters
    take = 1 + takeProfit
    stop = 1 - stopLoss
    info = client.get_symbol_info(symbol)['filters'][2]['minQty']
    info = info.split(".",1)[1]
    for i, value in enumerate(info, start=1):
        if value != "0":
            precision = i
```

```
    try:
        while True:
            print("------------------")
            print(datetime.now().strftime("%H:%M:%S"))
            df=getKlines(client, symbol, '1m', '1 day ago
UTC').astype(float)
            df['ema8'] = bta.ema(df, period=8).df
            df['ema21'] = bta.ema(df, period=21).df
            ema8=df.ema8[-1]
            ema21=df.ema21[-1]
            price=float(df['close'][-1])

            print("Cijena:", price)
            print("Kupovna cijena:", buyPrice)
            print("Prodajna cijena profitom:", buyPrice*take)
            print("Prodajna cijena gubitkom:", buyPrice*stop)
            print("Profit :", profit)

            if(bought):
                print("Otvorena pozicija")

            if(ema8 > ema21 and prev_ema8 and not bought):
                if(prev_ema8 < prev_ema21):
                    try:
                        quantity = round(amount/price, precision)
                        buy_order = client.order_market_buy(symbol =
symbol, quantity = quantity)
                        bought = True
                        buyPrice = price
                        buyTime = datetime.now().strftime("%d.%m.%Y.
%H:%M:%S")
                        orders.append(Order(quantity, buyPrice, buyTime))

                    except BinanceAPIException as e:
                        print('Dogodila se greška:', e)

                    except BinanceOrderException as e:
                        print('Dogodila se greška:', e)

                    except Exception as e:
                        print('Dogodila se greška:', e)


            if(bought and (price > buyPrice*take or price
<  buyPrice*stop)):
                try:
                    sell_order = client.order_market_sell(symbol = symbol,
quantity = quantity)
                    bought = False
                    sellPrice = price
                    sellTime = datetime.now().strftime("%d.%m.%Y.
%H:%M:%S")
                    profit += (sellPrice-buyPrice) * quantity
                    orders[-1].sell(sellPrice, sellTime)
                    buyPrice = sellPrice = 0
```

```
                    if(money+profit >= takeMoney or money+profit <=
leaveMoney):
                        break

                except BinanceAPIException as e:
                    print(e)

                except BinanceOrderException as e:
                    print(e)

            prev_ema8 = ema8
            prev_ema21 = ema21

            sleep(60 - time() % 60)

    except KeyboardInterrupt:
        print("Manualno avršena sesija!")

    except Exception as e:
        print("Desila se greška :", e)

    end = datetime.now().strftime("%d.%m.%Y. %H:%M:%S")
    session = Session(symbol, money, takeMoney, leaveMoney, amount,
takeProfit, stopLoss, profit, orders, start, end)
    session.save("ema", continued)
    if(bought):
        fl = Path("./continue/" + symbol + "_ema")
        fl.touch(exist_ok=True)

    print("-----------------\nTrejdanje završeno")
    print("Profit: ", profit)
```

4.6.2. Funkcija macd

Logika funkcije *macd* gotovo je identična funkciji *ema*, izuzev toga što ne koristi varijable sa predikatom *ema* već umjesto toga varijablu *histogram*, ima drugačiji uvjet kupnje i ne postoje pomoćne varijable prošle iteracije. Sljedeće slike prikazuju kod korišten u funkciji *ema* te njihov ekvivalent u funkciji *macd*.

```
#ema
ema8 = ema21 = prev_ema8 = prev_ema21 = None

#macd
histogram = None
```

Slika 4.5: Razlike kod inicijalizacije varijabli

```
#ema
df['ema8'] = bta.ema(df, period=8).df
df['ema21'] = bta.ema(df, period=21).df
ema8=df.ema8[-1]
ema21=df.ema21[-1]

#macd
macd = bta.macd(df, pfast=12, pslow=26, psignal=9)
df = df.join([macd.df])
histogram = df['histogram'][-1]
```

Slika 4.6: Razlike kod postavljanja vrijednosti

```
#ema
if(ema8 > ema21 and prev_ema8 and not bought):
    if(prev_ema8 < prev_ema21):

#macd
if(histogram > 0 and not bought):
```

Slika 4.7: Razlike kod uvjeta kupnje

4.7. Klase

4.7.1. Order

Instanca klase *Order* (programski kod 4.13) koristi se za spremanje parametara korištenih kod kupovanja i prodavanja kriptovaluta. Inicijalizacija klase definira postavljanje svih podatkovnih članova dostupnih kod kupnje kriptovalute te postavljanje članova *sellPrice* i *sellTime* u vrijednost praznog teksta. Prazni tekstovi se pomoću metode *sell*, prilikom prodaje postave na realne vrijednosti. Također je definirana *__str__* metoda koja se poziva kada je od objekta tražena povratna vrijednost tipa tekst.

Programski kod 4.13: Klasa Order

```
class Order:

    def __init__(self, quantity, buyPrice, buyTime):
        self.quantity = quantity
        self.buyPrice = buyPrice
        self.sellPrice = ""
        self.buyTime = buyTime
        self.sellTime = ""

    def sell(self, sellPrice, sellTime):
        self.sellPrice = sellPrice
        self.sellTime = sellTime

    def __str__(self):
        return (
            '''
_____________________________________
quantity:   ''' + str(self.quantity) + '''
buyPrice:   ''' + str(self.buyPrice) + '''
sellPrice:  ''' + str(self.sellPrice) + '''
buyTime:    ''' + str(self.buyTime) + '''
sellTime:   ''' + str(self.sellTime)
        )
```

4.7.2. Session

Instanca klase *Session* (programski kod 4.14) korištena je za spremanje svih parametara koji ulaze u sesiju, uz informacije o vremenu njezinog početka i kraja. Također ima definiranu *__str__* metodu koja nakon formatiranog ispisa svih podatkovnih članova, za svaki član tipa *Order* u polju *orders* poziva njegovu *__str__* metodu, te njezin izlaz nadodaje na izlaznu vrijednost metode. Za spremanje objekta *Session* radi kasnijeg korištenja koristi se funkcija *save*. Funkcija pomoću, prije u sesiji definiranog, simbola kriptovalute i proslijeđenog naziva algoritma provjerava postoji li u poddirektoriju „statistics“ datoteka s navedenim nazivom te ako je nema kreira se prazna datoteka. Sadržaj te datoteke se pomoću pickleove funkcije *load* deserijalizira i spremi u polje *sessions*, a ukoliko datoteka nema sadržaja polje *sessions* se inicijalizira kao prazno polje. Ako je proslijeđena varijabla *continued* jednaka *true* onda se iz polja *session* izbacuje posljedni element prije dodavanja trenutačne sesije, kako u slučaju nastavljanja ne bi imali dupli zapis. Za kraj pokazivač se premješta na početak datoteke te se pomoću pickleove funkcije *dump* stari sadržaj prebriše s novim. Funkcija *getParameters* vraća polje parametara sesije, vrijeme početka sesije i polje s objektima *Order*.

Programski kod 4.14: Klasa Session

```
class Session:

    def __init__(self, symbol, money, takeMoney, leaveMoney, amount,
                 takeProfit, stopLoss, profit, orders, start, end):
        self.symbol = symbol
        self.money = money
        self.takeMoney = takeMoney
        self.leaveMoney = leaveMoney
        self.amount = amount
        self.takeProfit = takeProfit
        self.stopLoss = stopLoss
        self.profit = profit
        self.orders = orders
        self.start = start
        self.end = end

    def __str__(self):
        orderStr = ""
        for order in self.orders:
            orderStr += str(order)

        return (
            '''
start:      ''' + str(self.start) + '''
end:        ''' + str(self.end) + '''
symbol:     ''' + str(self.symbol) + '''
money:      ''' + str(self.money) + '''
takeMoney:  ''' + str(self.takeMoney) + '''
leaveMoney: ''' + str(self.leaveMoney) + '''
amount:     ''' + str(self.amount) + '''
takeProfit: ''' + str(self.takeProfit) + '''
stopLoss :  ''' + str(self.stopLoss) + '''
profit:     ''' + str(self.profit) + '''

orders:''' + orderStr)

    def save(self, algorithm, continued):
        name = "./statistics/" + self.symbol + "_" + algorithm
+ '.bin'
        fl = Path(name)
        fl.touch(exist_ok=True)
        #write prebriše prije čitanja
        with open(name ,'rb+') as file:
            try:
                sessions = pickle.load(file)
            except EOFError:
                sessions = []

            if continued:
                sessions.pop()
            sessions.append(self)

            file.seek(0)
            pickle.dump(sessions, file)
```

```
    def getParameters(self):
        return [self.money, self.takeMoney, self.leaveMoney,
self.amount, self.takeProfit, self.stopLoss], self.start, self.orders
```

4.7.3. Statistics

Klasa *Statistics* (programski kod 4.15) kod inicijalizacije postavlja primljeni naziv algoritma i simbol kriptovalute te kreira prazno polje *sessions*. Funkcija *load* popunjava prije navedeno polje *sessions* sa objektima tipa *Session* iz deserijaliziranih datoteka sa sesijama. Funkcija *continueSession* poziva navedenu funkciju *load* za dohvat podataka te na zadnjem elementu iz polja *sessions* poziva prije spomenutu funkciju *getParameters* i vraća njezin rezultat.

Programski kod 4.15: Klasa Statistics

```
class Statistics:

    def __init__(self, algorithm, symbol):
        self.sessions = []
        self.algorithm = algorithm
        self.symbol = symbol

    def load(self):
        with open("./statistics/" + self.symbol + "_" +
self.algorithm + ".bin","rb") as file:
            self.sessions = pickle.load(file)

    def continueSession(self):
        self.load()
        return self.sessions[-1].getParameters()
```

4.8. Prikaz statistika

Prema programskom kodu 4.16 opcija prikaz statistika od korisnika traži da odabere algoritam razmjene za koji želi prikazati statistike. Prema unosu varijabla *algorithm* postavlja se na tekstualnu vrijednost naziva odabranog algoritma. Nakon to slijedi odabir simbola kriptovalute za prikaz statistika, koji radi na isti način kao prije spomenuta funkcija *symbolInput*. U *try* bloku poziva se funkcija *showStatistics* za prikaz statistike te ukoliko ne postoji datoteka s kombinacijom unesenog algoritma i simbola, *catch* blok javlja poruku o

grešci. Nakon prikaza statistika čeka se korisnikov unos kako bi se nastavilo s radom programa.

Programski kod 4.16: Funkcija statisticOptions

```
def statisticOptions(client):
    clr()
    while True:
        print(
        '''Odaberite algoritam za prikaz statistika: 0)Povratak
                                                1)MACD
                                                2)EMA'''
        )
        algorithm = input()
        clr()
        if algorithm == "0":
            return
        elif algorithm == "1":
            algorithm = "macd"
            break
        elif algorithm == "2":
            algorithm = "ema"
            break
        else:
            print("Pogrešan unos")

    balance = client.get_account()
    balance = [x for x in balance["balances"] if x["asset"] not in
stables]
    clr()

    while True:
        print("Odaberite željenu kriptovalutu: 0)Povratak")
        for i, token in enumerate(balance, start=1):
            print("                                    "+str(i) + ")" +
token['asset'])
        symbol = input()
        clr()
        if symbol == "0":
            return

        try:
            symbol = balance[int(symbol)-1]["asset"] + "USDT"
            break
        except:
            print("Pogrešan unos")

    try:
        showStatistics(algorithm, symbol)
    except:
        print("Nema dostupnih statistika")

    input("\nPritisnite \"Enter\" za izlaz:")
    clr()
```

Funkcija *showStatistics* (programski kod 4.17) pomoću proslijeđenih podataka o algoritmu i simbolu kriptovalute kreira objekt tipa *Statistics*. Nad objektom se pozove prije spomenuta funkcija *load* kako bi se njegovo polje *sessions* napunilo s vrijednostima te se u *for* petlji pomoću prije opisanih __*str*__ metoda ispisuju podaci sa svaku sesiju i transakciju.

Programski kod 4.17: Funkcija showStatistics

```
def showStatistics(algorithm, symbol):
    statistics = classes.Statistics(algorithm, symbol)
    statistics.load()
    for session in statistics.sessions:
        print(str(session))
```

4.8.1. Primjeri statistika

Nažalost zbog sporoga i nekvalitetnoga rada razvojne mreže nije zabilježena značajna količina podataka za analizu. Prilikom testiranja zaključeno je kako se cijene na razvojnoj mreži ponašaju neprirodno, tj. rijetko kada se dešavaju značajni padovi ili porasti. Korišteni tehnički indikatori računaju se prema prosječnim vrijednostima pojedinih kriptovaluta te su potrebni veći uzastopni skokovi u cijeni kako bi se pojavio signal za kupnju. Kada se pojavi signal za kupnju u većini slučajeva već je prekasno i cijena počinje padati te zbog toga većina prodaja završi s negativnim profitom. Također pošto uvjet za prodaju ovisi o postotku od cijene kriptovalute kod skupljih kriptovaluta (npr. Bitcoin), velike su razlike između trenutačne i prodajnih cijena te bi trebao proći jako dugi period između kupnje i prodaje.

Pregledom i analizom zabilježenih statistika smatra se da program ne radi optimalno te da bi bila potrebna dorada algoritama i testiranje na glavnoj mreži sa simulacijom kupnje. Na sljedećim slikama prikazani su primjeri prikaza statistika.

```
start:      28.09.2022. 22:29:30
end:        09.10.2022. 19:23:16
symbol:     LTCUSDT
money:      200.0
takeMoney:  1250.0
leaveMoney: 20.0
amount:     200.0
takeProfit: 0.04
stopLoss :  0.02
profit:     0

orders:
_____________________________________________
quantity:   3.72995
buyPrice:   53.62
sellPrice:
buyTime:    28.09.2022. 22:32:02
sellTime:

Pritisnite "Enter" za izlaz:
```

Slika 4.8: Primjer statistike sesije koja se može nastaviti

```
start:      28.09.2022. 13:05:51
end:        28.09.2022. 20:16:35
symbol:     BNBUSDT
money:      200.0
takeMoney:  250.0
leaveMoney: 150.0
amount:     40.0
takeProfit: 0.02
stopLoss :  0.03
profit:     0.825

orders:
_____________________________________________
quantity:   0.15
buyPrice:   272.4
sellPrice:  277.9
buyTime:    28.09.2022. 13:19:03
sellTime:   28.09.2022. 17:51:03
_____________________________________________
quantity:   0.14
buyPrice:   277.9
sellPrice:
buyTime:    28.09.2022. 17:52:03
sellTime:

Pritisnite "Enter" za izlaz:
```

Slika 4.9: Primjer statistike sa profitom

```
start:      08.10.2022. 17:02:56
end:        09.10.2022. 02:06:00
symbol:     TRXUSDT
money:      250.0
takeMoney:  300.0
leaveMoney: 200.0
amount:     50.0
takeProfit: 0.02
stopLoss :  0.02
profit:     -1.427916000000003

orders:

_____________________________________________
quantity:   802.2
buyPrice:   0.06233
sellPrice:  0.06055
buyTime:    08.10.2022. 17:03:00
sellTime:   08.10.2022. 21:01:03
_____________________________________________
quantity:   803.7
buyPrice:   0.06221
sellPrice:
buyTime:    08.10.2022. 21:05:02
sellTime:

Pritisnite "Enter" za izlaz:
```

Slika 4.10: Primjer statistike sa gubitkom

5. ZAKLJUČAK

Postojanjem velikog broja raznih kriptovaluta i svakodnevnim naglim promjenama na tržištu, za čovjeka postaje nemoguće cijelo vrijeme biti ažuran sa svima informacijama. Ovim radom cilja se na olakšavanje procesa prikupljanja općenitih informacija o kriptovalutama na burzi te uz korisnikove unesene parametre automatska razmjena kriptovalute bez daljnje korisnikove interakcije. Izvršene transakcije se uz unesene parametre spremaju u datoteke s ciljem kasnije analize, kako bi se mogao optimizirati proces automatske razmjene. Nažalost zbog naglih promjena na tržištu i svih vanjskih faktora koji utječu na tržište, teško je samo uz statističku analizu generirati profit. Uzevši u obzir sve okolnosti, autor ovog rada smatra da bi najbolje bilo kreirati program koji prati sve kriptovalute na tržištu i korisnika obavještava o mogućim promjenama na tržištu. O primljenim informacijama, saznanjima i vanjskim faktorima tržišta, a korisnik bi trebao potvrditi hoće ili neće kupiti predložene kriptovalute. U tom slučaju bot bi se koristio za dohvat i analizu podataka, bilježenje statistike te korištenje predefiniranih strategija za kupnju i prodaju kriptovaluta uz povremenu intervenciju korisnika prilikom odlučivanja o preddefiniranim osjetljivim transakcijama. Bot predstavljen u ovom radu je funkcionalan, ali postoji prostor za implementaciju dodatnih značajki: dohvat veće količine povijesnih podataka, kreiranje algoritma za prodaju bez parametara, testiranje drugih algoritma za kupnju/prodaju te prema statistikama implementiranje daljnjih poboljšanja. Zbog prosječno negativnog profita smatra se da ovaj bot nije ispunio očekivani rezultat.

6. LITERATURA

Primjeri:

[1] Što je kriptovaluta i kako funkcionira? [Online]. 2022. Dostupno na: https://kriptomat.io/hr/kriptovalute/sto-je-kriptovaluta/ (20.10.2022)

[2] Python 3.9.14 documentation [Online]. 2022. Dostupno na: https://docs.python.org/3.9/ (15.9.2022)

[3] Jignesh Davda. Binance Python API – A Step-by-Step Guide [Online]. 2021. Dostupno na: https://algotrading101.com/learn/binance-python-api-guide/ (2.9.2022)

[4] Fernet (symmetric encryption) [Online]. 2022. Dostupno na: https://cryptography.io/en/latest/fernet/ (19.8.2022)

[5] Pandas documentation [Online]. 2022. Dostupno na: https://pandas.pydata.org/docs/ (2.9.2022)

[6] Documentation for Visual Studio Code [Online]. 2022. Dostupno na: https://code.visualstudio.com/docs (2.7.2022)

Popis programskih kodova

Programski kod 4.1: Datoteka main.py 7
Programski kod 4.2: Funkcija clr 7
Programski kod 4.3: Funkcija mainOptions 8
Programski kod 4.4: Funkcija login 9
Programski kod 4.5: Funkcija getBalance 10
Programski kod 4.6: Funkcija getPrices 11
Programski kod 4.7: Funkcija algorithOptions 11
Programski kod 4.8: Funkcija algorithParameters 13
Programski kod 4.9: Funkcija symbolInput 13
Programski kod 4.10: Funkcija moneyInput 14
Programski kod 4.11: Funkcija getKlines 15
Programski kod 4.12: Funkcija ema 18
Programski kod 4.13: Klasa Order 22
Programski kod 4.14: Klasa Session 23
Programski kod 4.15: Klasa Statistics 24
Programski kod 4.16: Funkcija statisticOptions 25
Programski kod 4.17: Funkcija showStatistics 26

Popis slika

Slika 4.1: Prikaz direktorija Code u programu VS Code 6
Slika 4.2: Pokretanje programa pomoću naredbenog retka 6
Slika 4.3: Glavni izbornik 8
Slika 4.4: Primjer rada algoritma 16
Slika 4.5: Razlike kod inicijalizacije varijabli 21
Slika 4.6: Razlike kod postavljanja vrijednosti 21
Slika 4.7: Razlike kod uvjeta kupnje 21
Slika 4.8: Primjer statistike sesije koja se može nastaviti 27
Slika 4.9: Primjer statistike sa profitom 27
Slika 4.10: Primjer statistike sa gubitkom 28

7. OZNAKE I KRATICE

PoW - Proof of Work

PoS - Proof of Stake

PoH - Proof of History

API - Application programming interface

JSON - JavaScript Object Notation

EMA - Exponential Moving Average

MACD - Moving Average Convergence Divergence

8. SAŽETAK

Naslov: Izrada programske aplikacije za interakciju i trgovanje na kripto burzi

Pomoću uređivača koda Visual Studio Code i programskoga jezika Python, izrađen je program za interakciju s Binance kripto burzom, kroz interaktivno tekstualno sučelje. Podaci za prijavu se iz sigurnosnih razloga, pomoću kriptografskih algoritama zapisuju u šifrirane datoteke. Korištenjem programa korisnik može zatražiti za trgovanje bitne informacije o računu i kripto tržištu. Pomoću dobivenih podataka i tehničkih indikatora za trgovanje kupuju se definirane kriptovalute te se ih pomoću korisnikovih unesenih parametara prodaje. Program podržava nastavljanje prisilno završenih sesija, bez ponovnog unosa parametara. Informacije o transakcijama i sesijama zapisuju se pomoću serijalizacije u datoteke, kako bi se kasnije koristile kod analize podataka.

Ključne riječi: Kriptovalute, Python, kriptografija, algoritmi, statistika

9. ABSTRACT

Title: Development of a software application for interaction and trading on a crypto exchange

Using the code editor Visual Studio Code and the Python programming language, the program is created for interaction with the Binance crypto exchange, through an interactive text interface. For security reasons, login data is saved in encrypted files using cryptographic algorithms. Using the program, the user can request information about trading essential account information and the crypto market. Using the obtained data and technical indicators for trading, we buy defined cryptocurrencies and sell them using the parameters entered by the user. The program supports continuing forcibly finished sessions, without re-entering the parameters. Transaction and session information is written using serialization to files for later use in data analysis.

Keywords: Cryptocurrencies, Python, cryptography, algorithms, statistics

IZJAVA O AUTORSTVU ZAVRŠNOG RADA

Pod punom odgovornošću izjavljujem da sam ovaj rad izradio/la samostalno, poštujući **načela akademske čestitosti, pravila struke te pravila i norme standardnog hrvatskog jezika. Rad je moje autorsko djelo i svi su preuzeti citati i parafraze u njemu primjereno označeni.**

Mjesto i datum	Ime i prezime studenta/ice	Potpis studenta/ice
U Bjelovaru, 20.10.2022.	LEON NOVAK	Leon Novak

Prema Odluci Veleučilišta u Bjelovaru, a u skladu sa Zakonom o znanstvenoj djelatnosti i visokom obrazovanju, elektroničke inačice završnih radova studenata Veleučilišta u Bjelovaru bit će pohranjene i javno dostupne u internetskoj bazi Nacionalne i sveučilišne knjižnice u Zagrebu. Ukoliko ste suglasni da tekst Vašeg završnog rada u cijelosti bude javno objavljen, molimo Vas da to potvrdite potpisom.

Suglasnost za objavljivanje elektroničke inačice završnog rada u javno dostupnom **nacionalnom repozitoriju**

LEON NOVAK

ime i prezime studenta/ice

Dajem suglasnost da se radi promicanja otvorenog i slobodnog pristupa znanju i informacijama cjeloviti tekst mojeg završnog rada pohrani u repozitorij Nacionalne i sveučilišne knjižnice u Zagrebu i time učini javno dostupnim.

Svojim potpisom potvrđujem istovjetnost tiskane i elektroničke inačice završnog rada.

U Bjelovaru, 20.10.2022.

Leon Novak

potpis studenta/ice