Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Andrej Burja

Aplikacija za trgovanje na kriptoborzah

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: viš. pred. dr. Robert Rozman

Ljubljana, 2019

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Trgovanje na t.i. kriptoborzah je dokaj zapleteno in nepregledno opravilo. Pojavlja se vse več novih kriptoborz, ki vsaka na svoj način uporabniku grafično prikazujejo svoje storitve. Izdelajte aplikacijo za trgovanje na več kriptoborzah, ki bo imela enoten grafični prikaz. Aplikacija naj omogoča osnovne operacije, kot so trgovanje, pregled stanja uporabnika, prenos kriptovalut in pregled zgodovine uporabnika. Aplikaciji dodajte še druge pomembne informacije, ki izboljšajo uporabniško izkušnjo. Zasnujte jo modularno in dodajte nekaj izbranih kriptoborz ter na njih preizkusite delovanje. Zagotovite tudi nadaljnje širitve z možnostjo enostavnega dodajanja novih kriptoborz.

Kazalo

Povzetek

Abstract

1 Uvod **1**

2 Kriptovalute **5**

2.1 Tehnologija verige blokov 6

2.2 Veriga blokov Ethereum 7

2.3 Pametne pogodbe 8

2.4 Standard za kriptožetone ERC-20 8

3 Kriptoborze **11**

3.1 Vrste kriptoborz 11

3.1.1 Centralizirane borze 12

3.1.2 Decentralizirane borze 13

3.2 Trgovanje 14

3.2.1 Vrste naročil 16

3.3 Stanje uporabnika 17

3.4 Zgodovina uporabnika 18

3.5 Borzni programski vmesniki - API-ji 19

4 Kriptoborza ForkDelta **21**

4.1 Programski vmesnik ForkDelta 24

4.2 Izdelava nove ponudbe 26

4.3 Trgovanje 28
4.4 Preklic ponudbe 31
4.5 Dobroimetje uporabnika 32
4.6 Prenos kriptovalut 34
4.6.1 Prenos kriptovalut na pametno pogodbo EtherDelta . . 34
4.6.2 Prenos kriptovalut s pametne pogodbe EtherDelta . . . 35
4.6.3 Prenos kriptovalut z denarnice 36
4.7 Zgodovina transakcij 36

5 Kriptoborza KuCoin **39**
5.1 Programski vmesnik REST 39
5.1.1 Avtentikacija programskega vmesnika 42
5.2 Programski vmesnik WebSocket 43

6 Kriptoborza HitBTC **45**
6.1 Programski vmesnik REST 46
6.1.1 Avtentikacija programskega vmesnika 48
6.2 Programski vmesnik WebSocket 49

7 Aplikacija **53**
7.1 Struktura aplikacije 54
7.2 Trgovanje 56
7.2.1 Trgovalni pari 56
7.2.2 Knjiga naročil 59
7.2.3 Zgodovina 60
7.2.4 Zgodovina trgovanja uporabnika 61
7.2.5 Aktivna naročila uporabnika 61
7.2.6 Grafični prikaz zgodovine trgovanja 62
7.2.7 Obrazec za izdelavo novega naročila 64
7.3 Stanje kriptovalut in upravljanje z njimi 65
7.3.1 Prenos kriptovalut med denarnico in trgovalnim računom 68
7.3.2 Polog kriptovalut 68

7.3.3 Dvig kriptovalut 69
7.4 Zgodovina uporabnika 71
7.5 Beleženje delovanja aplikacije 73
7.5.1 Knjižnica log4net 73
7.5.2 Beleženje zahtev in odgovorov strežnika 74
7.6 Nastavitve aplikacije 76

8 Zaključek **77**

Literatura **80**

Seznam uporabljenih kratic

kratica	angleško	slovensko
API	application programming interface	aplikacijski programski vmesnik
JSON	JavaScript object notation	notacija za označevanje JavaScript objektov
P2P	peer to peer	omrežje vsak z vsakim
PoW	Proof of work	dokaz o delu
2FA	two-factor authentication	dvostopenjska avtentifikacija
WPF	windows presentation foundation	sistem za izdelavo grafičnih vmesnikov
DNS	domain name system	sistem domenskih imen
HTTP	hypertext transfer protocol	protokol za prenos nadbesedila
URL	uniform resource locator	enolični krajevnik vira
UTC	coordinated universal time	univerzalni koordinirani čas
HMAC	hash-based message authentication code	koda za preverjanje pristnosti sporočila z razpršilno funkcijo
SHA256	secure hash algorithm - 256 bit	kriptografska zgoščevalna funkcija SHA z rezultatom velikim 256 bitov
TxId	transaction hash ID	enolični identifikator transakcije
ABI	application binary interface	binarni vmesnik aplikacije
REST	representational state transfer	reprezentativni prenos stanja
XML	extensible markup language	razširljivi označevalni jezik
GUI	graphical user interface	grafični uporabniški vmesnik

Povzetek

Naslov: Aplikacija za trgovanje na kriptoborzah

Avtor: Andrej Burja

V okviru diplomske naloge je bila izdelana aplikacija z grafičnim uporabniškim vmesnikom za trgovanje na kriptoborzah. Ta aplikacija ima vse osnovne funkcionalnosti, ki nam jih ponujajo današnje trgovalne platforme. Te funkcionalnosti so: trgovanje, pregled zgodovine uporabnika, pregled stanja uporabnika in upravljanje z njim. Aplikacija je spisana v programskem jeziku C# v ogrodju .NET in trenutno podpira trgovanje na treh različnih kriptoborzah: ForkDelta, KuCoin in HitBTC. Te borze so v aplikacijo dodane kot knjižnice in so bile tudi izdelane v okviru diplomske naloge. Aplikacija poleg aplikacijskih programskih vmesnikov (API) borz uporablja tudi druge programske vmesnike: CoinMarketCap (za dostop do povprečnih cen kriptovalut na trgu), Infura.io (za povezavo z verigo blokov Ethereum, kar potrebujemo pri delu z borzo ForkDelta). Aplikacija je zasnovana tako, da se lahko v prihodnosti doda podpora še za druge kriptoborze.

Ključne besede: veriga blokov, kriptovalute, kriptoborza, ForkDelta, HitBTC, KuCoin, CoinMarketCap, WebSocket API, REST API, Socket.IO, GUI, Ethereum.

Abstract

Title: Application for trading in crypto exchanges

Author: Andrej Burja

In this thesis, an application with a graphical user interface for trading on crypto exchanges was created. This application has all the basic functionalities offered by today's trading platforms. These functionalities include: trading, reviewing user history, reviewing and managing user status. The application is written in the C# programming language using the .NET framework and currently supports trading on three different crypto exchanges: ForkDelta, KuCoin, and HitBTC. These stock exchanges have been added to the application as libraries and were also implemented in the framework of the thesis. In addition to the API of crypto exchanges, the application also uses other software interfaces: CoinMarketCap (for access to the average crypto prices on the market), Infura.io (for connection to the Ethereum blockchain, which we need to interface with the ForkDelta stock exchange). The application is written in a modular approach that allows the addition of the support for other crypto exchanges in the future.

Keywords: blockchain, cryptocurrency, crypto exchanges, ForkDelta, HitBTC, KuCoin, CoinMarketCap, WebSocket API, REST API, Socket.IO, GUI, Ethereum.

Poglavje 1

Uvod

Na svetu se pojavlja vse več novih kriptovalut. Te kriptovalute nimajo vrednosti, če z njimi ne moremo trgovati, jih porabiti ali jih zamenjati za nekaj drugega. Kriptoborze (angl. *crypto exchanges*) nam omogočajo, da lahko z njimi trgujemo. Z rastjo novih kriptovalut prihajajo tudi nove kriptoborze. Kriptoborze nam omogočajo trgovanje neke kriptovalute z drugo kriptovaluto. Primer: trgovalni par `ETH/BTC`. Skoraj vse kriptoborze omogočajo, da z njimi komuniciramo preko aplikacijskega programskega vmesnika (API). To daje razvijalcem možnost, da razvijejo različne programe, oziroma vmesnike, s katerimi lahko uporabniki trgujejo. Ti programi so ponavadi aplikacije za trgovanje, programi za analizo cen trgovanja, programi za avtomatsko trgovanje (arbitraža, trgovalne strategije itn.) itd.

Problem kriptoborz je, da vsaka na svoj način prikazuje uporabnikom svoje storitve. S to aplikacijo smo želeli poenotiti prikaz in trgovanje na kriptoborzah. Diplomska naloga opisuje razvoj in delovanje aplikacije namenjene osnovnim operacijam, ki nam jih dandanes ponujajo vse normalne kriptoborze. To so: trgovanje, prikaz uporabnikove zgodovine in prikaz stanja ter upravljanje z njim. Aplikacija trenutno podpira tri kriptoborze, ima približno 9000 vrstic spisane kode (v številko niso vključene avtomatsko generirane kode, npr. Windows Forms, DataSet itd.).

Med podobnimi obstoječimi aplikacijami je naši rešitvi najbolj soroden

program Silex [58], vendar ta ne podpira kriptoborz HitBTC in ForkDelta, katere dve sami največ uporabljamo. Zato smo izdelali svojo aplikacijo, ki jo lahko po želji dopolnjujemo z novimi kriptoborzami. Več o podobnih obstoječih aplikacijah je napisano v 7. poglavju (*Aplikacija*).

V 2. poglavju (*Kriptovalute*) bomo na kratko opisali, kaj so kriptovalute, kako delujejo in na kaj jih delimo. Opisali bomo tehnologijo verige blokov, ki se uporablja pri večini kriptovalut. Razložili bomo tudi, kaj je platforma Ethereum, kako delujejo pametne pogodbe na njej in kaj so kriptožetoni (standard ERC-20). Vse to bomo opisali, da bo v naslednjih poglavjih bolj razumljivo, kako deluje decentralizirana borza ForkDelta.

V 3. poglavju (*Kriptoborze*) bomo na kratko opisali vrste kriptoborz, ki so trenutno na voljo uporabnikom. Bolj podrobno bomo opisali tiste, ki smo jih tudi uporabili v aplikaciji. Tukaj bomo še na kratko opisali osnovne storitve, ki nam jih ponujajo spletne borze, to so: trgovanje, pregled uporabnikove zgodovine in upravljanje s kriptovalutami uporabnika. Na koncu poglavja bomo opisali še uporabo borznih programskih vmesnikov (API-jev), ki nam jih borze ponujajo za komunikacijo z njimi.

V 4. poglavju (*Kriptoborza ForkDelta*) bomo opisali, na kakšen način komunicira naša knjižnica ForkDelta_API z borzo, kakšne so slabosti ForkDelte in na kakšen način pridobiti bolj točno zgodovino. Opisali bomo, kako poslati novo naročilo v knjigo naročil, kako pridobimo vso zgodovino uporabnika ter kako dostopamo do stanja kriptovalut, ki jih ima uporabnik na pametni pogodbi borze. V nadaljevanju bomo predstavili, kako izvesti trgovanje preko pametne pogodbe in kako moramo poslati transakcijo v omrežje Ethereum. Na koncu poglavja bomo omenili še, kako prenesemo kriptovalute na in s pametne pogodbe EtherDelta.

V 5. poglavju (*Kriptoborza KuCoin*) bomo našteli vse KuCoin funkcije API-ja, ki jih uporablja naša aplikacija za delovanje. Opisana bosta oba API-ja, ki nam ju borza ponuja.

V 6. poglavju (*Kriptoborza HitBTC*) bomo na isti način kot pri prejšnjem

poglavju opisali delovanje API-ja za kriptoborzo HitBTC.

V 7. poglavju (*Aplikacija*) bomo opisali strukturo aplikacije in vsa okna, ki jih aplikacija uporablja za delovanje. Pri vsakem oknu bo na kratko opisano, kaj nam ta prikazuje, ter poleg tudi posebnosti pri nekaterih borzah. Opisali bomo tudi, kako beležimo delovanje aplikacije ter vse knjižnice, ki smo jih uporabili v aplikaciji.

Poglavje 2

Kriptovalute

Kriptovalute so digitalne valute (elektronski denar), ki za delovanje uporabljajo kriptografijo za zaščito transakcij. Razvoj kriptovalut se je začel, ko je Satoshi Nakamoto leta 2009 objavil programsko kodo za kriptovaluto Bitcoin. Do sedaj se je ustvarilo že več kot 5000 različnih kriptovalut, skoraj vse te kriptovalute so nastale kot alternative (izboljšana verzija) digitalni valuti Bitcoin, zaradi česar jih kličemo tudi angl. *altcoins*. Sam namen kriptovalut je, da so decentralizirane (nimajo glavnega člena ali nekoga, ki bi jih lahko ugasnil). Za pošiljanje in prejemanje kriptovalut ne potrebujemo posrednika (*no middle man*). Pri omrežju Bitcoin se v ta namen uporablja peer-to-peer (P2P) omrežje. Skoraj vse kriptovalute so odprtokodne, kar običajno daje uporabnikom več zaupanja [70].

Kriptovalute delimo na kriptokovance in kriptožetone. Kriptokovanci so vgrajeni v verigo blokov, ki jo uporabljajo, in se ponavadi uporabljajo za plačevanje transakcij in storitev na verigi blokov. Kriptožetoni so narejeni na že delujoči verigi blokov, ponavadi preko pametne pogodbe [61].
Primeri kriptokovancev: Bitcoin, Ethereum, Neo, Waves, Tron itn.
Primeri kriptožetonov: OmiseGO, Basic Attention Token, BitTorrent Token itn.

Skoraj vse kriptovalute za delovanje uporabljajo verigo blokov. Nekatere izjeme uporabljajo tudi usmerjene neciklične grafe („Directed Acyclic

Graph“, krajše DAG) ali pa kombinacijo obeh. Nekatere te izjeme so: IOTA (Tangle network), Nano (Block-lattice), Fantom itn [62, 33].

2.1 Tehnologija verige blokov

Verige blokov sta prvič omenila Stuart Haber in W. Scott Stornetta leta 1991, ko sta hotela implementirati sistem za časovno žigosanje datotek, katerih žigosanj uporabnik ne more spremeniti [44]. Prvič pa je veriga blokov bila uporabljena 2009, ko je Satoshi Nakamoto zagnal omrežje Bitcoin [69]. Tukaj bomo na kratko opisali delovanje verige blokov, ki jo uporablja Bitcoin.

Veriga blokov (angl. *blockchain*), so med seboj povezani bloki, v katerih je seznam zapisov (transakcije). Vsak blok vsebuje podatke, zgoščeno vrednost (angl. *hash*) bloka in zgoščeno vrednost bloka pred njim. Zgoščena vrednost je podpis podatkov („prstni odtis“), ki nastane, ko zgoščevalna funkcija podatke poljubne velikosti preslika v besedilo konstantne dolžine. V verigi blokov so vse zgoščene vrednosti blokov unikatne, izračunajo se, ko se blok ustvari. Če v bloku podatke spremenimo, se spremeni tudi zgoščena vrednost bloka. Kar daje verigi blokov varnost, je to, da vsi (z izjemo prvega (*Genesis block*)) bloki vsebujejo zgoščeno vrednost prejšnjega bloka, s čimer dobimo verigo med seboj povezanih blokov.

Podatki v bloku so lahko različni, odvisni pa so tipa verige blokov. Če vzamemo npr. Bitcoin, potem ti podatki predstavljajo transakcije, ki povedo s katerega naslova je znesek prišel (from), na kateri naslov je namenjen (to) in količino v Bitcoinih (amount). V primeru, da nekdo v nekem bloku spremeni podatke, se zgoščena vrednost bloka spremeni, kar pomeni, da so vsi bloki po tem bloku neveljavni, saj se z zgoščeno vrednostjo ne povezujejo več med seboj. Če na novo izračunamo vse zgoščene vrednosti po tem bloku, je veriga spet veljavna. Da ne pride do takšne zlorabe, nekatere verige blokov uporabljajo t.i. dokaz o delu (*proof-of-work*); med njimi je tudi Bitcoin.

Dokaz o delu je mehanizem, ki oteži (upočasni) izdelavo novega bloka v verigi blokov. To se naredi tako, da se oteži izračun zgoščene vrednosti bloka.

Ko je neka transakcija zapisana v verigo blokov, jo je zelo težko spremeniti. Dokaz o delu prinese s seboj tudi rudarje (angl. *miners*).

Rudarji tekmujejo med seboj in poskušajo izračunati zgoščeno vrednost bloka z določeno težavnostno stopnjo (angl. *difficulty*). Ko rudarju uspe izračunati zgoščeno vrednost bloka, je nagrajen v obliki kriptokovancev, prav tako pa dobi plačilo (angl. *fee*) uporabnikov, ki plačajo, da se njihovo transakcijo vključi v blok. Več kot uporabnik plača za obdelavo transakcije, prej pride na vrsto. Veriga blokov postane še bolj varna, ko jo porazdelimo (*distributed ledger*). Bitcoin uporablja omrežje P2P in vsak ima možnost, da se mu pridruži in postane eno od vozlišč (*node*). Iz omrežja lahko pridobimo kopijo verige blokov.

Ko se ustvari nov blok, je ta posredovan vsem vozliščem v omrežju in vsako vozlišče preveri, če je blok veljaven, v primeru, da je, ga vsako vozlišče doda v svojo verigo blokov, v nasprotnem primeru, ga vozlišča zavrnejo.

Za dostop do svojih kriptovalut uporabnik potrebuje privatni ključ (*private key*). Iz privatnega ključa lahko enostavno po algoritmu pridemo do javnega ključa (*public key*) in naslova, kamor se lahko pošlje kriptovalute. Za pošiljanje kriptovalut moramo transakcijo podpisati s privatnim ključem in jo poslati v omrežje [69, 5].

2.2 Veriga blokov Ethereum

Ethereum je decentralizirana javna mreža verig podatkovnih blokov. Ponuja nam platformo, na kateri lahko uporabniki razvijejo svoje pametne pogodbe (decentralizirane aplikacije). Ustanovil jo je Vitalik Buterin v letu 2013. Sredstva za razvoj platforme Ethereum so prišla iz spletne množične prodaje (*online crowdsale*). Platforma je zaživela 30. julija 2015, in je odprtokodna. Ethereum uporablja kriptokovanec Ether (ETH), s katerim uporabniki plačujejo transakcije, ga uporabljajo kot plačilno sredstvo in tudi za storitve na omrežju. Lahko ga tudi zaslužijo z rudarjenjem [27].

Omrežje Ethereum (*main Ethereum network*) ima trenutno približno 8500

vozlišč (P2P), na katerih so popolne kopije verige blokov. Vozlišča so lahko nameščena na različnih operacijskih sistemih, lahko uporabljajo različno programsko opremo in so lahko napisana v različnih programskih jezikih. Najbolj razširjeno programsko orodje za vozlišče je Geth, obstajajo pa tudi druga programska orodja, kot so: Parity, Parity-Ethereum itd. [27, 26, 31, 30].

2.3 Pametne pogodbe

Tukaj bomo na kratko opisali delovanje pametnih pogodb Ethereum. Pametna pogodba je programska koda shranjena v verigi blokov. Ko je enkrat v verigi blokov, se je ne da izbrisati ali spremeniti. Lahko dela le tisto, za kar je sprogramirana. Seveda lahko pametno pogodbo na začetku sprogramiramo tako, da jo lahko kasneje uničimo (`selfdestruct` funkcija), ali pa spremenimo njeno delovanje s pomočjo različnih parametrov (npr. naslov knjižnice katere kliče pametna pogodba), ki jih pogodba uporablja. Ethereum za pisanje pametnih pogodb uporablja programski jezik Solidity. Ta koda se izvede v Ethereum Virtual Machine (EVM), ki je na vsakem vozlišču v omrežju Ethereum. V poglavju *4 Kriptoborza ForkDelta* bomo tudi opisali, kako komuniciramo s pametno pogodbo [31].

2.4 Standard za kriptožetone ERC-20

ERC-20 je standard za kriptožetone, ki so narejeni na verigi blokov Ethereum. Sam standard opisuje, katere funkcije in katere dogodke naj bi vsebovale pametne pogodbe kriptožetonov. S poenotenjem osnovnih funkcij je tako razvijalcem lažje ponuditi orodja, ki uporabljajo te kriptožetone. Da kriptožetoni izpolnjujejo standard ERC-20 morajo imeti implementirane vse te naštete funkcije: `totalSupply, balanceOf, transfer, transferFrom, approve in allowance`, ter dogodke: `Transfer` in `Approval`. Pametna pogodba EtherDelta je narejena za trgovanje z žetoni ERC-20, to pogodbo pa bomo bolj podrobno opisali v poglavju *4 Kriptoborza ForkDelta*. Nekatere

pametne pogodbe žetonov so le delno skladne s standardom ERC-20 in nimajo nekaterih funkcij, ki jih ERC-20 določa, eden od takih žetonov je npr. Golem Network Token (GNT) [18]. Obstajajo tudi drugi standardi za pametne pogodbe kriptožetonov, na primer ERC-721 [17]. Ti standardi imajo kakšno dodatno lastnost, a standard ERC-20 je do sedaj najbolj razširjen.

Poglavje 3

Kriptoborze

Kriptoborze pravimo jim tudi kripto menjalnice uporabnikom omogočajo trgovanje s kriptovalutami. Trenutno obstaja okoli 200 spletnih kriptoborz, med seboj pa se tudi razlikujejo. Obstaja več vrst kriptoborz, v naslednjih poglavjih pa bomo opisali predvsem trgovalne platforme, pod katere spadajo tudi vse borze, ki jih uporabljamo v aplikaciji. Osnovne funkcije kriptoborz so trgovanje, pregled zgodovine transakcij in upravljanje s stanjem kriptovalut uporabnika [10, 65, 15, 32].

3.1 Vrste kriptoborz

Kriptoborze lahko delimo z več vidikov, najpogosteje pa jih delimo glede na način trgovanja in stopnje centraliziranosti. Na spletu je več člankov, ki opisujejo različne vrste kriptoborz, med sabo pa jih tudi različno opisujejo in poimenujejo. Spodaj bomo opisali najbolj pogoste opise vrst kriptoborz.

Vrste kriptoborz po načinu trgovanja:

- Trgovalne platforme (*Trading Platforms*). Spletna stran, ki povezuje nakupovalce in prodajalce. Transakcije (nakupi in prodaje) so ponavadi plačljive. Običajno imajo te borze knjigo naročil. Delimo jih na centralizirane in decentralizirane. Sem sodi večina kriptoborz, npr. Binance, KuCoin, HitBTC, Bitstamp, ForkDelta idr.

- Direktno trgovanje (*Direct trading* ali *Peer to Peer Exchange*). Te borze omogočajo direktno trgovanje po principu oseba z osebo (Over the Counter (OTC)). Primeri takšnih borz: AirSwap.io, LocalBitcoins.com. Sem spada tudi ForkDelta, saj gre za trgovanje oseba z osebo, pametna pogodba le zadrži kriptovalute, dokler ponudba ni plačana. Prav tako lahko uporabnik trguje s katerokoli ponudbo, ki je v knjigi naročil.

- Posredniki (*Brokers*). Pri tej vrsti kriptoborz nakupno in prodajno ceno kriptovalute določi posrednik. Primeri takšnih kriptoborz: Coinbase itn.

Kam torej spada ForkDelta, EtherDelta? Lahko rečemo, da v obe skupini (trgovalne platforme in direktno trgovanje) [10, 65, 15, 32].

Z vidika stopnje centraliziranosti pa kriptoborze delimo še na centralizirane in decentralizirane.

3.1.1 Centralizirane borze

Teh borz je največ, narejene so s strani podjetij, ki jih tudi vodijo ter odgovarjajo v primeru izgub kriptovalut. So bolj popularne kot decentralizirane borze. Pri trgovanju so hitre, malo počasnejše pa pri nalaganju kriptovalut na borzo (*deposit*) in prenosu kriptovalut z borze (*withdraw*). Ko uporabnik prenese kriptovalute na neko borzo, se lastništvo kriptovalut prenese na borzo (le borza ima dostop do privatnih ključev denarnice, na katero so bile kriptovalute prenešene). Uporabnik pa je le oseba, ki ima pravico do upravljanja s kriptovalutami. Na takšne borze se je največkrat treba registrirati in prijaviti. Pri večjih vsotah kriptovalut morajo uporabniki iti skozi postopek potrjevanja identitete. Nekatere borze omogočajo bolj napredna trgovalna naročila. V primerjavi z decentraliziranimi borzami, so lažja tarča hekerskih napadov [11].

3.1.2 Decentralizirane borze

Decentralizirane borze nimajo glavnega člena za delovanje in običajno ne potrebujejo prijav in registracij ter ne vodijo postopka preverjanja identitete uporabnika. Uporabniku dajo popoln nadzor nad njegovim kriptovalutami. Vsa koda, ki jo decentralizirana borza uporablja, je odprtokodna. Te borze naj bi bile bolj odporne na hekerske napade, med uporabniki pa zaradi več vzrokov niso tako priljubljene: likvidnost je nizka, za trgovanje se običajno uporablja le osnovna naročila, večinoma so bolj kompleksne kot centralizirane borze, nekatere so zaradi uporabe verige blokov tudi počasnejše itd [11].

Decentralizirane borze se med sabo lahko zelo razlikujejo. Za uporabo potrebujemo privatni ključ, ki nam služi kot geslo, ali pa privatni ključ denarnice. Za lažjo uporabo omogočajo tudi kripto denarnice, ki se namestijo v spletni brskalnik. Ti programi delujejo kot kripto denarnica za podpisovanje transakcij (npr. MetaMask, Waves Keeper itd.), lahko pa se povežejo tudi na strojno kripto denarnico, kot je na primer Ladger Nano S [11].

Le malo je kriptoborz, ki bi bile v celoti decentralizirane; problem največkrat nastane pri gostovanju spletne strani, strežnikih API-jev, strežnika DNS, skrbnikih pametnih pogodb. Ljudje imajo zato o decentraliziranih borzah različna mnenja. Decentralizirane borze običajno uporabljajo pametne pogodbe na neki verigi blokov. Problem lahko nastane, ker nekatere od teh pogodb uporabljajo tudi skrbnike, ki skrbijo za nadgradnjo pametne pogodbe, in pa v primeru kakršnih koli težav s pametno pogodbo [11]. Kar nekaj kraj se je že zgodilo, ko so nepridipravi prišli do privatnih ključev skrbnikov pametne pogodbe. Nedavno se je to zgodilo kriptoborzi Bancor [4].

Obstajajo tudi verige blokov, ki so bile narejene z namenom, da se preko njih trguje. Ena od teh je BitShares [6].

Med decentralizirane borze sodijo: ForkDelta (EtherDelta), IDEX, Waves, Openledger, CryptoBridge, Radar Relay, OasisDex, Stellar Dex.

3.2 Trgovanje

Kriptoborze nam omogočajo trgovanje neke kriptovalute z drugo kriptovaluto (plačljiva menjava). Vedno potrebujemo par, znotraj katerega lahko trgujemo, tem parom rečemo valutni ali trgovalni par.
Primer trgovalnega para: `ETH/BTC` (ETH = Ethereum, BTC = Bitcoin).
Pri trgovalnih parih se uporabljajo kratice kriptovalute in ne cela imena. Borze imajo lahko za isto kriptovaluto različne kratice (simbole). Prvi valuti v valutnem paru pravimo osnovna valuta, drugi pa kotirana valuta.

Prikaz za trgovanje ponavadi prikazuje ta polja:

Knjiga naročil. Knjiga naročil (angl. *orderbook*) nam prikazuje vsa aktivna naročila. Knjiga naročil se stalno osvežuje, vsa naročila so urejena po ceni, z barvo ločena na zelene nakupne (*bid*) in prodajna rdeča (*ask*) naročila [64]. Ob ceni je tudi podatek o količini in skupni vrednosti naročila (cena pomnožena s količino).

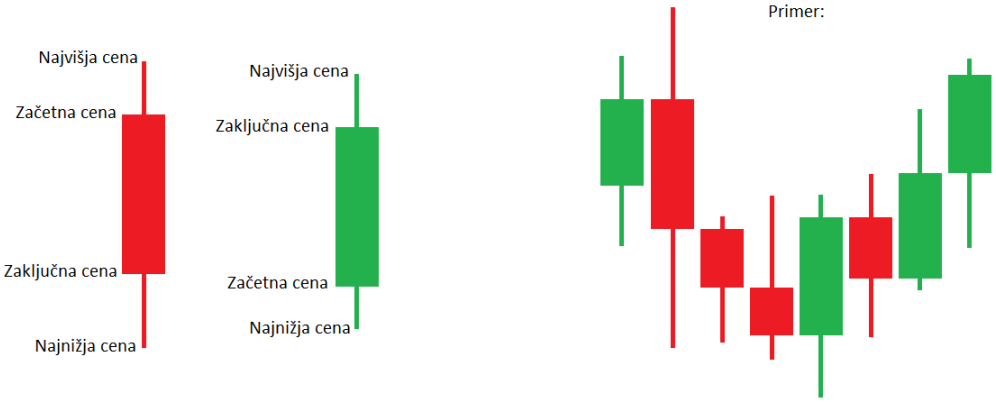
Aktivna naročila uporabnika. Vsa aktivna naročila, ki jih ima uporabnik odprta, z vsemi izpolnjenimi podatki. Poleg naročila imamo gumb za preklic naročila, ki je običajno brezplačen, kar pa ne velja pri trgovanju preko pametne pogodbe EtherDelta.

Zgodovina trgovanja valutnega para. Krajši seznam, v katerem so zapisane nedavne transakcije s podatki: datum in čas transakcije, z barvo je predstavljeno, ali je šlo za nakup (zelena) ali prodajo (rdeča), količina kriptovalute, cena ene kriptovalute v kotirani valuti in skupna vrednost (količina pomnožena s ceno) naročila.

Zgodovina uporabnika pri trgovanju z valutnim parom. Uporablja iste vrste podatkov kot zgodovina trgovanja, poleg pa je tudi strošek naročila (angl. *fee*).

Stanje uporabnika. Stanje kriptovalut, ki so v lasti uporabnika, za oba kriptovalutna para. Prikazuje razpoložljive kriptovalute in kriptovalute, ki so trenutno v uporabi (vsota odprtih pozicij).

Graf. Graf prikazuje vrednost osnovne valute valutnega para na osi Y, skozi čas na osi X. Skoraj vsaka borza omogoča, da si graf pogledamo po različnih časovnih obdobjih npr. eno časovno obdobje v grafu lahko predstavlja 1 minuto, 5 minut ali 15 minut, 30 minut, 1 uro, 4 ure ali en dan. Obdobja so predstavljena z japonskimi svečniki. Pod tem grafom je ponavadi še en graf, ki se natančno ujema z osjo X zgornjega grafa (časovni podatki na osi X so si enaki). Gre za stolpični graf, ki prikazuje vsoto količin trgovanj osnovne valute v nekem časovnem obdobju. En japonski svečnik nam pove ceno ob začetku obdobja, najnižjo ceno v tem obdobju, najvišjo ceno v tem obdobju in ceno ob koncu obdobja. Sama barva svečnika pa nam pove, ali je začetna cena višja (cena pada, rdeča barva) ali nižja (cena narašča, zelena barva) od zaključne cene (glej sliko 3.1).



Slika 3.1: Prikaz z japonskimi svečniki.

Polja za vnos novega naročila. Vsebuje sledeča polja: polje v kateremu lahko izberemo, kakšno vrsto naročila bomo izvedli (osnovna naročila: *limit, market, stop limit, stop market*) - privzeta vrednost tega polja je *limit* naročilo. Polje v katerega vpišemo količino kriptovalute, ki jo kupujemo ali prodajamo. Vsa ostala polja so odvisna od izbire vrste naročila. Vrste naročil bomo opisali v naslednjem podpoglavju *3.2.1*

Vrste naročil. V primeru, da je naročilo limit, potem moramo vnesti še ceno po kateri želimo prodajati ali kupiti kriptovaluto. Če je naročilo vrste stop moramo vnesti še ceno, ob kateri se bo sprožilo limit ali market naročilo [34, 63, 7].

3.2.1 Vrste naročil

Osnovne vrste naročil:

- Market.
  Najenostavnejši način za kupovanje in prodajo. Pri market naročilu moramo nastaviti samo količino valute, ki jo želimo kupiti ali prodati. Ko naročilo izvedemo, bo borza avtomatsko kupila ali prodala nastavljeno količino (izvedla bo naročilo nad knjigo naročil). Pri takem trgovanju ponavadi kupimo ali prodamo valuto po slabši ceni, saj je vse odvisno od trenutnega stanja knjige naročil.

- Limit.
  Nastavljen je na skoraj vseh borzah kot privzeto naročilo. Temu tipu naročila lahko rečemo tudi omejeno naročilo ali ponudba, nastaviti ji moramo ceno (po kateri želimo kupiti ali prodajati) in količino te valute. V primeru, da je ponudbi možno ustreči takoj, se izvede nad knjigo naročil (kot market naročilo), morebitni ostanek ponudbe pa se doda kot naročilo v knjigo naročil, prav tako celotna ponudba, če ni bila ugodena. Ponudba čaka v knjigi naročil, dokler ji ni ugodeno.

- Stop market.
  Trgovci se večkrat zaščitijo s tem naročilom. Nastavimo dve vrednosti: stop ceno in pa količino valute. Naročilo se izvede takoj, ko cena valute na neki borzi prestopi stop ceno.
  Primer prodaja: 1 ETH = 100 EUR, uporabnik se poskuša zavarovati v primeru, da vrednost valute pade. Nastavi stop maket naročilo: stop cena 95 EUR in količina 20 ETH. V primeru, da v prihodnosti cena

pade na ali pod 95 EUR, se izvede market naročilo nad knjigo naročil (takojšna prodaja po ceni, ki jo trenutno ponuja knjiga naročil).
Primer nakup: 1 ETH = 100 EUR, uporabnik pri tem nastavi, ob kateri ceni (stop cena 105 EUR) v primeru, da bi cena valute rasla, želi kupiti neko količino valute. Če cena valute preskoči 105 EUR, se izvede market naročilo nad knjigo naročil (nakup po ceni, ki jo trenutno ponuja knjiga naročil).

- Stop limit.
  Deluje po istem principu, kot stop market, le da se ob sprožitvi stop cene namesto market naročila izvede limit naročilo. Vnesti pa moramo tudi ceno limit naročila. Pri večji vrednosti naročila je uporaba stop limit naročila bolj priporočljiva kot uporaba stop market naročila, saj v primeru nižje likvidnost trgovalnega para ceno naročila lahko omejimo, da ne izvedemo nakupa ali prodaje po slabši ceni [7, 64].

3.3 Stanje uporabnika

Prikazuje količine za vse kriptovalute, ki si jih uporabnik lasti na izbrani borzi.

Pri vsaki kriptovaluti imamo poleg pogleda stanja tudi dve možnosti:

- Prenos kriptovalute na borzo (*deposit*).
  Ob kliku na deposit, se uporabniku prikaže naslov, kamor mora poslati kriptovalute, da jih prejme na uporabniški račun. Vsak uporabnik ima drugačen naslov. Pri nekaterih kriptovalutah pa poleg naslova potrebujemo še drugi podatek, ki je odvisen od same kriptovalute, vsaka pa ga lahko kliče drugače. Brez tega podatka borza ne bi mogla ugotoviti, kdo je lastnik poslanih kriptovalut. Ta podatek ima več imen, kot so: *public key, memo, message, destination tag, attachment* itd [53, 7].

- Prenos kriptovalute z borze (*withdraw*).

Za prenos kriptovalut z borze mora uporabnik izpolniti zahtevana polja, to sta: naslov, kamor pošilja kriptovalute in količino kriptovalute. Pri nekaterih kriptovalutah se mora izpolniti še dodatno polje, ki spada k naslovu (pri nekaterih kriptovalutah to ni potrebno). Ob potrditvi prenosa borza običajno poskuša s ponovno varnostno avtentikacijo uporabnika. Ta dvostopenjska avtentikacija (2FA) je ponavadi koda, ki jo pridobimo preko Google Authenticatorja, e-pošte ali SMS sporočila [43, 7].

Nekatere borze imajo še tretjo opcijo. Sama borza ima lahko več „navideznih“ denarnic, med katerimi lahko prenašamo kriptovalute (*transfer*) [34, 63].

3.4 Zgodovina uporabnika

Ta pogled nam predstavlja dve vrsti zgodovine:

- Zgodovina uspešno izvedenih naročil.
  Vsa trgovanja, ki jih je uporabnik izvedel, z vsemi podatki: datum in čas, simbol trgovalnega para, ID naročila, ID trgovane transakcije, tip naročila (nakup ali prodaja), vrsta naročila (limit, market, stop market itd.), količina kriptovalute, ki se je kupila ali prodala, cena po kateri se je kupila ali prodala, vsota, ki jo plačamo za nakup ali prodajo in strošek trgovanja (*fee*).

- Zgodovina transakcij.
  Prikazuje vse podatke o prenosih kriptovalut na borzo in z nje. Podatki vsebujejo: datum in čas, simbol kriptovalute, količino prenesene kriptovalute, strošek prenosa kriptovalute (*fee*), ID (enolični identifikator) transakcije na borzi, ID transakcije na verigi blokov (npr. TxId ali TxHash) in status transakcije (npr. uspešno, neuspešno, v postopku, napaka itn.) [13].

Pogled je urejen po času od najmlajšega do najstarejšega dogodka.

3.5 Borzni programski vmesniki - API-ji

Skoraj vse kriptoborze ponujajo uporabnikom aplikacijske programske vmesnike (API), s katerimi lahko trgujejo na sami borzi. Uporabniki lahko te vmesnike uporabijo pri različnih programih in aplikacijah. Največ se jih uporablja pri mobilnih aplikacijah, s katerimi lahko uporabniki trgujejo in pregledujejo svoje stanje kriptovalut. Uporabljajo jih tudi pri aplikacijah, programih za analizo stanja in trgovanja, programih za avtomatsko trgovanje (arbitraža, trgovalne strategije itn.) in ostalih programih, ki jih uporabniki izdelajo sami.

Poglavje 4

Kriptoborza ForkDelta

ForkDelta je decentralizirana kriptoborza, ki deluje preko pametne pogodbe Ethereum. Preko nje lahko trgujemo z vsemi kriptožetoni ERC-20 [34]. Gre za kopijo (*fork*) borze EtherDelta. Borza uporablja EtherDeltino pametno pogodbo za trgovanje.

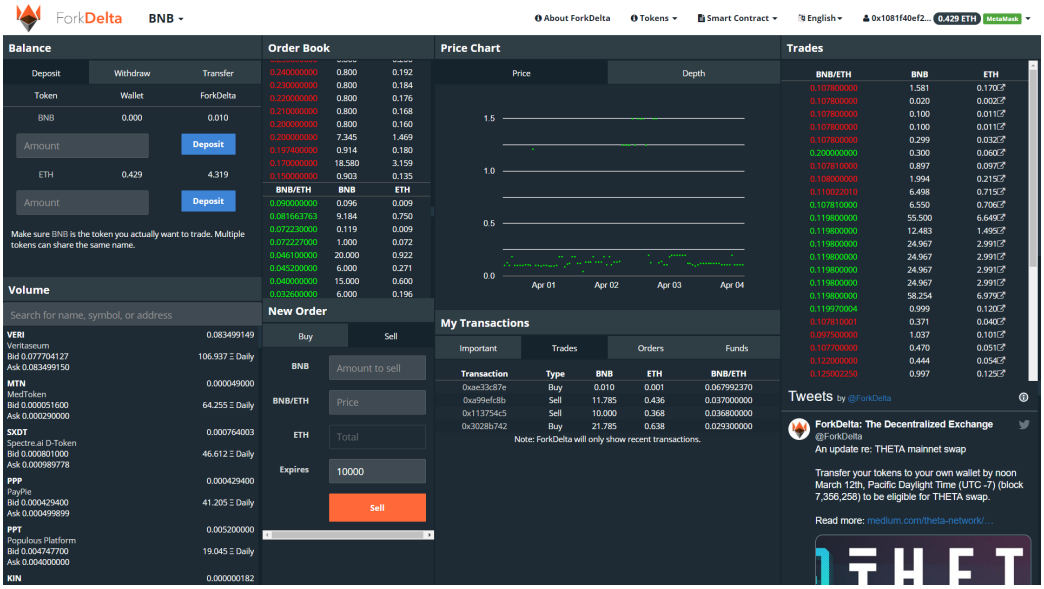
EtherDelta je dostopna na spletni strani `https://etherdelta.com`. Začela je delovati leta 2016. Koda spletne strani in API strežnika sta javno dostopna na githubu [20]. Prav tako je dostopna koda pametnih pogodb, ki jih je sama borza uporablja. Do sedaj se je pojavilo že več kopij te borze, ampak le ForkDelta se je obdržala in je sedaj bolj priljubljena kot EtherDelta [21].

Eden od soustanoviteljev EtherDelte je tudi bil kaznovan s kaznijo skoraj 400 tisoč dolarjev, ker se kriptoborza ni držala zakonov ZDA [25]. Borza je doživela tudi hekerski napad, v katerem je nepridipravom uspelo izprazniti denarnice uporabnikov za približno 270 000 dolarjev [22]. Spletna stran se je prodala anonimnemu kitajskemu podjetju [24].

ForkDelto je ustanovil nekdo pod imenom „freeatnet“ na redditu v januarju 2018. Ustvarjena je bila, ker se upravitelji EtherDelte niso več držali prvotnih načrtov [57]. Prav tako je vsa koda odprtokodna in dostopna na githubu [36].

Ali sta ForkDelta in EtherDelta decentralizirani borzi? Borzi trdita, da

sta, ker imajo uporabniki privatne ključe za dostop do vseh svojih kriptovalut in ker trgovanje poteka preko decentralizirane aplikacije (pametne pogodbe). Nekateri uporabniki pa imajo mnenje, da je nekaj vmes med centralizirano in decentralizirano, ker sama spletna stran (gostovanje) in knjiga naročil (API) nista decentralizirani [34, 37]. Spletna stran borze je dostopna na `https://forkdelta.app` (slika 4.1).



Slika 4.1: Spletna stran kriptoborze ForkDelta.

Da lahko trgujemo na ForkDelti, potrebujemo denarnico Ethereum (privatni ključ denarnice), na kateri imamo nekaj kriptokovancev Ether. Da uporabimo to denarnico na spletni strani borze, moramo povezati našo denarnico s samo spletno stranjo borze, kar lahko storimo na tri različne načine:

- Privaten ključ denarnice vnesemo v samo spletno stran. Ta metoda ni priporočljiva, saj je na spletu več lažnih spletnih strani, ki so na videz enake kot ForkDelta, ki od uporabnika poskušajo pridobiti privatni ključ denarnice. V preteklosti se je že zgodil hekerski napad na EtherDelto, ko so nepridipravi preko strežnika DNS preusmerili samo spletno stran na drug IP naslov [22].

- Uporaba MetaMask dodatka za spletne brskalnike Chrome, Firefox in Opera. MetaMask je denarnica Ethereum, ki omogoča lažjo uporabo Ethereum decentraliziranih aplikacij. Preko MetaMask lahko pošiljamo Ether, upravljamo z drugimi kriptožetoni, podpisujemo transakcije itd. Je najbolj enostaven način za uporabo ForkDelte v spletnem brskalniku. MetaMask omogoča tudi povezavo s strojnimi denarnicami kot sta: Ledger in Trezor [54].

- Uporaba strojne/mehanske denarnice Ledger Nano S. Je najbolj varen način za shranjevanje kriptovalut. Na kratko: gre za USB na katerem so shranjeni privatni ključi kriptovalut. Strojna denarnica omogoča, da se vse transakcije podpišejo na sami napravi, tako noben privatni ključ ne zapusti te naprave. Pri vsakem podpisu transakcije mora uporabnik fizično potrditi transakcijo (npr. pritisniti gumb na sami napravi) [34, 52].

Za komunikacijo z vozliščem Ethereum smo v naši aplikaciji uporabljali knjižnico Nethereum, povezovali pa smo se na vozlišče ponudnika `Infura.io`, ki brezplačno ponuja vmesnik API za dostop do verige blokov [46]. Nethereum je odprtokodna knjižnica narejena za ogrodje .NET. Knjižnica Nethereum nam omogoča, da na enostaven način komuniciramo z vozliščem Ethereuma. Ponuja nam vse potrebne funkcije, ki jih potrebujemo za delo z verigo blokov Ethereum [47].

Težave na sami borzi. Sama spletna stran borze je zelo neprijazna uporabnikom. Razlogov za to je več, tukaj bomo opisali le tiste, ki se nam zdijo rešljivi s strani borze:

- Ko opravimo neko transakcijo (trgovanje, preklic naročila, prenos kriptovalut), nimamo dobrega indikatorja, ki bi nam pokazal stanje transakcije v realnem času. Trenutno se nam ob izvedbi transakcije prikaže med uporabnikovimi trgovanji vrstica, v kateri piše „pending...“. Ko se transakcija doda v blok, pa se ta vrstica spremeni v vrstico, ki prikazuje zgodovino (npr. podatki o trgovanju) transakcije. Težava je v tem, da

API ne deluje vedno pravilno. Uporabniku bi prav prišlo polje, v katerem bi se dalo videti, če se podatki o stanju transakcije posodabljajo. Trenutno večina uporabnikov uporablja `etherscan.io` (spletna stran, ki nam prikazuje podroben pogled v transakcije in druge podatke, ki so na verigi blokov), da preverijo stanje transakcije.

- API ni dovolj zanesljiv. Večkrat imajo uporabniki težave, ker se njihovo stanje po končani transakciji ne posodobi. Prav tako imajo uporabniki probleme z manjkajočo zgodovino, tako svojo, kot zgodovino trgovanja na trgovalnih parih.

- Spletna stran nam ponuja le en prikaz namenjen trgovanju. Sama borza ne ponuja pregleda celotne zgodovine uporabnika in tudi ne ponuja pogleda čez celotno stanje uporabnika. Večina uporabnikov zato uporablja spletno stran `deltabalances.github.io`, ki nam za katerega koli uporabnika prikaže njegovo zgodovino in stanje kriptovalut [34, 37].

4.1 Programski vmesnik ForkDelta

Vse funkcionalnosti ForkDelte in kodo za izvajanje transakcij na verigi blokov Ethereum, smo spisali v knjižnico ForkDelta_API. To knjižnico uporablja glavni del aplikacije, ki povezuje prikaz in logiko delovanja borz.

ForkDeltin programski vmesnik za povezavo uporablja `Socket.IO`, omogoča komunikacijo strežnik-uporabnik v obe smeri v realnem času. Socket.IO se avtomatsko posodobi v povezavo WebSocket, če je to možno. V aplikaciji smo za komunikacijo uporabili knjižnico WebSocket4Net [68, 59].

Namen vmesnika API je, da nam sporoča vse nove ponudbe, ki so prišle v knjigo naročil, in vso dogajanje, ki se izvede preko pametne pogodbe EtherDelta. Vsa komunikacija poteka v JSON obliki.

Dogodki, o katerih nas obvešča vmesnik API:

- `orders`

  Vse nove ponudbe z vsemi potrebnimi podatki, ki so bile posredovane

skozi API vmesnik.

- `trades`
  Vsa trgovanja, ki se izvedejo na pametni pogodbi EtherDelta.

- `funds`
  Vsa nakazila (deposit) in vsa dvigovanja (withdraw), ki se izvedejo na pametni pogodbi EtherDelta.

ForkDeltin aplikacijski programski vmesnik vsebuje dve funkciji:

- `getMarket (token address, user address)`
  Podamo dva parametra, naslov pametne pogodbe kriptožetona, s katerim želimo trgovati, in naslov naše denarnice.

  Kot odgovor pa dobimo:

  - trgovalne podatke za nekatere trgovalne pare (returnTicker),
  - knjigo naročil za ta kriptožeton (orders),
  - naše aktivne ponudbe v tej knjigi naročil (myOrders),
  - zgodovino trgovanja za ta kriptožeton (trades),
  - našo zgodovino trgovanja (myTrades),
  - našo zgodovino prenosov kriptovalut, za ta kriptožeton in kriptokovanec ETH (myFunds).

- `message (order)`
  Uporabljamo, da posredujemo v omrežje novo ponudbo. Strežnik preveri, če ponudba ustreza kriterijem. Če ustreza, se ponudba doda v knjigo naročil, mi pa dobimo odgovor:
  `42["messageResult",[202,"Good job!"]]`.
  V nasprotnem primeru dobimo opis napake [35].

Sam vmesnik API ni decentraliziran! Glavni namen vmesnika je, da skrbi za knjigo naročil. Vse ostale funkcije, ki jih vmesnik ponuja, so le

pomoč pri pridobivanju zgodovine [34]. Te podatke lahko pridobimo tudi sami brez ForkDeltinega aplikacijskega programskega vmesnika. Pri testiranju smo ugotovili, da je zgodovina, ki jo pridobimo s strani vmesnika, zelo nezanesljiva (manjkajoči podatki). Kako pridemo do teh podatkov brez ForkDeltinega vmesnika, pa je opisano v podpoglavju *4.7 Zgodovina transakcij*.

4.2 Izdelava nove ponudbe

Ponudba v JSON obliki, ki jo posredujemo preko ForkDelta API-ja:


```
42["message",{
    "contractAddr":"0x8d12a197cb00d4747a1fe03395095ce2a5cc6819",
    "tokenGet":"0x0000000000000000000000000000000000000000",
    "amountGet":"1092369888908051",
    "tokenGive":"0x0e8d6b471e332f140e7d9dbb99e5e3822f728da6",
    "amountGive":"22742484189968340747",
    "expires":6963915,
    "nonce":1672544967,
    "v":28,
    "r":"0xfb1bece8fcebbacc180bd27c4543ae4a5478...a99a63d5ff6b16b",
    "s":"0x76c1a378bd7cba90292992cc5e11509dccdd...ace17aa7d69f792",
    "user":"0x2f0813eaa1107e634c2d81f40ef6011084bd75db"
    }
]
```


Naslov v atributu `tokenGet` predstavlja kriptokovanec Ether.
Atributi v naročilu:

- `contractAddr` naslov pametne pogodbe, na kateri trgujemo (naslov pametne pogodbe EtherDelta).
- `tokenGet` naslov pametne pogodbe kriptožetona, ki ga dobimo.
- `amountGet` količina kriptožetona, ki ga dobimo (napisana z vsemi decimalnimi mesti)
- `tokenGive` naslov pametne pogodbe kriptožetona, ki ga ponujamo.

- `amountGive` količina kriptožetona, ki ga dajemo (napisana z vsemi decimalnimi mesti)
- `nonce` je naključno število (uint).
- `expires` pove, do kdaj ta ponudba velja in predstavlja številko bloka.
- `v, r, s` uporabljamo za podpis transakcije (naročila).
- `user` naslov uporabnika, ki ponuja ponudbo [35].

Podpis naredimo tako, da najprej izračunamo zgoščeno vrednost podatkov:

```
SHA256(contractAddress, tokenGet, amountGet, tokenGive, amountGive,
    expires, nonce)
```

Pred zgoščeno vrednost vstavimo besedilo (`"\u0019Ethereum Signed Message:\n32"`), pretvorjeno v bajte. Vse to skupaj kriptiramo s funkcijo `Sha3Keccack` in nato podpišemo (ECDSA) s privatnim ključem. Pri tem dobimo podatke podpisa `v, r, s`.

```
bytes32 hash = sha256(contractAddress, tokenGet, amountGet,
    tokenGive, amountGive, expires, nonce);

var sig = secp256k1.sign(sha3("\u0019Ethereum Signed Message:\n32",
    hash), privateKey);
bytes32 r = sig.signature.slice(0, 32);
bytes32 s = sig.signature.slice(32, 64);
bytes8 v = sig.recovery + 27;
```

Iz `v, r, s` in podatkov naročila lahko za nazaj ugotovimo, kdo je uporabnik, ki je podpisal to transakcijo. Na ta način tudi strežnik API in pametna pogodba EtherDelta preverita, če je ponudba pravilno podpisana [23, 28, 35].

```
address user = ecrecover(sha3("\u0019Ethereum Signed Message:\n32",
    hash),v,r,s)
```

4.3 Trgovanje

Trgovanje na decentralizirani borzi ForkDelta poteka precej drugače kot na drugih borzah. Ob vsakem trgovanju se mora ponudba izvesti na pametni pogodbi in se zabeležiti na verigi blokov. Za trgovanje potrebujemo dve osebi (glej sliko 4.2), nekoga, ki izda ponudbo (*maker* - Bob), in nekoga, ki to ponudbo sprejme (*taker* - Alice). Pri običajnih borzah ne vemo, kdo trguje na drugi strani, pri ForkDelti pa so vsa trgovanja z vsemi podatki javna. Še preden sprejmemo ponudbo, poznamo naslov denarnice, ki trguje, kar nam daje vpogled v stanje naslova in celotno zgodovino naslova [34].

Funkcije v pametni pogodbi so lahko bralne `query/read contract` ali pisalne `write contract`. Bralne funkcije ne morejo spremeniti ali shraniti stanja, lahko le vračajo rezultat. Pisalne funkcije pa lahko spremenijo stanje, zaradi česar so tudi plačljive [56].

Ponudbo sprejmemo tako, da kličemo funkcijo `trade` na pametni pogodbi EtherDelta.

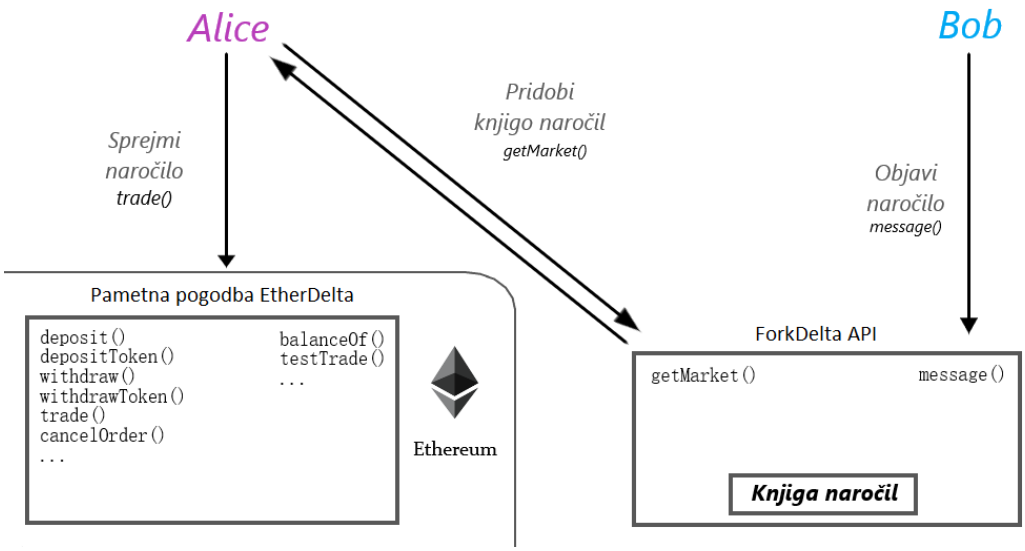
Parametri funkcije `trade` na pametni pogodbi EtherDelta:

```
function trade (address tokenGet, uint amountGet, address tokenGive,
    uint amountGive, uint expires, uint nonce, address user, uint8 v
    , bytes32 r, bytes32 s, uint amount)
```

Vsi parametri so isti, kot nam jih daje ForkDelta API v knjigi naročil, razen zadnjega, ki predstavlja količino (*amount*) kriptožetonov, ki jih nekdo kupuje od prodajalca.

Preden izvedemo funkcijo `trade`, kličemo še poizvedbeno funkcijo `testTrade` z istimi parametri kot `trade` funkcija in enim dodatnim parametrom, to je naslov uporabnika, ki bi izvedel trgovanje. `TestTrade` funkcija nam pove, ali bo `trade` funkcija uspešno zaključena. Ko je funkcija uspešno zaključena, nadaljujemo s funkcijo `trade`, v nasprotnem primeru pa javimo in zabeležimo napako. S tem se izognemo nepotrebni transakciji, ki bi jo morali plačati (*gas*) [36, 23].

Da izvedemo funkcijo `trade`, potrebujemo ABI pametne pogodbe EtherDelta, v katerem je zapisana struktura parametrov funkcije.



Slika 4.2: Postopek trgovanja preko pametne pogodbe EtherDelta.

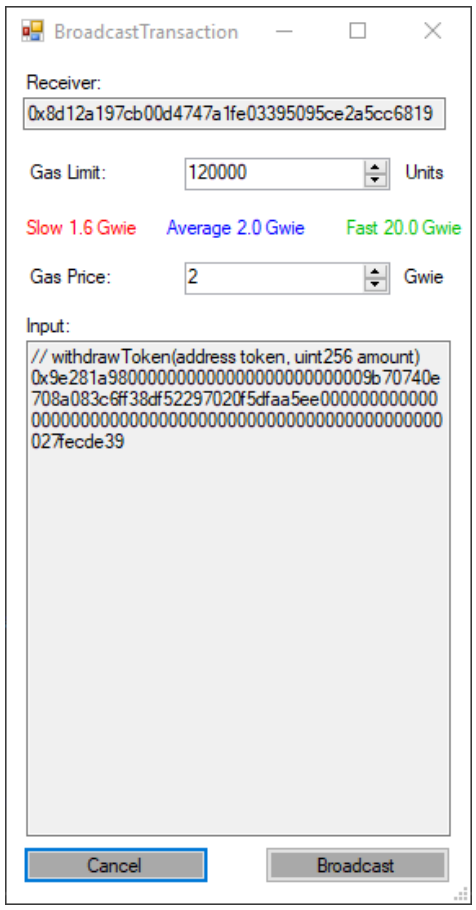
Koda v aplikaciji za pripravo podatkov naročila:

```
var functionInput = new object[] {
                order.TokenGet,
                order.AmountGet,
                order.TokenGive,
                order.AmountGive,
                order.Expires,
                order.Nonce,
                order.User,
                order.V,
                order.R.HexToByteArray(),
                order.S.HexToByteArray(),
                amount
            };

int gasLimit = 120000;
decimal gasPrice = 0;

Contract etherDeltaContract = _web3.Eth.GetContract(abi, Config.
    EtherDeltaSmartContractAddress);
var tradeFunction = etherDeltaContract.GetFunction("trade");
var data = tradeFunction.GetData(functionInput);
```

```
// Open form.
bool send = OpenBroadcastTransactionForm(ref gasLimit, ref gasPrice,
     $"// trade/n{data}");
if (send) {
     BigInteger gasPriceX = Web3.Convert.ToWei(gasPrice,
         UnitConversion.EthUnit.Gwei);
     return EtherDeltaSendTransaction(data, gasLimit, gasPriceX, 0,
         feedbackJob: "trade").Result;
}
```



Slika 4.3: Okno za podpisovanje Ethereum transakcij.

Vsi podatki, ki bi bili vsebovani v transakciji, se pripravijo. Preden podpišemo in pošljemo transakcijo, aplikacija od uporabnika zahteva potrdi-

tev. Prikaže se okno „Broadcast Transaction“ (glej sliko 4.3), v katerem so prikazani vsi podatki. Na vrhu se nam prikaže `Receiver` (v našem primeru je to naslov pametne pogodbe EtherDelta). Vidimo tudi dva polja, ki ju lahko spremenimo. Prvo polje je `gas limit`, drugo pa `gas price`, nad tem poljem vidimo tudi trenutno statistiko priporočenih vrednosti za `gas` ceno transakcije. Podatke o trenutnih cenah transakcij dobimo iz vmesnika API, ki ga ponuja spletna stran `ethgasstation.info` [19]. Kot privzeta se vzame povprečna vrednost, lahko pa vstavimo tudi nižjo ceno, vendar bo transakcija prišla na vrsto kasneje. Če ponudimo več, bo prišla na vrsto prej, saj dajejo rudarji prednost bolj dobičkonosnim transakcijam.

Končna cena transakcije je porabljen `gas limit` pomnožen z `gas` ceno. `Gas limit` ali po slovensko „limit goriva“ je največja vrednost goriva, ki smo jo pripravljeni plačati za transakcijo. Poraba goriva transakcije je odvisna od količine izvedene kode na verigi blokov. Pri transakcijah moramo biti previdni, da nastavimo `gas limit` na dovolj veliko vrednost. V primeru, da je `gas limit` premajhen, se transakcija ne bo uspešno zaključila [29].

Največje polje (*Input*) nam besedilno prikazuje ime funkcije in tipe parametrov za boljšo predstavo naslednjih podatkov. Ti podatki so v obliki bajtov. Prvi štirje bajti od `0x` naprej so zgoščena vrednost funkcije (ime funkcije in tipi parametrov), ostali bajti pa so parametri v funkciji. S klikom na gumb „Sign“ dovolimo, da se transakcija podpiše in posreduje v omrežje Ethereum.

4.4 Preklic ponudbe

Posebnost ForkDelte je tudi, da ima vsaka ponudba rok veljavnosti. Ta je predstavljen s številko bloka, po katerem se konča veljavnost ponudbe. Rudarji za izdelavo novega bloka porabijo povprečno 14,5 sekund. Rok veljavnosti lahko ob izdaji ponudbe nastavi vsak uporabnik. Rok veljavnosti je v vsaki ponudbi zapisan pod vrednostjo atributa „expires“. Če veljavnost ponudbe še ni potekla in želimo ponudbo preklicati moramo, klicati funk-

cijo `cancelOrder` na verigi blokov [35]. To pomeni, da je preklic ponudbe plačljiv, kot vsaka druga transakcija, ki jo posredujemo v omrežje Ethereum.

Funkcija `cancelOrder` potrebuje vse parametre, ki smo jih poslali preko ForkDelta API-ja. Če se nam s preklicom mudi, lahko rudarjem ponudimo več (večji `gas price`), da našo transakcijo prej spravijo v blok [23].

Parametri funkcije `cancelOrder` na pametni pogodbi EtherDelta:

```
function cancelOrder(address tokenGet, uint amountGet, address
    tokenGive, uint amountGive, uint expires, uint nonce, uint8 v,
    bytes32 r, bytes32 s)
```

Prenos vseh kriptovalut, ki smo je navedli v ponudbi, same ponudbe ne prekliče. V tem primeru ponudbi ne bo mogoče ustreči. Če bi kriptovalute spet prenesli nazaj na pametno pogodbo, bi bilo spet mogoče ustreči ponudbi, dokler je ta veljavna.

4.5 Dobroimetje uporabnika

Vsi kriptožetoni ERC-20 imajo v pametni pogodbi funkcijo `balanceOf`, ki sprejema en parameter, to je naslov uporabnika. S to funkcijo lahko preverimo količino kriptožetonov nekega uporabnika. Potem, ko smo kriptožetone prenesli na pametno pogodbo EtherDelta, nam ta ponuja funkcijo za poizvedbo o stanju na njej. To je funkcija `balanceOf`, ki sprejema dva parametra: naslov pametne pogodbe kriptožetona in naslov uporabnika.

Sama ForkDelta in EtherDelta nam ne omogočata, da bi lahko preverili stanje vseh kriptožetonov naenkrat, kot omogočajo vse centralizirane borze. Problem pridobivanja stanja imajo tudi druge decentralizirane borze, kot so IDEX, Token store, Token jar, Paradex idr. Borze to ponavadi rešujejo preko svojih strežnikov, nekatere teh podatkov ne prikazujejo, druge pa pošljejo tudi po več tisoč poizvedb na vozlišče Ethereuma, da pridobijo trenutno stanje enega uporabnika (IDEX). Da decentraliziranim borzam, ki delujejo na verigi blokov Ethereum, poenostavijo takšno iskanje, je nastala spletna stran `deltabalances.github.io`. Ta uporablja pametno pogodbo, s katero

lahko z eno poizvedbo pridobimo stanje nekega naslova za več borz in več kriptožetonov [14, 13, 45].

V aplikaciji smo uporabili poizvedbeno funkcijo `allBalances`. Ta funkcija potrebuje tri parametre: naslov pametne pogodbe borze, naslov uporabnika in listo naslovov pametnih pogodb kriptožetonov, za katere nas zanima stanje. Rezultat funkcije je seznam 32-bajtnih podatkov, ki predstavljajo količino nekega kriptožetona na pametni pogodbi borze v lasti uporabnika in na denarnici (naslov) uporabnika [14, 13].

Koda za poizvedbo o stanju uporabnika:

```
string abi = File.ReadAllText(Config.DeltaBalanceAbiFile);
Contract deltaBalanceContract = _web3.Eth.GetContract(abi, Config.
    DeltaBalanceSmartContractAddress);
var deltaBalancesFunction = deltaBalanceContract.GetFunction("
    allBalances");
var balances = deltaBalancesFunction.CallAsync<List<BigInteger>>(
    Config.EtherDeltaSmartContractAddress, user, listOfTokenAddresses
    .ToArray());


List<AddrBalance> balancesForkDelta = new List<AddrBalance>();
for (int i = 0; i < balances.Result.Count; i += 2)
{
    // FD
    AddrBalance result = new AddrBalance();
    result.Addr = listOfTokenAddresses.ElementAt(i / 2);
    result.Balance = balances.Result.ElementAt(i);
    result.Wallet = false;
    balancesForkDelta.Add(result);

    // Wallet
    AddrBalance resultWallet = new AddrBalance();
    resultWallet.Addr = listOfTokenAddresses.ElementAt(i / 2);
    resultWallet.Balance = balances.Result.ElementAt(i + 1);
    resultWallet.Wallet = true;
    balancesForkDelta.Add(resultWallet);
}
```

4.6 Prenos kriptovalut

Na spletni strani ForkDelte imamo v levem zgornjem kotu polja s katerimi lahko prenašamo kriptovalute. Ta polja imajo kljub istemu imenu drugačen pomen, kot na drugih centraliziranih kriptoborzah. Primer funkcionalnosti deposit: na ForkDelti pri tej funkcionalnosti pridobimo polja v katera vnesemo podatke za prenos kriptovalut na pametno pogodbo EtherDelta. Pri drugih borzah pa ta isto poimenovana funkcionalnost prikaže uporabniku naslov, kamor mora uporabnik prenesti kriptovalute, da se mu te prenesejo na navidezno denarnico borze. V naslednjih podpoglavjih bomo opisali kako te funkcionalnosti delujejo in kako jih uporabljamo v naši aplikaciji.

4.6.1 Prenos kriptovalut na pametno pogodbo EtherDelta

Prenos kriptokovanca ETH na pametno pogodbo poteka popolnoma drugače, kot prenos kriptožetonov. Za prenos kriptokovanca Ether (ETH) moramo poslati eno transakcijo, medtem, ko moramo za prenos kriptožetonov ERC-20 poslati dve.

Za prenos ETH uporabljamo funkcijo `deposit` na pametni pogodbi EtherDelta, ki sicer ne potrebuje nobenega parametra, moramo pa pri tovrstni transakciji poslati neko vsoto kriptokovanca ETH.

Pri prenosu kriptožetonov ERC-20 na pametno pogodbo moramo, tako kot pri kriptokovancu ETH, na našem naslovu imeti neko vsoto nekega kriptožetona. Seveda moramo imeti tudi neko vsoto kriptokovanca ETH, s katerim plačamo gorivo (gas), ki se porabi pri izvedbi transakcije.

Pri prenosu žetonov na pametno pogodbo EtherDelta moramo najprej izvesti funkcijo `approve` nad pametno pogodbo kriptožetona in nato še funkcijo `depositToken`. Standard ERC-20 določa, da morajo vsi kriptožetoni imeti funkcijo `approve`, ki omogoča, da damo nekemu naslovu dovoljenje za prenos naših kriptožetonov.

Funkcija `approve` ali po slovensko „odobri“, sprejema dva parametra:

`spender` (porabnik) predstavlja naslov uporabnika, ki mu odobrimo prenašanje naših kriptožetonov in `tokens`, ki predstavlja količino tega žetona.

Druga funkcija, ki jo je potrebno klicati, da se žetoni prenesejo na pametno pogodbo borze, je `depositToken`. Funkcija je na pametni pogodbi EtherDelta in sprejema dva parametra, to sta: naslov pametne pogodbe kriptožetona, ki ga želimo prenesti, in drugi parameter `amount`, ki prestavlja količino. Funkcija v kodi kliče funkcijo, ki je v pametni pogodbi kriptožetona `transferFrom`. `TransferForm` je standardna funkcija (ERC-20), uporablja pa se za prenašanje odobrenih kriptožetonov [18, 23].

Koda funkcije `depositToken` na pametni pogodbi EtherDelta [23]:

```
function depositToken(address token, uint amount) {
    //remember to call Token(address).approve(this, amount) or this
        contract will not be able to do the transfer on your behalf.
    if (token==0) throw;
    if (!Token(token).transferFrom(msg.sender, this, amount)) throw;
    tokens[token][msg.sender] = safeAdd(tokens[token][msg.sender],
        amount);
    Deposit(token, msg.sender, amount, tokens[token][msg.sender]);
}
```

4.6.2 Prenos kriptovalut s pametne pogodbe EtherDelta

Za dvigovanje kriptokovanca Ether se uporablja funkcija `withdraw`. Sprejema en parameter, to je količina kovanca Ether, ki ga želimo prenesti s pametne pogodbe EtherDelta.

Dvigovanje kriptožetonov ERC-20 s pametne pogodbe EtherDelta pa poteka preko funkcije `withdrawToken`, ki sprejema dva parametra: naslov pametne pogodbe kriptožetona in količino kriptožetona, ki ga želimo dvigniti. Po uspešno izvedeni transakciji se kriptožetoni premaknejo na naslov (denarnico) uporabnika [23].

4.6.3 Prenos kriptovalut z denarnice

Za prenos kriptožetonov z naslova na naslov se uporablja ERC-20 standardna funkcija `transfer`. To funkcijo kličemo nad pametno pogodbo kriptožetona, damo ji dva parametra: naslov, kamor želimo poslati kriptožetone in njihovo količino. Kriptožetone lahko prenesemo tudi tako, da odobrimo prenos kriptožetonov nekemu drugemu naslovu (funkcija `approve`), ki pa jih potem pošlje na želen naslov s funkcijo `transferFrom` [18].

4.7 Zgodovina transakcij

Pridobivanje zgodovine trgovanja iz ForkDelte API-ja je precej nezanesljivo, saj podatki pogosto manjkajo [66, 37]. Da dobimo vse podatke, moramo komunicirati z verigo blokov, v kateri so shranjeni vsi dogodki, ki so se zgodili na pametni pogodbi.

Pametna pogodba EtherDelta ima nekaj funkcij, ki jih moramo uporabniki borze uporabljati, da lahko trgujemo na njej. Te funkcije so:

- `deposit`
- `depositToken`
- `withdraw`
- `withdrawToken`
- `cancelOrder`
- `order`
- `trade`

Če kličemo eno od teh funkcij in se transakcija uspešno izvede, klicana funkcija sproži dogodek (`event`). Ta dogodek se zabeleži v verigo blokov, kar nam omogoča, da ga kasneje lažje poiščemo. Ethereum (JSON RPC API)

ima funkcijo `eth_getLogs`, s katero lahko poiščemo vse zabeležene dogodke, ki so se zgodili na tej pametni pogodbi [30, 23].

Spodaj je koda, s katero poiščemo vse zabeležene dogodke. Ta koda uporablja knjižnico Nethereum. Spodnja koda potrebuje dva parametra: na katerem bloku začnemo iskanje in na katerem bloku končamo iskanje. Vrača pa nam vse dogodke, ki so se zgodili v tem obdobju.

```
string abi = File.ReadAllText("EtherDeltaAbi.json");
Contract etherDeltaContract = _web3.Eth.GetContract(abi,
    EtherDeltaSCAddrerss);
NewFilterInput filterInput = etherDeltaContract.
    GetDefaultFilterInput(new BlockParameter(fromBlock), new
    BlockParameter(toBlock));
FilterLog[] logs = await _web3.Eth.Filters.GetLogs.SendRequestAsync(
    filterInput);
```

Vmesnik ABI je v JSON obliki spisana struktura funkcij in dogodkov, ki jih ponuja neka pametna pogodba. Iz vmesnika ABI lahko razberemo, kakšne parametre moramo podati funkciji in kakšne oblike rezultata nam vrne funkcija [9].

Ko dobimo vse dogodke, ki so se pripetili v tem obdobju, jih lahko ločimo med seboj s pomočjo topica. Topic je 256 bitna zgoščena vrednost, ki jo Ethereum uporablja za iskanje dogodkov, narejen pa je z imena dogodka in tipa parametrov dogodka, te pa pridobimo iz vmesnika ABI pametne pogodbe EtherDelta [12].

Primer postopka za pridobivanje zgoščene vrednosti topica za dogodek `trade`:

```
Keccak256("Trade(address,uint256,address,uint256,address,address)");
```

Rezultat funkcije je zgoščena vrednost:

```
6effdda786735d5033bfad5f53e5131abcced9e52be6c507b62d639685fbed6d
```

Programska koda za preverjanje, če so dogodki tipa `trade`. Če so, preslikamo podatke dogodkov v listo objektov `TradeLog`.

```
if (events.Key == "0x6effdda786735d5033bfad5f53e513.....")
{
    List<EventLog<TradeLog>> tradeEvents = Event.DecodeAllEvents<
        TradeLog>(events.Value.ToArray());
    CheckTrades(tradeEvents);
}
```

Iskanje zgodovine nekega uporabnika je enostavno, saj je treba le prebrati vse transakcije, ki jih je izvedel nek uporabnik (naslov). Problem nastane, če želimo pridobiti zgodovino trgovanja, ko so drugi uporabniki sprejeli naše ponudbe. Problem takega iskanja trgovalne zgodovine je v tem, da je zelo počasno. Pri tem obremenimo strežnik, ki poišče vse dogodke. Ko dobimo vse dogodke, moramo iz teh poiskati svoje, kar je lahko zelo zamudno, saj si moramo prenesti vso zgodovino trgovanja neke borze, da lahko najdemo podatke o naši zgodovini [30].

Poglavje 5

Kriptoborza KuCoin

KuCoin je razmeroma majhna kriptoborza iz Hong Konga. Začela je delovati septembra 2017, dostopna pa je na spletni strani `kucoin.com` [48]. Trenutno ima odprtih približno 400 trgovalnih parov. Ponuja tudi mobilno aplikacijo za naprave Android in iOS, ki omogoča vsa osnovna dejanja, ki jih ponuja borza sama, to so: trgovanje, pregled stanja, pregled zgodovine in prenos kriptovalut na in z borze. Borza trenutno podpira le osnovno vrsto naročil limit [51].

KuCoin uporablja tudi svojo kriptovaluto KuCoin Shares (KSC), s katero uporabniki dobijo številne ugodnosti, kot so: popust pri plačilu trgovanja (*trading fees*), razdelitev 50% vseh plačil trgovanja med uporabnike kriptožetona KSC in druge ugodnosti [2].

KuCoin uporablja dve vrsti vmesnika API:

- REST API in
- WebSocket API.

5.1 Programski vmesnik REST

Uporablja dve vrsti zahtev GET in POST. Vsi klici morajo potekati po protokolu HTTPS, odgovori nanje pa so v JSON obliki. Za razčlenjevanje JSON objektov smo uporabljali knjižnico Newtonsoft.JSON za ogrodje .NET.

Klici, ki se navezujejo na uporabnika morajo biti avtenticirani. KuCoin uporabnikom omogoča uporabo več API ključev. Ob kreaciji novega API ključa lahko izberemo, kakšna dovoljenja bo ključ imel [50].

Kratek opis klicev, ki se uporabljajo v naši aplikaciji (v oklepajih je ime klica katerega uporablja borza):

- **Seznam vseh kriptovalut na tej borzi** (*List coins (Open)*).
  Seznam kriptovalut in podatki o njih. Vsaka kriptovaluta ima tudi druge podatke, ki se uporabljajo v aplikaciji: ime kriptovalute, ali je prenos kriptovalute na borzo aktiven, ali je prenos kriptovalute z borze trenutno možen, minimalna možna količina prenosa z borze, plačilo za prenos kriptovalute z borze idr.

- **Seznam vseh trgovalnih parov, ki jih borza ponuja** (*List trading symbols tick (Open)*).
  V seznamu so tudi drugi podatki o kriptovalutah, kot so vsota količin trgovanj, nakupna cena, prodajna cena, maksimalna cena v 24-ih urah, minimalna cena v 24-ih urah, sprememba cene v 24-ih urah in drugi podatki.

- **Pridobi stanje za vse kriptovalute** (*Get balance of coins*).
  Nam pokaže stanje vseh kriptovalut na našem računu, tistih, ki so nam na voljo, in tistih, ki so zamrznjene (so že v uporabi).

- **Pridobi naslov za prenos kriptovalut na borzo** (*Get coin deposit address*).
  Klic nam za želeno kriptovaluto vrne naslov, kamor lahko prenesemo kriptovalute, da lahko potem z njimi upravljamo na borzi. Pri nekaterih kriptovalutah poleg naslova pridobimo še en podatek, ki je največkrat nujno potreben, da borza lahko določi pošiljatelja kriptovalute.

- **Prenos kriptovalut z borze** (*Create withdrawal apply*).
  S tem klicem lahko prenesemo neko kriptovaluto na nek drug naslov. Če je prenos kriptovalut trenutno možen, in druge podatke, ki se nanašajo

na prenos kriptovalut z borze, dobimo skupaj s seznamom vseh kriptovalut.

- **Ustvari novo naročilo** (*Create an order*).
  Klic potrebuje štiri podatke: simbol trgovalnega para, tip naročila (nakup ali prodaja), ceno in količino. Trenutno borza podpira le limit naročila. Če je bil klic uspešno izveden, nam strežnik vrne enolični identifikator naročila.

- **Seznam vseh odprtih naročil** (*List active orders in kv formats*).
  Nam vrne seznam vseh naših odprtih naročil za vse trgovalne pare. Podatki vsebujejo enolični identifikator naročila ter podatke o samem naročilu.

- **Prekliči naročilo** (*Cancel orders*).
  Za preklic naročila potrebujemo enolični identifikator `oid`, simbol trgovalnega para in tip naročila (nakup ali prodaja), vse te podatke dobimo v seznamu vseh odprtih naročil.

- **Seznam naše zgodovine trgovanj** (*List dealt orders (marged)*).
  Vse informacije o naših trgovanjih. Če parametra `symbol` (simbol trgovalnega para) ne podamo, dobimo zgodovino za vse trgovalne pare, če parameter `symbol` izpolnimo, dobimo zgodovino za izbran trgovalni par. Te podatke uporabljamo pri prikazu zgodovine. Klic nam lahko prikaže največ 100 izvedenih trgovanj na vseh trgovalnih parih. Če kličemo zgodovino le za določen par, pa je omejitev 20 trgovanj. Če potrebujemo več podatkov, lahko ponovno kličemo API in dodamo parameter `page` (številka strani).

- **Zgodovina trgovanja** (*Recently deal orders (Open)*).
  Zgodovina trgovanja na borzi za en trgovalni par. Konstantna sveža trgovanja lahko dobimo z vmesnikom WebSocket.

- **Pridobi knjigo naročil** (*Order books (Open)*).

Knjiga naročil za nek trgovalni par. Konstantna sveža naročila lahko dobimo z vmesnikom WebSocket.

- **Pridobi podatke za graf** (*Get kline data (Open)*).
  Lahko dobimo podatke za 1, 5, 15, 30, 60 minutna, 8 urna, dnevna in tedenska obdobja. Vsako obdobje ima podatke o ceni na začetku in ob koncu obdobja, minimalno in maksimalno ceno, kdaj se je obdobje začelo ter vsoto trgovanj v tem obdobju [50].

Slaba lastnost te borze je, da ne omogoča celotnega pregleda zgodovine prenosa kriptovalut na borzo in z nje. Da pridemo do teh podatkov preko spletne strani, moramo posebej izbrati vsako kriptovaluto in preveriti zgodovino za smer prenosa (*deposit* ali *withdraw*). REST API nam ponuja le klic `List deposit & withdrawal records`, ta pa nam lahko vrne le zgodovino za eno samo kriptovaluto za obe smeri.

Opisani klici, ki jih uporabljamo v aplikaciji, so zahteve GET, trije pa uporabljajo zahtevo POST (ustvari novo naročilo, prekliči naročilo in izvedi prenos kriptovalut z borze).

5.1.1 Avtentikacija programskega vmesnika

KuCoin pri API avtentikaciji uporablja ključ (*key*) in skrivnost (*secret*). Pri vsakem avtenticiranem klicu se podatki avtentikacije zadržujejo v glavi zahteve.
Za avtentikacijo so v glavi klica potrebni trije podatki:

- KC-API-KEY (API ključ),
- KC-API-NONCE (timestamp uporabnika v milisekundah) in
- KC-API-SIGNATURE (kriptografski podpis URL-ja).

KC-API-NONCE (nonce) je trenutni UTC čas predstavljen s številom milisekund od 1. januarja 1970. Nonce ne sme biti starejši od treh sekund v primerjavi s strežniškim časom, če je, strežnik vrne napako.

Kriptografski podpis je narejen iz podatkov (samo pot (*path*) in poizvedba (*query*) v URL-ju) pretvorjenih v bajte. Ti bajti se pretvorijo v `Base64` besedilo. To besedilo pretvorimo v bajte in jih damo na vhod funkciji HMAC-SHA256, ki skupaj s skrivnostjo (*secret*) vrača zgoščeno vrednost (*hash*) v obliki bajtov, ki jih je potrebno le še pretvoriti v besedilo [50].

5.2 Programski vmesnik WebSocket

Več o protokolu WebSocket je napisano v podpoglavju *Kriptoborza HitBTC - 6.2 Programski vmesnik WebSocket*. Da se lahko povežemo na KuCoin WebSocket strežnik, potrebujemo žeton `bulletToken`. Žeton dobimo preko REST API-ja, da ga dobimo, pa nam ni treba iti skozi avtentikaciski postopek.

Za vzpostavitev povezave WebSocket se uporablja knjižnica WebSocket4Net.

Koda za vzpostavitev povezave WebSocket v aplikaciji:

```
string url = $"{apiUrl}?bulletToken={bulletToken}&format=json&
    resource=api";
_ws = new WebSocket(url, userAgent: "Test connection");
_ws.Opened += SocketOpened;
_ws.Error += SocketError;
_ws.Closed += SocketClosed;
_ws.MessageReceived += SocketMessageReceived;
_ws.Open();
```

KuCoin WebSocket nas trenutno lahko osvežuje s štirimi različnimi podatki. Za prejem teh podatkov se je potrebno naročiti (`subscribe`).
Ti podatki so lahko (v oklepajih je ime klica katerega uporablja borza):

- Posodabljanje knjige naročil (*Orderbook level2*),
- Osvežena zgodovina trgovalnega para (*History*),
- Osveženi podatki o trgovalnem paru (*Tick*) in
- Statistični podatki trgovalnih parov (*Market*).

V aplikaciji uporabljamo le podatke za knjigo naročil in zgodovino trgovalnega para. Po naročilu dobivamo samo posodobitve, za pridobitev celotne knjige naročil še vedno potrebujemo REST API. Ko spremenimo trgovalni par, se moramo pred naročilom na drug trgovalni par odjaviti (`unsubscribe`) od prejšnjih naročil. Primer zahteve, ki jo pošljemo na strežnik WebSocket, da se naročimo na knjigo naročil:

```
{
    "id":123,
    "type":"subscribe",
    "topic":"/trade/ETH-BTC_TRADE",
    "req":1
}
```

Tudi WebSocket API vrača odgovore v JSON obliki. Da ostanemo povezani na WebSocket API moramo vsakih 60 sekund kontaktirati (`ping`) strežnik, s čimer potrdimo, da še vedno uporabljamo povezavo. KuCoin API dokumentacija nam svetuje, da se strežniku javimo vsakih 40 sekund. Če se ne javimo več kot 60 sekund, bo strežnik povezavo prekinil [50].

Poglavje 6

Kriptoborza HitBTC

Kriptoborza HitBTC je začela delovati leta 2013. Borza je dostopna na spletni strani `hitbtc.com` [39] in je ena od večjih kriptoborz. Trenutno ima odprtih približno 900 trgovalnih parov. Uporabnikom omogoča tudi bolj napredno trgovanje, kot so stop naročila, naročila aktivna le določen čas itd. Je ena redkih borz, ki nima omejitev za prenos kriptovalut na borzo in z nje. Ima tudi podroben pogled v trenutno statistiko prenosov kriptovalut (System Health) na borzo in z nje [41].

Slabost borze je, da ima pri trgovalnih parih decimalna mesta „nenavadno“ omejena, trgovalni par ETH/BTC je, na primer, omejen na 6 decimalnih mest, trgovalni par DOGE/BTC je omejen na 11 decimalnih mest. Skoraj vse borze uporabljajo fiksna decimalna mesta, največkrat je to 8 decimalnih mest.

Borza uporablja tri vrste programskih vmesnikov API:

- REST API,
- Socket.IO in
- Streaming API (WebSocket API)

`Socket.IO` API, se uporablja samo za posodabljanje podatkov o izvedenih naročilih [42]. Tega API-ja v aplikaciji nismo uporabili, ker te podatke pridobivamo preko Streaming API-ja (WebSocket API) [40].

6.1 Programski vmesnik REST

Vsi odgovori so v JSON obliki, za razčlenjevanje smo uporabljali knjižnico Newtonsoft JSON.

Na kratko opisani klici uporabljeni v aplikaciji (v oklepajih je ime klica, ki ga uporablja borza):

- **Kriptovalute** (*Currencies*).
  Seznam vseh kriptovalut na borzi. Za vsako kriptovaluto so podani podatki: ali je na ali z borze mogoče prenesti kriptovalute, ali je mogoč prenos kriptovalut med „navideznima“ denarnicama (*trading*, *account*) koliko je treba plačati za prenos kriptovalute z borze itn.

- **Trgovalni pari** (*Symbols*).
  Seznam vseh trgovalnih parov z naslednjimi podatki: minimalna možna količina pri trgovanju, najmanjša možna enota pri ceni trgovalnega para, faktor plačila pri kupcu (taker), faktor povračila ustvarjalcu (maker) naročila.

- **Trgovalni podatki za trgovalne pare** (*Tickers*).
  Statistični podatki za vse trgovalne pare, kot so: najvišja nakupna cena, najnižja prodajna cena, zadnja cena izvedenega naročila, najvišja cena v 24-ih urah, najmanjša cena v 24-ih urah, zadnja trgovana cena pred 24 urami, vsota količin vseh trgovanj v zadnjih 24-ih urah za osnovno valuto in kotirano valuto.

- **Japonski svečniki** (*Candles*).
  Zahteva potrebuje podatek o časovni dolžini obdobja, na izbiro imamo: 1, 3, 5, 15 in 30 minutna, 1 in 4 urna, enodnevno, tedensko in mesečno obdobje. Nastavimo lahko tudi druge podatke, v aplikaciji je uporabljen le limit količine obdobij.

- **Stanje kriptovalut na denarnici account in denarnici trading** (*Account balance, Trading balance*).

Da pridobimo vse podatke o stanju naših kriptovalut na borzi, moramo poslati dve zahtevi, saj ima HitBTC dve „navidezni“ denarnici. `Trading balance` nam za vsako kriptovaluto poda razpoložljivo stanje in stanje, ki je trenutno v uporabi (odprta naročila ali prenos med denarnico). `Account balance` nam za vsako kriptovaluto vrača razpoložljivo stanje za prenos ter zadržane kriptovalute (prihajajoče transakcije).

- **Ustvarjanje novega naročila** (*Create New Order*).

- **Pridobi aktivna naročila** (*Get Active orders*).
  Podatki za vsa naša odprta naročila. Med podatki je tudi `clientOrderId`, ki ga potrebujemo v primeru preklica naročila.

- **Preklic naročila preko** `clientOrderId` (*Cancel order by clientOrderId*).
  Za preklic naročila pošljemo zahtevo `DELETE` z enolični identifikatorjem `clientOrderId`.

- **Zgodovina trgovanja uporabnika** (*Trades history*).
  Zgodovino lahko poiščemo po trgovalnem paru in jo lahko tudi sortiramo in uredimo. Aplikacija uporablja le parameter `from`. Ta sprejema datum in čas, od katerega naj nam vrne trgovanja. Limit pridobljenih trgovanj je prevzeto nastavljen na 100, lahko pa ga spremenimo.

- **Pridobi naslov za prenos kriptovalut na borzo** (*Deposit crypto address*).
  Strežniku moramo poslati le simbol kriptovalute, za katero želimo dobiti naslov. Da pridobimo obstoječ naslov, pošljemo zahtevo GET, če želimo nov naslov, pa pošljemo zahtevo POST.

- **Prenos kriptovalut med denarnicama** (*Transfer money between trading and account*).
  HitBTC uporablja dve „navidezni“ denarnici. Prva `account` denarnica

(uporablja tudi ime `main account`) nam omogoča prenos kriptovalut z borze in prenos kriptovalut na `trading` denarnico. Ko kriptovalute naložimo na svoj naslov, ki nam ga da HitBTC borza, se kriptovalute pojavijo na `account` denarnici. Druga `trading` denarnica je namenjena trgovanju, kriptovalute pa lahko prenesemo tudi nazaj na `account` denarnico.

- **Prenos kriptovalut z borze** (*Withdraw crypto*).
  Preden začnemo s prenosom, preverimo, če borza trenutno to omogoča. Te in še nekatere podatke, kot so količina plačila pri prenosu, dobimo s seznamom vseh kriptovalut. Za prenos moramo strežniku poslati POST zahtevo z naslednjimi podatki: simbol kriptovalute, količina kriptovalute skupaj s plačilom prenosa (*fee*), parameter `includeFee` z vrednostjo `true`, s katero sporočamo strežniku, da bo v količini kriptovalute že všteto plačilo prenosa, naslov kamor pošiljamo. Če kriptovaluta uporablja še dodaten naslov, dodamo parameter `paymentId`.

- **Zgodovina transakcij** (*Get transactions history*).
  Zgodovino transakcij dobimo na isti način kot zgodovino trgovanj, le odgovor je drugačen. Strežnik nam vrne seznam podatkov transakcij, ki so: enolični identifikator transakcije na borzi, indeks transakcije, simbol in količina kriptovalute, plačilo prenosa, naslov, zgoščena vrednost s statusom in tipom transakcije ter čas kreiranja in posodobitve transakcije [40, 1].

6.1.1 Avtentikacija programskega vmesnika

REST API uporablja Basic Authentication. Gre za uporabniško ime in geslo ločeno z „:“, ki je pretvorjeno v besedilo formata Base64, ki ga dodamo med glave HTTPS zahteve. Ime glave je „Authorization“, vrednost pa „Basic “ in naše `Base64` besedilo [40, 1].

Koda v aplikaciji za avtentikacijo REST API-ja:

```
using (var client = new HttpClient())
```

```
{
    client.DefaultRequestHeaders.Authorization = new System.Net.Http
        .Headers.AuthenticationHeaderValue("Basic", Convert.
        ToBase64String(Encoding.ASCII.GetBytes($"{_userName}:{
        _password}")));
    client.Timeout = TimeSpan.FromSeconds(10);
    var result = await client.GetAsync(link);
    return await result.Content.ReadAsStringAsync();
}
```

6.2 Programski vmesnik WebSocket

WebSocket je komunikacijski protokol, ki nam nudi komunikacijo v obe smeri preko ene TCP povezave. Ko je povezava odprta, lahko strežnik kadarkoli in brez uporabnikove zahteve pošlje podatke uporabniku. Komunikacija na ta način je precej hitrejša od komunikacije s HTTP protokolom [67]. WebSocket uporabljajo različne aplikacije, kot so spletne klepetalnice, borze za prikaz trgovanja, igre itd.
WebSocket v URL uporablja shemo `ws` ali varno različico `wss`. Primer za HitBTC WebSocket API:

```
wss://api.hitbtc.com/api/2/ws
```

WebSocket API lahko uporabljamo za trgovanje, pridobivanje podatkov stanja na trgovalni denarnici in za pridobivanje svežih podatkov. WebSocket API omogoča naročanje na knjigo naročil, podatke o trgovanju in ceni trgovalnega para (*ticker*), podatke za graf ter podatke o izvedbi naših naročil. Pri trgovanju, pridobivanju stanja denarnice in naročanju na izvedbo naših naročil nas postopek vodi skozi avtentikacijo. Uporabili smo `BASIC Authentication`, lahko bi uporabili tudi avtentikacijo HMAC-SHA256.

Avtentikacija WebSocket povezave:

```
private void SocketSessionAuthentication()
{
    string json = CreateJsonRequest(RequestMethod.login, new { algo
        = "BASIC", pKey = _acc.ApiKey, sKey = _acc.ApiSecret });
```

```
    _ws.Send(json);
}
```

Odgovori se nahajajo v atributu `params` in so isti kot tisti, ki bi jih dobili z REST API-jem. Pri izjemah WebSocket vrne še kakšen dodaten atribut. V primeru, da strežnik vrača več podatkov (`array`), se uporabi še dodaten atribut `data`, ki vsebuje seznam podatkov.

Primer odgovora (protokol JSON-RPC):

```
{
  "jsonrpc": "2.0",
  "method": "snapshotTrades",
  "params": {
    "data": [
      {
        "id": 65757865,
        "price": "0.054656",
        "quantity": "0.057",
        "side": "buy",
        "timestamp": "2017-10-19T16:33:42.821Z"
      }
    ],
    "symbol": "ETHBTC"
  }
}
```

Funkcionalnosti WebSocketa, ki so uporabljene v aplikaciji:

- **Naročanje na osvežene podatke knjige naročil** (`subscribeOrderbook, unsubscribeOrderbook`).
  Ko se naročimo, dobimo na začetku tudi kopijo knjige naročil (`snapshotOrderbook`), ki vsebuje urejen seznam nakupnih in prodajnih naročil. Pri naročanju lahko omejimo velikost knjige naročil, ki jo želimo prejeti. V aplikaciji smo uporabili limit 500 naročil.

  Koda (JSON-RPC), ki jo pošljemo preko protokola WebSocket, da se naročimo na knjigo naročil:

  ```
  {
      "method": "subscribeOrderbook",
  ```

```
    "params": {
        "symbol": "ETHBTC",
        "limit: 500
    },
    "id": 123
}
```

- **Naročanje na trgovanja za nek trgovalni par** (`subscribeTrades`, `unsubscribeTrades`).
  Na začetku dobimo nekaj nedavnih trgovanj na trgovalnem paru. Za vsako trgovanje imamo naslednje podatke: čas izvedbe naročila, ID posla, tip naročila, količino in ceno.

- **Naročanje na izvedbo naših naročil** (`subscribeReports`).
  Lahko se le prijavimo. Strežnik nas obvesti ob vsaki izvedbi naročil v lasti uporabnika.

Preden se prijavimo na drug trgovalni par, se moramo odjaviti (`unsubscribe`) od starega trgovalnega para [1, 40].

Poglavje 7

Aplikacija

Vse več je kriptoborz, ki poleg spletne strani ponujajo uporabnikom tudi svoje grafične uporabniške vmesnike. To so običajno mobilne aplikacije in programi za določene operacijske sisteme, vsi pa se za delovanje povezujemo na aplikacijski programski vmesnik borze.

Vse več je tudi programov, s katerimi lahko trgujemo na več borzah, a nam včasih ne ponujajo vseh možnosti (odvisno od kriptoborz, ki jih uporabljamo). Nekaj takšnih programov je:

- **Silex** (`http://silexapp.com`) Brezplačen program za operacijski sistem Mac OS in Windows. Vsi API ključi so šifrirani in shranjeni na samem računalniku, vsi API klici direktno komunicirajo z borzo. Aplikacija ne zahteva prijavljanja v program, ima le vstopni pin. Trenutno podpira 10 borz, podpira tudi prenašanje kriptovalut [58].

- **Coinigy** (`https://www.coinigy.com`) Plačljiva platforma za spletni brskalnik ter mobilne naprave iOS in Android. Trenutno podpira trgovanje na 14 borzah, na 28 borzah omogoča pregled stanja uporabnika, na 42 borzah pa omogoča spremljanje grafa trgovanja. Omogoča nam obveščanje o cenah tudi preko SMS in e-mail sporočil ter brskalnika. Vsi API ključi so na strežniku. Omogoča boljšo analizo nad uporabnikovimi kriptovalutami [8, 16].

- **TabTrader** (`https://tab-trader.com`) Aplikacija za iOS in Android. Podpira trgovanje na 25 kriptoborzah. Vsi API ključi so na uporabnikovi mobilni napravi. Na aplikaciji lahko nastavimo alarme za obveščanje o prekoračitvi cene trgovalnega para. Ne podpira prenašanja kriptovalut, omogoča pa trgovanje, pregled zgodovine trgovanja in spremljanje stanja naših kriptovalut [60].

Aplikacija, ki jo opisujem v tej diplomski nalogi, je najbolj podobna programu Silex. Silex trenutno ne podpira trgovanja na kriptoborzi HitBTC in ForkDelta. Deluje na operacijskih sistemih Windows, spisan pa je v programskem jeziku C# za ogrodje .NET. Vsi API ključi so shranjeni na disku računalnika, v datoteki `Settings.xml`. Uporabnik mora za pravilno delovanje aplikacije vnesti te ključe. Grafični prikaz je narejen z Windows Forms, aplikacija uporablja 7 različnih prikazov (oken). Za pisanje programske kode je bilo uporabljeno orodje Visual Studio.

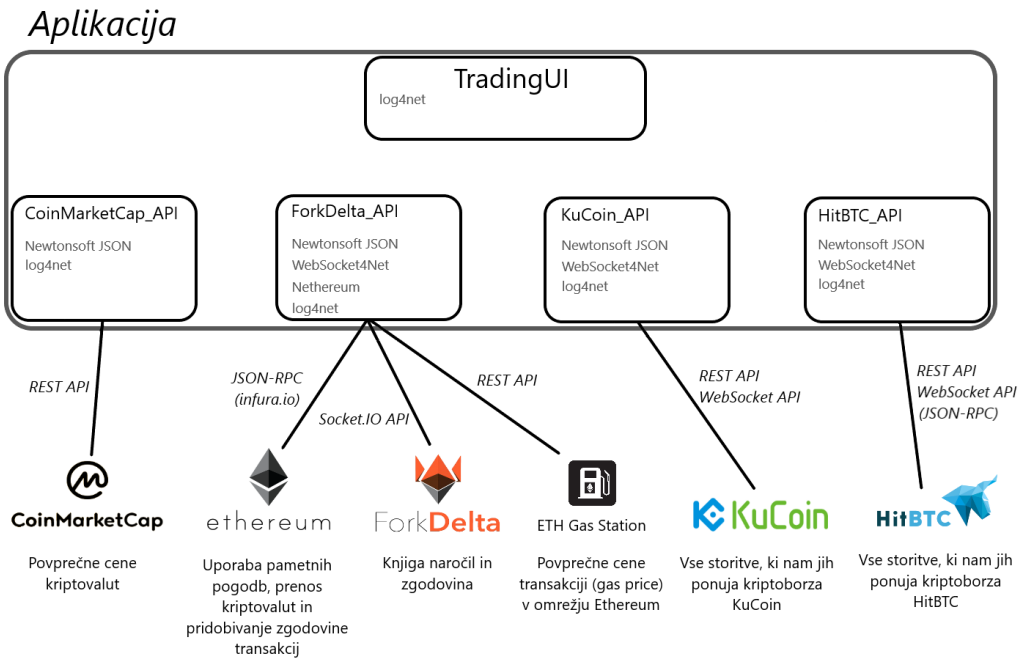
Knjižnice drugih razvijalcev uporabljene v aplikaciji:

- Newtonsoft.Json (za razčlenjevanje JSON besedila) [55],
- WebSocket4Net (knjižnica za uporabo WebSocket povezave) [68],
- Nethereum (knjižnica za komunikacijo z verigo blokov Ethereum) [47],
- log4net (knjižnica za beleženje delovanja aplikacije).

7.1 Struktura aplikacije

Aplikacija je sestavljena iz petih projektov (glej sliko 7.1):

- `TradingUI`. Grafični prikaz in logika, ki združuje spodnje štiri knjižnice v aplikacijo za trgovanje. Grafični prikaz sestavlja šest različnih oken (Windows Forms).
- `CoinMarketCap_API`. Izdelana knjižnica, ki nam daje podatke o cenah na CoinMarketCap spletni strani. Vsebuje bazo, v kateri sta tabeli s



Slika 7.1: Struktura aplikacije, uporaba knjižnic in uporaba API-jev.

podatki o vseh kriptovalutah, ki so na CoinMarketCap spletni strani in njihovih cenah. Podatki se avtomatsko osvežijo na dve minuti. Ta knjižnica se uporablja pri prikazu povprečne cene v knjigi naročil in predvidene vrednosti uporabnikovih kriptovalut v oknu za prikaz stanja kriptovalut uporabnika.

- `HitBTC_API`. Knjižnica vsebuje vse potrebne funkcionalnosti za trgovanje na borzi HitBTC.

- `KuCoin_API`. Knjižnica vsebuje vse potrebne funkcionalnosti za trgovanje na borzi KuCoin.

- `ForkDelta_API`. Knjižnica vsebuje funkcije za komunikacijo z verigo blokov Ethereum, z njimi pridobimo vse potrebne podatke za zgodovino uporabnika in stanje kriptovalut uporabnika. Te funkcije nam omogočajo, da lahko trgujemo na pametni pogodbi EtherDelta. Knji-

žnica ima tudi eno okno (Broadcast transaction) za prikaz podatkov o transakciji, ki bo poslana v omrežje Ethereum, če jo uporabnik potrdi. V knjižnici se tudi nahaja implementiran Socket.IO API za ForkDelto, preko katerega pridobimo knjigo naročil in druge podatke.

Naša aplikacija ima približno 9000 vrstic spisane kode. V to številko ni všteta koda, ki jo Visual Studio generira avtomatsko (koda za prikaz oken, DataSet idr.).

Projekt „TradingUI“ ima za vsako borzo implementiran razred, ki podeduje abstraktni razred Exchange. Ta od razreda borze zahteva, da definira vse potrebne funkcije (abstraktne metode), ki jih aplikacija potrebuje za delovanje. Te funkcije vsebujejo kodo, ki preslikajo podatke, pridobljene z borze (knjižnice borze), v objekte, s katerimi upravlja TradingUI (npr. HitBTC.JsonStructures.Order - TradingUI.Order).

Vsaka knjižnica, ki ima v sebi funkcije za povezavo z borzo, ima kot projekt mapo JsonStructures, v kateri so z razredom opisani vsi večji odgovori API klicev. Te razrede uporabljamo kot izhode funkcij, ko aplikacija razčlenjuje JSON odgovore API-ja. Vsi izhodi funkcij so vrnjeni kot objekti ali kot seznam objektov. Na ta način se v aplikaciji lažje sprehajamo po vrnjenih podatkih.

7.2 Trgovanje

Trgovalno okno nam prikazuje vse informacije, ki jih uporabnik potrebuje za trgovanje na kriptoborzi. Ob zagonu aplikacije se preko programskih vmesnikov kriptoborz pridobijo vsi podatki o kriptovalutah in trgovalnih parih, ki nam jih borze ponujajo. Spodaj v podpoglavjih bomo opisali vse polja, ki nam jih prikazuje aplikacija v trgovalnem oknu.

7.2.1 Trgovalni pari

Ob izbiri borze (v zgornjem levem kotu, glej sliko 7.2) se vsi trgovalni pari naložijo v polje parov, kjer so razvrščeni po abecedi. Priljubljeni pari (ozna-

Slika 7.2: Okno namenjeno trgovanju.

čeni z zvezdico) so pred ostalimi pari in so tudi razvrščeni po abecednem vrstnem redu. Ob kliku na ime stolpca se uredijo po obratnem vrstnem redu.

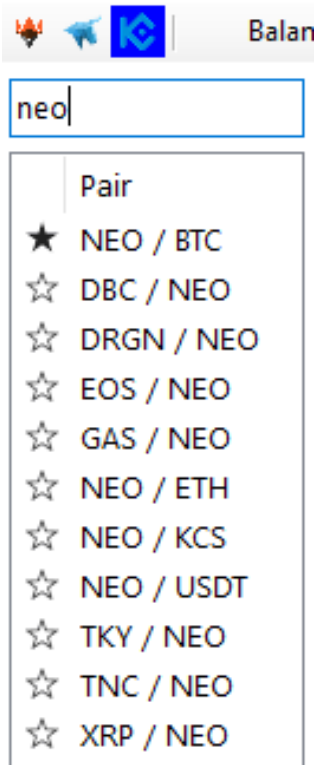
Pri prikazu trgovalnih parov je kriptoborza ForkDelta izjema, saj na njej lahko trgujemo z vsemi kriptožetoni ERC-20. Čeprav nam borza ponuja več tisoč kriptožetonov, vmes tudi kakšen manjka. Da bi lahko trgovali na spletni strani ForkDelte z manjkajočim kriptožetonom, potrebujemo njegov naslov (naslov pametne pogodbe kriptožetona). Spletna stran ForkDelte nam omogoča, da naslov preprosto prilepimo v URL in s tem omogočimo trgovanje z njim. ForkDelta poskuša iz naslova pridobiti podatke o imenu kriptožetona in številu decimalnih mest, ki jih žeton uporablja. Druga možnost je, da kriptožeton ročno vnesemo v seznam žetonov, pri čemer moramo izpolniti tri polja: naslov pametne pogodbe kriptožetona, ime kriptožetona in decimalna mesta, ki jih ta uporablja [66, 34].

Za ročni vnos kriptožetonov v našo aplikacijo se uporablja datoteka v

formatu XML, ki jo aplikacija ob vsakem zagonu prebere. Datoteka se imenuje `ManuallyListedTokens.xml`, v njej pa so kriptožetoni, opisani kot v spodnjem primeru:

```
<Tokens>
    <LongName>Viuly</LongName>
    <ShortName>VIU OLD</ShortName>
    <Address>0x519475b31653e46d20cd09f9fdcf3b12bdacb4f5</Address>
    <Decimals>18</Decimals>
</Tokens>
```

Iskanje trgovalnih parov. Nad seznamom trgovalnih parov je polje, s katerim lahko poiščemo želen trgovalni par (slika 7.3). Ob vsaki spremembi v polju se seznam trgovalnih parov posodobi glede na rezultate iskanja.



Slika 7.3: Seznam trgovalnih parov v oknu za trgovanje.

Priljubljeni pari. Skoraj vse borze imajo možnost, da si trgovci označijo

svoje priljubljene trgovalne pare in s tem hitreje do njih dostopajo. Večkrat pa se je zgodilo, da smo to funkcijo pogrešali na borzi ForkDelta, zato smo jo tudi dodali v aplikacijo. Vse priljubljene trgovalne pare beležimo v datoteko `Settings.xml`, v kateri so shranjene nastavitve uporabnika. Datoteka se ob zagonu aplikacije prebere v podatkovni tip DataSet, kjer najdemo tabelo `Pairs`. Vsakič, ko nek par spremenimo v priljubljenega ali obratno, se podatki znova zapišejo v datoteko.

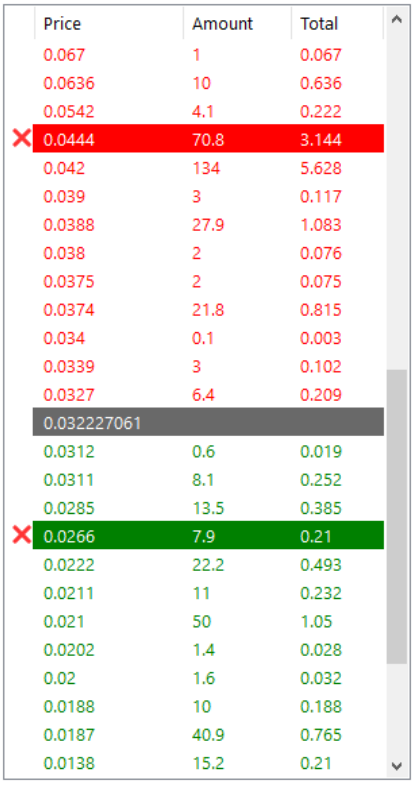
Primer enega zapisa priljubljenega trgovalnega para v datoteki `Settings.xml`:

```
<Pairs>
    <IdOnExchange>VERI-ETH</IdOnExchange>
    <Exchange>ForkDelta</Exchange>
    <Favorite>true</Favorite>
</Pairs>
```

7.2.2 Knjiga naročil

Seznam naročil je urejen po ceni od najdražjega do najcenejšega. Nakupna naročila imajo besedilo obarvano z zeleno, prodajna pa z rdečo barvo. Vrstice, ki so v celoti obarvane z rdečo ali zeleno (in imajo na levi strani vrstice rdeč križec), vsebujejo odprta naročila uporabnika (glej sliko 7.4). Količina teh vrstic je vsota vseh naročil z isto ceno. Seznam se lahko osveži tudi več kot 5x na sekundo. Ponavadi je med najvišjo nakupno in najmanjšo prodajno ceno, ali blizu teh ponudb, sivo obarvana vrstica, v kateri je povprečna cena, pridobljena s spletne strani `CoinMarketCap`. Cena se poišče po simbolu kriptovalute in se osveži vsaki 2 minuti. Ob kliku na vrstico se cena naročila prenese v nakupno in prodajno polje obrazca za izdelavo novega naročila. Če kliknemo nakupno naročilo, se prenese količina v obrazec za prodajo, če pa kliknemo na prodajno naročilo, se količina prenese v obrazec za nakup.

Eno ceno navadno predstavlja ena vrstica (naročilo), v kateri je vsota količin vseh uporabnikov, ki imajo podobno naročilo po isti ceni, kar pa ne velja za ForkDelto, pri kateri ena vrstica pomeni eno naročilo nekega uporabnika, zaradi česar imamo lahko več vrstic z isto ceno. Pri vsaki vrstici

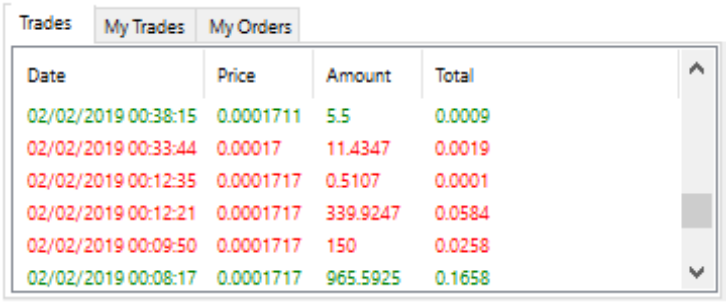


Slika 7.4: Knjiga naročil v trgovalnem oknu aplikacije.

je objekt celotnega ForkDelta naročila, s katerim lahko sprejmemo ponudbo. Ob kliku na naročilo se pri nakupnem ali prodajnem obrazcu prikaže gumb `TAKE` za sprejetje ponudbe.

7.2.3 Zgodovina

Zgodovina trgovalnega para se sproti osvežuje in prikazuje: čas, ceno, količino ter vsoto in tip naročila v obliki obarvane vrstice (zelena za nakup, rdeča za prodajo). Prikaz izvedenih naročil je urejen po času od najmlajšega do najstarejšega (glej sliko 7.5).



Slika 7.5: Nekaj zadnjih trgovanj na trgovalnem paru v trgovalnem oknu aplikacije.

7.2.4 Zgodovina trgovanja uporabnika

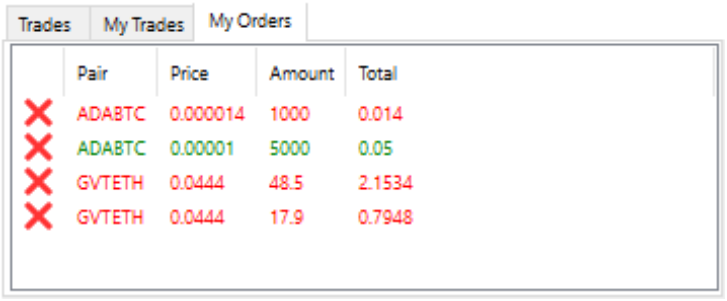
Zgodovina uporabnika trgovalnega para se sproti osvežuje. Prikazuje čas, ceno, količino in vsoto naročila. Prikaz izvedenih naročil je razvrščen po času od najmlajšega do najstarejšega (glej sliko 7.6).

Slika 7.6: Trgovanja uporabnika za izbran trgovalni par.

7.2.5 Aktivna naročila uporabnika

Prikazuje vsa odprta (aktivna) naročila uporabnika za vse trgovalne pare, ki jih borza ponuja (glej sliko 7.7). Prikazuje: ime trgovalnega para, ceno, količino, vsoto naročila in tip naročila v obliki barve vrstice. Pri tem je ForkDelta izjema, saj API uporabniku ne omogoča pridobitve vseh njegovih

odprtih naročil z enim klicem. Za že izbran trgovalni par lahko uporabnik vidi svoja naročila tudi v knjigi naročil, kjer so vrstice obarvane z rdečo ali zeleno (slika 7.4). V teh vrsticah je količina vsota vseh naročil po tej isti ceni, razen pri ForkDelti, pri kateri je vsaka vrstica svoje naročilo.



Slika 7.7: Uporabnikova odprta naročila.

7.2.6 Grafični prikaz zgodovine trgovanja

Ob izbiri trgovalnega para se naloži graf s 30-minutnimi časovnimi obdobji. Če je mogoče, se graf vedno prikaže z 200 obdobji. Zgornji večji graf je trgovalni graf, predstavljen z japonskimi svečniki. Spodnji manjši graf pa je stolpčni graf, ki prikazuje količino trgovanih kriptovalut osnovnega para (glej sliko 7.8).

Japonski svečniki se uporabljajo povsod po svetu za analize trgovanja. Predstavljajo zgodovino trgovanja, pri čemer en svečnik predstavlja eno časovno obdobje. Ponavadi so svečniki obarvani z zeleno ali rdečo. Zelena pomeni, da je v tem obdobju cena zrasla (začetna cena pod zaključno ceno), rdeča pa, da je cena padla (zaključna cena pod začetno ceno).

Borze nam podatke za graf običajno podajo v obliki tabele, v kateri so za vsako obdobje podatki (japonski svečnik): `open` (začetna cena), `close` (zaključna cena), `low` (najnižja cena), `high` (najvišja cena), `timestamp` (datum in čas začetka obdobja), `volume` (vsota trgovanj v osnovni valuti v tem obdobju), v nekaterih primerih dobimo tudi `volumeQuote` (vsota trgovanj v

kotirani valuti v tem obdobju).

Slika 7.8: Grafični prikaz zgodovine trgovanja.

Menjava period. Vse borze, ki imajo vmesnik API za pridobivanje podatkov grafa, omogočajo določitev dolžine časovnih obdobij (japonskih svečnikov). V aplikaciji lahko v levem zgornjem delu grafa izbiramo dolžino časovnih obdobji, ki so nam na voljo na borzi.

Vsak svečnik običajno predstavlja podatke o količini pretrgovanih kriptovalut osnovnega para. Te podatke lahko razberemo iz spodnjega grafa, vendar ne natančno, zato ima aplikacija pomožni prikaz podatkov. Ko miško zadržimo nad svečnikom, se nam v zgornjem delu grafa pokažejo natančni podatki svečnika, vključno s količino.

Da hitreje in lažje odčitamo podatke z grafa, je bil dodan sledilnik miški, ki nam pomaga tako, da prikazuje pravokotni črti na os X in os Y. Natančno pozicijo miške za os Y lahko preverimo v rdečem pravokotniku na strani, kjer

so zapisane vrednosti za odčitavanje cene. Na osi X pa se prikazujeta datum in čas, ki veljata za oba grafa.

Grafa sta narejena z eno samo „chart“ komponento. Med sabo si delita isto os X, ki predstavlja čas.

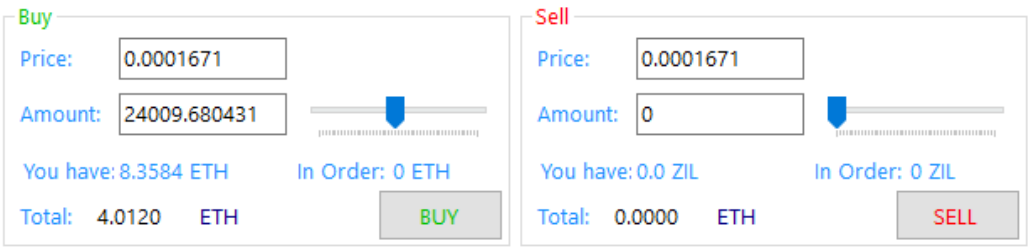
Če se nam zdi pogled svečnikov preveč obširen, si te približamo ali oddaljimo. Ob podrobnejšem pregledovanju svečnikov si lahko pomagamo z miško, s katero prestavljamo povečani del grafa (click and drag).

V primeru, da podatkov za trgovalni graf ni, se ta ne prikaže.

ForkDelta nima podatkov za graf, zato je graf v aplikaciji sestavljen s podatki trgovanj. Vsa trgovanja se razdelijo v obdobja po 30 minut. Iz podatkov za eno obdobje se poišče minimum, maksimum, prvo ceno in zadnjo ceno trgovanja, nato pa se izračuna vsoto količin vseh izvedenih trgovanj in vsoto vseh končnih zneskov trgovanj.

7.2.7 Obrazec za izdelavo novega naročila

Aplikacija ima dva na videz podobna obrazca (glej sliko 7.9), enega za izdelavo nakupnih in drugega za izdelavo prodajnih ponudb. Obrazca sta namenjena izdelavi limit naročil. En obrazec sestavljata polje za vnos cene in polje za vnos količine kriptovalute. Poleg je tudi drsnik, s katerim lahko enostavno vstavimo količino kriptovalute, ki jo bomo kupili ali prodali. S spodnjim gumbom le še potrdimo naročilo.



Slika 7.9: Obrazec za prikaz stanja in odpiranje novih naročil.

Izjema je ForkDelta, ki uporablja dodaten gumb `TAKE`, ki se prikaže ob

kliku na naročilo. S tem gumbom začnemo postopek sprejema naročila drugega uporabnika, pri čemer je uporabljen le podatek v polju količine. Da naročilo izvedemo moramo še podpisati transakcijo.

V obrazcu imamo še informacijo o razpoložljivi količini kriptovalut uporabnika in količino kriptovalut, ki so v odprtih naročilih.

Vsaka borza, ki je implementirana v aplikaciji, ima za pridobivanje stanja kriptovalut uporabnika dve funkciji:

```
public abstract void GetPairBalances();
public abstract void GetAllBalances();
```

Funkcija `GetPairBalances`, ki mora biti implementirana za vsako borzo posebej, lahko podatke dobi na več načinov. Običajno API-ji borz omogočajo pridobitev stanja samo za določeno kriptovaluto in za vse kriptovalute, ki jih borza uporablja. API-ji borze največkrat ne omogočajo pridobitve stanja dveh kriptovalut naenkrat, zato moramo opraviti dva klica ali pa pošljemo zahtevo, da pridobimo stanje za vse kriptovalute z enim samim klicem.

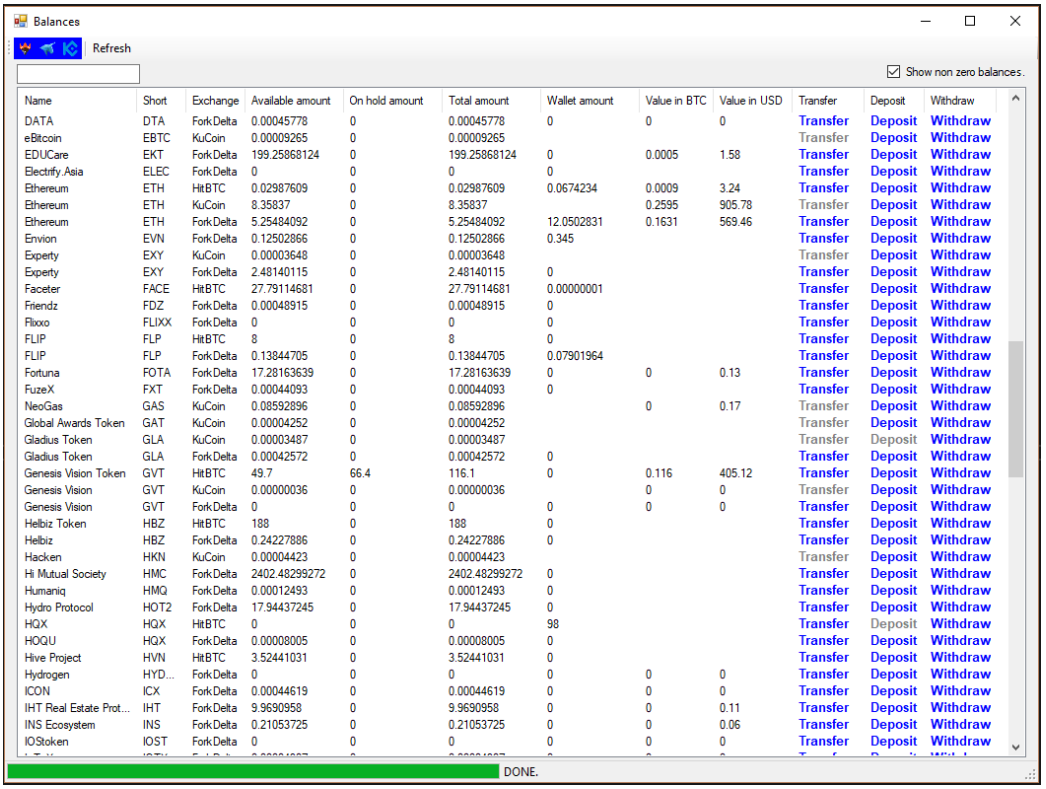
Stanje za oba para aplikacija posodablja vsakič, ko: se zamenja trgovalni par, prekličemo naročilo, izdamo novo ponudbo ali ko nas borza obvesti, da je prišlo do izvedbe naših naročil.

7.3 Stanje kriptovalut in upravljanje z njimi

To okno nam prikazuje stanje kriptovalut uporabnika na kriptoborzah (slika 7.10). Okno nam lahko prikazuje naenkrat vse borze, ali pa le željene. Prikaz borz lahko uporabnik spremeni v zgornjem levem kotu s klikom na ikone borz. Ko je borza izbrana, se ozadje njene ikone spremeni iz belega v modro.

Ponavadi nas zanima le stanje kriptovalut na našem računu, ki so večje od 0. Opcija za prikaz le pozitivnih stanj je vgrajena v zgornji desni kot (`Show non-zero balances`). Če nas zanima stanje za določeno kriptovaluto, jo lahko poiščemo v zgornjem levem kotu. Seznam se osvežuje ob vsaki spremembi v iskalnem polju. Išče se po simbolu ali imenu kriptovalute.

Vse stolpce, ki vsebujejo podatke je mogoče sortirati s klikom na stolpec.



Name	Short	Exchange	Available amount	On hold amount	Total amount	Wallet amount	Value in BTC	Value in USD	Transfer	Deposit	Withdraw
DATA	DTA	ForkDelta	0.00045778	0	0.00045778	0	0	0	Transfer	Deposit	Withdraw
eBitcoin	EBTC	KuCoin	0.00009265	0	0.00009265				Transfer	Deposit	Withdraw
EDUCare	EKT	ForkDelta	199.25868124	0	199.25868124	0	0.0005	1.58	Transfer	Deposit	Withdraw
Electrify.Asia	ELEC	ForkDelta	0	0	0	0			Transfer	Deposit	Withdraw
Ethereum	ETH	HitBTC	0.02987609	0	0.02987609	0.0674234	0.0009	3.24	Transfer	Deposit	Withdraw
Ethereum	ETH	KuCoin	8.35837	0	8.35837		0.2595	905.78	Transfer	Deposit	Withdraw
Ethereum	ETH	ForkDelta	5.25484092	0	5.25484092	12.0502831	0.1631	569.46	Transfer	Deposit	Withdraw
Envion	EVN	ForkDelta	0.12502866	0	0.12502866	0.345			Transfer	Deposit	Withdraw
Experty	EXY	KuCoin	0.00003648	0	0.00003648				Transfer	Deposit	Withdraw
Experty	EXY	ForkDelta	2.48140115	0	2.48140115	0			Transfer	Deposit	Withdraw
Faceter	FACE	HitBTC	27.79114681	0	27.79114681	0.00000001			Transfer	Deposit	Withdraw
Friendz	FDZ	ForkDelta	0.00048915	0	0.00048915	0			Transfer	Deposit	Withdraw
Flixxo	FLIXX	ForkDelta	0	0	0	0			Transfer	Deposit	Withdraw
FLIP	FLP	HitBTC	8	0	8	0			Transfer	Deposit	Withdraw
FLIP	FLP	ForkDelta	0.13844705	0	0.13844705	0.07901964			Transfer	Deposit	Withdraw
Fortuna	FOTA	ForkDelta	17.28163639	0	17.28163639	0	0	0.13	Transfer	Deposit	Withdraw
FuzeX	FXT	ForkDelta	0.00044093	0	0.00044093	0			Transfer	Deposit	Withdraw
NeoGas	GAS	KuCoin	0.08592896	0	0.08592896		0	0.17	Transfer	Deposit	Withdraw
Global Awards Token	GAT	KuCoin	0.00004252	0	0.00004252				Transfer	Deposit	Withdraw
Gladius Token	GLA	KuCoin	0.00003487	0	0.00003487				Transfer	Deposit	Withdraw
Gladius Token	GLA	ForkDelta	0.00042572	0	0.00042572	0			Transfer	Deposit	Withdraw
Genesis Vision Token	GVT	HitBTC	49.7	66.4	116.1	0	0.116	405.12	Transfer	Deposit	Withdraw
Genesis Vision	GVT	KuCoin	0.00000036	0	0.00000036		0	0	Transfer	Deposit	Withdraw
Genesis Vision	GVT	ForkDelta	0	0	0	0	0	0	Transfer	Deposit	Withdraw
Helbiz Token	HBZ	HitBTC	188	0	188	0			Transfer	Deposit	Withdraw
Helbiz	HBZ	ForkDelta	0.24227886	0	0.24227886	0			Transfer	Deposit	Withdraw
Hacken	HKN	KuCoin	0.00004423	0	0.00004423				Transfer	Deposit	Withdraw
Hi Mutual Society	HMC	ForkDelta	2402.48299272	0	2402.48299272	0			Transfer	Deposit	Withdraw
Humaniq	HMQ	ForkDelta	0.00012493	0	0.00012493	0			Transfer	Deposit	Withdraw
Hydro Protocol	HOT2	ForkDelta	17.94437245	0	17.94437245	0			Transfer	Deposit	Withdraw
HQX	HQX	HitBTC	0	0	0	98			Transfer	Deposit	Withdraw
HOQU	HQX	ForkDelta	0.00008005	0	0.00008005	0			Transfer	Deposit	Withdraw
Hive Project	HVN	HitBTC	3.52441031	0	3.52441031	0			Transfer	Deposit	Withdraw
Hydrogen	HYD...	ForkDelta	0	0	0	0	0	0	Transfer	Deposit	Withdraw
ICON	ICX	ForkDelta	0.00044619	0	0.00044619	0	0	0	Transfer	Deposit	Withdraw
IHT Real Estate Prot...	IHT	ForkDelta	9.9690958	0	9.9690958	0	0	0.11	Transfer	Deposit	Withdraw
INS Ecosystem	INS	ForkDelta	0.21053725	0	0.21053725	0	0	0.06	Transfer	Deposit	Withdraw
IOStoken	IOST	ForkDelta	0	0	0	0	0	0	Transfer	Deposit	Withdraw

Slika 7.10: Okno za prikaz stanja kriptovalut uporabnika.

Tabela ima več stolpcev:

- Ime kriptovalute (*Name*).
- Simbol kriptovalute (*Short name*).
- Ime borze, za katero prikazujemo podatke (*Exchange*).
- Količina kriptovalute, ki je na voljo uporabniku (*Available amount*).
- Količina, ki je trenutno v uporabi (*On hold amount*).
  Uporabljena količina v odprtih naročilih.
- Skupna vsota (*Total amount*).
  Vsota prejšnjih dveh stolpcev (*Available amount + On hold amount*).

- Količina, ki jo imamo na denarnici (*Wallet amount*).
  Če ima kriptoborza le eno „navidezno“ denarnico, je polje prazno.

- Predvidena vrednost kriptovalute v Bitcoinu (*Value in BTC*).
  Pomožen stolpec uporabniku prikazuje predvideno vrednost neke kriptovalute v kriptovaluti Bitcoin.

- Predvidena vrednost kriptovalute v ameriških dolarjih (*Value in USD*).
  Pomožen stolpec uporabniku prikazuje predvideno vrednost neke kriptovalute v valuti ameriški dolar.

- Gumb za prikaz okna, s katerim prenašamo kriptovalute med denarnico in trgovalnim računom (*Transfer*).

- Gumb za prikaz okna, s katerim pridobimo naslov, preko katerega lahko naložimo kriptovalute na borzo (*Deposit*).

- Gumb za prikaz okna, s katerim izvedemo prenos kriptovalute z borze (*Withdraw*).

Predvidena vrednost kriptovalute. Izračun dobimo po formuli: Predvidena vrednost = vsa količina uporabnika (total amount) * cena kriptovalute na `CoinMarketCap`. Te podatke za izračun dobi aplikacija preko CoinMarketCap API-ja. Iste podatke uporablja tudi knjiga naročil za prikaz „povprečne“ cene na trgu (siva vrstica). Če cene kriptovalute na CoinMarketCap ne najdemo ali pa je stanje kriptovalute uporabnika 0, se podatek ne prikaže. Ob zagonu aplikacije se preko CoinMarketCap API-ja pridobijo cene vseh kriptovalut, ki jih CoinMarketCap prikazuje na svoji spletni strani. Cene se pridobijo v kriptovaluti Bitcoin (BTC) in valuti ameriški dolar (USD). Podatki se osvežijo vsaki dve minuti.

7.3.1 Prenos kriptovalut med denarnico in trgovalnim računom

Da pridobimo okno `Transfer`, kliknemo na modro obarvano povezavo `Transfer`, če borza nima več denarnic (npr. KuCoin) ali pa ta funkcija trenutno ni na voljo, je povezava obarvana s sivo barvo.

Na levi strani okna imamo denarnico, na desni pa trgovalni račun. Okno nam omogoča, da lahko med njima prenašamo kriptovalute (glej sliko 7.11).

Na levi strani pod vnosnim poljem vidimo količino, ki jo imamo na denarnici (ForkDelta) ali na glavni navidezni denarnici borze (account balance - HitBTC).

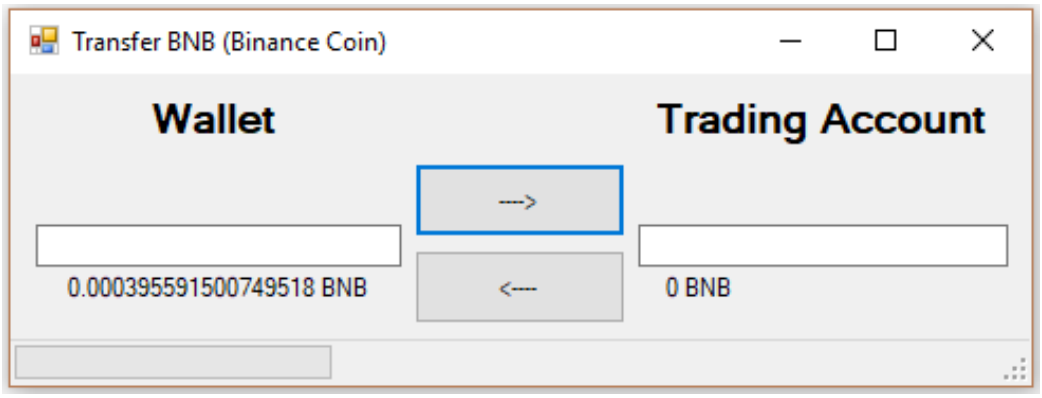
Na desni strani pod vnosnim poljem vidimo količino, ki je trenutno na voljo uporabniku za trgovanje (available amount). Ob kliku na to količino se ta vnese v polje spodaj.

Ob kliku na gumb se kriptovalute prenesejo iz ene strani na drugo, aplikacija zatem pošlje poizvedbo za pridobitev stanja kriptovalut uporabnika. Polja se osvežijo, uporabnik pa izve, če je bil prenos uspešno izveden. Uspešnost prenosa je razvidna iz statusne vrstice, v kateri je trak napredka, ki z zeleno uporabniku prikazuje napredek, poleg pa je tudi besedilo, ki sporoča ime zadnjega napredka. Zahteva in odgovor strežnika se zabeležita v datoteko, ki hrani vse prenose.

Pri ForkDelti se nam pred tem pokaže še okno za potrditev Ethereum transakcije. Da izvemo, če je bila transakcija (prenos kriptovalut) uspešno izvedena, moramo ročno osvežiti seznam kriptovalut, kar naredimo v Balances oknu zgoraj levo z gumbom `Refresh`.

7.3.2 Polog kriptovalut

Za prenos kriptovalut na borzo potrebujemo naslov na katerega pošljemo kriptovalute, da se nam te prikažejo na uporabniškem računu borze. Pri ForkDelti se pridobi naslov denarnice uporabnika saj trgovanje poteka preko

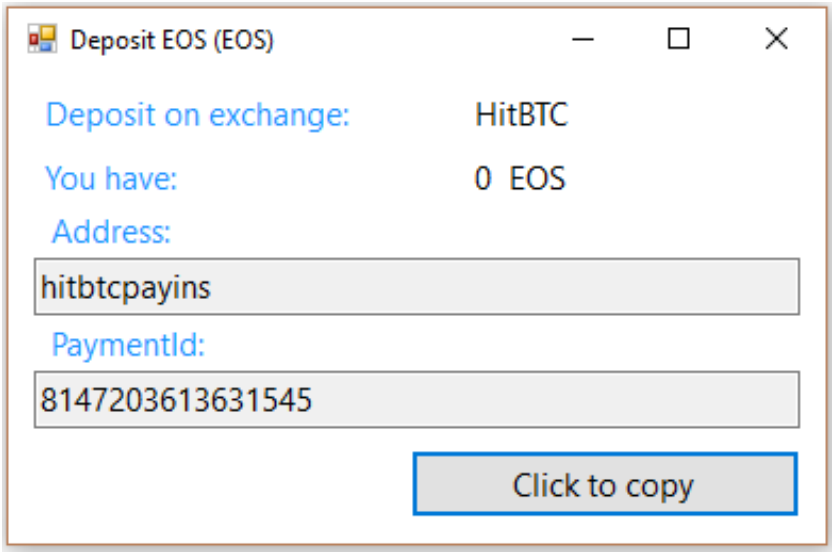


Slika 7.11: Okno za prenos kriptovalut med denarnicama borze.

pametne pogodbe. Okno Deposit pridobimo s klikom na modro obarvano povezavo `Deposit`, ki kliče funkcijo, ki za izbrano kriptovaluto na borzi pošlje poizvedbo za naslov. Borza lahko pošlje samo naslov ali pa poleg naslova še drug podatek. Ta drug podatek ima več različnih imen, kot so: *public key, memo, message, destination tag, attachment* itd. Običajno je to ID, s katerim lahko borza ugotovi, kdo je poslal kriptovalute. V primeru dveh podatkov se prikažeta dve polji. Ob kliku na gumb `Click to copy`, se kopira vsebina v naslovnem polju, tako, da lahko uporabnik prilepi naslova, kakor želi. V oknu so še druge informacije: količina razpoložljivih kriptovalut na borzi, za katero kriptovaluto je prikazan podatek in na kateri borzi.

7.3.3 Dvig kriptovalut

Preko okna za prenos kriptovalut s kriptoborze, posredujemo borzi naslov kamor bi radi prenesli naše kriptovalute. Izjema je ForkDelta, pri kateri ta funkcionalnost omogoča, da uporabnik prenese kriptovalute s svoje denarnice. Da pridemo do okna `Withdraw`, kliknemo na modro povezavo v seznamu kriptovalut. Siva povezava pomeni, da prenos kriptovalut trenutno ni mogoč. Te podatke aplikacija pridobi skupaj s seznamom vseh kriptovalut in podatkov o njih. Poleg teh podatkov so tudi podatki za: minimalni

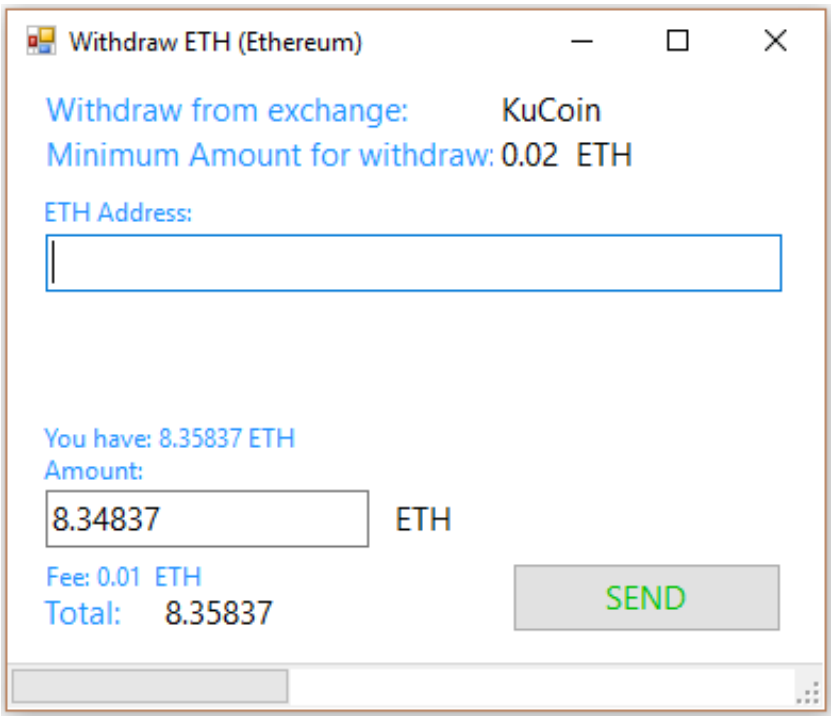


Slika 7.12: Prikaz okna za prenos kriptovalut na borzo.

možni prenos kriptovalut z borze, plačilo pri prenosu kriptovalut, ali je poleg naslova potreben še kakšen podatek itd.

Pri nalaganju okna se vsi našteti podatki še enkrat pridobijo. Okno nam nato prikaže polje za vnos naslova (glej sliko 7.13), na katerega želi uporabnik poslati svoje kriptovalute. Pri nekaterih kriptovalutah se pojavi še polje za dodaten podatek k naslovu. Spodnje vpisno polje je namenjeno vpisu količine, pod njim je zapisana tudi količina, ki je na voljo uporabniku, ob kliku nanjo, se ta vpiše v polje za količino. Okno nam prikazuje tudi minimalno možno količino poslanih kriptovalut (določi borza) in znesek plačila borzi za prenos. Z gumbom `SEND` se strinjamo in pošljemo kriptovalute.

Aplikacija bo avtomatsko preverila, če je znesek večji od minimalnega možnega zneska, ki nam ga borza predpisuje. Če ni, nam aplikacija vrne napako, v tem primeru se nam ta prikaže v novem oknu, napaka pa se beleži na enak način kot zahteva in vsak odgovor borze pri uporabi funkcij za prenos kriptovalut in trgovanja. Ali je bilo zahtevi ugodeno, lahko spremljamo skozi spodnjo statusno vrstico. Ta vsebuje statusni trak, ki z zeleno ponazarja stanje poleg pa je besedilo, ki opisuje zadnje stanje v procesu.



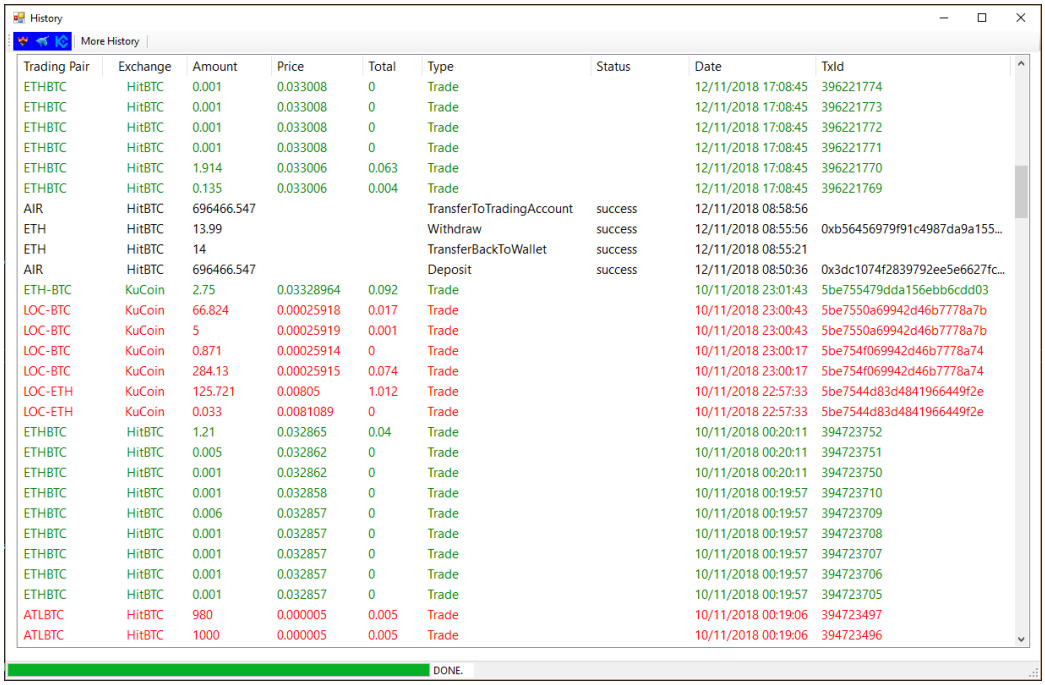
Slika 7.13: Okno za izvedbo prenosa kriptovalut z borze.

7.4 Zgodovina uporabnika

Okno nam v tabeli prikazuje zgodovino transakcij in zgodovino trgovanj (slika 7.14). Na začetku se naložijo podatki za vse borze, ki so vgrajene v aplikaciji. Na spodnji strani okna imamo statusno vrstico, ki nam prikazuje napredovanje pri pridobivanju zgodovine. V levem zgornjem kotu (ikone borz) lahko nastavimo za katere borze želimo prikazati podatke.

Tabela ima sledeče stolpce:

- Ime trgovalnega para (*Trading pair*).
- Ime kriptoborze (*Exchange*).
- Količina kriptovalute (*Amount*).
- Cena (*Price*).



Trading Pair	Exchange	Amount	Price	Total	Type	Status	Date	TxId
ETHBTC	HitBTC	0.001	0.033008	0	Trade		12/11/2018 17:08:45	396221774
ETHBTC	HitBTC	0.001	0.033008	0	Trade		12/11/2018 17:08:45	396221773
ETHBTC	HitBTC	0.001	0.033008	0	Trade		12/11/2018 17:08:45	396221772
ETHBTC	HitBTC	0.001	0.033008	0	Trade		12/11/2018 17:08:45	396221771
ETHBTC	HitBTC	1.914	0.033006	0.063	Trade		12/11/2018 17:08:45	396221770
ETHBTC	HitBTC	0.135	0.033006	0.004	Trade		12/11/2018 17:08:45	396221769
AIR	HitBTC	696466.547			TransferToTradingAccount	success	12/11/2018 08:58:56	
ETH	HitBTC	13.99			Withdraw	success	12/11/2018 08:55:56	0xb56456979f91c4987da9a155...
ETH	HitBTC	14			TransferBackToWallet	success	12/11/2018 08:55:21	
AIR	HitBTC	696466.547			Deposit	success	12/11/2018 08:50:36	0x3dc1074f2839792ee5e6627fc...
ETH-BTC	KuCoin	2.75	0.03328964	0.092	Trade		10/11/2018 23:01:43	5be755479dda156ebb6cdd03
LOC-BTC	KuCoin	66.824	0.00025918	0.017	Trade		10/11/2018 23:00:43	5be7550a69942d46b7778a7b
LOC-BTC	KuCoin	5	0.00025919	0.001	Trade		10/11/2018 23:00:43	5be7550a69942d46b7778a7b
LOC-BTC	KuCoin	0.871	0.00025914	0	Trade		10/11/2018 23:00:17	5be754f069942d46b7778a74
LOC-BTC	KuCoin	284.13	0.00025915	0.074	Trade		10/11/2018 23:00:17	5be754f069942d46b7778a74
LOC-ETH	KuCoin	125.721	0.00805	1.012	Trade		10/11/2018 22:57:33	5be7544d83d4841966449f2e
LOC-ETH	KuCoin	0.033	0.0081089	0	Trade		10/11/2018 22:57:33	5be7544d83d4841966449f2e
ETHBTC	HitBTC	1.21	0.032865	0.04	Trade		10/11/2018 00:20:11	394723752
ETHBTC	HitBTC	0.005	0.032862	0	Trade		10/11/2018 00:20:11	394723751
ETHBTC	HitBTC	0.001	0.032862	0	Trade		10/11/2018 00:20:11	394723750
ETHBTC	HitBTC	0.001	0.032858	0	Trade		10/11/2018 00:19:57	394723710
ETHBTC	HitBTC	0.006	0.032857	0	Trade		10/11/2018 00:19:57	394723709
ETHBTC	HitBTC	0.001	0.032857	0	Trade		10/11/2018 00:19:57	394723708
ETHBTC	HitBTC	0.001	0.032857	0	Trade		10/11/2018 00:19:57	394723707
ETHBTC	HitBTC	0.001	0.032857	0	Trade		10/11/2018 00:19:57	394723706
ETHBTC	HitBTC	0.001	0.032857	0	Trade		10/11/2018 00:19:57	394723705
ATLBTC	HitBTC	980	0.000005	0.005	Trade		10/11/2018 00:19:06	394723497
ATLBTC	HitBTC	1000	0.000005	0.005	Trade		10/11/2018 00:19:06	394723496

Slika 7.14: Okno za prikazovanje zgodovine uporabnika.

- Skupni znesek (*Total*).
- Vrsta dogodka (*Type*). Lahko so: trade, deposit, withdraw, transfer to trading account, transfer back to wallet itd.
- Status naročila (*Status*).
- Datum in čas (*Date and time*).
- Enolični identifikator transakcije (*TxId*).

Če gre za trgovanje, so vrstice prikazane v rdečem (prodaja) ali zelenem (nakup) besedilu. Vse druge so prikazane v črnem besedilu (deposit, withdraw in transfer).

Vse stolpce je mogoče sortirati, na začetku pa so sortirani po času. Okno ima tudi gumb `More history`, s katerim lahko pridobimo več uporabnikove zgodovine, če jo uporabnik ima.

KuCoin trenutno nima API klica, s katerim bi lahko pridobili celotno zgodovino transakcij prenosov kriptovalut uporabnika. Naslednja verzija vmesnika API pa naj bi to možnost imela, več o tem je napisano v *zaključku*.

Za pridobitev zgodovine trgovanja pri borzi ForkDelta bi aplikacija potrebovala precej časa. Težava nastane pri pridobivanju podatkov trgovanj, ko so drugi uporabniki borze sprejemali naše ponudbe, zato je aplikacija nastavljena, da išče zgodovino ForkDelte le za nekaj dni nazaj. Več o tem je napisano v poglavju *Kriptoborza ForkDelta - 4.7 Zgodovina transakcij*.

7.5 Beleženje delovanja aplikacije

Ko razvijalci programske opreme pišejo kodo, si velikokrat pomagajo z beleženjem podatkov. Ko program pišejo, beleženje uporabljajo, da preverijo, če program pravilno deluje. Ko je program že narejen, beleženje uporabljajo pri bolj kompliciranih dogodkih in za beleženje napak. Na ta način lahko razvijalci lažje ugotovijo, kje v kodi se nahaja napaka. Nekateri programi so sestavljeni tako, da lahko uporabnik zažene program v razvojni različici (`debug mode`), program s tem beleži še več podatkov.

7.5.1 Knjižnica log4net

Za beleženje aplikacije se uporablja odprtokodna knjižnica log4net. Ta vse pomembne dogodke shranjuje v datoteko `AppLog.log`, ki se shrani na lokacijo, kjer je bila aplikacija pognana. Sama knjižnica nam daje možnost, da lahko beležene podatke pošiljamo preko e-mail sporočil, jih vnašamo v podatkovno bazo SQL in še ostale bolj napredne funkcionalnosti. V primeru, da bi radi še kaj več, lahko knjižnico spreminjamo in ji dodajamo nove funkcionalnosti [3].

Nastavitve log4net knjižnice ima aplikacija zapisane v datoteki `log4-net.config`. Za beleženje v programih imamo na voljo pet različnih nivojev: Debug, Info, Warning, Error in Fatal. V vsakem razredu, ki uporablja beleženje, je inicializirana vrstica:

```
private static readonly log4net.ILog log = log4net.LogManager.
    GetLogger(System.Reflection.MethodBase.GetCurrentMethod().
    DeclaringType);
```

Da zabeležimo nek podatek, kličemo v kodi npr. `log.Warn("Podatek");`

7.5.2 Beleženje zahtev in odgovorov strežnika

Beleženje zahtev in odgovorov strežnika se izvede ob dogodkih, ki vplivajo na stanje uporabnika.

Funkcije, ki vplivajo na stanje uporabnika:

- prenos kriptovalut med denarnicami borze (*transfer*),
- prenos kriptovalut z borze (*withdraw*),
- trgovanje (*trade*) in
- preklic naročila (*cancel order*).

Tako lahko uporabnik pogleda zgodovino upravljanja s kriptovalutami, ki so bile premaknjene ali trgovane s pomočjo te aplikacije. Beleženje se shranjuje v datoteko `RequestResponseLog.xml`.

Vsako beleženje je sestavljeno iz dveh delov: zahteve, ki jo aplikacija pošlje preko vmesnika API na strežnik (request) in odgovor strežnika na to zahtevo (response).

Beležijo se podatki (v oklepajih so imena parametrov):

- Enolični identifikator zahteve (*Id*).
- Ime borze (*Exchange*).
- Kaj beležimo (*Job*).
  Gre za opis, kaj se beleži, npr. withdraw, transfer to wallet, trade itd.
- URL zahteve (*RequestUrl*).

- Podatki v zahtevi (*RequestData*).
- Napaka aplikacije (*Error*).
  Če gre za napako aplikacije, potem je ta zastavica postavljena na `true` in zastavica „Successful“ na `false`. Če gre za zavrnitev s strani strežnika je zastavica postavljena na `false`.
- Ali je bila izvedba uspešna (*Sucsessful*).
  Če odgovor strežnika pove, da je bilo zahtevi ugodeno, bo zastavica postavljena na `true`.
- Podatki odgovora strežnika (*ResponseData*).
- Informacije (*ResponseInfo*).
  V primeru napake se v to polje shrani besedilo napake, v nasprotnem primeru pa se shranijo drugi podatki, ki so bili pridobljeni iz odgovora strežnika.
- Datum in čas

Primer beležene zahteve in odgovora na njo:

```
<Feedback>
    <Id>JDXXC9N8SIVWSL6PV122</Id>
    <Exchange>HitBTC</Exchange>
    <Job>Transfer to trading acc.</Job>
    <RequestUrl>https://api.hitbtc.com/api/2/account/transfer</
       RequestUrl>
    <RequestData>[currency, FLP], [amount, 8], [type, bankToExchange
       ]</RequestData>
    <Error>false</Error>
    <Successful>true</Successful>
    <ResponseData>{"id":"7ab82f27-ed92-4261-bb4b-317c177b9806"}</
       ResponseData>
    <ResponseInfo>7ab82f27-ed92-4261-bb4b-317c177b9806</ResponseInfo
       >
    <Time>2019-01-30T15:10:25.7396289+01:00</Time>
</Feedback>
```

Ti beleženi podatki so tudi uporabljeni v aplikaciji, da uporabnika preko statusne vrstice obvestijo o napredovanju zahtev in o uspešnosti zahteve.

7.6 Nastavitve aplikacije

Preden začnemo uporabljati aplikacijo, moramo v datoteko `Settings.xml` nastaviti podatke za avtentikacijo API-ja ali pa privatni ključ denarnice. Podatki se preberejo ob zagonu v podatkovni tip DataSet in se uporabijo pri avtentikaciji. Ta ista datoteka vsebuje tudi podatke o priljubljenih trgovalnih parih uporabnika.

Primer:

```
<ApiCredentials>
    <Exchange>HitBTC</Exchange>
    <Key>i5634h8564gjergh8rghr34</Key>
    <Secret>8gsd7gh89sd798g7d89g7d89g7dagdsg7s8d89</Secret>
</ApiCredentials>
```

Poglavje 8

Zaključek

Aplikacija nam za vse tri izbrane kriptoborze ponuja enoten prikaz trgovanja, pogleda zgodovine uporabnika in okna za upravljane s kriptovalutami. Z aplikacijo lahko trgujemo, preverjamo odprta naročila uporabnika, spremljamo knjigo naročil in si ogledujemo grafični prikaz zgodovine trgovalnega para. Aplikacija nam omogoča tudi prenos kriptovalute z borze, prenos kriptovalute med virtualnimi denarnicami borze in pridobitev naslova, na katerega položimo kriptovalute, da jih prejmemo na virtualno denarnico borze.

Aplikacija je narejena tako, da bi lahko v prihodnosti (če bi se izkazala za koristno) dodali še kakšno novo kriptoborzo. V prihodnosti bi lahko aplikaciji dodali tudi možnost uporabe drugih vrst naročil, kot so *stop limit, stop market* itn., seveda, če nam borza to omogoča.

Izdelana aplikacija nam v praksi najbolj koristi pri trgovanju na Fork-Delti, saj sama spletna stran za trgovanje ni dovolj prijazna uporabniku. Za še boljšo uporabniško izkušnjo, bi aplikacija potrebovala dodatno logiko, s katero bi prikazovali stanja vseh transakcij, ki so bila poslana v omrežje Ethereum.

Nedavno je borza KuCoin javnosti predstavila podatke o naslednji verziji vmesnika API 2.0, ki naj bi izboljšal funkcionalnost API-ja in dodal tudi nove klice. Iz dokumentacije API-ja je razvidno, da je med novimi klici tudi klic, s katerim lahko pridobimo zgodovino vseh prenosov na borzo in z nje,

kar smo pri dosedajšnjem vmesniku API pogrešali. Prav tako so borzi dodali nove tipe naročil, kot so: *stop limit, market, stop market* itd. Borza se je tudi odločila, da bo imela dve „navidezni“ denarnici (main account, trading account) na podoben način, kot jih ima HitBTC. Nov vmesnik API ima prenovljene vse klice, kar pomeni, da, ko bo vmesnik API 2.0 začel delovati, v aplikaciji vse funkcionalnosti KuCoina ne bodo več delovale [49].

Ker je aplikacija zelo obširna, je v njej gotovo še nekaj napak, ki jih bo potrebno popraviti. Prav tako bi bilo potrebno testirati pretvorbo nekaterih komponent Windows Forms v komponente WPF, za bolj stabilen prikaz. WPF prikaz bi bilo treba testirati pri prikazu knjige naročil (ListView), zdaj namreč ob osvežitvi knjige naročil (tudi do 5-krat na sekundo) prikaz večkrat na hitro utripne, kar je včasih moteče. To se dogaja pri bolj trgovanih trgovalnih parih.

Literatura

[1] About HitBTC API. Dosegljivo: https://api.hitbtc.com/. [Dostopano: 26. 04. 2019].

[2] About KuCoin bonus. Dosegljivo: https://kucoin.zendesk.com/hc/en-us/articles/360014925893-About-KuCoin-Bonus. [Dostopano: 29. 04. 2019].

[3] Apache log4net Manual - Configuration. Dosegljivo: https://logging.apache.org/log4net/release/manual/configuration.html. [Dostopano: 29. 04. 2019].

[4] Decentralized Crypto Exchange Bancor Hacked, 12M USD in Ether Stolen. Dosegljivo: https://www.ccn.com/decentralized-crypto-exchange-bancor-hacked-12m-in-ether-stolen. [Dostopano: 26. 04. 2019].

[5] White paper. Bitcoin: A Peer-to-Peer Electronic Cash System. Dosegljivo: https://bitcoin.org/bitcoin.pdf. [Dostopano: 28. 04. 2019].

[6] Decentralized Asset Exchange BitShares. Dosegljivo: https://bitshares.org/technology/decentralized-asset-exchange. [Dostopano: 25. 04. 2019].

[7] Bitstamp FAQ. Dosegljivo: https://www.bitstamp.net/faq/. [Dostopano: 28. 04. 2019].

[8] Spletna stran Coinigy. Dosegljivo: https://www.coinigy.com/. [Dostopano: 27. 04. 2019].

[9] Contract ABI specification. Dosegljivo: https://solidity.readthedocs.io/en/v0.5.3/abi-spec.html. [Dostopano: 29. 04. 2019].

[10] Cryptocurrency exchanges explained. Dosegljivo: https://cryptocurrencyfacts.com/what-is-a-cryptocurrency-exchange/. [Dostopano: 26. 04. 2019].

[11] Decentralized vs. centralized exchanges. Dosegljivo: https://hacken.io/decentralized-and-centralized-exchanges-advantages-vs-disadvantages-aa9a27da4584/. [Dostopano: 29. 04. 2019].

[12] Deep dive into Ethereum logs. Dosegljivo: https://codeburst.io/deep-dive-into-ethereum-logs-a8d2047c7371. [Dostopano: 26. 04. 2019].

[13] DeltaBalances GitHub. Dosegljivo: https://github.com/DeltaBalances/DeltaBalances.github.io. [Dostopano: 25. 04. 2019].

[14] Koda pametne pogodbe deltabalances.github.io. Dosegljivo: https://etherscan.io/address/0x3e25f0ba291f202188ae9bda3004a7b3a803599a#code. [Dostopano: 24. 04. 2019].

[15] Different types of crypto exchanges. Dosegljivo: https://coindoo.com/different-types-of-crypto-exchanges/. [Dostopano: 28. 04. 2019].

[16] Coinigy Digital currency trading made easy. Dosegljivo: https://www.coinigy.com/features/. [Dostopano: 29. 04. 2019].

[17] What is ERC-721? Dosegljivo: http://erc721.org/. [Dostopano: 27. 04. 2019].

[18] ERC20 Token Standard. Dosegljivo: https://theethereum.wiki/w/index.php/ERC20_Token_Standard. [Dostopano: 28. 04. 2019].

[19] Spletna stran ETH Gas Station. Dosegljivo: ethgasstation.info. [Dostopano: 27. 04. 2019].

[20] EtherDelta API GitHub. Dosegljivo: https://github.com/etherdelta. [Dostopano: 24. 04. 2019].

[21] EtherDelta FAQ. Dosegljivo: http://etherdeltafaq.com/. [Dostopano: 27. 04. 2019].

[22] EtherDelta Hack. Dosegljivo: https://news.bitcoin.com/one-week-etherdelta-hack-funds-still-stolen/. [Dostopano: 25. 04. 2019].

[23] Koda pametne pogodbe EtherDelta. Dosegljivo: https://etherscan.io/address/0x8d12a197cb00d4747a1fe03395095ce2a5cc6819#code. [Dostopano: 29. 04. 2019].

[24] EtherDelta sold to china. Dosegljivo: https://chainstate.org/2018/11/08/etherdelta-founder-charged-with-operating-an-unregulated-exchange/. [Dostopano: 25. 04. 2019].

[25] SEC Charges EtherDelta Founder With Operating an Unregistered Exchange. Dosegljivo: https://www.sec.gov/news/press-release/2018-258. [Dostopano: 27. 04. 2019].

[26] P2pf wiki ethereum. Dosegljivo: http://wiki.p2pfoundation.net/Ethereum. [Dostopano: 29. 04. 2019].

[27] Wikipedia ethereum. Dosegljivo: https://en.wikipedia.org/wiki/Ethereum. [Dostopano: 27. 04. 2019].

[28] Ethereum ecrecover signature verification and encryption. Dosegljivo: https://ethereum.stackexchange.com/questions/2256/ethereum-ecrecover-signature-verification-and-encryption. [Dostopano: 29. 04. 2019].

[29] What is Gas?, Ethereum. Dosegljivo: https://kb.myetherwallet.com/gas/what-is-gas-ethereum.html. [Dostopano: 29. 04. 2019].

[30] Ethereum JSON RPC API dokumentacija. Dosegljivo: https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getlogs. [Dostopano: 27. 04. 2019].

[31] Ethereum White Paper. Dosegljivo: https://github.com/ethereum/wiki/wiki/White-Paper. [Dostopano: 29. 04. 2019].

[32] Exchange types. Dosegljivo: https://www.bestbitcoinexchange.io/exchange-types/. [Dostopano: 29. 04. 2019].

[33] Spletna stran kriptovalute Fantom. Dosegljivo: https://fantom.foundation/. [Dostopano: 26. 04. 2019].

[34] ForkDelta about. Dosegljivo: https://forkdelta.app/about/. [Dostopano: 25. 04. 2019].

[35] ForkDelta API GitHub. Dosegljivo: https://github.com/forkdelta/backend-replacement/tree/master/docs/api. [Dostopano: 29. 04. 2019].

[36] ForkDelta GitHub. Dosegljivo: https://github.com/forkdelta. [Dostopano: 26. 04. 2019].

[37] ForkDelta Review. Dosegljivo: https://www.forexbrokerz.com/brokers/fork-delta-review. [Dostopano: 28. 04. 2019].

[38] When Do I Need a Blockchain? A Hands-on Comparison of Decentralized and Centralized Application Development, Deseta prosojnica v

prezentaciji prikazuje delovanje trgovanja na pametni pogodni EtherDelta. Dosegljivo: https://www.forexbrokerz.com/brokers/fork-delta-review. [Dostopano: 29. 04. 2019].

[39] Spletna stran kriptoborze HitBTC. Dosegljivo: https://hitbtc.com. [Dostopano: 26. 04. 2019].

[40] HitBTC API Guide. Dosegljivo: https://demo.hitbtc.com/api#. [Dostopano: 27. 04. 2019].

[41] HitBTC Review. Dosegljivo: https://coincentral.com/hitbtc-review-is-it-safe/. [Dostopano: 27. 04. 2019].

[42] HitBTC Socket.IO API dokumentacija. Dosegljivo: https://demo.hitbtc.com/api#socketio. [Dostopano: 28. 04. 2019].

[43] How to perform a withdrawal. Dosegljivo: https://support.hitbtc.com/hc/en-us/articles/115005140465-How-to-perform-a-withdrawal. [Dostopano: 29. 04. 2019].

[44] Haber, Stuart; Stornetta, W. Scott (January 1991). How to time-stamp a digital document. Dosegljivo: https://www.anf.es/pdf/Haber_Stornetta.pdf. [Dostopano: 25. 04. 2019].

[45] Spletna stran kriptoborze IDEX, My balances. Dosegljivo: https://idex.market/balances. [Dostopano: 29. 04. 2019].

[46] Infura API. Dosegljivo: https://infura.io. [Dostopano: 24. 04. 2019].

[47] Knjižnica Nethereum. Dosegljivo: https://nethereum.com/. [Dostopano: 27. 04. 2019].

[48] Spletna stran kriptoborze KuCoin. Dosegljivo: https://www.kucoin.com/#/. [Dostopano: 29. 04. 2019].

[49] KuCoin API Getting Started dokumentacija. Dosegljivo: https://docs.kucoin.com/. [Dostopano: 28. 04. 2019].

[50] Kucoin API Documentation. Dosegljivo: https://kucoinapidocs.docs.apiary.io/#. [Dostopano: 25. 04. 2019].

[51] KuCoin Exchange Review. Dosegljivo: https://coincentral.com/kucoin-exchange-review/. [Dostopano: 28. 04. 2019].

[52] Spletna stran s prikazanimi podatki o Ledger Nano S. Dosegljivo: https://www.ledger.com/products/ledger-nano-s. [Dostopano: 28. 04. 2019].

[53] HitBTC making a deposit. Dosegljivo: https://support.hitbtc.com/hc/en-us/articles/115005090545-Making-a-deposit. [Dostopano: 27. 04. 2019].

[54] Spletna stran MetaMask. Dosegljivo: https://metamask.io/. [Dostopano: 25. 04. 2019].

[55] Spletna stran knjižnice Newtonsoft JSON. Dosegljivo: https://www.newtonsoft.com/json. [Dostopano: 27. 04. 2019].

[56] Reading values from a contract: When do I need transactions? Dosegljivo: https://ethereum.stackexchange.com/questions/7058/reading-values-from-a-contract-when-do-i-need-transactions. [Dostopano: 25. 04. 2019].

[57] Reddit. "I forked EtherDelta": ForkDelta. Dosegljivo: https://www.reddit.com/r/EtherDelta/comments/7odsni/i_forked_etherdelta_forkdelta/. [Dostopano: 27. 04. 2019].

[58] Spletna stran programa Silex. Dosegljivo: https://silexterminal.com/#/. [Dostopano: 29. 04. 2019].

[59] Socket.IO (engine.io) Github. Dosegljivo: https://github.com/socketio/engine.io. [Dostopano: 29. 04. 2019].

[60] Spletna stran programa Tab trader. Dosegljivo: https://tab-trader.com/. [Dostopano: 29. 04. 2019].

[61] Token vs Coin: What's the difference? Dosegljivo: https://www.bitdegree.org/tutorials/token-vs-coin/. [Dostopano: 27. 04. 2019].

[62] Top 5 "blockless" blockchain projects leveraging on DAG technology. Dosegljivo: https://cryptoverze.com/top-5-blockless-blockchain-dag/. [Dostopano: 26. 04. 2019].

[63] HitBTC trading tools. Dosegljivo: https://support.hitbtc.com/hc/en-us/articles/360000713609-Trading-tools. [Dostopano: 28. 04. 2019].

[64] Trgovalni slovar. Dosegljivo: https://admiralmarkets.si/education/traders-glossary. [Dostopano: 26. 04. 2019].

[65] What is the difference between a cryptocurrency trading platform and a cryptocurrency broker exchange? Dosegljivo: https://www.quora.com/What-is-the-difference-between-a-cryptocurrency-trading-platform-and-a-cryptocurrency-broker-exchange. [Dostopano: 29. 04. 2019].

[66] User Experience on 8 Popular DEXs – EtherDelta / ForkDelta. Dosegljivo: https://joyso.io/user-experience-on-8-popular-dexs-etherdelta-forkdelta/. [Dostopano: 26. 04. 2019].

[67] The WebSocket Protocol RFC 6455. Dosegljivo: https://tools.ietf.org/html/rfc6455. [Dostopano: 29. 04. 2019].

[68] Knjižnica WebSocket4Net na GitHub. Dosegljivo: https://github.com/kerryjiang/WebSocket4Net. [Dostopano: 27. 04. 2019].

[69] Wikipedia blockchain. Dosegljivo: https://en.wikipedia.org/wiki/Blockchain. [Dostopano: 25. 04. 2019].

[70] Wikipedia cryptocurrency. Dosegljivo: https://en.wikipedia.org/wiki/Cryptocurrency. [Dostopano: 24. 04. 2019].