

**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I
INFORMACIJSKIH TEHNOLOGIJA OSIJEK**

Sveučilišni studij

Simulator trgovanja kriptovalutama

Završni rad

Ante Bartulović

Osijek, 2018.

Sadržaj

1. UVOD	1
1.1. Zadatak završnog rada.....	1
2. KORIŠTENE TEHNOLOGIJE.....	3
2.1. Android operacijski sustav	3
2.2. Android Studio	4
2.3. Kotlin.....	4
2.4. Aplikacije za trgovanje.....	5
2.5. Kriptovalute.....	5
2.6. Burze kriptovaluta	6
3. MOBILNA APLIKACIJA ZA TRGOVANJE KRIPTOVALUTAMA	7
3.1. MVP	7
3.2. API	8
3.3. Baza podataka	12
3.4. Kupovina i prodaja	12
3.5. Portfolio.....	15
4. TESTIRANJE I USPOREDBA S VEĆ POSTOJEĆIM RJEŠENJIMA	16
4.1. Testiranje aplikacije	16
4.2. TradeHero.....	18
4.3. WallStreet Survivor.....	19
4.4. Coinbase	20
5. ZAKLJUČAK	21
LITERATURA.....	23
SAŽETAK.....	25
ABSTRACT	26
ŽIVOTOPIS	27

1. UVOD

Tema završnog rada je osmisliti i implementirati mobilnu aplikaciju na operacijskom sustavu Android, koja omogućuje korisniku simulaciju trgovanja kriptovaluta korištenjem virtualnog novca. Osnove funkcionalnosti aplikacije su pregled cijena 10 najvećih kriptovaluta korištenjem coinmarketcap.com API-a, mogućnost kupovine i prodaje tih istih kriptovaluta korištenjem virtualnog novca, pregled portfelja korisnika s određenim karakteristikama.

Potreba za ovakvom aplikacijom na tržištu postoji, jer omogućuje korisnicima jednostavno i sigurno sudjelovanje na virtualnom tržištu bez korištenja vlastitih financijskih sredstava. Ideja je omogućiti korisnicima Android operacijskog sustava trgovanje bez rizika, tj. predočiti im način rada burzi kriptovaluta većinom mlađim korisnicima koji bi jednog dana htjeli sudjelovati na otvorenom tržištu.

Na početku rada opisat će se osnovne funkcije aplikacija za trgovanje kriptovaluta, kriptovalute i o burzama kriptovaluta, način rada, broj korisnika, mogućnosti i primjeri takvih burzi. U drugom dijelu rada, bit će govora o tehnologijama korištenim kako bi se uspješno realizirao zadatak završnoga rada, kratak opis svake tehnologije i osnovne funkcionalnosti tih tehnologija koji se koriste u ovome radu, primjer korištenih biblioteka, modela i funkcija. To su Android operacijski sustav, Android Studio integrirano razvojno okruženje, programski jezik Kotlin, API servisi za spajanje na coinmarketcap.com API kako bi dohvatili cijene 10 najvećih kriptovaluta korištenjem Retrofit biblioteke i Room Android biblioteku za konfiguriranje i korištenje baze podataka.

1.1. Zadatak završnog rada

Koristeći Android Studio i programski jezik Kotlin, potrebno je osmisliti i napisati mobilnu aplikaciju za simulaciju trgovanja kriptovaluta. Osnovne funkcionalnosti aplikacije su spajanje na vanjski server te dohvaćanje podataka u stvarnom vremenu koristeći aplikacijsko programsko sučelje (API), prikazivanje cijena na stvarnom tržištu, omogućiti korisniku kupovinu i prodaju kriptovaluta po trenutnim cijenama na simuliranom tržištu, te pregled portfelja. Opisati osnovne

principe rada Android Studia, te samog Android operacijskog sustava i tehnologije koje su korištene u izradi ovoga rada.

2. KORIŠTENE TEHNOLOGIJE

U ovom segmentu ukratko ćemo nabrojati i opisati tehnologije korištene za realizaciju zadatka korištenjem aplikacije namjenjene Android operacijskom sustavu. Aplikacija će se koristiti na Android pametnim telefonima s verzijom 4.4+, KitKat i nasljednici. Kao IDE (integrirano razvojno okruženje) korišten je Android Studio. Kod je pisan u programskom jeziku Kotlin, a grafičko sučelje u jeziku XML. Za cijene, korišten je API (aplikacijsko programsko sučelje) web stranice coinmarketcap.com, a za bazu podataka korišten je Room, biblioteka za konfiguriranje i korištenje baze podataka unutar Androida.

2.1. Android operacijski sustav

Android operacijski sustav, kao platforma za korištenje aplikacije, odabrana je radi najveće zastupljenosti na tržištu unutar posljednjih 12 mjeseci, statistika dostupna na [1], te radi jednostavnosti samog procesa kreiranja aplikacije uz pomoć Android Studia. Najniža verzija operacijskog sustava na kojoj će aplikacije raditi je Android KitKat 4.4 jer obuhvaća 90,1% svih uređaja baziranih na Android-u.

ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
4.0 Ice Cream Sandwich	15	
4.1 Jelly Bean	16	99,2%
4.2 Jelly Bean	17	96,0%
4.3 Jelly Bean	18	91,4%
4.4 KitKat	19	90,1%
5.0 Lollipop	21	71,3%
5.1 Lollipop	22	62,6%
6.0 Marshmallow	23	39,3%
7.0 Nougat	24	8,1%
7.1 Nougat	25	1,5%

Slika 1. Zastupljenost verzije Android operacijskog sustava na uređajima.

2.2. Android Studio

Android Studio je služben integrirano razvojno okruženje (IDE) razvijeno za potrebe dizajniranja Android aplikacija od strane Google-a, kreatora Android operacijskog sustava, te JetBrains-a, kreatora IntelliJ IDEA razvojnog okruženja na kojem je baziran Android Studio. Osim Android Studia, na tržištu postoji veliki broj Android IDE-a ovisno o tome na koju tehnologiju se bazirate, najpoznatiji IDE uz Android Studio su: Visual Studio, IntelliJ IDEA, Eclipse i NetBeans. Osim razvoja mobilnih aplikacija, Android Studio nudi mogućnosti razvoja aplikacija za pametne satove, tablete, pametne televizore i aplikacije za Android auto. Prednosti Android Studia naspram spomenutih je osim službene podrške Google-a i mogućnost simuliranja aplikacije koristeći službeni Android emulator koji je ugrađen unutar Android Studio-a, te omogućava korištenje predložaka, pregledavanje layout-a/GUI-a (grafičkog korisničkog sučelja), dostupnost na mnogim platformama i jednostavnost integriranja Google-ovih servisa.

2.3. Kotlin

Android Studio nudi opciju pisanja koda aplikacije u više različitih jezika, primarno Java, no ovaj rad će u potpunosti biti napisan u programskom jeziku Kotlin, razvijenog od strane JetBrains-a za potrebe Android operacijskog sustava, te u potpunosti podržan od strane Google-a. Glavne prednosti Kotlina nad Javom su null safety, te nedostatak boilerplate koda, a na službenoj stranici Kotlina dostupnoj na [2], navodi se da je količina koda potrebna za jednaku radnju manja do 40%. Kotlin je relativno novi jezik s brzo rastućom zajednicom kako u svijetu, tako i u Hrvatskoj. Izvršavanje Kotlina odvija se u JVM-u (Java Virtual Machine), te je u potpunosti interkompatabilan s programskim jezikom Java, što omogućava kombiniranje Java-e i Kotlin-a unutar istog projekta.

2.4. Aplikacije za trgovanje

U posljednjih nekoliko godina s razvojem mobilnog sklopovlja i tehnologija korištenih za dizajniranje mobilnih aplikacija, omogućeno je kreiranje sofisticiranih programerskih rješenja za trgovanje na raznim burzama korištenjem mobilnih uređaja. Sve veće kompanije namijenjene trgovanju u svom portfelju posjeduju mobilne aplikacije za tu svrhu. Jedna od najpoznatijih aplikacija namijenjena trgovanju je zasigurno Robinhood, koja je ove godine prema [3] prešla brojku od 4 milijuna korisnika, a omogućava trgovanje dionicama i kriptovalutama bez plaćanja provizije. Još od poznatijih mobilnih rješenja na tržištu su E*Trade sa 3.7 milijuna korisnika i TD Ameritrade Mobile. Osim aplikacija koje sudjeluju na stvarnom tržištu, postoje i aplikacije koje simuliraju tržište, te su većinom namijenjene edukaciji korisnika. Primjer takve mobilne aplikacije je TradeHero koja prema [4] broji 3.6 milijuna korisnika u cijelom svijetu. TradeHero je odličan primjer takve aplikacije i od nedavno nude simulaciju trgovanja kriptovalutama. Osim TradeHero-a, postoji još i Wall Street Survivor, web aplikacija koja simulira Wall Street burzu dostupna na [5].

2.5. Kriptovalute

Svoju povijest kriptovalute započinju 1993. godine pojavom ideje o korištenju procesorske moći kako bi se izbjegli DDoS napadi na mrežu. Protokol za zaštitu nazvan je „Proof of Work“ ili „Proof of computational effort“, a osnovni princip rada je izvršavanje kompleksnog zadatka na procesoru računala koje je poslalo zahtjev za nekom uslugom na mreži, kratko opisana ideja rada dostupna je na [6]. Začetnici te ideje su Cynthia Dwork i Moni Naor, a tom idejom su omogućili kreiranje „Reusable Proofs of Work“ dostupnog na [7] protokol koji je kreiran od strane Hal Finney-a 2004. godine baziran na radu Nick-a Szabo-e „Shelling Out: The Origins of Money“ koji je dostupan na [8]. Još kao inspiraciju za kriptovalute bitno je spomenuti rad Wei Dai-a iz 1998. godine „B-Money“ dostupnog na [9] koji opisuje osnovne principe rada današnjih kriptovaluta, te „Bit Gold“, rad Nick-a Szabo-e iz 2005. godine, dostupnog na [10].

Prva prava kriptovaluta nastaje 3.1.2009. godine izlaskom rada „Bitcoin: A Peer-to-Peer Electronic Cash System“, rad je dostupan na [11], od strane Satoshi Nakamoto-a, te rudarenjem Genesis bloka blockchaina Bitcoin-a. Bitcoin koristi Blockchain tehnologiju za spremanje i arhiviranje svih transakcija ikad izvršenih na mreži. Blockchain još možemo nazvati distribuiranom javnom knjigom transakcija. Najveće prednosti kriptovaluta je upravo korištenje

kriptografije javnog-privatnog ključa za sigurnost vlastitih sredstava na mreži, te rješavanje problema Bizantskih generala i duple potrošnje (double spending). Bitcoin-i se generiraju „rudarenjem“, a posao „rudara“ je potvrđivanje i zapisivanje transakcija unutar Blockchain-a. Za „rudarenje“, „rudari“ su nagrađeni novootkrivenim Bitcoin-ima. Osnovne prednosti Bitcoin-a su mogućnost jednostavnog korištenja, niski troškovi transakcija bilo gdje u svijetu, transparentnost transakcija, anonimnost, decentraliziranost, te je ne moguće povratiti transakciju, jer jednom kad se transakcija pošalje na mrežu, nije ju moguće stopirati.

Od pojave Bitcoin-a pa do danas, na tržištu postoji preko 1500 različitih kriptovaluta namijenjene različitim potrebama, prema podacima dostupnim na [12].

2.6. Burze kriptovaluta

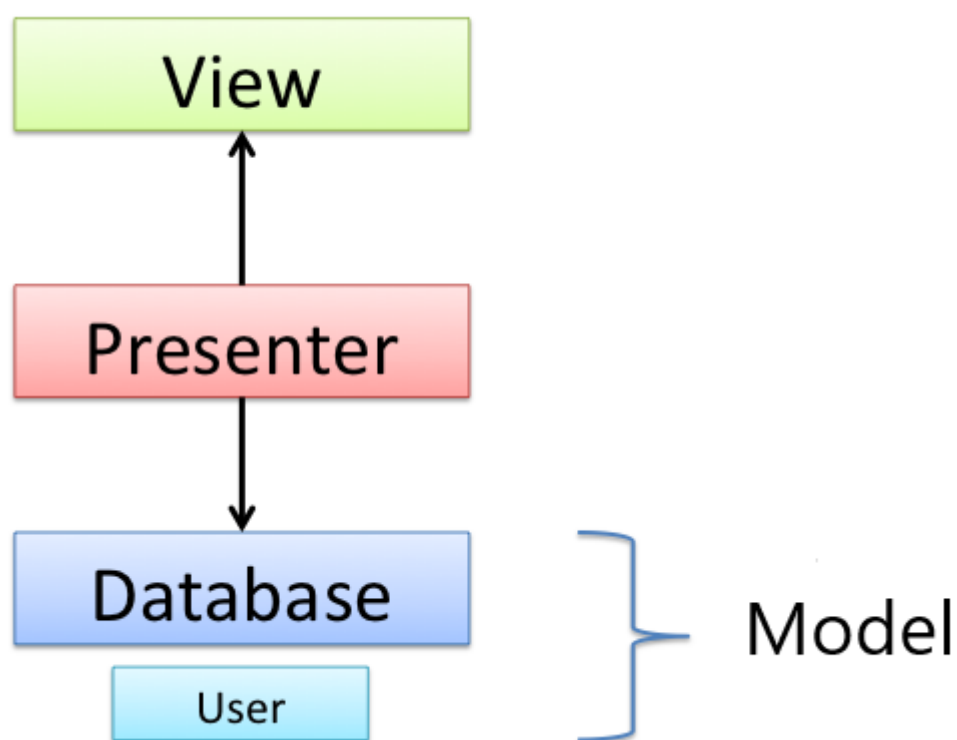
Trgovanje kriptovalutama obično se odvija na raznim burzama specijaliziranim za takvu vrstu razmjene, te za razliku od burzi dionica, burze kriptovaluta rade svaki dan u tjednu, bez prestanka. Postoje dvije kategorije burzi, to su mjenjačnice na kojima se kriptovalute (većinom Bitcoin-i) kupuju korištenjem tradicionalnih valuta i burze na kojima se koristeći kriptovalute trguje drugim kriptovalutama bez korištenja tradicionalnih valuta. Većina burzi trguje isključivo kriptovalutama, dok neke nude i trgovanje tradicionalnim valutama, većinom Dolarom, Eurom i Yenom. Funkcionalnosti koje svaka mjenjačnica mora sadržavati su: mogućnost kupovine i prodaje kriptovaluta, mogućnost skladištenja kriptovaluta i/ili prebacivanja tih istih kriptovaluta na vlastiti „wallet“, te stop limit nalog koji omogućuje kupovinu/prodaju kriptovaluta kada ona dostigne određenu vrijednost na tržištu. Sve navedene mogućnosti osim skladištenja i prebacivanja kriptovaluta, nudi brzorastuća američka kompanija Robinhood, njene prednosti su ne plaćanje za sudjelovanje na tržištu (commission) i jednostavnost. Najveća i najkorištenija mjenjačnica za kriptovalute je Coinbase nastala 2012. godine. Radi svog jednostavnog grafičkog sučelja, mobilne aplikacije i mogućnosti kupovine kriptovaluta sa tradicionalnim valutama, postala je najveća mjenjačnica s 13 300 000 registriranih korisnika, izvor broja korisnika je dostupan na [13]. Među burzama kriptovaluta, najpoznatije u svijetu su Bitstamp, Bitmex, Poloniex, Binance i Bitfinex, dok ih u svijetu postoji 210, zaključno sa 17.06.2018. godine, preuzeto sa [14]. Većina velikih burzi kriptovaluta također podržava trgovanje putem mobilnih aplikacija, što korisniku omogućuje stalno praćenje i trgovanje.

3. MOBILNA APLIKACIJA ZA TRGOVANJE KRIPTOVALUTAMA

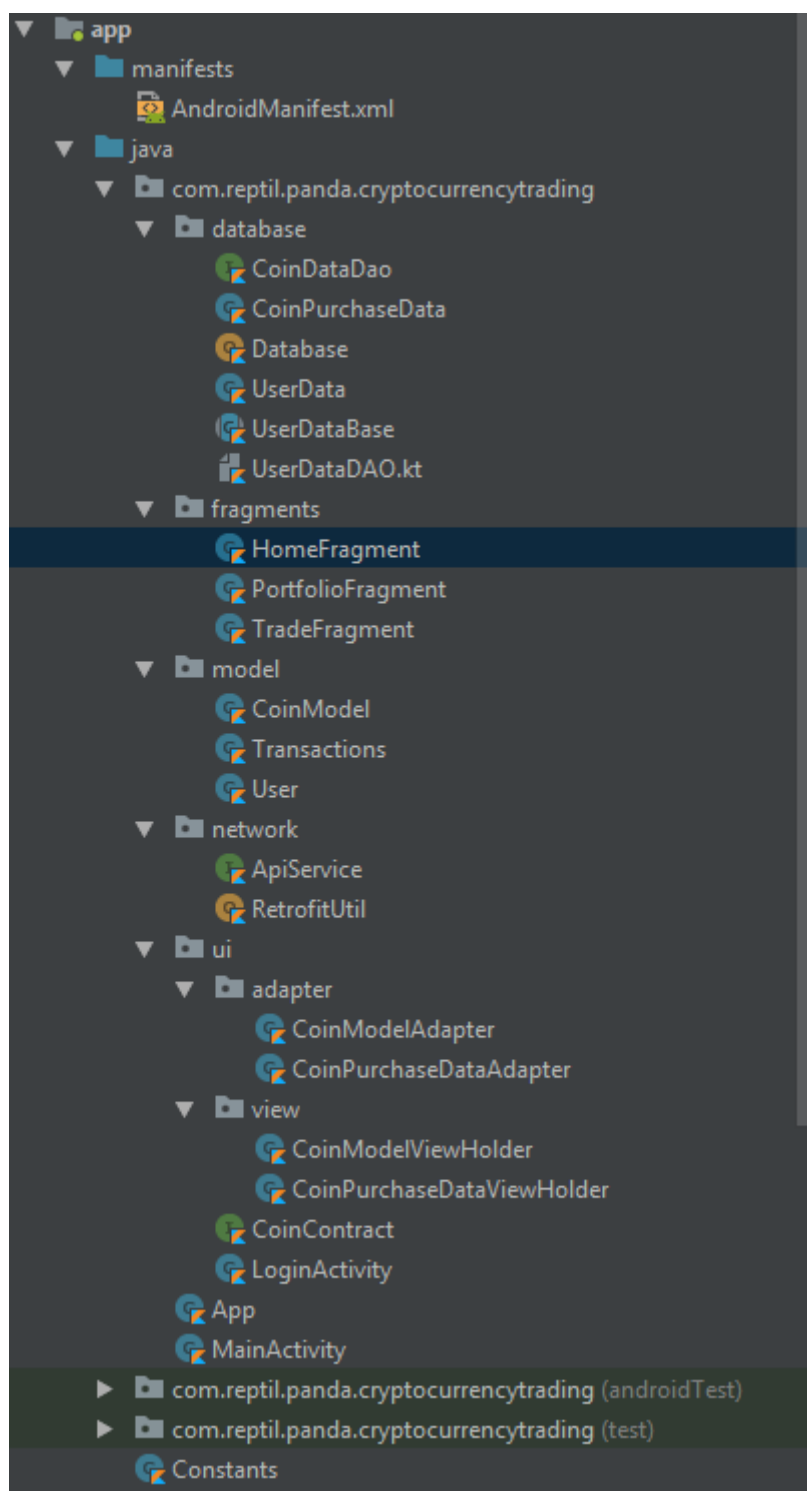
U ovom segmentu pisat će se o arhitekturi aplikacije, potrebnim funkcionalnostima aplikacije kako bi korisniku omogućila osnovne mogućnosti trgovanja na burzi, pregledu portfolia, te prosječnih cijena kriptovaluta preuzetih sa coinmarketcap.com stranice koristeći API servis.

3.1. MVP

MVP ili model-view-presenter je oblikovni obrazac namjenjen Android aplikacijama. Koristi se za strukturiranje koda radi lakšeg uređivanja.



Slika 2. Model View Presenter obrazac, slika prevedena i preuzeta sa [15]

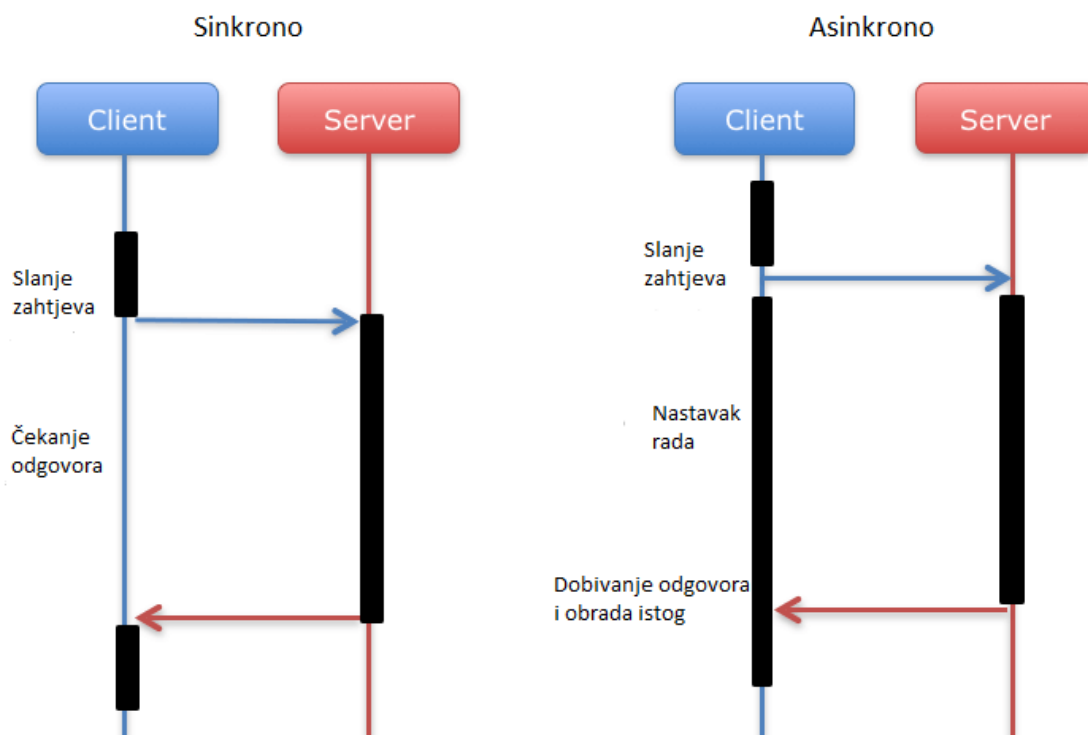


Slika 3. Struktura direktorija gotovog rada

3.2. API

API ili aplikacijsko programsko sučelje je definirano programersko sučelje koje omogućava korisnicima pristup uslugama i servisima napisanih od strane drugih programera ili kompanija. API je skup mrežnih pravila koja opisuju komunikaciju između korisnika i računala,

aplikacije ili servera, a najčešće dolazi u obliku biblioteke. Postoji nekoliko različitih tipova API-a, ali najkorišteniji su WebSocket API i RESTful (Representational State Transfer) API. Postoji više metoda za rad s aplikacijskim programskim sučeljem, no u ovom radu će se koristiti GET metoda za dohvaćanje podataka sa servera. API servis ima mogućnost rada na dva načina: sinkrono i asinkrono. Asinkroni za razliku od sinkronog tijekom čekanja odgovora poslužitelja nastavlja s radom, dok sinkroni čeka odgovor poslužitelja i nakon odgovora nastavlja raditi.



Slika 4. Razlika sinkronog i asinkronog modela rada zahtjeva, slika prevedena i preuzeta sa [16]

Za početak, kako bi uopće postojala ikakva funkcionalnost aplikacije, potrebno je spojiti se na API servis coinmarketcap.com stranice. U ovom slučaju mobilna aplikacija je klijent, do je web server coinmarketcap.com poslužitelj. Klijent šalje GET poziv za dohvaćanje podataka poslužitelju, poslužitelj prima zahtjev, te klijentu šalje informacije. Za dohvaćanje podataka koristi se GET metoda unutar Retrofit biblioteke, čija je dokumentacija dostupna na [17]. Retrofit koristi HTTP API, te omogućava korištenje navedenog API-ja unutar Androida. API servis ima mogućnost rada na dva načina: sinkrono i asinkrono. Asinkroni za razliku od sinkronog tijekom čekanja odgovora poslužitelja nastavlja s radom, dok sinkroni čeka odgovor poslužitelja i nakon odgovora nastavlja raditi.

Kako bi koristili biblioteku Retrofit potrebno je kreirati klase ApiService i RetrofitUtil koje dohvaćaju podatke sa servera CoinMarketCap.

Klasa ApiService.kt:

```
1. interface ApiService {
2.
3.     @GET("/v1/ticker/?start=0&limit=10")
4.     fun getCoinData(): Flowable<List<CoinModel>>
5.
6.     @GET("/v1/ticker/{id}")
7.     fun getCoin(@Path("id") coinId: String): Flowable<List<CoinModel>>
8. }
```

Klasa RetrofitUtil.kt:

```
1. object RetrofitUtil {
2.
3.     private val retrofit by lazy { createRetrofit() }
4.
5.     val apiService: ApiService by lazy { retrofit.create(ApiService::class.java) }
6.
7.     private fun createRetrofit(): Retrofit {
8.         return Retrofit.Builder()
9.             .baseUrl(Constants.COIN_BASE_URL)
10.            .addConverterFactory(GsonConverterFactory.create())
11.            .addCallAdapterFactory(RxJava2CallAdapterFactory.create())
12.            .build()
13.    }
14. }
```

Klasa Constants.kt:

```
1. const val COIN_BASE_URL = "https://api.coinmarketcap.com"
```

Gore navedeni kod opisuje dohvaćanje 10 najvećih kriptovaluta sa servisa, a to je opisano unutar GET naredbe. „start“ označava početak liste, a „limit“ količinu kriptovaluta okja se dohvaća, poredane po ukupnoj vrijednosti na tržištu (Market cap).

```
@GET("/v1/ticker/?start=0&limit=10")
```

Varijabla „COIN_BASE_URL“ čuva glavnu adresu servera koja se koristi unutar klase „RetrofitUtil.kt“, dok se putanja na kojoj se nalaze potrebne informacije stavlja unutar „GET“ naredbe.

Odgovor servera dobivamo u JSON (JavaScript Object Notation) obliku, te ga je potrebno parsirati. U dolje navedenom kodu, prikazat ćemo primjer dolazne informacije korištenjem gore navedenog koda.

```
1. {
2.     "id": "bitcoin",
3.     "name": "Bitcoin",
4.     "symbol": "BTC",
5.     "rank": "1",
6.     "price_usd": "7036.03005894",
7.     "price_btc": "1.0",
8.     "24h_volume_usd": "4581314548.84",
9.     "market_cap_usd": "121275737040",
10.    "available_supply": "17236387.0",
11.    "total_supply": "17236387.0",
12.    "max_supply": "21000000.0",
13.    "percent_change_1h": "-0.3",
14.    "percent_change_24h": "4.11",
15.    "percent_change_7d": "9.03",
16.    "last_updated": "1535475026"
17. }
```

Kako bi se omogućilo korištenje varijabli preuzetih sa servisa, potrebno ih je napraviti klasu koja služi kao model, odnosno napraviti parsiranje. Uloga parsiranja je deserijalizacija podatka u nativni kod. Kod parsiranja potrebno je paziti na upotrebu pravilnog tipa podatka za varijable, kao što je navedeno u dolje dostupnom kodu.

```
1. class CoinModel {
2.
3.     @SerializedName("id")
4.     val id: String? = null
5.     @SerializedName("name")
6.     val name: String? = null
7.     @SerializedName("symbol")
8.     val symbol: String? = null
9.     @SerializedName("price_usd")
10.    val priceUsd: Double? = null
11.    @SerializedName("24h_volume_usd")
12.    val volume_24h: String? = null
13.    @SerializedName("market_cap_usd")
14.    val market_cap_usd: String? = null
15.    @SerializedName("percent_change_1h")
16.    val percent_change_1h: String? = null
17.    @SerializedName("percent_change_24h")
18.    val percent_change_24h: String? = null
19.    @SerializedName("percent_change_7d")
20.    val percent_change_7d: String? = null
21.
22. }
```

Pomoću navedenih kodova, omogućeno nam je korištenje API servisa, te rada s dostupnim podacima. Kako bi pozivali varijable dobivene pomoću API servisa, koristimo parsirane varijable. Primjerice za cijenu u dolarima koristimo varijablu „priceUsd“.

3.3. Baza podataka

Kako bi omogućili korisniku da koristi mogućnosti kupovine i prodaje kriptovaluta, te pregled vlastitog portfolia, potrebno je implementirati bazu podataka. Baza podataka implementirana je pomoću biblioteke Room koja koristi SQL naredbe za rad.

Za početak potrebno je kreirati tablicu pod nazivom „userData“ koja sadržava informacije o korisniku. Osnovne informacije potrebne za rad su „id“ korisnika, odnosno identifikacija, te „currentFiatAmount“ odnosno trenutna raspoloživa količina novaca iskazana u FIAT valuti dolaru.

```
1. @Entity(tableName = "userData")
2. data class UserData(@PrimaryKey(autoGenerate = true) var id: String = "",
3.                    @ColumnInfo(name = "USD Amount") var currentFiatAmount: Double = 10
                        000.0)
```

Kako bi smo dobili informacije o korisniku, koristimo funkciju „getUserDao“.

```
1. fun getUserDao() = database.userDataDao()
```

Osim tablice sa podacima korisnika, potrebno je kreirati još jednu tablicu pod nazivom „coinData“ koja sadržava „id“ odnosno identifikaciju korisnika, „coinId“ odnosno identifikaciju kriptovalute, „currentPrice“ koji nam označava trenutnu cijenu kriptovalute u američkom Dollar-u, „amount“ koji nam označava količinu kriptovalute, te „timestamp“ odnosno vrijeme transakcije.

```
1. @Entity(tableName = "coinData")
2. data class CoinPurchaseData(@PrimaryKey(autoGenerate = true) var id: Long? = null,
3.                             var coinId: String? = null,
4.                             var userId: String? = null,
5.                             @ColumnInfo(name = "USD price") var currentPrice: Double =
                        0.0,
6.                             @ColumnInfo(name = "Amount") var amount: Double = 0.0,
7.                             var timestamp: Long = 0)
```

Kako bi smo dobili informacije o transakciji, koristimo funkciju „getCoinDao“.

```
1. fun getCoinDao() = database.coinDataDao()
```

3.4. Kupovina i prodaja

Za kupovinu i prodaju kriptovaluta potrebno je korisniku dodijeliti novčana sredstva kako bi mogao sudjelovati na tržištu. Burze imaju definirane cijene trgovanja, no unutar ove simulacija cijena trgovanja ne postoji, odnosno koristimo cijene dobivene sa API-ja koji je opisan u prethodnom poglavlju. Aplikacija prilikom pokretanja kreira novog korisnika te mu dodjeljuje 10000\$ za trgovanje.

```
1. val userDao = Database.getUserDao()
2.
```

```

3.         val users = userDao.getAll()
4.
5.         if (users.isEmpty()) {
6.             userDao.insert(userData = UserData(UUID.randomUUID().toString(), 10000.0))
7.
8.             val intent = Intent(this, MainActivity::class.java)
9.
10.            startActivity(intent)
11.        }

```

Prilikom pokretanja aplikacije provjera se postoji li već korisnik unutar memorije mobilnog telefona, ako korisnik ne postoji, kreira se novi korisnik te mu se dodjeljuju gore navedena sredstva.

U slučaju da korisnik već postoji, aplikacija prelazi na „MainActivity.kt“, odnosno glavni prozor aplikacije.

```

1.  if (users.isNotEmpty()){
2.
3.      val intent = Intent(this, MainActivity::class.java)
4.
5.      startActivity(intent)
6.  }

```

Kupovina kriptovaluta je moguća ako korisnik ima dovoljno sredstava na „računu“ kako bi mogao izvršiti transakciju.

```

1.  private fun buyCoin(coinModel: CoinModel, amount: Double, coinName: String) {
2.      val user = userDao.getAll().first()
3.
4.      val newPurchase = CoinPurchaseData(coinId = coinModel.id,
5.          userId = user.id,
6.          coinName = coinName,
7.          amount = amount,
8.          currentPrice = coinModel.priceUsd?.toDouble() ?: 0.0,
9.          timestamp = System.currentTimeMillis()
10.      )
11.
12.      coinDataDao.storePurchase(newPurchase)
13.  }

```

Funkcija „buyCoin“ koristi se za kupovanje kriptovaluta korištenjem sredstava na računu korisnika. Klasa sadrži identifikaciju kriptovalute, identifikaciju korisnika, količinu kriptovalute i trenutnu cijenu, te vrijeme transakcije.

Funkcija „buyCoin“ pokreće se unutar funkcije „onBuyClicked“ kojom pokrećemo kupovinu.

```

1.  private fun onBuyClicked() {
2.      val coin = coinName.text.toString()
3.      val amount = tradeAmount.text.toString().toDouble()
4.      val tradePrice = tradePrice.text.toString()
5.
6.      if (!tradePrice.isBlank()) {

```

```

7.         val user = userDao.getAll().first()
8.         val totalCost = tradePrice.toDouble() * amount
9.
10.
11.         if (user.currentFiatAmount > totalCost) {
12.             user.currentFiatAmount = user.currentFiatAmount - totalCost
13.             userDao.insert(user)
14.
15.             currentCoin?.run { buyCoin(this, amount, coin) }
16.         }
17.     }
18. }

```

Kako bi se trgovina odvila, potrebno je zadovoljiti količinu novaca koju korisnik posjeduje, odnosno količina novaca koju korisnik posjeduje mora biti veća od cijene same transakcije. Ako je uvjet ispunjen, korisniku se omogućuje kupovina zadane količine kriptovaluta, u protivnom korisniku se ispisuje poruka na ekran da nije u mogućnosti kupiti navedenu količinu kriptovaluta, odnosno da nema dovoljno novčanih sredstava na računu kako bi izvršio transakciju. U gore navedenom kod imamo primjer provjere : „user.currentFiatAmount > totalCost“, uz to da je „totalCost“ jednak „tradePrice“ koji u ovom dijelu označava cijenu kriptovalute, pomnožen sa „amount“ odnosno količinom kriptovalute. Oduzimanje sredstava sa korisnikovog računa opisano je jednadžnom : „user.currentFiatAmount = user.currentFiatAmount – totalCost“ .

Prodaja kriptovaluta je moguća uz provjeru stanja računa korisnika, ako korisnik ima količinu željnu za prodaju na svome računu omogućuje mu se prodaja kriptovaluta, uz to da je potrebno platiti troškove trgovanja na burzi. Troškovi trgovanja unutar ove simulacije ne postoje.

```

1. private fun onSellClicked() {
2.     val coin = coinName.text.toString()
3.     val amount = tradeAmount.text.toString().toDouble()
4.     val tradePrice = tradePrice.text.toString()
5.
6.     if (!tradePrice.isBlank()) {
7.         val user = userDao.getAll().first()
8.         val myCoin = coinDataDao.getCoinsForUser(user.id).firstOrNull { it.coinId =
= coin }
9.         ?: return
10.
11.         if (myCoin.amount >= amount) {
12.             val totalCost = tradePrice.toDouble() * amount
13.
14.             user.currentFiatAmount += totalCost
15.             userDao.insert(user)
16.
17.             myCoin.amount -= amount
18.
19.             coinDataDao.storePurchase(myCoin)
20.         }
21.     }
22. }

```

U gore navedenom kodu, vidimo da postoji provjera količine kriptovaluta : „myCoin.amount >= amount), bez koje nebi bilo moguće izvršiti samu transakciju, te nakon izvršenja transakcije, količina novaca dobivena od prodaje, dodaje se na račun korisnika, a količina kriptovaluta se oduzima.

3.5. Portfolio

Nakon što je korisnik u mogućnosti kupiti i prodati određenu kriptovalutu, potrebno mu je omogućiti pregled stanja svih kriptovaluta koje posjeduje. Portfolio sadrži trenutno količinu sredstva izraženu u američkim dolarima, naziv i količinu kriptovaluta. Kako bi koristili portfolio, potrebno je unutar kreirati model portfolia, za to je korištena biblioteka Room.

```
1. @Entity(tableName = "coinData")
2. data class CoinPurchaseData(@PrimaryKey(autoGenerate = true) var id: Long? = null,
3.                             var coinId: String? = null,
4.                             var coinName: String? = null,
5.                             var userId: Long = 0,
6.                             @ColumnInfo(name = "USD price") var currentPrice: Double =
7.                             0.0,
8.                             @ColumnInfo(name = "Amount") var amount: Double = 0.0,
9.                             var timestamp: Long = 0)
```

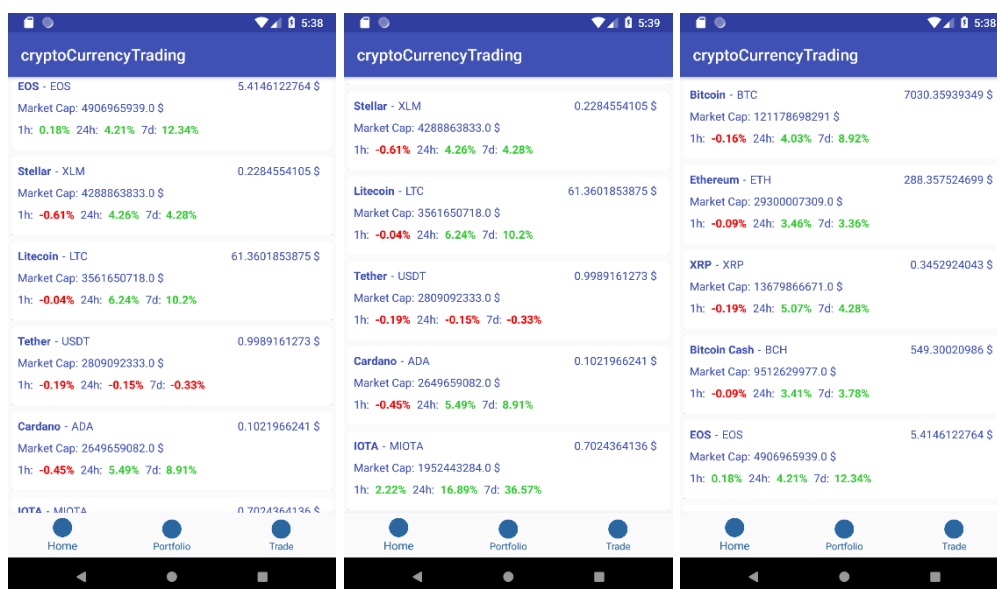
4. TESTIRANJE I USPOREDBA S VEĆ POSTOJEĆIM RJEŠENJIMA

U ovom segmentu, obratit ćemo pažnju na već postojeća rješenja prisutna na tržištu te njihove mogućnosti, nedostatke i moguće nadogradnje već postojećih sustava. Od postojećih rješenja na tržištu koja trenutno rade, jedna aplikacija nudi simulaciju trgovanja kriptovalutama, TradeHero. Osim nje postoje brojne aplikacije koje nude mogućnosti trgovanja dionicama, a jedna od najpoznatijih i najkorištenijih je WallStreet Survivor. Uz prethodno dvije nabrojane, spomenut ćemo i Coinbase mobilnu aplikaciju kao primjer jednostavne i brze aplikacije za sudjelovanje na stvarnom tržištu.

4.1. Testiranje aplikacije

Testiranje vizualnom dijela aplikacije, prikaz podataka prikupljenih sa API servisa, vizulani prikaz portfelja iz baze podataka korisnika, te vizualni prikaz trgovanja, odnosno uspješnosti trgovanja.

Prvi pogled aplikacije prikazuje informacije o kriptovalutama. Korištenjem API servisa, prikupili smo podatke o nazivu, kratici, cijeni, vremenskoj promjeni, te ukupnoj tržišnoj vrijednosti deset najvećih kriptovaluta.

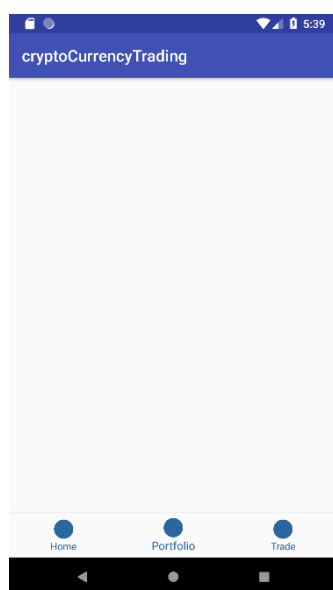


cryptoCurrencyTrading	cryptoCurrencyTrading	cryptoCurrencyTrading
EOS - EOS Market Cap: 4906965939.0 \$ 1h: 0.18% 24h: 4.21% 7d: 12.34%	Stellar - XLM Market Cap: 4288863833.0 \$ 1h: -0.61% 24h: 4.26% 7d: 4.28%	Bitcoin - BTC Market Cap: 121178698291 \$ 1h: -0.16% 24h: 4.03% 7d: 8.92%
Stellar - XLM Market Cap: 4288863833.0 \$ 1h: -0.61% 24h: 4.26% 7d: 4.28%	Litecoin - LTC Market Cap: 3561650718.0 \$ 1h: -0.04% 24h: 6.24% 7d: 10.2%	Ethereum - ETH Market Cap: 29300007309.0 \$ 1h: -0.09% 24h: 3.46% 7d: 3.36%
Litecoin - LTC Market Cap: 3561650718.0 \$ 1h: -0.04% 24h: 6.24% 7d: 10.2%	Tether - USDT Market Cap: 2809092333.0 \$ 1h: -0.19% 24h: -0.15% 7d: -0.33%	XRP - XRP Market Cap: 13679866671.0 \$ 1h: -0.19% 24h: 5.07% 7d: 4.28%
Tether - USDT Market Cap: 2809092333.0 \$ 1h: -0.19% 24h: -0.15% 7d: -0.33%	Cardano - ADA Market Cap: 2649659082.0 \$ 1h: -0.45% 24h: 5.49% 7d: 8.91%	Bitcoin Cash - BCH Market Cap: 9512629977.0 \$ 1h: -0.09% 24h: 3.41% 7d: 3.78%
Cardano - ADA Market Cap: 2649659082.0 \$ 1h: -0.45% 24h: 5.49% 7d: 8.91%	IOTA - MIOTA Market Cap: 1952443284.0 \$ 1h: 2.22% 24h: 16.89% 7d: 36.57%	EOS - EOS Market Cap: 4906965939.0 \$ 1h: 0.18% 24h: 4.21% 7d: 12.34%
IOTA - MIOTA Market Cap: 1952443284.0 \$ 1h: 2.22% 24h: 16.89% 7d: 36.57%		

Slike 5., 6., 7. Prikaz podataka prikupljenih sa API servisa

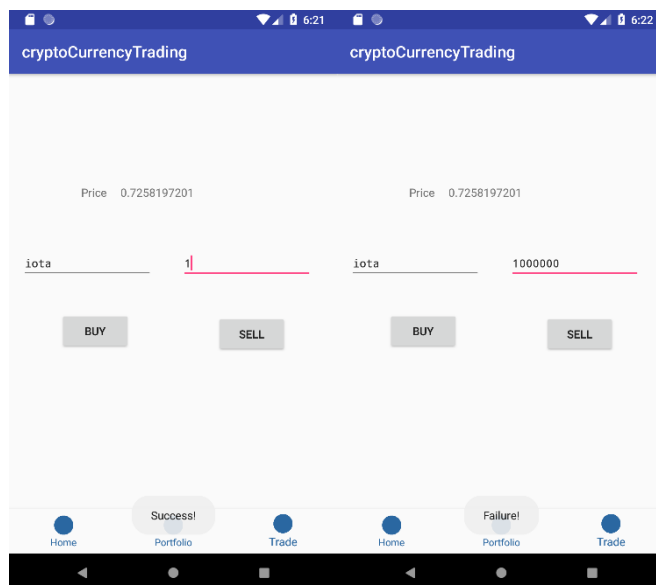
U gore navedenim slikama možemo vidjeti uspješno prikupljene i prikazane podatke, te obojane vremenske promjene cijena kriptovaluta. Zelena boja označava pozitivnu promjenu cijene, dok crvena boja označava negativnu promjenu cijene.

Dugi pogled aplikacije prikazuje nam portfelj korisnika preuzet iz baze podataka. Pogled sadrži trenutnu količinu virtualnog novca u vlasništvu korisnika, te nazive, količinu i trenutnu vrijednost kriptovaluta.



Slika 8. Prikaz podataka portfelja preuzetih iz baze podataka

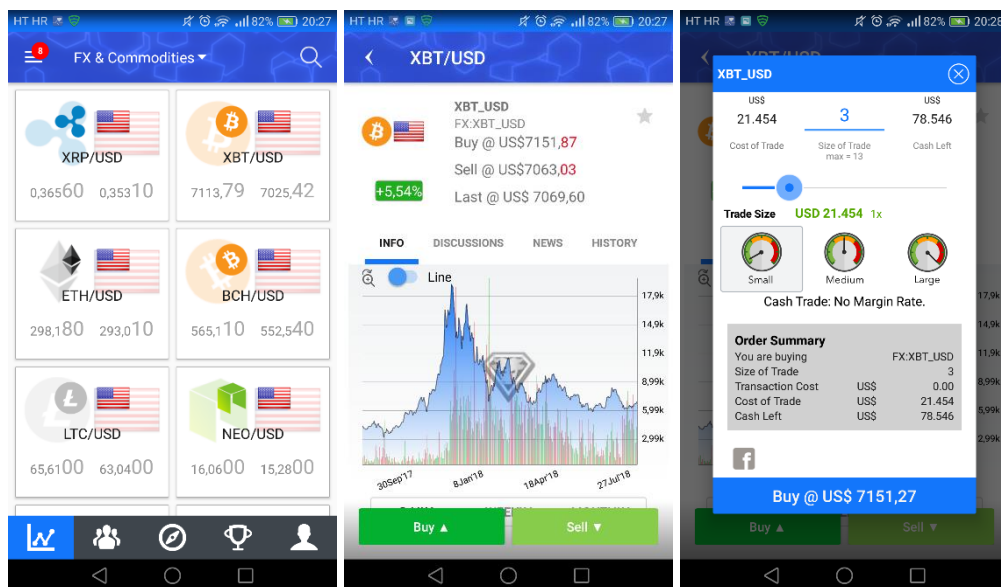
Treći pogled aplikacije prikazuje nam sučelje za trgovanje. Sučelje za trgovanje omogućava korisniku unos naziva kriptovalute i količine kriptovalute koju želi kupiti ili prodati koristeći gumbove „BUY“ ili „SELL“. Uz to sučelje prikazuje uspjeh ili neuspjeh transakcije u obliku poruka „Success!“ ili „Failure“.



Slike 9., 10. Prikaz sučelja trgovanja, te uspjeha odnosno neuspjeha transakcije

4.2. TradeHero

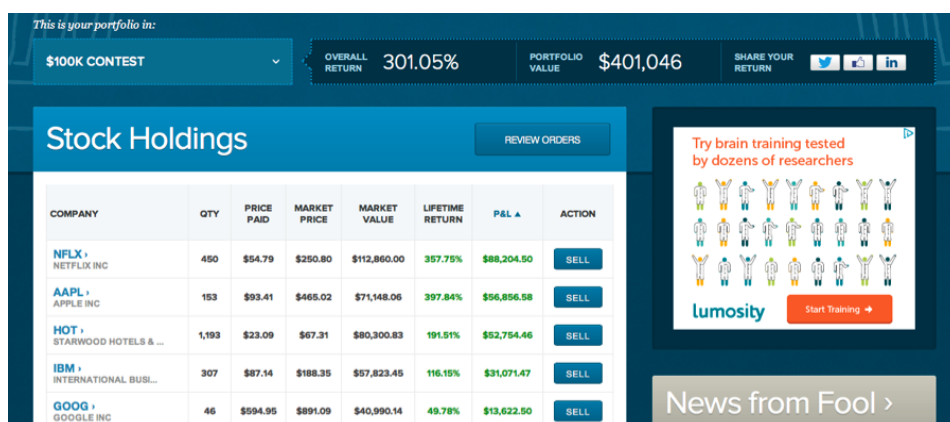
TradeHero je mobilna aplikacija namjena simulaciji trgovanja dionicama, resursima i određenim kriptovalutama. Aplikacije ima veliki broj funkcionalnosti poput edukacijskog segmenta, natjecateljskog segmenta, grafova i svijeća cijena resursa. TradeHero vrlo je jednostavna aplikacija za korištenje s puno mogućnosti. Najveći nedostatak aplikacije je ne mogućnost trgovanja kriptovalutama u vrijeme kada je burza zatvorena, što se protivi osnovnim principima rada mjenjačnica odnosno burzi kriptovaluta.



Slike 11., 12., 13. Prikaz grafičkog sučelja aplikacije TradeHero

4.3. WallStreet Survivor

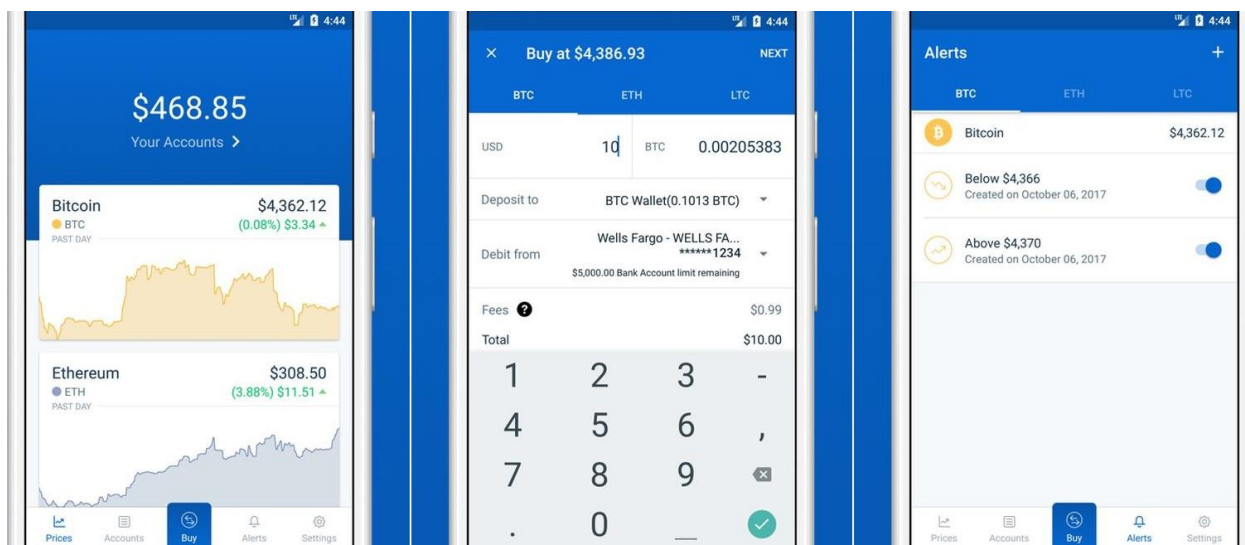
WallStreet Survivor iako ne nudi mogućnost simulacije trgovanja kriptovalutama nudi mogućnost simulacije trgovanja dionicama na WallStreet burzi. Iako ne postoji zasebna mobilna aplikacija, web aplikacija prilagođena je korištenju na mobilnom uređaju. Aplikacija ima vrlo jednostavno i brzo korisničko sučelje uz veliki niz funkcionalnosti i mogućnosti. Nudi pregled portfolia, svih dosad odrađenih trgovanja, pregled tržišta i natjecanje u različitim vremenskim periodima i ligama. Osim simulaciji, aplikacija ima poseban dio posvećen edukaciji korisnika. Svrha same aplikacije, kako navode kreatori aplikacije je edukacija.



Slika 14. Prikaz grafičkog sučelja portfelja aplikacije Wall Street Survivor, slika preuzeta sa [19]

4.4. Coinbase

Iako Coinbase nije aplikacija namijenjena trgovanja kriptovaluta, već mjenjačnica kriptovaluta, važno ju je spomenuti kao aplikaciju s najjednostavnijim i najbržim grafičkim sustavom trenutno na tržištu. Aplikacija je namjena svim korisnicima kojima je potrebna jednostavnost i brzina pri korištenju mjenjačnice. Ne posjeduje određene funkcionalnosti kao burza, no to nije niti njena namjena.



Slika 14. Grafički prikaz Coinbase mobilne aplikacije slika uređena i preuzeta sa [18]

5. ZAKLJUČAK

U radu su opisane tehnologije korištene za dizajniranje i implementiranje simulatora trgovanja kriptovaluta. Aplikacija je namijenjena korisnicima Android operacijskog sustava, te većinom mlađim osobama. Tijekom izrade rada, bilo je potrebno obratiti pažnju na jednostavnost korištenja aplikacije. Tehnologije korištene za izradu završnoga rada su industrijske poznate i većinom najčešće korištene tehnologije, što je rezultiralo vrlo velikom količinom potrebnih informacija za implementaciju.

Prednosti korištenja ovakve aplikacije su edukacijske prirode. Aplikacija ima svrhu naučiti korisnika osnovnim funkcionalnostima trgovanja, te principe ponude i potražnje na tržištu koji rezultiraju cijenom resursa, u ovom slučaju 10 najvećih kriptovaluta. Kao i što je navedeno, aplikacija je najviše usmjerena mlađim generacijama, srednjoškolcima, zbog jednostavnosti i ne mogućnosti vlastitog sudjelovanja na raznim tržištima.

Tijekom izrade rada bilo je potrebno naučiti osnovne principe rada Android operacijskog sustava i sam grafički dizajn sustava i aplikacija, programskog jezika Kotlin, Model View Presenter obrasca, kreiranje baze podataka i rad sa istom, te pozivanje i korištenje APIa. Najviše pažnje posvećeno je kreiranju i korištenju baze podataka, te proučavanju načina rada i implementaciji API servisa unutar Android operacijskog sustava. Osim API servisa i baze podataka, puno pažnje posvećeno je proučavanju rada Android operacijskog sustava kao platforme za rad aplikacija i MVP obrasca. Uz to, vrlo širok spektar mogućih biblioteka za korištenja kod rješavanja određenog problema oduzima veliku količinu vremena programerima koji se upoznavanju s platformom. Problema s osnovnim funkcionalnostima rada trgovanja na burzama nije bilo zbog ranije upoznatosti s tim istim segmentima.

Dodatne mogućnosti aplikacije koje nisu uvedene radi kompleksnosti rješenja edukacijski videi ili pdf-ovi, ljestvica korisnika po zaradi u određenom vremenskom razdoblju i mogućnost pregleda stanja korisnika izuzevši odabrane transakcije. Edukacijski videi ili tekstovi donjeli bi korisniku osnovne informacije o načinu rada same burze i trgovanja. Rangiranje korisnika po zaradi u određenom vremenskom periodu (tjedno, mjesečno, godišnje) omogućilo bi korisnicima natjecanje, te želju za većim znanjem o tržištu i događajima koji utječu na same cijene. Osim globalnog rangiranja korisnika, bilo bi moguće kreirati grupe korisnika (primjer: prijatelji, obitelj) koji se međusobno natječu. Koncept mogućnost pregleda stanja korisnika izuzevši odabrane transakcije izvršavao bi se na način da korisnik po odabiru izolira određenu transakciju u povijesti

svojih transakcija, te dobije vrijednosno stanja koje bi imao u danom trenutku da nije izvršio odabranu transakciju.

LITERATURA

- [1] StatCounter, [online], StatCounter, 24. lipnja 2018. dostupno na:
<http://gs.statcounter.com/os-market-share/mobile/worldwide> [24. lipnja 2018.]
- [2] Kotlinlang, [online], JetBrains, 14. lipnja 2018. dostupno na:
<https://kotlinlang.org/docs/reference/faq.html> [24. lipnja 2018.]
- [3] Fortune, [online], Meredith Corporation, 24. lipnja 2018. dostupno na:
<http://fortune.com/2018/05/10/robinhood-users-trading-app-tops-ettrade> [24. lipnja 2018.]
- [4] Enterprise Singapore, [online], Enterprise Singapore, 24. lipnja 2018. dostupno na:
<https://ie.enterprisesg.gov.sg/Venture-Overseas/SgGoesGlobal/TradeHero> [24. lipnja 2018.]
- [5] WallStreet Survivor [online], Stok-Trak, 24. lipnja 2018. dostupno na:
<http://www.wallstreetsurvivor.com/> [24. lipnja 2018.]
- [6] Wikimedia Foundation, [online] , Wikimedia Foundation, 24. lipnja 2018. dostupno na:
https://en.wikipedia.org/wiki/Proof-of-work_system [24. lipnja 2018.]
- [7] Satoshi Nakamoto Institute, [online], Satoshi Nakamoto Institute, 2004. dostupno na:
<https://nakamotoinstitute.org/finney/rpow/> [24. lipnja 2018.]
- [8] Nick Szabo, Shelling Out: The Origins of Money, 2002. dostupno na:
<https://nakamotoinstitute.org/shelling-out/> [24. lipnja 2018.]
- [9] Wei Dai, B-Money, 1998. dostupno na:
<http://www.weidai.com/bmoney.txt> [24. lipnja 2018.]
- [10] Nick Szabo, Bit Gold, 29.12.2005. dostupno na:
<https://nakamotoinstitute.org/bit-gold/> [24. lipnja 2018.]
- [11] Satoshi Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, 2008. dostupno na:
<https://bitcoin.org/bitcoin.pdf> [24. lipnja 2018.]
- [12] CoinMarketCap, [online], CoinMarketCap, 24. lipnja 2018. dostupno na:
<https://coinmarketcap.com/all/views/all/> [24. lipnja 2018.]
- [13] Wikimedia Foundation, [online] , Wikimedia Foundation, 24. lipnja 2018. dostupno na:
<https://en.wikipedia.org/wiki/Coinbase> [24. lipnja 2018.]
- [14] CoinMarketCap, [online], CoinMarketCap, 24. lipnja 2018. dostupno na:
<https://coinmarketcap.com/exchanges/volume/24-hour/all/> [24. lipnja 2018.]

- [15] Hannes Dorfman, [online], Hannes Dorfman, 2015. dostupno na:
<http://hannesdormann.com/mosby/mvp/> [23. kolovoza 2018.]
- [16] Smashing Magazine, [online], Smashing Media AG, 8. ožujka 2017., dostupno na:
<https://www.smashingmagazine.com/2017/03/simplify-android-networking-volley-http-library/> [24. lipnja 2018.]
- [17] Square, Inc, [online], Github, 2013. dostupno na:
<http://square.github.io/retrofit/> [24. lipnja 2018.]
- [18] Best Apps Guru, [online], Best Apps Guru, 2018. dostupno na:
<http://bestappsguru.com/best-bitcoin-apps/> [23. kolovoz 2018.]
- [19] Neutron Group, [online], Neutron Group, 2018. dostupno na:
<http://neutrongroup.cachefly.net/wss.blog/portfolioryan.png> [23. kolovoz 2018.]

SAŽETAK

Glavni zadatak rada je bio osmisliti i implementirati Android aplikaciju namijenjenu simulaciji trgovanja kriptovaluta. Kod aplikacije pisan je u programskom jeziku Kotlin, a dizajn u jeziku XML. Pregled cijena kriptovaluta u stvarnom vremenu postignut je spajanjem na coinmarketcap.com API servis uz pomoć Retrofit biblioteke i GET metode. Glavne funkcionalnosti aplikacije su kupovina i prodaja kriptovaluta, te mogućnost pregleda portfolia korisnika. Baza podataka same aplikacije implementirana je uz pomoć Room biblioteke. Uz implementaciju baze podataka, omogućeno je korištenje i pregled portfolia korisnika.

Ključne riječi: kriptovalute, Android, trgovanje, API, Room

ABSTRACT

Cryptocurrency trading simulation

The main task of this thesis was to design and implement Android based simulation for cryptocurrency trading. Main application code was written in Kotlin programming language and design was written in XML. To receive real market prices of cryptocurrency we used coinmarketcap.com API service using Retrofit library and GET method. Main functions of applications are buying and selling cryptocurrencies, and user ability to view portfolio. Database of application has been implemented using Room library. With database implementation, we enabled user to use and view its own portfolio.

Keywords: cryptocurrency, Android, trading, API, Room

ŽIVOTOPIS

Ante Bartulović rođen je 29. prosinca 1994. godine u Zagrebu. OŠ Ivan Mažuranić završava u Vinkovcima u kojima živi sve do upisa na fakultet. U Vukovaru 2009. godine upisuje Matematičku Gimnaziju, maturira 2013. godine, te iste te godine upisuje Elektrotehnički fakultet u Osijeku, današnji Fakultet elektrotehnike, računarstva i informacijskih tehnologija, smjer Računarstvo.