Razvoj web aplikacije za praćenje trendova kriptovaluta

Težak, Filip

Master's thesis / Diplomski rad

2021

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:003102

Rights / Prava: Attribution-NonCommercial 3.0 Unported/Imenovanje-Nekomercijalno 3.0

Download date / Datum preuzimanja: **2024-10-04**

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Repository / Repozitorij:

Faculty of Organization and Informatics - Digital Repository

zir.nsk.hr

DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Filip Težak

RAZVOJ WEB APLIKACIJE ZA PRAĆENJE TRENDOVA KRIPTOVALUTA

DIPLOMSKI RAD

Varaždin, 2021.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Filip Težak

Matični broj: 0016118297

Studij: ***Informacijsko i programsko inženjerstvo***

RAZVOJ WEB APLIKACIJE ZA PRAĆENJE TRENDOVA KRIPTOVALUTA

DIPLOMSKI RAD

Mentor:

dr. sc. Matija Novak

Varaždin, rujan 2021.

Filip Težak

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

U radu je najprije predstavljen pojam kripto valuta, nakon toga je ukratko opisana povijest nastanka kripto valuta te njihova definicija i arhitektura. Nadalje je opisan pojam aplikacijskih programskih sučelja (eng. Application Programing Interface - API), što predstavlja, koje su svrhe korištenja istih, koje vrste postoje te njihov dizajn. Također, predstavljena su i opisana korištena programska sučelja za dohvaćanje podataka o kripto valutama u praktičnom dijelu rada. Kao praktičan dio rada izrađena je vlastita web aplikacije za praćenje trendova kripto valuta naziva PinOpt. Za izradu pozadinske aplikacije korištena je tehnologija ASP.NET Core Web API, korisničko sučelje je izrađeno u Angularu te je korištena Microsoft SQL Server baza podataka. Aplikacija je isporučena na vlastiti server korištenjem IIS-a.

Ključne riječi: kripto; valuta; kapitalizacija; transakcije; .net; angular; api; rest;

Sadržaj

1. Uvod

Zadnjih nekoliko godina kripto valute su postale jedna od glavnih tema u mnogim područjima, od ekonomije kao najperspektivnija zamjena za klasični novac i trgovinu u budućnosti pa sve do informatike kao zanimljiva nova tehnologija čiji su limiti još neistraženi i čiji potencijal je tek potrebno otkriti. Kao i sa svime što sadrži neku vrijednost pa tako i s kripto valutama čovjek je počeo trgovati. Trgovanje kripto valutama svakim danom sve više raste te samim time raste i potreba za raznim alatima koji olakšavaju praćenje kretanja cijena kripto valuta, odnosno praćenje trendova istih. Upravo to je razlog glavne motivacije za pisanje ovog rada.

Prvi dio ovog rada je teorijski pregled kripto valuta, API-a te korištenih alata i tehnologija. Nakon toga je praktični dio koji sadrži opis vlastite web aplikacije za praćenje trendova kripto valuta.

Izrada rada započela je pretraživanjem dostupne literature kao izvor osnovnih informacija za pisanje rada.

Za potrebe izrade vlastite web aplikacije u praktičnom dijelu rada korišteni su primjeri dobre prakse, službene dokumentacije proizvođača te razni vodiči. Korištene tehnologije su ASP.NET Core za pozadinsku aplikaciju, Angular za korisničko sučelje te Microsoft SQL Server za bazu podataka. Alati korišteni za izradu aplikacije su Visual Studio 2019, Visual Studio Code, SQL Server Management Studio te IIS. Svi navedeni alati su kasnije detaljnije opisani.

2. Kripto valute

Kripto valute su definitivno jedna od najaktualnijih tema otkako se 2008. godine pojavila prva, danas i najpoznatija kripto valuta Bitcoin. Kripto valute su od potpune enigme za prosječnog čovjeka postale toliko česta tema u društvenim krugovima da ih danas i ne treba puno objašnjavati.

Kripto valute su digitalne valute koje se mogu koristiti za kupnju, ali u današnje vrijeme najpopularnije je trgovanje kripto valutama gdje se kripto valute ponašaju gotovo jednako kao i dionice kompanija na klasičnoj burzi. Za takvo trgovanje kripto valutama danas postoji bezbroj različitih mjenjačnica i platformi koje omogućuju upravo to, konverziju jedne valute u drugu kako bi se pravodobnom promjenom ostvarila dobit. Brze i nepredvidive promjene vrijednosti kripto valuta ovakvo trgovanje čine veoma zanimljivim i dinamičnim.

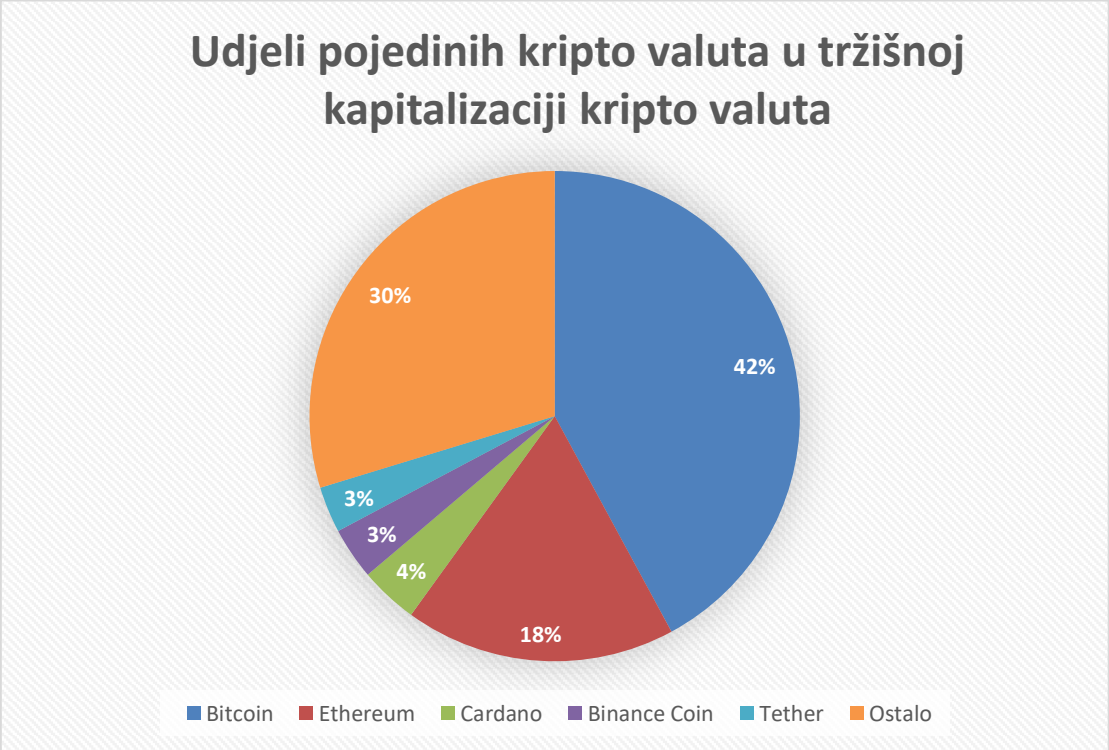
Trgovanje kripto valutama za razliku od klasičnih trgovanja, na primjer dionicama kompanija, privlači mnogo više različitih ljudi radi svoje jednostavnosti i potencijalno mnogo veće zarade. Potrebno je napomenuti kako je u većini slučajeva za ulazak u trgovanje kripto valutama potreban izuzetno mali početni ulog iz razloga što većina kripto valuta danas svoju vrijednost mjeri u dijelovima centa.

2.1. Tržište kripto valuta

U široj javnosti još uvijek vlada mišljenje kako su kripto valute nešto novo i nešto prolazno, ali kada tržište kripto valuta čija tržišna kapitalizacija (eng. market capitalization) u vrijeme pisanja iznosi približno 2 bilijuna američkih dolara usporedimo s vodećim svjetskim kompanijama možemo uočiti kako je tržište kripto valuta jedno od najvrjednijih sustava u povijesti, iako isto postoji svega nešto više od 7 godina. Slika 1 prikazuje usporedbu tržišne kapitalizacije svih kripto valuta i vodećih svjetskih kompanija. [1] [2]

Slika 1 Tržišna kapitalizacija kripto valuta u odnosu na vodeće svjetske kompanije [2]

Slika 2 prikazuje tržišnu kapitalizaciju pojedinih kripto valuta u cijelom tržištu kripto valuta na dan pisanja. Možemo primijetiti kako kripto valute Bitcoin sa 42% te Ethereum sa 18% čine 60% ukupne tržišne kapitalizacije kripto valuta. [3]



Slika 2 Udjeli pojedinih kripto valuta u tržišnoj kapitalizaciji kripto valuta [1]

U cilju da se pokaže koliko je tržište kripto valuta pogodno za trgovanje, ali istovremeno i nepredvidivo, možemo pogledati primjer popularne kripto valute zvane Dodgecoin koja u trenutku pisanja drži 7. mjesto po tržišnoj kapitalizaciji od 2.05% te ima vrijednost 0.3202 američka dolara. Na dan 27. siječnja 2021. godine ova je kripto valuta imala vrijednost od svega 0.007482 američka dolara, a da bi na dan 7. svibnja iste godine vrijednost narasla na 0.6848 američka dolara. [3]

Ukoliko ne obratimo dovoljno pažnje, moglo bi nam se radi malih iznosa učiniti kako to i nije neki rast, ali pretvorimo li ovu promjenu u postotnu promjenu dobivamo rast od otprilike 9150% u svega 100 dana. Nevjerojatnu nepredvidivost ovog tržišta možemo prepoznati u informaciji da je nakon 7. svibnja u naredna 3 dana, dakle 10. svibnja vrijednost ove kripto valute iznosila 0.45 američkih dolara što je pad od 34 % u svega 3 dana. [3]

2.2. Formalna definicija

Kao što navodi Jan Lansky [4], da bi nešto bilo sustav kripto valuta ono mora zadovoljiti svih šest navedenih uvjeta, a to su:

1. Sustav ne zahtjeva središnje tijelo, već se njegovo stanje održava distribuiranim konsenzusom;

2. Sustav vodi zapis o jedinicama kripto valuta i njihovom vlasništvu;

3. Sustav definira mogu li se stvoriti nove jedinice kripto valuta te ako se mogu tada sustav definira okolnosti njihovog podrijetla i način kako utvrditi njihovo vlasništvo;

4. Vlasništvo kripto valute može se dokazati isključivo pomoću kriptografije;

5. Sustav omogućuje izvršavanje transakcija koje mijenjaju vlasništvo nad jedinicama kripto valuta. Izvršavanja takvih transakcija mogu pokrenuti isključivo vlasnici tih jedinica koji mogu dokazati vlasništvo nad njima;

6. U slučaju da se istovremeno pokušaju izvršiti dvije različite transakcije za promjenom vlasništva jedinice kripto valuta, sustav tada izvršava najviše jednu od njih;

2.3. Arhitektura

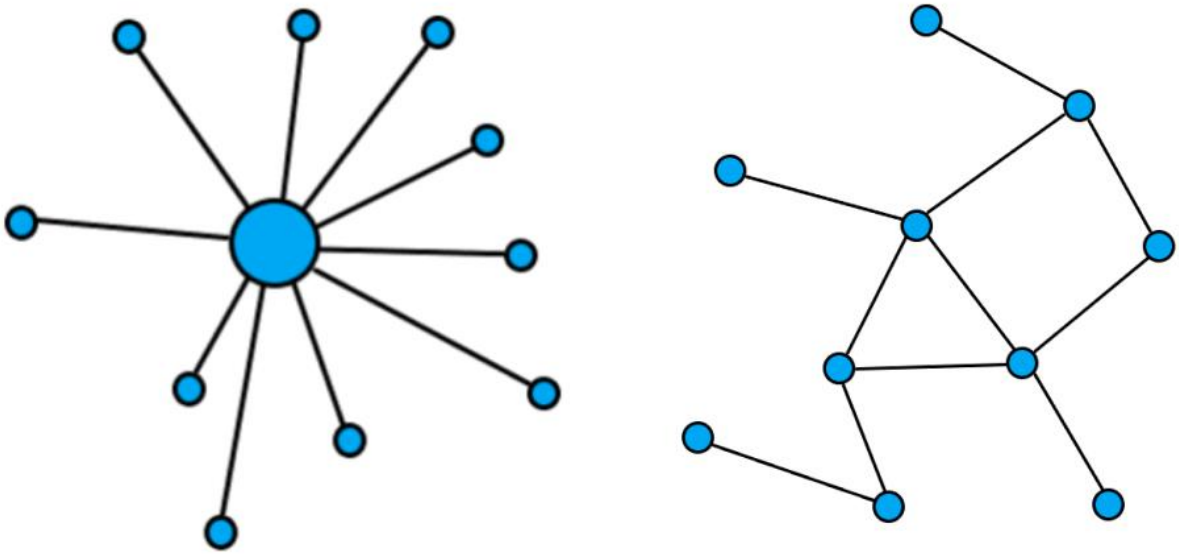
Glavna tehnologija koja kripto valutama omogućuje siguran, pouzdan i decentraliziran rad je lanac blokova (eng. blockchain). Lanac blokova je tehnologija koja se prvi puta spominje 2008. godine u radu pod nazivom „Bitcoin: A Peer-to-Peer Electronic Cash System“ čiji je autor osoba ili grupa osoba pod pseudonimom Satoshi Nakamoto. U tome radu primarno je opisana nova kripto valuta Bitcoin koja u svojem radu koristi sustav lanca blokova kako bi implementirala potrebne osobine jedne kripto valute, a to su sigurnost i transparentnost. Kako u izvornom radu navodi Satoshi Nakamoto, svrha kripto valute je omogućiti provođenje online transakcija između dvije strane bez potrebe za obrađivanjem transakcije od strane neke financijske institucije. [5]

Implementacijom sustava lanca blokova Bitcoin je prva kripto valuta u povijesti koja je uspjela riješiti problem dvostruke potrošnje (eng. Double-spending) bez potrebe za centraliziranim sustavom, što je do tada bio jedan od glavnih problema i razloga zašto nijedna kripto valuta do tada nije zaživjela. Dizajn Bitcoina inspirirao je mnoge druge kripto valute te danas u svijetu postoji nebrojeno mnogo kripto valuta koje rade na jednak način.

Lanac blokova možemo reprezentirati kao neprekidno rastuću listu zapisa koje još nazivamo i blokovima koji su povezani i osigurani koristeći kriptografiju. [6] [7]

Za razliku od standardnog financijskog sustava koji je centraliziran, odnosno sve transakcije i sav novac mora proći kroz postupak verifikacije u nekoj od financijskih središnjica kao što je centralna banka, lanac blokova je decentraliziran.

Na slici 3 možemo vidjeti prikaz standardnog financijskog sustava koji je centraliziran oko jednog centralnog čvorova, a možemo vidjeti i prikaz sustava lanca blokova kod kojeg svaki korisnik predstavlja zaseban čvor te ne postoji potreba za jednim nadređenim tijelom koje će nadgledati sve transakcije i često za to uzimati određenu proviziju. Decentralizacijom sustava postiže se veća transparentnost i sigurnost jer za promjenu podataka nije moguće podatak promijeniti na samo jednom mjestu već ga je potrebno promijeniti na svim mjestima u istom trenutku. [8]



Slika 3 Dijagram centraliziranog [lijevo] i decentraliziranog [desno] sustava [autorski rad]

3. Razvoj web aplikacija

Stolne aplikacije su najstarija vrsta aplikacija koje se pokreću lokalno na računalu korisnika gdje se obavlja pozadinski rad aplikacije te se korisniku prikazuje rezultat odnosno informacija. U početku uporabe računala korisniku je kao jedina opcija komunikacije s računalom bila komandna linija te su se kasnije pojavila grafička sučelja aplikacija. [9]

Za razliku od stolnih aplikacija, web aplikacije se pokreću preko mreže, odnosno na poslužitelju, a zatim se rezultat, odnosno informacija prikazuje na ekranu korisnika putem internetskog preglednika. Pojavom web-a (World Wide Web - WWW) pojavile su se i prve web aplikacije, odnosno web stranice. Najprije se radilo o stranicama baziranima na tekstu radi limita tadašnje tehnologije kao što je brzina prijenosa podataka, propusnosti te sirove procesorske snage obrade podataka na poslužitelju. Jednako tako razvojem tehnologije razvijale su se i same web stranice, odnosno aplikacije koje su polako postale sposobne obavljati jednake zadatke kao i one stolne.

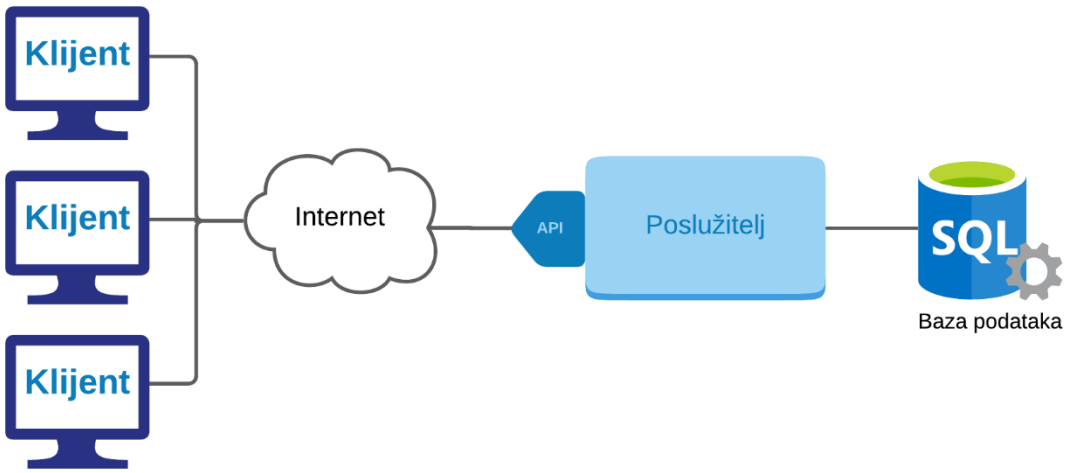
Web aplikacije za razliku od onih stolnih imaju mnoge prednosti poput automatskih nadogradnji bez potrebe za interakcijom korisnika, neovisnost o hardveru korisnika te brži razvoj s obzirom na to da je za razvoj web aplikacije potrebno razviti jednu verziju aplikacije, dok je u slučaju stolnih aplikacije u nekim slučajevima ovisno o hardveru i operacijskom sustavu korisnika potrebno razvijati i do nekoliko različitih verzija. [9]

U današnje vrijeme veliki se dio rada na računalu odvija putem web-a upravo radi brojnih prednosti koje takav način rada pruža, ali i dalje se može naći veliki broj aplikacija koje svoj rad zasnivaju na upravo stolnim aplikacijama, a razlozi za to mogu biti potreba za velikom procesorskom snagom, potreba za pristupom specijaliziranom hardveru, potreba za velikim kapacitetom pohrane ili pak s druge strane veoma skup i dugotrajan proces ponovnog razvoja. Najčešći primjeri stolnih aplikacija danas su razna ERP (eng. Enterprise resource Planning) rješenja, aplikacije koje zahtijevaju veliku procesorsku snagu poput onih za obradu slika i video zapisa te u kontekstu ovoga rada razni programski alati za razvoj aplikacija.

3.1. Aplikacijsko programsko sučelje

Aplikacijsko programsko sučelje ili sučelje za programiranje aplikacija je skup definicija i protokola za izradu, integraciju i prijenos podataka između aplikacija. Korištenje API-a omogućuje fleksibilnost, pojednostavljuje razvoj aplikacija te štedi vrijeme i novac. Također, API-i omogućavaju povezanost jednog programskog proizvoda s drugim bez direktnog znanja o tome kako je taj programski proizvod implementiran, u kojem programskom jeziku ili primjenom koje arhitekture. Jedna od glavnih svrha API-a je i skrivanje internih detalja kako sustav radi, odnosno njegov unutarnji dizajn te razotkrivanje samo onih dijelova koji su potrebni za implementiranje i korištenje takvog sustava. Korištenje API-a također omogućava da ostajemo dosljedni te da se sustav može koristiti bez obzira na interne promjene u sustavu. [10]

Slika 4 prikazuje primjer sustava uporabe API-a u tipičnom scenariju današnjeg interneta, odnosno web aplikacija. Prikazani sustav čini 5 elemenata, a to su klijent, server, API, internet te baza podataka. Klijent predstavlja korisnika web aplikacije, odnosno korisničku aplikaciju. Poslužitelj predstavlja pozadinsku aplikaciju koje prema mreži, odnosno internetu izlaže svoje API sučelje. Poslužitelj sadrži kod poslovne logike te nudi određene usluge. Poslužitelj također za potrebe rada može koristiti vlastitu bazu podataka. Više klijenata putem interneta mogu pozivati API metode poslužitelja i na taj način od poslužitelja tražiti obavljanje određenih funkcija. Poslužitelj zatim vraća odgovor u zadanom formatu.



Slika 4 Prikaz uporabe API-a

Kada govorimo o web i stolnim aplikacijama, danas nije nužno strogo odvajati ta dva pojma. Odvajanje poslovne logike, odnosno pozadinskog koda u web servise nije rezervirano isključivo za cjelokupne web aplikacije već se i pojedini dijelovi stolnih aplikacija mogu odvojiti u zasebne web servise koje zatim stolna aplikacija može koristiti u svom radu. Upravo API svojom sposobnošću za povezivanje dva različita proizvoda omogućuje povezivanje web i stolnih aplikacija na način da se stolna aplikacija sve više može promatrati kao korisnička aplikacija u konceptu web razvoja, ali sa svim prednostima stolnih aplikacija kao što je pristup većoj procesorskoj snazi, pristup većim kapacitetima za pohranu podataka ili pak pristup specijaliziranom hardveru.

Bilo da razvijamo novu ili održavamo postojeću aplikaciju, API nam može pomoći olakšati i pojednostavniti taj proces. Neke od glavnih prednosti uporabe API-a su [11]:

- **Poboljšana suradnja** - 1990. godine prosječno poduzeće koristilo je između 5 i 10 različitih poslovnih aplikacija dok današnje prosječno poduzeće koristi približno 1200 raznih aplikacija u oblaku od kojih veći dio nije povezan. [12] Korištenje API-a u ovakvim slučajevima omogućava integraciju između dosad nepovezanih aplikacija koje pomoću API-a mogu besprijekorno komunicirati jedna s drugom bez obzira na programski jezik ili arhitekturu u kojoj su implementirane. Preko takvih integracija kompanije mogu automatizirati radne procese i poboljšati suradnju.

- **Lakša inovativnost** - API omogućava kompanijama da nude nove servise i usluge na tržištu gdje su već prisutne, a ultimativno nudi mogućnost za širenje na nova tržišta uz velik povrat investicija. Također ovakav pristup nudi otvaranje dijelova sustava koji se tada mogu nuditi kao usluga drugim zainteresiranim stranama

- **Monetizacija podataka** - API-i predstavljaju izvrstan način za monetizaciju vrijednih podataka. Iako mnoge tvrtke nude besplatne pakete koji primarno služe da privuku pažnju i reklamiraju kompaniju, mnoge kompanije nude brojne pakete kojima se uspješno mogu naplatiti, tj. prodati vrijedni digitalni resursi

- **Dodana sigurnost** - Jedna od osnovnih svrha API-a je i sigurnost, odnosno stvaranje dodatnog sloja između podataka i servera.

3.1.1. Tipovi API-a

Kada izdajemo API, odnosno postavljamo API na server i puštamo ga u uporabu postoje 3 glavne politike kako API radi. Prema politikama koje API može implementirati poznajemo i tri glavna tipa API-a, a to su [10] [11]:

- **Privatni API** - Privatni API je namijenjen isključivo za internu uporabu što daje potpunu kontrolu kompaniji nad odlukama kako će i što API raditi. Glavna svrha privatnog ili internog API-a je poboljšati produktivnost i komunikaciju između različitih internih razvojnih timova

- **Javni API** - Javni API dostupan je svima na istoj mreži gdje je objavljen i sam API. Ova vrsta API-a omogućuje trećim stranama korištenje takvog API-a te daje pristup resursima koji se mogu iskoristiti za unapređenje i inovativnost vlastite kompanije

- **Partnerski API** - Partnerski API predstavlja vrstu API-a koja je podijeljena isključivo sa specifičnim partnerima. Tipično takva vrsta API-a prema mreži djeluje kao i javni API, ali je za njegovo korištenje potrebno osigurati podatke za prijavu od partnera u čijem je vlasništvu sam API.

3.1.2. Tipovi API protokola

Sve šira uporaba API-a dovela je do potrebe za definiranjem skupa pravila, odnosno protokola kojima se jasno definira koje komande te koje sve tipove podataka API prima. Neki od najvažnijih API protokola danas su [11] [10] [13] [14]:

- **XML/JSON-RPC -** RPC (eng. Remote Procedural Call) je protokol kojim jedno računalo na mreži može od drugog računala na istoj mreži tražiti neku uslugu. Za razliku od REST API-a, RPC API-i ne koriste direktno HTTP metode za pristup resursima, već postavljaju vlastite operacije koja sadrže resurse te zatim pozivaju te metode putem jedne od HTTP metoda. Kod RPC API-a razlikujemo XML-RPC koji koristi XML format u svojim pozivima i JSON-RPC koji koristi JSON format.

- **SOAP** (eng. Simple Object Access Protocol) predstavlja API protokol koji definira XML kao format API poruka te HTTP i SMTP kao način primanja podataka. Pomoću SOAP API-a olakšana je komunikacija i dijeljenje podataka između aplikacija koje se pokreću u različitim okolinama ili su pak pisane u različitim jezicima.

- **REST** (eng. Representational State Transfer) predstavlja skup API arhitekturnih principa za dizajniranje web servisa fokusiranih prema resursima sustava što ujedno

znači da ne postoji službeni standard. [15] Ako API poštuje arhitekturna načela REST-a tada se takav API naziva RESTful API. Kao što navodi Roy Fielding [16] API se smatra RESTful tako dugo dok zadovoljava šest sljedećih uvjeta:

1. Klijent-poslužitelj arhitektura gdje se razdvajaju ovisnosti o korisničkom sučelju i podacima na klijentu i poslužitelju.

2. API mora implementirati princip rada bez stanja, odnosno svaki zahtjev klijenta prema serveru mora sadržavati sve potrebne informacije kako bi se zahtjev mogao implementirati te se ti podaci ne smiju pohranjivati na serveru. Ovim načinom rada stanje se pohranjuje isključivo na strani klijenta. Implementacijom ovog principa rada poboljšavaju se vidljivost, pouzdanost i skalabilnost.

3. Kako bi se smanjio mrežni promet potrebno je svaki odgovor sa servera označiti kao sadržaj koji se sprema u predmemoriju ili kao onaj koji se ne sprema. Na ovaj način klijent može ponovo upotrijebiti primljene podatke za ekvivalentne zahtjeve.

4. Uniformno sučelje predstavlja najbitniju značajku koja razlikuje arhitektonski stil REST od ostalih. Kako bi se postiglo uniformno sučelje potrebno je nekoliko ograničenja, a ona su: identifikacija resursa u zahtjevima, manipulacija resursima kroz reprezentaciju, samoopisne poruke te hipermedija kao osnova promjene stanja.

5. Slojevitost sustava stvara arhitekturu gdje se sustav sastoji od hijerarhijskih slojeva te tako ograničava komponente sustava na način da svaka komponenta ne može vidjeti dalje od sloja s kojim je u neposrednoj interakciji. Ovakva arhitektura omogućuje uravnoteživanje opterećenja te povećanje sigurnosti sustava. Ovakav pristup ima nedostatak što dovodi do povećanja latencije te smanjenja performansi.

6. Kod na zahtjev predstavlja način rada gdje funkcionalnosti klijenta mogu biti proširene na način da klijent od poslužitelja prima izvršni kod koji zatim izvršava. Na ovaj način smanjujemo količinu funkcionalnosti koja mora biti pred implementirana na klijentu te se samim time klijent pojednostavljuje.

3.2. Arhitekture razvoja aplikacija

Kod razvoja aplikacija postoje mnogi arhitekturni stilovi, a neki od njih su: [11] [10]

- **Monolitna arhitektura -** Predstavlja arhitekturu gdje je aplikacija samostalna i neovisna te je sama aplikacija odgovorna za izvođenje svakog koraka neke funkcije. Monolitna se arhitektura u velikom dijelu zajednice razvoja aplikacija smatra zastarjelim načinom razvoja aplikacija radi brojnih nedostataka. Kao najveći nedostatak monolitne arhitekture smatra se kompleksnost prilikom skaliranja i održavanja, kompleksnost implementacije novih mogućnosti radi velike povezanosti i utjecaja jednog dijela sustava na drugi te kao jedan od najvećih nedostataka nemogućnost uvođenja novih tehnologija bez ponovnog razvoja cijele aplikacije. Ovakva je arhitektura pogodna za razvoj malih, brzo implementiranih aplikacija bez mnogo poslovne logike i potrebe za skaliranjem.

- **Klijent-poslužitelj arhitektura -** Ova arhitektura predstavlja distribuiranu aplikacijsku strukturu gdje poslužitelj, odnosno davatelj određene usluge obavlja posao koji prima od zahtjevatelja usluge, odnosno klijenta. U praktičnom dijelu rada, upravo ovakvu arhitekturu možemo prepoznati u načinu implementacije korisničke i pozadinske aplikacije, gdje korisnička aplikacije, odnosno klijent zahtjeva određene usluge od pozadinske aplikacije, odnosno poslužitelja.

- **Arhitektura temeljena na komponentama -** Arhitektura temeljena na razdvajanju ovisnosti sustava na manje, neovisne, ponovo upotrebljive i slabo povezane komponente koje se spajanjem implementiraju u sustav. Komponente u ovakvom sustavu međusobno komuniciraju samo preko sučelja pojedine komponente, dok su ostali dijelovi komponente skriveni i učahureni u vlastiti paket. Ovakva je arhitektura vidljiva u korisničkoj aplikaciji praktičnog dijela rada gdje je aplikacija zasnovana na komponentama.

- **Peer-to-peer -** Arhitektura zasnovana na mreži čvorova (eng. peer) koji se prema ostalim čvorovima u mreži ponašaju kao klijent i poslužitelj odjednom. Peer-to-peer arhitektura predstavlja distribuiranu aplikacijsku arhitekturu čiji je cilj podijeliti zadatke i opterećenja aplikacije između pojedinih čvorova. Za razliku od arhitekture klijent-poslužitelj koja predstavlja centralizirani sustav, peer-to-peer arhitektura predstavlja decentralizirani sustav. Također za razliku od klijent-poslužitelj arhitekture gdje svi klijenti ovise o radu poslužitelje u peer-to-peer sustavu više čvorova pruža istu uslugu što ovakav sustav čini pouzdanijim i bržim, ali zahtjevnijim za implementaciju.

Osim gore navedenih, prilikom korištenja API-a uobičajeno ćemo slijediti jednu od dvije arhitekture, a to su servisno orijentirana arhitektura (SOA) i mikroservisna arhitektura.

3.2.1. Servisno orijentirana arhitektura

Servisno orijentirana arhitektura ili skraćeno SOA predstavlja pristup razvoja softvera na način da se softverske komponente mogu ponovo koristiti preko servisnih sučelja. U SOA arhitekturi svaka komponenta odnosno servis se sastoji od koda i podataka potrebnih za izvršavanje određene poslovne funkcije. Komponente ove servisno orijentirane arhitekture su slabo povezane što znači da se mogu međusobno pozivati i koristiti iako međusobno ne posjeduju mnogo informacija jedne o drugima. [17] [18]

Što točno predstavlja servisno-orijentirana arhitektura može se definirati na sljedeće načine [19]:

- Skup servisa i funkcionalnosti koje ponuditelj usluge želi nuditi svojim korisnicima, partnerima ili pak drugim područjima unutar iste organizacije
- Arhitekturni stil koji zahtijeva pružatelja servisa, korisnika servisa zajedno sa opisom servisa i potencijalnog posrednika
- Skup arhitektonskih načela, uzoraka i kriterija koji određuju karakteristike kao što su modularnost, ponovna iskoristivost, odvojena odgovornost, itd.
- Model programiranja upotpunjen standardima, alatima i tehnologijama kako bi podržao razne vrste servisa
- Posredničko rješenje optimizirano za sastavljanje servisa, njihovo upravljanje i nadziranje

Pozadinska je aplikacija u praktičnom dijelu rada temeljena upravo na servisno orijentiranoj arhitekturi.

Servisno orijentirana arhitektura omogućuje bržu isporuku gotovih aplikacija te nudi veću efikasnost prilikom implementacije novih funkcionalnosti. S obzirom da se sustav sastoji od različitih slabo povezanih komponenti veoma je jednostavno u aplikaciji implementirati nove funkcionalnosti bez uznemiravanja ostatka sustava. Također, servisno orijentirana arhitektura omogućuje izlaganje postojećih aplikacija novim okruženjima. Primjena servisno orijentirane arhitekture u velikim sustavima može dovesti do bolje suradnje između poslovnog i razvojnog tima. Ovakva se suradnja zasniva na razlogu što je određena komponenta zadužena za izvršavanje jedne poslovne funkcije čiji razvojni tim može

efikasnije surađivati s timom poslovnih analitičara zaduženih za jednaku poslovnu funkciju kao i razvijena komponenta.

3.2.2. Mikroservisna arhitektura

Mikroservisi predstavljaju arhitekturni stil gdje je velika i kompleksna aplikacija sastavljena od jednog ili više servisa koji mogu biti neovisno postavljeni na poslužitelje te također mogu biti pisani u različitim jezicima. Ovakva svojstva mikroservisa omogućuje upravo komunikacija između njih, a to je komunikacija putem jezično neutralnog aplikacijsko programskog sučelja, odnosno API-a kao što je to na primjer REST. [19]

Svrha ovakve arhitekture je podijeliti cjelokupni rad sustava na manje zadatke na način da je jedan mikroservis zadužen za izvršavanje jednog i samo jednog zadatka. [19]

Realizacija aplikacije na način da je ona sastavljena od skupa odvojenih servisa omogućuje lakše testiranje, održavanje i skaliranje same aplikacije. [11]

Kada uspoređujemo mikroservisnu arhitekturu i servisno orijentiranu arhitekturu, iako se u oba slučaja radi o skupu servisa, namjena i ambicije tih servisa su ponešto drugačije. S jedne strane SOA pokušava kreirati servise na način da su oni dostupni svima koji ih žele koristiti, dok s druge strane mikroservisna arhitektura pokušava kreirati servise sa znatno fokusiranijim i ograničenim ciljem na način da se ponaša kao dio jedinstvenog distribuiranog sustava. Takav distribuirani sustav često nastaje na način da veliku monolitnu aplikaciju dijelimo na manje dijelove odnosno mikroservise čiji je zadatak nastaviti raditi zajedno kao jedinstvena aplikacija. [19]

Ovakav način implementacije aplikacije omogućuje neovisno postavljanje dijelova sustava na poslužitelj te na taj način određene greške i nedostaci u jednom dijelu sustava ne utječu na ostale dijelove. Na ovaj način je u sustav moguće implementirati nove funkcionalnosti ili pak implementirati poboljšanja u postojeće bez potrebe za prestankom rada cijelog sustava. Pojedini mikroservisi sustava trebaju biti dovoljno mali kako bi mogli biti razvijeni, testirani i postavljeni na poslužitelj od strane jednog razvojnog tima. Prilikom razvoja i dodavanja funkcionalnosti u monolitnom sustavu potrebno je voditi računa o mnogim ovisnostima u sustavu radi velike isprepletenosti koda što dovodi do potrebe za korekcijom mnogo različitih dijelova koda kako bi se mogla implementirati nova funkcionalnost. Upravo radi stroge nepovezanosti pojedinih mikroservisa ovisnosti su minimizirane te je implementacija novih funkcionalnosti u sustav znatno jednostavnija.

Još jedna od velikih prednosti mikroservisne arhitekture je i upravljanje iznimkama, odnosno ukoliko pojedini mikroservis sustava postane nedostupan ostatak sustava će moći nastaviti s radom pod uvjetom da su ostali mikroservisi u stanju ispravno obraditi pogreške.

4. Vanjski servisi

Kao što je prethodno napomenuto aplikacija za svoj rad koristi tri vanjska servisa, odnosno API-a. U nastavku će se detaljnije opisati svaki od tih servisa kao što su krajnje točke koje aplikacija koristi, koje podatke servis prima, koje podatke šalje te koji su uvjeti rada. Također opisat će se proces odabira API-a te zašto su odabrani baš ova tri kao izvor podataka u aplikaciji.

Prethodno su opisana tri tipa API-a, a to su privatni, javni i partnerski. U slučaju API-a odabranih kao izvor podataka o kripto valutama u ovoj aplikaciji radi se o partnerskim API-ima te je potrebno od vlasnika API-a zatražiti API ključeve pomoću kojih je moguće pozivati same API metode.

4.1. Odabir API-a

Kod odabira odgovarajućeg API-a u obzir su se uzimali parametri poput limita broja zahtjeva u određenom vremenu, podaci i detalji koje API nudi za podatke kripto valuta te koje povijesne podatke nudi i koliko daleko u povijest. Također, važno je napomenuti kako se u radu koriste samo besplatne verzije API-a. Prilikom izbora odgovarajućeg API-a u izboru su bili Nomics API, CoinAPI, Poligon.io API, CryptoCompare API, CoinMarketCap API i Binance API. Svi navedeni API-i implementirani su kao REST API.

Binance API

Ovaj API nije bilo potrebno uspoređivati sa ostalim API-ima iz razloga što Binance API nudi podatke novčanika korisnika na Binance platformi te je iste podatke moguće dobiti isključivo putem ovog API-a.

Nomics API

Ovaj API nudi informacije o trenutnoj vrijednosti kripto valuta, ali kao nedostatke ima nemogućnost pružanja povijesnih podataka te je broj poziva u jedinici vremena limitiran na 1 poziv u sekundi što za svega nekoliko istovremenih korisnika aplikacije nije dovoljno. Upravo iz tog razloga i iz razloga što ne nudi povijesne podatke ovaj API nije bio valjan izbor za implementaciju praćenja trendova kripto valuta. [20]

Polygon.io API

Ovaj API nudi limit broja poziva u jedinici vremena od 5 poziva u minuti što je još restriktivniji limit nego prethodni Nomics API. Za razliku od Nomics API-a ovaj API nudi povijesne podatke za prethodne 2 godine. S obzirom na veoma limitiran broj poziva u jedinici

vremena ovaj API također nije odabran kao izvor podataka. Također ovaj API nudi 2 godine povijesnih podataka dok idući CryptoCompare API nudi podatke od postojanja pojedine kripto valute što je u slučaju kripto valute Bitcoin i do više od 8 godina. [21]

CryptoCompare API

Ovaj API nudi kao što je prethodno napomenuto povijesne podatke od početka postojanja kripto valute za svaki dan. Također API nudi povijesne podatke po satu za zadnja 3 mjeseca te po minuti za zadnja 24 sata. Što se tiče limita broja zahtjeva u jedinici vremena ovaj API nudi 100,000 poziva mjesečno te 250,000 poziva ukupno što je daleko dovoljno za potrebe ove aplikacije. S obzirom na količinu i detaljnost podataka koje nudi i na povoljan limit poziva u jedinici vremena ovaj je API odabran kao primaran izvor informacija o vrijednosti kripto valuta. [22]

CoinMarketCap API

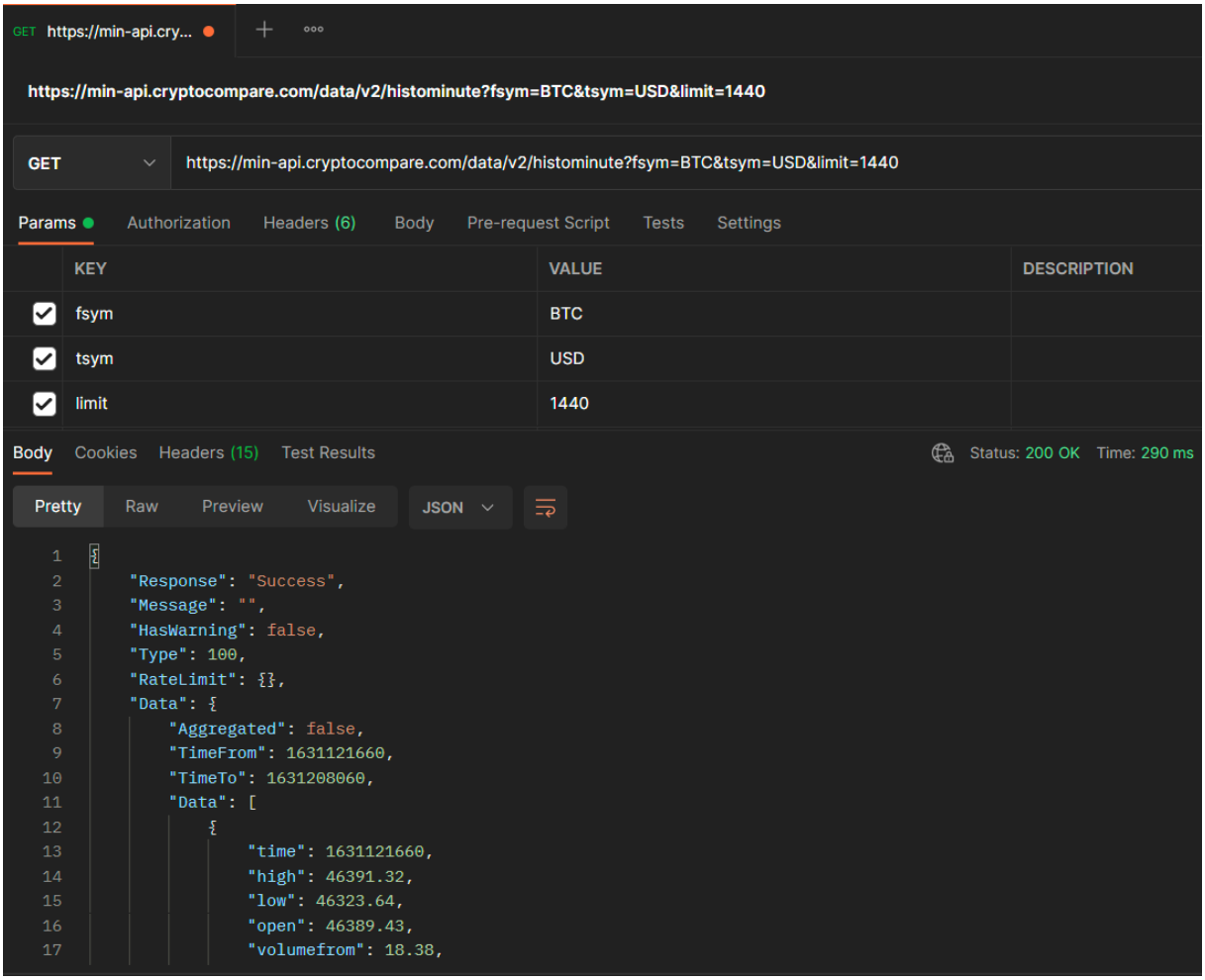
Ovaj API ne nudi povijesne podatke te je broj poziva u jedinici vremena limitiran na 10,000 poziva mjesečno. Kao jedna od najprestižnijih i najkorištenijih internetskih platformi za praćenje vrijednosti kripto valuta i njihovo rangiranje, ovaj je API odabran kao izvor podataka o samim kripto valutama i njihovim osnovnim informacijama bez vrijednosti. [23]

CoinAPI

Ovaj API nudi podatke o trenutnim vrijednostima kripto valute te također nudi i povijesne podatke o vrijednostima kripto valuta. Mana ovog API-a je njegova limitiranost broja zahtjeva u jedinici vremena od 100 zahtjeva dnevno što predstavlja relativno mali broj dostupnih zahtjeva i ne zadovoljava potrebe rada same aplikacije. [24]

Za testiranje API-a, njihovih poziva i podataka koje oni vraćaju korišten je alat Postman1. Postman je alat koji nudi jednostavno sučelje za korištenje i testiranje API-a. Na slici 5 prikazan je primjer testiranja API krajnje točke servisa CryptoCompare za dohvaćanje povijesnih podataka kripto valute za svaku minutu u zadnja 24 sata. Ovakvi alati uvelike olakšavaju testiranje API-a bez potrebe za konkretnom implementacijom u kodu.

[1] https://www.postman.com/



Slika 5 Primjer testiranja API krajnje točke alatom Postman [autorski rad]

4.2. Binance API

Binance je u trenutku pisanja vodeća burza kripto valuta u svijetu. Binance je osnovan 2017. godine u Hong Kongu, ali je tvrtka od 2018. godine radi sve većih restrikcija na trgovanje kripto valutama koje uvodi Kina svoje sjedište preselila na Maltu. Binance na svojoj platformi omogućuje trgovanje sa više od 500 različitih kripto valuta. [25]

O samom uspjehu i popularnosti Binance platforme svjedoče i brojke iskazane na službenoj stranici gdje je napomenuto kako se na platformi u jednome danu napravi 2 milijarde američkih dolara prometa te više od 1.4 milijuna transakcija u sekundi. [26]

Kao što je napomenuto Binance je platforma za trgovanje kripto valutama koja ujedno služi i kao novčanik gdje korisnici platforme čuvaju kripto valute koje posjeduju te su upravo to podaci koje će ova aplikacija dohvaćati sa Binance API-a. Aplikacija će pomoću Binance API-a dohvaćati podatke o stanju korisnikova novčanika na Binance platformi te će na taj način sinkronizirati stanje korisnikova novčanika unutar aplikacije sa onim na Binance

platformi. Binance API nudi besplatnu API krajnju točku za preuzimanje podataka o korisnikovom novčaniku, a za dohvaćanje navedenih podataka korisnik je potreban osigurati podatke potrebne za autentikaciju, a to su API ključ (eng. key) i tajni ključ (eng. secret key). Navedene ključeve svaki korisnik Binance platforme može kreirati kreiranjem novog API-a unutar postavki Binance platforme. Oba su ključa reprezentirana kao niz od 64 znakova.

Osnovna krajnja točka API-a je https://api.binance.com, a u slučaju problema sa performansama na raspolaganju su još tri dodatne https://api1.binance.com, https://api2.binance.com i https://api3.binance.com. Sve krajnje točke API-a vraćaju podatke u obliku JSON formata ili pak polja (eng. array).

Za potrebe ove aplikacije potrebna je specifična API krajnja točka koja spada u kategoriju krajnje točke novčanika (eng. wallet endpoints), a to je krajnja točka dnevnog pregleda računa (eng. Daily Account Snapshot). Radi se o GET metodi na adresi /sapi/v1/accountSnapshot. Navedena API krajnja točka prima ukupno 6 parametara koji se šalju kao parametri u URL-u (eng. query string). Od ukupno šest mogućih parametara potrebna su njih četiri, a to su type, startTime, recvWindow i timestamp. Za verifikaciju na ovoj krajnjoj točki potreban su dva dodatna parametra, a to su signature kao parametar u URL-u te parametar naziva X-MBX-APIKEY kao parametar zaglavlja u kojem se prosljeđuje API ključ korisnika. Parametar signature predstavlja HMAC SHA256 potpis gdje je tekst skup svih URL parametara, a kriptografski ključ tajni ključ korisnika. [27]

Primjer ovakvog poziva:

https://api.binance.com/sapi/v1/accountSnapshot?type=SPOT&recvWindow=50000&startTime=1629241200000&timestamp=1629287107857&signature=5af88b1942eedd73f85952fd9d6b100b285761688c4a57706e18e1f6c6bf0334

Dio odgovora ovakvog poziva:


```
{
    "code": 200,
    "msg": "",
    "snapshotVos": [
        {
            "type": "spot",
            "updateTime": 1629244799000,
            "data": {
                "totalAssetOfBtc": "0.00507142",
                "balances": [
                    {
                        "asset": "BTC",
                        "free": "0.00034149",
                        "locked": "0"
                    },
```

```
                    {
                        "asset": "ETH",
                        "free": "0",
                        "locked": "0"
                    },
                ]
            }}]
}
```


4.3. CryptoCompare API

CryptoCompare je platforma koja svojim korisnicima pruža podatke o tržištu kripto valuta. U svojoj bazi podataka CryptoCompare sadrži podatke od preko 5,300 kripto valuta. [28]

Ova platforma pruža veoma cjelovito API rješenje koje koriste korisnici širom svijeta, a sam API obrađuje preko 25 milijardi zahtjeva za preko 30 miliona korisnika mjesečno što predstavlja veoma respektabilne brojke. Za korištenje ovog API-a postoje nekoliko različitih planova od personalnog plana koji je besplatan za određene osobne upotrebe pa sve do enterprise plana koji može postići cijenu i do nekoliko desetaka tisuća američkih dolara. Za potrebe ove aplikacije korištena je besplatna verzija API plana koja korisniku omogućuje 100,000 poziva mjesečno te 250,000 poziva ukupno što je za potrebe ove aplikacije sasvim dovoljno. [29]

CryptoCompare API pruža mnogo različitih krajnjih točaka, a najinteresantnije su krajnje točke za dohvaćanje povijesnih podataka za svaku minutu i za svaki sat te krajnja točka za dohvaćanje trenutne vrijednosti više valuta. U slučaju ove aplikacije CryptoCompare API se poziva u tri slučaja korištenja, a to je slučaj da korisnik pregledava kripto valute u aplikaciji, slučaj inicijalnog punjenja baze podataka povijesnim podacima o vrijednosti kripto valuta i slučaj da pozadinski servis aplikacije periodično preuzima podatke za potrebe spremanja istih u bazu podataka. Od API-a tražimo 1440 zapisa što predstavlja podatke za prethodna 24 sata u slučaju korištenja pregleda kripto valuta dok u slučaju pozadinskog servisa od API-a tražimo trenutne vrijednosti svih kripto valuta u bazi podataka. U slučaju inicijalnog punjenja baze podataka povijesnim podacima o vrijednosti kripto valuta API krajnja točka se poziva za 2000 zapisa što predstavlja podatke za svaki sat u prethodnih nešto više od 5 godina.

Osnovna krajnja točka API-a je https://min-api.cryptocompare.com/data. Sve su tri krajnje točke prezentirane GET metodom.

Krajnja točka za preuzimanje trenutne vrijednosti više kripto valuta je na adresi /pricemulti te prima dva obavezna parametra koji se šalju u URL-u, a to su fsyms i tsyms. Oba parametra su tipa string gdje fsyms predstavlja simbole kripto valuta odvojene zarezima

za koje se vrijednosti dohvaćaju, a tsyms predstavlja simbole valuta u kojima želimo te vrijednosti izraziti.

Ako bi željeli poslati zahtjev koji vraća vrijednosti kripto valuta Bitcoin i Etherium u valutama euro i američki dolar tada bi poziv izgledao ovako:

```
API krajnja točka + /pricemulti?fsyms=BTC,ETH&tsyms=USD,EUR
```

Odgovor navedenog poziva API-a izgleda ovako:

```
{
    "BTC": {
        "USD": 45830.26,
        "EUR": 39270.3
    },
    "ETH": {
        "USD": 3111.32,
        "EUR": 2668.63
    }
}
```

Krajnja točka za preuzimanje povijesnih vrijednosti kripto valuta za svaki sat je na adresi /v2/histohour te prima mnogo parametara, a za potrebe ove aplikacije korištena su tri parametra koji se šalju u URL-u, a to su limit, fsym i tsym. Parametar limit je tipa int dok su druga dva parametra tipa string gdje limit predstavlja ograničenje koliko zapisa želimo primiti, fsym predstavlja simbol kripto valute za koju se vrijednosti dohvaćaju, a tsym predstavlja simbol valute u kojoj želimo te vrijednosti izraziti.

Ako bi željeli poslati zahtjev koji vraća vrijednosti kripto valute Bitcoin izražene u valuti američki dolar tada bi poziv izgledao ovako:

```
API krajnja točka + /v2/histohour?fsym=BTC&tsym=USD&limit=2000
```

Dio odgovora navedenog poziva API-a izgleda ovako:

```
{
    "Response": "Success",
    "Message": "",
    "HasWarning": false,
    "Type": 100,
    "RateLimit": {},
    "Data": {
        "Aggregated": false,
        "TimeFrom": 1629288000,
        "TimeTo": 1629306000,
        "Data": [
            {
                "time": 1629288000,
                "high": 45112.19,
```

```
                "low": 44670.47,
                "open": 44759.45,
                "volumefrom": 914.87,
                "volumeto": 41072724.98,
                "close": 45003.59,
                "conversionType": "direct",
                "conversionSymbol": ""
            },
            {
                "time": 1629291600,
                "high": 45393.84,
                "low": 45003.59,
                "open": 45003.59,
                "volumefrom": 1685.71,
                "volumeto": 76278844.81,
                "close": 45288.11,
                "conversionType": "direct",
                "conversionSymbol": ""
            },
            ...
        ]
    }
}
```


Krajnja točka za preuzimanje povijesnih vrijednosti kripto valuta za svaku minutu je na adresi /v2/histominute te se po pitanju parametara ne razlikuje od prethodnog API poziva. Ako bi željeli poslati zahtjev koji vraća vrijednosti kripto valute Bitcoin izražene u valuti američki dolar za svaku minutu prethodnih 24 sata tada bi poziv izgledao ovako:

```
API krajnja točka + /v2/histominute?fsym=BTC&tsym=USD&limit=1440
```

Odgovor ovakvog poziva identičan je onom prethodnog API poziva.

4.4. CoinMarketCap API

CoinMarketCap je najpoznatija i najposjećenija internetska platforma za praćenje cijena kripto valuta. Platforma je osnovana 2013. godine te je ubrzo postala najpopularnija platforma za praćenje cijena kripto valuta sadržavši podatke tisuća valuta. Zanimljiv je podatak da vlada SAD-a za potrebe istraživanja tržišta kripto valuta koristi upravo podatke sa ove platforme. [30]

CoinMarketCap API pruža mnogo različitih krajnjih točaka, a nama je najinteresantnija krajnja točka za dohvaćanje liste kripto valuta.

Osnovna krajnja točka API-a je https://pro-api.coinmarketcap.com/v1. Krajnja točka za preuzimanje liste kripto valuta je GET metoda na adresi /cryptocurrency/map. Ova krajnja točka prima mnogo parametara, a u slučaju ove aplikacije potreban je samo parametar sort kojim određujemo kako će biti sortirani podaci koje API vraća.

Ako bi željeli poslati zahtjev koji vraća listu kripto valuta sortiranu prema važnosti to je podatak koji određuje sama platforma tada bi poziv izgledao ovako:

```
API krajnja točka + /cryptocurrency/map?sort=cmc_rank
```

Odgovor navedenog poziva API-a izgleda ovako:


```
{
    "status": {
        "timestamp": "2021-08-19T13:35:10.544Z",
        "error_code": 0,
        "error_message": null,
        "elapsed": 10,
        "credit_count": 1,
        "notice": null
    },
    "data": [
        {
            "id": 1,
            "name": "Bitcoin",
            "symbol": "BTC",
            "slug": "bitcoin",
            "rank": 1,
            "is_active": 1,
            "first_historical_data": "2013-04-28T18:47:21.000Z",
            "last_historical_data": "2021-08-19T13:29:03.000Z",
            "platform": null
        },
        {
            "id": 1027,
            "name": "Ethereum",
            "symbol": "ETH",
            "slug": "ethereum",
            "rank": 2,
            "is_active": 1,
            "first_historical_data": "2015-08-07T14:49:30.000Z",
            "last_historical_data": "2021-08-19T13:29:03.000Z",
            "platform": null
        },
            ...
    ]
}
```

5. Izrada vlastite aplikacije

Kao praktičan dio ovoga rada izrađena je PinOpt web aplikacija koja služi kao pomoć korisnicima u praćenju trendova kripto valuta. Aplikacija korisniku nudi mogućnost pregleda podataka kripto valuta, pregled kretanja cijene kripto valute u nekom vremenskom razdoblju grafički, dodavanje kripto valuta u favorite te novčanik korisnika gdje korisnik može spremati koliko koje kripto valute posjeduje te mu aplikacija na temelju tih podataka pruža dodatne njemu relevantne informacije. Aplikacija također nudi mogućnost pretplate na e-mail obavijesti o vrijednostima pojedinih kripto valuta te konfiguraciju postavki same pretplate.

Aplikacija se sastoji od ASP.NET Core WebApi projekta za pozadinsku aplikaciju te Angular projekta kao korisničke aplikacije. U narednom tekstu će se detaljnije opisati korišteni alati, dizajn sustava te glavne funkcionalnosti aplikacije.

5.1. Alati

Razvoj pozadinske aplikacije vršen je u alatu Visual Studio 2019 koji predstavlja integrirano razvojno okruženje razvijeno od strane Microsofta namijenjeno za razvoj različitih vrsta programa. Ovo razvojno okruženje sadrži sve komponente potrebne za razvoj cjelokupnog programskog rješenja kao što su alati za razvoj i analizu koda, testiranje i isporuku gotove aplikacije. Za ovaj projekt korištena je Community verzija 16.11.

Za razvoj klijentske aplikacije, odnosno Angular aplikacije korišten je razvojni alat Visual Studio Code verzije 1.59.0. Za razliku od Visual Studio 2019, Visual Studio Code predstavlja laganiju verziju razvojnog alata koja je tipa otvorenog koda te kao jednu od glavnih značajki posjeduje sposobnost pokretanja na raznim platformama. Također za razliku od standardnog alata Visual Studio 2019 koji posjeduje bogatstvo alata i mogućnosti, Visual studio Code je nešto siromašniji na tom području, ali uz mnoštvo dostupnih ekstenzija i postavki nudi mnogo mogućnosti uz brz radi i malu potrošnju resursa. Upravo iz tog razloga i želje za razdvajanjem razvojnog okruženja pozadinske i klijentske aplikacije Visual Studio Code je odabran za razvojno okruženje klijentske odnosno Angular aplikacije. Verzija Angulara korištena u programskom rješenju je 12.

Za kreiranje i upravljanje bazom podataka korišten je alat SQL Server Management Studio koji predstavlja integrirano okruženje za upravljanje SQL infrastrukturama. U ovom projektu korišten je SQL Server. Verzija alata je 18.8.

Aplikacija je također postavljena na vlastiti server koji je samostalno konfiguriran unutar računala koje pokreće Windows 10 te sadrži vlastitu statičnu IP adresu koja je

potrebna kako bi server bio vidljiv drugim računalima na internetu. Za konfiguriranje IIS servera korišten je alat Internet Information Services Manager u verziji 10.0.19041.1.

5.2. Pregled funkcionalnosti

Web aplikacija PinOpt služi za praćenje trendova kripto valuta. Aplikacija je podijeljena na nekoliko osnovnih kategorija funkcionalnosti, a to su:

- **Korisnik** ima mogućnost registracije novog korisničkog profila gdje se definiraju korisničko ime i lozinka kojima se korisnik kasnije može verificirati u svrhu prijave u sustav. Prilikom registracije korisniku se otvara forma za registraciju gdje je potrebno upisati tražene podatke. Aplikacija zatim provjerava ispravnost unesenih podataka te korisniku javlja poruku o eventualnim greškama i neispravnostima. Nakon uspješne registracije korisnik se pomoću forme za prijavu može prijaviti u aplikaciju. Aplikacija provjerava ispravnost unesenih podataka te ako su podaci ispravni prijavljuje korisnika. Za autentikaciju korisnika aplikacija koristi JSON Web tokene (JWT).

Kako bi korisnik mogao pristupati naprednim funkcionalnostima aplikacije, osim registracije i prijave, potrebno je i potvrditi e-mail adresu korisnika.

- **Kripto valute -** Glavne funkcionalnosti ove aplikacije odnose se upravo na kripto valute. Početna stranica aplikacije sadrži tablični popis svih kripto valuta koje aplikacija podržava. Ovdje su korisniku dostupni podaci o trenutnoj cijeni, maksimalnoj dostignutoj cijeni te postotnoj promjeni cijene u 24 sata, 7 dana i mjesec dana za svaku kripto valutu.
- Odabirom pojedine kripto valute na početnoj stranici korisnika se vodi na stranicu koja sadrži detaljnije podatke odabrane kripto valute koja sadrži grafički prikaz kretanja cijena kripto valuta u nekom vremenskom intervalu. Također uz grafički prikaz dostupni su i prethodno navedeni podaci sa početne stranice za odabranu kripto valutu.
- Sve prethodno navedene funkcionalnosti dostupne su za sve vrste korisnika, pa tako i neprijavljenog. Nakon uspješne prijave korisnika na stranici detalja kripto valute otvaraju se mogućnosti dodavanja kripto valute u listu favorita za korisnika te dodavanje kripto valute u novčanik korisnika o čemu će više riječi biti u nastavku.

- **Novčanik** je funkcionalnost aplikacije da korisnik svome profilu može pridružiti određene kripto valute. Prilikom dodavanja kripto valuta u novčanik potrebno je unijeti podatke o tome o kojoj se kripto valuti radi, o kojoj količini se radi te na koji datum je određena količina određene kripto valute kupljena. Na temelju tih podataka aplikacija izračunava i korisniku prezentira okvirne podatke o stanju korisnikova novčanika kao što je ukupna

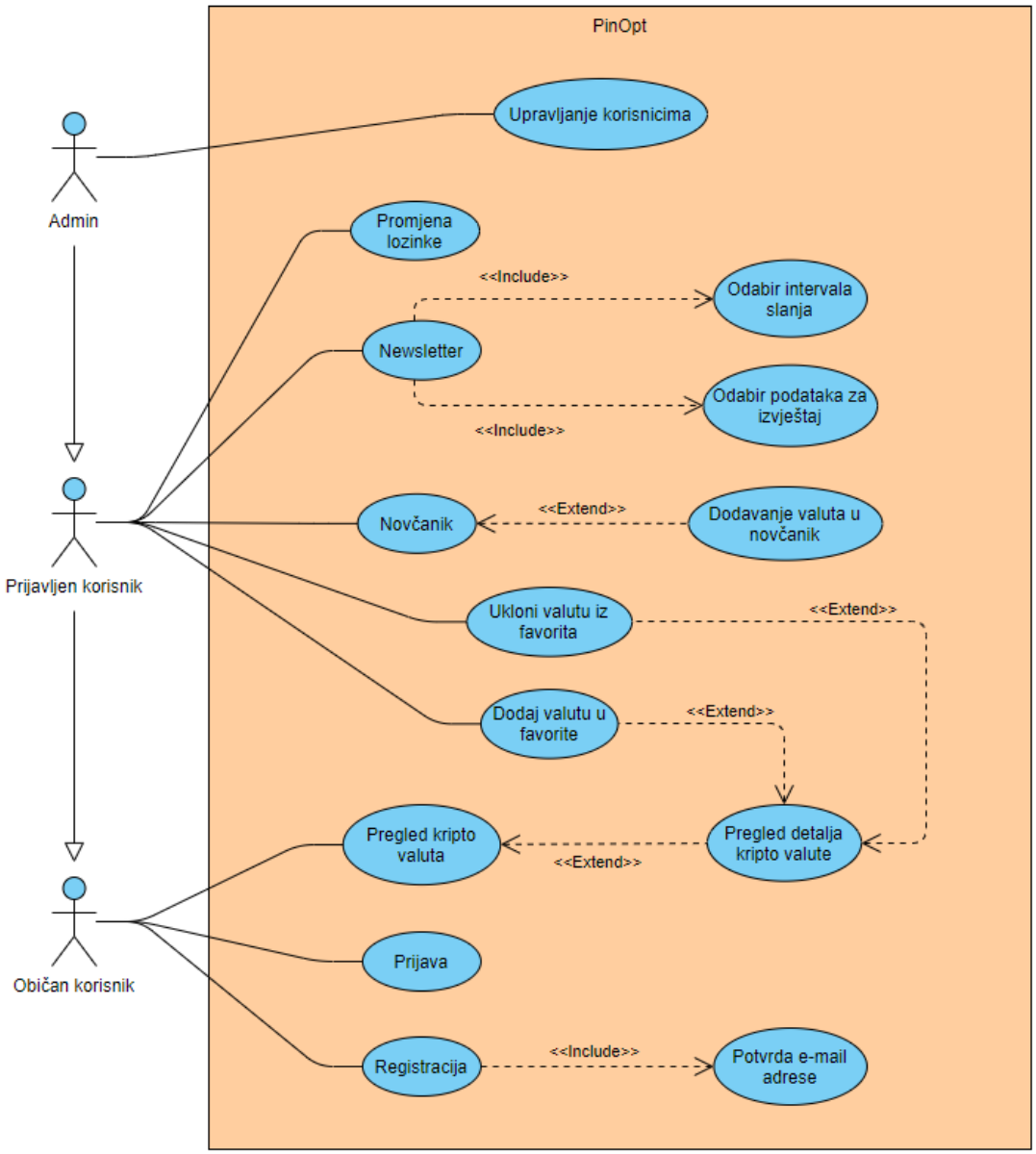
trenutna vrijednost svih kripto valuta u novčaniku korisnika, ukupna trenutna vrijednost pojedinih valuta u novčaniku te profit odnosno gubitak prikazan postotno i brojčano kako za cijeli novčanik tako i za svaku pojedinu valutu u njemu.

- **Bilten** (eng. Newsletter) je funkcionalnost aplikacije koja nudi korisniku opciju odabira želi li primati automatski generirane personalizirane izvještaje na svoju e-mail adresu. Korisniku se nudi opcija odabira u kojim vremenskim intervalima želi primati izvještaje, a moguće je odabrati jedan od 5 ponuđenih intervala koji mogu biti od jednom dnevno (u podne) pa sve do svakoga sata. Osim intervala korisnik može odabrati koje podatke želi primati u izvještaju, a na raspolaganju mu stoje podaci o kripto valutama koje se nalaze unutar liste favorita te podaci o stanju novčanika korisnika. Kako bi korisnik mogao koristiti funkcionalnost newslettera mora imati status prijavljenog korisnika.

- **Administrator -** Unutar aplikacije postoji jedan korisnik sa statusom administratora, takav korisnik je unaprijed definiran te on ima prava upravljanja korisnicima i brisanje istih.

5.3. Opis dizajna sustava

Opis dizajna sustava ćemo započeti dijagramom slučajeva korištenja gdje je detaljnije opisano korištenje same aplikacije. Na slici 6 vidimo prikaz dijagrama slučajeva korištenja na kojem postoje 3 sudionika, a to su administrator, običan korisnik te prijavljeni korisnik.



Slika 6 Dijagram slučajeva korištenja aplikacije [autorski rad]

Administrator sustava može kao što je već prije napomenuto upravljati korisnicima. Također administrator ima i sve ostale mogućnosti, a nasljeđuje ih od prijavljenog korisnika.

Prijavljeni korisnik ima opcije promjene lozinke, pretplate na newsletter odnosno e-mail izvještaje, dodavanje kripto valuta u favorite te funkciju novčanika. Funkcija newslettera uključuje konfiguriranje pretplate na način da se određuje interval slanja i podaci koje će izvještaj sadržavati. Funkcija novčanika uključuje dodavanje valuta u novčanik. Prijavljeni korisnik ima i sve mogućnosti običnog korisnika.

Običan ili neprijavljen korisnik ima opcije prijave i registracije, a uz to može pregledavati kripto valute i detaljnije podatke za pojedinu kripto valutu.

Slika 7 prikazuje dijagram komponenata na kojem možemo vidjeti najbitnije dijelove sustava i kako su ti dijelovi povezani. Aplikaciji se pristupa preko internetskog pretraživača gdje se nalazi korisničko sučelje, odnosno korisnička aplikacija. Drugi dio sustava predstavlja pozadinska aplikacija koja se sastoji od **6** servisa od kojih svaki izvršava jednu temeljnu funkciju sustava, a to su:

- **Account servis (korisnički račun) -** Služi za provođenje operacija nad korisnicima i korisničkim računima, kao što su prijava, registracija, promjena lozinke, itd.

- **Admin servis (administrator) -** Provodi operacije administratora aplikacije kao što su kontrola korisnika, kontrola kripto valuta, itd.

- **Crypto servis (kripto) -** Servis koji služi za provođenje operacija nad kripto valutama kao što su dohvat kripto valuta, dohvat detaljnih podataka pojedine kripto valute, dodavanja kripto valute u listu favorita korisnika, itd.

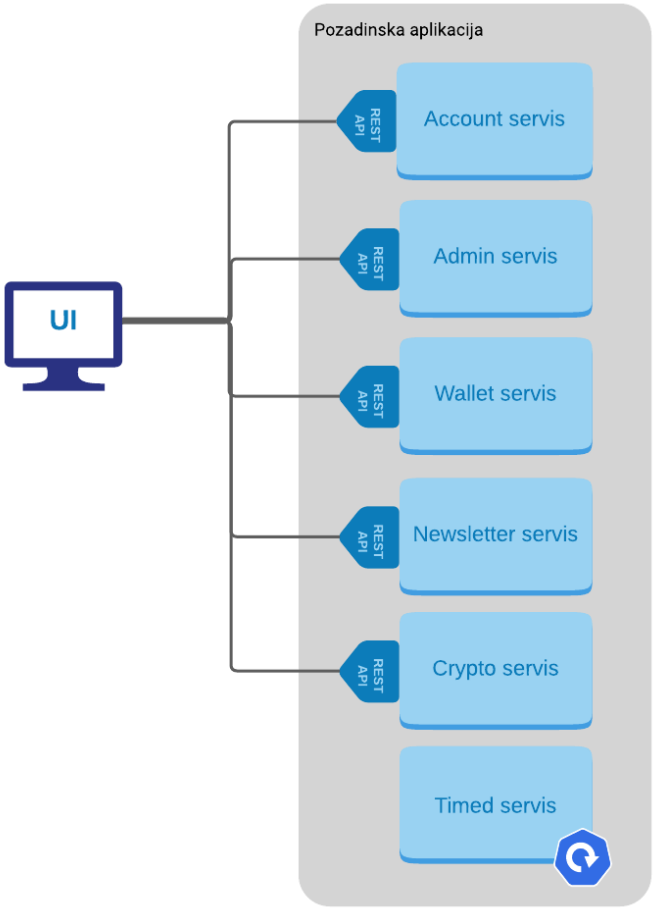
- **Wallet servis (novčanik)**

    - Ovaj servis služi kao potpora provođenja operacija nad novčanikom korisnika kao što su dodavanje kripto valuta u novčanik korisnika ili brisanje istih, računanje vrijednosti samog novčanika, itd.

- **Newsletter servis (pretplata na bilten) -** Služi za provođenje operacija nad automatskim obavijestima koje korisnik prima na svoju e-mail adresu. Operacije koje ovaj servis podržava su konfiguracija vremenskog intervala slanja obavijesti, konfiguracija podataka koje obavijest, odnosno izvještaj sadržava, itd.

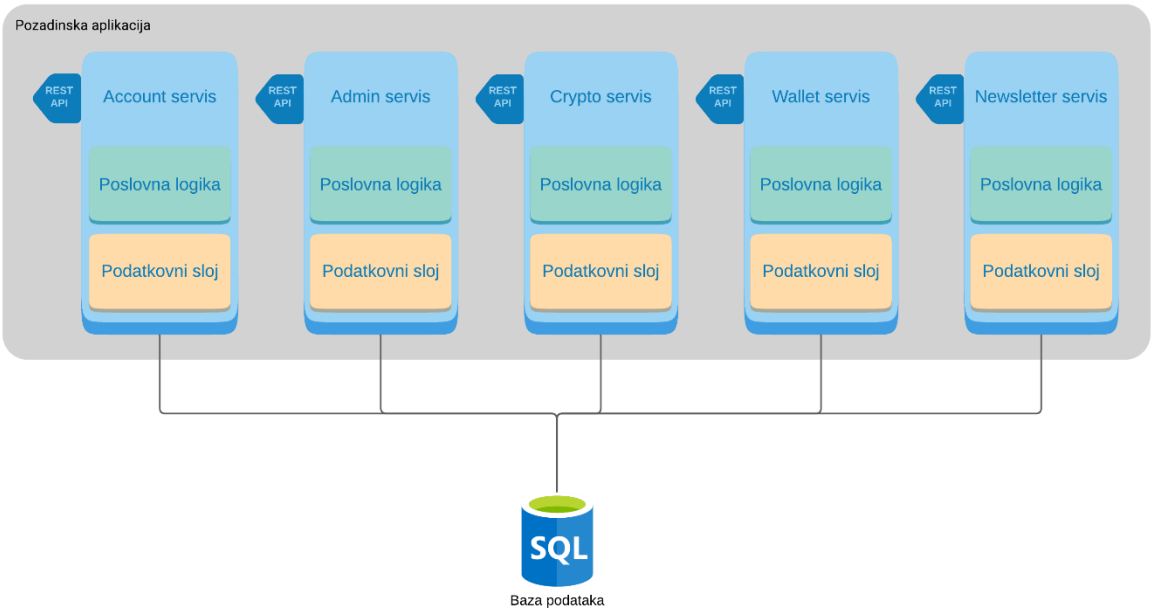
- **Timed servis -** Ovaj servis služi kao pozadinski proces koji ima svrhu periodičnog preuzimanja novih podataka o kripto valutama te spremanje istih u bazu podataka. Također ovaj servis svojim izvršavanjem periodički šalje e-mail obavijesti korisnicima ovisno o njihovoj konfiguraciji. O ovome će se servisu više govoriti kasnije u sekciji implementacije.

Svaki od prethodno navedenih servisa, osim Timed servisa, sadrži svoju vlastitu krajnju točku (eng. endpoint), odnosno REST API preko koje mu je moguće pristupati.



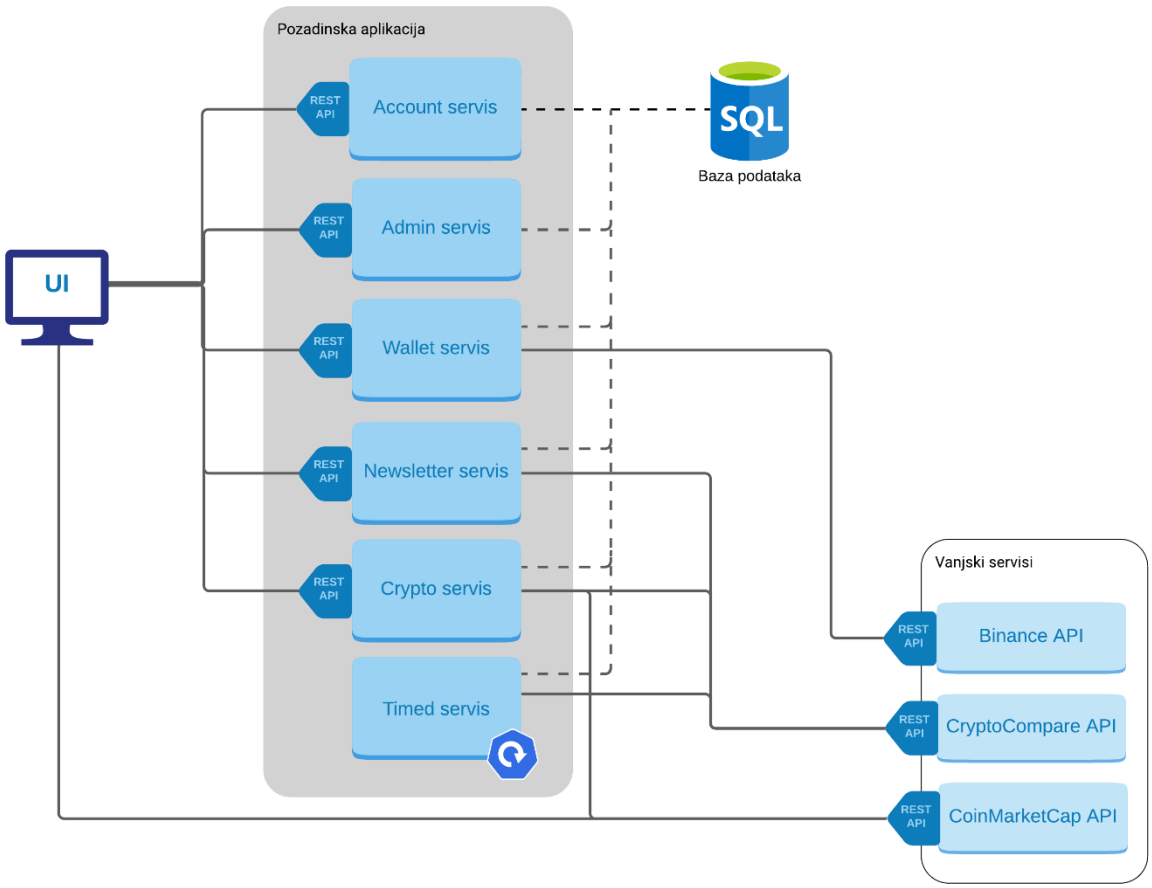
Slika 7 Dijagram komponenata [autorski rad]

Kao što je već prije spomenuto, pozadinska aplikacija se sastoji od ukupno 6 servisa, a njih je 5 implementirano u troslojnoj arhitekturi koja sadrži sloj REST API sučelja, sloj poslovne logike te sloj pristupa podacima ili podatkovni sloj. Navedenu arhitekturu možemo vidjeti na slici 8. Podatkovni sloj servisa služi za komunikaciju sa bazom podataka, a u slučaju ove aplikacije radi se o jednoj bazi podataka koju koriste svi servisi te sadrži sve potrebne podatke za rad aplikacije.



Slika 8 Arhitektura servisa pozadinske aplikacije [autorski rad]

Za dohvaćanje podataka osim baze podataka aplikacija koristi još 3 vanjska servisa koja će se kasnije detaljnije opisati, a radi se o REST API servisima koji nude dohvaćanje raznih podataka, od liste kripto valuta, vrijednosti kripto valuta pa sve do podataka o stvarnim novčanicima korisnika na drugoj platformi za trgovanje kripto valutama. Servisi, odnosno API-i koji se koriste su Binance API, CryptoCompare API te CoinMarketCap API. Slika 9 prikazuje glavne komponente sustava i njihovo međusobno korištenje.



Slika 9 Arhitektura sustava [autorski rad]

Dok se baza podataka koristi u svim servisima pozadinske aplikacije, vanjski resursi se koriste po potrebi u pojedinim servisima, a jednim dijelom i u korisničkoj aplikaciji.

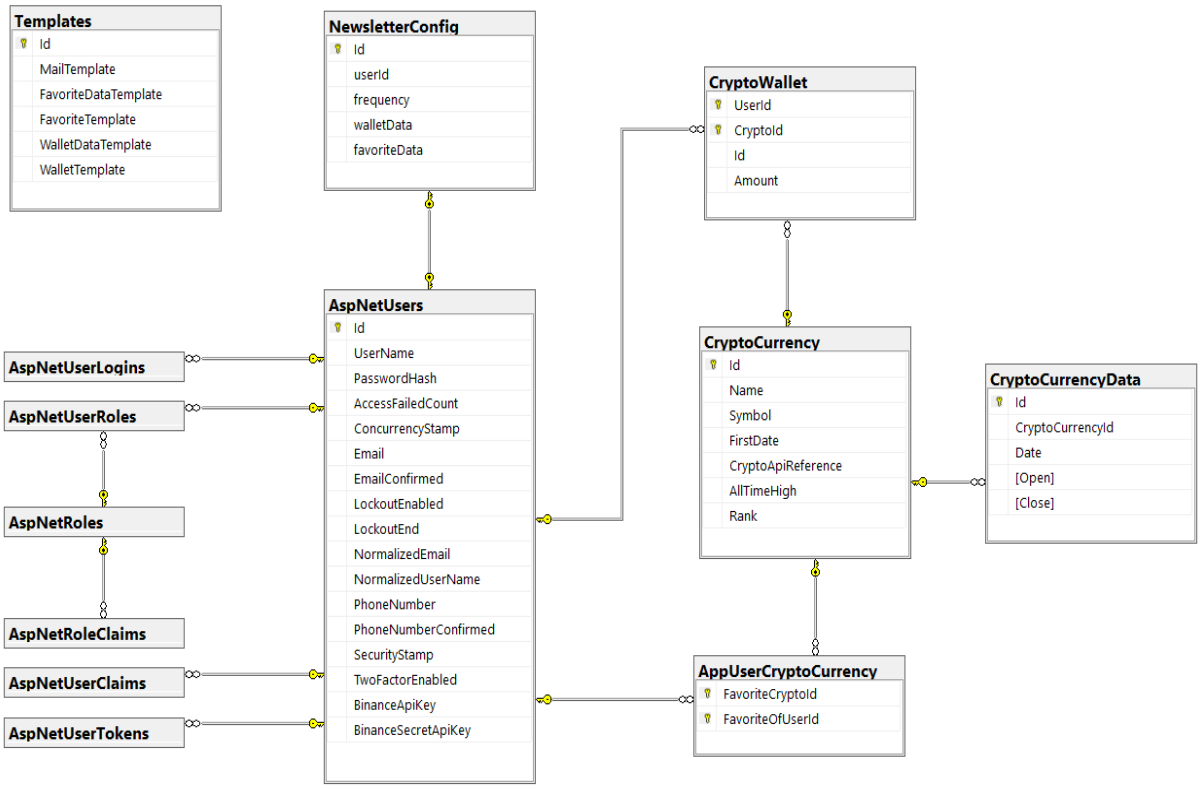
Binance API se koristi isključivo u Wallet servisu za novčanik korisnika gdje služi kao izvor podataka o stanju novčanika korisnika na istoimenoj platformi.

CryptoCompare API se koristi u tri servisa te predstavlja glavni izvor podataka o vrijednosti kripto valuta u danom vremenskom intervalu.

CoinMarketCap API se koristi za početno preuzimanje informacija o kripto valutama. Naime, istoimena je platforma u vrijeme pisanja ovog rada jedan od lidera za pregled informacija o aktualnim kripto valutama u svijetu. Osim za inicijalno preuzimanje podataka, ovaj API se također koristi i za preuzimanje simbola, odnosno ikona kripto valuta.

ERA modeli

Aplikacija za potrebe rada koristi SQL Server bazu podataka CryptoApp koja se sastoji od ukupno 13 tablica. Era model baze podataka prikazan je na slici 10.



Slika 10 ERA dijagram baze podataka CryptoApp [Autorski rad]

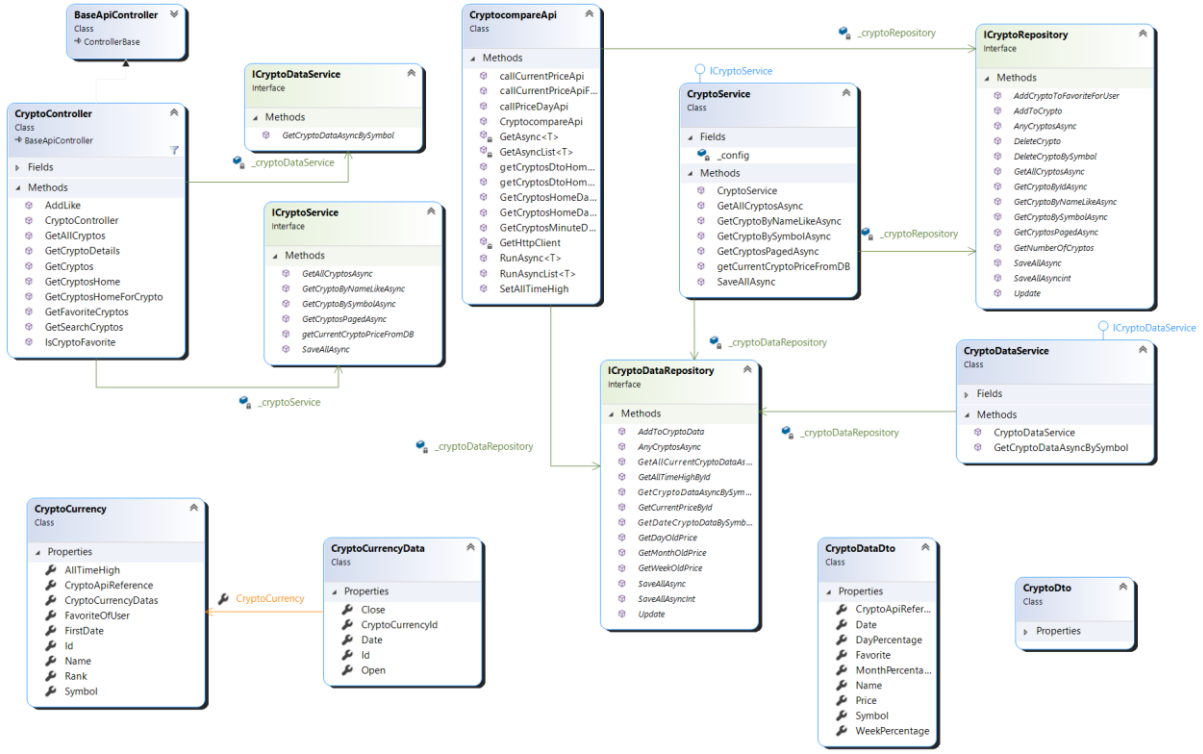
Ovih 13 tablica raspoređeno je u 4 skupina, a to su:

- Kategorija korisnik
  - U ovu kategoriju spadaju sve tablice sa prefiksom „AspNet“, a to su:
    - AspNetUserLogins
    - AspNetUserRoles
    - AspNetRoles
    - AspNetRoleClaims
    - AspNetUserClaims
    - AspNetUserTokens
    - AspNetUsers

    - Tablice u ovoj kategoriji kreirane su od strane ASP.NET Core Identity okvira. Ovo je okvir razvijen od strane Microsoft-a koji nudi funkcionalnosti potrebne za upravljanje korisnicima. Neke od funkcionalnosti korištenih u aplikaciji su prijava korisnika, dodavanje novog korisnika i generiranje tokena za potvrdu mail-a korisnika. Prilikom implementacije ASP.NET Core Identity-a okvir kreira ovih 7 osnovnih tablica potrebnih za rad.
- Kategorija kripto valuta
    - U ovoj su kategoriji tablice potrebne za rad sa kripto valutama, a to su:
        - CryptoCurrency
        - CryptoCurrencyData
        - AppUserCryptoCurrency
        - CryptoWallet
    - Tablica CryptoCurrency sadrži sve kripto valute i njihove atribute.
    - Tablica CryptoCurrencyData sadrži podatke o cijeni pojedine kripto valute u određeno vrijeme te jedna kripto valuta može imati više podataka o cijeni.
    - AppUserCryptoCurrency tablica služi kao pomoćna tablica u vezi više naprema više između tablice korisnika i tablice kripto valuta te predstavlja one kripto valute koje je korisnik dodao u vlastitu listu favorita.
    - Tablica CryptoWallet također služi kao pomoćna tablica u vezi više naprema više između tablice korisnika i tablice kripto valuta te predstavlja listu valuta u novčaniku korisnika. Ova tablica sadrži dodatne atribute o samoj količini kripto valuta u novčaniku korisnika.
- Newsletter - Tablica NewsletterConfig sadrži atribute koji opisuju konfiguraciju pojedinog korisnika o tome želi li primati e-mail obavijesti aplikacije, u kojem periodu te koje podatke želi primati.
- Templates - Tablica Templates je vrlo jednostavna tablica koja sadrži 5 atributa u kojima su zapisani HTML predlošci kojima se aplikacija služi prilikom generiranja e-mail obavijesti u Newsletter funkcionalnosti.

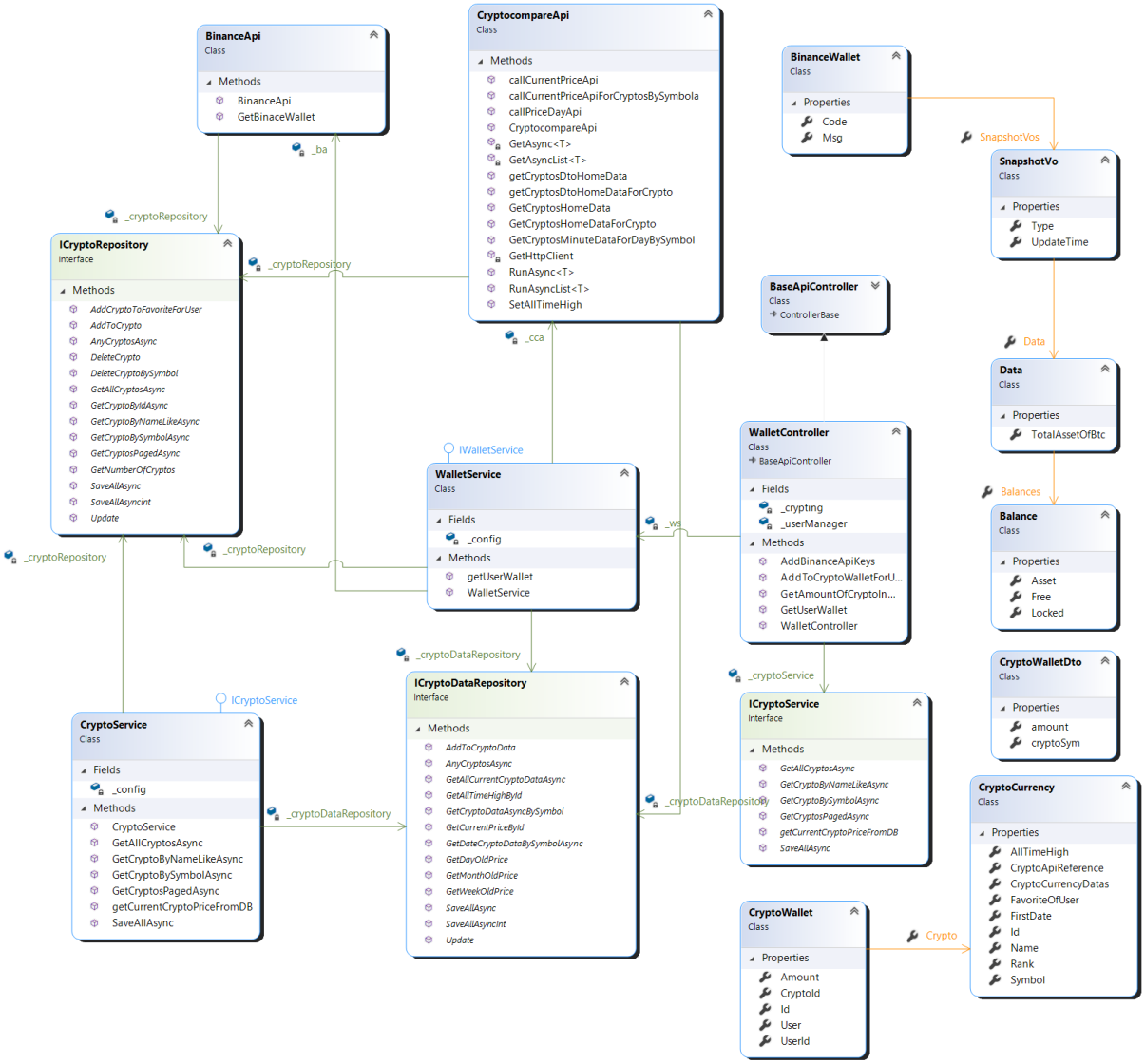
Dijagrami klasa

Slika 11 prikazuje dijagram klasa dijela aplikacije zadužene za rad sa kripto valutama te su prikazane najvažnije klase i njihove veze. Postoje dva sučelja koja definiraju rad sa kripto valutama, a to su ICryptoService i ICryptoDataService. Implementacije su ovih sučelja u klasama CryptoService i CryptoDataService. Za dohvaćanje podataka iz baze podataka koriste se dva repozitorija koji su definirani sučeljima ICryptoRepository i ICryptoDataRepository. Kontroler CryptoController poziva metode navedenih servisa. Svaki kontroler aplikacije nasljeđuje klasu BaseApiController kako bi svaki kontroler imao jednake postavke. Kako bi se pozivi servisa i repozitorija mogli koristiti potrebno je injektirati ovisnosti, odnosno implementirati uzorak dizajna poznatiji i kao injektiranje ovisnosti (eng. dependency injection). Na dijagramu su također prikazani i entiteti korišteni u navedenim pozivima.



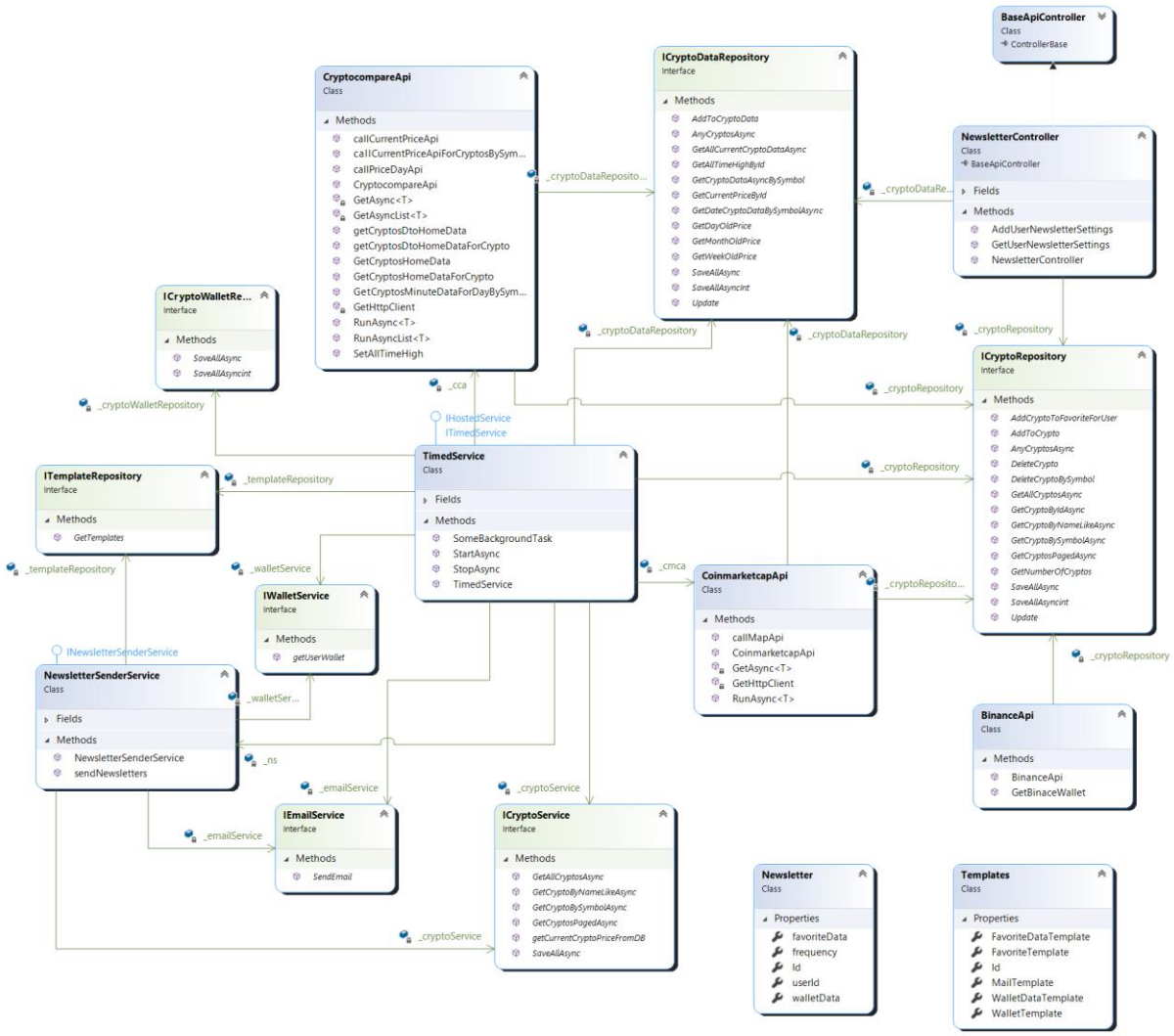
Slika 11 Dijagram klasa dijela aplikacije za kripto valute [autorski rad]

Slika 12 prikazuje dijagram klasa dijela aplikacije zadužene za rad sa novčanikom korisnika te su prikazane najvažnije klase i njihove veze. Implementacija rada sa novčanikom korisnika je u klasama CryptoService, WalletService i CryptocompareAPI. Za dohvaćanje podataka iz baze podataka koriste se dva repozitorija koji su definirani sučeljima ICryptoRepository i ICryptoDataRepository. Kontroler WalletController poziva metode navedenih servisa. Na dijagramu su također prikazani i entiteti korišteni u navedenim pozivima.



Slika 12 Dijagram klasa dijela aplikacija zadužene za novčanik korisnika [autorski rad]

Slika 13 prikazuje dijagram klasa dijela aplikacije zadužene za rad sa newsletterom te su prikazane najvažnije klase i njihove veze. Implementacija rada sa newsletterom je u klasama TimedService, NewsletterSenderService, CryptocompareAPI, BinanceAPI te CoinmarketcapApi. Za dohvaćanje podataka iz baze podataka koriste se četiri repozitorija koji su definirani sučeljima ICryptoRepository, ICryptoDataRepository, ICryptoWalletRepository i ITemplateRepository. Kontroler NewsletterController poziva neke metode navedenih servisa. Na dijagramu su također prikazani i entiteti korišteni u navedenim pozivima.



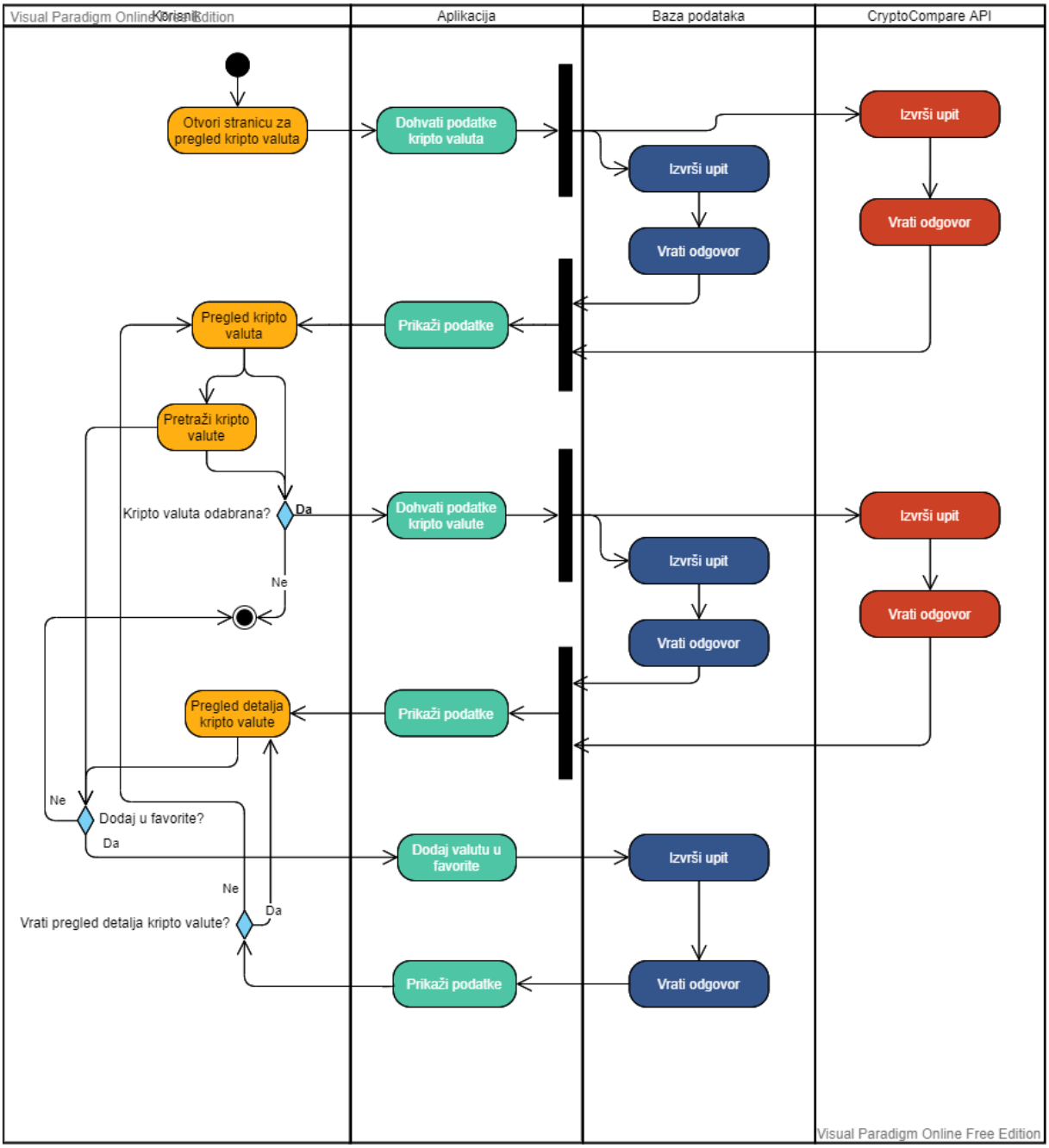
Slika 13 Dijagram klasa dijela aplikacije zadužene za rad sa newsletterom [autorski rad]

Dijagrami aktivnosti glavnih funkcionalnosti

U sljedećim dijagramima aktivnosti biti će prikazane glavne funkcionalnosti aplikacije, a to su funkcionalnost rada sa kripto valutama, funkcionalnost novčanika te funkcionalnost newsletter-a.

Funkcionalnost pregleda kripto valuta prikazana na slici 14 započinje otvaranjem stranice za pregled kripto valuta, što je u slučaju ove aplikacije i početna stranica. Korisniku se najprije prikaže korisničko sučelje koje se zatim puni podacima dohvaćenima iz baze podataka i vanjskog CryptoComapre API-a za dohvaćanje cijena kripto valuta. Na istoj stranici korisnik ima mogućnost odabira pojedine prikazane valute, a također valute može i pretraživati po nazivu te odabrati traženu valutu. Odabirom valute iz baze podataka i vanjskog CryptoCompare API-a se dohvaćaju detaljniji podaci o odabranoj kripto valuti koji

se tada prikazuju korisniku. Na bilo kojem od ovih pregleda korisnik može u bilo kojem trenutku određenu kripto valutu dodati u listu favorita.



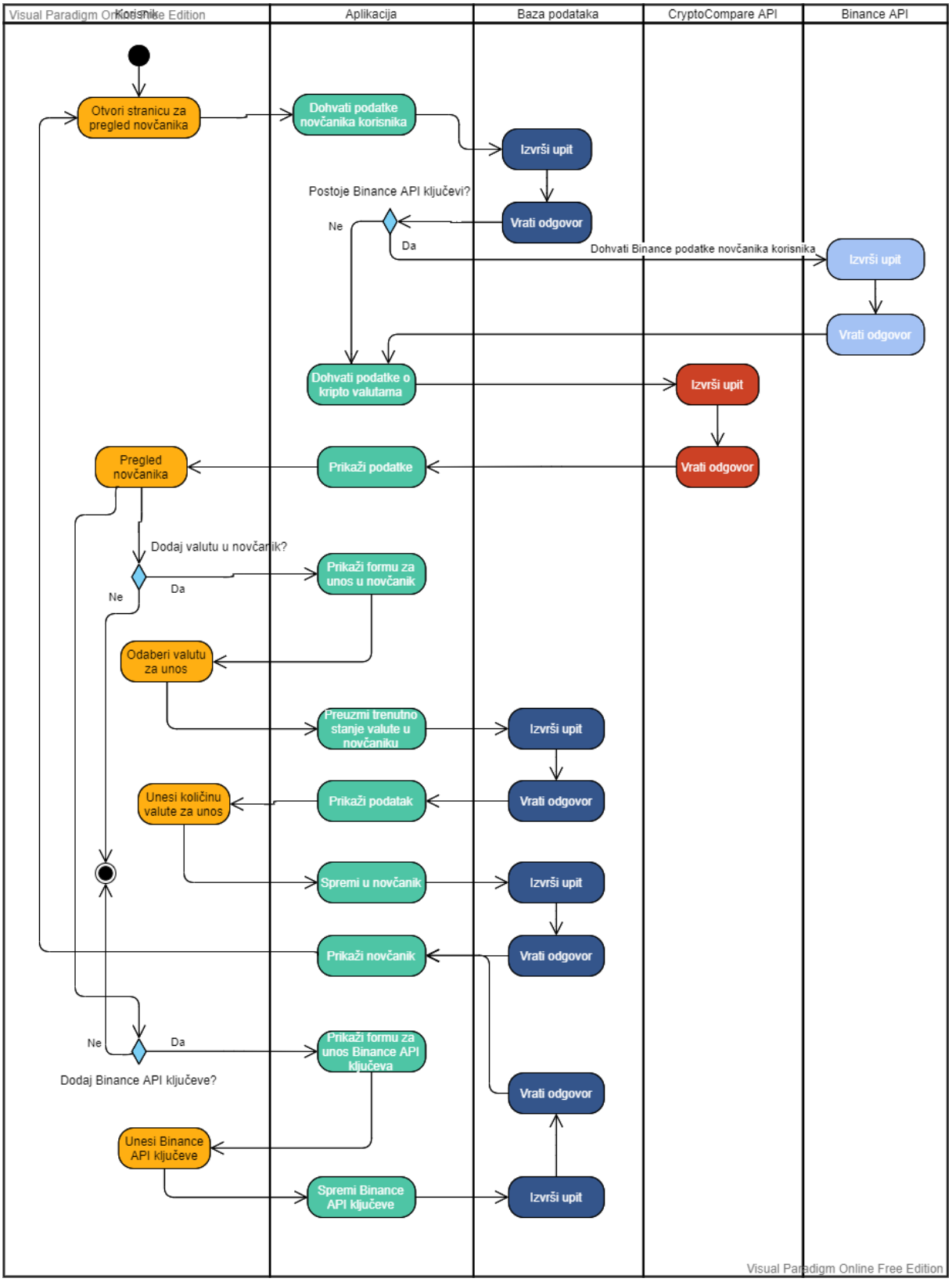
Slika 14 Dijagram aktivnosti za rad sa kripto valutama [autorski rad]

Funkcionalnost novčanika korisnika prikazana na slici 15 započinje otvaranjem stranice za pregled novčanika korisnika. Korisniku se najprije prikaže korisničko sučelje koje se zatim puni podacima dohvaćenima iz baze podataka i vanjskih API servisa Binance API te CryptoCompare API. Prilikom preuzimanja podataka najprije se iz baze podataka preuzimaju

podaci o novčaniku korisnika. Ukoliko u podacima novčanika korisnika postoje zapisani API ključevi Binance API-a tada se sa istoimenog API servisa preuzimaju podaci korisnikova novčanika na Binance platformi. Nakon što su preuzeti podaci o količinama kripto valuta u novčaniku korisnika unutar aplikacije i Binance novčaniku preuzimaju se podaci o cijenama kripto valuta od CryptoCompare vanjskog API-a. Dohvaćeni podaci se zatim prikazuju korisniku.

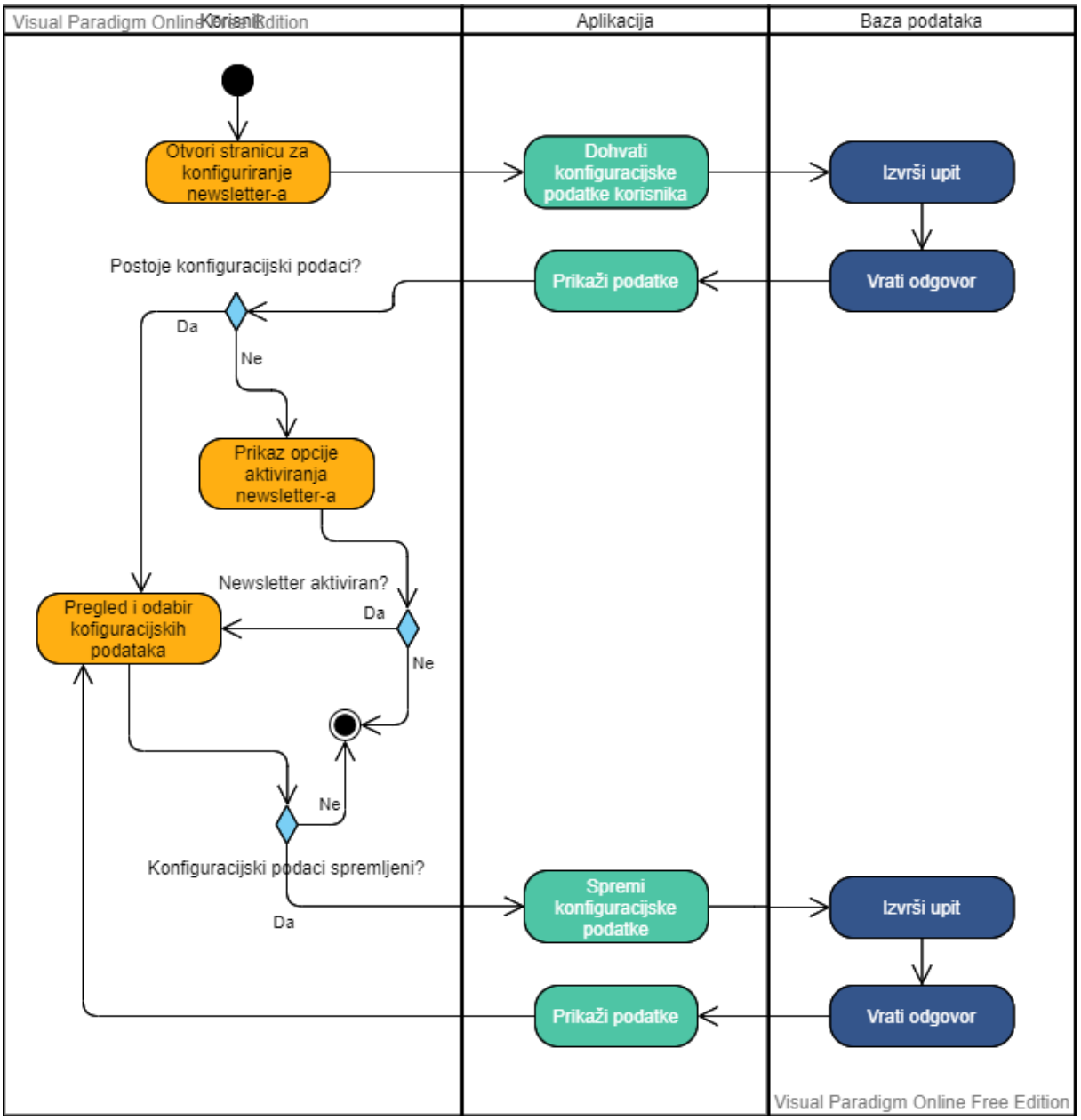
Korisnik ima mogućnost dodavanja valute u novčanik gdje se korisniku otvara forma za unos iste. Najprije korisnik odabire kripto valutu koju želi unijeti u novčanik. Aplikacija zatim iz baze podataka preuzima podatak o trenutnoj količini odabrane valute u novčaniku korisnika te taj podatak prezentira korisniku. Korisnik tada može upisati koliki iznos odabrane valute želi unijeti u novčanik. Unosom količine valuta se sprema u bazu podataka te se korisniku ponovo prikazuje pregled novčanika.

Osim mogućnosti dodavanja kripto valuta u novčanik korisniku se nudi i opcija povezivanja aplikacije sa Binance novčanikom korisnika. Ova mogućnost dostupna je samo onim korisnicima koji se već nisu povezali sa Binance novčanikom. Korisniku se najprije prikazuje forma za unos Binance API ključeva čijim se spremanjem podaci upisuju u bazu podataka te se korisniku ponovo prikazuje pregled novčanika.



Slika 15 Dijagram aktivnosti za rad sa novčanikom [autorski rad]

Funkcionalnost newsletter-a prikazana na slici 16 započinje otvaranjem stranice za pregled konfiguracije newsletter-a. Korisniku se najprije prikaže korisničko sučelje koje se zatim puni podacima dohvaćenima iz baze podataka. Ukoliko već postoji odabrana konfiguracija zapisana u bazi podataka korisniku je tada moguće mijenjati podatke konfiguracija. Ukoliko korisnik nema spremljenu konfiguraciju, odnosno nema aktiviranu opciju primanja newsletter-a tada je forma za korisnika skrivena i dostupna mu je samo opcija aktiviranja primanja newsletter-a čijim se aktiviranjem prikazuje navedena forma. Nakon odabira željene konfiguracije konfiguracija se sprema u bazu podataka te se korisniku ponovo prikazuje pregled konfiguracije newsletter-a.



Slika 16 Dijagram aktivnosti za rad sa newsletterom [autorski rad]

5.4. Pozadinska aplikacija

U ovom će dijelu rada biti opisan proces izrade najvažnijih dijelova aplikacije. Najprije će se opisati način izrade i upravljanja bazom podataka, zatim repozitoriji, kontroleri i servisi. Detaljnije će se opisati i način autorizacija, način grafičkog prikaza podataka, kriptiranje osjetljivih podataka korisnika te uporaba ASP.NET Core Identity okvira. Osim samog razvoja objasnit će se osnove postavljanja same aplikacije na server.

5.4.1. Baza podataka

Za implementaciju ove aplikacije odabran je pristup implementacije zvan „Code-First“ kod kojega se najprije u kodu definiraju entiteti i potrebne klase te se kasnije baza podataka usklađuje sa onim u kodu. Upravo ovakav način razvoja omogućuje Entity Framework Core. U nastavku je prikazan entitet `CryptoCurrency` koji predstavlja pojedinu kripto valutu:

```
public class CryptoCurrency
{
    public int Id { get; set; }
    public int Rank { get; set; }
    public int CryptoApiReference { get; set; }
    public string Name { get; set; }
    public string Symbol { get; set; }
    public DateTime FirstDate { get; set; }
    public ICollection<CryptoCurrencyData> CryptoCurrencyDatas { get; set; }
    public ICollection<AppUser> FavoriteOfUser { get; set; }
    public decimal AllTimeHigh { get; set; }
}
```

U ovakvom pristupu ovakav entitet predstavlja tablicu u bazi podataka. Želimo li kreirati tablicu sa navedenim atributima u bazi podataka potrebno je u klasi koja nasljeđuje klasu `DbContext` dodati set kojim se definira tablica u bazi podataka. U slučaju ove aplikacije takva se klasa zove `DataContext`.

Možemo primijetiti kako navedeni entitet sadrži kolekciju entiteta `CryptoCurrencyData` koji predstavlja podatke pojedine kripto valute. Ovaj je entitet prikazan u nastavku:

```
public class CryptoCurrencyData
    {
        public int Id { get; set; }
```

```
        public CryptoCurrency CryptoCurrency { get; set; }
        public int CryptoCurrencyId { get; set; }
        public DateTime Date { get; set; }
        public decimal Open { get; set; }
        public decimal Close { get; set; }
    }
```

U ovom entitetu možemo primijetiti kako ovaj entitet sadrži objekt tipa `CryptoCurrency` te integer atribut jednakog naziva uključujući sufiks `Id`. Ovakav raspored atributa i konvencija imenovanja u ova dva entiteta određuje kako se radi o tablicama više naprema jedan, gdje jedna kripto valuta može imati više podataka, a dok jedan podatak može pripadati samo jednoj kripto valuti. Želimo li kreirati navedene tablice u bazi podataka potrebno je dodati sljedeći set u klasu `DataContext`:

```
public DbSet<CryptoCurrency> CryptoCurrency { get; set; }
public DbSet<CryptoCurrencyData> CryptoCurrencyData { get; set; }
```

Želimo li stvarno i obaviti promjene na samoj bazi podataka potrebno je kreirati migraciju pomoću koje Entity Framework Core izvršava operacije promjene baze podataka. Migraciju kreiramo pomoću naredbe `dotnet ef migrations add <naziv_migracije>`. Ovom komandom Entity Framework Core će kreirati migraciju na temelju svega prethodno napomenutog. Migracija sadrži SQL kodove kojima će se obaviti potrebne promijene nad bazom podataka. Kako bismo samu migraciju izvršili na bazi podataka potrebno je izvršiti naredbu `dotnet ef database update`.

Želimo li dohvatiti podatke iz baze podataka potrebno je koristiti repozitorij gdje je implementirano čitanje i pisanje u bazi podataka. U klasi repozitorija injektiran je kontekst potreban za rad s bazom podataka. U sljedećem primjeru prikazana su dva primjera kako u repozitoriju za kripto valute CryptoRepository koji implementira sučelje ICryptoRepository možemo dohvatiti sve kripto valute u bazi podataka te kako možemo dohvatiti kripto valutu na temelju simbola.

```
return await _context.CryptoCurrency.ToListAsync();
```

```
return await _context.CryptoCurrency.SingleOrDefaultAsync(c => c.Symbol == symbol);
```

5.4.2. Servisi

Kako bi se kreirani servisi mogli koristiti potrebno je unutar ConfigureServices metode u klasi Startup.cs definirati sam servis, a to za primjer servisa za rad sa kripto valutama izgleda ovako:

```
services.AddTransient<ICryptoService, CryptoService>();
```

Prilikom definiranja samog servisa potrebno je navesti i sučelje i servis koji ga implementira. Komponente koje zahtijevaju određene servise od sustava ne zahtijevaju servise direktno, već zahtijevaju servis na temelju sučelja koje taj servis treba implementirati, a upravo ovakvim definiranjem servisa zajedno sa njihovim sučeljem omogućuje aplikaciji ispravno injektiranje traženih servisa.

Da bi smo mogli koristiti kreirane repozitorije u servisima aplikacije najprije je potrebno injektirati potrebne repozitorije. Primjer metode za dohvaćanje svih kripto valuta u servisu `CryptoService` izgleda ovako:

```
public async Task<List<CryptoCurrency>> GetAllCryptosAsync()
  {
      return await _cryptoRepository.GetAllCryptosAsync();
  }
```

Aplikacija se sastoji od 6 glavnih servisa koji su potrebni za rad aplikacije, a to su:

- CryptoService – zadužen za obradu kripto valuta
- CryptoDateService – zadužen za obradu podataka kripto valuta
- EmailService – zadužen za obradu slanja mail-a
- NewsletterSenderService – zadužen za obradu slanja i kreiranja newslettera
- TimedService – zadužen za periodično obavljanje zadataka preuzimanje podataka kripto valuta i slanja newslettera. Ovaj servis nasljeđuje klasu IHostedService što omogućuje pokretanje servisa prilikom kreiranja same aplikacije. Navedeni servis se pokreće unutar datoteke Program.cs u metodi IHostBuilder dodavši sljedeći dio koda:

  ```
  .ConfigureServices(services =>
   {
       services.AddHostedService<TimedService>();
   });
  ```

- WalletService – zadužen za obradu podataka o novčaniku korisnika

5.4.3.Kontroleri

Kontroleri su zaduženi za primanje, izvršavanje i vraćanje odgovora na API zahtjeve odgovarajuće krajnje točke. Svaki kontroler aplikacije nasljeđuje BaseApiController koji pak nasljeđuje ControlerBase. Nasljeđivanje BaseApiController-a automatski određuje kontroleru anotacije [ApiController] i [Route(„[controller]“)]. Anotacija ApiController naznačuje kako je njegova svrha posluživanje HTTP API zahtjeva. Anotacija Route određuje URL putanju gdje se nalazi kontroler, a anotacija zadana na ovakav način naznačuje kako se svaki kontroler nalazi na adresi „/<nazivKontrolera>“.

Osim ovih zadanih anotacija, kontroler može biti definiran mnogim drugim anotacijama, a u ovoj aplikaciji još je bitno napomenuti anotaciju [Authorize] za definiranje zahtjeva autorizacija.

Aplikacija sadrži 4 najvažnijih kontrolera, a to su:

- AccountController – zadužen za obradu korisnika
- CryptoController – zadužen za obradu kripto valuta i njihovih podataka
- NewsletterController – zadužen za obradu newslettera
- WalletController – zadužen za obradu novčanika korisnika

U nastavku je prikaz dijela CryptoControllera zaduženog za obradu API zahtjeva za kripto valute, a prikazana je API krajnja točka na adresi „`/Crypto/{simbolValute}`“ zaduženu za dodavanje kripto valute u favorite korisnika. Radi se o POST metodi koja zahtjeva autorizaciju korisnika.

```
public class CryptoController : BaseApiController
    {


        public CryptoController(ICryptoService cryptoService, …)
        {
            _cryptoService = cryptoService;
        }



       [Authorize]
       [HttpPost("{cryptoSym}")]
        public async Task<ActionResult> AddLike(string cryptoSym)
        {

            var sourceUser = await _userManager.Users
                .Include(r => r.FavoriteCrypto)
                .Where(u => u.Id == User.GetUserId())
                .SingleOrDefaultAsync();


          var crypto = await _cryptoService.GetCryptoBySymbolAsync(cryptoSym);
```

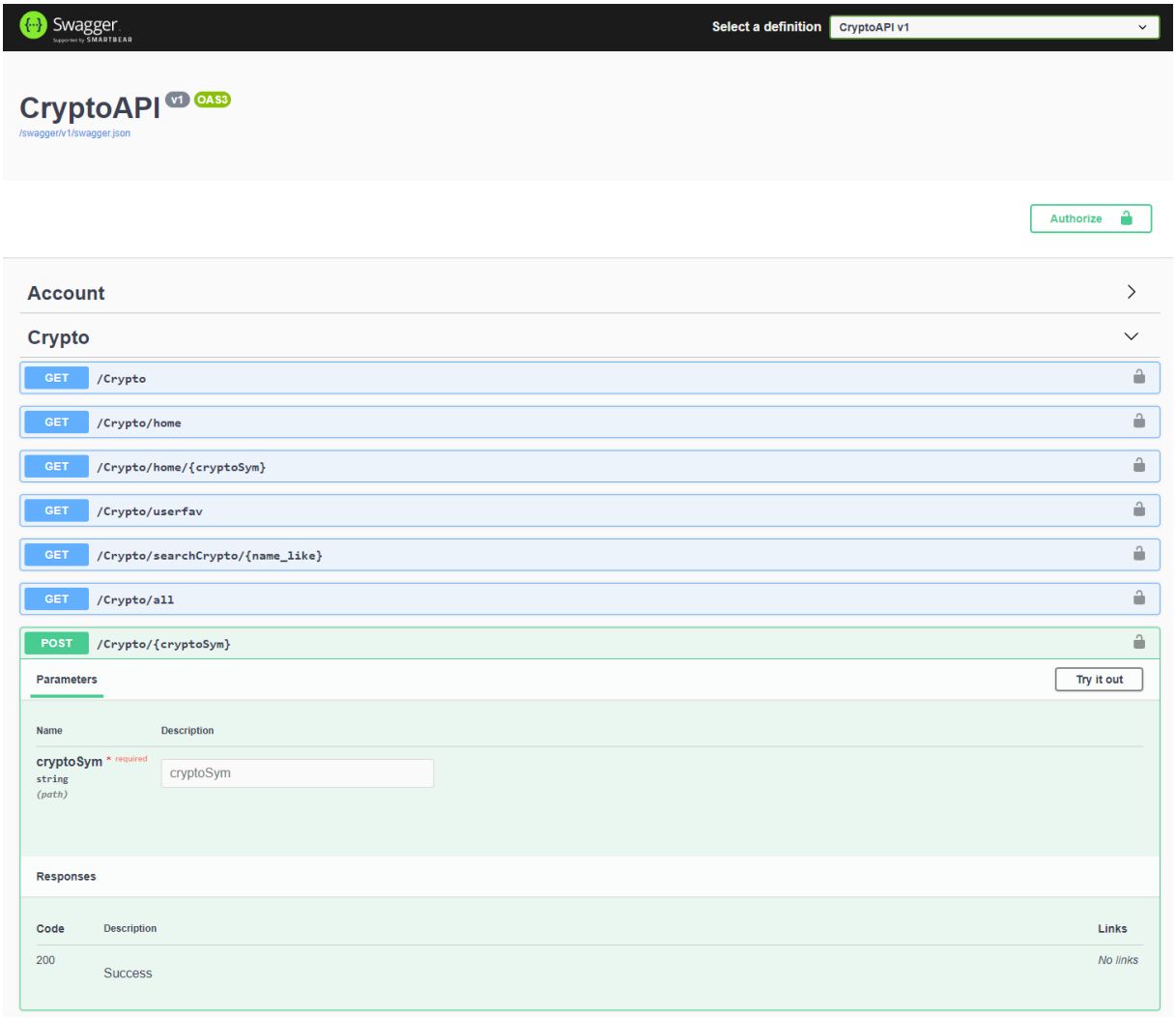
```
            if (sourceUser.FavoriteCrypto.Contains(crypto))
            {
                sourceUser.FavoriteCrypto.Remove(crypto);
            }
            else
            {
                sourceUser.FavoriteCrypto.Add(crypto);
            }


            await _cryptoRepository.SaveAllAsync();


            return Ok();

        }

    }
```

5.4.4. Autorizacija

Kao autorizacija prilikom pristupanja resursima koji istu zahtijevaju koristi se JSON web token kao Bearer shema. Kako bi autorizacija bila valjana potrebno je svakom HTTP zahtjevu u zaglavlju dodati autorizacijsko zaglavlje koje sadrži token dobiven od same aplikacije prilikom uspješne prijave u sustav. Takav token kada ga korisnička aplikacija u našem slučaju Angular aplikacija primi, ona ga sprema u lokalno spremište te ga po potrebi koristi.

5.4.5. Dokumentiranje i testiranje pozadinske aplikacije

Kao alat za dokumentiranje samih API-a pozadinske aplikacije korišten je alat Swagger koji dolazi već predefiniran prilikom samog kreiranja projekta. Definiranje Swagger-a obavlja se u Startup.cs klasi. Jednom kada pokrenemo aplikaciju dostupna nam je i sama Swagger stranica koja prikazuje osnovne informacije o API-u kao što su krajnje točke, podaci koje pojedine krajnje točke očekuju te odgovori koje vraćaju. Swagger stranica također nudi i testiranje samih API-a. Na slici 17 prikazan je dio Swagger stranice ove aplikacije.



Slika 17 Swagger ekran aplikacije [autorski rad]

5.5. Korisnička aplikacija

Korisničko sučelje aplikacije napravljeno je u razvojnom okviru Angular koristeći biblioteke kao što su ngx-bootstrap i Angular Material UI za mnoge osnovne komponente aplikacije. Arhitektura ovakve aplikacije je temeljena na komponentama kojima se postepeno gradi cijela korisnička aplikacija. Kao pristup podacima Angular komponente pristupaju API krajnjim točkama pozadinske aplikacije te na taj način dohvaćaju potrebne podatke. Takva je komunikacija sa API-ima implementirana u servisima kojih u ovoj aplikaciji ima 6, a to su:

- Account.service - Ovaj je servis zaslužan za upravljanje korisnikom, kao što je prijava korisnika, odjava, registracija te potvrda maila

- Admin.service - Ovaj je servis zadužen za operacije admina, a to je postavljanje uloga korisnicima te upravljanje korisnicima
- Crypto.service - Ovaj je servis zadužen za upravljanje kripto valutama, kao što je dohvaćanje kripto valuta, dohvaćanje podataka o kripto valutama ili pak dodavanje kripto valuta u listu favorita korisnika
- Newsletter.service - Ovaj je servis zadužen za upravljanje konfiguracijskim podacima funkcionalnosti newsletter za korisnika
- Wallet.service - Ovaj je servis zadužen za upravljanje novčanikom korisnika kao što je dodavanje valuta u novčanik, dohvaćanje podataka o novčaniku korisnika ili pak unos API ključeva korisnika za Binance API

Servisi se unutar Angular aplikacije injektiraju u ostale komponente aplikacije kako bi im se moglo pristupati i koristiti. Kako bi se servise moglo injektirati u ostale komponente potrebno im je postaviti dekorator `@Injectable`. Definiranje servisa Crypto.service izgleda ovako:

```
@Injectable({
  providedIn: 'root'
})
export class CryptoService {…}
```

Ovako se definiran servis može injektirati kroz konstruktor ostalim komponentama aplikacije.

Kako bi servis Angular aplikacije mogao izvršavati pozive prema krajnjim točkama API-a pozadinske aplikacije potrebno je u sam servis injektirati klasu HttpClient koja sadrži metode potrebne za izvršavanje HTTP zahtjeva. Injektiranje ove klase izvršava se u konstruktoru servisa te izgleda ovako:

```
constructor(private http: HttpClient) { }
```

Kako bi se u servisu mogla pozivati prethodno opisana API krajnja točka pozadinske aplikacije na adresi „`/Crypto/{simbolValute}`“ definirana je metoda servisa koja izgleda ovako:

```
addToFavorite(crypto: string){
    return this.http.post(this.baseUrl + 'Crypto/' + crypto, {});
  }
```

Pošto se ovdje radi o POST metodi potrebno je pozvati post metodu HttpClient klase kojoj se prosljeđuje URL adresa na kojoj se nalazi željena API krajnja točka te željeno podaci odnosno parametri. U ovom primjeru zahtjev očekuje samo parametar putanje dok se u tijelu poruke prosljeđuje prazan objekt. Ovakva metoda vraća objekt tipa Observable te je potrebno da se komponenta koja poziva ovu metodu servisa pretplati te će jednom kada podaci stignu biti obaviještena. Ovakav način rada omogućuje bolju responzivnost aplikacije te asinkroni rad. U ovom se dijelu može prepoznati Observer uzorak dizajna.

Poziv ove metode iz druge komponente aplikacije u koju je ovaj servis injektiran izgleda ovako:

```
this.cryptoService.addToFavorite(crypto.symbol).subscribe(() => {
      this.toastr.success(this.msg);
    });A
```

U gore prikazanom pozivu metode servisa komponenta aplikacije se pretplaćuje na metodu servisa pomoću metode subscribe te po primitku odgovora ispisuje odgovarajuću poruku. Ispis poruke odrađuje Toastr servis također injektiran u istu komponentu.

U primjeru ispod prikazan je programski kod metode Crypto servisa zaduženog za poziv API krajnje točke pozadinske aplikacije na adresi „/Crypto/Home".

```
getCryptosCurrentPrice(page?: number, itempPerPage?: number) {

    let params = new HttpParams();

    if(page !== null && itempPerPage !== null){
      params = params.append('pageNumber', page.toString());
      params = params.append('pageSize', itempPerPage.toString());
    }
    return this.http.get<CryptoCurrentData[]>(this.baseUrl + 'Crypto/home', {ob
serve: 'response', params}).pipe(
      map(response => {
        this.paginatedResult.result = response.body;
        if(response.headers.get('Pagination') !== null) {
          this.paginatedResult.pagination = JSON.parse(response.headers.get('Pa
gination'));
        }
        return this.paginatedResult
      })
    )

  }
```

Ovdje se radi o GET metodi koja vraća podatke kripto valuta. Ova metoda prima skup URL parametara kojim se određuje paginacija zahtjeva, odnosno određuje se koliko zapisa te koju po redu stranicu aplikacija zahtjeva. Primjer ovakvih URL parametara izgleda ovako:

```
?pageNumber=3&pageSize=10
```

Navedeni parametri određuju kako aplikacija zahtjeva treću stranicu te 10 zapisa, odnosno 10 kripto valuta po stranici. Ovakav će se zahtjev u pozadinskoj aplikaciji izvršiti na način da će se dohvatiti podaci o 10 kripto valuta na način da se preskoči prvih 20.

Poziv ove metode servisa u drugoj komponenti aplikacije izgleda ovako:

```
      this.cryptoService.getCryptosCurrentPrice(this.pageNumber, this.pageSize)
.subscribe(response => {
            this.cryptosCurrentData = response.result;
            this.pagination = response.pagination;
          })
```

Metodi servisa najprije se prosljeđuju dva parametra, a to su koji po redu stranicu zapisa zahtjeva te koliko zapisa svaka stranica sadržava. Nakon toga komponenta se pretplaćuje te po primitku odgovora rezultat, odnosno primljene podatke sprema u lokalnu varijablu `cryptosCurrentData`. Podatke o paginaciji također sprema u lokalnu varijablu `pagination`. Varijabla `cryptosCurrentData` je javna varijabla kojoj sada može pristupati HTML datoteka komponente gdje se navedenoj varijabli može pristupiti na način da se unutar dvije vitičaste zagrade definira naziv varijable, odnosno objekta i njegovih atributa. Isječak HTML datoteke gdje je prikazan navedeni koncept izgleda ovako:

```
<tbody>
    <tr class="pointer" *ngFor='let crypto of cryptosCurrentData'>
      <td [routerLink]="['/crypto', crypto.symbol]">
      <img style="…"
      src="https://s2.coinmarketcap.com/.../{{crypto.cryptoApiReference}}.png"
            class="…" alt="...">{{crypto.symbol}} - {{crypto.name}}
      </td>
      <td [routerLink]="['/crypto', crypto.symbol]">
        <h5><span class="…">${{crypto.price}}</span></h5>
      </td >
      <td  [routerLink]="['/crypto', crypto.symbol]" [ngStyle]="…">
           {{crypto.dayPercentage}}%
      </td>
      <td  [routerLink]="['/crypto', crypto.symbol]" [ngStyle]="…">
           {{crypto.weekPercentage}}%
      </td>
      <td  [routerLink]="['/crypto', crypto.symbol]" [ngStyle]="…">
           {{crypto.monthPercentage}}%
      </td>

        <td *ngIf="(accountService.currentUser$ | async)">
          <button (click)="addToFavorite(crypto)" *ngIf="!crypto.favorite">
            <i style="…" tooltip="Add to favorites"></i>
          </button>
          <button (click)="addToFavorite(crypto)" *ngIf="crypto.favorite">
            <i style="…" tooltip="Remove from favorites"></i>
          </button>
        </td>
    </tr>
  </tbody>
</table>
```

Prethodno prikazani dio koda zaslužan je za generiranje tablice koja sadrži redak za svaku kripto valutu u varijabli `cryptosCurrentData`. Korištene su dvije strukturne direktive Angular razvojnog okvira, a to su ngIf te ngFor. Direktiva `ngFor` primijenjena na elementu retka tablice `<tr>` određuje kako će se za svaku kripto valutu varijable `cryptosCurrentData` kreirati novi redak tablice te izgleda ovako:

```
<tr class="pointer" *ngFor='let crypto of cryptosCurrentData'>
```

Unutar svakog novo generiranog retka kripto valuti kojoj pripada zadani redak i njezinim atributima možemo pristupiti pomoću varijable `ngFor` direktive `crypto`. Direktiva `ngIf` primijenjena na elementu ćelije retka `<td>` određuje kako će se zadana ćelija generirati ovisno o istinitosti zadanog izraza koji u gore navedenom slučaju provjerava da li je korisnik aplikacije prijavljen te ukoliko je prijavljen aplikacija će generirati ćelije tablice koje sadrže opciju dodavanja pojedine kripto valute u listu favorita korisnika. Dio koda direktive `ngIf` izgleda ovako:

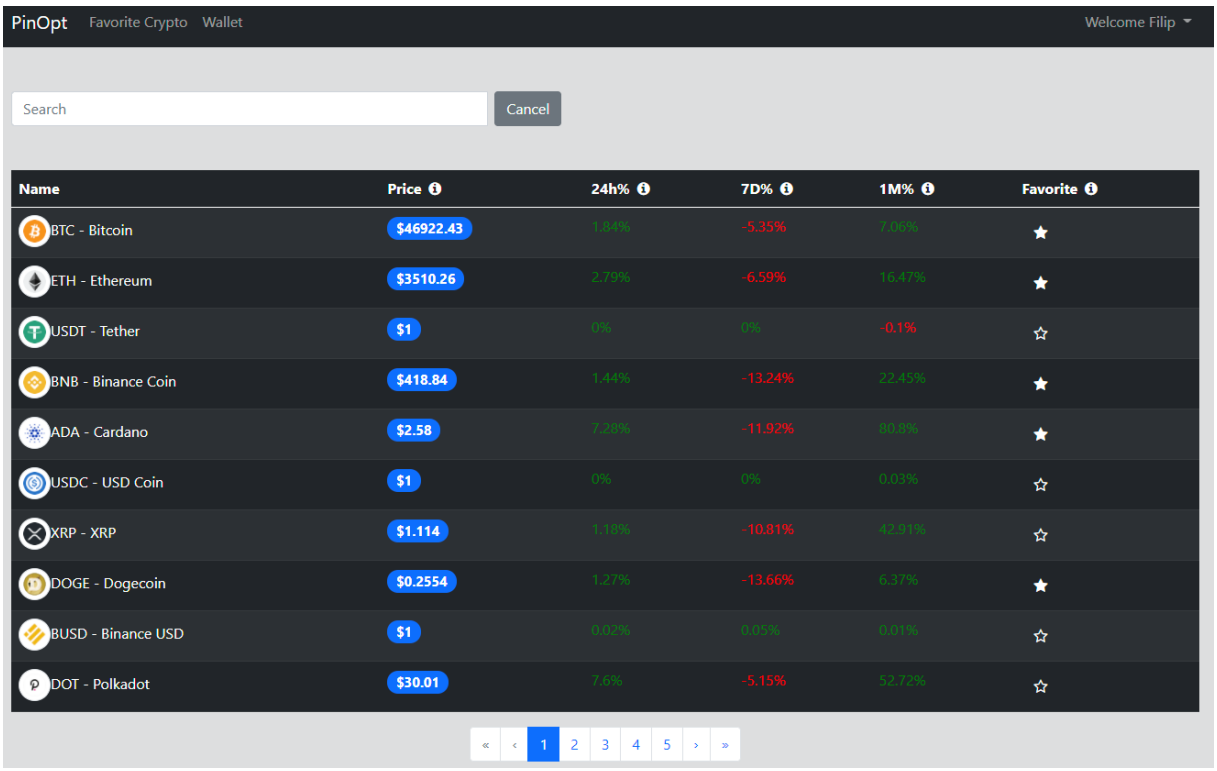
```
<td *ngIf="(accountService.currentUser$ | async)">
```

Također u gore prikazanom kodu korišteno je i vezivanje događaja, odnosno radnji korisnika na pojedine elemente te je tako na elementi gumba `<button>` vezan događaj pritiska koji u slučaju pritiska korisnika na zadani element izvršava prethodno prikazanu metodu komponente `addToFavorite()` kojoj prosljeđuje samu kripto valutu za koju je radnja obavljena. Slika 18 prikazuje tablicu generiranu na temelju prethodno prikazanog koda.

Name	Price	24h%	7D%	1M%	Favorite
BTC - Bitcoin	$46904.43	1.35%	-6.22%	7.02%	★
ETH - Ethereum	$3505.4	2.94%	-7.17%	16.31%	★
USDT - Tether	$1	0%	0%	-0.1%	☆
BNB - Binance Coin	$416.39	1.29%	-15.42%	21.74%	★
ADA - Cardano	$2.596	8.12%	-14.35%	81.92%	★

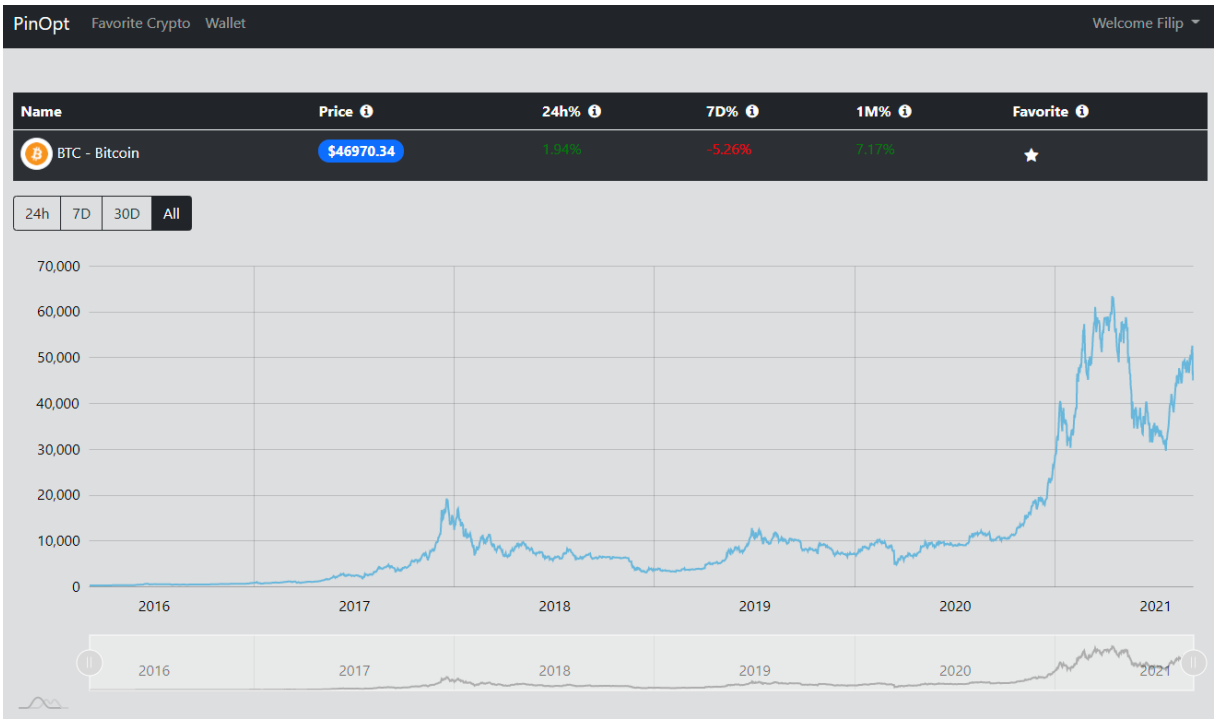
Slika 18 Dio stranice generiran kodom [autorski rad]

Korisnička se aplikacija sastoji od 4 glavnih ekrana, odnosno prozora, a to su početni ekran, ekran prikaza detaljnih informacija odabrane kripto valute, ekran pregleda novčanika korisnika te ekran konfiguracije newslettera. U nastavku će se detaljnije opisati rad kroz aplikaciju te najbitniji elementi same aplikacije. Na vrhu svakog ekrana nalazi se alatna traka preko koje se u bilo kojem trenutku može doći na početni ekran, ekran novčanika, ekran favorita te ekran konfiguracije newslettera. Za razliku od prijavljenog korisnika koji ima dozvoljen pristup svim navedenim ekranima aplikacije, neprijavljen korisnik ima pristup samo do početnog ekrana i ekrana pregleda detaljnih informacija odabrane kripto valute. Na slici 19 prikazan je početni ekran aplikacije koji sadrži tablični prikaz svih kripto valuta za koje se u aplikaciji vode podaci. Podaci su na ovoj stranici straničeni te je navigacija kroz straničenje moguća kroz kontrole ispod tablice. Također, iznad same tablice nalazi se polje za pretraživanje kripto valuta prema nazivu. U tablici su ispisani podaci o kripto valutama kao što je naziv, trenutna cijena, informacija da li se pojedina kripto valuta nalazi u listi favorita korisnika te se također nudi informacija o postotnoj promijeni cijene u odnosu na onu od prije 24 sata, 7 dana i mjesec dana. Korisniku se nudi opcija odabira pojedine kripto valute bilo u tablici ili preko polja za pretraživanje kripto valuta te se korisniku odabirom kripto valute prikazuje ekran detaljnih informacija o kripto valuti.



Slika 19 Prikaz početnog ekrana aplikacije [autorski rad]

Korisniku se na početnoj stranici nudi opcija odabira pojedine kripto valute bilo u tablici ili preko polja za pretraživanje kripto valuta te se korisniku odabirom kripto valute prikazuje ekran detaljnih informacija o kripto valuti. Navedeni je ekran vidljiv na slici 20. Na vrhu ekrana detaljnih podataka pojedine kripto valute nalaze se osnovni podaci kao i na početnom ekranu. Ispod osnovnih podataka nalazi se grafički prikaz kretanja cijena kripto valute u vremenu od početka prikupljanja podataka pojedine kripto valute. Grafički prikaz podataka implementiran je koristeći programsku biblioteku za vizualizaciju podataka amCharts[2]. Grafički prikaz je u obliku linijskog grafa gdje se na x-osi nalazi komponenta vremena, a na y-osi cijena kripto valute u danom intervalu. Grafički prikaz nudi mnoge opcije navigacije kroz graf kao što je odabir raspona podataka ili vremenskog perioda za koji želi prikazati podatke. Također iznad grafa se nalaze kontrole za jednostavan odabir raspona podataka, a moguće je prikazati podatke za prethodna 24 sata, 7 dana, 30 dana ili pa sve prikupljene podatke.

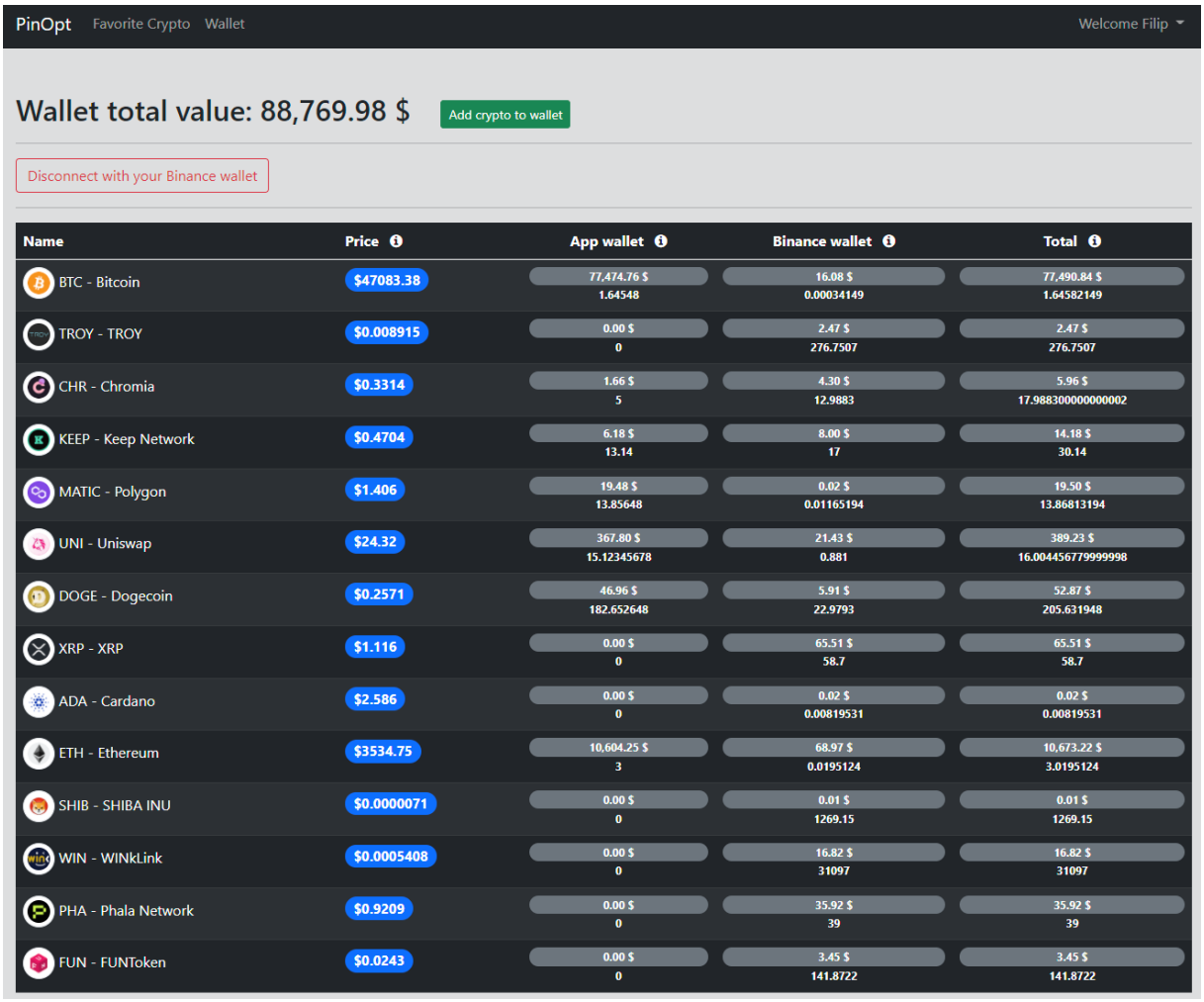


Slika 20 Prikaz ekrana detaljnih informacija kripto valute [autorski rad]

Sljedeći ekran je ekran pregleda novčanika korisnika. Na ovom su ekranu dostupni podaci o novčaniku korisnika te se na vrhu ekrana nalazi informacija o ukupnoj vrijednosti novčanika. Pokraj informacije o ukupnoj vrijednosti novčanika nalazi se opcija unosa kripto valute u novčanik korisnika unutar aplikacije. Ispod informacije o ukupnoj vrijednosti novčanika nalazi se opcija sinkronizacije aplikacije i novčanika korisnika na platformi Binance

[2] https://www.amcharts.com/

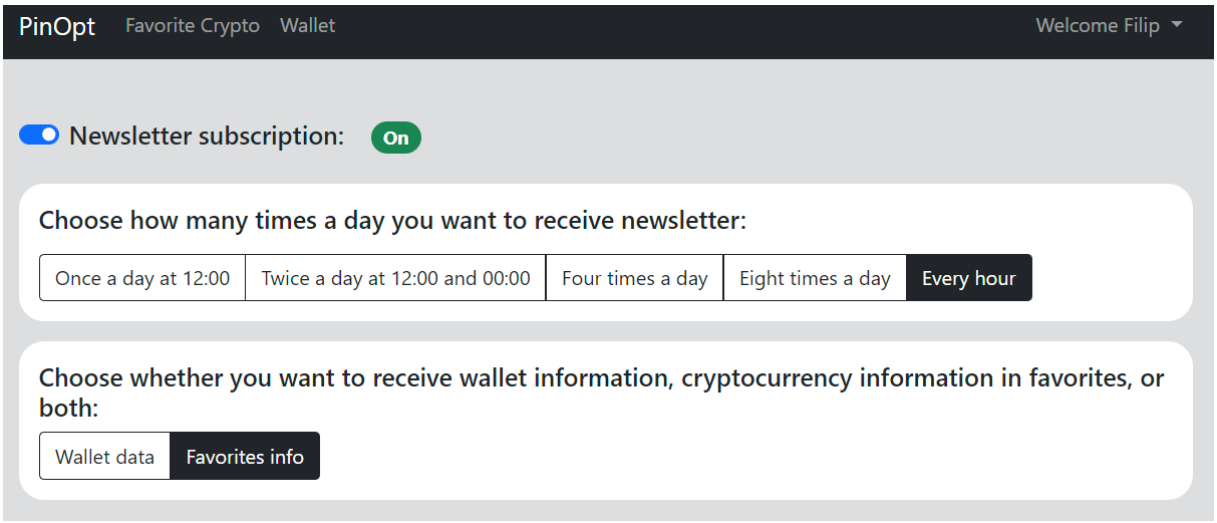
koja se izvršava pohranom dvaju API ključeva koje osigurava sama Binance platforma. U trenutku kada korisnik unese tražene ključeve preuzimaju se podaci novčanika korisnika te se umjesto sinkronizacije nudi prestanak sinkronizacije sa platformom Binance. Ispod svega do sada navedenog nalazi se tablični prikaz novčanika korisnika. U tablici se nalaze sve kripto valute koje korisnik posjeduje, a od podataka su za svaku kripto valutu dostupni podaci poput naziva, trenutne cijene, stanja novčanika aplikacije, stanja novčanika na Binance platformi te ukupno stanje oba novčanika zajedno. Slika 21 prikazuje ekran pregleda novčanika korisnika.



Slika 21 Prikaz ekrana novčanika korisnika [autorski rad]

Osim tri navedena ekrana važno je napomenuti i ekran konfiguracije newslettera korisnika. Dolaskom korisnika na ovaj ekran prvi puta nakon registracije korisniku se nudi opcija uključenja primitka newslettera na e-mail korisnika. Jednom kada korisnik uključi primanje newslettera tada je moguće podešavati konfiguracijske podatke kao što je vremenski period odnosno frekvencija primanja newslettera te odabir što je sve primljeni newsletter sadržavati. Od navedenih opcija moguće je odabrati 5 predefiniranih frekvencija

slanja, a frekvencije se kreću u rasponu od jednom dnevno pa do svakoga sata. Od podataka koje će newsletter sadržavati moguće je odabrati podatke novčanika korisnika te podatke o kripto valutama u listi favorita korisnika. Navedeni ekran konfiguracije newslettera korisnika prikazan je na slici 22.



Slika 22 Prikaz ekrana konfiguracije newslettera [autorski rad]

Postavljanje aplikacije na poslužitelj

Nakon završetka same implementacije aplikacije preostalo je aplikaciju postaviti na poslužitelj. Poslužitelj je u ovom slučaju računalo pokretano Windows 10 operativnim sustavom. Kako bi računalo, odnosno poslužitelj bio vidljiv drugim računalima na internetu te kako bi se na isti mogla preusmjeriti web domena potrebno je da računalo posjeduje vlastitu statičku IP adresu. Na poslužitelju je zatim instalirana i kreirana SQL Server baza podataka te IIS (eng. Internet Information Services) kao web server pokretan na Windowsima. Za upravljanje IIS-om koristi se alat Internet Information Services Manager unutar kojega je najprije kreiran novi server, na serveru je zatim kreiran aplikacijski bazen (eng, Application Pool) koji služi kako bi se određeni skup URL-ova mogao usmjeriti na određeni proces kreiran od strane aplikacije. Na disku servera kreiran je direktorij jednakog naziva kao i aplikacija te su u njemu kreirani posebni direktoriji Api za pohranu datoteka pozadinske aplikacije te App za pohranu datoteka korisničke aplikacije. Unutar IIS-a je zatim kreirana nova stranica (eng. site) kojoj se odredila putanja na prethodno kreirane direktorije i URL-ovi čije zahtjeve ova stranica poslužuje.

6. Kritički osvrt na razvoj vlastite aplikacije

Za preuzimanje podataka o vrijednosti kripto valuta koriste se vanjski API servisi koji nude dohvat odgovarajućih podataka, međutim u svojim besplatnim planovima uporabe takvi API servisi nude brojna ograničenja, a ono najznačajnije je ono u obliku limitiranog broja zahtjeva koji se mogu izvršiti. Upravo iz tog razloga u aplikaciji je implementiran pozadinski servis koji u intervalima od jednoga sata preuzima trenutnu vrijednost kripto valuta i te podatke pohranjuje u vlastitu bazu podataka. Također, prilikom inicijalnog pokretanja aplikacije baza podataka se puni povijesnim podacima o vrijednosti kripto valuta od početka prikupljanja podataka pojedine kripto valute. Na ovaj način ukoliko je aplikaciji potreban podatak o vrijednosti pojedine kripto valute u određeno vrijeme u prošlosti taj podatak može dohvatiti direktno iz baze te nije potrebno slati zahtjev na vanjski API servis. Ovakav pristup aplikaciju čini bržom i responzivnijom, a ujedno se broj zahtjeva koje aplikacije izvršava prema vanjskim API servisima drastično smanjuje. U vrijeme pisanja baza podataka aplikacije sadrži 859,606 zapisa o vrijednostima kripto valuta na određeni datum. Radi prethodno napomenutog limitiranog broja poziva API servisa u aplikaciji nije bilo moguće implementirati sve dostupne kripto valute već se odlučilo na 500 najpopularnijih kripto valuta u trenutku preuzimanja podataka. U nastavku će se objasniti više o ovom procesu i zašto aplikacija sadrži podatke o 377 kripto valuta, a ne prethodno navedenih 500.

Spremanjem podataka u vlastitoj bazi podataka uvelike je umanjilo vrijeme dohvata željenih podataka, ali uslijed tolikog broja zapisa u bazi podataka vrijeme izvršavanja standardnih upita za dohvaćanje podataka postaje veoma neoptimizirano. Kao odgovor na ovaj problem u bazi podataka kreiran je indeks nad atributima datuma i ID oznake kripto valute tablice u kojoj su zapisani podaci o vrijednosti kripto valuta na određen datum. Ovakav je indeks odabran iz razloga što je najčešći upit nad ovom tablicom dohvat podataka pojedine kripto valute u određenom vremenskom periodu, a upravo ovakav indeks značajno ubrzava dohvat ovakvih podataka. Ovakav je indeks moguće jednostavno kreirati dodavši dekorator `Indeks` entitetu aplikacije za koji je kreirana sama tablica, a u ovom slučaju to je entitet `CryptoCurrencyData`, a definiranje navedenog indeksa izgleda ovako:

```
[Index(nameof(Date), nameof(CryptoCurrencyId))]
 public class CryptoCurrencyData
 {…}
```

Kao još jedan odgovor na veliku količinu podataka u bazi podataka implementirano je i straničenje samih podataka te je na taj način osigurano da se u jednom trenutku iz baze podataka ili pak vanjskih API servisa ne dohvaća preveliki broj zapisa.

Prilikom inicijalnog pokretanja aplikacije u tablicu kripto valuta baze podataka uneseno je 500 najbolje rangiranih, odnosno najpopularnijih kripto valuta dohvaćenih putem vanjskog API servisa CoinMarketCap. Prilikom pokušaja preuzimanja povijesnih podataka o vrijednostima pojedinih kripto valuta preko vanjskog API servisa CryptoCompare uočeno je kako za velik broj kripto valuta ovaj API ne posjeduje podatke. Iz tog razloga iz baze podataka su uklonjene sve kripto valute za koje nije bilo moguće dohvatiti podatke.

Aplikacija kao što je već objašnjeno nudi korisniku povezivanje aplikacije sa Binance platformom za trgovanje kripto valutama kako bi se dohvatili podaci o stvarnom novčaniku korisnika na spomenutoj platformi. Za pristup API-u Binance platforme korisnik je dužan osigurati vlastite ključeve Binance API-a. Navedeni ključevi su tajna korisnika te ukoliko navedeni ključevi imaju to pravo mogu služiti i za manipulaciju samim novčanikom, odnosno kripto valutama korisnika. Ovakve ovlasti Binance API-a i ključeva korisnika stvaraju rizik za samog korisnika te je navedene ključeve bilo potrebno sigurno spremati u bazu podataka. Za ove potrebe kreirana je pomoćna klasa `Crypting` koja sadrži dvije metode, a to su `EncryptString` za kriptiranje stringa te `DecryptString` za dekriptiranje stringa koristeći tajni ključ. Obje metode za kriptiranje, odnosno dekriptiranje koriste AES specifikaciju. Pomoću ovih metoda ključevi se prije spremanja u bazu podataka kriptiraju kako bi se sigurno pohranili te se kasnije dekriptiraju kako bi se mogli koristiti u API pozivima.

7. Zaključak

U današnje vrijeme kripto valute postaju sve popularnija tema, a tržište kripto valuta nikada nije bilo veće nego u zadnjih nekoliko godina. Do prije nekoliko godina kripto valutama su trgovali isključivo entuzijasti i ljubitelji novih i uzbudljivih tehnologija, dok se danas broj ljudi koji trguju kripto valutama broji u desecima milijuna. Ovakva popularnost kripto valuta i njihovim trgovanjem dovela je do razvoja mnogih platformi i servisa koji nude pristup informacijama o kretanju cijena kripto valuta, odnosno nude uslugu praćenja trendova kripto valuta. Potpuno je neupitno kako je tržište kripto valuta tek u počecima te da će rastom tržišta rasti i potreba za upravo ovakvim vrstama aplikacija.

Kao izvor podataka o kripto valutama postoje mnoge opcije te postaje veoma važno prepoznati relevantne izvore i one izvore koje je moguće efektivno primijeniti. API tehnologije su u današnjem svijetu prepunom podataka i prijenosa istih žila kucavica koja uvelike olakšava dohvaćanje i dijeljenje podataka. Kako sam skup podataka sam po sebi za prosječnog korisnika ne donosi relevantnu informaciju veoma je važno velik skup podataka prikazati na način da se iz njih može iščitati određena informacija relevantna za korisnika. U ovom radu odabrani su načini prezentacije podataka u obliku tablica te grafički prikaz. Upravo se ovakav način prikaza velikog broja podataka prilikom realizacije ove aplikacije pokazao kao najbolje rješenje koje istovremeno nudi velik broj podataka, ali na pregledan način.

U praktičnom dijelu rada razvijena je web aplikacija za praćenje trendova kripto valuta. U aplikaciji su korištena tri izvora podataka koji se međusobno nadopunjuju te stvaraju kompletan skup potrebnih podataka. Pozadinska je aplikacija realizirana u .NET Core razvojnom okviru zadužena za dohvaćanje i obradu podataka sa vanjskih API servisa, obradu i praćenje podataka korisnika te pružanje API sučelja kako bi se ostvarila komunikacija između pozadinske i korisničke aplikacije. Korisnička je aplikacija realizirana pomoću Angular razvojnog okvira. Korisnička aplikacije poziva API krajnje točke pozadinske aplikacije i na taj način dohvaća podatke koje zatim prikazuje korisniku. Pozadinska aplikacija koristi vlastitu SQL Server bazu podataka.

Literatura

[1] CoinMarketCap, »Global Cryptocurrency Charts,« CoinMarketCap, [Mrežno]. Available: https://coinmarketcap.com/charts/. [Pokušaj pristupa 20 08 2021].

[2] CoinMarketCap, »Bitcoin vs The Biggest Companies And Assets In The World by Market Cap,« CoinMarketCap, [Mrežno]. Available: https://coinmarketcap.com/largest-companies/. [Pokušaj pristupa 20 08 2021].

[3] CoinMarketCap, »Today's Cryptocurrency Prices by Market Cap,« CoinMarketCap, [Mrežno]. Available: https://coinmarketcap.com/. [Pokušaj pristupa 20 08 2021].

[4] J. Lansky, »Possible State Approaches to Cryptocurrencies,« *Journal of Systems Integration,* svez. 9, pp. 19-31, 2018.

[5] S. Nakamoto, »Bitcoin: A Peer-to-Peer Electronic Cash System,« 2008. [Mrežno]. Available: https://bitcoin.org/bitcoin.pdf. [Pokušaj pristupa 29 07 2021].

[6] The Economist, »The great chain of being sure about things,« 31 10 2015. [Mrežno]. Available: https://www.economist.com/briefing/2015/10/31/the-great-chain-of-being-sure-about-things. [Pokušaj pristupa 29 07 2021].

[7] A. Narayanan, J. Bonneau, E. Felten, A. Miller i S. Goldfeder, Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction, Princeton: Princeton University Press, 2016.

[8] N. Darlington, »Blockchain For Beginners: What Is Blockchain Technology? A Step-by-Step Guide,« 05 07 2021. [Mrežno]. Available: https://blockgeeks.com/guides/what-is-blockchain-technology/. [Pokušaj pristupa 29 07 2021].

[9] V. HENRICH, E. HINRICHS, M. HINRICHS i T. ZASTROW, SERVICE-ORIENTED ARCHITECTURES:, Bucharest, Romania: University of Tübingen, Department of Linguistics, 2010.

[10] redhat, »What is an API?,« [Mrežno]. Available: https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces. [Pokušaj pristupa 02 08 2021].

[11] IBM Cloud Education, »Application Programming Interface (API),« 19 08 2020. [Mrežno]. Available: https://www.ibm.com/cloud/learn/api. [Pokušaj pristupa 02 08 2021].

[12] F. Fatemi, »How APIs Can Transform Your Company,« 21 03 2019. [Mrežno]. Available: https://www.forbes.com/sites/falonfatemi/2019/03/21/how-apis-can-transform-your-company/?sh=d522d3d668c2. [Pokušaj pristupa 03 08 2021].

[13] M. Maleshkova, C. Pedrinaci i J. Domingue, Investigating Web APIs on the World Wide Web, Eighth IEEE European Conference on Web Services, 2010, pp. 107-114, doi:

10.1109/ECOWS.2010.9., 2010.

[14] M. Masse, REST API Design Rulebook, Sebastopol, CA 95472: O'Reilly Media, 2011.

[15] A. Rodriguez, »RESTful Web services: The basics,« *BM developerWorks,* svez. 33, 06 11 2008.

[16] R. T. Fielding, »Architectural Styles and,« University of California, Irvine, 2000.

[17] I. C. IBM Cloud Team, »SOA vs. Microservices: What's the Difference?,« 14 05 2021. [Mrežno]. Available: https://www.ibm.com/cloud/blog/soa-vs-microservices. [Pokušaj pristupa 13 08 2021].

[18] I. C. Education, »SOA (Service-Oriented Architecture),« 17 07 2019. [Mrežno]. Available: https://www.ibm.com/nl-en/cloud/learn/soa. [Pokušaj pristupa 13 08 2021].

[19] S. Daya, N. V. Duy, K. Eati, C. Ferreira, D. Glozic, V. Gucer, M. Gupta, S. Joshi, V. Lampkin, M. Martins, S. Narain i R. Vennam, Microservices from Theory to Practice: Creating Applications in IBM Bluemix Using the Microservices Approach, IBM Redbooks, 2015.

[20] nomics.com, »Nomics pricing,« nomics.com, [Mrežno]. Available: https://p.nomics.com/pricing#. [Pokušaj pristupa 07 09 2021].

[21] Polygon.io, »Polygon.io pricing,« Polygon.io, [Mrežno]. Available: https://polygon.io/pricing. [Pokušaj pristupa 07 09 2021].

[22] CryptoCompare, »CryptoCompare pricing,« CryptoCompare, [Mrežno]. Available: https://min-api.cryptocompare.com/pricing. [Pokušaj pristupa 07 09 2021].

[23] CoinMarketCap, »CoinMarketCap pricing,« CoinMarketCap , [Mrežno]. Available: https://coinmarketcap.com/api/pricing/. [Pokušaj pristupa 07 09 2021].

[24] CoinAPI, »CoinAPI pricing,« CoinAPI, [Mrežno]. Available: https://www.coinapi.io/Pricing. [Pokušaj pristupa 07 09 2021].

[25] K. Peters, »Binance Exchange,« 08 07 2021. [Mrežno]. Available: https://www.investopedia.com/terms/b/binance-exchange.asp. [Pokušaj pristupa 18 08 2021].

[26] binance.com, »About Binance,« Binance, [Mrežno]. Available: https://www.binance.com/en/about. [Pokušaj pristupa 18 08 2021].

[27] Binance, »Binance API,« Binance, [Mrežno]. Available: https://binance-docs.github.io/apidocs/spot/en/#all-coins-39-information-user_data. [Pokušaj pristupa 18 08 2021].

[28] CryptoCompare, »About Us,« CryptoCompare, [Mrežno]. Available: https://www.cryptocompare.com/about-us/. [Pokušaj pristupa 18 08 2021].

[29] CryptoCompare, »THE PREMIUM {API} SOLUTION,« CryptoCompare, [Mrežno]. Available:

https://min-api.cryptocompare.com/. [Pokušaj pristupa 18 08 2021].

[30] CoinMarketCap, »About CoinMarketCap,« CoinMarketCap, [Mrežno]. Available: https://coinmarketcap.com/about/. [Pokušaj pristupa 19 08 2021].

Popis slika