Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Nejc Pisk

Realno-časovna vizualizacija gibanja tečajev kriptovalut

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: viš. pred. dr. Aljaž Zrnec

Ljubljana, 2018

Copyright. Rezultati diplomske naloge so intelektualna lastnina avtorja in Fakultete za računalništvo in informatiko Univerze v Ljubljani. Za objavo in koriščenje rezultatov diplomske naloge je potrebno pisno privoljenje avtorja, Fakultete za računalništvo in informatiko ter mentorja.

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Realno-časovna vizualizacija gibanja tečajev kriptovalut

Tematika naloge:

Popularnost kripto trga v zadnjih letih skokovito narašča, zato prihaja do vse večje zahteve po kakovostnih orodjih za trgovanje na volatilnemu trgu, kjer je vsaka sekunda pomembna. V okviru diplomskega dela razvijte prototip orodja, ki bo omogočal prikaz in vizualizacijo podatkov in spremljanje gibanja tečajev kriptovalut v realnem času. Izberite najprimernejši podatkovni vir kripto-valutnih tečajev. Razvito orodje mora podpirati izbor in prikaz različnih kriptovalut na poljubno izbrani časovni vrsti. Preučite različne možnosti za izris, ki jih imate na voljo in za posamezne vizualizacije izberite najprimernejše možnosti.

Zahvaljujem se mentorju viš. pred. dr. Aljažu Zrnecu za pomoč in hitro odzivnost pri izdelavi diplomskega dela. Prav tako bi se rad zahvalil sošolcem, ki so mi polepšali vsa tri leta študija, še posebej Žigi in Domnu. Hvala vsem sodelavcem in bližnjim za podporo na moji študijski poti.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
API	Application Programming Interface	aplikacijski programski vmesnik
SHA	Secure Hash Algorithm	Secure Hash Algorithm
EVM	Ethereum Virtual Machine	virtualni stroj Ethereum
CLI	Common Language Infrastructure	skupna jezikovna infrastruktura
IDE	Integrated Development Environment	integrirano razvojno okolje
FCL	Framework Class Library	ogrodje razredne knjižnice
HTML	Hyper Text Markup Language	označevalni jezik
JSON	JavaScript Object Notation	notacija objekta JavaScript
XML	Extensible Markup Language	razširljivi označevalni jezik
MSI	Microsoft Software Installation	Microsoft namestitev programske opreme
XPATH	XML path annotation	jezik za naslavljanje delov dokumentov XML

Povzetek

Naslov: Realno-časovna vizualizacija gibanja tečajev kriptovalut

Avtor: Nejc Pisk

Kriptovalute v zadnjih letih postajajo vedno bolj popularne, zaradi česar se je trg kriptovalut občutno povečal in s seboj prinesel večje število novih investitorjev. Anonimnost, deregulirana narava in volatilnost trga omogočajo investitorjem hitrejši vstop in večje zaslužke v primerjavi s trgovanjem z delnicami. Povečana popularnost je povečala zahtevo po kvalitetnih orodjih za spremljanje takšnega trga. Namen diplomske naloge je razvoj realno-časovnega sistema za spremljanje in vizualizacijo gibanja tečajev kriptovalut, ki pridobiva podatke s pomočjo spletnega pajka in jih shrani v podatkovno bazo, nad katero se kasneje izvajajo vizualizacije in analize podatkov. Spletni pajek je izdelan kot namizna konzolna aplikacija v okolju .NET, ki lušči podatke iz podatkovnih virov in jih razčleni v primerno obliko. Za podatkovno hrambo in iskanje je uporabljeno orodje Elasticsearch. Prikaz, analiza in realno-časovne vizualizacije nad temi podatki pa se izvajajo s pomočjo orodja Kibana.

Ključne besede: kriptovalute, realno-časovna vizualizacija, borze, veriga blokov, Elastic, spletni pajki.

Abstract

Title: Real-time visualization of cryptocurrency exchange rate movements

Author: Nejc Pisk

Cryptocurrencies have become increasingly popular in recent years, and therefore the cryptocurrency market has increased significantly and brought in new investors. Anonymity, deregulated nature and volatility of the market enable investors quicker entry and higher earnings in comparison with stock trading. Increased popularity has increased the demand for quality tools, which monitor these markets. The purpose of this thesis is the development of a real-time system for monitoring and visualizing the movement of cryptocurrency exchange rates, which acquires data using a web crawler and stores it in a database. The stored data can then have visualizations and analysis performed on. A web crawler is designed as a desktop console application in a .NET environment that acquires data from data sources and parses it into a suitable format. Elasticsearch is used for data storage and search. The display, analysis and real-time visualization of data are performed by Kibana.

Keywords: cryptocurrencies, real-time visualization, exchanges, blockchain, Elastic, web crawlers.

Poglavje 1

Uvod

V zadnjih letih se je trg s kriptovalutami občutno povečal, kar je prineslo večje število novih investitorjev. Kripto trg zaradi svoje anonimnosti in deregulirane narave omogoča investitorjem relativno hiter vstop na trg v primerjavi s klasičnim trgovanjem z delnicami. S prihodom novih uporabnikov se je povečala zahteva po kakovostnih orodjih za spremljanje kripto trga.

V diplomskem delu želimo razviti orodje za spremljanje gibanja tečajev kriptovalut v realnem času. Želimo ga zasnovati tako, da bo možno spremljati tri kriptovalute na poljubno izbrani časovni vrsti. Programsko kodo bomo oblikovali tako, da bo preprosto prilagodljiva, če bi uporabnik želel dodati nove kriptovalute. Uporabnik bo pridobljene podatke lahko prikazal z različnimi vizualizacijami v obliki grafikonov.

1.1 Motivacija

V kripto svetu trenutno obstaja večje število orodij za spremljanje gibanja tečajev kriptovalut, vendar ta niso dovolj izpopolnjena. Večjemu številu uporabnikov vstop na tovrstni trg predstavlja veliko težavo, naprednim uporabnikom pa je potrebno omogočiti večje število funkcionalnosti in svobodnejšo izbiro pri obdelavi podatkov, kot so na primer vizualizacije z različnimi vrstami grafikonov in iskanje po surovih podatkih tečajev kriptovalut.

Uporabnikom želimo omogočiti lažjo uporabo in boljši pregled gibanja tečajev na kripto trgu. Orodje bo dovolj zahtevno, da bo koristno naprednim uporabnikom, hkrati pa bo dovolj enostavno za vse začetnike.

1.2 Cilji

Osrednji cilj diplomske naloge je izdelati orodje, ki bo uporabniku omogočalo spremljanje gibanja tečajev kriptovalut v realnem času. Razviti je potrebno orodje, ki bo v realnem času črpalo ažurne podatke s spletne strani CoinMarketCap. Ti podatki bodo dostopni v orodju in predstavljeni uporabnikom z uporabo različnih vizualizacij.

Ugotovitve diplomskega dela bodo koristile vsem razvijalcem podobnih orodij, ki se bodo v prihodnosti ukvarjali s tovrstnim razvojem. Rezultati diplomske naloge jim bodo omogočili vpogled v težave in prednosti razvoja kripto orodij.

1.3 Struktura diplomskega dela

Struktura diplomske naloge je naslednja:

- poglavje 2 predstavi sorodna dela, ki so podobna rešitvi, ki jo želimo razviti v okviru diplomskega dela,
- poglavje 3 opisuje problemsko domeno kriptovalut ter verige blokov,
- poglavje 4 predlaga načrt ter arhitekturo sistema in pregled tehnologij, uporabljenih tekom razvoja diplomske naloge,
- poglavje 5 opisuje celoten razvoj in implementacijo orodja,
- poglavje 6 predstavi analizo, rešitve in rezultate diplomskega dela,
- poglavje 7 opisuje sklepne ugotovitve in predstavi zaključek diplomskega dela.

Poglavje 2

Sorodna dela

Na trgu obstaja že kar nekaj rešitev, ki ponujajo spremljanje gibanja tečajev kriptovalut. Preden se poglobimo v razvito rešitev, si najprej oglejmo sorodna dela s tega področja ter njihove prednosti in slabosti.

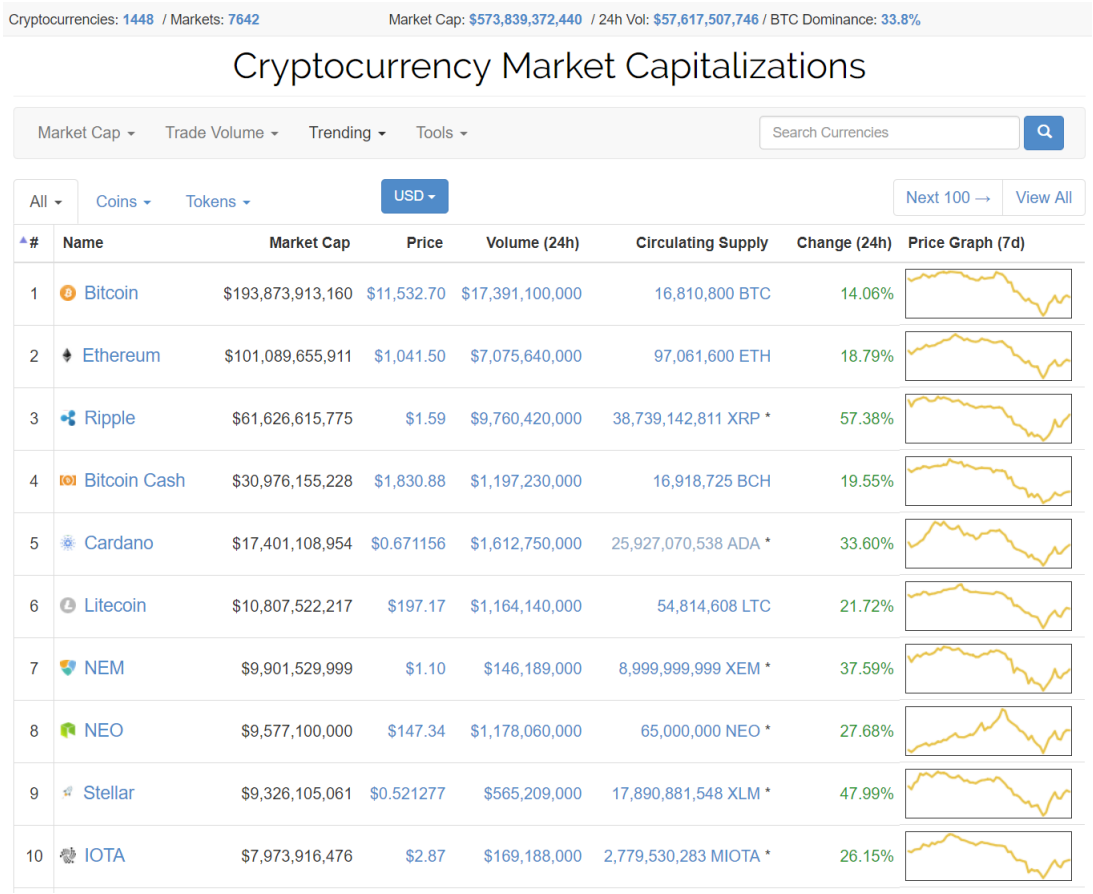
2.1 CoinMarketCap

CoinMarketCap [22] ali Cryptocurrency Market Capitalizations je najpopularnejša spletna stran za spremljanje tečajev kriptovalut [7, 3]. Trenutno podpira več kot 8500 trgov s kriptovalutami in več kot 1500 različnih kriptovalut [22].

Omogoča preprost pregled vseh kriptovalut z njihovimi dnevnimi spremembami cen, kot je to prikazano na sliki 2.1. Za vsako valuto se lahko preveri njen grafikon gibanja tečaja, prav tako vsebuje podatke o trgih, na katerih se s to valuto lahko trguje. Podroben pogled posamezne kriptovalute je viden na sliki 2.2. Omogoča tudi pregled zgodovinskih podatkov o ceni. Grafični vmesnik je intuitiven in preprost za uporabo. Zaradi ogromnega števila podprtih trgov, so podatki zanesljivi in točni. Nove valute na trgu so relativno hitro dodane [24]. Dodano imajo tudi funkcionalnost, ki omogoča pregled socialnih omrežij, kjer so objavljene novice o izbrani kriptovaluti.

Napredni uporabniki bodo pogrešali preglednejše in lažje nastavljive grafi-

kone s poljubno časovno vrsto in prilagodljivo skalo. Manjkajo tudi obvestila o spremembi cene preko e-pošte. Njihov aplikacijski programski vmesnik (API) [23] ponuja samo eno metodo, ki vrača trenutno ceno izbrane kriptovalute. Razvijalcem bi prav prišla metoda, ki bi omogočala pregled zgodovinskih podatkov. Čeprav imajo pomanjkljiv API, pa vseeno spletna stran omogoča neomejeno branje s spletnim pajkom.



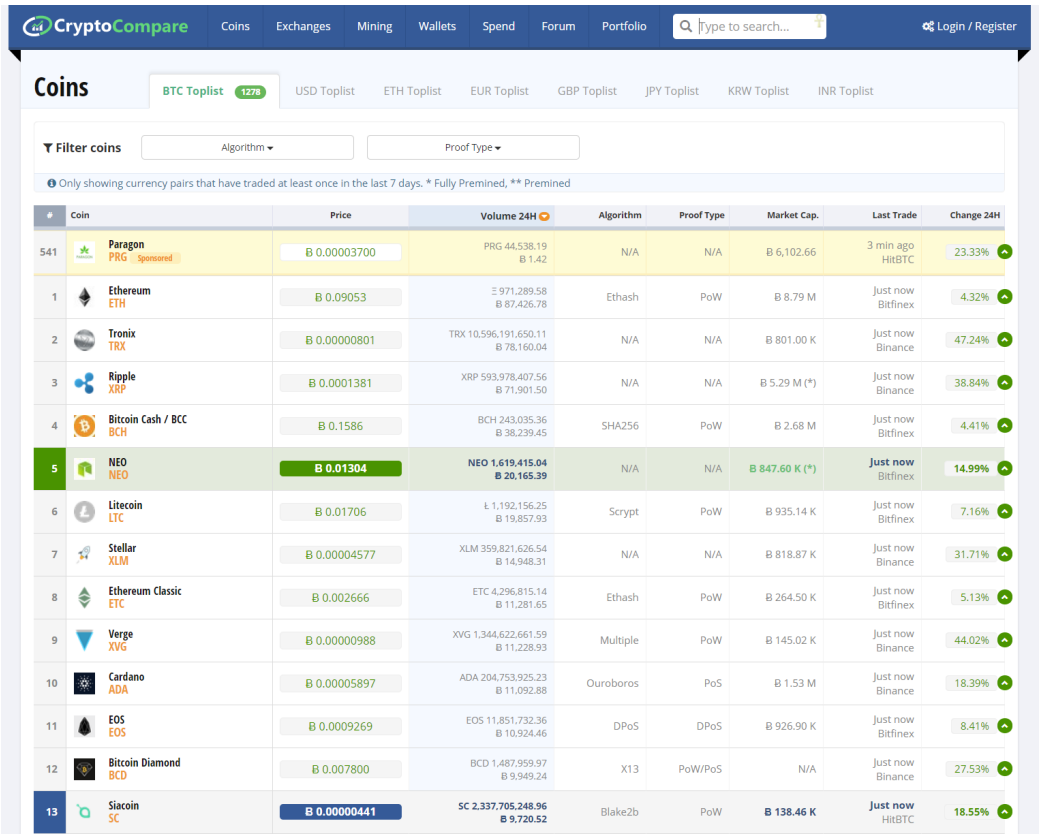
Slika 2.1: Začetna stran CoinMarketCap, ki prikazuje pregled vseh kriptovalut.

Slika 2.2: Podrobni pogled kriptovalute Bitcoin na spletni strani CoinMarketCap.

2.2 CryptoCompare

CryptoCompare [26] je spletna storitev, ki se uporablja za spremljanje kriptovalut in za pregled novic ter trendov kriptovalut na socialnih omrežjih. Podpira več kot 65 trgov s kriptovalutami in več kot 1000 različnih valut. Pregled vseh kriptovalut je viden na sliki 2.3, ki poleg imena prikazuje še ceno, količino v zadnjih štiriindvajsetih urah, šifrirni algoritem, tržno kapi-

talizacijo in spremembo cene v odstotkih.



Slika 2.3: Začetna stran CryptoCompare, ki prikazuje pregled vseh kriptovalut.

Podatki se prikazujejo v realnem času. Spletna stran ponuja večje število orodij za spremljanje kriptovalut. Eno izmed njih je orodje Portfolio [64], ki uporabniku omogoča pregled nad lastnim portfoliom s kriptovalutami. Uporabnik lahko dodaja kupljene kriptovalute in spremlja gibanje njihovih tečajev.

Na začetni strani je vidno, da podpirajo manjše število kriptovalut in trgov. Cene kriptovalut se razlikujejo v primerjavi s točnejšimi podatki spletne strani CoinMarketCap. Veliko število kriptovalut ni podprtih in upravljalci spletne strani redkokdaj dodajo nove. Na strani je veliko oglasov in promo-

viranih kriptovalut.

Podroben pogled določene kriptovalute je viden na sliki 2.4. Grafični prikaz gibanja tečajev podpira poljubno skalo, poljubno nastavljanje časovne vrste in dodajanje poljubnih likov ter besedil na grafikone. Za pregled zgodovinskih podatkov, starejših od treh mesecev, je na spletni strani potrebno ustvariti nov uporabniški račun. Neregistriranim uporabnikom ta funkcionalnost ni na voljo. Spremljati je možno vse cene na podprtih trgih, mesečno analizo in nova sporočila na socialnih omrežjih, ki omenjajo izbrano kriptovaluto.

API [27] razvijalcem ponuja večje število metod, ki se lahko uporabijo za spremljanje kriptovalut, vendar je omejen za uporabo s številom zahtevkov.

Slika 2.4: Podroben pogled kriptovalute Ethereum na spletni strani CryptoCompare.

Poglavje 3

Problemska domena

3.1 Kriptovalute

Kriptovalute so digitalna sredstva, zasnovana tako, da delujejo kot menjalno sredstvo [67]. Za zaščito in varnost svojih transakcij uporabljajo šifriranje oziroma kriptografijo. Šifriranje se prav tako uporabi pri izdelavi dodatnih enot in pri preverjanju prenosa sredstev. Razvrščamo jih v podmnožico digitalnih, alternativnih in virtualnih valut. Uporabljajo decentraliziran nadzor v primerjavi s centraliziranim elektronskih denarjem in centralnimi bančnimi sistemi. Za decentralizacijo se uporablja veriga blokov, ki je javna transakcijska baza podatkov in deluje kot porazdeljena knjiga finančnih računov [17]. Podrobneje si verigo blokov pogledamo v poglavju 3.2.

Bitcoin, ustanovljen leta 2009, predstavlja prvo decentralizirano kriptovaluto [15]. Od takrat so bile ustvarjene številne druge kriptovalute. Te se pogosto imenujejo alternativni kovanci (angleško *altcoins*) [8]. V diplomski nalogi smo se odločili, da bomo v implementaciji prototipa rešitve podprli tri kriptovalute:

- Bitcoin (glej poglavje 3.3),
- Litecoin (glej poglavje 3.4) in
- Ethereum (glej poglavje 3.5).

Decentralizirane kriptovalute proizvaja njen sistem, ki je javno znan. Število proizvedenih kriptovalut je določeno s stopnjo, ki je znana, ko je sistem ustvarjen. V centraliziranih bančnih in gospodarskih sistemih, kot je sistem Zveznih rezerv, upravni odbori ali vlade nadzorujejo dobavo valute s tiskanimi enotami fiat denarja. V primeru decentralizirane kriptovalute podjetja ali vlade ne morejo proizvajati novih enot. Veriga blokov, na katerem temeljijo decentralizirane kriptovalute, je bila prvič uporabljena pri digitalni valuti Bitcoin [63].

Trenutno, februarja 2018, obstaja več kot 1500 kriptovalut [22]. Večina izhaja iz prvotne decentralizirane kriptovalute Bitcoin. Obstajajo tudi kriptovalute, ki uporabljajo centraliziran sistem, kot na primer Ripple [20]. Znotraj sistemov kriptovalut obstaja tudi skupnost med seboj nepovezanih strank, ki se imenujejo rudarji. To so javni uporabniki sistema, ki uporabljajo svoje računalnike za rudarjenje. Z rudarjenjem uporabniki preverjajo in časovno žigosajo transakcije in v zamenjavo prejmejo valuto. Uspešno ovrednotene transakcije so kasneje dodane v knjigo transakcij. Uporabniki imajo v takšnem sistemu finančno spodbudo za rudarjenje [25]. Rudarjenje bolj podrobno opišemo v poglavju 3.3.4.

Večina kriptovalut, podobnih Bitcoinu, je zasnovanih tako, da postopoma zmanjšujejo proizvodnjo valute, s čimer se določi zgornja meja števila denarnih enot v obtoku. Ta sistem je bil izbran, ker je podoben sistemu, ki se uporablja pri plemenitih kovinah. Postopek pridobivanju teh valut se zato imenuje rudarjenje. V primerjavi z navadnimi valutami, ki jih posedujejo finančne institucije, je kriptovalute težje zaseči s strani organov pregona, ker temeljijo na tehnologiji šifriranja [59].

3.2 Veriga blokov

Veriga blokov (angleško *blockchain*) je nenehno naraščajoč seznam zapisov, ki se imenujejo bloki. Ti so med seboj povezani in zavarovani z uporabo kriptografije. Vsak blok običajno vsebuje zgoščevalno funkcijo za šifriranje,

ki deluje kot povezava na prejšnji blok. Prav tako vsebuje časovni žig in podatke o transakciji. Transakcija predstavlja prenos bitcoinov, ki se odda v omrežje in zbere v blokih.

Veriga blokov je v osnovi oblikovana tako, da je odporna na spremembo podatkov [17]. Gre za javno, distribuirano in decentralizirano digitalno knjigo, ki omogoča učinkovito in zanesljivo ter trajno beleženje transakcij med dvema strankama. Porazdeljena knjiga lahko deluje, če verigo blokov upravlja omrežje enakovrednih uporabnikov, ki se kolektivno držijo protokola za potrjevanje novih blokov [33]. Zabeleženih podatkov v kateremkoli danem bloku ni mogoče spremeniti brez spreminjanja vseh nadaljnjih blokov [17].

Veriga blokov je zavarovana s šifriranjem in je primer porazdeljenega računalniškega sistema z visokim nivojem odpornosti proti napakam. Decentralizirano soglasje je doseženo z verigo blokov, zaradi česar je ta potencialno primerna za dejavnosti, ki upravljajo z zapisi, kot so upravljanje identitet, obdelava transakcij ali glasovanje [77].

Prvo idejo verige blokov je leta 2008 konceptualizirala anonimna oseba ali skupina, znana kot Satoshi Nakamoto. Ta se je kasneje, leta 2009, uporabila kot glavna komponenta kriptovalute Bitcoin, kjer je služila kot javna knjiga transakcij [15].

3.2.1 Struktura verige blokov

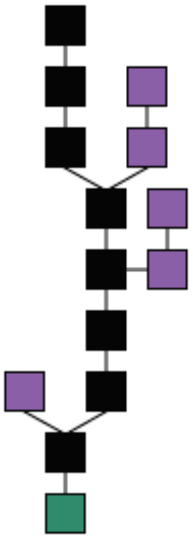
Kot smo prej omenili, je veriga blokov decentralizirana in distribuirana digitalna knjiga, ki se uporablja za beleženje transakcij med različnimi uporabniki. Zapisi se ne morejo spremeniti, kar omogoča uporabnikom sistema, da lahko transakcije preverijo in pregledajo v raziskovalcu blokov [19]. Podatkovna baza verige blokov se upravlja z omrežjem enak z enakim (angleško *peer-to-peer*) in s porazdeljenim strežnikom za časovne žige [12]. Transakcije se preverjajo s sodelovanjem večjega števila uporabnikov. Rezultat je robustnost, kjer je negotovost udeležencev glede varnosti podatkov zanemarljiva, ker se uporablja protokol, ki omogoča zaupanje med uporabniki z uporabo „pametne" in transparentne programske kode.

Uporaba verige blokov iz digitalnega sredstva, kot na primer iz kriptovalute, odstrani neskončno ponovljivost digitalne enote in rešuje dolgotrajni problem dvojne porabe. Problem dvojne porabe je potencialna pomanjkljivost digitalnih valut, kjer se digitalna enota lahko porabi več kot enkrat. To je mogoče, ker digitalne enote vsebujejo digitalne podatke, ki se lahko podvojijo ali ponaredijo. Veriga blokov to rešuje tako, da se vsaka vrednost enote prenese le enkrat. To je možno, ker je veriga blokov javna digitalna knjiga transakcij, kjer se mora vsaka transakcija preveriti s strani rudarjev. Verigo blokov opisujemo tudi kot protokol za izmenjavo vrednosti [13]. Takšna izmenjava, ki temelji na blokih, se lahko zaključi hitreje, varneje in ceneje kot pri tradicionalnih sistemih.

Bloki držijo serije veljavnih transakcij, ki so zapisane in kodirane v drevesu Merkle [49]. Vsak blok vključuje zgoščevalno funkcijo predhodnega bloka v vrsti, ki ju med seboj povezuje. Povezani bloki tvorijo verigo. Povezovanje blokov v verigo je iterativni proces, ki potrjuje integriteto prejšnjega bloka vse do prvotnega geneznega bloka. Omenjen proces je viden na sliki 3.1.

Čas bloka je povprečni čas, ki je potreben, da omrežje ustvari en dodaten blok. Nekatere verige blokov lahko ustvarijo nov blok vsakih pet sekund. Podatki postanejo preverljivi, ko se blok zaključi. Pri kriptovalutah je to takrat, ko se denarna transakcija zgodi. Krajši čas bloka torej pomeni, da se bo transakcija hitreje izvedla [66].

Veriga blokov shranjuje podatke po celotnem omrežju, kar odpravi tveganja, ki so prisotna pri centralno shranjenih podatkih. Njeno omrežje nima centraliziranih točk ranljivosti, ki bi jih lahko izkoristila zlonamerna orodja. Prav tako nima nobene osrednje točke odpovedi [17, 63]. Varnostne metode verige blokov uporabljajo asimetrično šifriranje z javnim ključem. Javni ključ je dolgo in naključno zaporedje znakov in številk [65]. Predstavlja naslov na verigi blokov. Vrednostni žetoni, ki se pošljejo po omrežju, se zabeležijo kot pripadniki naslovu, zasebni ključ pa deluje kot geslo, ki svojemu lastniku omogoča dostop do digitalnih sredstev, ki jih vsebujejo bloki. Podatke, shra-



Slika 3.1: Struktura verige blokov [18]. Glavna veriga črne barve, genezni ali izvorni blok zelene barve, sirotni bloki (angleško *Orphan blocks*) vijolične barve.

njene v bloku, ni mogoče spreminjati, kar je največja prednost verige blokov [12]. Centralizirane podatke je lažje kontrolirati, vendar so manipulacije z informacijami in podatki pogoste, kar ne zagotavlja transparentnosti sistema in vidljivosti podatkov. Z decentralizacijo so vsi podatki v verigi pregledni za vse vpletene.

Vsako vozlišče ima v decentraliziranemu sistemu kopijo verige blokov. Kakovost podatkov se ohranja z velikim številom replikacij baze podatkov in z računalniškim zaupanjem. V informacijski varnosti računalniško zaupanje predstavlja zaupanje uporabnikov s pomočjo kriptografije ali zunanjih zaupanja vrednih organov (na primer Let's Encrypt [55]).

V sistemu ne obstaja nobena centralizirana kopija, hkrati pa noben uporabnik ni več zaupanja vreden kot katerikoli drugi [6]. Transakcije se v omrežje oddajajo z uporabo programske opreme. Uporabnik v programski opremi določi število digitalnih enot, ki jih želi poslati v omrežje. Vozlišča skenirajo celotno omrežje verige blokov in preverijo, če ima uporabnik dovolj

digitalnih enot. Ko je to preverjeno, se transakcija vključi v blok [16]. Rudarska vozlišča potrjujejo transakcije in jih dodajajo v blok, ki ga gradijo. Z rudarjenjem uporabniki preverjajo in časovno žigosajo transakcije in v zameno prejmejo digitalne enote. Zaključeni bloki se nato porazdelijo tudi na druga vozlišča v omrežju. S tem se okrepi varnost podatkov v verigi blokov. V primeru, da bi drugo vozlišče spremenilo svojo kopijo verige blokov, to ne bi vplivalo na varnost omrežja. Vsako vozlišče vsebuje svojo kopijo verige blokov, kar pomeni, da ni potrebno zaupati drugim vozliščem in njihovim kopijam verige blokov. Veriga uporablja različne programe časovnega žigosanja za zaporedne spremembe, kot na primer dokazilo o delu, lastništvu in avtorstvu [54].

Rast decentraliziranih verig spremlja tudi tveganje, kjer postanejo računalniški viri dražji za obdelavo večje količine podatkov. Ko velikost verige blokov naraste, se poveča tudi velikost, ki je potrebna za shranjevanje podatkov. Prav tako naraste potreba po višji pasovni širini in računski moči, ki je potrebna za sodelovanje v omrežju. V določenem trenutku to postane tako drago, da le nekaj vozlišč lahko obdela bloke, kar pa pripelje do tveganja centralizacije [50].

3.3 Bitcoin

Bitcoin je kriptovaluta, ki se lahko uporablja po vsem svetu in je prva decentralizirana digitalna valuta [15]. Transakcije potekajo neposredno med uporabniki, ki so si med seboj enakovredni. Uporablja se decentraliziran sistem zaupanja, namesto tradicionalnega sistema zaupanja druge centralizirane finančne institucije [9]. Bitcoin je mogoče zamenjati za druge valute, izdelke in storitve. Od februarja 2015 je več kot 100.000 trgovcev in prodajalcev sprejemalo Bitcoin kot plačilno sredstvo [30]. Bitcoin je izumila neznana oseba ali skupina ljudi z imenom Satoshi Nakamoto. Izdan je bil kot odprtokodna programska rešitev v letu 2009 [15].

3.3.1 Tehnologija

Tehnologija, prisotna pri valuti Bitcoin, je veriga blokov. Podrobneje smo jo opisali v poglavju 3.2.

3.3.2 Transakcije

Transakcije so opredeljene s skriptnim jezikom, ki je podoben programskemu jeziku Forth [45]. Transakcije so sestavljene iz enega ali več vhodov in enega ali več izhodov. Vhod transakcije predstavlja število poslanih bitcoinov. Izhod pa lahko predstavlja število bitcoinov, ki so bili uporabljeni za plačilo in število bitcoinov, ki so bili povrnjeni uporabniku kot ostanek plačila. Ko uporabnik pošlje bitcoine, mora označiti vsak naslov in količino, ki se pošlje na ta naslov. Dvojna poraba se prepreči tako, da se mora vsak vnos nanašati na prejšnji neporabljeni izhod na verigi blokov [48]. Transakcije imajo lahko več različnih izhodov, kar pomeni, da lahko uporabniki v eni transakciji pošljejo bitcoine več različnim prejemnikom. Vsota kovancev za plačilo lahko presega predvideni znesek plačil, podobno kot pri gotovinski transakciji. V takem primeru se uporabi dodaten izhod, ki vrne razliko nazaj plačniku [48]. Vsi neobračunani kovanci v transakcijskih izhodih postanejo provizije. Plačilo provizije za transakcijo ni obvezno. Rudarji lahko izberejo, katere transakcije se obdelujejo, in prednostno obravnavajo tiste, ki plačajo višje provizije. Velikost transakcije je odvisna od števila vhodov in izhodov, ki se uporabijo za generiranje transakcije [48]. Bitcoin transakcije se merijo z enoto satoshi na bajt.

3.3.3 Lastništvo

V verigi blokov so bitcoini registrirani na Bitcoin naslove. Ustvarjanje Bitcoin naslova ni nič drugega kot izbiranje naključnega veljavnega zasebnega ključa in izračunanje ustreznega naslova. To preračunavanje se zgodi v delčku sekunde. Obratna operacija je matematično neizvedljiva, kar uporabnikom omogoča, da svoj javni naslov sporočijo drugim uporabnikom brez ogrožanja

svojega zasebnega ključa [65]. Poleg tega je število veljavnih zasebnih ključev tako veliko, da je izjemno malo verjetno, da bo nekdo izračunal ključ, ki je že v uporabi. Veliko število zasebnih ključev onemogoči izračun le-teh s surovo silo. Uporabnik mora za porabo bitcoinov uporabiti svoj zasebni ključ in transakcijo digitalno podpisati. Omrežje potrdi podpis s pomočjo javnega ključa [1]. V primeru izgubljenega zasebnega ključa omrežje ne bo priznalo drugih dokazil o lastništvu. Kovanci postanejo neuporabni in izgubljeni.

3.3.4 Rudarjenje

Rudarjenje je vodenje evidence zapisov na verigi blokov, ki se opravlja z uporabo računalniškega procesiranja. Rudarji ohranjajo verigo blokov dosledno in nespremenljivo. To pomeni, da je veriga blokov konsistentna in omogoča varnost in transparentnost podatkov, ki se nahajajo v njej. To je mogoče z večkratnim preverjanjem in zbiranjem na novo oddanih transakcij v bloke [25]. Vsak blok vsebuje zgoščevalno funkcijo prejšnjega z uporabo algoritma SHA-256 [70]. Za sprejem bloka v omrežje mora novi blok vsebovati dokazilo o delu [54]. Dokazilo o delu zahteva, da rudarji najdejo številko, ki se imenuje nonce. Nonce predstavlja arbitrarno številko, ki se lahko uporabi samo enkrat [28]. Dokaz vozlišča je enostavno preveriti, vendar pa ga je časovno zahtevno ustvariti. Vsakih 2016 blokov se težavnostni cilj prilagodi glede na nedavno uspešnost omrežja. Na ta način se sistem samodejno prilagaja skupni količini rudarjenja v omrežju. Sistem dokazil o delu poleg verižnih blokov povzroči izredno težke spremembe bloka, saj mora napadalec spremeniti vse nadaljnje bloke [1]. Novi bloki se ves čas rudarijo, kar konstantno povečuje čas, ki je potreben za spreminjanje bloka. Proces rudarjenja bitcoinov je sledeč [5]:

1. Nova transakcija se odda vsem vozliščem.
2. Vsako rudarsko vozlišče zbere novo transakcijo v blok.
3. Vsako rudarsko vozlišče dela na potrjevanju svojega bloka.

4. Vozlišče najde dokaz dela in blok odda vsem vozliščem.

5. Nove bitcoine pridobi vozlišče, ki je našlo dokaz dela.

6. Vozlišča sprejmejo blok le takrat, ko so vse transakcije v njem veljavne in niso porabljene.

3.3.5 Denarnice

Denarnica shranjuje podatke, ki so potrebni za prenos bitcoinov. Denarnica je nekaj, kar shrani digitalne enote kriptovalute, s katerimi lahko dostopamo do njih in jih porabimo [46]. Denarnica je zbirka javnega in zasebnega ključa. Obstaja več vrst denarnic [46]:

- programske denarnice, ki se povezujejo v omrežje in dovoljujejo porabo bitcoinov, prav tako pa vsebujejo poverilnice, ki dokazujejo lastništvo,
- spletne denarnice, ki ponujajo podobno funkcionalnost kot programske, vendar so lažje za uporabo, ker so poverilnice shranjene pri ponudniku denarnice in ne na strojni opremi uporabnika,
- fizične denarnice, ki shranjujejo poverilnice, potrebne za uporabo bitcoinov, brez internetne povezave.

3.4 Litecoin

Litecoin je kriptovaluta in odprtokodni projekt, ki deluje na sistemu omrežja enak z enakim (angleško *peer-to-peer*) [57]. Navdih za izdelavo kovanca je bil pridobljen iz kriptovalute Bitcoin, ki mu je v tehnologiji skoraj identičen, kot je bilo opisano v poglavju 3.3.

Glavne razlike med Litecoinom in Bitcoinom so:

- namen mreže Litecoin je obdelati blok vsaki dve minuti in pol, namesto Bitcoinovih deset, kar omogoča hitrejše potrjevanje transakcij,

- Litecoin uporablja scrypt za dokazovanje dela, to je zaporedna pomnilniška funkcija [68], ki zahteva asimptotično več pomnilnika,
- naprave, ki se uporabljajo za rudarjenje litecoinov so zaradi uporabe algoritma scrypt kompleksnejše in dražje od tistih, ki jih uporablja Bitcoin [68].

3.5 Ethereum

Ethereum je javna, odprtokodna in distribuirana platforma, ki deluje na verigi blokov. Je tudi operacijski sistem, ki omogoča funkcionalnost pametnih pogodb s skriptnimi jeziki [42]. Ether je kriptovaluta, ki ji verigo blokov generira platforma Ethereum [38]. Ether se lahko prenaša med računi in se uporablja za kompenzacijo udeležencev rudarskih vozlišč za opravljene izračune [74]. Ethereum ponuja decentraliziran virtualni Turingov stroj. To je virtualni stroj Ethereum (EVM), ki lahko izvaja skripte z uporabo mednarodne mreže javnih vozlišč. Notranji mehanizem za določanje transakcij se imenuje plin (angleško *gas*) in se uporablja za ublažitev neželene vsebine in dodeljevanje virov v omrežju [42]. Enota za plin se običajno meri v enoti gwei. Ethereum je konec leta 2013 predlagal Vitalik Buterin, raziskovalec kriptovalut in programer [40].

Kot pri ostalih kriptovalutah je veljavnost vsakega etherja zagotovljena z verigo blokov, ki je nenehno naraščajoč seznam zapisov, zavarovanih s šifriranjem. Veriga blokov je po svoji naravi odporna proti spremembi podatkov [17]. Tehnologijo verige blokov smo opisali v poglavju 3.2.

Naslovi so sestavljeni iz predpone $0x$, ki ji sledi niz 40 številk in črk [76], ki ustrezajo določenemu javnemu naslovu. Primer:

$$0x4DC1bce8Ca44bf34fd1517317600E0b26A0ecd5a. \tag{3.1}$$

3.5.1 Primerjava z Bitcoinom

Ether se razlikuje od Bitcoina v več vidikih:

- Njegov čas bloka je 14 do 15 sekund, Bitcoina pa 10 minut [39].

- Rudarjenje ustvarja nove kovance z določeno hitrostjo. To pomeni, da se kovanci ustvarjajo s konsistentno hitrostjo, ki le redko odstopa od povprečja. Hitrost se občasno spreminja med razcepi, medtem ko se za Bitcoin stopnja prepolovi vsaka štiri leta [39].

- Stroški transakcij se razlikujejo glede na računsko kompleksnost, uporabo pasovne širine in potrebe po shranjevanju, medtem ko Bitcoin transakcije konkurirajo med seboj z velikostjo transakcije v bajtih [39].

- Enote za plin imajo svojo ceno, ki jo je mogoče določiti v transakciji. Ta se običajno meri v enoti gwei. Bitcoin transakcije imajo provizije, ki so določene z enoto satoshi za bajt [76].

- Provizije transakcij so običajno znatno nižje kot pri Bitcoinu, decembra 2017 je povprečna transakcija za Ether stala 0,33 dolarja, Bitcoin transakcija pa 23 dolarjev [73, 39].

- Ethereum uporablja sistem, kjer se bremenijo računi in se knjižijo v dobro drugega, v nasprotju z Bitcoinovim sistemom, ki je bolj podoben porabi denarja [39].

3.5.2 Virtualni stroj Ethereum

Virtualni stroj Ethereum (EVM) [43] je okolje, ki omogoča izvajanje kode pametnih pogodb na platformi Ethereum. Temeljni mehanizem predstavlja 256-bitni register. To je programski „peskovnik”, ki je popolnoma izoliran od omrežja, datotečnega sistema ali drugih procesov gostiteljskega računalniškega sistema. Peskovnik je testno okolje, ki izolira ne-testirano kodo in spremembe od produkcijskega okolja. Vsako vozlišče v omrežju izvaja EVM implementacijo in enaka navodila. Virtualni stroji Ethereum so bili implementirani v jezikih C++, Go, Haskell, Java, JavaScript, Python, Ruby in Rust [43].

3.5.3 Pametne pogodbe

Pametne pogodbe Ethereum temeljijo na računalniškem jeziku, ki ga razvijalci uporabljajo za programiranje lastnih funkcij. Standardna pogodba opisuje svoje pogoje, ki postanejo pravnomočni in izvršljivi z zakonom. Pametne pogodbe pa uveljavljajo svoje pogoje s kriptografijo in programsko kodo [41]. Preprost primer uporabe je, da uporabnik pošlje ether drugemu uporabniku in določi datum transakcije z uporabo pametne pogodbe.

Pametne pogodbe so abstrakcije programiranja na visoki ravni, ki se prevedejo v programsko kodo EVM in se kasneje izvajajo na verigi blokov [41]. Napisane so lahko v programskih jezikih, kot na primer Solidity [71], Serpent [69], LLL [58] in Mutan [60]. Obstaja tudi raziskovalno usmerjen jezik Viper [53], ki je še vedno v razvoju.

Pametne pogodbe so lahko javne, kar odpre možnost dokazovanja transparentnosti funkcionalnosti pogodbe. Problem pametnih pogodb je, da so hrošči in varnostne luknje javno vidne, vendar jih ni mogoče hitro odpraviti [41].

3.5.4 Decentralizirane aplikacije

Decentralizirane aplikacije na verigi blokov Ethereum navadno imenujemo DApps [32], ker temeljijo na decentraliziranem EVM in na pametnih pogodbah. Primeri uporabe takšnih aplikacij so lahko na področju financ, interneta stvari, pridobivanja električne energije ter športnih stav. Ethereum je od leta 2017 naprej glavna platforma za začetne projekte za prodajo kovancev [31]. Od januarja 2018 obstaja več kot 250 decentraliziranih aplikacij na platformi Ethereum [56], pri čemer jih je na stotine še v razvoju.

Poglavje 4

Načrt

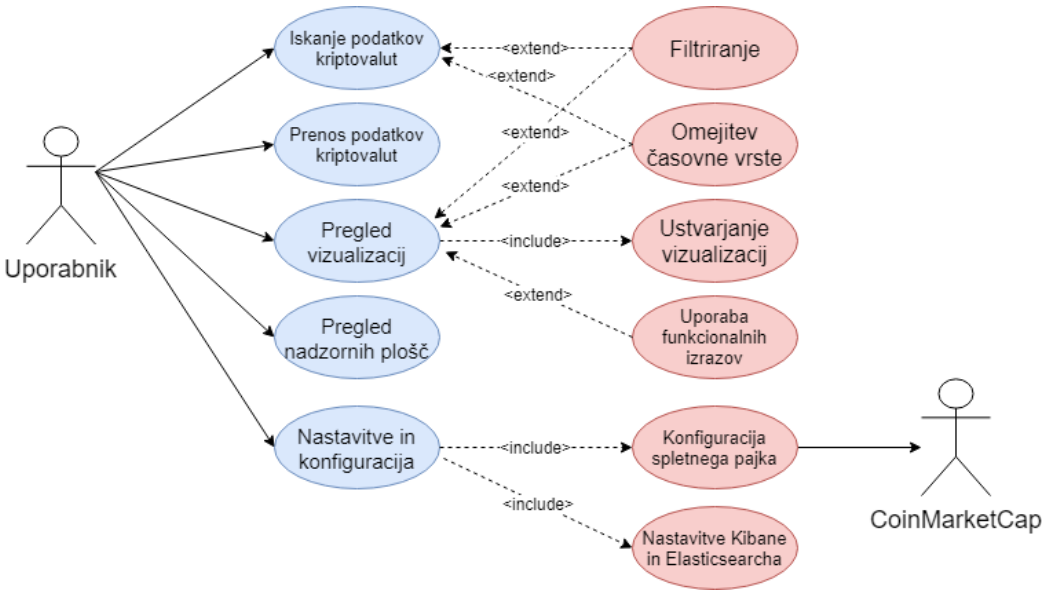
V pričujočem poglavju bomo opisali celoten načrt izdelave svoje realnočasovne aplikacije za spremljanje in vizualizacijo gibanja tečajev kriptovalut Bitcoin, Litecoin in Ethereum. Osredotočili se bomo na arhitekturo sistema in pregledali vse uporabljene tehnologije in orodja. V nadaljevanju je predstavljen diagram primerov uporabe, ki prikazuje vse funkcionalnosti naše aplikacije, akterje, ki jo uporabljajo in zunanje sisteme na katere se aplikacija povezuje. Diagram primerov uporabe je prikazan na sliki 4.1.

4.1 Arhitektura sistema

Arhitektura sistema bo sestavljena iz več plasti, in sicer iz vira podatkov, programske opreme, baze podatkov in vizualizacijskega orodja. Celotna arhitektura je vidna na sliki 4.2.

4.1.1 Vir podatkov

Vir podatkov predstavlja spletna stran CoinMarketCap [22], ki smo jo opisali v poglavju 2.1. Zanjo smo se odločili, ker ponuja ažurne in točne podatke gibanja tečajev kriptovalut iz večjega števila borz. Prav tako ponuja zgodovinske podatke, ki jih bomo lahko uporabili za zgodovinsko analizo. Čeprav ima spletna stran API [23], ga ne bomo uporabili, ker le-ta ne ponuja zgodo-



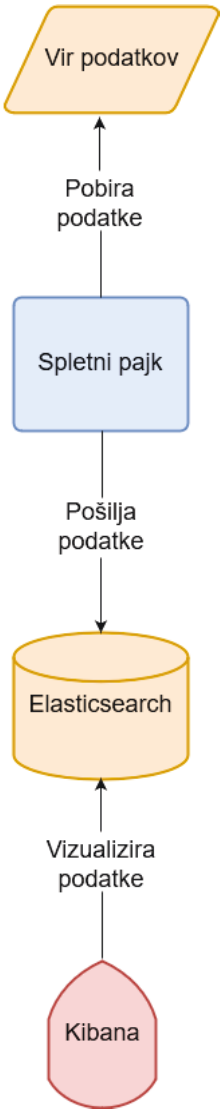
Slika 4.1: Diagram primerov uporabe aplikacije.

vinskih in ažurnih podatkov iz različnega števila borz. Podatke bomo zato črpali s spletnim pajkom.

4.1.2 Programska oprema

Razvili bomo svojo programsko opremo, ki bo črpala podatke iz vira in hkrati pošiljala le-te v bazo podatkov. V Visual Studiu bomo ustvarili projekt C#, ki bo uporabljal okolje .NET. Omenjeni tehnologiji in orodje smo opisali v poglavjih 4.2.1, 4.2.3 in 4.2.2.

Program bo črpal podatke s spletne strani CoinMarketCap [22] in bo deloval kot spletni pajek. Iz pridobljenega besedila HTML se bodo luščili podatki, ki jih potrebujemo. Potrebovali bomo ažurne podatke tečaja kriptovalute na različnih borzah in agregirane zgodovinske podatke. Te podatki se bodo kasneje poslali v bazo podatkov, ki nam jo bo v našem sistemu predstavljala Elasticsearch.



Slika 4.2: Diagram arhitekture sistema.

4.1.3 Baza podatkov

Za bazo podatkov bomo uporabljali Elasticsearch, ki jo bomo podrobno opisali v poglavju 4.2.6. Elasticsearch predstavlja denormalizirano podatkovno hrambo NoSQL. Omogoča shranjevanje, iskanje in analiziranje podatkov v realnem času [34]. Preko aplikacijskega programskega vmesnika bomo na

Elasticsearch pošiljali podatke, ki smo jih pridobili z razvito programsko opremo.

4.1.4 Vizualizacija

Podatke bomo vizualizirali z grafičnim orodjem Kibana, ki je opisano v poglavju 4.2.7. Orodje bo podatke, pridobljene iz vira, vizualiziralo na številne načine, ki jih bo uporabnik specificiral. Prikazani podatki in rezultati iskanja se bodo pridobili s pomočjo Elasticsearcha.

V Kibani si bo uporabnik lahko ustvaril poljubno nadzorno ploščo, kamor bo lahko vključil svoje vizualizacije. Vizualizacije bodo lahko različni grafikoni, tabele in meritve. Na podlagi ustvarjenih vizualizacij se bodo izvajale analize in iskanja po specifičnih podatkih. Omogočeno bo tudi filtriranje podatkov.

4.2 Pregled uporabljenih tehnologij in orodij

V tem poglavju se bomo posvetili pregledu tehnologij in orodij, ki jih bomo uporabili pri razvoju aplikacije. Opisali bomo tehnologije, ki so bile uporabljene v okviru razvoja spletnega pajka, podatkovni hrambi in vizualizacijskem orodju.

4.2.1 Microsoft Visual Studio

Microsoft Visual Studio je integrirano razvojno okolje (IDE) podjetja Microsoft. Uporablja se za razvoj namiznih aplikacij, spletnih mest, storitev in mobilnih aplikacij. Pri razvoju naše aplikacije smo ga uporabili za pisanje in urejanje programske kode spletnega pajka. Visual Studio uporablja vmesnike za razvoj programske opreme, kot so Windows API, Windows Forms, Windows Presentation Foundation, Windows Store in Microsoft Silverlight [75]. Uporablja se lahko tako domorodno kot tudi upravljano kodo. Upravljana

koda (angleško *managed code*) je programska koda, ki se lahko izvaja samo v programskem okolju CLR, ki ga podrobneje opišemo v poglavju 4.2.2.

Visual Studio vključuje urejevalnik kode, razhroščevalnik in omogoča oblikovanje kode. Druga vgrajena orodja so tudi orodja za analiziranje kode, orodje za izdelavo grafičnih aplikacij, spletni oblikovalec, oblikovalec razreda in oblikovalec sheme baz podatkov. Omogoča tudi poljubne vtičnike, ki izboljšujejo funkcionalnost, na primer grafični oblikovalec za domensko specifičen jezik [75]. Vgrajeni jeziki vključujejo C, C++, Visual Basic .NET, C#, F#, JavaScript, TypeScript, XML, XSLT, HTML in CSS . Podpora za druge jezike, kot so Python, Ruby in Node.js, je na voljo preko vtičnikov [75].

4.2.2 Microsoft .NET

Microsoft .NET je programsko okolje, ki se izvaja predvsem v operacijskem sistemu Microsoft Windows. Vključuje veliko knjižnico z imenom Framework Class Library (FCL). Ta zagotavlja jezikovno izmenjavo, kar pomeni, da vsak jezik lahko uporablja kodo, napisano v drugih jezikih [44]. Programi, napisani za okolje .NET, se izvajajo v programskem okolju CLR. To je navidezni stroj, ki nudi storitve, kot so varnost, upravljanje pomnilnika in upravljanje z izjemami [21]. FCL in CLR skupaj tvorita okolje .NET.

FCL nudi uporabniški vmesnik, dostop do podatkov, šifriranje, razvoj spletnih aplikacij, uporabo numeričnih algoritmov in omrežne komunikacije [44]. Programerji izdelujejo programsko opremo tako, da kombinirajo svojo izvorno kodo z okoljem .NET in drugimi knjižnicami. Okolje se pri večini novih aplikacij uporablja za platformo Windows.

Okolje .NET se je začelo kot lastniška programska oprema [62], čeprav je Microsoft skoraj nemudoma standardiziral programsko opremo, tudi pred prvo izdajo. Kljub prizadevanjem za standardizacijo so razvijalci, predvsem tisti v brezplačnih in odprtokodnih skupnostih, izrazili nelagodje z možnostmi kakršnegakoli prostega in odprtokodnega izvajanja, zlasti glede patentov programske opreme. Okolje je pripeljalo do družine platform .NET, ki ciljajo na mobilno računalništvo, vgrajene naprave, druge operacijske sisteme in

vtičnike spletnih brskalnikov. Kompaktna različica ogrodja, .NET Compact, je na voljo na platformah Windows CE in Mobile za pametne telefone. Silverlight je bil na voljo kot vtičnik spletnih brskalnikov. Mono je na voljo za številne operacijske sisteme in je prilagojen najpogostejšim operacijskim sistemom za pametne telefone, kot sta Android in iOS [62].

4.2.3 C#

C# je programski jezik, ki vključuje strogo tipiziranje in ukazna, deklarativna, funkcionalna, generična ter objektno usmerjena načela programske kode. Razvil ga je Microsoft v okviru pobude .NET, kasneje pa ga je Ecma potrdila kot standard [2]. C# je eden od programskih jezikov, ki so namenjeni skupni jezikovni infrastrukturi (CLI). Je splošen, objektno usmerjen programski jezik. Najnovejša različica je C# 7.2 [29], ki jo bomo tudi uporabili v implementaciji spletnega pajka.

Značilnosti, ki jih za jezik C# predpisuje Ecma [2]:

- C# je preprost, splošnonamenski in je objektno usmerjen,
- jezik in njegove implementacije zagotavljajo podporo načelom programskih jezikov, kot so strogo preverjanje tipa, preverjanje mejnih območij polj, odkrivanje poskusov uporabe neinicializiranih spremenljivk in samodejno zbiranje smeti,
- namenjen je uporabi pri razvoju programske opreme, primerne za distribucijo v distribuirano okolje,
- sintaksa je primerna za programerje, ki že poznajo C, C++ ali Javo,
- ima podporo za internacionalizacijo,
- primeren je za pisanje aplikacij za gostiteljske in vgrajene sisteme - od zelo velikih, ki uporabljajo sofisticirane operacijske sisteme (na primer Microsoft Windows, macOS ali Linux), do zelo majhnih, ki imajo namensko specifične funkcije,

- aplikacije so varčne glede porabe pomnilnika in zahtevnosti procesiranja, vendar pa jezik ni bil namenjen neposrednemu konkuriranju zmogljivosti in velikosti s C ali zbirnim jezikom.

4.2.4 Bitbucket

Bitbucket je sistem za upravljanje izvorne kode v obliki spletne storitve, ki omogoča shranjevanje izvorne kode in razvojnih projektov, ki uporabljajo Mercurial ali Git sistem za nadzor revizij kode [14]. Sistem je napisan v programskem jeziku Python. Za pregled revizij in projektov se uporablja orodje Sourcetree [72], ki smo ga uporabili tudi v sklopu razvoja diplomskega dela.

4.2.5 Elastic Stack

Elastic Stack ali ELK je kratica, ki predstavlja tri odprtokodne projekte: Elasticsearch, Logstash in Kibano. V razvoju diplomskega dela bomo za podatkovno hrambo in iskanje uporabili Elasticsearch, ki je opisan v poglavju 4.2.6, in vizualizacijsko orodje Kibana, ki je opisano v poglavju 4.2.7.

4.2.6 Elasticsearch

Elasticsearch je iskalnik s shrambo podatkov [36], ki temelji na programski knjižnici Apache Lucene [11]. Je porazdeljen sistem, kar pomeni, da se ga lahko uporabi na neomejenem številu strežnikov in da lahko upravlja z več petabajti podatkov hkrati. Večnamenski iskalnik besedila vsebuje spletni vmesnik HTTP in uporablja strukturirane dokumente v obliki JSON [36]. Elasticsearch je razvit v programskemu jeziku Java, odjemalci zanj pa so na voljo v jezikih Java, C#, PHP, Python, Apache Groovy in mnogih drugih.

Elasticsearch se lahko uporablja za iskanje vseh vrst dokumentov. Zagotavlja iskanje v realnem času in podpira več odjemalcev hkrati. Odjemalec je na primer program, ki dostopa do aplikacijskega programskega vmesnika.

Dokumenti, ki vsebujejo podatke in so shranjeni v podatkovni hrambi, uporabljajo indekse, ki se nahajajo v ločeni datoteki. Indeksi so razdeljeni na manjše dele in vsak manjši del ima lahko nič ali več replik. Vsako vozlišče vsebuje enega ali več manjših delov indeksov in deluje kot koordinator, ki prenese operacije na pravilen manjši del. Sorodni podatki pogosto uporabljajo isti indeks, ki je sestavljen iz enega ali več manjših delov in nič ali več kopij replik. Ko je indeks ustvarjen, števila primarnih manjših delov ni mogoče spremeniti [36]. V primeru odpovedi strežnika lahko obnovimo indeks. Elasticsearch podpira zahteve v realnem času, kar ustreza zahtevam naše diplomske naloge.

Elasticsearch ponuja razvijalcem tudi svoj API z metodami, ki jih uporablja Kibana. API omogoča metode za branje, pisanje, posodobitev in izbris podatkov v realnem času [37]. Vse operacije, ki so mogoče na grafičnem vmesniku Kibane, so možne tudi z uporabo API. Na GET API lahko na primer pošljemo zahtevek za iskanje točno določenih podatkov, kot smo to prikazali v poglavju 6.1. Za testiranje zahtevkov se lahko uporabi orodje za razvijalce znotraj Kibane, ki omogoča pošiljanje zahtevkov in prikaz rezultatov.

4.2.7 Kibana

Kibana je odprtokodno orodje za vizualizacijo podatkov, ki se nahajajo v Elasticsearchu. Ponuja možnosti vizualizacije nad vsebino, indeksirano na gruči Elasticsearch [51]. Uporabniki lahko iz velikih količin podatkov ustvarijo različne vizualizacije v obliki preprostih grafikonov, lahko pa ustvarijo tudi bolj napredne vizualizacije, na primer časovna vrsta, ki prikaže zgrajeno časovno vrsto z uporabo funkcionalnih izrazov. Bolj podrobno vizualizacije opišemo v poglavju 6.2.

Poglavje 5

Razvoj aplikacije

V pričujočem poglavju bomo opisali celoten postopek razvoja aplikacije za spremljanje gibanja tečajev v realnem času. Opisali bomo vse korake, ki so bili potrebni za vzpostavitev okolja, v katerem bo delovala aplikacija. Razvoj aplikacije je potekal na delovni postaji, kjer so bili nameščeni vsi potrebni strežniki.

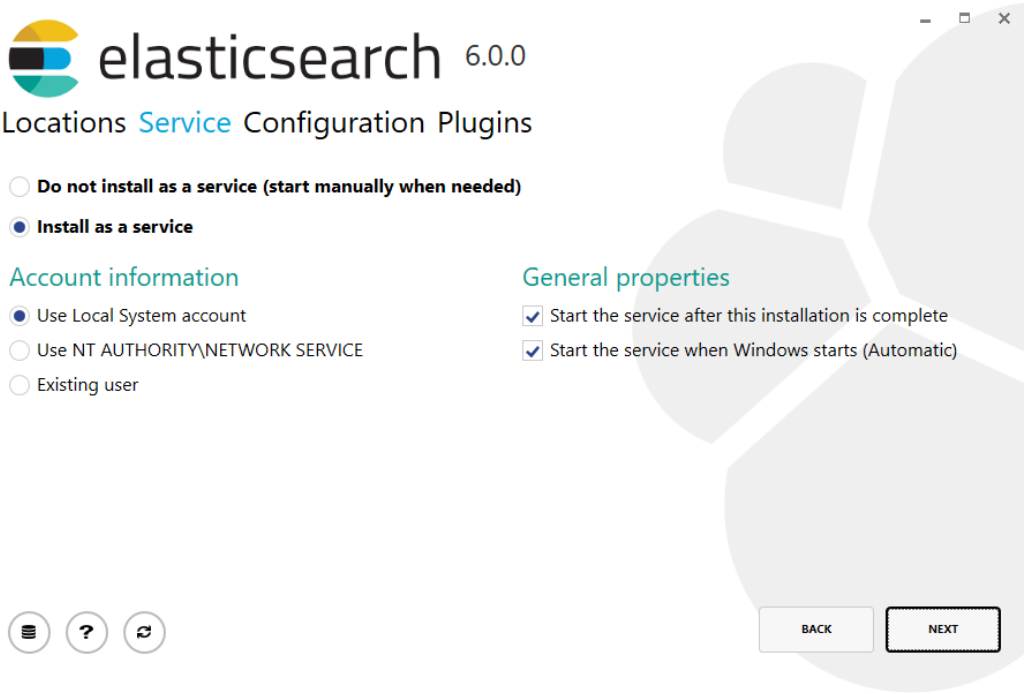
5.1 Namestitev podatkovne hrambe Elasticsearch

Prenesli smo namestitveno datoteko MSI [35] za Elasticsearch, ki namesti Elasticsearch na računalnik kot storitev Windows. Odpremo ukazno vrstico, se postavimo v mapo namestitvene datoteke in izvedemo ukaz, ki je viden na izpisu 5.1.

```
1 start /wait msiexec.exe /i elasticsearch-6.1.3.msi /qn
```

Izpis 5.1: Ukaz za namestitev Elasticsearcha.

Pojavi se namestitveno okno, kot je vidno na sliki 5.1. Program namestimo kot storitev Windows z lokalnim sistemskim računom. Uporabimo privzete nastavitve za imenike, ki naj jih Elasticsearch uporabi. Vse ostale nastavitve prav tako pustimo privzete in program namestimo.



Slika 5.1: Namestitveno okno programa Elasticsearch.

Po končanem procesu si ogledamo beležko namestitve z ukazom, vidnim na izpisu 5.2. Tako preverimo, da med namestitvijo ni prišlo do kakršnekoli nepričakovane napake.

```
1 start /wait msiexec.exe /i elasticsearch-6.1.3.msi /qn
    /l install.log
```

Izpis 5.2: Ukaz za ogled namestitvene beležke.

Do podatkovne hrambe lahko dostopamo preko spletnega naslova `http://localhost:9200/`. Dostop do omenjenega spletnega naslova nam mora vrniti podatke o Elasticsearchu, kot je vidno na izpisu 5.3.

```
{
  "name" : "Cp8oag6",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "AT69_T_DTp-1qgIJlatQqA",
  "version" : {
    "number" : "6.1.3",
    "build_hash" : "f27399d",
    "build_date" : "2016-03-30T09:51:41.449Z",
    "build_snapshot" : false,
    "lucene_version" : "7.1.0",
    "minimum_wire_compatibility_version" : "1.2.3",
    "minimum_index_compatibility_version" : "1.2.3"
  },
  "tagline" : "You Know, for Search"
}
```

Izpis 5.3: Podatki vozlišča Elasticsearch v obliki JSON.

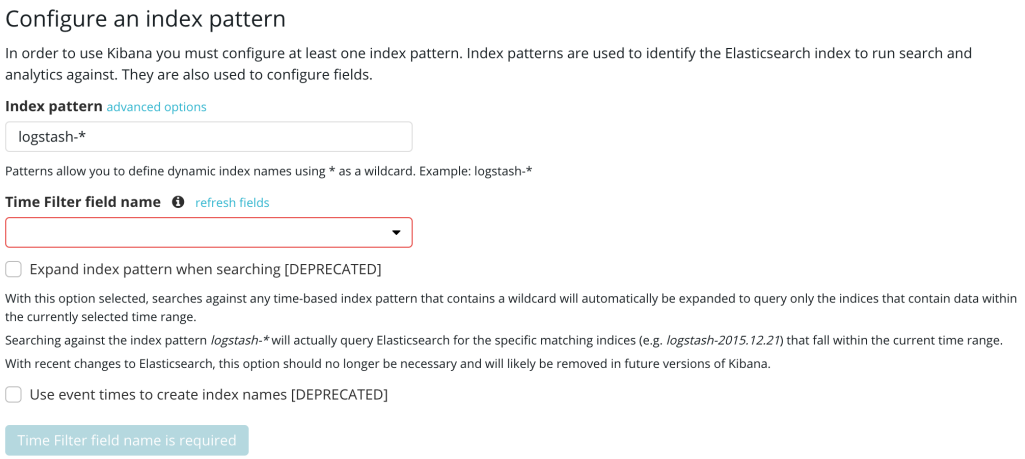
5.2 Namestitev Kibane

Po uspešni vzpostavitvi podatkovne hrambe Elasticsearch, kar je opisano v poglavju 5.1, sedaj lahko namestimo še grafično orodje Kibana. Prenesemo arhivsko datoteko [52], ki jo najdemo na uradni spletni strani. Arhivsko datoteko razširimo na poljubno lokacijo. Odpremo ukazno vrstico PowerShell in se pomaknemo v namestitveno mapo ter izvedemo ukaz, viden na izpisu 5.4.

```
1    .\kibana.bat
```

Izpis 5.4: Ukaz za namestitev Kibane.

Orodje se avtomatsko poveže s storitvijo Elasticsearch. Dostopno je preko spletnega naslova `http://localhost:5601/`. Ob povezavi na omenjeno spletno mesto moramo ustvariti vsaj en indeks, ki se bo kasneje uporabljal za naše dokumente, ki bodo vsebovali podatke tečajev kriptovalut. V našem primeru ustvarimo indeks z imenom `ticker`. Konfiguracija indeksa je vidna na sliki 5.2.



Slika 5.2: Prikaz konfiguracije indeksa v Kibani.

5.3 Razvoj spletnega pajka

V okolju .NET bomo ustvarili nov projekt, ki nam bo služil kot namizna konzolna aplikacija za črpanje podatkov s spletne strani CoinMarketCap. Aplikacija bo napisana v programskem jeziku C# in bo delovala kot spletni pajek. Spletni pajek je avtomatiziran program oziroma skripta, ki samostojno preiskuje spletne strani na internetu. Ob tem skuša izvleči želene podatke iz spletnih strani [4].

Črpati želimo podatke za kriptovalute Bitcoin, Litecoin in Ethereum. Te so na voljo na spletnem naslovu „`https://coinmarketcap.com/currencies/kriptovaluta/`”, kjer `kriptovaluta` predstavlja ime kriptovalute v angleščini.

Vse podprte kriptovalute nam bo predstavljala podatkovna struktura slovar, viden na izpisu 5.5. Slovar uporabimo z namenom, da polno ime kriptovalute kasneje povežemo z njeno okrajšavo, ki jo pošljemo v Elasticsearch.

```
private static readonly Dictionary<string, string> Currencies = new
    Dictionary<string, string>
{
    {"bitcoin", "BTC"},
    {"litecoin", "LTC"},
    {"ethereum", "ETH"},
};
```

Izpis 5.5: Slovar s podprtimi kriptovalutami.

Ustvarili bomo metodo, ki bo črpala ažurne podatke izbrane kriptovalute. Pridobili bomo besedilo HTML s spletne strani CoinMarketCap, ki ga bomo razčlenili in s tem pridobili podatke o tečaju kriptovalute na različnih borzah. Metoda bo vračala podatke v obliki podatkovne strukture `List<MarketData>`, kjer `MarketData` predstavlja razred, ki ga bomo pošiljali na Elasticsearch. Struktura razreda je vidna na izpisu 5.6. Razred vsebuje ime borze, časovni žig, ime kriptovalute, 24-urno količino trgovanja, ceno in tržno omejitev v ameriških dolarjih.

Besedilo HTML vsebuje elemente HTML, te pa se začnejo in končajo z oznakami, kot na primer in . S spletnim pajkom besedilo HTML in njegove elemente shranimo v pomnilnik v obliki drevesa, v katerem se nahajajo vozlišča. Iz pridobljenega besedila HTML želimo izločiti samo vozlišče `markets-table`, ki je vidno na izpisu 5.7. Za razčlembo besedila HTML bomo uporabili knjižnico Html Agility Pack [47], ki uporablja algoritme, ki se sprehajajo po vseh vozliščih HTML. Združuje uporabo jezika HTML ter xPath [47]. Metoda je prikazana na izpisu 5.8.

Ažurne podatke lahko uspešno pridobimo z metodo `GetTicker`, sedaj pa želimo razviti še metodo, ki bo pridobivala zgodovinske podatke kriptovalut. Ti so agregirani iz vseh podprtih borz in so prikazani na dnevni osnovi.

```
public class MarketData
{
    public DateTime Timestamp { get; set; }
    public string Exchange { get; set; }
    public string Currency { get; set; }
    public decimal Volume24h { get; set; }
    public decimal PriceUsd { get; set; }
    public decimal MarketCapUsd { get; set; }
}
```

Izpis 5.6: Razred MarketData.

Omenjena metoda je prikazana na izpisu 5.9.

Zgodovinski podatki so na voljo na istem spletnem naslovu, vendar pa moramo naslovu dodati še začetni in končni datum. Razčlemba podatkov bo potekala po istem principu, ki smo ga uporabili pri metodi `GetTicker`. Metodi smo prav tako dodali pridobivanje cene, ki omogoča pridobitev podatkov kriptovalut za točno določen datum.

Pri zgodovinskih podatkih prejmemo štiri različne cene za specifično obdobje:

- najvišjo ceno,
- najnižjo ceno,
- ceno ob odprtju,
- ceno ob zaprtju.

Ob branju ažurnih podatkov z metodo `GetTicker` pridobimo povprečno ceno kriptovalute. Ob branju zgodovinskih podatkov pridobimo štiri različne cene, zato moramo iz le-teh naknadno izračunati povprečje, kot vidno na izpisu 5.9.

```
<table id="markets-table" class="table no-border table-condensed">
<thead>
<tr>
    <th class="text-right sortable">#</th>
    <th id="th-source" class="sortable">Source</th>
    <th id="th-pair" class="sortable">Pair</th>
    <th class="text-right sortable">Volume (24h)</th>
    <th class="text-right sortable">Price</th>
    <th class="text-right sortable">Volume ()</th>
    <th class="text-right sortable">Updated</th>
</tr>
</thead>
<tbody>
    <tr >
        <td class="text-right">1</td>
        <td><a href="/exchanges/bitfinex/">Bitfinex</a></td><td><a
            href="https://www.bitfinex.com/t/ETH:USD"
            target="_blank">ETH/USD</a></td>
        <td class="text-right">

            <span class="volume" data-usd="460349000.0" data-btc="55419.9"
                data-native="550222.0">
                $460,349,000
            </span>
        </td>
        <td class="text-right">

            <span class="price" data-usd="836.66" data-btc="0.100723"
                data-native="836.66">
                $836.66
            </span>
        </td>
        <td class="text-right">
            <span data-format-percentage
                data-format-value="8.87619954766">8.88</span>
        </td>
        <td class="text-right " >Recently</td>
    </tr>
```

Izpis 5.7: Del pridobljenega besedila HTML.

```
private static List<MarketData> GetTicker(string currency)
{
  using (var client = new WebClient())
  {
    string html = client.DownloadString(
    $"https://coinmarketcap.com/currencies/{currency}/");
    var doc = new HtmlDocument();
    doc.LoadHtml(html);

    bool GetUsd(HtmlAttribute x) => x.Name == "data-usd";

    HtmlNode table = doc.GetElementbyId("markets-table");
    HtmlNode markets = table.SelectSingleNode("tbody");

    var marketCap = TryParseOrDefault(doc.DocumentNode.SelectSingleNode(
                "/html/body/div[4]/div/div[1]/div[4]/" +
                "div[1]/div[1]/div[2]/span[1]/span[1]").InnerText);

    return (from market in markets.SelectNodes("tr")
        select market.SelectNodes("td")
        into nodes
        let volumeValue = nodes[3].SelectSingleNode("span").
        Attributes.First(GetUsd).Value
        let priceValue = nodes[4].SelectSingleNode("span").
        Attributes.First(GetUsd).Value
        select new MarketData
        {
            Exchange = nodes[1].InnerText,
            Currency = nodes[2].InnerText,
            Volume24h = TryParseOrDefault(volumeValue),
            PriceUsd = TryParseOrDefault(priceValue),
            Timestamp = DateTime.UtcNow,
            MarketCapUsd = marketCap
        }).ToList();
  }
}
```

Izpis 5.8: Metoda, ki se uporablja za pridobivanje ažurnih podatkov s spletne strani CoinMarketCap.

```
private static List<MarketData> GetHistory(string currency, DateTime?
    forDate = null)
{
  using (var client = new WebClient())
  {
    string html =
        client.DownloadString($"https://coinmarketcap.com/currencies/" +
        $"{currency}/historical-data/?start=20101212&end=20201212");
    var doc = new HtmlDocument();
    doc.LoadHtml(html);
    HtmlNodeCollection table =
        doc.DocumentNode.SelectNodes("//table//tbody//tr");
    var data = new List<MarketData>();
    foreach (HtmlNode row in table)
    {
        var tableData = row.SelectNodes("td");
        decimal[] prices =
        {
            TryParseOrDefault(tableData[1].InnerText),
            TryParseOrDefault(tableData[2].InnerText),
            TryParseOrDefault(tableData[3].InnerText),
            TryParseOrDefault(tableData[4].InnerText)
        };
        var timestamp = DateTime.Parse(tableData[0].InnerText);
        var marketDataRow = new MarketData
        {
            Timestamp = new DateTime(timestamp.Year, timestamp.Month,
                timestamp.Day, 23, 59, 59),
            Currency = Currencies[currency],
            Exchange = "All",
            MarketCapUsd = TryParseOrDefault(tableData[6].InnerText),
            PriceUsd = prices.Average(),
            Volume24h = TryParseOrDefault(tableData[5].InnerText)
        };
        data.Add(marketDataRow);
        if (forDate != null && timestamp.Date == forDate.Value.Date)
        {
            return new List<MarketData> {marketDataRow};
        }
    }
    return data;
  }
}
```

Izpis 5.9: Metoda, ki se uporablja za pridobivanje zgodovinskih podatkov s spletne strani CoinMarketCap.

5.4 Pošiljanje podatkov

Podatke, ki jih pridobiva spletni pajek, želimo poslati v svojo podatkovno hrambo, ki jo predstavlja Elasticsearch. Za komunikacijo med konzolno namizno aplikacijo in Elasticsearchom se bo uporabila knjižnica NEST [61]. Knjižnica ponuja funkcionalnost visokonivojskega odjemalca, ki se poveže z Elasticsearchom. Na nivoju okolja .NET uporablja strogo tipizirane poizvedbe, ki se preobrazijo v poizvedbe, ki jih prepozna Elasticsearch [61].

Podatke želimo pošiljati v Elasticsearch in uporabiti indeks, ki smo ga določili v konfiguraciji. Tega smo že ustvarili, kar smo opisali v poglavju 5.2. Ustvarili bomo razred `ElasticHelper` z metodo `SendToIndex`, ki bo zadolžena za pošiljanje podatkov. Podatke se bo pošiljalo v paketih po 100.000 zapisov, kar omogoča manjše število klicev vmesnika Elasticsearch in posledično hitrejše pošiljanje podatkov. Metoda bo ustvarjena generično, kar omogoči razvijalcu, da lahko na poljuben indeks pošlje katerokoli podatkovno strukturo. Omenjena metoda je vidna na izpisu 5.10.

5.5 Konzolna aplikacija

V aplikaciji smo ustvarili funkcionalnost, ki omogoča pobiranje in pošiljanje podatkov. Ustvarjene metode želimo uporabiti v svoji namizni konzolni aplikaciji. Vstopna točka naše aplikacije bo glavna metoda `Main`, kjer bomo pognali metode in preko konzolnega uporabniškega vmesnika obveščali uporabnika o napakah in ostalih informacijah, kot je vidno na izpisu 5.11.

Aplikacija bo vsakih 10 sekund črpala podatke s spletne strani CoinMarketCap in jih nato pošiljala v podatkovno bazo, ki jo predstavlja Elasticsearch. S tem bomo uporabnikom omogočili prikaz realno-časovnih podatkov, ki bodo na voljo v Kibani. Metoda se imenuje `StartTickers` in je vidna na izpisu 5.12.

Prav tako želimo imeti na voljo zgodovinske podatke, ki se raztezajo od začetka trgovanja s kriptovalutami. Te bomo pridobili samo enkrat ob vzpostavitvi celotnega sistema, kasneje pa se bodo agregirani zgodovinski podatki

```
public async Task SendToIndex<T>(string index, IList<T> records)
                        where T : class
{
    var settings = new ConnectionSettings().DefaultIndex(index);
    var client = new ElasticClient(settings);

    int lineCount = records.Count;
    int total = 0;

    while (lineCount > total)
    {
        var descriptor = new BulkDescriptor();
        descriptor.IndexMany(records.Skip(total).Take(100000));
        var result = await client.BulkAsync(descriptor);
        total += 100000;
    }
}
```

Izpis 5.10: Metoda, ki omogoča pošiljanje podatkov na Elasticsearch.

črpali dnevno. Metodo, ki je vidna na izpisu 5.13, lahko uporabimo za črpanje celotne zgodovine, kar specificiramo s parametrom `parseEntireHistory`.

```
public static void Main(string[] args)
{
    try
    {
        Console.ForegroundColor = ConsoleColor.Green;
        StartHistory(false);
        StartTickers();
        Console.WriteLine("Downloading enabled. Type exit to stop the
            program.");
        while (Console.ReadLine()?.ToLowerInvariant() != "exit") { }
    }
    catch (Exception ex)
    {
        Console.Clear();
        Console.ForegroundColor = ConsoleColor.Red;
        Console.WriteLine("An exception has occured, info:");
        Console.ForegroundColor = ConsoleColor.White;
        Console.WriteLine("{1}{0}{1}Press any button to exit...", ex,
            Environment.NewLine);
        Console.Read();
    }
}
```

Izpis 5.11: Vstopna točka namizne konzolne aplikacije.

```
private static async void StartTickers()
{
    var helper = new ElasticHelper();
    while (true)
    {
        var results = new List<MarketData>();
        foreach (var currency in Currencies)
        {
            results.AddRange(GetTicker(currency.Key));
        }

        await helper.SendToIndex(TickerIndex, results);

        Thread.Sleep(TimeSpan.FromSeconds(10));
    }
}
```

Izpis 5.12: Metoda, ki se uporablja za črpanje in pošiljanje ažurnih podatkov.

```
private static async void StartHistory(bool parseEntireHistory)
{
    var helper = new ElasticHelper();
    var results = new List<MarketData>();

    if (parseEntireHistory)
    {
        foreach (var currency in Currencies)
        {
            results.AddRange(GetHistory(currency.Key));
        }
        await helper.SendToIndex(TickerIndex, results);
        return;
    }

    while (true)
    {
        results = new List<MarketData>();
        foreach (var currency in Currencies)
        {
            results.AddRange(GetHistory(currency.Key,
                DateTime.UtcNow.AddDays(-1)));
        }
        await helper.SendToIndex(TickerIndex, results);
        Thread.Sleep(TimeSpan.FromDays(1));
    }
}
```

Izpis 5.13: Metoda, ki se uporablja za črpanje in pošiljanje zgodovinskih podatkov.

Poglavje 6

Rezultati

V tem poglavju se bomo posvetili analizi podatkov, ki smo jih pridobili s spletnim pajkom in jih zapisali v podatkovno hrambo Elasticsearch. Rezultat pridobljenih podatkov bo viden v obliki gibanja tečajev kriptovalut v orodju Kibana. Naše orodje pridobiva podatke s spletne strani CoinMarketCap, ki podpira več kot 8500 različnih trgov z več kot 1500 kriptovalutami [22]. Skušali bomo najti najprimernejše vizualizacije, ki koristijo uporabnikom implementiranega orodja.

6.1 Iskanje podatkov

Kibana omogoča iskanje po surovih podatkih, ki so bili zapisani v podatkovni hrambi Elasticsearch. Surovi podatki so podatki, ki še niso bili obdelani s strani uporabnika. To omogoča funkcionalnost `Discover`, ki je na voljo na spletni strani orodja Kibana na naslovu `http://localhost:5601/`. Funkcionalnost ponuja iskanje preko poizvedovalnega jezika Apache Lucene [11]. Podatki, shranjeni v Elasticsearchu, se nahajajo v obliki JSON. Primer poslanega zapisa je viden na izpisu 6.2. Poizvedovanje po teh podatkih nam prikaže tudi bolj strukturirano obliko v obliki tabele, kot je vidno na sliki 6.1. Poizvedujemo lahko po kateremkoli polju, ki je določeno v indeksu. Na grafičnem vmesniku lahko določimo poljubno časovno vrsto. Časovna vrsta

je časovno urejeno zaporedje številčnih podatkov, ki izražajo vrednost neke spremenljivke. Podatki so navadno izmerjeni v enakih časovnih intervalih, na primer vsako minuto, vsako uro, vsak dan [10]. Na že pridobljeni množici zapisov lahko uporabimo še filter, ki se ga določi v uporabniku prijazni grafični obliki. Filtri se uporabijo za hitro iskanje in delo s podmnožico podatkov, ki smo jih pridobili. Pri prikazu končnih zapisov lahko izločimo polja, ki nas ne zanimajo.

Na primer, če bi želeli poiskati vse pridobljene podatke, ki veljajo za valutni par `ETH/USD` na borzi BitStamp 15. januarja, potem bi vnesli poizvedbo, ki je vidna na zapisu 6.1. Rezultat poizvedbe je množica zapisov, ki je vidna na sliki 6.2.

```
currency:"ETH/USD" AND exchange:"BitStamp" AND
   timestamp:"2018-01-15"
```

Izpis 6.1: Poizvedba v obliki poizvedovalnega jezika Apache Lucene.

6.2 Vizualizacije

Kibana podpira številne oblike vizualizacij, ki jih uporabnik lahko ustvari na podlagi pridobljenih podatkov. Ustvarimo lahko vizualizacije, kot na primer:

- ploščinski grafikon, viden na sliki 6.7, ki poudari količino pod črtnim grafikonom,
- toplotno karto, ki zasenči celice v matriki podatkov,
- palični ali stolpčni grafikon, viden na sliki 6.5, ki prikaže neprekinjeno spremenljivko na obeh oseh,
- črtni grafikon, viden na sliki 6.4, ki poudari trende,

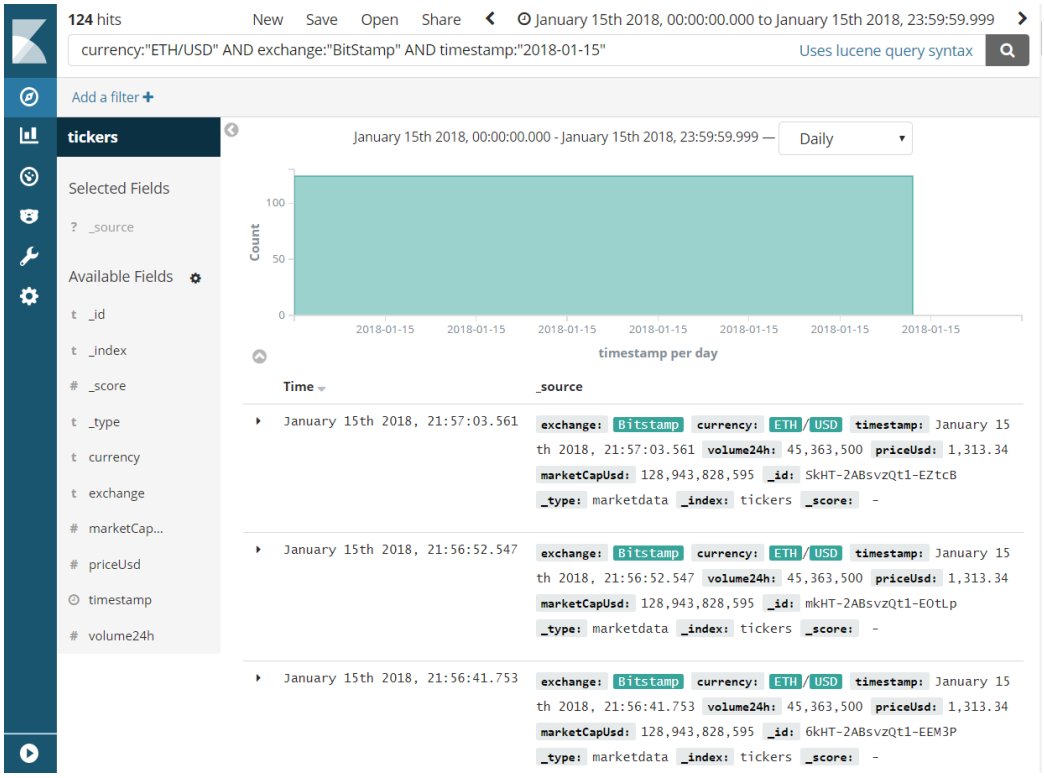
```
{
  "_index": "tickers",
  "_type": "marketdata",
  "_id": "BuSH-2ABdidQt1-E911B",
  "_score": 1,
  "_source": {
    "timestamp": "2018-01-15T21:57:03.5613172Z",
    "exchange": "Bitstamp",
    "currency": "ETH/USD",
    "volume24h": 45363500,
    "priceUsd": 1313.34,
    "marketCapUsd": 128943828595
  },
  "fields": {
    "timestamp": [
      "2018-01-15T21:57:03.561Z"
    ]
  }
}
```

Izpis 6.2: Zapis poslanega razreda `MarketData` v obliki JSON.

t	_id	SkHT-2ABsvzQt1-EZtcB
t	_index	tickers
#	_score	1
t	_type	marketdata
t	currency	ETH/USD
t	exchange	Bitstamp
#	marketCapUsd	128,943,828,595
#	priceUsd	1,313.34
🕒	timestamp	January 15th 2018, 21:57:03.561
#	volume24h	45,363,500

Slika 6.1: Zapis poslanega razreda `MarketData` v obliki strukturirane tabele.

- tortni grafikon, viden na sliki 6.6, ki primerja dele celote,
- podatkovno tabelo, ki prikaže vrednosti v tabeli,
- merilnik, viden na sliki 6.8, ki označuje stanje meritve glede na referenčno vrednost,
- ciljni grafikon, ki prikaže bližino do ciljne referenčne vrednosti,
- meritev, ki prikazuje izračun kot številko,
- zemljevid, ki na mapi prikaže koordinate podatkov,
- časovno vrsto, ki prikaže zgrajeno časovno vrsto z uporabo funkcionalnih izrazov. Funkcionalni izrazi so matematične funkcije, ki jih lahko izvajamo nad časovno vrsto. Namesto grafičnega vmesnika lahko uporabimo posebno sintakso, ki omogoča bolj razširjeno funkcionalnost, kot na primer izris podatkov različnih indeksov ali virov podatkov,

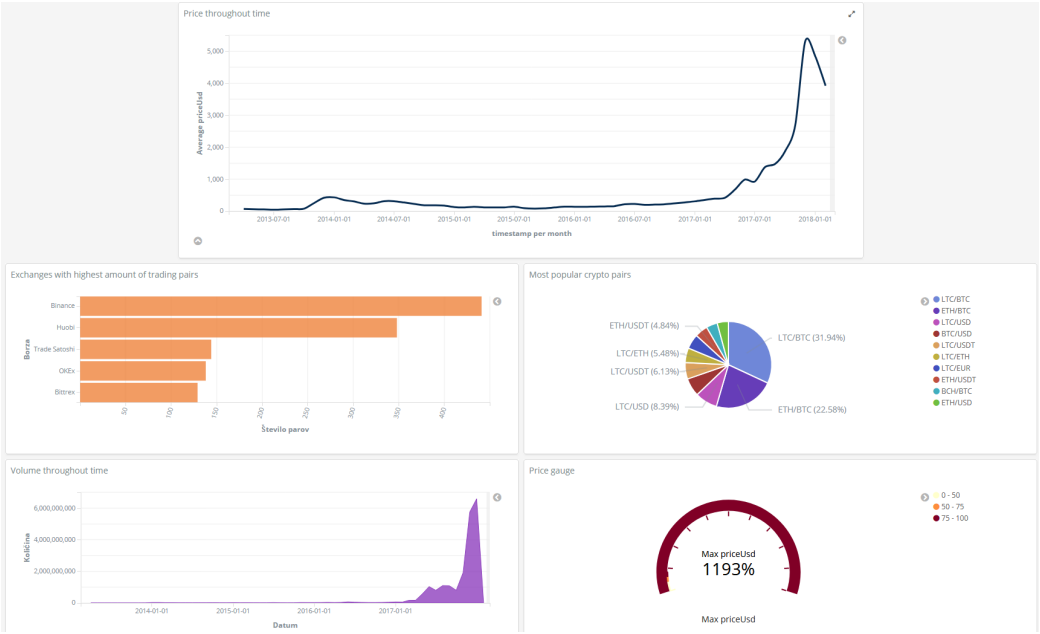


Slika 6.2: Pregled rezultatov poizvedbe v Kibani.

- skupino besed, kjer so velikosti besed določene z meritvami.

Za različne vrste vizualizacij lahko uporabimo filtre in poizvedbe, ki smo jih opisali v poglavju 6.1. Različne vizualizacije lahko postavimo na skupno mesto, ki ga predstavlja pregledna plošča, nad katero lahko prav tako izvajamo filtriranje in poizvedovanje. Ustvarjena pregledna plošča je vidna na sliki 6.3. V naslednjem odlomku bomo prikazali in opisali vizualizacije, ki jih lahko ustvarimo ter s tem prikazali zmožnosti orodja Kibana. Ustvarjene vizualizacije bomo primerjali v naslednjem poglavju 6.3.

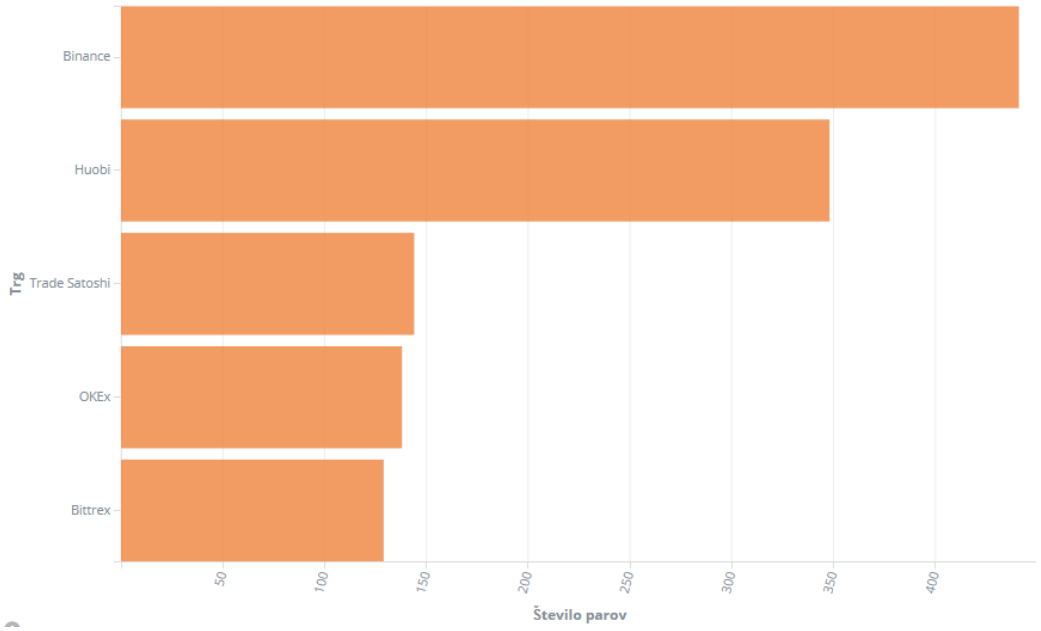
Na začetku bomo ustvarili najbolj tipičen grafikon, ki se uporablja pri spremljanju gibanja tečajev kriptovalut. To je črtni grafikon, s katerim bomo poudarili ceno gibanja tečaja. Na ordinatno os bomo postavili polje `priceUsd`, ki bo predstavljalo povprečno ceno kriptovalute. Na abscisno



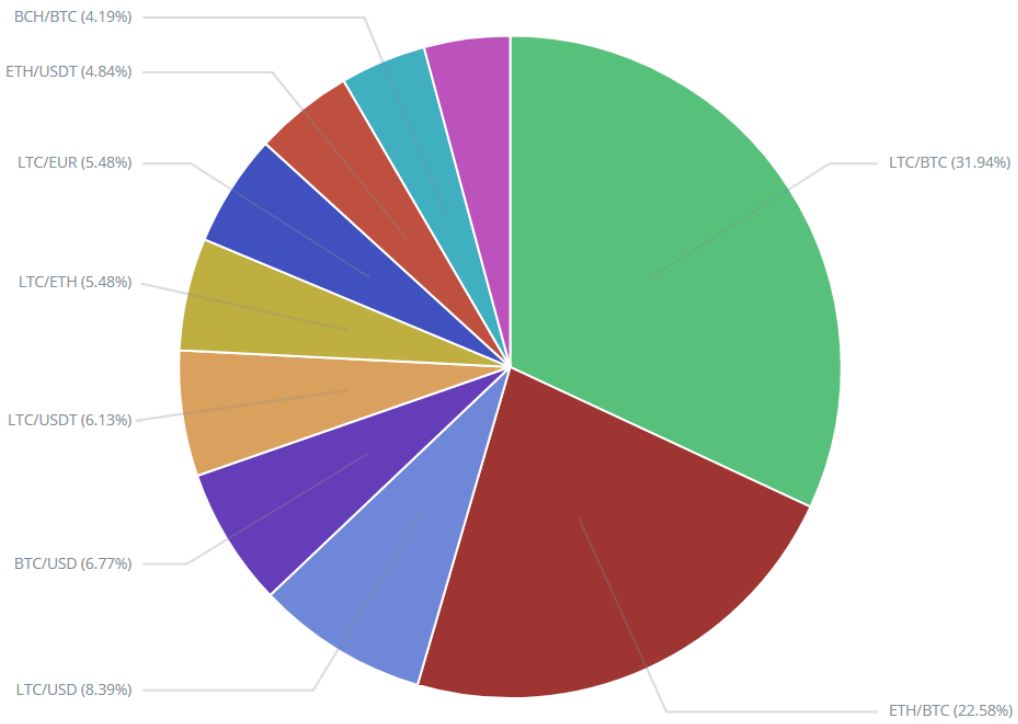
Slika 6.3: Pregledna plošča s poljubnimi grafikoni.

os bomo postavili polje `timestamp`, ki predstavlja časovni žig, nad katerim je možno izvajati operacije časovnih vrst. Uporabnik bo kasneje lahko na ustvarjenem grafikonu spremenil vrsto skale, na primer iz linearne v logaritmično. Za spremljanje količine trgovanja bomo ustvarili ploščinski grafikon. Na ordinatno os bomo postavili polje `volume24h`, ki bo predstavljalo povprečno trgovano količino kriptovalute. Na abscisno os bomo za potrebe operacij časovnih vrst postavili polje `timestamp`. Za prikaz borz, ki podpirajo največje število kriptovalutnih parov, bomo ustvarili palični grafikon. Za izračun števila borz bomo uporabili polje `exchange`, nad katerim bo Kibana apliciral agregacijo. Deset najpopularnejših kriptovalutnih tržnih parov lahko prikažemo kot del celote s tortnim grafikonom, kjer uporabimo polje `currency`. Ustvarili smo tudi merilnik cene, ki uporablja polje `priceUsd`. Merilnik v obliki odstotkov prikazuje trenutno ceno glede na določeno referenčno vrednost.

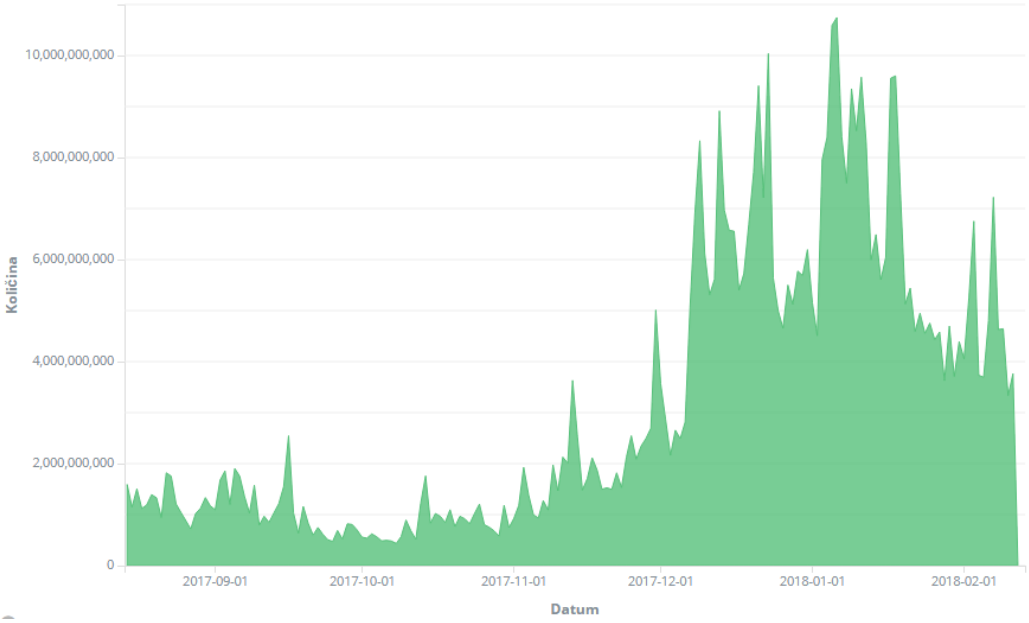
Slika 6.4: Črtni grafikon za spremljanje gibanja tečaja kriptovalute skozi čas.



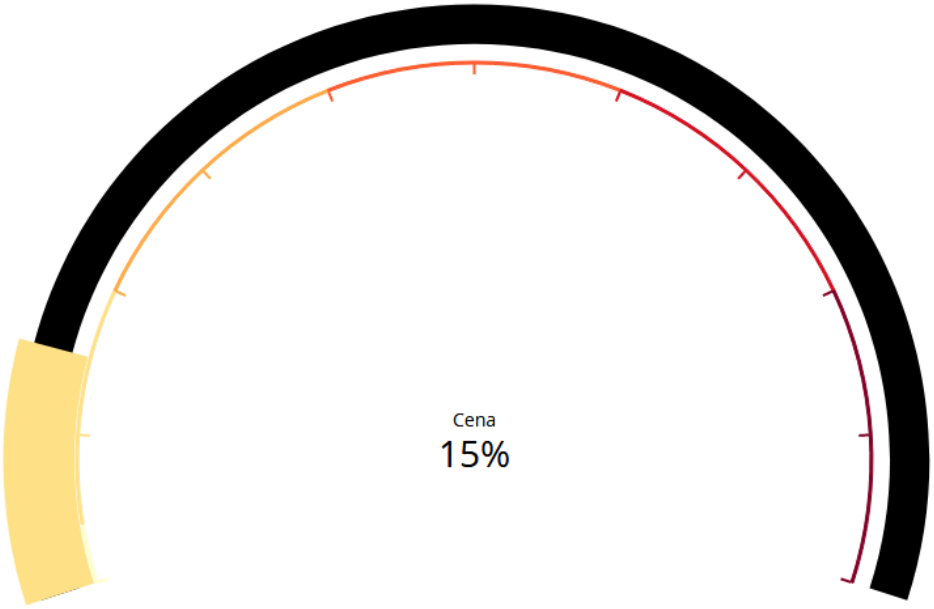
Slika 6.5: Palični grafikon za spremljanje borz z največjim številom kriptovalutnih parov.



Slika 6.6: Tortni grafikon za spremljanje deset najpopularnejših kriptovalutnih parov.



Slika 6.7: Ploščinski grafikon za spremljanje količine trgovanja kriptovalute skozi čas.



Slika 6.8: Merilnik, ki označuje stanje cene kriptovalute glede na referenčno vrednost.

6.3 Primerjava s sorodnimi deli

Sorodna dela ter njihove prednosti in slabosti smo že opisali v poglavju 2. To sta spletni strani CoinMarketCap in CryptoCompare. Sedaj se bomo osredotočili še na primerjavo z našo aplikacijo.

Omogočamo dostop do surovih podatkov, ki so bili poslani v podatkovno hrambo Elasticsearch. Surovi podatki predstavljajo besedilo s tečaji kriptovalut v obliki JSON. Uporabnik lahko podatke prenese na svoj računalnik v obliki tekstovne datoteke ali pa jih pridobi z uporabo Elasticsearch API, ki smo ga opisali v poglavju 4.2.6. CoinMarketCap in CryptoCompare uporabniku ne omogočata preprostega prenosa besedilnih datotek s podatki. CryptoCompare sicer omogoča dostop do podatkov preko njihovega API, vendar je ta omejen s številom zahtevkov. Naš API ni omejen. Do podatkov v naši podatkovni bazi lahko dostopamo preko našega API kar s poizvedovalnim jezikom Apache Lucene, kar omogoča večjo fleksibilnost od CryptoCompare API, kjer moramo najti specifično metodo, ki vrača rezultate, ki jih potrebujemo.

Iskanje podatkov je v naši aplikaciji mogoče izvajati kar preko grafičnega vmesnika orodja Kibana, pri CryptoCompare je to mogoče samo preko API, CoinMarketCap pa tega sploh ne omogoča. Uporabnik lahko vsako iskanje filtrira in izloči polja, ki jih ne potrebuje. Filtriranja sorodni deli ne podpirata.

CoinMarketCap in CryptoCompare omogočata samo eno vrsto vizualizacije in sicer črtni grafikon, kot vidno na slikah 2.2 in 2.4. Ta prikazuje in poudari ceno gibanja tečaja kriptovalute. Naša aplikacija podpira več kot 10 različnih vrst vizualizacij, ki smo jih ustvarili in opisali v poglavju 6.2. Vizualizacije lahko uporabnik ustvari preko grafičnega vmesnika in uporabi katerokoli polje, ki je na voljo v indeksu tečajev kriptovalut. Vizualizacija se lahko ustvari generično, kasneje pa se za prikaz podatkov na vizualizaciji uporabi poizvedovanje in filtriranje. Poizvedbe in filtre lahko shranimo in jih uporabimo na nadzorni plošči, ki prikazuje več vrst vizualizacij. Omogočamo tudi prilagodljivo skalo, spreminjanje barv grafikonov ter časovne vrste s funkci-

onalnimi izrazi. Sorodni deli ne vsebujeta nadzornih plošč in ne omogočata ustvarjanje poljubnih vizualizacij.

Poglavje 7

Sklepne ugotovitve

V diplomski nalogi smo na začetku preverili sorodna dela s podobnimi funkcionalnostmi, ki smo si jih zastavili za cilj. Izbrali smo tri kriptovalute, ki smo jih podprli v prototipu rešitve. Začrtali smo si želeno arhitekturo sistema, ki smo jo kasneje realizirali v obliki realno-časovne aplikacije za vizualizacijo gibanja tečajev kriptovalut. Izbrali smo najprimernejši vir podatkov. To je spletna stran CoinMarketCap. V lokalnem okolju smo vzpostavili celoten ekosistem začrtane aplikacije. V zalednem sistemu smo uporabili spletnega pajka, ki smo ga razvili v okolju .NET. Njegov namen je bil, da črpa informacije o gibanju tečajev kriptovalut s spleta in jih pretvori v primerno obliko. Izluščene podatke smo poslali v podatkovno shrambo Elasticsearch. Podatke smo kasneje vizualizirali v obliki grafikonov z orodjem Kibana.

Ugotovili smo, kako s spletnih strani izluščiti podatke in jih pretvoriti v razvijalcu prijazno obliko. Prav tako smo ugotovili, kako komunicira okolje .NET z okoljem Elastic. Poleg tega smo predstavili teoretično ozadje problemske domene kriptovalut. Podrobno smo opisali izbrane kriptovalute ter njihovo tehnologijo. Spoznali smo se s tehnologijo verige blokov in z vsemi njenimi prednostmi ter slabostmi. Ugotovili smo, da uporabljene tehnologije, pristopi in arhitektura sistema niso omejeni samo na problemsko domeno kripto sveta. Izdelana rešitev se lahko prenese na številne ostale probleme, kjer je potrebna realno-časovna vizualizacija podatkov. Pokazali

smo, kako preprosto je uporabljati orodje Elastic, še posebej vizualizacijsko orodje Kibana. Ugotovili smo, da je Kibana uporabniku prijazna rešitev in ponuja večje število vizualizacij. Ekosistem .NET s programskim jezikom C# je idealna izbira za razvijalca, ki bi želel razviti napredno rešitev s spletnim pajkom.

Izdelana rešitev vseeno pušča še veliko prostora za nadgradnje. Spletnega pajka bi lahko dodelali, da bi deloval hitreje in uporabljal več podatkovnih virov. S tem bi uporabniku omogočili natančnejše podatke in posledično zanesljivejšo analizo. V aplikacijo bi prav tako lahko dodali večje število kriptovalut, ker smo trenutno podprli samo tri. Celotni sistem je bil nameščen v lokalnem okolju, kar bi pri prehodu v produkcijo morali prenesti na storitev, ki omogoča gostovanje tovrstnih orodij. Možna izboljšava programa, ob prehodu na produkcijsko okolje, bi bila, da bi projekt gostovali na strežniku.

Literatura

Viri

[1] Andreas Antonopoulos. *Mastering Bitcoin: Unlocking Digital Crypto-Currencies*. O'Reilly Media, 2014.

[2] „C# Language Specification". V: *Standard Ecma* 334.5 (2017).

[3] Matic Jazbec. *Analiza dimenzij kakovosti informacij spletnih strani slovenskih podjetij*. Diplomska naloga. Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2016.

[4] Dejan Petrović. *Spletni pajki*. Diplomska naloga. Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2013.

[5] Marko Pucelj. *Bitcoin*. Diplomska naloga. Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2014.

[6] Raval Siraj. *Decentralized Applications: Harnessing Bitcoin's Blockchain Technology*. O'Reilly Media, 2016.

Spletni viri

[7] *Alexa - Traffic Statistics*. https://www.alexa.com/siteinfo/coinmarketcap.com. Dostopano: 21. 12. 2017.

[8] *Altcoin Definition*. https://www.investopedia.com/terms/a/altcoin.asp. Dostopano: 25. 12. 2017.

[9] Jerry Brito in Andrea Castillo. *Bitcoin: A Primer for Policymakers*. https://www.mercatus.org/system/files/Brito_BitcoinPrimer.pdf. Dostopano: 15. 1. 2018. 2013.

[10] Manca Golmajer in Andrejka Smukavec. *Desezoniranje časovnih vrst, splošna metodološka pojasnila*. http://www.stat.si/dokument/5298/DesezoniranjeCasovnihVrstMPslosna.pdf. Dostopano: 6. 3. 2018. 2017.

[11] *Apache Lucene*. https://lucene.apache.org/. Dostopano: 25. 1. 2018.

[12] Stephen Armstrong. *Move over Bitcoin, the blockchain is only just getting started*. https://www.wired.co.uk/article/unlock-the-blockchain. Dostopano: 3. 1. 2018.

[13] Kariappa Bheemaiah. *Block Chain 2.0: The Renaissance of Money*. https://www.wired.com/insights/2015/01/block-chain-2-0/. Dostopano: 3. 1. 2018.

[14] *Bitbucket Features*. https://www.atlassian.com/software/bitbucket/features. Dostopano: 22. 1. 2018.

[15] *What is Bitcoin.* https://www.bitcoin.com/info/what-is-bitcoin. Dostopano: 25. 12. 2017.

[16] *How do Bitcoin Transactions Work?* https://www.coindesk.com/information/how-do-bitcoin-transactions-work/. Dostopano: 1. 3. 2018.

[17] *Blockchain Definition.* https://www.investopedia.com/terms/b/blockchain.asp. Dostopano: 28. 12. 2017.

[18] *Blockchain Structure.* https://en.bitcoin.it/wiki/File:Blockchain.png. Dostopano: 15. 2. 2018.

[19] *Block Explorer.* https://blockexplorer.com/. Dostopano: 1. 3. 2018.

[20] Michael del Castillo in Bailey Reutzel. *$100 Billion Controversy: XRP's Surge Raises Hard Questions for Ripple.* https://www.coindesk.com/100-billion-controversy-xrps-surge-raises-hard-questions-ripple/. Dostopano: 4. 3. 2018.

[21] *CLR (Common Language Runtime).* https://docs.microsoft.com/en-us/dotnet/standard/clr. Dostopano: 21. 1. 2018.

[22] *Spletna stran CoinMarketCap.* https://coinmarketcap.com. Dostopano: 21. 12. 2017.

[23] *CoinMarketCap API.* https://coinmarketcap.com/api/. Dostopano: 21. 12. 2017.

[24] *CoinMarketCap FAQ.* https://coinmarketcap.com/faq/. Dostopano: 18. 2. 2018.

[25] Damien Cosset. *Blockchain: What is Mining?* https://dev.to/damcosset/blockchain-what-is-mining-2eod. Dostopano: 28. 12. 2017.

[26] *Spletna stran CryptoCompare.* https://www.cryptocompare.com/. Dostopano: 22. 12. 2017.

[27] *CryptoCompare API.* https://www.cryptocompare.com/api/. Dostopano: 22. 12. 2017.

[28] *Cryptographic nonce*. http://searchsecurity.techtarget.com/definition/nonce. Dostopano: 10. 1. 2018.

[29] *Welcome to C# 7.2*. https://blogs.msdn.microsoft.com/dotnet/2017/11/15/welcome-to-c-7-2-and-span/. Dostopano: 22. 1. 2018.

[30] Anthony Cuthbertson. *Bitcoin now accepted by 100,000 merchants worldwide*. http://www.ibtimes.co.uk/bitcoin-now-accepted-by-100000-merchants-worldwide-1486613. Dostopano: 15. 1. 2018.

[31] Emmanuel Darko. *ICO Market Research: The Leading Blockchain Platforms Of 2017*. https://icowatchlist.com/blog/ico-market-research-leading-blockchain-platforms-2017/. Dostopano: 17. 1. 2018.

[32] *What is a Decentralized Application?* https://www.coindesk.com/information/what-is-a-decentralized-application-dapp/. Dostopano: 17. 1. 2018.

[33] *Distributed Ledgers*. https://www.investopedia.com/terms/d/distributed-ledgers.asp. Dostopano: 28. 12. 2017.

[34] *Elastic*. https://www.elastic.co/. Dostopano: 21. 1. 2018.

[35] *Install Elasticsearch*. https://www.elastic.co/guide/en/elasticsearch/reference/current/windows.html. Dostopano: 1. 2. 2018.

[36] *Elasticsearch*. https://www.elastic.co/products/elasticsearch. Dostopano: 23. 1. 2018.

[37] *Elasticsearch API*. https://www.elastic.co/guide/en/elasticsearch/reference/current/docs.html. Dostopano: 6. 3. 2018.

[38] *Ethereum: Blockchain App Platform*. https://www.ethereum.org/. Dostopano: 10. 1. 2018.

[39] *Why is Ethereum different to Bitcoin?* https://www.cryptocompare.com/coins/guides/why-is-ethereum-different-to-bitcoin/. Dostopano: 17. 1. 2018.

[40] *About the Ethereum Foundation*. https://www.ethereum.org/foundation. Dostopano: 11. 1. 2018.

[41] *How Do Ethereum Smart Contracts Work?* https://www.coindesk.com/information/ethereum-smart-contracts-work/. Dostopano: 17. 1. 2018.

[42] *Understanding Ethereum*. https://www.coindesk.com/research/understanding-ethereum-report/. Dostopano: 10. 1. 2018. 2016.

[43] *Ethereum Virtual Machine*. http://ethdocs.org/en/latest/introduction/what-is-ethereum.html. Dostopano: 17. 1. 2018.

[44] *Framework Class Library*. https://msdn.microsoft.com/en-us/library/gg145045(v=vs.110).aspx. Dostopano: 21. 1. 2018.

[45] *What is the Forth programming language?* https://www.forth.com/forth/. Dostopano: 10. 1. 2018.

[46] *How does Bitcoin work?* https://bitcoin.org/en/how-it-works. Dostopano: 10. 1. 2018.

[47] *Knjižnica Html Agility Pack*. http://html-agility-pack.net/. Dostopano: 4. 2. 2018.

[48] Joshua A. Kroll in Ian C. Davey in Edward W. Felten. *The Economics of Bitcoin Mining, or Bitcoin in the Presence of Adversaries*. http://www.econinfosec.org/archive/weis2013/papers/KrollDaveyFeltenWEIS2013.pdf. Dostopano: 15. 1. 2018. 2013.

[49] Alex Chumbley in Karleigh Moore. *Merkle Tree*. https://brilliant.org/wiki/merkle-tree/. Dostopano: 3. 1. 2018.

[50] Preethi Kasireddy. *Blockchains don't scale. Not today, at least. But there's hope.* https://hackernoon.com/blockchains-dont-scale-not-today-at-least-but-there-s-hope-2cb43946551a. Dostopano: 10. 3. 2018. 2017.

[51] *Kibana*. https://www.elastic.co/products/kibana. Dostopano: 1. 2. 2018.

[52] *Installing Kibana*. https://www.elastic.co/guide/en/kibana/current/targz.html. Dostopano: 1. 2. 2018.

[53] David Knott. *Vyper*. https://github.com/ethereum/vyper. Dostopano: 17. 1. 2018.

[54] Janus Kopfstein. *The Mission To Decentralize the Internet*. https://www.newyorker.com/tech/elements/the-mission-to-decentralize-the-internet. Dostopano: 15. 1. 2018.

[55] *Let's Encrypt*. https://letsencrypt.org/. Dostopano: 28. 2. 2018.

[56] Joe Liebkind. *How Did Ethereum's Price Perform In 2017?* https://www.investopedia.com/news/how-did-ethereums-price-perform-2017. Dostopano: 17. 1. 2018.

[57] *Spletna stran Litecoin*. https://litecoin.com. Dostopano: 10. 1. 2018.

[58] *LLL Introduction*. https://lll-docs.readthedocs.io/en/latest/lll_introduction.html. Dostopano: 26. 2. 2018.

[59] Jon Matonis. *How Cryptocurrencies Could Upend Banks' Monetary Role*. https://www.americanbanker.com/opinion/how-cryptocurrencies-could-upend-banks-monetary-role. Dostopano: 28. 12. 2017.

[60] *Mutan*. https://github.com/obscuren/mutan. Dostopano: 26. 2. 2018.

[61] *Knjižnica NEST*. https://github.com/elastic/elasticsearch-net. Dostopano: 5. 2. 2018.

[62] *Microsoft .NET*. https://www.microsoft.com/net/. Dostopano: 22. 1. 2018.

[63] David Parkins. *The great chain of being sure about things - Blockchains*. https://www.economist.com/news/briefing/21677228-technology-behind-bitcoin-lets-people-who-do-not-know-or-trust-each-other-build-dependable. Dostopano: 28. 12. 2017.

[64] *Portfolio - Manage and track your cryptocurrency.* https://www.cryptocompare.com/portfolio/. Dostopano: 22. 12. 2017.

[65] *Public key cryptography.* https://www.ibm.com/support/knowledgecenter/en/SSB23S_1.1.0.13/gtps7/s7pkey.html. Dostopano: 15. 1. 2018.

[66] Jamie Redman. *Dragonchain, an Interoperable Ledger.* https://news.bitcoin.com/disney-dragonchain-interoperable-ledger/. Dostopano: 3. 1. 2018.

[67] Ameer Rosic. *What is Cryptocurrency: Everything You Need To Know.* https://blockgeeks.com/guides/what-is-cryptocurrency/. Dostopano: 25. 12. 2017.

[68] *What is Scrypt?* https://www.cryptocompare.com/coins/guides/what-is-scrypt/. Dostopano: 10. 1. 2018.

[69] *Serpent.* https://github.com/ethereum/wiki/wiki/Serpent. Dostopano: 26. 2. 2018.

[70] *All about SHA1, SHA2 and SHA256 hash algorithms.* https://www.tbs-certificates.co.uk/FAQ/en/sha256.html. Dostopano: 10. 1. 2018.

[71] *Solidity.* https://solidity.readthedocs.io/en/develop/. Dostopano: 26. 2. 2018.

[72] *Sourcetree.* https://www.sourcetreeapp.com/. Dostopano: 23. 1. 2018.

[73] Kyle Torpey. *Will Bitcoin's Lightning Network Kill Off Altcoins Focused On Cheap Transactions?* https://www.forbes.com/sites/ktorpey/2017/12/28/will-bitcoins-lightning-network-kill-off-altcoins-focused-on-cheap-transactions. Dostopano: 17. 1. 2018.

[74] Usman W. Chohan. *Cryptocurrencies: A Brief Thematic Review*. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3024330. Dostopano: 11. 1. 2018.

[75] *Visual Studio IDE*. https://www.visualstudio.com/vs/. Dostopano: 21. 1. 2018.

[76] Dr. Gavin Wood. *Ethereum: A Secure Decentralised Generalised Transaction Ledger*. http://yellowpaper.io/. Dostopano: 11. 1. 2018.

[77] Bryan Yurcan. *How Blockchain Fits into the Future of Digital Identity*. https://www.americanbanker.com/news/how-blockchain-fits-into-the-future-of-digital-identity. Dostopano: 3. 1. 2018.