D3.js platforma za vizualizaciju poslovnih podataka

Sudec, Alen

Master's thesis / Diplomski rad

2021

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Economics in Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Ekonomski fakultet u Osijeku**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:145:492990

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-04**

Repository / Repozitorij:

EFOS REPOSITORY - Repository of the Faculty of Economics in Osijek

Sveučilište Josipa Jurja Strossmayera u Osijeku

Ekonomski fakultet u Osijeku

Diplomski studij (Poslovna informatika)

Alen Sudec

D3.js platforma za vizualizaciju poslovnih podataka

Diplomski rad

Osijek, 2021.

Sveučilište Josipa Jurja Strossmayera u Osijeku

Ekonomski fakultet u Osijeku

Diplomski studij (Poslovna informatika)

Alen Sudec

D3.js platforma za vizualizaciju poslovnih podataka

Diplomski rad

Kolegij: Sustavi poslovne inteligencije

JMBAG: 0165062215

Email: alensudec@hotmail.com

Mentor: doc. dr. sc. Slobodan Jelić

Osijek, 2021.

Josip Juraj Strossmayer University of Osijek

Faculty of Economics in Osijek

Graduate Study (Business informatics)

Alen Sudec

D3.js web platform for visualization of business data

Graduate paper

Osijek, 2021.

IZJAVA

O AKADEMSKOJ ČESTITOSTI, PRAVU PRIJENOSA INTELEKTUALNOG VLASNIŠTVA, SUGLASNOSTI ZA OBJAVU U INSTITUCIJSKIM REPOZITORIJIMA I ISTOVJETNOSTI DIGITALNE I TISKANE VERZIJE RADA

1. Kojom izjavljujem i svojim potpisom potvrđujem da je DIPLOMSKI (navesti vrstu rada: završni / diplomski / specijalistički / doktorski) rad isključivo rezultat osobnoga rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu. Potvrđujem poštivanje nepovredivosti autorstva te točno citiranje radova drugih autora i referiranje na njih.
2. Kojom izjavljujem da je Ekonomski fakultet u Osijeku, bez naknade u vremenski i teritorijalno neograničenom opsegu, nositelj svih prava intelektualnoga vlasništva u odnosu na navedeni rad pod licencom *Creative Commons Imenovanje – Nekomercijalno – Dijeli pod istim uvjetima 3.0 Hrvatska.*
3. Kojom izjavljujem da sam suglasan/suglasna da se trajno pohrani i objavi moj rad u institucijskom digitalnom repozitoriju Ekonomskoga fakulteta u Osijeku, repozitoriju Sveučilišta Josipa Jurja Strossmayera u Osijeku te javno dostupnom repozitoriju Nacionalne i sveučilišne knjižnice u Zagrebu (u skladu s odredbama Zakona o znanstvenoj djelatnosti i visokom obrazovanju, NN br. 123/03, 198/03, 105/04, 174/04, 02/07, 46/07, 45/09, 63/11, 94/13, 139/13, 101/14, 60/15).
4. izjavljujem da sam autor/autorica predanog rada i da je sadržaj predane elektroničke datoteke u potpunosti istovjetan sa dovršenom tiskanom verzijom rada predanom u svrhu obrane istog.

Ime i prezime studenta/studentice: ALEN SUDEC

JMBAG: 0165062215

OIB: 90365872818

e-mail za kontakt: ALENSUDEC@HOTMAIL.COM

Naziv studija: POSLOVNA INFORMATIKA

Naslov rada: D3.js PLATFORMA ZA VIZUALIZACIJU POSLOVNIH PODATAKA

Mentor/mentorica diplomskog rada: SLOBODAN SELIĆ

U Osijeku, 09.09.2021. godine

Potpis Sudec

D3.js platforma za vizualizaciju poslovnih podataka

SAŽETAK

Rad se bavi izradom internetske aplikacije koja se bavi vizualizacijom poslovnih podataka kroz d3.js biblioteku. Internetska aplikacija se izrađuje korištenjem D3.js biblioteke koja služi za manipuliranje dokumenata na temelju podataka. D3 oživljava skupove podataka pomoću HTML-a, SVG-a i CSS-a kroz koje kombinira moćne komponente vizualizacije i pristup manipulaciji DOM-om temeljen na podacima. Rad će prikazati korake u izradi internetske aplikacije koja je izrađena koristeći Visual Studio Code alat. Aplikacija će imati mogućnost podizanja skupa podataka u formatu .csv datoteke, nakon čega će se korisniku prikazati sve moguće varijable povučene iz skupa podataka. Korisnik odabire određenu varijable i grafikon koji želi, nakon čega se ispred korisnika crta grafikon pomoću d3.js biblioteke. Izrada je podijeljena u dva dijela, frontend i backend. Kako je React.js jedan od najpopularnijih tehnologija za korisničko sučelje, zbog svoje optimiziranosti i brzine dohvaćanja podataka kroz React stanja, ova tehnologija će se koristiti u poglavlju frontend. Što se tiče backend dijela, koristiti će se Node.js tehnologija za poslužitelj, i PostgreSQL kao baza podataka. Uz izradu aplikacije rad će prikazati skice ciljane aplikacije kako bi se prikazao proces planiranja i izrade korisničkog sučelja, izrade komponente za React.js tehnologiju kroz koje je napravljeno korisničko sučelje, izradu poslužitelja i njegove krajnje točke te korištenje baze podataka. Na kraju rada se prolazi kroz prednosti i nedostatke d3.js biblioteke, te njezin doprinos u internetskom razvoju.

Ključne riječi: Internetska aplikacija, d3.js biblioteka, JavaScript, React.js, Node.js, PostgreSQL

D3.js web platform for visualization of business data

ABSTRACT

The paper deals with the development of an Internet application that deals with the visualization of business dana through the D3.js library. The web application is created with D3.js library which is used to manipulate documents based on dana. D3 brings datasets to life using HMTL, SVG and CSS through which it combines powerful visualization components and a dana-based approach to DOM manipulation. The paper will show the steps of creating an Internet application created using Visual Studio Code tool. The application will have the ability to load a dataset in a .csv format, after which the user will be shown all possible variables extracted from the dataset. The user selects a specific variable and the graph he wants, after which a graph is drawn in front of the user using the d3.js library. The production is divided into two parts, frontend and backend. As React.js is one of the most popular user interface technologies, due to its optimization and speed of retrieving dana through React states, this technology will be use din the frontend chapter. As for the backend part, Node.js technology will be used, and PostgreSQL as the database. In addition to creating the application, the paper will present mockups of the target application to show the process of planning and creating the user interface, creating a component for React.js technology through which the user interface is made, creating the server and its endpoints, and using the database. At the end of the paper, we go through the advantages and disadvantages of the d3.js library, and its contribution to Internet development.

Keywords: Internet application, D3.js library, JavaScript, React.js, Node.js, PostgreSQL

Sadržaj

1. Uvod

U današnje vrijeme, Internet se koristi u svim sferama života. Od poslovanja tvrtki do privatnih razloga poput društvenih mreža. Svi bitni i ostali podaci i informacije se nalaze „online“ kojima se može pristupiti u bilo koje vrijeme. Ti podaci i informacije se mogu prikazivati i koristiti na različite načine kroz internetske stranice, internetske aplikacije i različite mobilne aplikacije. Internetskim stranicama i aplikacijama može se pristupiti kroz internetske preglednike poput Internet Explorera, Chrome, Opera i ostali, dok za mobilne aplikacije nam je potreban mobilni uređaj. Internetska mjesta (http://primjer.hr) se sastoje od više internetskih stranica (http://primjer.hr/kontakt) koja se uglavnom koriste za prikaz većeg broja podataka koja ne stanu na jednu stranicu. Internetska aplikacija je vrsta internetske stranice čija svrha nije samo prikaz podataka, nego i manipulacija podataka kroz učitavanje datoteka. Razvoj internetskih aplikacija je proces stvaranja ili razvijanja sučelja za upotrebu preko internetskih preglednika na računalima i ako je aplikacija osjetljiva na različite veličine ekrana, može se koristiti i na mobilnim uređajima. Internetske aplikacije se koriste zbog svojih funkcionalnosti, što znači da korisnik može preko internetskog preglednika odraditi sve što je potrebno bez skidanja različitih programa na svoje računalo ili mobilni uređaj.

Ovaj rad se bavi problematikom izrade internetske aplikacije koja će kroz 2.3. D3.js biblioteka biblioteku izraditi različite grafikone. Predmet i cilj diplomskog rada je istražiti 2.2. JavaScript programski jezik, isplanirati i izraditi internetsku aplikaciju vezanu za stvaranje grafikona kroz 2.3. D3.js biblioteka 2.2. JavaScript programski jezik biblioteku.

Drugo poglavlje uključuje teorijsku podlogu o 2.2. JavaScript programski jeziku i 2.3. D3.js biblioteka biblioteci.

Treće poglavlje prikazuje tehnologije i programe koji će se implementirati za izradu internetske aplikacije poput 3.1.1. Visual Studio Code, 3.2.3. React.js razvojni okvir i ostali.

Četvrto poglavlje se bazira na koracima u izradi internetske aplikacije, istraživanjima vezanih za navedene korake i prikaz finalnog programskog rješenja.

2. Teorijska podloga i prethodna istraživanja

U okviru rada „D3.js platforma za vizualizaciju poslovnih podataka“ provedena su teorijska istraživanja o razvoju internetskih aplikacija, o 2.2. JavaScript programski jezik i 2.3. D3.js biblioteka. Rezultati tih istraživanja praktično se primjenjuju i prikazuju u obliku internetske aplikacije izrađene kroz 2.2. JavaScript programski jezik, 2.3. D3.js biblioteka i različitih 2.2. JavaScript programski jezik razvojnih okvira.

2.1. Općenito o razvoju internetskih aplikacija

Internetske aplikacije važne su za razvoj Interneta. Poboljšavaju kvalitetu rada preglednika i cijeli Internet može savršeno funkcionirati. Internetska aplikacija je softverska aplikacija koju klijent pokreće u internetskom pregledniku. Glavna funkcija preglednika je prikazati podatke primljene sa poslužitelja i poslati podatke korisnika natrag. Glavna prednost ovog pristupa je činjenica da klijenti ne ovise o operativnom sustavu korisnika, stoga se internetske aplikacije mogu koristiti kroz više platformi. Zbog ove univerzalne značajke, internetske aplikacije postale su vrlo popularne 1990-ih i 2000-ih godina. Programeri ne moraju pripremiti različite verzije iste aplikacije za Microsoft Windows, Mac OS, Linux, itd. Aplikacija se stvara samo jednom za bilo koju platformu i može raditi na bilo kojem operativnom sustavu. Međutim, različita praktična realizacija 3.2.1. HTML, 3.2.2. CSS, DOM i drugih sučelja u preglednicima može uzrokovati probleme tijekom razvoja internetskih aplikacija i njihove daljnje podrške. Štoviše, internetska aplikacija može raditi pogrešno zbog korisnikove mogućnosti da promijeni postavke preglednika na način koji želi. (Devsaran, 2021)

Povijest razvoja internetskih aplikacija prilično je komplicirana. Postoje mnoge tehnologije (Flash, Java, Silverlight, itd.) koje čine rad na Internetu što je moguće lakšim. Korisnik može slušati audio, gledati video zapise i crtati na ekranu uz pomoć klika miša. Interaktivnost Interneta je postala ogromna, a u budućnosti će biti još učinkovitija i raznolika. Ajax je jedan od najboljih primjera skupa tehnologija koje poboljšavaju razinu interaktivnosti između korisnika i stroja. (Devsaran, 2021)

2.2. JavaScript programski jezik

JavaScript (često skraćeno na JS) je lagani, interpretirani, objektno orijentiran jezik s prvoklasnim funkcijama, a najpoznatiji je kao skriptni jezik za internetske stranice, ali se koristi i u mnogim okruženjima koja nisu preglednici. To je prototipni, multi paradigmatski skriptni jezik koji je dinamičan i podržava objektno orijentirane, imperativne i funkcionalne stilove programiranja. (Mozzila, 2021)

JavaScript radi na klijentskoj strani interneta, koji se može koristiti za programiranje ponašanja internetskih stranica pri pojavi događaja. JavaScript je jednostavan za učenje i uz to moćni skriptni jezik koji se naširoko koristi za kontrolu ponašanja internetskih stranica. (Mozzila, 2021)

Suprotno popularnom mišljenju, JavaScript nije „interpretirana Java“. Ukratko, JavaScript je dinamički skriptni jezik koji podržava konstrukciju objekata temeljenih na prototipovima. Osnovna sintaksa namjerno je slična Javi i C++ jeziku kako bi se smanjio broj novih koncepata potrebnih za učenje jezika. Jezične konstrukcije, kao što su naredbe, „for“ i „while“ petlje, „switch“ i „try catch“ blokovi funkcioniraju isto kao u ovim jezicima. (Mozzila, 2021)

JavaScript može funkcionirati i kao proceduralni i kao objektno orijentirani jezik. Objekti se stvaraju programski u JavaScriptu, dodavanjem metoda i svojstava inače praznim objektima za vrijeme izvođenja, za razliku od sintaktičkih definicija klasa uobičajenih u prevedenim jezicima kao C++ i Jave. Nakon što je objekt izgrađen može se koristiti kao nacrt ili prototip za stvaranje sličnih objekata. (Mozzila, 2021)

Dinamičke mogućnosti JavaScripta uključuju izgradnju objekta tijekom izvođenja, popise varijabli parametra, varijable funkcija, kreiranje dinamičke skripte, introspekciju objekata i oporavak izvornog koda. (Mozzila, 2021)

Slika 1 - JavaScript logo
(medium.com, 2021.)

2.3. D3.js biblioteka

D3.js je JavaScript biblioteka za manipuliranje dokumenata na temelju podataka. D3 pomaže oživiti podatke pomoću 3.2.1. HTML, SVG i 3.2.2. CSS elementa. Naglasak D3 na internetskim standardima daje pune mogućnosti suvremenih preglednika bez vezivanja za vlasničke razvojne okvire, kombinirajući moćne komponente vizualizacije i pristup manipulaciji DOM-om temeljen na podacima. D3 omogućuje vezivanje proizvoljnih podataka na DOM (eng. Document Object Model), a zatim primjenu transformacija na temelju podataka u dokumentu. Na primjer, D3 se može koristiti za generiranje 3.2.1. HTML tablica iz niza brojeva, ili korištenje istih podataka za stvaranje interaktivnog stupčastog grafikona sa glatkim prijelazima i interakcijom kao SVG element. D3 nije monolitni razvojni okvir koji nastoji pružiti svaku zamislivu značajku. Umjesto toga, D3 rješava srž problema: učinkovita manipulacija dokumentima na temelju podataka. Time se izbjegava vlasničko predstavljanje i pruža izuzetna fleksibilnost, otkrivajući sve mogućnosti internetskog standarda kao što su 3.2.1. HTML, SVG i 3.2.2. CSS. D3 je izuzetno brz, podržava velike skupove podataka i dinamičko ponašanje za interakciju i animaciju. Funkcionalni stil D3 omogućuje ponovnu upotrebu koda kroz raznoliku zbirku službenih modula razvijenih u zajednici. (Bostock, 2021)

Slika 2 - D3.js logo
(pinterest.com, 2021.)

3. Metodologija rada

U ovom poglavlju se opisuju korišteni alati u izradi internetske aplikacije, opisuju se tehnologije koje su korištene u razvoju aplikacije, te njezine specifikacije za izradu.

3.1. Korišteni alati u razvoju aplikacije

3.1.1. Visual Studio Code

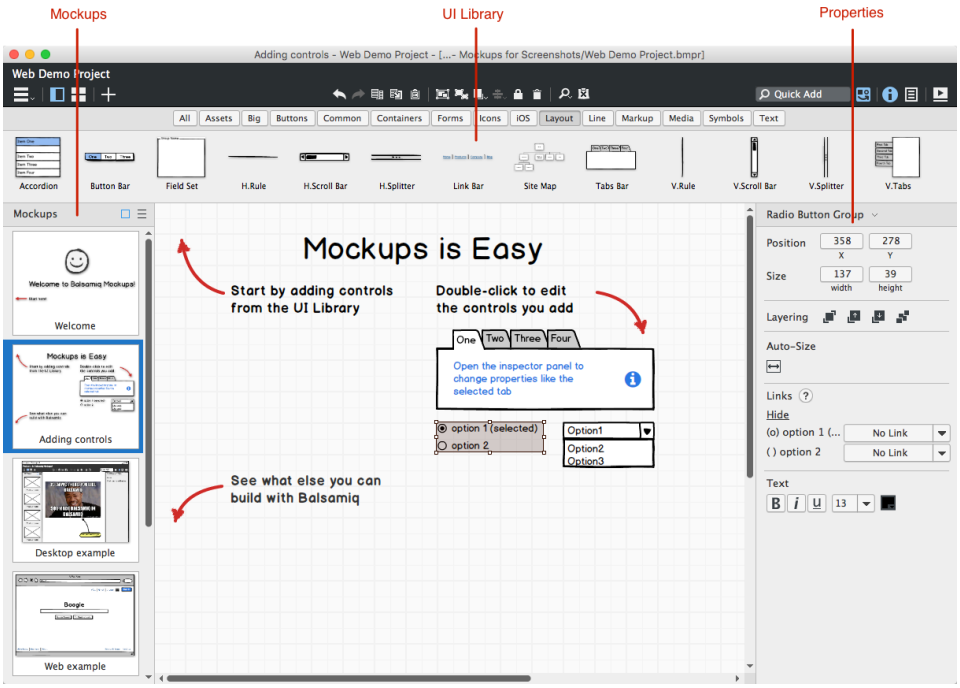
VSCode uređivač je izvornog koda koji je izradio Microsoft za Windows, Linux i MacOs. Značajke uključuju podršku za ispravljanje pogrešaka, isticanje sintakse, inteligentno dovršavanje koda, isječke, preradu koda i ugrađeni Git. Korisnici mogu promijeniti temu, prečace tipaka, postavke i instalirati proširenja koja dodaju dodatne funkcije. Uređivač u kojem se mogu koristiti razni programski jezici, uključujući Javu, 2.2. JavaScript programski jezik, Go, 3.2.4. Node.js, Python i C++. Temelji se na Electron razvojnom okviru, koji se koristi za razvoj Node.js internetskih aplikacija koje se izvode na Blink razvojnom okviru. Koristi istu komponentu uređivača (kodnog naziva „Monaco“) koja se koristi u Azure DevOps-u. Umjesto projektnog sustava, korisnicima omogućuje otvaranje jednog ili više direktorija, koji se zatim mogu spremiti u radne prostore za buduću ponovnu uporabu. To mu omogućuje da radi kao uređivač koda za bilo koji programski jezik. Podržava brojne programske jezike i skup značajki koji se razlikuju od jezika do jezika. Neželjene datoteke i mape mogu se isključiti iz stabla projekta putem postavki. Mnoge značajke nisu izložene putem izbornika ili korisničkog sučelja, ali im se može pristupiti putem naredbene palete. Visual Studio Code može se proširiti putem proširenja, dostupnih kroz središnji sustav. To uključuje dodatke uređivaču i jezičnu podršku. Značajna značajka je mogućnost stvaranja proširenja koja dodaju podršku za nove jezike, teme i programe za ispravljanje pogrešaka, izvode statičku analizu koda i dodaju podmetače koda pomoću protokola Language Server. (Kanjilal, 2021)

Visual Studio Code uključuje više proširenja za FTP (eng. File Transfer Protocol), dopuštajući softver da se koristi kao besplatna alternativa za internetski razvoj. Kod se može sinkronizirati pomoću uređivača i poslužitelja, bez preuzimanja dodatnog softvera. Omogućuje korisnicima da postave kodnu stranicu u koju je spremljen aktivni dokument, znak novog retka i programski jezik aktivnog dokumenta. To omogućuje korištenje na bilo kojoj platformi, na bilo kojem jeziku i za bilo koji programski jezik. (Microsoft, 2021)

Slika 3 – VS Code logo
(tmssoftware.com, 2021.)

3.1.2. Balsamiq Wireframe

Balsamiq Wireframe je brzi alat za stvaranje skica niske vrijednosti koji reproducira iskustvo skiciranja na bilježnici ili bijeloj ploči, ali pomoću računala. Prisiljava korisnike da se usredotoče na strukturu i sadržaj, izbjegavajući duge rasprave o bojama i detaljima koji bi trebali uslijediti kasnije u procesu. (Balsamiq Studios, 2021)



Slika 4 - Balsamiq Wireframe sučelje
(media.balsamiq.com, 2021.)

3.1.3. Adobe Photoshop

Uređivač rasterske grafike koji je razvio i objavio Adobe inc. za Windows i MacOs. Izvorno su ga stvorili Thomas i John Knoll 1988. godine. Od tada je softver postao industrijski standard, ne samo u uređivanju rasterske grafike, već i u digitalnoj umjetnosti u cjelini. Photoshop može uređivati i sastavljati rasterske slike u više slojeva i podržava maske, alfa kompoziciju i nekoliko modela boja, uključujući RGB, CYMK, CIELAB, mrljaste boje i dvostruke boje. Photoshop koristi vlastite PSD i PSB formate datoteka kako bi podržao te značajke. Uz rastersku grafiku, ovaj softver ima ograničene mogućnosti za uređivanje ili vizualizacija tekstualne ili vektorske grafike (posebice putem isječka), kao i 3D grafike i videa. Njegov skup značajki može se proširiti dodatcima; programi razvijeni i distribuirani neovisno o Photoshopu koji rade unutar njega i nude nove ili poboljšane značajke. Uz Photoshop, Adobe razvija i objavljuje Photoshop Elements, Photoshop Lightroom, Photoshop Express, Photoshop Fix, Photoshop Sketch i Photoshop Mix. (Manovich, 2021)

Slika 5 - Photoshop logo
(aspgo.com, 2021.)

3.2. Korištene tehnologije u izradi aplikacije

3.2.1. HTML

HTML (HyperText Markup Language) je najosnovniji element Interneta. Definira značenje i strukturu internetskog sadržaja. Druge tehnologije osim HTML-a općenito se koriste za opisivanje izgleda/prezentacije internetske stranice (3.2.2. CSS) ili funkcionalnosti/ponašanja (2.2. JavaScript programski jezik). „Hypertext“ se odnosi na veze koje povezuju internetske stranice, bilo unutar jedne internetske stranice ili između internetskih stranica. Prijenosom sadržaja na Internet i povezivanjem sa stranicama koje su stvorili drugi ljudi, korisnici postaju aktivni sudionici na Internetu. HTML koristi „markup“ za označavanje teksta, slika i drugog sadržaja za prikaz u internetskom pregledniku. HTML „markup“ uključuje posebne elemente kao što su *<head>*, *<title>*, *<body>*, *<header>*, *<footer>*, *<article>*, *<section>*, *<p>*, * *, **, *< img>*, *<aside>*, *<audio>*, *<canvas>*, *<datalist>*, *<details>*, *<embed>*, *<nav>*, *<output>*, *<progress>*, *<video>*, *<ul>*, *<ol>* , *<li>* i mnogi drugi. HTML element odvojen je od drugog teksta u dokumentu pomoću oznaka, koje se sastoje od naziva elementa okruženog s „*<* “ i „*>* “. Naziv elementa unutar oznake ne razlikuje velika i mala slova. To jest, može se pisati velikim, malim ili mješavinom velikih i malih slova. Na primjer, oznaka *<title>* može se napisati kao *<Title>*, *<TITLE>* ili na bilo koji drugi način. (Mozzila, 2021)

Slika 6 - HTML logo
(oxfordwebstudio.com, 2021.)

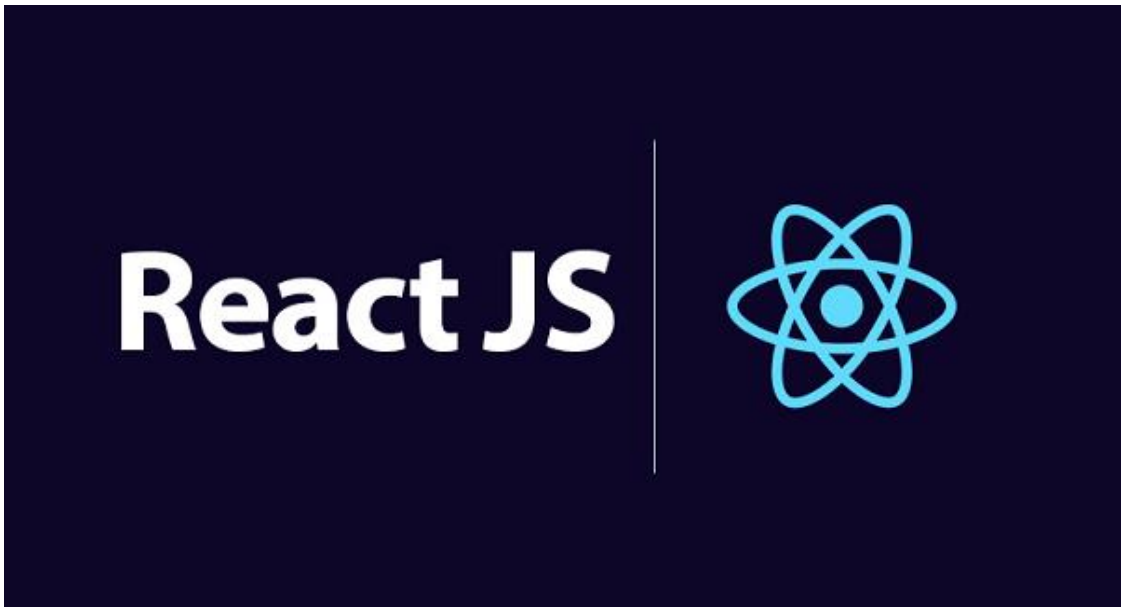
3.2.2. CSS

CSS (Cascading Style Sheets) je jezik stilske tablice koji se koristi za opisivanje prezentacije dokumenta napisanom u 3.2.1. HTML-u ili XML-u (uključujući XML dijalekte kao što su SVG, MathML, ili XHTML). CSS opisuje kako se elementi trebaju prikazati na ekranu, papiru, u govoru ili na drugim medijima. CSS je jedan od osnovnih jezika otvorenog Interneta i standardiziran je u svim internetskim preglednicima prema W3C specifikacijama. U prošlosti, razvoj različitih dijelova CSS specifikacije je bio sinkroniziran, što je dopuštalo verziju najnovijih preporuka. (CSS1, CSS2.1, CSS3). Od CSS3, opseg specifikacija se značajno povećao, a napredak na različitim CSS modulima počeo se toliko razlikovati, da je postalo učinkovitije razvijati i objavljivati preporuke zasebno po modulu. Umjesto izmjena CSS specifikacije, W3C sada povremeno objavljuje verziju najnovijeg stabilnog stanja CSS specifikacije. (Mozzila, 2021)

Slika 7 - CSS logo
(w3docs.com, 2021.)

3.2.3. React.js

React.js/React je otvoreni frontend razvojni okvir koji se temelji na 2.2. JavaScript programski jezik-u, razvijen od tvrtke koja je razvila Facebook, a najpoznatiji je po svojoj virtualnoj DOM značajci. Uz React preporučuje se Express.js/Express kao pozadinska usluga. To je backend razvojski okvir za Node.js, a također je i temeljna biblioteka za mnoge druge 3.2.4. Node.js internetske razvojne okvire. ReactJS jedna je od najpopularnijih JavaScript biblioteka i prvenstveno se koristi za stvaranje korisničkog sučelja. Predstavio ga je Facebook i održava se uz pomoć mnogih pojedinačnih programera i tvrtki. Glavni razlog za korištenje React-a za projekte je njegova sposobnost dohvaćanja podataka koji se brzo mijenjaju, a koje je također potrebno evidentirati. To je tehnologija otvorenog koda koja se može koristiti sa strukturom MVC (eng. Model-View-Controller) s Reactom koji doprinosi sloju prikaza. Jedna od glavnih prednosti React-a je ta što se može koristiti za izradu komponenti koje se sastoje uglavnom od prilagođenih HTML elemenata. Ove se komponente mogu ponovno koristiti za učinkovit razvoj korisničkih sučelja. React također može organizirati način pohranjivanja i rukovanja podacima pomoću stanja (eng. React State). React je također najbolje rješenje za probleme kao što su niske performanse i spora interakcija korisnika uzrokovana manipulacijom DOM-a. React rješava te probleme pomoću virtualnog DOM-a u koji se spremaju sve promjene, a razlika dvaju stanja korisničkog sučelja rješava se analizom algoritma. (Naik & Wickramasinghe, 2021)

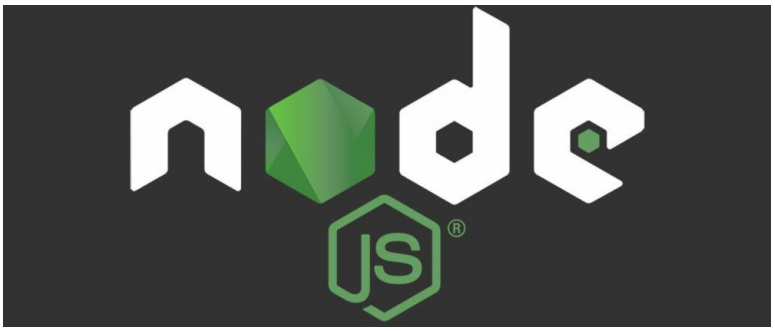


Slika 8 - ReactJS logo

(blog.knoldus.com, 2021.)

3.2.4. Node.js

Node.js je JavaScript okruženje otvorenog koda, koje je izvorno razvio Ryan Dahl 2009. godine. Iako je početno izdanje podržavalo samo Linux, ono se razvilo kao istinsko više-platformsko 2.2. JavaScript programski jezik okruženje za razvoj širokog spektra aplikacija. Radno okruženje interpretira 2.2. JavaScript programski jezik pomoću Google-ovog V8 JavaScript algoritma. Osnovna funkcionalnost Node.js implementirana je unutar 2.2. JavaScript programski jezik biblioteke. Node.js omogućuje stvaranje internetskog poslužitelja i mrežnih alata pomoću 2.2. JavaScript programski jezik-a i zbirke modula koji obrađuju različite osnove funkcije. Dostupni su moduli za I/O datotečnog sustava, umrežavanje (poput HTTP-a, DNS-a, TCP-a, TLS/SSL-a ili UDP-a), binarne podatke (među spremnici), funkcije kriptografije, tokove podataka i druge osnove funkcije. Moduli Node.js-a koriste API (eng. Application Programming Interface) dizajniran za smanjenje složenosti pisanja serverskih aplikacija. Node.js se prvenstveno koristi za izradu mrežnih aplikacija koristeći komunikaciju u stvarnom vremenu koja se zahtjeva preko krajnjih točaka (eng. End-point). Node.js je odličan u aplikacijama upravljanim I/O (eng. Input/Output) operacijama. Najveća razlika između Node.js i PHP-a je ta što se većina funkcija u PHP-u blokira do završetka (tj. naredbe se izvršavaju tek nakon što su prethodne naredbe završene), funkcije u Node.js-u su dizajnirane tako da ne blokiraju komunikaciju (naredbe se izvršavaju paralelno i koriste povratne pozive za signalizaciju dovršetka ili neuspjeha zadatka). Node.js donosi programiranje usmjereno na događaje na internetske poslužitelje, omogućavajući razvoj brzih internetskih poslužitelja u 2.2. JavaScript programski jezik-u. Programeri mogu stvoriti visoko skalabilne poslužitelje koristeći pojednostavljeni model programiranja na temelju događaja koji koristi povratne pozive za signalizaciju dovršetka zadatka. Node.js je stvoren jer je istovremenost teška u mnogim programskim jezicima na strani poslužitelja i često dovodi do loših performansi. Node.js povezuje jednostavnost skriptnog jezika sa snagom mrežnog programiranja. Node.js se često koristi kao dio MEAN Stack-a koji se sastoji od MongoDB, Express.js, Angular i Node.js, za izradu dinamičkih internetskih stranica i aplikacija. (training.com, 2021)



Slika 9 - NodeJS logo
(dev.to, 2021.)

3.2.5. PostgreSQL

PostgreSQL je moćan, objektno-relacijski sustav baze podataka otvorenog koda koji koristi i proširuje SQL (eng. Structured Query Language) jezik u kombinaciji s mnogim značajkama koje sigurno pohranjuju i skaliraju najkompliciranija radna opterećenja podataka. Podrijetlo PostgreSQL-a datira od 1986. godine u sklopu projekta POSTGRES na Kalifornijskom sveučilištu u Berkelyu i ima više od 30 godina aktivnog razvoja na jezgri platforme. PostgreSQL zaslužio je snažnu reputaciju svojom dokazanom arhitekturom, pouzdanošću, integritetom podataka, robusnim skupom značajki, proširivošću i predanošću zajednice otvorenog koda koja stoji iza softvera kako bi dosljedno isporučivao učinkovita i inovativna rješenja. PostgreSQL radi na svim većim operativnim sustavima, od 2001. usklađen je s ACID-om i ima moćne dodatke, poput popularnog proširenja geoprostorne baze podataka PostGIS. Nije čudno što je PostgreSQL postao relacijska baza podataka otvorenog koda za mnoge ljude i organizacije. (PostgreSQL Tutorial, 2021)

Slika 10 - PostgreSQL logo
(agix.com, 2021.)

3.3. Specifikacije za izradu internetske aplikacije

3.3.1. Svrha aplikacije

Svrha internetske aplikacije je omogućiti korisniku podizanje .csv datoteke sa određenim varijablama na poslužitelj, nakon podizanja prikazati sve moguće grafikone, i sve moguće varijable iz već navedene .csv datoteke. Nakon odabira, 3.2.3. React.js šalje odabrane varijable i grafikone u 3.2.5. PostgreSQL bazu podataka. I pri završetku korisniku se prikazuje njegov grafikon, za kojeg može zatražiti *„iframe“* link u kojem zapravo postoji grafikon koji vodi na 3.2.4. Node.js poslužitelj.

3.3.2. Korisnici aplikacije

Korisnici aplikacije mogu biti sve osobe koje imaju potrebu izrade grafikona, ali ciljana grupa korisnika je ona koja želi prikazati određene financijske podatke, i želi koristiti stvoreni grafikon na svojim internetskim stranicama.

3.3.3. Funkcijski zahtjevi

Aplikacija će omogućiti podizanje .csv datoteke na poslužitelj, nakon čega se prikazuje prozor u kojem korisnik odabire varijable i vrstu grafikona. Nakon odabira stvara se i prikazuje grafikon.

4. Opis istraživanja i rezultati istraživanja

Cilj istraživanja je bio razviti internetsku aplikaciju koristeći 2.3. D3.js biblioteka biblioteku, koja je usko vezana za „frontend“ tehnologije. Ona je izrađena u kombinaciji sa alatima koji su navedeni u poglavlju „ 3.1. Korišteni alati u razvoju aplikacije“ i tehnologijama koji su navedeni u poglavlju „3.2. Korištene tehnologije u izradi aplikacije“.

4.1. Idejno rješenje

Internetska aplikacija u idejnom rješenju trebala bi biti u mogućnosti olakšati korisnicima stvaranje više različitih grafikona preko .csv dokumenata kroz 2.3. D3.js biblioteka biblioteku. Dizajn aplikacije treba biti ugodan oku i treba pratiti svjetske standarde kako bi mogao konkurirati ostalim sličnim internetskim aplikacijama. Uz dizajn, funkcionalnost aplikacije treba pratiti konkurente u performansama i korisnošću i inovativnost i poboljšanje u odnosu na ostale. Uz korištenje 3.2.3. React.js i 3.2.4. Node.js tehnologije, brzina aplikacije trebala bi biti jako malo ovisna o internetskoj brzini uređaja s kojeg je pristupljeno. Kako je danas korištenje Interneta na velikoj snazi kroz sve uređaje, internetska aplikacija treba biti osjetilna na različite veličine ekrana kroz većinu uređaja. Na kraju, internetska aplikacija bi trebala biti lagano održiva i skalabilna, ako se u skorije vrijeme nađu greške u kodu, ili ako bude postojala potreba za promjenom dizajna.

4.2. Mockup Wireframe skica

Pomoću alata 3.1.2. Balsamiq Wireframe stvara se skica aplikacije u kojoj se oslikava funkcionalnost aplikacije i njezin dizajn. Skica se koristi u svim fazama izrade aplikacije a najbitnija je u zadnjem dijelu, tj. uređivanje izgleda.

4.2.1. Dizajniranje skice pomoću alata Balsamiq Wireframe

Internet aplikacija biti će izrađena u stilu jedne stranice, gdje će sav sadržaj biti prikazan a nakon klika na određeni gumb stvara se prozor u kojem korisnik bira opcije za grafikon. Nakon odabira u prozoru se stvara slika grafikona koja će biti interaktivna.

Navigacijska traka

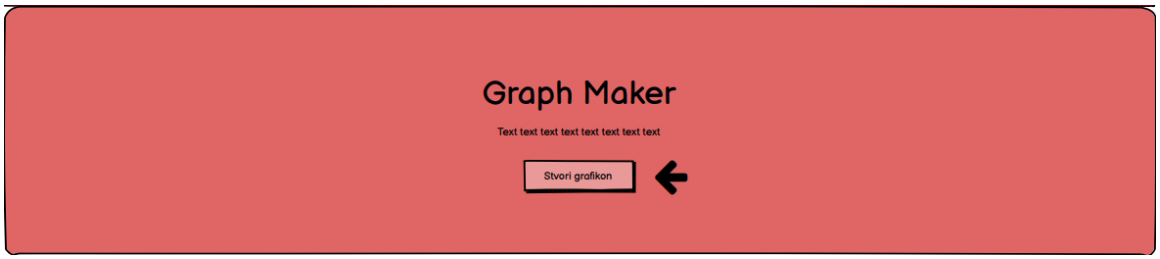
Navigacijska traka sadržavati će naslov aplikacije, te gumbove koji predstavljaju određeni sadržaj stranice. Klikom na određeni gumb (npr. Primjeri grafikona) animira se tranzicija na sekciju „Primjeri grafikona“ kako bi se korisniku prikazala fluidnost aplikacije. Navigacijska traka prilikom korištenja aplikacije u slučaju pomicanja prema dolje ostaje u fiksnom položaju na vrhu aplikacije. U internetskim preglednicima na računalu navigacijska traka je produžena kroz cijeli ekran kao što je prikazano na Slika 11, dok je na uređajima manjih ekrana vidljiv samo naslov, a prilikom klika na gumb na desnoj strani, stvara se lista gumbova koji se koriste za navigacijom stranice kao što je prikazano na Slika 12.

Slika 11 - skica navigacijske trake
Izvor: izradio autor

Slika 12 - skica navigacijske trake na mobilnim uređajima
Izvor: izradio autor

Zaglavlje

Zaglavlje sadržava centrirani logo „Graph Maker“ te kratak tekst o aplikaciji. Ispod teksta nalazi se dugme kojim se pokreće proces stvaranja grafikona kao što je prikazano na Slika 13.



Slika 13 - Skica zaglavlja aplikacije
Izvor: izradio autor

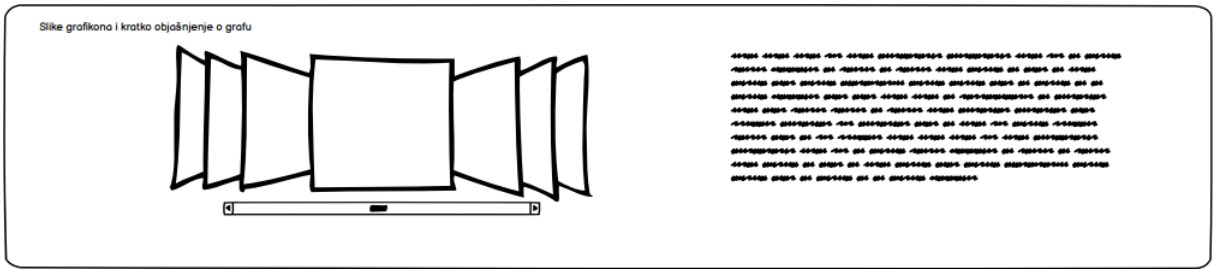
O stranici

Komponenta „O stranici“ sadržava kratak uvod o 2.3. D3.js bibliotekabiblioteci, za što se točno koristi i tko ju je izvorno napravio. Na desnoj strani komponente nalazimo sliku primjera jednog od grafikona. Prikazano na Slika 14.

Slika 14 - Skica komponente O stranici
Izvor: izradio autor

Primjeri grafikona izrađeni kroz aplikaciju

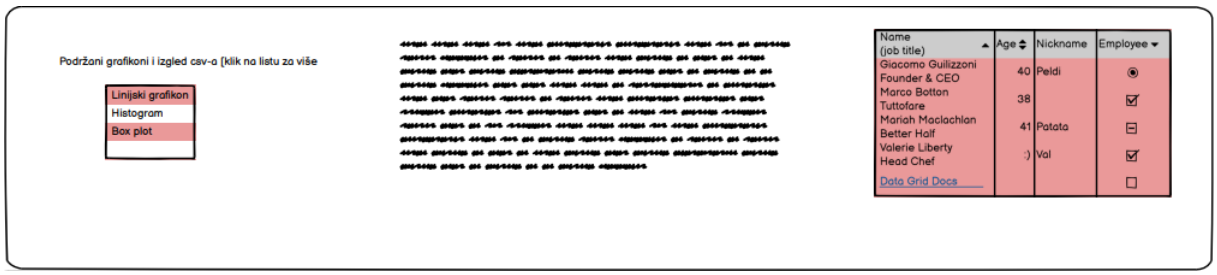
Ova komponenta se sastoji od dva elementa, 3.2.1. HTML element pod nazivom *„Carousel“* i paragraf teksta o grafikonu koji se upravo prikazuje kao što je prikazano na Slika 15. Element *Carousel* će imati automatsku animaciju „klizanja“ slika grafikona, te će se paragraf teksta mijenjati sukladno o prikazanom grafikonu.



Slika 15 - Skica komponente
Primjeri Izvor: izradio autor

Podržani grafikoni i izgled .csv datoteke

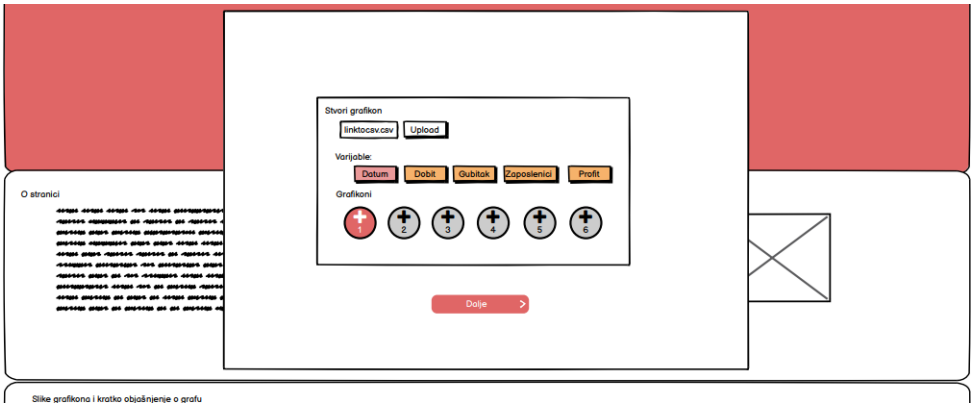
Nakon komponente „Primjeri grafikona“ prikazuje se komponenta u kojem se nalazi lista podržanih grafikona u aplikaciji, kratak tekst o definiciji odabranog grafikona i kako bi .csv datoteka trebala izgledati za taj odabrani grafikon. Prikazano na Slika 16.



Slika 16 - Skica komponente Podržani grafikoni
Izvor: izradio autor

Stvaranje grafikona

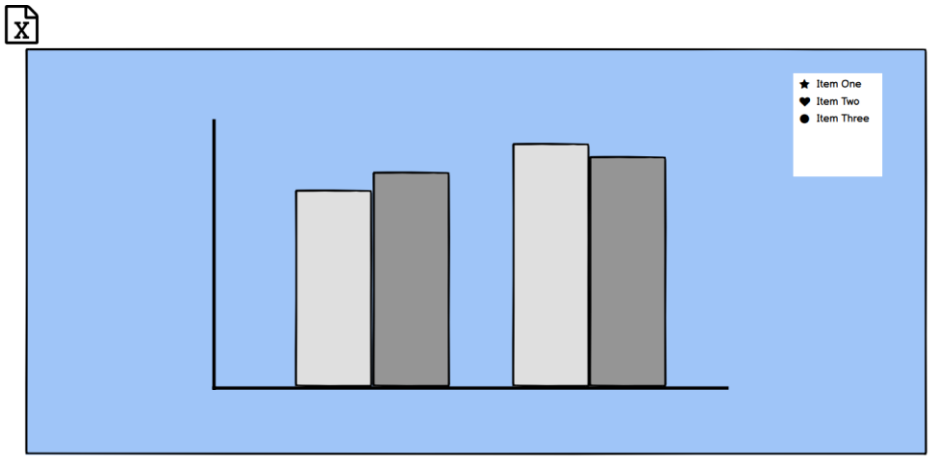
Prilikom klika na gumb u komponenti „Zaglavlje“ otvara se prozor u kojemu korisnik treba podići .csv datoteku na poslužitelj preko 3.2.1. HTML elementa <*input*>. Nakon podizanja datoteke na poslužitelj, prikazuju se sve varijable iz podignute datoteke i svi podržani grafikoni. Korisnik odabire varijable koje želi prikazati u grafikonu, također korisnik odabire i grafikon za kojeg je .csv datoteka namijenjena. Nakon odabira varijabli i grafikona prikazuje se gumb „Dalje“ koji vodi korisnika na iduću komponentu. Prikazano na Slika 17.



Slika 17 - Skica komponente Stvaranje grafikona
Izvor: izradio autor

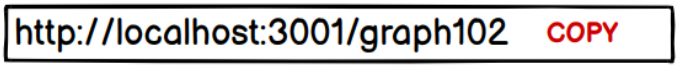
Prikaz grafikona

Nakon klika na gumb „Dalje“ koji se nalazi u komponenti „Stvaranje grafikona“, korisnika se vodi na komponentu „Prikaz grafikona“ u kojem se odabrani grafikon i njegove odabrane varijable prikazuju u obliku prozora. Ispod prozora gdje se nalazi grafikon, nalazi se gumb „Stvori link“, prilikom klika na taj gumb, stvara se link tog prikazanog grafikona kojem korisnik može pristupiti kad god i uz to može koristiti stvoreni grafikon na svojim poslovnim ili privatnim internetskim stranicama. Skica prikazana na Slika 18, Slika 19 i Slika 20.



Slika 18 - Skica prikaza grafikona
Izvor: izradio autor

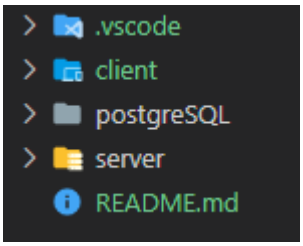
Slika 19 - Skica gumba za stvaranje poveznice
Izvor: izradio autor



Slika 20 - Skica prikaza poveznice
Izvor: izradio autor

4.3. Izrada aplikacije

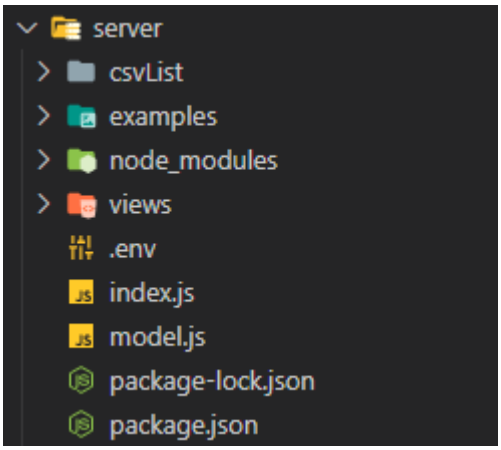
Aplikacija se sastoji od 2 dijela, frontend i backend. U frontendu se koristi 3.2.3. React.js tehnologija, koja je poznata za odličan menadžment o stanjima virtualnih DOM-ova. Pomoću 3.2.3. React.js-a stvara se izgled aplikacije te njezina funkcionalnost. U backend dijelu koristi se 3.2.4. Node.js tehnologija za komunikaciju između poslužitelja i frontenda, te 3.2.5. PostgreSQL kao baza podataka u kojoj se spremaju svi bitni podaci. Aplikacija je strukturirana putem client, postgreSQL i server direktorija kao što je prikazano na Slika 21. Client direktorij predstavlja frontend (3.2.3. React.js), postgreSQL direktorij se sastoji od SQL backup baze i server predstavlja backend (3.2.4. Node.js).



Slika 21 - Prikaz cjelokupne strukture datoteka
Izvor: izradio autor

4.3.1. Backend

3.2.4. Node.js se primarno koristi za poslužitelje koji ne blokiraju zahtjeve, koji su upravljani događajima. 3.2.4. Node.js je savršen za izradu jednostavne aplikacije poput ovog projekta, jer svaki zahtjev prema serveru je potreban nakon nekog događaja.



Slika 22 - Prikaz strukture podataka vezan za backend
Izvor: izradio autor

3.2.4. Node.js poslužitelj se sastoji od indeks.js datoteke u kojoj se stvaraju funkcije krajnjih točaka (eng. Endpoint) i logika koja je potrebna za svaku tu točku. Uz indeks.js datoteku stvorena je i model.js datoteka koja se koristi za komunikaciju sa 3.2.5. PostgreSQL bazom podataka kao što je prikazano na Slika 22.

```
const express = require('express')
const multer = require('multer')
const cors = require('cors');
require('dotenv').config()
const app = express()
const port = process.env.SRVR_PORT;
const model = require('./model')
```

Slika 23 - Inicijalizacija potrebnih paketa
Izvor: izradio autor

Na Slika 23 nalaze se svi potrebni paketi koji su uvezeni u datoteku indeks.js. Express je paket koji nam dopušta stvaranje krajnjih točaka koristeći *app()* funkciju.

Prvi korak se izvršava kada korisnik pokušava podići .csv datoteku na poslužitelj preko frontenda. Za to je potreban paket „multer“.

```
//file upload and save
const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null,'csvList')
  },
  filename: function (req, file, cb) {
    cb(null, d.getDate() + '-'  + (d.getMonth()+1) + '-' + file.originalname)
  }
})

const upload = multer ({ storage: storage }).single('file')
```

Slika 24 - Spremanje datoteke
Izvor: izradio autor

Preko multer paketa određuje se u koji direktorij datoteka se sprema. U navedenom kodu, datoteke se spremaju u direktorij nazivom *„csvList“* koji se nalazi u poslužitelju kao što je prikazano na Slika 24. Također kako ne bi došlo do imena istih datoteka, uz ime datoteke dodan je trenutni dan i mjesec, tako da datoteka koja se zove „grafikon123.csv“ nakon spremanja će se zvati „10-08-grafikon123.csv“.

```
app.post('/upload', function(req,res){

  upload(req, res, function(err) {
    if(err instanceof multer.MulterError){
      return res.status(500).json(err)
    } else if (err){
      return res.status(500).json(err)
    }
    return res.status(200).send(req.file)
  })
});
```

Slika 25 - Krajnja točka /upload
Izvor: izradio autor

Koristeći krajnju točku *app.post(„/upload“)* prikazana na Slika 25, datoteka se šalje i sprema na poslužitelj u direktorij csvList.

Kako je potrebno spremanje određenih podataka za kasnije korištenje, potrebna nam je baza podataka. Ona će se izraditi pomoću alata 3.2.5. PostgreSQL i njegovog korisničkog sučelja pgAdmin 4. Unutar već stvorene baze podataka koja se dobije prilikom instalacije alata, stvara se tablica pod imenom csvlist koja će sadržavati varijable id, csvname, graph, var1, var2, var3, var4, var5. Stvaranje tablice prikazano na Slika 26.

```
CREATE TABLE public.csvlist (
    id integer NOT NULL,
    csvname character varying(200),
    graph character varying(50),
    var1 character varying(50),
    var2 character varying(50),
    var3 character varying(50),
    var4 character varying(50),
    var5 character varying(50)
);
```

Slika 26 - Stvaranje tablice csvlist
Izvor: izradio autor

Treći korak je spremanje imena datoteka, varijable koje su odabrane i grafikon koji je odabran u bazu podataka. To će se odraditi kroz krajnju točku „/list“.

```
app.post('/list',(req,res) => {
  model.insertIntoList(req.body)
  .then(response=> {
    res.status(200).send(response);

  })
  .catch(error => {
    res.status(500).send(error);
  })
})
```

Slika 27 – Krajnja točka /list
Izvor: izradio autor

App.post() funkcija navedena na slici se poziva kada se pozove adresa „/list“ koja u trenutku korištenja poziva funkciju iz model.js datoteke kao što je prikazano na Slika 27.

```
const insertIntoList = (body) => {
  return new Promise(function(resolve,reject){
    const {csvname, graph, var1, var2, var3, var4, var5} = body
    pool.query('INSERT INTO csvlist (csvname, graph, var1, var2, var3, var4, var5) values ($1, $2, $3, $4, $5, $6, $7)   RETURNING *',
    [csvname, graph, var1, var2, var3, var4, var5], (error,results)=> {
      if(error){
        reject(error)
      }
      console.log(results.rows[0].id);
      resolve(results);
    })
  })
}
```

Slika 28 - Spremanje podataka u bazu podataka
Izvor: izradio autor

Koristeći *pool.query()* funkciju, koja se koristi za SQL naredbe, ubacuju se potrebni podaci u 3.2.5. PostgreSQL bazu podataka, te ako je uspješno funkcija *insertIntoList()*, koja je prikazana na Slika 28, vraća sve varijable ubačene u 3.2.5. PostgreSQL bazu podataka kroz *resolve()* funkciju. Ako funkcija nije uspješna, zahtjev vraća grešku kroz *reject()* funkciju.

	id [PK] integer	csvname character varying (200)	graph character varying (50)	var1 character varying (50)	var2 character varying (50)	var3 character varying (50)	var4 character varying (50)	var5 character varying (50)
145	145	28-7-DOGE_USD_2020-02-23_...	Linijski grafikon	Date	Closing Price	24h Open	24h High	24h Low
146	146	28-7-DOGE_USD_2020-02-23_...	Linijski grafikon	Date	Closing Price	24h Open	24h High	24h Low
147	147	28-7-DOGE_USD_2020-02-23_...	Linijski grafikon	Date	Closing Price	24h Open	24h High	24h Low
148	148	28-7-DOGE_USD_2020-02-23_...	Linijski grafikon	Date	Closing Price	24h Open	24h High	24h Low
149	149	28-7-histogram-data-2var.csv	Histogram	type	value	[null]	[null]	[null]
150	150	28-7-DOGE_USD_2020-02-23_...	Linijski grafikon	Date	Closing Price	24h Open	24h High	24h Low
151	151	28-7-scatterplot.csv	Scatter plot	GrLivArea	SalePrice	[null]	[null]	[null]

Slika 29 - Izgled tablice csvlist nakon spremanja podataka
Izvor: izradio autor

Kako sada postoje svi podaci, prikazani na Slika 28, koji su potrebni za izradu grafikona, stvara se nova krajnja točka koja će se baviti dohvaćanjem tih podataka određenim prema broju Id.

```
app.get('/graph:id', function(req,res) {
  let parsedCsv = {};
  model.getListById(req.params.id)
  .then(response => {
      function resolveReadStream(){
        return new Promise(resolve=> {
          fs.createReadStream(__dirname+'/csvList/' + response.rows[0].csvname).pipe(
            parse({columns:true, trim:true, encoding:true,bom:true},function(err,records){
              parsedCsv = records;
              resolve(parsedCsv);
            }))
        })
      }
```

Slika 30 - Krajnja točka /graph:id
Izvor: izradio autor

Kroz krajnju točku „/graph:id“, prikazana na Slika 30, i funkciju *getListById()*, prikazana na Slika 31, dohvaćaju se svi podaci koji se nalaze uz navedenu id varijablu. Što bi značilo ako se upiše „/graph35“ funkcija vraća sve varijable koje su unesene u tablici csvlist pod id brojem 35.

```
const getListById = (id) => {
  return new Promise(function(resolve, reject) {
    const idc = parseInt(id);
    pool.query('SELECT * FROM csvlist WHERE id = $1', [idc], (error, results) => {
      if(error){
        reject(error)
      }
      resolve(results)
    })
  })
}
```

Slika 31 - Dohvaćanje podataka iz baze podataka
Izvor: izradio autor

Ako se funkcija pozitivno izvrši, vraća podatke u obliku *results* varijable. Nakon dohvaćanja podataka iz baze podataka, poziva se paket fs i njegova funkcija *createReadStream()*, prikazana na Slika 30, koji dohvaća datoteku pod određenim imenom sa poslužitelja. Ako je funkcija *createReadStream()* uspješna, ta dohvaćena datoteka se raščlanjuje kroz paket csv-parse i sprema u varijablu parsedCsv. Kada paket za raščlanjivanje vrati rezultat u *parsedCsv* varijablu, poziva se funkcija *send()* koja je prikazana na Slika 32.

```
    async function send(){
      await resolveReadStream();
      res.render('graph', {
        'sendGraphInfo' : response,
        'parsedCsv' : parsedCsv
      })
    }
    send();
  })
})
```

Slika 32 - Prijenos podataka u .ejs view datoteku
Izvor: izradio autor

Funkcija *send()* sadrži u sebi *res.render()* funkciju koja se koristi za slanje određenih podataka (sendGraphInfo, parsedCsv) u .ejs datoteku koja se koristi za prikazivanje 3.2.1. HTML-a preko poslužitelja. Graph.ejs datoteka služi za prikaz D3 grafikona, koji se stvaraju svaki put kad korisnik pristupi određenom linku, u ovom slučaju http://localhost:3001/graph100, umjesto broja 100 potrebno je upisati broj Id-a grafikona kojem se želi pristupiti.

D3.js biblioteka (.ejs view)

Kao i u svakom 3.2.1. HTML dokumentu, biblioteke se pozivaju u <*head*> elementu. Za potrebe ovog rada, koristiti će se paket d3.v4.min.js, jquery.min.js i moment.min.js koji su inicijalizirani u <*head*> element, prikazano na Slika 33.

```
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
    <script src="https://d3js.org/d3.v4.min.js"></script>
    <script src = "https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.1/moment.min.js"></script>
```

Slika 33 - Inicijalizacija potrebnih paketa
Izvor: izradio autor

Kako bi mogli pristupiti varijablama koji su poslani kroz funkciju *send()*, potrebno je koristiti *JSON.stringify()* funkciju kako bi ih pretvorili u veliku tekstualnu varijablu (eng. string), te nakon toga preko *JSON.parse()* funkcije potrebno ih je raščlaniti kako bi se pretvorili u objekte koji se mogu koristiti. Proces prikazan na Slika 34.

```
// <!-- getting variables from table -->
var graphInformation = '<%-JSON.stringify(sendGraphInfo)%>';
var csvInformation = '<%-JSON.stringify(parsedCsv)%>';

graphInfo = JSON.parse(graphInformation);
csvInformation = JSON.parse(csvInformation);
```

Slika 34 - Raščlanjivanje podataka
Izvor: izradio autor

Nakon raščlanjivanja, podaci se mogu koristiti u obliku objekata. Koristeći moment biblioteku, kao što je prikazano na Slika 35, potrebno je provjeriti postoje li datumi u .csv datoteci, jer 2.3. D3.js biblioteka prepoznaje samo određeni format.

```
//check if there are any dates in csv
d3.keys(csvInformation[1]).forEach(
  function(d){
    datum = moment(csvInformation[1][d],"YYYY-MM-DD",true);
    if(datum > 0){
      stupacDatum = d;
    }
  }
)
```

Slika 35 - Provjera podataka
Izvor: izradio autor

Ako datumi postoje unutar određenog skupa podataka, varijabla stupacDatum nam daje ime varijable u kojoj se nalaze datumi, pa kroz *Date.parse()* funkciju datume raščlanjujemo u format koji 2.3. D3.js biblioteka može prepoznati kao što je prikazano na Slika 36.

```
//parsiranje datuma
if(!(stupacDatum === "")){
    csvInformation.forEach(function(row){
      row[stupacDatum] = Date.parse(row[stupacDatum]);
    })
}
```

Slika 36 - Raščlanjivanje potrebnih podataka
Izvor: izradio autor

Podatke izvučene iz .csv datoteke potrebno je podijeliti na X os i Y os. Aplikacija je napravljena tako da prva varijabla koja je odabrana ide na X os a sve ostale na Y os u svrhu jednostavnosti.

```
keyX = trueOdabrano[0];

keysY = trueOdabrano.slice(1);
```

Slika 37 - Raščlanjivanje podataka u varijable
Izvor: izradio autor

keyX varijabla sadrži ime prve odabrane varijable, dok keysY je polje svih ostalih odabranih imena varijabli. Nakon podijele, varijable na X osi je potrebno poredati od manje do veće kako bi grafikon bio čitljiv kao što je prikazano na Slika 38, a varijable koje se odnose na Y os se spremaju u polje valuesY.

```
csvInformation.sort((x,y) => {
  a = x[keyX];
  b = y[keyX];
  if (a > b) { return 1; } if( a < b) { return -1; } return 0;
});

csvInformation.forEach(
  function(row){
    keysY.forEach(
      function(key){
        valuesY.push(row[key]);
      }
    )
  }
)
```

Slika 38 - Sortiranje potrebnih podataka
Izvor: izradio autor

Kako bi se prikazala korisnost 2.3. D3.js biblioteka, odabrana su 6 grafikona zbog svojih različitosti i korisnosti. To su linijski grafikon, histogram, grafikon kutije i brkova, grafikon rasipanja, stupčasti grafikon i toplinska karta.

Linijski grafikon

Svaka izrada grafikona počinje sa određivanjem visine, širine, margine i dodavanjem svg elementa na već postojeći div element sa oznakom „#nacrtajGraf“, prikazano na Slika 39.

```
//linijski
if(graphName == "Linijski grafikon"){
  var margin = { top: 30, right: 120, bottom: 30, left: 50 },
        width = 790 - margin.left - margin.right,
        height = 360 - margin.top - margin.bottom,
        tooltip = { width: 100, height: 100, x: 10, y: -30 };

  var svg = d3.select("#nacrtajGraf").append("svg")
          .attr("width", width + margin.left + margin.right)
          .attr("height", height + margin.top + margin.bottom)
          .append("g")
              .attr("transform", "translate(" + margin.left + "," + margin.top + ")");
```

Slika 39 - Definiranje dimenzija linijskog grafikona
Izvor: izradio autor

Nakon određivanja svojstva svg elementa, potrebno je stvoriti x i y os. Kako je moguće da X os se sastoji od varijabla koje predstavljaju vrijeme potrebno je koristiti funkciju *d3.scaleTime()*. U slučaju da datumi ne postoje, potrebno je koristiti *d3.scaleLinear()*, prikazano na Slika 40.

```
if(stupacDatum == keyX){
  var x = d3.scaleTime()
    .domain(d3.extent(csvInformation, function(d) { return d[keyX]; }))
    .nice()
    .range([0,width]);
}
else {
  var x = d3.scaleLinear()
    .domain(d3.extent(csvInformation, function(d) { return d[keyX]; }))
    .nice()
    .range([0,width]);
}

xAxis = svg.append("g")
  .attr("transform", "translate(0," + height + ")")
  .call(d3.axisBottom(x));
```

Slika 40 - Crtanje X osi za linijski grafikon
Izvor: izradio autor

Varijabla x kroz *domain()* funkciju dobiva kroz funkciju d3.extent() najmanje i najveće vrijednosti koje idu na x os. Funkcija *nice()* se koristi nakon *domain()* funkcije za zaokruživanje prve i zadnje vrijednosti. Na kraju, varijabli xAxis dodajemo element g koji služi za grupiranje svg elemenata i kroz funkciju *d3.axisBottom()* stvaramo x os sa svojstvima x varijable, kao što je prikazano na Slika 40.

```
var y = d3.scaleLinear()
  .domain([d3.min(valuesY), d3.max(valuesY)])
  .nice()
  .range([ height, 0 ]);

yAxis = svg.append("g")
  .attr("height", d3.max(valuesY))
  .call(d3.axisLeft(y));
```

Slika 41 - Crtanje Y osi za linijski grafikon
Izvor: izradio autor

Procedura za stvaranje y os je vrlo slična kao i za x os, samo što se na kraju poziva funkcija *d3.axisLeft()*.

```
//zoom
var clip = svg.append("defs")
                .append("svg:clipPath")
                .attr("id","clip")
                .append("svg:rect")
                    .attr("width", width)
                    .attr("height",height)
                    .attr("x",0)
                    .attr("y",0);

var brush = d3.brushX()
    .extent( [ [0,0], [width,height]])
    .on("end", updateChart);

keysY.forEach(function(d,i){
    line[i] = svg.append("g")
        .attr("clip-path","url(#clip)");
});
```

Slika 42 - Dodavanje mogućnosti približavanja u linijskom grafikonu
Izvor: izradio autor

Dodavanje mogućnosti približavanja (eng. Zoom) u linijskom grafikonu je moguće kroz *d3.brushX()* funkciju koja prima *extent()* funkciju, i prilikom završetka odabira kutije približavanja poziva se funkcija *updateChart()* u kojoj se ponovno crta grafikon. Funkcija prikazana na Slika 42.

```
keysY.forEach(
    function(key,i){
        line[i].append("path")
            .data([csvInformation])
            .attr("fill","none")
            .attr("stroke",colors[i])
            .attr("stroke-width", 1.5)
            .attr("class","line")
            // .style("stroke","red")
            .attr("d",d3.line()
                .x(function(d) { return x(d[keyX])})
                .y(function(d){ return y(d[key])}));
    }
);
```

Slika 43 - Popunjavanje linijskog grafikona
Izvor: izradio autor

Navedeni kod prikazan na Slika 43, za svaku varijablu koja se odnosi na y os, dodaje se „path“ ili linija koja poziva .csv datoteku kroz *data()* funkciju, nakon koje se dodaje atribut koji poziva *d3.line()* funkciju koja prima X i Y koordinate kroz koje 2.3. D3.js biblioteka stvara pravac ili liniju.

```
function updateChart(){
    extent = d3.event.selection;


    xAxis.transition()
        .duration(1000)
        .call(d3.axisBottom(x));

    keysY.forEach(
    function(key,i){
            line[i].select(".line")
                .transition()
                .duration(1000)
                .attr("d",d3.line()
                    .x(function(d){ return x(d[keyX])})
                    .y(function(d){ return y(d[key])})
                )
        }
    );
}
```

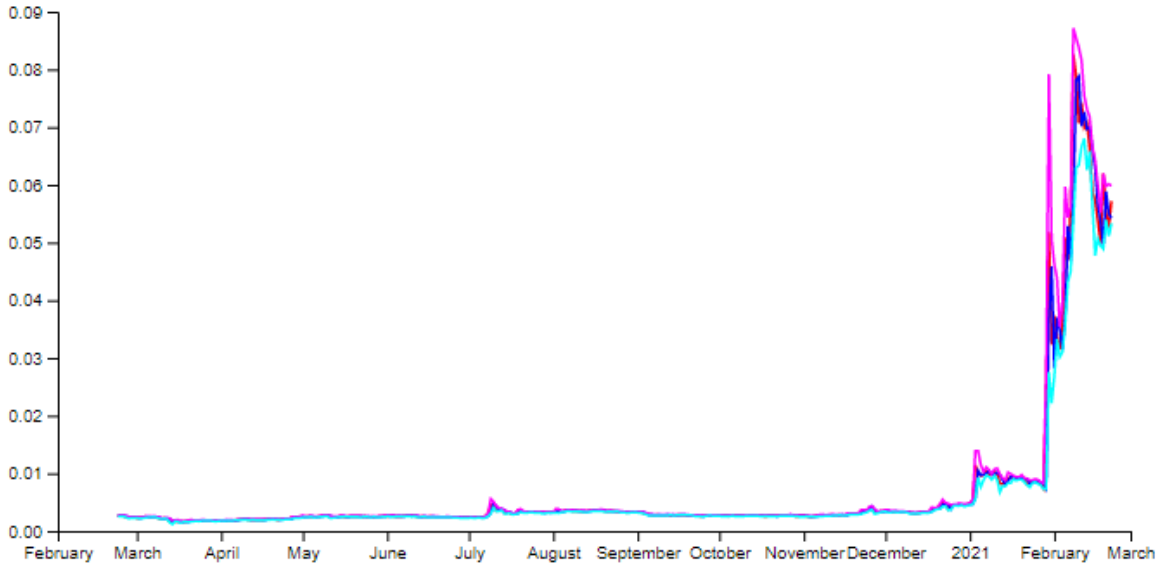
Slika 44 - Ponovno popunjavanje grafikona
Izvor: izradio autor

Nakon pozivanja funkcije *updateChart()*, linijski grafikon se ponovno crta sa novim vrijednostima kako bi dao dojam približavanja.

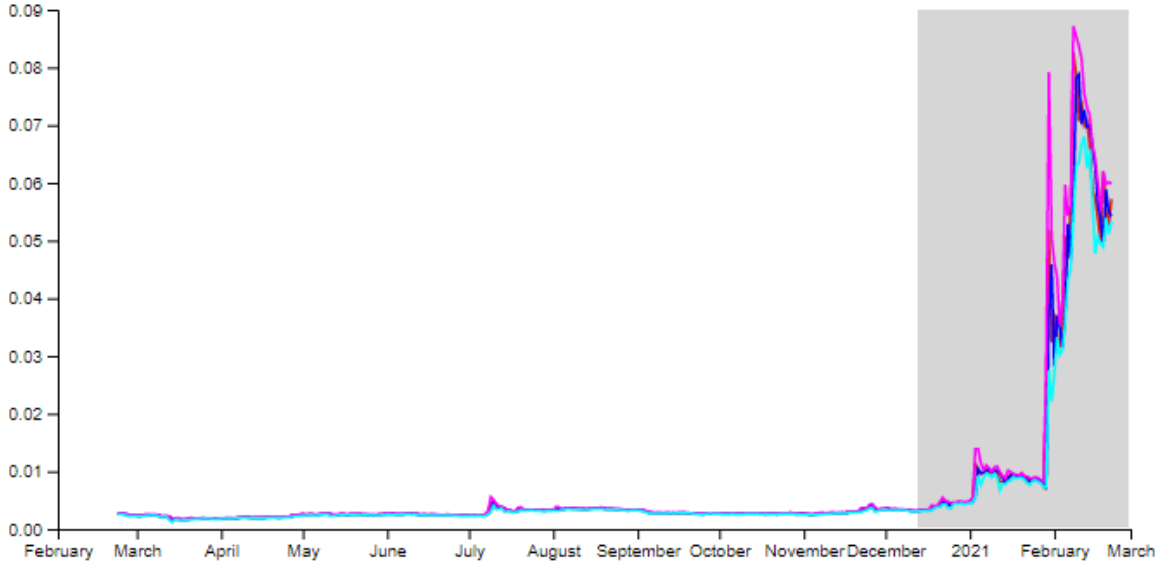
```
Date,Closing Price,24h Open,24h High,24h Low
2020-02-23,0.0026136125,0.0026834801,0.0026902328,0.0025290483
2020-02-24,0.0026692541,0.0026136253,0.002691479,0.0025365152
2020-02-25,0.0025909713,0.0026692637,0.002691688,0.0025267846
2020-02-26,0.0024537252,0.0025909494,0.002599641,0.002375244
2020-02-27,0.0023466349,0.0024529883,0.0025146996,0.0021805741
2020-02-28,0.0023389135,0.002346808,0.0024181598,0.0022529364
2020-02-29,0.0023377592,0.0023390171,0.0023751916,0.0022210552
2020-03-01,0.0022471773,0.0023442655,0.002344865,0.0022350934
2020-03-02,0.0022886421,0.0022472164,0.0023317819,0.0021648912
2020-03-03,0.0023726661,0.0022884641,0.0023877255,0.0022338828
2020-03-04,0.0025235114,0.0023725221,0.0025415043,0.0022997676
2020-03-05,0.0024120316,0.0025235198,0.0025383908,0.0023675188
2020-03-06,0.0025141416,0.0024111311,0.0025426673,0.0023994653
2020-03-07,0.002477819,0.0025141973,0.0025285644,0.0024461818
2020-03-08,0.0024120749,0.0024777307,0.0025197651,0.0023935657
2020-03-09,0.0022003277,0.002412254,0.002462131,0.0021749744
2020-03-10,0.0022028712,0.0022003625,0.0022363426,0.0020806593
2020-03-11,0.0022193933,0.0022028712,0.0022726427,0.0021506527
2020-03-12,0.0021950679,0.0022195088,0.0022571482,0.0020883073
2020-03-13,0.0016970346,0.002195417,0.0022066637,0.0015181702
2020-03-14,0.0018056642,0.0016965899,0.0018181075,0.0011763269
2020-03-15,0.0016892934,0.0017988884,0.0018399709,0.0016525391
2020-03-16,0.0016379067,0.0016917116,0.0019592959,0.0016243435
2020-03-17,0.0015822947,0.0016664692,0.001706553,0.0014046375
2020-03-18,0.0016745348,0.0015819058,0.0016954234,0.0015515448
2020-03-19,0.0016040272,0.0016742521,0.001680259,0.0015414407
2020-03-20,0.0018036115,0.0016040578,0.0018622201,0.0015576034
```

Slika 45 - Izgled DOGE-USDT.csv datoteke
Izvor: izradio autor

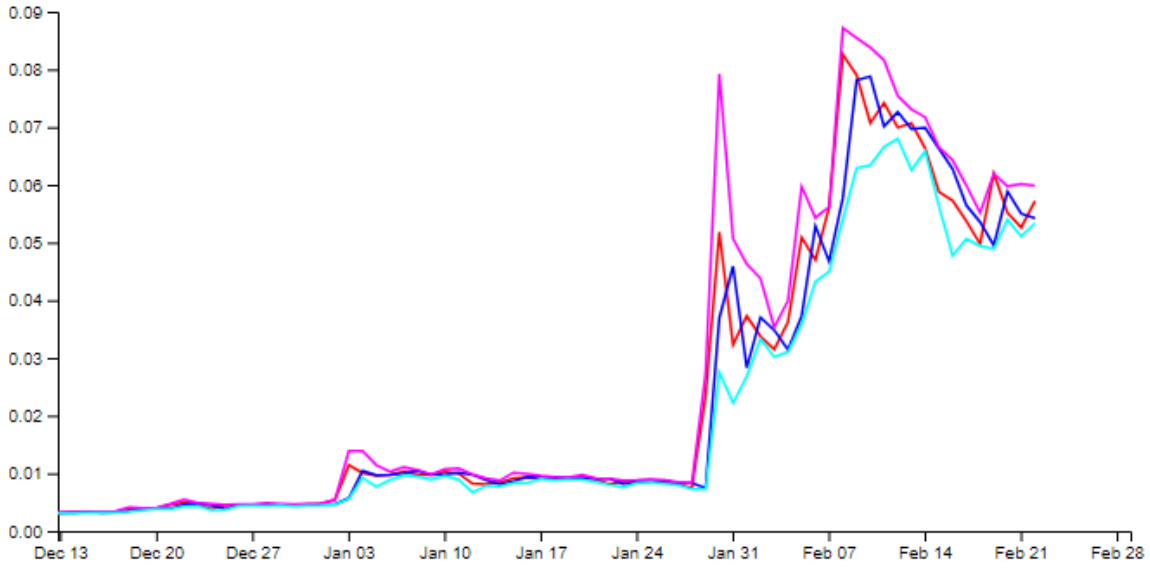
Koristeći financijske podatke .csv datoteke za DOGE kripto valutu, podijeljene delimiterom ',', stvara se .csv datoteka, koja prilikom unošenja svih potrebnih varijabli, stvara linijski grafikon prikazan na Slika 46.



Slika 46 - Linijski grafikon DOGE-USDT.csv datoteke
Izvor: izradio autor



Slika 47 - Izgled grafikona prilikom približavanja
Izvor: izradio autor



Slika 48 - Izgled grafikona nakon približavanja
Izvor: izradio autor

Histogram

Kao i u linijskom grafikonu, potrebno je inicijalizirati varijablu margin i varijablu svg koja se poziva na <*div*> element naziva „#nacrtajGraf“ nakon čega se stvara X i Y osi na sličan način.

```
var histogram = d3.histogram()
  .value(function(d) { return d[keyX]; })
  .domain(x.domain())
  .thresholds(x.ticks(nBin));

var bins1 = histogram(csvInformation);
```

Slika 49 - Definiranje histograma
Izvor: izradio autor

Za razliku od linijskog grafikona, za stvaranje histograma koristimo *d3.histogram()* funkciju koja prima parametre value, domain i thresholds koji su vidljivi na Slika 49. U value parametru se nalaze sve vrijednosti koji idu u histogram, parametar domain prima funkciju *x.domain()* koja vraća najmanje i najveće vrijednosti, dok parametar thresholds vraća brojač varijabli koji definira izgled histograma.

```
var u = svg.selectAll("rect")
.data(bins1);
u
    .enter()
    .append("rect")
    .merge(u)
    .transition()
    .duration(1000)
    .attr("x", 1)
        .attr("transform", function(d) { return "translate(" + x(d.x0) + "," + y(d.length) + ")"; })
        .attr("width", function(d) { return x(d.x1) - x(d.x0) -1 ; })
        .attr("height", function(d) { return height - y(d.length); })
        .style("fill", "#69b3a2")

u
    .exit()
    .remove()
```

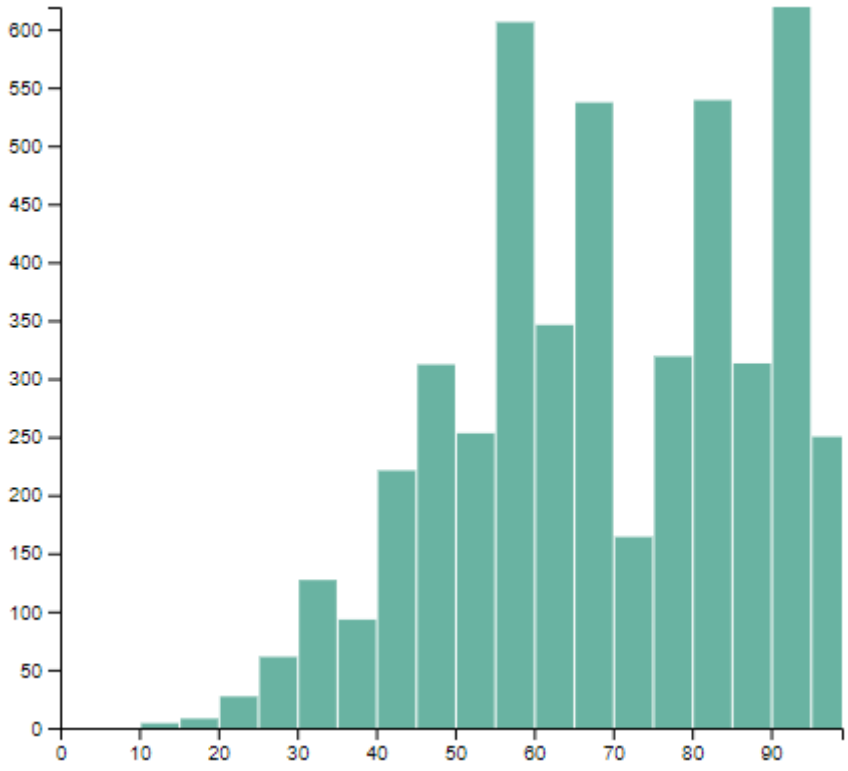
Slika 50 - Popunjavanje histograma
Izvor: izradio autor

Kroz navedeni kod, kutije histograma se crtaju unutar već stvorene x i y osi kao što je prikazano na Slika 50.

Koristeći financijske podatke iz .csv datoteke (prikazana na Slika 51) moguće je nacrtati histogram pomoću jedne varijable koji je prikazan na Slika 52.

```
price
75.0
104.0
369.0
300.0
92.0
64.0
265.0
35.0
287.0
69.0
52.0
23.0
287.0
87.0
114.0
114.0
98.0
137.0
87.0
90.0
63.0
69.0
80.0
113.0
```

Slika 51 - Izgled price.csv datoteke
Izvor: izradio autor



Slika 52 - Izgled histograma sa jednom varijablom
Izvor: izradio autor

Ako je potrebno nacrtati histogram sa dvije varijable, potrebno je napraviti preinake u kodu.

```
if(trueOdabrano.length == 1){···
else if(trueOdabrano.length == 2){
  var histogram = d3.histogram()
    .value(function(d){ return +d[keysY]; })
    .domain(x.domain())
    .thresholds(x.ticks(nBin));

  csvInformation.forEach(
    function(d){
      console.log(d[keyX]);
        if(variableNames.indexOf(d[keyX]) === -1){
          variableNames.push(d[keyX]);
        }
    }
  )

  var bins1 = histogram(csvInformation.filter(function(d){ return d[keyX] === variableNames[0]}));
  var bins2 = histogram(csvInformation.filter(function(d){ return d[keyX] === variableNames[1]}));
```

Slika 53 - Definiranje histograma sa dvije varijable
Izvor: izradio autor

Ako postoje dvije odabrane varijable, potrebno je povući vrijednosti oboju varijabli kroz d3.histogram funkciju, i vrijednosti tih varijabli filtrirati i spremiti u posebne varijable kao što je prikazano na Slika 53.

```
var u2 = svg.selectAll("rect2")
.data(bins2);
u2
    .enter()
    .append("rect")
    .merge(u2)
    .transition()
    .duration(1000)
    .attr("x", 1)
        .attr("transform", function(d) { return "translate(" + x(d.x0) + "," + y(d.length) + ")";
        .attr("width", function(d) { return x(d.x1) - x(d.x0) -1 ; })
        .attr("height", function(d) { return height - y(d.length); })
        .style("fill", "#404080")
        .style("opacity",0.4)

u2
    .exit()
    .remove()
```

Slika 54 - Popunjavanje histograma sa dvije varijable
Izvor: izradio autor

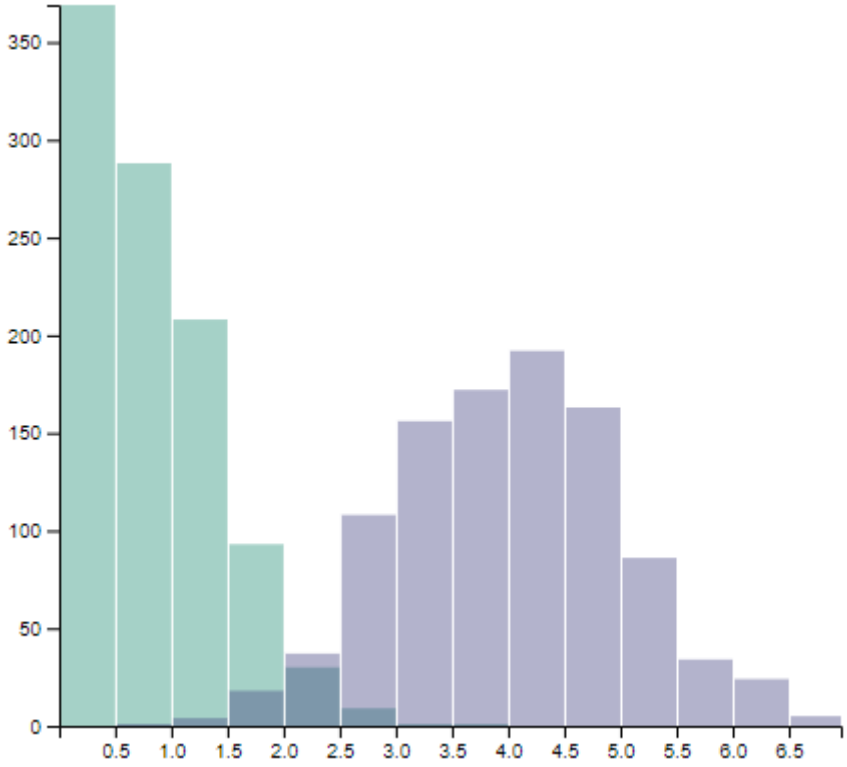
Nakon crtanja prve varijable kroz u varijablu, potrebno je dodati u2 varijablu (prikazana na Slika 54) koja crta podatke iz druge odabrane varijable i s time stvara histogram sa 2 prikazane varijable, koji je prikazan na Slika 54.

```
type,value
variable 1,1.77631469016691
variable 1,1.64657632287532
variable 1,0.488817419851397
variable 1,0.124094377926691
variable 1,0.0235808204897442
variable 1,0.0633392229143902
variable 1,1.12003877806003
variable 1,0.588100796118183
variable 1,0.910824156455993
variable 1,0.962126919708827
variable 1,1.01869346346051
variable 1,1.35875640761682
variable 1,1.24427385193651
variable 1,0.239585475810887
variable 1,0.568272231735263
variable 1,0.0103423314045569
variable 1,0.953989539491403
variable 1,1.60633418356557
variable 1,1.26566678276414
variable 1,1.79205408275276
variable 1,0.718862570787166
variable 1,0.114161283633124
```

Slika 55 - Izgled histogram.csv datoteke 1/2
Izvor: izradio autor

```
variable 2,4.23191027826733
variable 2,5.6252868981076
variable 2,1.74107095437678
variable 2,6.07056508776174
variable 2,5.28276157231583
variable 2,4.75504967045242
variable 2,1.65772266824258
variable 2,4.13051170056847
variable 2,3.62961527741702
variable 2,3.79004775393609
variable 2,6.0598581072695
variable 2,3.87367954564361
variable 2,3.25600084276851
variable 2,4.70364248563106
variable 2,3.22172374626842
variable 2,2.60595802256354
variable 2,6.64806991808888
variable 2,2.73149807418538
variable 2,2.65635809542417
variable 2,3.67884764080831
variable 2,3.34546764916297
variable 2,1.56714869942661
variable 2,3.1428737079587
```

Slika 56 - Izgled histogram.csv datoteke 2/2
Izvor: izradio autor



Slika 57 - Izgled histograma sa dvije varijable
Izvor: izradio autor

Grafikon kutije i brkova

Grafikon kutije i brkova je jedan od jednostavnijih grafikona za crtanje kroz 2.3. D3.js biblioteka biblioteku.

Kao i svi ostali grafikoni, potrebno je inicijalizirati varijable margin i svg.

```
csvInformation.forEach(function(d,i){
    boxPlotArray[i] = d[keyX];
})

var data_sorted = boxPlotArray.sort((a,b) => a-b);
var q1 = d3.quantile(data_sorted, .25);
var median = d3.quantile(data_sorted, .5);
var q3 = d3.quantile(data_sorted, .75);
var interQuantileRange = q3 - q1;
var min = q1 - 1.5 * interQuantileRange;
var max = q1 + 1.5 * interQuantileRange;
```

Slika 58 - Inicijalizacija potrebnih varijabli
Izvor: izradio autor

Prvi korak je spremiti sve vrijednosti unutar boxPlotArray polja kako bi se mogla izvršiti potrebna statistika za crtanje grafikona kutije i brkova kao što je prikazano na Slika 58.

```
var y = d3.scaleLinear()
  .domain([0,Math.max.apply(null,data_sorted)])
  .range([height, 0]);
  svg.call(d3.axisLeft(y));
```

Slika 59 - Crtanje Y osi za boxplot grafikon
Izvor: izradio autor

U drugom koraku crta se Y os, koja koristi d3.scaleLinear() funkciju koja ima domain parametar koji prima 0 kao najmanju vrijednost, i najveću vrijednost koja se dobiva kroz *Math.max()* funkciju, prikazano na Slika 59.

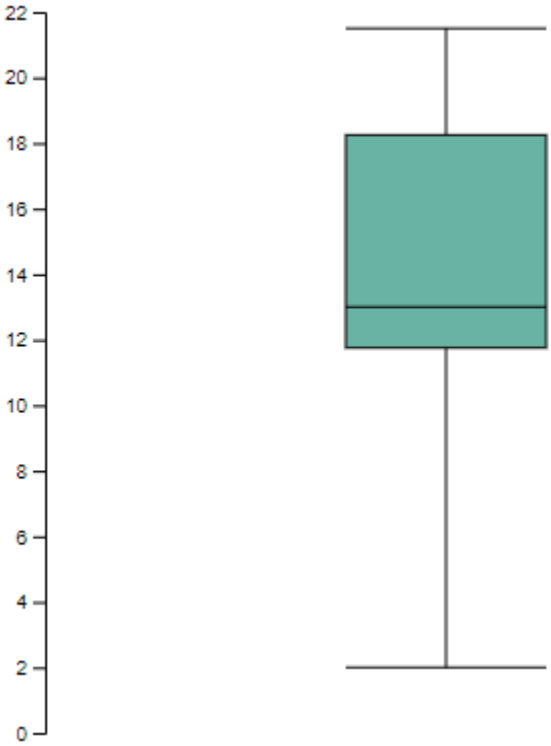
```
svg
  .append("line")
  .attr("x1", center)
  .attr("x2", center)
  .attr("y1", y(min) )
  .attr("y2", y(max) )
  .attr("stroke", "black");

  svg
    .append("rect")
    .attr("x", center - width/2)
    .attr("y", y(q3) )
    .attr("height", (y(q1)-y(q3)) )
    .attr("width", width )
    .attr("stroke", "black")
    .style("fill", "#69b3a2");

  svg
    .selectAll("toto")
    .data([min, median, max])
    .enter()
    .append("line")
    .attr("x1", center-width/2)
    .attr("x2", center+width/2)
    .attr("y1", function(d){ return(y(d))} )
    .attr("y2", function(d){ return(y(d))} )
    .attr("stroke", "black");
```

Slika 60 - Popunjavanje grafikona kutije i brkova
Izvor: izradio autor

Nakon crtanja Y osi, potrebno je dodati elemente „line“ i „rect“, prikazani na Slika 60, koji primaju atribute od iznad izračunatih vrijednosti i na kraju dodati podatke iz min, median i max varijabli s kojim grafikon dobiva izgled grafikon kutije i brkova prikazan na Slika 62.

```
podaci
12
19
11
13
12
22
13
4
15
16
18
19
20
12
11
9
```

Slika 61 - Izgled podaci.csv datoteke
Izvor: izradio autor



Slika 62 - Izgled grafikona kutije i brkova
Izvor: izradio autor

Grafikon rasipanja

Potrebno je inicijalizirati margin i svg varijable. Nakon čega se crtaju X i Y os koristeći *d3.scaleLinear()* funkciju.

```
var myCircle = svg.append('g')
                  .selectAll("dot")
                  .data(csvInformation)
                  .enter()
                  .append("circle")
                  .attr("cx", function (d) { return x(d[keyX]); } )
                  .attr("cy", function (d) { return y(d[keysY]); } )
                  .attr("r", 1.5)
                  .style("fill", "#69b3a2");
```

Slika 63 - Popunjavanje grafikona rasipanja
Izvor: izradio autor

Stvaranje varijable myCircle, prikazana na Slika 63, koja sadrži koordinate točaka koji se trebaju nacrtati u grafikonu.

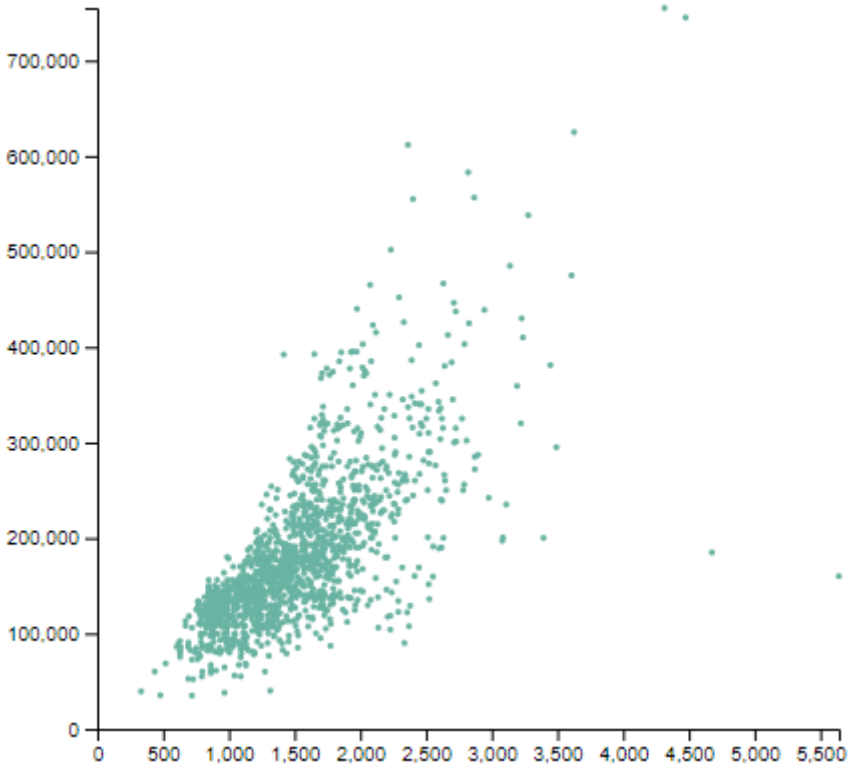
```
svg.select(".myXaxis")
                  .transition()
                  .duration(2000)
                  .attr("opacity","1")
                  .call(d3.axisBottom(x));

svg.selectAll("circle")
  .transition()
  .delay(function(d,i){ return (i*3)})
  .duration(2000)
  .attr("cx", function (d) { return x(d[keyX]); } )
  .attr("cy", function (d) { return y(d[keysY]); } )
```

Slika 64 - Dodavanje animacija u grafikon rasipanja
Izvor: izradio autor

U trenutku učitavanja grafikona, pomoću *.transition().duration()* funkcija, prikazane na Slika 64, animirana je X os kao i točkice koje se crtaju u grafikonu rasipanja koji je prikazan na Slika 66 .

```
GrLivArea,SalePrice
1710,208500
1262,181500
1786,223500
1717,140000
2198,250000
1362,143000
1694,307000
2090,200000
1774,129900
1077,118000
1040,129500
2324,345000
912,144000
1494,279500
1253,157000
854,132000
1004,149000
1296,90000
1114,159000
1339,139000
2376,325300
1108,139400
```

Slika 65 - Izgled AveragePrice.csv datoteke
Izvor: izradio autor



Slika 66 - Izgled grafikona rasipanja
Izvor: izradio autor

Stupčasti grafikon

Potrebno je inicijalizirati margin i svg varijable. Nakon čega se crtaju X i Y osi koristeći *d3.scaleBand()* funkciju.

```
csvInformation.sort(function(b,a){
  return a.Value - b.Value;
});
```

Slika 67 - Sortiranje podataka
Izvor: izradio autor

Nakon toga potrebno je sortirati podatke, prikazano na Slika 67, za stupčasti grafikon kako bi grafikon bio lakše čitljiv.

```
var x = d3.scaleBand()
        .range([ 0, width ])
        .domain(csvInformation.map(function(d) { return d[keyX]; }))
        .padding(0.2);

svg.append("g")
      .attr("transform", "translate(0," + height + ")")
      .call(d3.axisBottom(x))
      .selectAll("text")
          .attr("transform", "translate(-10,0)rotate(-45)")
          .style("text-anchor", "end");
```

Slika 68 - Crtanje X osi za stupčasti grafikon
Izvor: izradio autor

Stupčasti grafikon za crtanje X osi koristi *d3.scaleBand()* funkciju, prikazana na Slika 68, koja ravnomjerno dijeli udaljenost između elemenata. Kao i u primjerima prošlih grafikona, koristimo *d3.axisBottom()* funkciju za crtanje X osi.

```
var y = d3.scaleLinear()
    .domain([0,d3.max(csvInformation,function(d){ return +d[keysY];})])
    .range([ height, 0]);
    svg.append("g")
    .call(d3.axisLeft(y));
```

Slika 69 - Crtanje Y osi za stupčasti grafikon
Izvor: izradio autor

Vertikalna ili Y os se crta pomoću funkcije *d3.scaleLinear* i *d3.axisLeft()*. Prikazano na Slika 69.

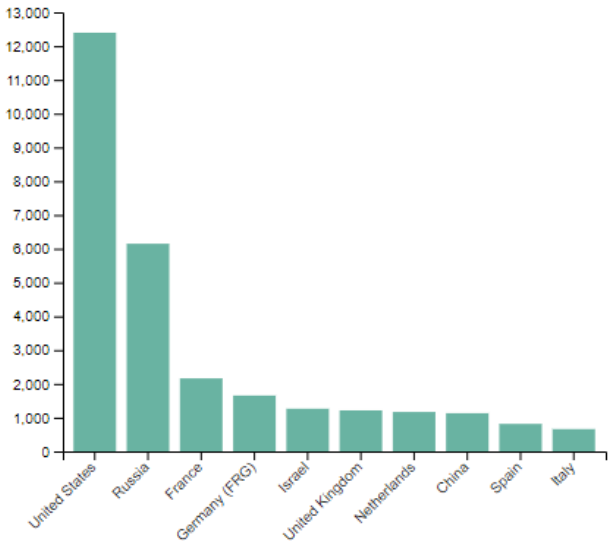
```
svg.selectAll("mybar")
        .data(csvInformation)
        .enter()
        .append("rect")
            .attr("x", function(d) { return x(d[keyX]); })
            .attr("y", function(d) { return y(d[keysY]); })
            .attr("width", x.bandwidth())
            .attr("height", function(d) { return height - y(d[keysY]); })
            .attr("fill", "#69b3a2");
```

Slika 70 - Popunjavanje stupčastog grafikona
Izvor: izradio autor

Kao zadnji korak izrade stupčastog grafikona, prikazan na Slika 70, crtaju se kutije (eng. Bar) pomoću podataka iz .csv datoteke.

```
Country,Value
United States,12394
Russia,6148
Germany (FRG),1653
France,2162
United Kingdom,1214
China,1131
Spain,814
Netherlands,1167
Italy,660
Israel,1263
```

Slika 71 - Izgled ValueperCountry.csv datoteke
Izvor: izradio autor



Slika 72 - Izgled stupčastog grafikona
Izvor: izradio autor

Toplinska karta

```
groupNames = [];

csvInformation.forEach(
  function(d){
      if(groupNames.indexOf(d[keyX]) === -1){
          groupNames.push(d[keyX]);
      }
      if(variableNames.indexOf(d[keysY[0]]) === -1){
          variableNames.push(d[keysY[0]]);
      }
  }
)
```

Slika 73 - Sortiranje podataka u polja
Izvor: izradio autor

Kako bi mogli nacrtati toplinsku kartu potrebno je grupirati imena i varijable koje se koriste na način koji je prikazan na Slika 73.

```
var x = d3.scaleBand()
    .range([ 0, width ])
    .domain(groupNames)
    .padding(0.05);

svg.append("g")
          .style("font-size",15)
          .attr("transform", "translate(0," + height + ")")
          .call(d3.axisBottom(x).tickSize(0));
```

Slika 74 - Crtanje X osi za toplinsku kartu
Izvor: izradio autor

```
var y = d3.scaleBand()
    .range([ height, 0 ])
    .domain(variableNames)
    .padding(0.05);

svg.append("g")
          .style("font-size",15)
          .call(d3.axisLeft(y).tickSize(0))
          .select(".domain").remove();
```

Slika 75 - Crtanje Y osi za toplinsku kartu
Izvor: izradio autor

Crtanje X i Y osi veoma je slično ostalim grafikonima, jedina razlika je u definiranju domene koja je prikazana na Slika 74 i Slika 75. Kod X osi koristimo polje groupNames, a kod Y osi koristimo polje variableNames.

```
var myColor = d3.scaleSequential()
    .interpolator(d3.interpolateInferno)
    .domain([1,100]);
```

Slika 76 - Definiranje boja za toplinsku kartu
Izvor: izradio autor

Kako bi toplinska karta zapravo imala boje potrebno je koristiti funkciju *d3.scaleSequential()*, koja je prikazana na Slika 76, koja stvara skalu pozvanog interpolatora, pri kojem se poziva *d3.interpolateInferno*.

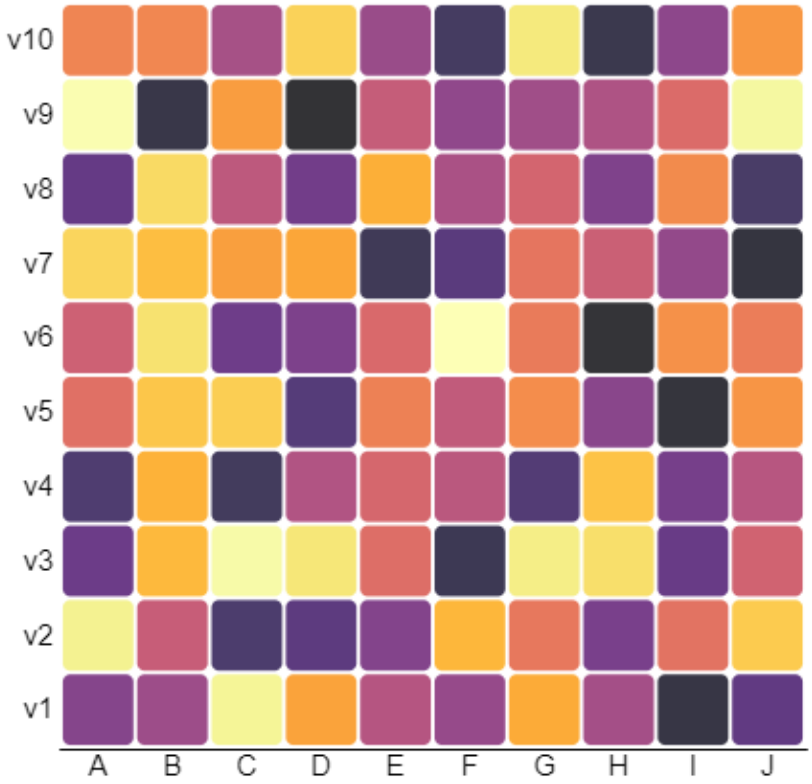
```
svg.selectAll()
    .data(csvInformation, function(d) {return d[keyX]+':'+d[keysY[0]];})
    .enter()
    .append("rect")
        .attr("x", function(d) { return x(d[keyX]) })
        .attr("y", function(d) { return y(d[keysY[0]]) })
        .attr("rx",4)
        .attr("ry",4)
        .attr("width", x.bandwidth() )
        .attr("height", y.bandwidth() )
        .style("fill", function(d) { return myColor(d[keysY[1]])} )
        .style("stroke-width",4)
        .style("stroke","none")
        .style("opacity",0.8)
    .on("mouseover",mouseover)
    .on("mousemove",mousemove)
    .on("mouseleave",mouseleave);
```

Slika 77 - Popunjavanje toplinske karte
Izvor: izradio autor

Dodavanje kutijica koji se određuju prema podacima iz .csv datoteke. Kako bi se nacrtala toplinska karta potrebne su nam grupe i varijable. Kombinacija grupe i varijable je pridružena sa određenom vrijednosti. Iznos te vrijednosti ovisi koje boje će ćelija biti ispunjena. Crtanje grafikona prikazano na Slika 77. Izgled grafikona prikazan na Slika 79.

```
group,variable,value
A,v1,30
A,v2,95
A,v3,22
A,v4,14
A,v5,59
A,v6,52
A,v7,88
A,v8,20
A,v9,99
A,v10,66
B,v1,37
B,v2,50
B,v3,81
B,v4,79
B,v5,84
B,v6,91
B,v7,82
B,v8,89
B,v9,6
B,v10,67
C,v1,96
C,v2,13
```

Slika 78 - Izgled heatmap.csv datoteke
Izvor: izradio autor



Slika 79 - Izgled toplinske karte
Izvor: izradio autor

Kako bi korisnik znao koju točnu vrijednost određena boja prikazuje, potrebno je dodati opis nakon selekcije miša.

```
var tooltipH = d3.select("#nacrtajGraf")
                .append("div")
                .style("opacity",0)
                .attr("class","tooltip")
                .style("background-color","white")
                .style("border","solid")
                .style("border-width","2px")
                .style("border-radius","5px")
                .style("padding","5px")
```

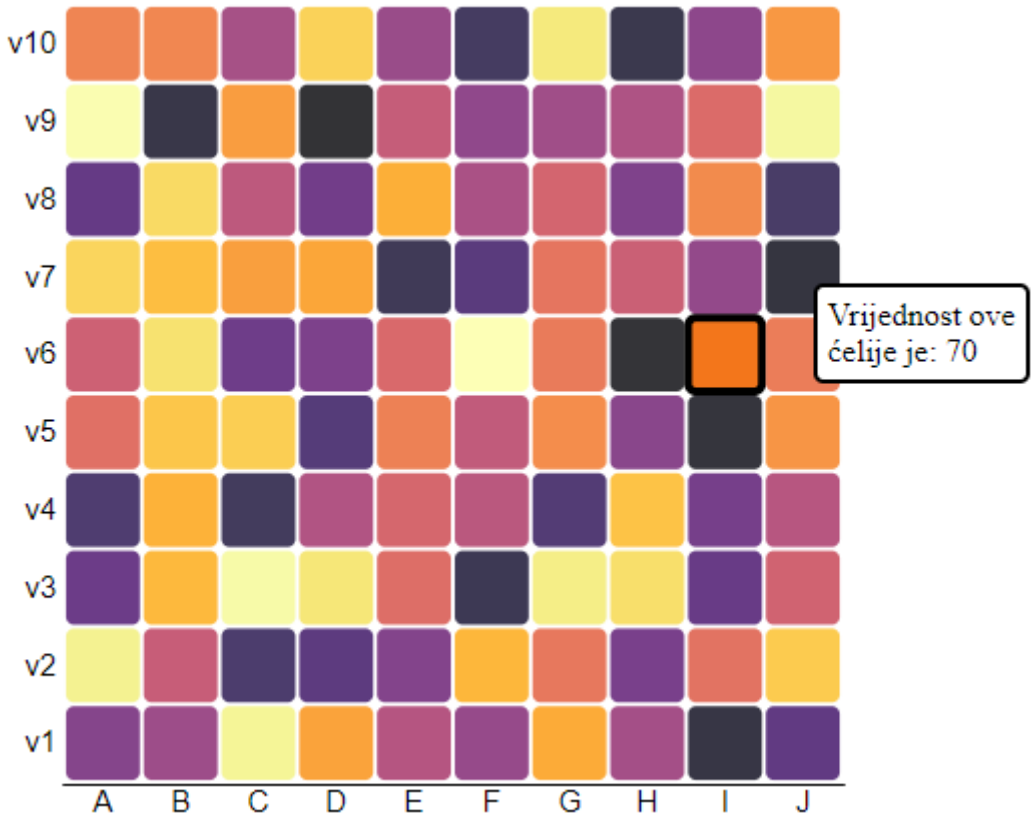
Slika 80 - Dodavanje opis ćelije u toplinsku kartu
Izvor: izradio autor

Dodjeljivanje div elementa u div identifikatora „#nacrtajGraf“ koji će prikazivati vrijednost.

```
var mouseover = function(d){
    tooltipH
        .style("opacity",1)
    d3.select(this)
        .style("stroke","black")
        .style("opacity",1)
}

var mousemove = function(d){
    tooltipH
        .html("Vrijednost ove <br>ćelije je: " + d[keysY[1]])
        .style("left", (d3.mouse(this)[0]+70) + "px")
        .style("top", (d3.mouse(this)[1]) + "px")
}

var mouseleave = function(d){
    tooltipH
        .style("opacity",0)
    d3.select(this)
        .style("stroke","none")
        .style("opacity",0.8)
}
```

Slika 81 - Reakcije na pokazivač miša
Izvor: izradio autor

Mouseover() funkcija se pokreće kada korisnik stavi pokazatelj miša na određenu ćeliju, *mousemove()* funkcija ispunjava tooltipH element sa vrijednostima ćelije te ju pozicionira pored pokazatelja miša, te *mouseleave()* funkcija sakriva tooltipH element. Navedene funkcije prikazane su na Slika 81.



Slika 82 - Toplinska karta sa opisom ćelije
Izvor: izradio autor

4.3.2. Frontend

3.2.3. React.js je poznat po odličnoj strukturiranosti projekta i velikoj lakoći u čitanju koda. Svaki projekt izrađen u 3.2.3. React.js razvojnom okviru, ako prati određena pravila, sastojati će se od jedne glavne .js datoteke nazivom Index.js u kojem se pozivaju sve komponente koji su potrebne.

client > src > index.js

```
1   import React from 'react';
2   import ReactDOM from 'react-dom';
3   import 'bootstrap/dist/css/bootstrap.min.css';
4   import App from './App.js';
5
6
7   ReactDOM.render(
8   <React.StrictMode>
9       <App />
10      </React.StrictMode>,
11      document.getElementById('root'));
12
```

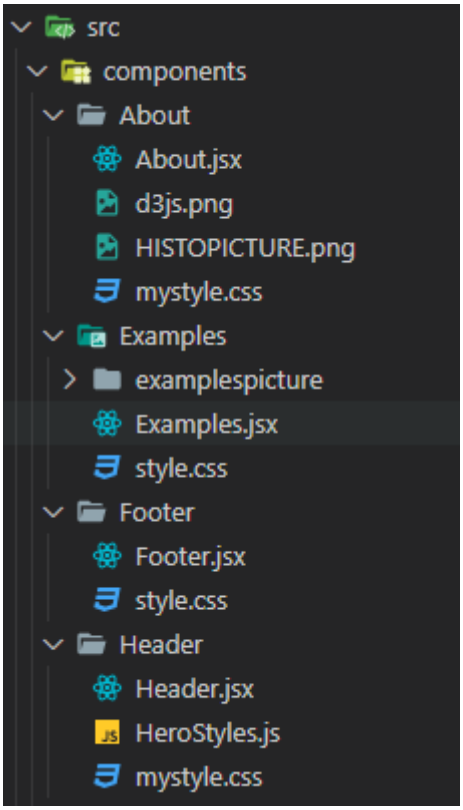
Slika 83 - Index.js datoteka
Izvor: izradio autor

Prema Slika 83, vidimo kako se poziva funkcija *ReactDOM.render()* koja se sastoji od komponenata koje želimo prikazati (App.js) i dokument u kojem želimo prikazati te komponente.

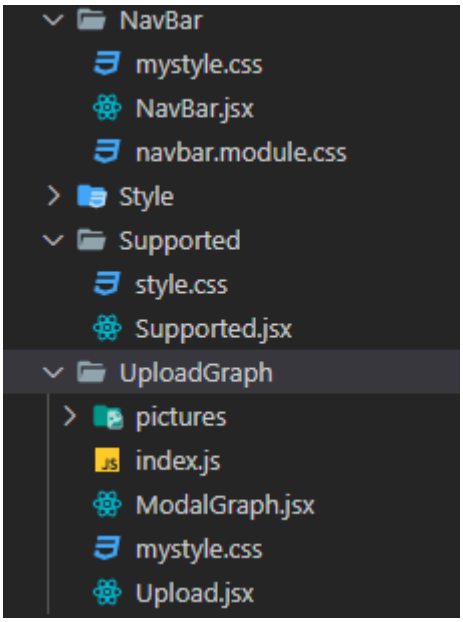
Preko App.js datoteke pozivamo sve komponente koje sami stvaramo i preko njih se vrši sva logika. U ovom primjeru prikazane su 6 komponenta koje se pozivaju kada pozovemo App.js datoteku, prikazano na Slika 84.

```
client > src > App.js > ...
1   import React from 'react';
2   import {  NavBar, Upload , About, Examples, Supported, Footer} from './components';
3   import 'bootstrap/dist/css/bootstrap.min.css';
4
5   const App = () => {
6       return (
7           <>
8               <NavBar/>
9               <Upload/>
10              <About/>
11              <Examples/>
12              <Supported/>
13              <Footer/>
14          </>
15      )
16  }
17
18  export default App;
```

Slika 84 - App.js datoteka
Izvor: izradio autor

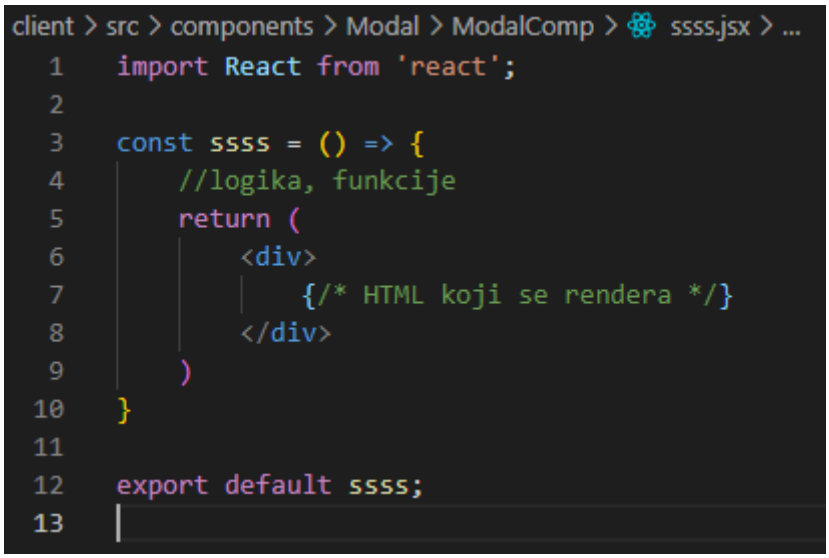


Slika 85 - Struktura datoteka ReactJS 1/2
Izvor: izradio autor



Slika 86 - Struktura datoteka ReactJS 2/2
Izvor: izradio autor

Svaka komponenta pozvana preko App.js datoteke zahtjeva 3.2.3. React.js komponente ili datoteke koji završavaju na .jsx. Kroz komponente mogu se slati varijable, vrijednosti, stanja koja se mogu koristiti u glavnoj komponenti poput App.js. Prikaz strukture datoteka vidljivo na Slika 86.



Slika 87- Početni izgled komponente
Izvor: izradio autor

Svaka .jsx komponenta se sastoji od funkcije koja u svom tijelu sadrži logiku potrebnu za rad komponente i završava sa *return()* funkcijom koja vraća HTML komponente. Izgled generičke komponente prikazan na Slika 87.

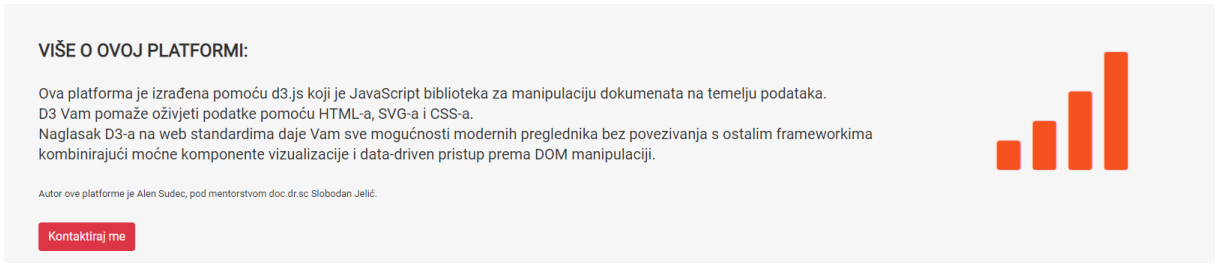
NavBar – Navigacijska traka

Navigacijska traka je stvorena kroz 3.2.1. HTML element <*nav*> te dodajući klase u njegove pod elemente dobije se lijepi animirani izgled. Uz korištenje jQuery biblioteke dodaje se animacija prilikom odabira navigacijskih dugmadi u traci. Izgled navigacijske trake vidljiv na Slika 88.

Slika 88 - Izgled navigacijske trake
Izvor: izradio autor

O platformi

Ova sekcija aplikacije objašnjava što je 2.3. D3.js biblioteka i kako se može koristiti. Uz paragraf teksta o 2.3. D3.js bibliotekabiblioteci, nalazi se dugme koji vodi na LinkedIn profil autora. Izgled komponente „O platformi“ prikazan na Slika 89.



Slika 89 - Izgled sekcije O platformi
Izvor: izradio autor

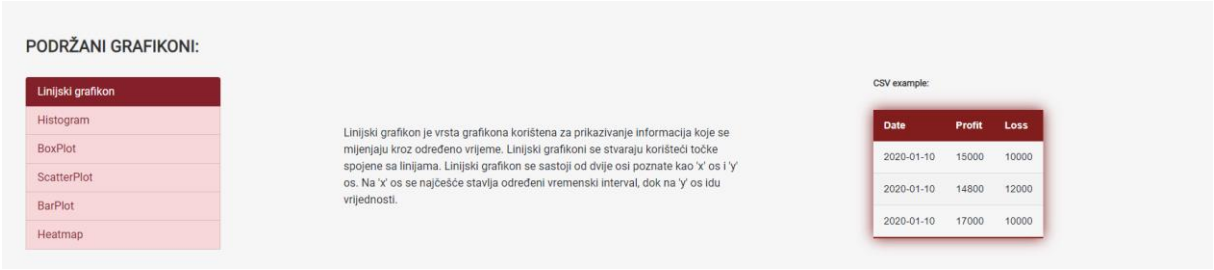
Primjeri grafikona

Ova komponenta sadrži HTML element *„Carousel“* koji prikazuje određene rotirajuće slike, prikazan na Slika 90. U ovom primjeru, slike prikazuju određene primjere grafikona koji su napravljeni kroz ovu internetsku aplikaciju.

Slika 90 - Izgled sekcije Primjeri
Izvor: izradio autor

Podržani grafikoni

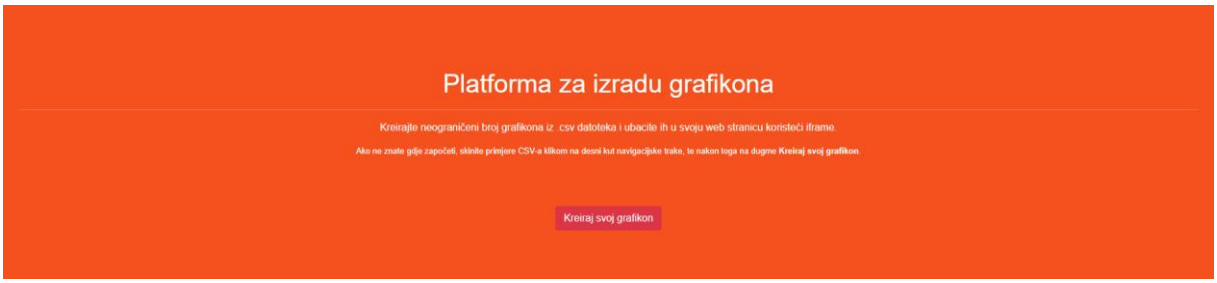
Nakon komponente „Primjeri grafikona“ prikazuje se komponenta koja sadržava sve podržane grafikone u aplikaciji u listi, objašnjenje grafikona i primjer .csv datoteke za odabrani grafikon. Izgled komponente „Podržani grafikoni“ vidljiv na Slika 91. Odabirom određenog grafikona u listi, mijenja se sadržaj paragrafa sukladno sa grafikonom, kao i sadržaj tablice koja prikazuje primjer .csv datoteke.



Slika 91 - Izgled sekcije Podržani grafikoni
Izvor: izradio autor

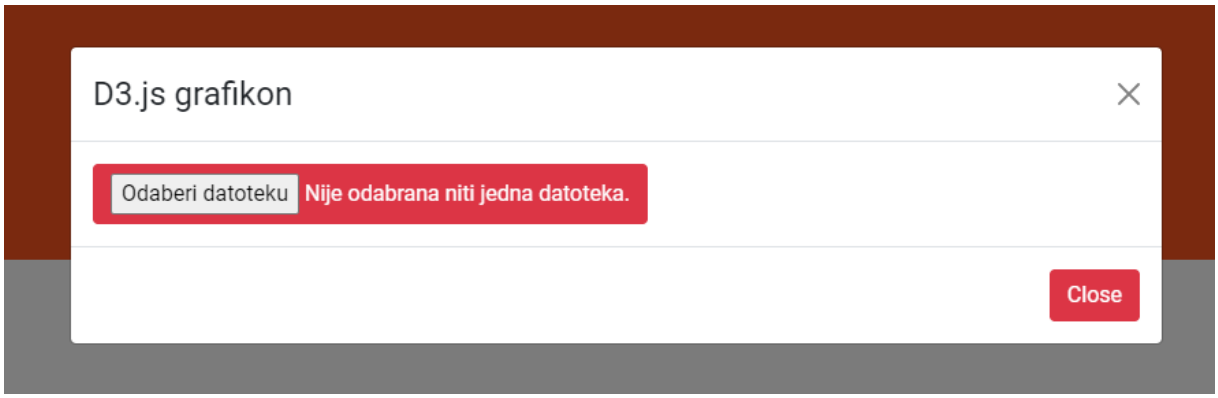
Zaglavlje i proces izrade grafikona

Komponenta koja sadrži naslov aplikacije, kratak uvod, informacije vezane za izradu grafikona i dugme koje prilikom klika pokreće proces izrade grafikona. Izgled komponente „Zaglavlje“ prikazan na Slika 92.



Slika 92 - Izgled zaglavlja
Izvor: izradio autor

Nakon klika na gumb, otvara se dijaloški okvir sa mogućnošću odabiranja datoteka. Jedini format datoteke koji je podržan je .csv datoteka. Podizanje .csv datoteke na poslužitelj prikazano na Slika 93.

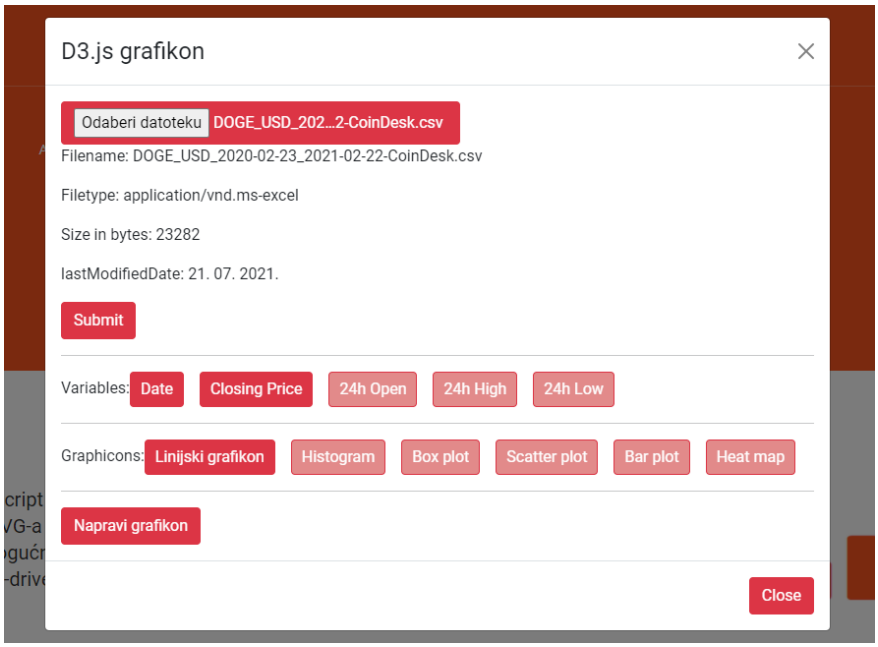


Slika 93 - Podizanje datoteke na poslužitelj 1/2
Izvor: izradio autor

Prilikom odabira datoteke, pojavljuje se više informacija o datoteci, poput ime datoteke, vrste datoteke, veličine datoteke i zadnji datum promjene datoteke kako bi korisnik bio siguran da je to datoteka koja je potrebna. Opis podignute datoteke prikazan na Slika 94.

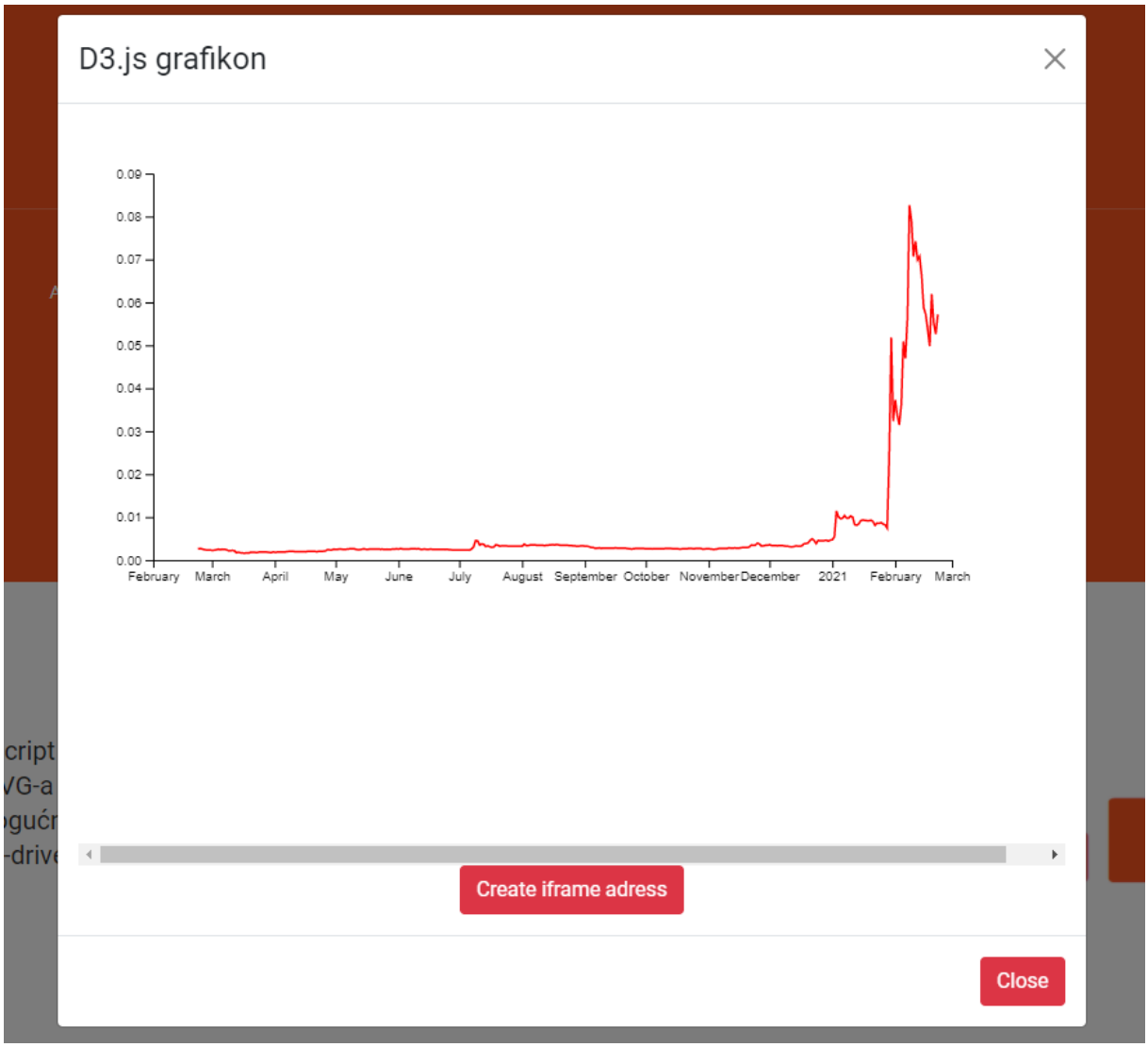
Slika 94 - Podizanje datoteke na poslužitelj 2/2
Izvor: izradio autor

Ako informacije odgovaraju korisniku, idući korak se odnosi na prijenos datoteke na poslužitelj klikom na „Submit“ dugme. Nakon klika, korisniku se prikazuju imena varijabli iz .csv datoteke, i mogući grafikoni. Potrebno je prvo odabrati varijablu koja se odnosi na x os, a nakon nje varijablu ili varijable koje se odnose na y os. Odabir varijabli i vrste grafikona prikazano na Slika 95.



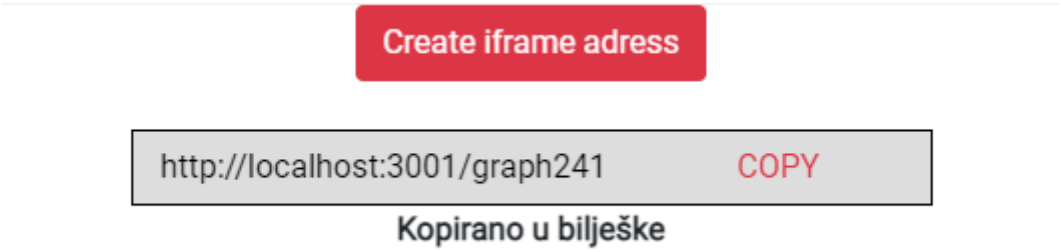
Slika 95 - Odabir varijabli i grafikona
Izvor: izradio autor

Nakon odabira datoteke, odabira varijabli i grafikona, klikom na gumb „Napravi grafikon“, prikazuje se određeni grafikon koji je stvoren kroz 4.3.1. Backend. Izgled linijskog grafikona prikazan na Slika 96.



Slika 96 - Izgled stvorenog grafikona
Izvor: izradio autor

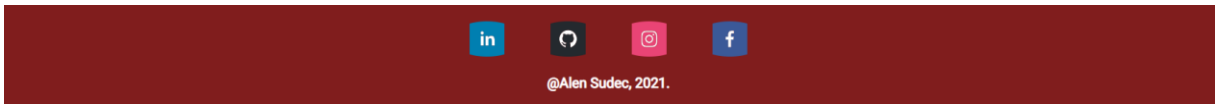
Ako je korisnik zadovoljan sa grafikonom, klikom na gumb „Create iframe adress“, otvara se kutijica sa linkom grafikona, te mogućnosti kopiranja u među spremnik za kasnije korištenje. Slika 97 prikazuje dobivenu poveznicu za grafikon.



Slika 97 - Dobivanje poveznice prema stvorenom grafikonu
Izvor: izradio autor

Podnožje

Na kraju aplikacije, prikazuje se komponenta „Podnožje“ koja sadrži slike na određene društvene mreže ili stranice. Klikom na određenu sliku, korisnika se vodi na stranice autora. Slika 98 prikazuje izgled podnožja.



Slika 98 - Izgled podnožja
Izvor: izradio autor

4.5. Opis funkcionalnosti aplikacije

Aplikacija omogućava korisnicima stvaranje različitih vrsta grafikona koji su stvoreni kroz korake podizanja .csv datoteke na poslužitelj i odabiranje varijabli i vrste grafikona kroz korisničko sučelje. Odabrani podaci su spremljeni u bazu podataka, kako bi se stvorila poveznica koja će korisnika voditi na njegov određeni grafikon. Nakon prikazanog grafikona, korisniku se nudi mogućnost prikazivanja poveznice. Koristeći određenu poveznicu, korisnik može prikazati grafikon na svojim poslovnim stranicama, koristeći <*iframe*> oznaku. Pomoću 2.3. D3.js bibliotekae, grafikoni imaju određene funkcionalnosti poput približavanja slike, prikazivanje određenih dijelova grafikona, itd.

4.6. Analiza upotrebljivosti aplikacije

Analiza i testiranje je odrađena ručnom provjerom funkcionalnosti i provjere ispravnosti aplikacije. Aplikacija je korištena od nekoliko korisnika sa različitim financijskim podacima. Prilikom testiranja korištena je metoda *Console.log()* koja šalje određeni tekst ili varijablu u konzolu, ovisno o tome da li se traži greška u korisničkom sučelju ili greška u poslužitelju. *Console.log()* metoda ubačena je u metode na koje se sumnjalo da postoji greška u radu. Nakon određenog vremena, nađene su male greške koje su popravljene. Na taj način završen je postupak testiranja prilikom kojeg su popravljene greške u aplikaciji.

Provedena je ponovna analiza i testiranje na aplikaciji nakon popravka prošlih grešaka u aplikaciji. Analiza i testiranje je ponovno rađena ručno provjerom funkcionalnosti i provjere ispravnosti aplikacije. Svaka komponenta i njezine korisnosti testirana je pojedinim klikom i praćenjem *Console.log()* metode. Aplikacija u trenutnom stanju izvršava sve funkcije za koje je namijenjena.

5. Rasprava

5.1. Prednosti i nedostatci D3.js biblioteke

Koristeći 2.3. D3.js bibliotekau, korisnici sa malo znanja o programiranju, mogu stvoriti interaktivne grafikone sa velikim brojem prilagodba kao što su boja, veličina, itd. Kao prednost nad ostalim bibliotekama koji imaju istu ili sličnu funkcionalnost, D3.js biblioteka je lagana na sistemskim resursima, pod pretpostavkom da skup podataka nije prevelik. Postoji mala krivulja učenja u izradi grafikona kroz 2.3. D3.js bibliotekau, ali je jednostavno iskoristiti 2.3. D3.js bibliotekau za ponovnu uporabu u više sličnih problema s malim promjenama. U dokumentaciji za 2.3. D3.js biblioteka biblioteku postoje mnogi primjeri (30+ grafikona) kako izraditi grafikone, dodati animacije, itd., pa se može reći da postoji povelika dokumentacija barem što se tiče izrade već postojećih grafikona. D3.js biblioteka koristi JSON i .csv formate kako bi se osigurala fleksibilnost podataka.

Jedan od važnijih nedostataka D3.js biblioteke je vezan za dokumentaciju, koristeći D3.js biblioteku teško je kreirati dinamično stvaranje grafikona iz bilo kojeg skupa podataka. Svaka dokumentacija vezana za D3.js se odnosi na stvaranje grafikona za jedan određeni skup podataka. Prilikom korištenja drugog skupa podataka za stvaranje istog grafikona, potrebno je mijenjati određene varijable i njihove funkcije. Ova aplikacija je uspjela približno stvoriti okružje gdje skup podataka nije direktno vezan za određeni grafikon, ali korisnik mora urediti skup podataka po pravilima kako bi se uspio kreirati grafikon. Koristeći velike skupove podataka, 2.3. D3.js biblioteka nije previše optimizirana, pa bi učitavanje i crtanje podataka moglo potrajati. Većina korisnika 2.3. D3.js bibliotekae govore kako ju je bilo teško koristiti i da nije stvorena za korisnike koji tek počinju sa internetskim razvojem.

5.2. Doprinos u izradi internetske aplikacije

2.3. D3.js biblioteka je biblioteka koja je stara 10 godina. Kroz vrijeme 2.3. D3.js biblioteka je postao jedan od projekata s otvorenim kodom s najviše zvjezdica na GitHub platformi, postao je sastavan dio svake velike tehnološke tvrtke, start-up tvrtke i dizajnerske tvrtke. Postao je de facto standard za izradu vizualizacije podataka na internetu. Zajednica je podijelila desetke tisuća kreativnih vizualizacija, koje mogu koristiti svi. Unutar 10 godina, biblioteka se razvila iz pojedinog repozitorija u fleksibilniji sustav modula. Dokumentacija se proširila na više od tisuću API metoda, a postoje stotine službenih primjera za demonstraciju tehnologije. Kako je biblioteka postala temelj za vizualizaciju internetskih podataka, bezbroj programera, dizajnera, analitičara podataka, biblioteka se počela nadograđivati. Ljudi su pisali udžbenike kako bi podijelili svoje znanje, neki su pisali knjige koje su postale polazište za mnoge druge programere, dizajnere i analitičare. (D3.js community, 2021)

6. Zaključak

Kako se broj poslovnih interakcija na internetu povećava iz dana u dan, potrebne su jednostavne tehnologije koje su u mogućnosti sa lakoćom i brzinom vizualizirati određene poslovne podatke koje se žele predstaviti široj publici. Prezentacija poslovnih podataka jedna je od važnijih čimbenika poslovnih interakcija, gdje ključna publika može s lakoćom pročitati i razumjeti podatke. Crtanje grafikona bez ikakvih dodataka može biti teško i za profesionalne programere, zbog toga izrađena je 2.3. D3.js biblioteka kako bi crtanje grafikona bilo moguće i za ljude koji nisu usavršili vještine programiranja. 2.3. D3.js biblioteka zahtjeva srednje znanje internetskog razvoja i 2.2. JavaScript programski jezik skriptnog jezika. Pomoću 2.3. D3.js biblioteka dokumentacije, svaki korisnik koji ima određenog iskustva u internetskom razvijanju aplikacija, mogu izraditi preko 30 vrsta grafikona.

Kako bi se prikazala korisnost 2.3. D3.js biblioteka biblioteke izrađena je internetska aplikacija koristeći alate poput 3.1.2. Balsamiq Wireframe, 3.1.1. Visual Studio Code i 3.1.3. Adobe Photoshop te tehnologije poput 3.2.3. React.js, 3.2.4. Node.js i 3.2.5. PostgreSQL-a. Cilj aplikacije je vezan za dinamičko stvaranje grafikona, kako bi korisnik morao samo odabrati određene parametre poput vrste grafikona i varijable koje se trebaju prikazati. Unutar

dokumentacije 2.3. D3.js biblioteka biblioteke, ne prikazuje se način stvaranja dinamičkih grafikona, nego stvaranje grafikona za jedan određeni skup podataka.

Nakon izrade aplikacije može se zaključiti kako je 2.3. D3.js biblioteka veoma koristan alat za vizualizaciju svih vrsta podataka, ne samo poslovnih podataka. Lakoća izrade grafikona je privukla bezbroj korisnika, koji su kroz godine još unapredili navedenu biblioteku. 2.3. D3.js biblioteka je savršen za crtanje grafikona za jedan određeni skup podataka. Kada se promijeni skup podataka, potrebno je mijenjati programski kod kako bi biblioteka prepoznala varijable. Korištenje 2.3. D3.js biblioteka biblioteke za dinamičko crtanje grafikona, potrebno je veliko znanje vezano za internetski razvoj aplikacija, manipulacijom nad skupovima podataka te prikazivanje skupa podataka unutar preglednika.

Literatura

1. Balsamiq Studios. (18. 08 2021). *Balsamiq Wireframes*. Dohvaćeno iz Balsamiq Wireframes: https://balsamiq.com/wireframes/

2. Bostock, M. (18. 08 2021). *Data-Driven Documents*. Dohvaćeno iz d3js.org: d3js.org

3. D3.js community. (25. 08 2021). *Powering the D3 Community*. Dohvaćeno iz Observable: https://observablehq.com/@d3/powering-the-d3-community

4. Devsaran. (18. 08 2021). *From History of Web Application Development*. Dohvaćeno iz Devsaran: https://www.devsaran.com/blog/history-web-application-development

5. Kanjilal, J. (18. 08 2021). *Visual Studio Code: A fast, lightweight, cross-platform code editor*. Dohvaćeno iz Infoworld: https://www.infoworld.com/article/2919555/microsoft-net/visual-studio-code-a-fast-lightweight-and-cross-platform-code-editor.html

6. Manovich, L. (18. 08 2021). *Inside Photoshop*. Dohvaćeno iz Computational Culture: http://computationalculture.net/inside-photoshop/

7. Microsoft. (18. 08 2021). *Extension API*. Dohvaćeno iz Visual Studio Code: https://code.visualstudio.com/api

8. Mozzila. (18. 08 2021). *About JavaScript*. Dohvaćeno iz MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript

9. Mozzila. (18. 08 2021). *CSS: Cascading Style Sheets*. Dohvaćeno iz MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/CSS

10. Mozzila. (18. 08 2021). *HTML: HyperText Markup Language*. Dohvaćeno iz MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/HTML

11. Naik, S., & Wickramasinghe, S. (18. 08 2021). *Most Compatible Frontend and Backend Framework Pairings*. Dohvaćeno iz crowdbiotics: https://blog.crowdbotics.com/most-compatible-frontend-backend-framework-pairings/

12. PostgreSQL Tutorial. (18. 08 2021). *What is PostgreSQL*. Dohvaćeno iz PostgreSQL Tutorial: https://www.postgresqltutorial.com/what-is-postgresql/

13. training.com. (18. 08 2021). *About Node.js, and why you should add Node.js to your skillset*. Dohvaćeno iz training.com: http://blog.training.com/2016/09/about-nodejs-and-why-you-should-add.html

Popis slika