ReactJS JavaScript programska zbirka

Tomašić, Karlo

Sveučilište
Sjever

Završni rad br. 800/MM/2022

ReactJS - JavaScript programska zbirka

Karlo Tomašić, 2910/336

Varaždin, rujan 2022. Godine

Sveučilište
Sjever

Odjel za Multimediju, oblikovanje i primjenu

Završni rad br. 800/MM/2022

ReactJS - JavaScript programska zbirka

Student
Karlo Tomašić, 2910/336

Mentor
Vladimir Stanisavljević, mr. sc.

Varaždin, rujan 2022.

Sveučilište Sjever
Sveučilišni centar Varaždin
104. brigade 3, HR–42000 Varaždin

UNIVERSITY NORTH

Prijava završnog rada

Definiranje teme završnog rada i povjerenstva

ODJEL Odjel za multimediju

STUDIJ preddiplomski stručni studij Multimedija, oblikovanje i primjena

PRISTUPNIK Karlo Tomašić | JMBAG 2910/336

DATUM 08.09.2022. | KOLEGIJ Programski alati 3

NASLOV RADA ReactJS JavaScript programska zbirka

NASLOV RADA NA ENGL. JEZIKU ReactJS JavaScript library

MENTOR Vladimir Stanisavljević | ZVANJE viši predavač

ČLANOVI POVJERENSTVA

1. doc.dr.sc. Andrija Bernik - predsjednik povjerenstva
2. pred. Dražen Crčić, dipl.ing. - član povjerenstva
3. mr.sc. Vladimir Stanisavljević, v.pred. - mentor
4. doc.dr. sc. Domagoj Frank - zamjenski član
5.

Zadatak završnog rada

BROJ 800/MM/2022

OPIS

ReactJS, jedna je od popularnih web tehnologija na temelju JavaScripta za razvoj korisničkih sučelja web aplikacija. Pri dizajnu ReactJS-a posebna pozornost posvećena je performansama aplikacije, posebice u interakciji s objektnim modelom dokumenta (engl. document object model - DOM). Aplikacije se prvenstveno grade kao jedno-stranične (engl. single page application), a pri modeliranju i izradi programskih rješenja poželjeno je da podržavaju MVC (engl. model-view-controler) ili neke slične oblikovne obrasce.

U ovom radu potrebno je:
* opisati povijesni razvoj i osnove ReactJS programske zbirke te njezinu primejnu za izradu korisničkog sučelja web aplikacija kroz interakciju s DOM-om,
* analizirati prednosti i ograničenja ReactJS-a i usproediti ga sa sličnim tehnologijama,
* opisati jednostraničnu i MVC arhitekturu izrade aplikacija i kako je ona podržana u ReactJS-u,
* opisati postupak instalacije i načine ugrađivanja u aplikacije,
* detaljno obraditi mogućnosti i sintaksu ReactJS-a te na primjerima pokazati osnove izgradnje korisničkih aplikacija,
* osmisliti i oblikovati demo web aplikaciju na kojoj će biti demonstrirane glavne mogućnosti ReactJS-a,

Detaljno dokumentirati sve korištene tehnologije i izrađeni programski kod te opisati stečena iskustva i postignute rezultate.

ZADATAK URUČEN 23.09.2022 | POTPIS MENTORA Vlados

SVEUČILIŠTE SJEVER

Predgovor

Glavna svrha ovog završnog rada jest proučiti *front-end* programsku zbirku za izradu web stranica/aplikacija, baziranu na JavaScriptu. Trenutno ima mnogo programskih zbirki i razvojnih okvira za izradu *front-end* aplikacija i web stranica. Programska zbirka ReactJS, jedna od najnovijih web tehnologija je među njima. Dokazano je da ReactJS ima jedno od najbržih renderiranja među programskim zbirkama. On se fokusira na "View" dio MVC oblikovnog obrasca (engl. mode-view-controller), te ima široku skalabilnost u razvoju web stranica i web aplikacija.

Razvijen od strane Facebook-a za internu upotrebu, pokazao se kao učinkovita i brza programska zbirka u odnosu na druge tehnologije. Međutim, tek je kasnije pušten za otvorenu upotrebu te obogaćen sa sve više funkcionalnosti od strane mnogih suradnika. Kad je u pitanju veliki broj podataka i korisnika, pokazao se veoma uspješnim u pružanju boljeg iskustva korisnika. Uz Facebook, mnoge druge velike organizacije koriste ReactJS i React Native za vlastiti razvoj. Instagram, Netflix i Airbnb su samo neka od velikih imena koje besprijekorno služe enormnom broju korisnika u cijelom svijetu.

Kroz istraživanje, glavni cilj je bio procijeniti te prikazati ReactJS kao kompatibilnu platformu za razvoj web stranica i web aplikacija, u vremenu kada ima mnogo opcija za izabrati. U ovom će radu biti prodiskutirane osnove, arhitektura, značajke, popularnost te prilagodljivost.

Sažetak

Namjera ovog završnog rada o ReactJS-u jest bila istražiti značajke, pobliže saznati što sve nudi, osnovne koncepte arhitekture, kako je drugačiji od ostalih programskih zbirki i razvojnih okvira, procese rukovanja s velikim brojem podataka te ostale funkcionalnosti. Razlog odabira ReactJS-a jest bio bolje upoznavanje sa samom programskom zbirkom, a i pozitivna iskustva s istim prilikom korištenja.

Kroz istraživanje, ustanovljeno je da je ReactJS manje kompliciranа programska zbirka u usporedbi s ostalima. Veoma je agilna u razvoju, te je brzo renderirajuća programska zbirka. Učenje, a i savladavanje ReactJS-a oduzima manje vremena u odnosu na druge iz razloga jer je to samo programska zbirka, a ne razvojni okvir. Uobičajeno, treba više vremena kako bi savladali ostale razvojne okvire te naučili njihovu terminologiju.

Osim toga, glavni koncept ReactJS-a su komponente. Sve što vidimo na *front-endu* je ništa više od običnih komponenti. Korisničko sučelje je jednostavno kolekcija komponenti. Promjene u jednoj komponenti, ne utječu nužno na druge komponente. Zato je ažuriranje aplikacije tijekom promjene u podacima olakšano i manje kompleksno. Za izvođenje ove funkcionalnosti React uvodi modificirani koncept objektnog modela dokumenta (engl. document object model - DOM), virtualni DOM.

Jednosmjerni protok podataka prilikom upravljanja istima je još jedna prednost Reacta. Podaci mogu biti promijenjeni s bilo koje točke u aplikaciji. S obzirom na to da podaci teku u jednom smjeru, React nam nudi stabilnu kontrolu stanja aplikacije. Dodatak nove sintakse, JSX, se također pokazao kao odličan dodatak brzom renderiranju.

Ključne riječi: ReactJS, programska zbirka, komponente, virtualni DOM, protok podataka

Summary

The intention of this final paper about ReactJS is to explore the features, learn more about what it offers, the basic concepts of architecture, how it's different from other libraries and frameworks, the process of handling a large amount of data and other functionalities. The reason for choosing ReactJS was to better familiarize myself with the library, as well as positive experience with its prior usage.

While exploring ReactJS, conclusion was made that it is a much easier library to learn compared to others. It is a very agile and fast-rendering library in development. Learning and mastering ReactJS takes less time compared to others because it is just a library and not a framework. Usually, it takes more time to master frameworks and learn their terminology.

Apart from that, the main concept of ReactJS are components. Everything we see on the front-end is nothing more than ordinary components. A user interface is simply a collection of components. Changes in one component do not necessarily affect other components. That's why updating apps during data changes is easier and less complex. This is possible because React uses its own concept of DOM, the React virtual DOM.

One-way flow of data when it is being managed is another advantage of React. Data can be changed at any point in the application. Giver that data flows in one direction, React offers stable control over the state of the application. The addition of a new syntax, JSX, also proved to be a great addition to fast rendering.

Keywords: ReactJS, library, components, virtual DOM, data flow

Popis korištenih kratica

DOM	Document Object Model
PHP	Hypertext Preprocessor
XHP	Hack + PHP
JSX	JavaScript XML
XML	Extensible Markup Language
MVC	Model - View - Controller oblikovni obrazac
HTML	Hyper Text Markup Language
ECMAScript	JavaScript standard
SPA	Single Page Aplikacija
SEO	Search Engine Optimization
CRA	Create React App
NPM	Node Package Manager
API	Application Programming Interface
Json	JavaScript Object Notation
ISBN	International Standard Book Number

Sadržaj

1. Uvod

Internet je postao užurbano središte za pretraživanje informacija i za obavljanje različitih zadataka koji su se prije internetskog doba obavljali ručno. Postoji ogroman broj mobilnih i web aplikacija koje su nam olakšale obavljanje različitih zadataka. Velik dio naših svakodnevnih obaveza možemo obaviti preko interneta u sadašnjem vremenu. Brži internet u kombinaciji s uređajima s visokim performansama također zahtijeva brže aplikacije.

Rapidno raste trend prebacivanja softvera ili aplikacija, koje smo prethodno koristili na stolnim računalima, na internet. Postoji mnogo aplikacija koje se mogu koristiti s interneta, a i s mobilnih uređaja. Nekoliko JavaScript razvojnih okvira i programskih zbirki koristi se za razvoj različitih vrsta aplikacija. U ovom trenutku postoje ReactJS, AngularJS, VueJS te mnogi drugi. ReactJS je jedan od njih za razvoj *front-end* dijela aplikacije.

React je popularna programska zbirka otvorenog izvora koju je razvio Facebook. React je široko popularan među zajednicama programera zbog svoje jednostavnosti i lakog, ali i učinkovitog procesa razvoja. React olakšava stvaranje interaktivnih korisničkih sučelja. Učinkovito se ažurira kroz renderiranje komponenti pa do prikaza svakog stanja aplikacije i svake promjene podataka u samoj aplikaciji.

U ReactJS-u, svaka komponenta upravlja vlastitim stanjem i sastavlja se prema korisničkom sučelju. Zbog ovog koncepta komponenti umjesto predložaka u JavaScriptu, mnogo podataka se može lako proslijediti kroz aplikaciju i tako držati stanje komponenti izvan DOM-a. Korištenjem NodeJS-a uz React se aplikacija također može prikazati na strani poslužitelja. Uz web aplikacije, za izradu mobilnih aplikacija možemo koristiti i React Native.

Svrha ovog završnog rada jest provesti dubinsko istraživanje ReactJS programske zbirke temeljene na JavaScriptu. U radu će biti obrađeni temeljni koncepti, karakteristike, značajke, procesi razvoja, temeljna arhitektura, istraživanje tržišta kao i kompatibilnost. Cilj je pružiti solidno razumijevanje ReactJS programske zbirke.

2. Povijesni razvoj ReactJS-a

ReactJS je stvorio Jordan Walke, softverski inženjer u Facebooku, koji je objavio rani prototip ReactJS-a pod nazivom “FaxJS”. Nastao je pod utjecajem XHP-a, programske zbirke HTML komponenti za PHP. Prvi put je objavljen na Facebook stranici novosti (engl. news feed) 2011. te kasnije na Instagramu u 2012. godini.

React Native, koji omogućuje razvoj aplikacija za Android i iOS, najavljen je na “React Conf-u” od Facebooka u veljači 2015., a pušten je u produkciju u ožujku 2015. godine.

18. travnja 2017. godine Facebook je najavio React Fiber, novi skup internih algoritama za renderiranje, za razliku od Reactovog starog algoritma za renderiranje koji se zvao Stack. React Fiber je postao temelj svih budućih poboljšanja i razvoja značajki React programske zbirke. Promjenom ovih algoritama, stvarna sintaksa ReactJS-a se ne mijenja; promijenio se samo način na koji se sintaksa izvodi. Stari sustav renderiranja Reacta, Stack, razvijen je u vrijeme kada dinamičke promjene sustava nisu bile u potpunosti shvaćene. Stack je bio spor u izvođenju složenih animacija iz razloga jer je sve to pokušao izvesti odjednom. React Fiber rastavlja animaciju na segmente koji se mogu rasporediti na više okvira. Isto tako, struktura stranice može se podijeliti na segmente koji se mogu zasebno održavati i ažurirati. JavaScript funkcije i objekti virtualnog DOM-a nazivaju se “vlakna”, a svakim od tih “vlakana” se može zasebno upravljati, što omogućuje glatkije renderiranje.

3. Zašto naučiti ReactJS

ReactJS je svijetu predstavljen 2013. godine i od tada je doživio impresivan rast, kako unutar tako i izvan Facebooka. Novi web projekti na Facebooku obično se izrađuju pomoću ReactJS-a u različitim oblicima, i široko se prihvaća u cijeloj industriji. Programeri i inženjeri biraju ReactJS jer omogućuje veći utrošak vremena na razvoj proizvoda, a manje trošenja vremena na učenje i borbu s razvojnim okvirom.

ReactJS aplikacija je zbirka diskretnih komponenti, od kojih svaka predstavlja jedan "Pogled" (engl. View) aplikacije. Ideja svake individualne komponente kao "Pogleda" olakšava fokus na razvoj produkta jer za promjene na jednom "Pogledu" ili komponenti nije potrebno uzeti u obzir kompletni sustav. Kada je aplikacija izrađena s ReactJS-om, kod je generalno predvidljiv, jer ReactJS "omota" DOM i pojednostavljuje model programiranja. Štoviše, aplikacija izgrađena s ReactJS-om je više skalabilna.

Kombinacija Reacta i brzog ciklusa iteracije weba, omogućena je izrada vrhunskih proizvoda uključujući mnoge Facebook komponente. JavaScript razvojni okvir zvan Relay, koji je također napravljen na temelju ReactJS-a, pojednostavljuje dohvaćanje podataka u velikim razmjerima.

3.1. Kratka i jednostavna krivulja učenja

Za razliku od nekih drugih JavaScript programskih zbirki i razvojnih okvira, za koje je potrebno puno vremena za učenje samog razvojnog okvira, u ReactJS-u nije potrebno previše truda za sami početak izrade aplikacije. React se sastoji od velikog broja značajki. Čitljivost koda je jedna od najvećih prednosti ReactJS-a. Lako je čitljiv čak i onima kojima nije poznat. Dok druge programske zbirke i razvojni okviri zahtijevaju učenje raznih koncepata o samoj programskoj zbirci ili razvojnom okviru, zanemarujući osnove jezika, ReactJS čini potpuno suprotno.

3.2. React je brz i agilan

ReactJS se odlikuje jednosmjernim protokom podataka između stanja, te između slojeva aplikacije. To znači da podaci teku samo u jednom smjeru između stanja aplikacije i slojeva. U dvosmjernom povezivanju podataka poput AngularJS-a, ako se model aplikacije promijeni, "Pogled" aplikacije se također mijenja i obrnuto. ReactJS puno brže renderira ažuriranja u DOM-u nego alternativni razvojni okviri, uzevši u obzir da je ReactJS puno manja programska zbirka. Stoga, lako je odabrati alate koji će kvalitetno odraditi posao.

3.3. React je predstavio JSX

JSX je sintaksa slična XML-u/HTML-u koju koristi ReactJS. JSX omogućuje da tekst sličan XML-u/HTML-u može surađivati s JavaScript/ReactJS kodom. Pretprocesori(pretvarači poput Babela) pretvaraju tekst nalik HTML-u, koji je smješten u JavaScript datotekama, u obične JavaScript objekte koje data JavaScript engine parsira.

U osnovi, korištenjem JSX-a, moguće je napisati sažete strukture nalik HTML/XML-u (npr. strukture poput DOM-a) u istoj datoteci u kojoj je pisan JavaScript kod. Tada će Babel transformirati te izraze u pravi JavaScript kod. U odnosu na prošlost, kada se JavaScript stavljao u HTML, JSX nam omogućuje da stavimo HTML u JavaScript. Koristeći JSX, može biti napisan sljedeći JSX/JavaScript kod:

```
let nav = (
  <ul class="nav">
    <li><a href="#">Početna</a></li>
    <li><a href="#">O nama</a></li>
    <li><a href="#">Kontakt</a></li>
  </ul>
)
```

Programski kod 2.1 Kod za navigaciju u običnom JavaScript kodu

Babel će to transformirati u:

```
let nav = React.createElement(
  "ul",
  { class: "nav" },
  React.createElement(
    "li",
    null,
    React.createElement(
      "a",
      { href: "#" },
      "Početna"
    )
  ),
  React.createElement(
    "li",
    null,
    React.createElement(
      "a",
      { href: "#" },
      "O nama"
    )
  ),
  React.createElement(
    "li",
    null,
    React.createElement(
      "a",
      { href: "#" },
      "Kontakt"
    )
  )
);
```

Programski kod 2.2 Kod za navigaciju u JSX kodu

JSX pruža saženu i prepoznatljivu sintaksu za definiranje strukture DOM-a s atributima koja ne zahtijeva napuštanje samog JavaScripta. I jedan i drugi argument može biti velika prednost pri izradi velikih aplikacija.

3.4. Velika razvojna zajednica

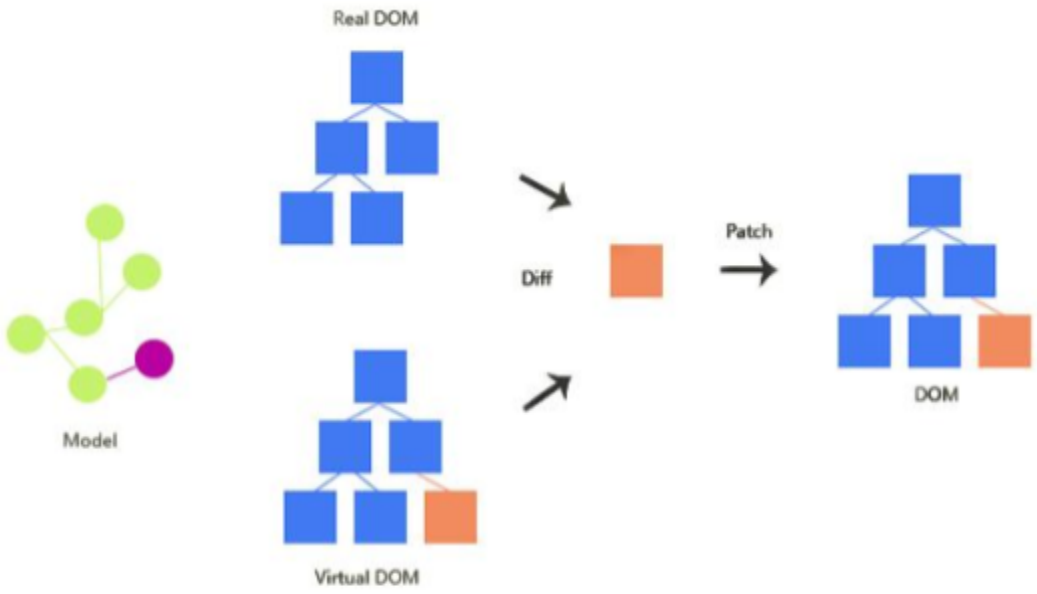
Velike kompanije poput New York Times, Airbnb, Facebook i Netflix koriste ReactJS u produkciji. Oni kontinuirano pridonose razvoju jezgre Reacta te u izgradnji korisnih i popularnih vanjskih programskih zbirki koje izvrsno rade s bilo kojom ReactJS aplikacijom. ReactJS ima jednu od najvećih zajednica dizajnera i programera od bilo kojeg programskog jezika danas. Također je otvorenog koda, što znači da svatko može preuzeti ReactJS kod, mijenjati ga, poboljšati ga te dijeliti s drugima kako bi se sama programska zbirka kontinuirano poboljšavala.

4. Arhitektura jezgre Reacta

4.1. React Virtualni DOM

DOM predstavlja objektni model dokumenta (engl. document object model). Manipulacija samim DOM-om je vrlo važna za moderne interaktivne web tehnologije. Često se i naziva srcem modernog weba, što je apstrakcija strukturiranog teksta. Negativna strana jest da radi sporije od ostalih naredbi i metoda u JavaScriptu jer većina JavaScript razvojnih okvira najčešće ažurira sami DOM čak i ako to nije potrebno raditi. To znači da ta ažuriranja nisu nužna za izvođenje određenih radnji, ali one i dalje rade prema zadanim postavkama. Na primjer, devet je stavki stavljeno u košaricu u online web trgovini. Ako je potrebno kupiti i naplatiti samo prvu stavku s liste, ovdje bi većina tehnologija ponovno izgradila cijeli popis koji je stavljen u košaricu. To znači da sam razvojni okvir mora nepotrebno raditi deset puta više. Zbog samo jedne promjene sustav mora opet izgraditi listu kako bi bila ista kao i u prvotnom stanju.

ReactJS nije izmislio Virtualni DOM, već ga samo koristi i pruža zajednici programera potpuno besplatno. Virtualni DOM je jednostavno apstrakcija HTML objektnog modela dokumenta. ReactJS ima odgovarajući Virtualni DOM objekt za svaki HTML DOM objekt poput nezahtjevne i lagane kopije. Virtualni DOM je također karakteriziran sličnim svojstvima kao i pravi DOM. Međutim, on ne može raditi nikakve promjene direktno na “Pogledu”. Manipulacija DOM-om je poprilično spor proces. Dok s druge strane, manipuliranje virtualnim DOM-om je mnogo brže jer nema nikakve veze s “Pogled“ dijelom aplikacije te ne vrši nikakve promjene na ekranu. U sljedećoj se slici se vidi ilustracija Virtualnog DOM-a u memoriji.



Slika 3.1 Prezentacija React Virtualnog DOM-a u memoriji

Kao što je prikazano na slici 3.1, ReactJS Virtualni DOM u memoriji je lagana kopija pravog DOM-a. React koristi metodu zvanu *diffing* što znači da renderiranje JSX elementa ažurira svaki pojedinačni dio Virtualnog DOM-a. Ovo možda zvuči neučinkovito, ali zapravo ne utječe nikako jer se Virtualni DOM prilično brzo ažurira, te nema nikakav utjecaj na proces. Nakon što se DOM ažurira, ReactJS uspoređuje ažurirani DOM s prethodnim stanjem DOM-a i određuje koja je stavka Virtualnog DOM-a promijenjena. Nakon što ReactJS otkrije sve promjene, tada ReactJS ažurira samo te objekte na stvarnom DOM-u.

Dakle, React ubrzava ažuriranje kroz Virtualni DOM. U prethodno spomenutom primjeru, ReactJS bi ažurirao samo označenu stavku s popisa, dok ostale stavke s liste ne bi dotaknuo. To čini veliku razliku pri ažuriranju stranice u aplikaciji jer ReactJS može mijenjati samo potrebne dijelove DOM-a. Ovaj proces manipuliranja virtualnim DOM-om jedan je od glavnih razloga zašto ReactJS dobiva na popularnosti među programerskom zajednicom.

4.1.1. Prednosti i nedostatci React Virtualnog DOM-a

Među brojnim prednostima React Virtualnog DOM-a, nekoliko ključnih prednosti jest ovdje opisano:

- "Diffing" algoritmi napisani u ReactJS-u su poprilično brzi i učinkoviti
- Uključenje JSX-a omogućuje izgradnju više *front-enda* za istu aplikaciju
- Vrlo je lagan i sposoban za pokretanje na svakom mobilnom uređaju
- Može se koristiti i samostalno bez ReactJS-a

Uz prednosti naravno da ima i par nedostataka:

- Zauzima priličnu količinu memorije, jer se u memoriji također osim DOM-a, nalazi i kopija DOM-a
- Statički elementi ne donose preveliku razliku

4.2. Jednosmjerni protok podataka

Razvojni okviri poput AngularaJS-a i EmberJS-a koriste dvosmjerno povezivanje podataka. U dvosmjernom povezivanju podataka, na primjer u AngularJS-u, ako se "Model" (engl. Model) aplikacije promijeni, također se i "Pogled" dio aplikacije automatski mijenja i obrnuto. Polje za unos u modelu također može utjecati na sami model. To dobro radi u većini aplikacija, ali ponekad može dovesti do kaskadnih ažuriranja te promjena u jednom modelu može uzrokovati ažuriranja na drugim modelima. Također, budući da stanje aplikacije mogu mijenjati i "Pogled" i "Kontroler" (engl. Controller), protok podataka može biti nepredvidljiv u nekim slučajevima. Flux ili Redux s ReactJS-om mogu biti rješenje za izbjegavanje tih nesigurnosti budući da obje arhitekture slijede jednosmjerni protok podataka. Jednosmjerni protok podataka ne čini kaskadna ažuriranja i promjene u "Pogled" dijelu aplikacije.

Jednosmjerni protok podataka osigurava protok podataka kroz aplikaciju u samo jednom smjeru kako bi se osigurala veća kontrola između stanja i modela aplikacije, te također čini arhitekturu manje kompliciranom i razumljivijom. Flux arhitektura je funkcionalni pristup. Ovdje se "Pogled" razmatra kao funkcija stanja aplikacije. Na kraju, ako se stanje promijeni, "Pogled" se također automatski re-renderira. Štoviše, slični "Pogled" se generira iz samog stanja i daje bolje razumijevanje i predvidljivost aplikacije.

Kako bi bilo što predvidljivije, u aplikaciji, podaci od Roditelj komponente do Dječje komponente teku u jednom smjeru. Svi podaci se mogu ažurirati iz bilo kojeg "Pogleda", u bilo koje vrijeme s ovim pristupom. U slučaju da nešto krene po zlu, otklanjanje pogrešaka je također manje kompliciranо na ovaj način.

4.3. React komponente

Komponente su veoma važne u ReactJS-u. Često se smatra srcem ReactJS-a, što čini skup svih komponenti. Komponenta je mali element korisničkog sučelja, s mogućnošću višekratne upotrebe, koji pruža podatke "Pogled" dijelu aplikacije i mijenja se tijekom vremena. Za stvaranje cijelog korisničkog sučelja, te male komponente su sastavljene zajedno, ugniježđene jedna unutar druge. Komponente omogućuju korisničko sučelje podijeliti na male dijelove te dizajnirati i izgraditi na sveobuhvatan način. Korisničko sučelje (engl. user interface - UI) je ono što se prikazuje na ekranu. Komponente su kao JavaScript funkcije. Doslovno obavljaju isti zadatak, ali u različitom okruženja i s različitim pristupom. Poput funkcija, komponente primaju vrijednosti koje se nazivaju "Props" i vraćaju ReactJS elemente. Ti elementi opisuju što korisnik vidi u sučelju na zaslonu. ReactJS komponente se mogu koristiti za izgradnju nekih dijelova sučelja, pa čak i za izgradnju kompletnog sučelja.

4.3.1. Izrada React komponente

React komponenta se može jednostavno napisati kao JavaScript funkcija. Ta funkcija prihvaća "propse", te vraća ReactJS element. One se nazivaju funkcionalnim komponentama. ES6 klase se također mogu koristiti za definiranje komponenta.

```
function Greeting(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

Programski kod 3.1 JavaScript funkcija koja vraća naslov razine 1

Koristeći arrow funkciju:

```
const Greeting = props => <h1>Hello, {props.name}</h1>
```

Programski kod 3.2 JavaScript funkcija koja vraća naslov razine 1 napisana koristeći ES6 standard

React komponenta se također može izgraditi na nekoliko drugih načina. Moguće je naslijediti ili izvesti klasu iz glavne komponente koja je pripojena objektu.

```
class Greeting extends React.Component {
  render() {
    const { name } = this.props;
    return (
        <h1>Hello, {name}</h1>
    )
  }
}
```

Programski kod 3.3 ReactJS komponenta koja vraća naslov razine 1 napisana kao klasna komponenta

4.4. ReactJS u Single Page aplikacijama

Jednostranična aplikacija(engl. single page application - SPA) je web aplikacija koja ne koristi zadanu metodu potpunog učitavanja novih stranica. Umjesto toga, preuzima nove podatke s web poslužitelja prilikom interakcije s web preglednikom i osvježava samo potrebne dijelove trenutne web stranice. HTML5 i AJAX koriste se za izradu responzivnih aplikacija, a za rukovanje velikim prometom na strani klijenta koriste se JavaScript razvojni okviri i programske zbirke kao što su AngularJS, ReactJS, VueJS i drugi. SPA pruža veliki raspon interakcija i glatkih efekata, koji pružaju izvrsno korisničko iskustvo. Od korištenja manje prometa web stranica do brzog vremena učitavanja, SPA su postale vitalne na današnjem tržištu.

Dok su prethodni JavaScript okviri poput jQuery-a izravno utjecali na HTML dokumente (DOM) kada je korisnik pokušao komunicirati s web aplikacijom, rezultirajući lošim performansama, ReactJS upravo zbog toga radi s virtualnom reprezentacijom DOM-a, pa zato, kada korisnik komunicira s aplikacijom, operacije se izvode na virtualnom DOM-u i promjene se izravno prikazuju na ekranu, a HTML dokumenti ostaju netaknuti, čime se postižu visoke performanse aplikacija.

5. React protiv ostalih razvojnih okvira

Na tržištu postoji mnogo JavaScript razvojni okvira za razvoj *front-end* aplikacija. Ipak, ReactJS nije razvojni okvir poput AngularJS-a. S obzirom na to da je ReactJS JavaScript programska zbirka, poprilično je kompatibilan kada je u pitanju razvoj web aplikacija u odnosu na popularne JavaScript razvojne okvire.

5.1. Usporedba

Usporedba između AngularJS-a i ReactJS-a danas je popularna tema u programerskoj zajednici. Osim ReactJS-a, također popularne web tehnologije su AngularJS, VueJS, EmberJS, BackboneJS te mnoge druge. Među svim tim tehnologijama, ReactJS i AngularJS su najšire prihvaćene za izradu SPA.

Tehnologija	**AngularJS**	**ReactJS**
Razvijen od strane	Google	Facebook
Godina izdanja	2009	2013
Idealno za	Izradu vrlo aktivnih i interaktivnih web aplikacija	Velike web aplikacije s često promjenjivim podatcima
Veličina aplikacije	Relativno mala	Relativno mala
Razina performansa	Visoka	Visoka
Protok podataka	Dvosmjeran	Jednosmjeran
Krivulja učenja	Strma	Umjerena
Najnovija dostupna verzija	14.1.0	18.0
Dinamično povezivanje UI-a	Povezivanje UI-a na razini običnog objekta	Direktno povezivanje stanja aplikacije s UI-om

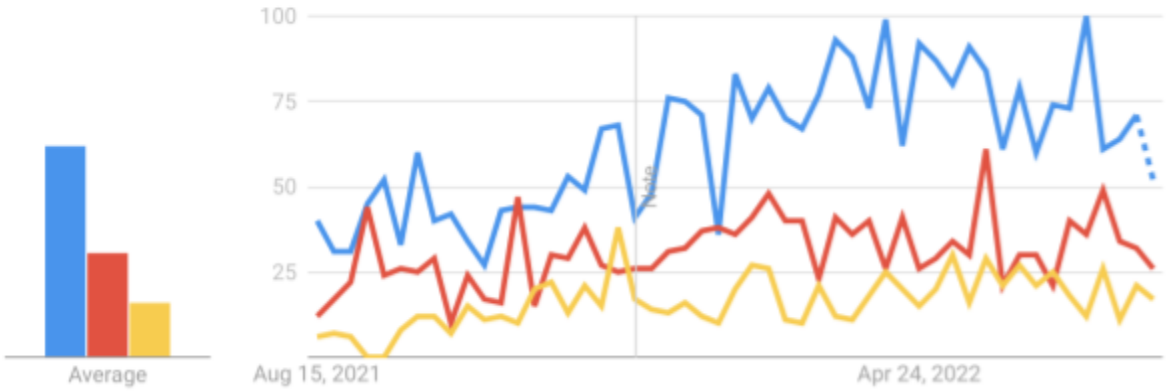
Renderiranje	Client/Server side	Client/Server side
Koje izabrati?	- TypeScript - Vrlo velika aplikacija - Čisti HTML - Objektno orijentirano programiranje	- Fleksibilnost - Veliki ekosistem - Manji tim - Veliki izbor paketa za nadogradnju aplikacije - Ljubav prema JavaScriptu

Tablica 4.1 Usporedba ReactJS-a i AngularJS-a

Kao što je prikazano u tablici 4.1, postoje neke temeljne razlike između AngularJS-a i ReactJS-a u smislu povezivanja podataka, manipulacije DOM-om te upotrebe razvojnog okvira/programske zbirke. Međutim, najveća razlika je u tome što je AngularJS razvojni okvir, dok je ReactJS programska zbirka, no oba se koriste za razvoj *front-end* aplikacija i web stranica.

5.2. Popularnost ReactJS-a te pozicija na tržištu

Mnoge tvrtke koje se bave poslovanjem, vijestima, putovanjima, društvenim mrežama u SAD-u, Velikoj Britaniji, Aziji, Francuskoj, Njemačkoj, Kanadi i mnogim drugim zemljama koriste i ReactJS i AngularJS. AngularJS je vrlo popularan među programerima jer je kompletan razvojni okvir koji dolazi s MVC oblikovnim obrascem, dok je ReactJS samo programska zbirka. React sadrži samo “Pogled” dio uzorka, dok mu nedostaju “Model” i “Kontroler”.



Slika 4.1 Preuzeto iz Google Trends, prikazuje omjer interesa pretraživanja za različite tehnologije u proteklih godinu dana uključujući ReactJS(plavo), AngularJS(crveno) i VueJS(žuto).

	ReactJS	**AngularJS**
Prednosti	- Pogodan za SEO - Vrlo jednostavno testiranje UI-a - Komponente možemo ponovno iskoristiti - Brz kad je u pitanju prikazivanje velikog broja podataka u komponentama - Specijalizirana Google Chrome ekstenzija olakšava pronalaženje pogrešaka	- Orijentiran na aplikacije s velikim brojem interaktivnog koda na strani klijenta - Dobro rješenje za dinamične SPA - Brz proces razvoja - U nekim slučajevima je potrebno manje koda - Napredne značajke za testiranje
Nedostatci	- U nekim slučajevima je potrebno napisati više koda - Ručno procesiranje prilikom promjene podataka	- Relativno spor kada je u pitanju velik broj podataka - Nije baš pogodan za SEO

Tablica 4.2 Prednosti i nedostaci između ReactJS-a i AngularJS-a

6. Praktični dio

Upotreba ReactJS razvojnog okvira, će biti prikazana kroz stranicu izrađenu u istom, u aplikaciji za evidenciju knjiga. U njoj će korisnik imati mogućnost pretraživanja knjiga s liste, dodavanja istih knjiga u favorite, kreiranja novih knjiga te isto tako i brisanja knjiga. Prikazana je upotreba ReactJS komponenti, raznih JavaScript ES6 značajka, te brzina renderiranja SPA napravljene u ReactJS-u.

6.1. Izrada radnog okruženja

Početno radno okruženje je generirano koristeći *Create React App (dalje u tekstu CRA)*. *CRA* je jednostavno i ugodno okruženje za, ne samo učenje ReactJS okvira, nego i za izradu složenih jednostraničnih aplikacija, bez prethodne konfiguracije, osim instaliranja *CRA-a*, te kreiranja samog projekta. *CRA* postavlja razvojno okruženje koje daje mogućnost korištenja najnovijih značajka JavaScript-a, pruža lijepo razvojno iskustvo i optimizira aplikaciju za proizvodnju.

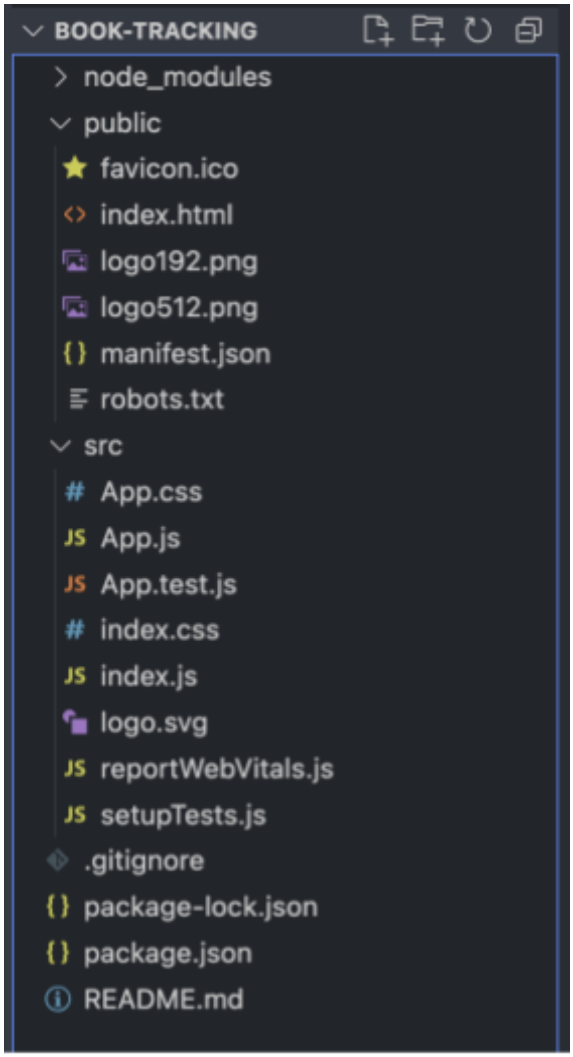
Prvi korak kreiranja okruženja jest instaliranje *CRA-a,* ulazak u direktorij projekta te pokretanje projekta koristeći *Node Package Manager (dalje u tekstu NPM)*.

```
npx create-react-app book-tracking
cd book-tracking
npm start
```

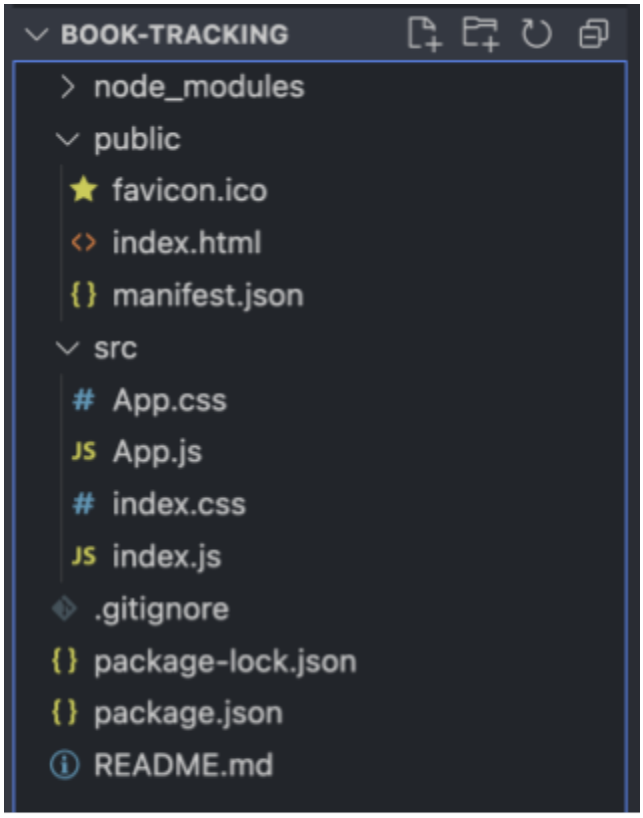
Programski kod 5.1 Naredbe potrebne za pokretanje CRA projekta

Pokretanjem ovih naredba, u lokalnom okruženju se pokreće ReactJS projekt, instaliraju se potrebne aplikacije treće strane, kreira se početna struktura direktorija te se automatski otvara u zadanom internetskom pregledniku.

Struktura automatski generiranih direktorija trenutno izgleda ovako, međutim, prilikom izrade vlastitog projekta je poželjno "počistiti" nekorištene datoteke.



Slika 5.1. Struktura automatski generiranih direktorija



Slika 5.2 Struktura direktorija nakon "čišćenja" nepotrebnih datoteka

6.2. Potrebne skripte za funkcioniranje ReactJS projekta

Kako bi ReactJS projekt radio, potrebne su određene skripte i zavisnosti, koje se instaliraju automatski prilikom korištenja CRA. Osim onih zadanih, korisnik ima mogućnost dodavanja vlastitih za proširenje mogućnosti projekta. Sve instalirane skripte i zavisnosti se nalaze u datoteci zvanoj *package.json* koja se prilikom pokretanja projekta automatski generira.

Slika 5.3 Potrebne skripte i dodatne instalacije za pokretanje ReactJS projekta

Na priloženoj slici, pod *dependencies* se nalaze zavisnosti koje su automatski instalirane prilikom pokretanja CRA, te također zavisnosti koje korisnik samostalno instalira, želi li neke dodatne mogućnosti. Zavisnosti kao što su *react, react-dom, react-scripts*, su neke od automatski instaliranih, koje služe za normalno funkcioniranje ReactJS projekta, *@testing-library* dolazi također kao zadana zavisnost te služi za pisanje testova. *React-router-dom* je samostalno instalirana zavisnost koja daje dodatne mogućnosti kreiranja ruta, te navigiranja kroz aplikaciju,

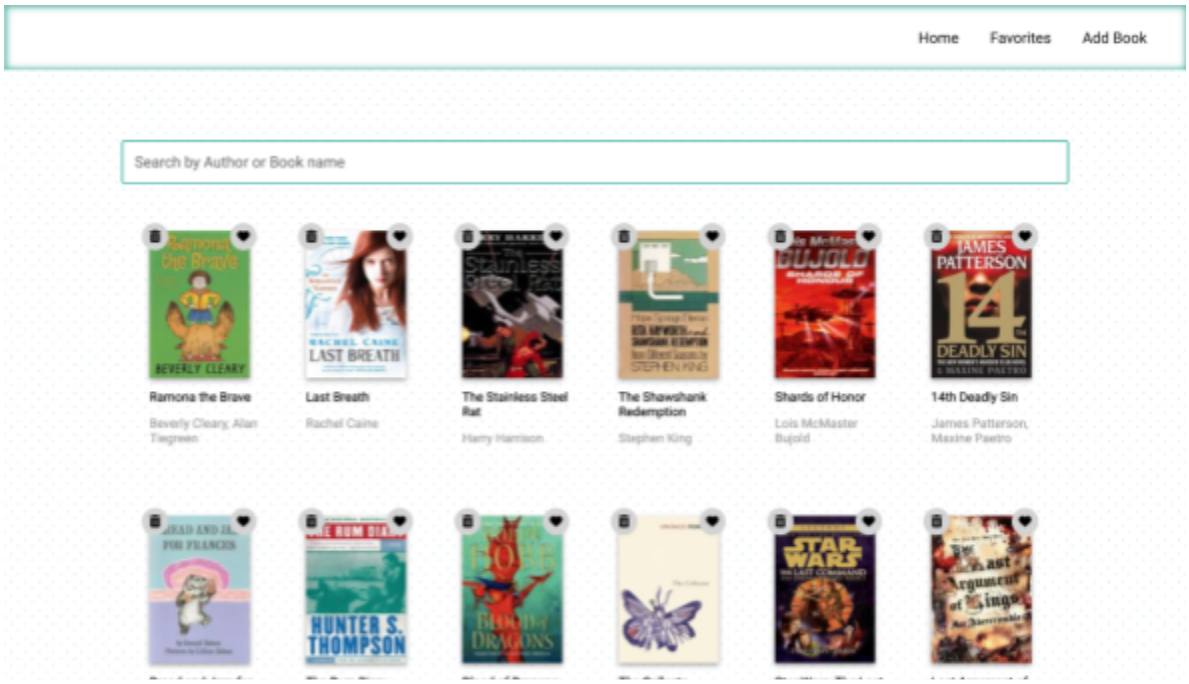
dok su *@fortawesome* zavisnosti zaslužne za ikone koju su korištene.

Također, u zadanim postavkama dolaze naredbe koje su izlistane pod *scripts*, koje služe za pokretanje projekta (*start*), te za izgradnju produkcijske verzije projekta (*build*).

6.3. Izrada stranica

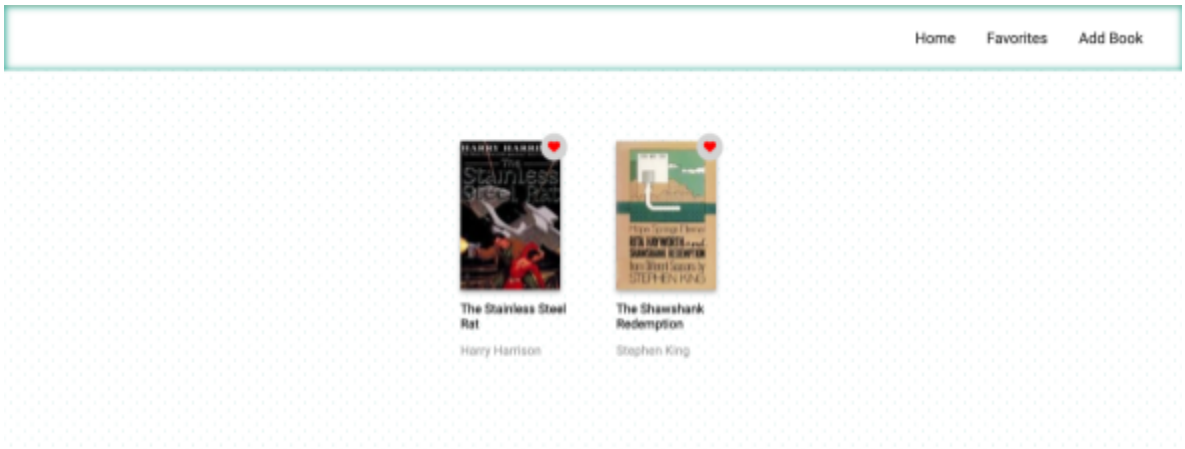
Aplikacija se sastoji od tri različite stranice:

- Početna stranica - Home, koja se sastoji od liste dostupnih knjiga, te trake za pretraživanje knjiga po imenu knjige ili po imenu autora. Prilikom upisivanja u polje autora ili naslova, popisa knjiga se automatski filtrira u odnosu na upisane pojmove. Također na svakoj pojedinačnoj knjizi, nalazi se ikona za dodavanje knjige u listu omiljenih knjiga, te ikona za brisanje istih, koje pritiskom na iste, odmah vraćaju povratnu informaciju korisniku je li knjiga u favoritima ili obrisana.



Slika 5.4 Početna stranica aplikacije

- Stranica gdje se nalaze omiljene knjige koje je korisnik izabrao - Favorites, na kojoj korisnik ima mogućnost uklanjanja knjiga s liste omiljenih knjiga.



Slika 5.5 Stranica omiljenih knjiga u aplikaciji

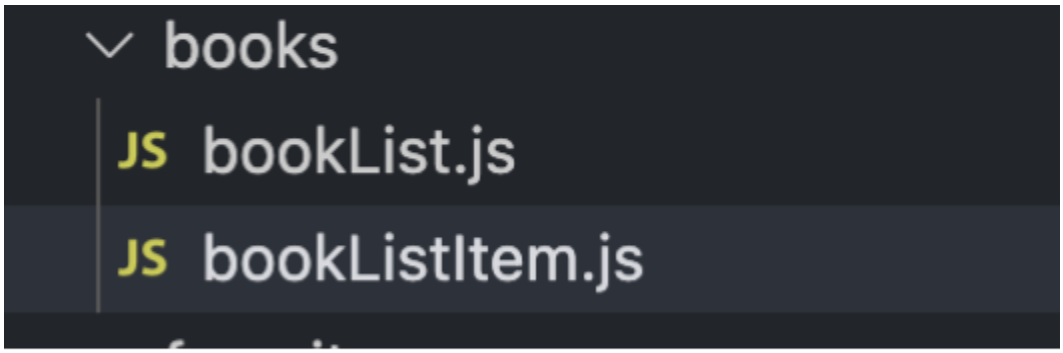
- Stranica na kojoj će korisnik moći dodati vlastite knjige - Add Book, na kojoj korisnik ima mogućnost unosa svih potrebnih podataka o knjizi, te dobiva povratnu informaciju o dodavanju knjige, ili, u slučaju greške, dobiva povratnu informaciju o neuspješnom dodavanju knjige. Također, osim vlastoručnog upisivanja informacija, korisnik ima mogućnost unošenja validnog ISBN-10 ili ISBN-13 koda, koji pritiskom na gumb, automatski dopunjuje polja pronađene knjige.

Slika 5.6 Dio stranice za unošenje knjiga gdje korisnik unosi vlastite podatke.

Slika 5.7 Dio stranice za unošenje knjiga gdje korisnik unosi ISBN-10 ili ISBN 13 kod

6.4. Izrada komponenti

Stranica je bazirana na komponentama, koje su srce ReactJS-a. Izrađena je zasebna komponenta za listu knjiga, u kojoj kroz petlju prikazujemo svaku pojedinačnu knjigu, koja također ima vlastitu komponentu.



Slika 5.8 Zasebna komponenta za listu knjiga, te komponenta za svaku pojedinačnu knjigu.

Isto tako je i izrađena zasebna komponenta za listu omiljenih knjiga u kojoj također kroz petlju prikazujemo svaku pojedinačnu knjigu, kao i komponenta za svaku pojedinačnu knjigu.

Slika 5.9 Zasebna komponenta za listu omiljenih knjiga, te komponenta za svaku pojedinačnu omiljenu knjigu.

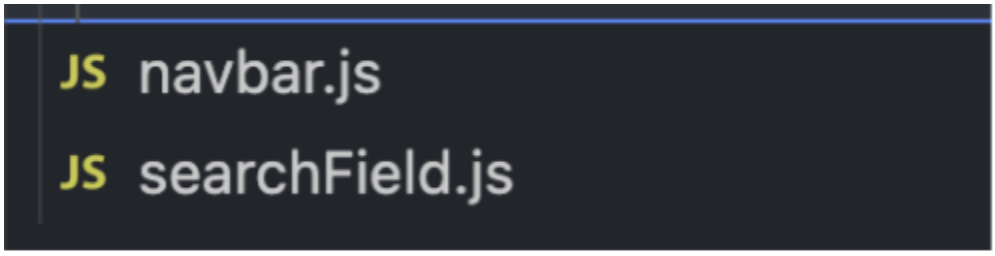
Stranica za dodavanje knjiga također ima vlastitu komponentu, u kojoj je smještena forma za unos podataka.

Slika 5.10 Zasebna komponenta koja drži formu za dodavanje knjiga.

```
<form className="add-book__form">
  <div className="add-book__input-wrapper">
    <input
      type="text"
      id="title"
      onChange={this.handleChange}
      placeholder=" "
      name="title"
      value={this.state.value}
    />
    <label htmlFor="title">Book Title</label>
  </div>
  <div className="add-book__input-wrapper">
    <input
      type="text"
      id="author"
      onChange={this.handleChange}
      placeholder=" "
      name="author"
      value={this.state.value}
    />
    <label htmlFor="author">Book Author</label>
  </div>
  <div className="add-book__input-wrapper">
    <input
      type="text"
      id="cover"
      onChange={this.handleChange}
      placeholder=" "
      name="cover"
      value={this.state.value}
    />
    <label htmlFor="cover">Book Cover</label>
  </div>
  <div className="add-book__input-wrapper">
    <input
      type="text"
      id="year"
      onChange={this.handleChange}
      placeholder=" "
      name="year"
      value={this.state.value}
    />
    <label htmlFor="year">Book Year</label>
  </div>
</form>
```

Slika 5.11 Forma za unos podataka nove knjige.

Kao i svi dijelovi stranice, tako i navigacija i traka za pretraživanje podataka imaju vlastite komponente.



Slika 5.12 Zasebne komponente za navigaciju i traku za pretraživanje.

```
1   import React, { Component } from 'react'
2   import { Link } from 'react-router-dom'
3
4   export class Navbar extends Component {
5     render() {
6       return (
7         <ul className="navbar">
8           <li className="navbar__item">
9             <Link to={'/'}>Home</Link>
10          </li>
11          <li className="navbar__item">
12            <Link to={'/favorites'}>Favorites</Link>
13          </li>
14          <li className="navbar__item">
15            <Link to={'/add-book'}>Add Book</Link>
16          </li>
17        </ul>
18      )
19    }
20  }
21
22  export default Navbar
```

Slika 5.13 Komponenta za navigaciju.

6.5. Dohvaćanje podataka

Dohvaćanje podataka se nalazi u zasebnoj datoteci, te je svaki zahtjev za određene podatke stavljen u vlastitu funkciju, zbog lakšeg pristupa dohvaćanju podataka kroz cijeli projekt. Ono se izvršava koristeći zadani JavaScript API, fetch API. Fetch API pruža sučelje za dohvaćanje resursa (uključujući i preko mreže).

```
4   export const getBooks = () =>
5     fetch(books, {
6       "Content-Type": "application/json"
7     })
8     .then(res => res.json())
```

Slika 5.14 Struktura jedne od funkcija za dohvaćanje podataka u kojoj korisnik dobiva listu knjiga.

U slici iznad je prikazan zahtjev za dohvaćanje liste knjiga, dok će u nastavnoj slici biti prikazan također jedan od bitnijih zahtjeva, a to je pretraživanje knjiga po imenu autora ili po naslovu. Korisnik u tom zahtjevu šalje ključnu riječ (engl. query) koju funkcija prima kao argument, koja se tada ubacuje u krajnju točku na koju se šalje zahtjev, u konačnici i dobivaju podaci, a smještena u `"?q=${query}"` dijelu krajnje točke.

```
export const searchBooks = (query) =>
  fetch(`http://localhost:8080/books?q=${query}`, {
    "Content-Type": "application/json"
  })
  .then(res => res.json())
```

Slika 5.15 Struktura funkcije za pretraživanje knjiga.

6.6. Struktura podataka

Podatke dohvaćamo koristeći *json-server*, što je paket treće strane pomoću kojeg je moguće izraditi lažni API, bez dodatnog pisanja koda, te bez znanja *backend* strane razvoja. Potrebno je samo izraditi "bazu" s podacima *json* oblika, te pokrenuti naredbu pomoću koje se lokalno pokreće server, na koji imamo pristup podatcima iz prethodno izrađene baze, te mogućnost slanja zahtjeva kao što su GET, POST, DELETE, PUT.

```
{
  "books": [
    {
      "id": 1,
      "title": "Ramona the Brave",
      "author": "Beverly Cleary, Alan Tiegreen",
      "cover":
"https://images.gr-assets.com/books/1408925322m/91248.jpg",
      "year": 1975,
      "isFavorite": false
    }
  ],
  "favorites": []
}
```

Programski kod 5.1 Struktura podataka o knjizi u bazi podataka

U odjeljku kod je prikazana struktura "baze" podatka u *json* obliku, gdje se nalazi lista knjiga, te polje za unos istih knjiga u favorite.

6.7. Izrada ruta

Rute, tj. stranice aplikacije su izrađene koristeći *react-router-dom*. *React-router-dom* jest paket koji sadrži potrebne komponente i poveznice za korištenje *react-router-a* u web aplikacijama. U poglavlju izrade komponenata je prikazana upotreba *Link* komponente u navigaciji koja služi za preusmjeravanje korisnika na određene rute. Te rute su izrađene u korijenskoj JavaScript datoteci, u kojoj se koristeći *Routes* i *Route* komponente određuju koje će rute postojati u samoj aplikaciji.

```
<div className="main__content">
  <Routes>
    <Route exact path='/' element={<BookList />} />
    <Route path='/favorites' element={<FavoritesList />} />
    <Route path='/add-book' element={<AddBook />} />
  </Routes>
</div>
```

Slika 5.16 Postojeće rute izrađene za svaku pojedinačnu stranicu aplikacije.

7. Zaključak

Cilj završnog rada je bio proučavanje i pregled *front-end* programske zbirke temeljene na JavaScriptu pod nazivom ReactJS. ReactJS je razvijen od strane Facebook-a za vlastite potrebe te je tek kasnije pušten u otvorenu upotrebu. Od samog početka, u vrlo kratkom vremenu, ReactJS je stekao ogromnu popularnost među programerima, a i među cijelom tehnološkom industrijom.

U ovom su dokumentu ilustrirane jasne upute o samom početku izrade projekta koristeći ReactJS, glavne značajke i funkcionalnosti ReactJS-a s primjerima, kada uzeti u obzir ReactJS u odnosu na alternative, te osnovni paketi koji su potrebni za izradu ReactJS aplikacije visoke kvalitete.

Budući da je ReactJS tražena i važna tehnologija u današnjem vremenu, učenje istog kroz daljnje istraživanje je veoma korisno za obogaćivanje vlastitih vještina modernog programiranja. ReactJS je nevjerojatno fleksibilan. Nakon dovoljno stečenog znanja, može se primijeniti za izgradnju web aplikacija i web stranica na raznim platformama.

Konačno, sa sigurnošću se može reći da je ReactJS privlačna tehnologija koju je veoma korisno naučiti i svakako je vrijedna razmatranja za primjenu u proizvodnji. ReactJS je donio novu dimenziju u razvoju web aplikacija. Programska zbirka brzog renderiranja, ubrzava učinkovitost aplikacije i može se vidjeti da ReactJS ima svijetlu budućnost te da je učenje istog vrijedno truda.

Sveučilište
Sjever

IZJAVA O AUTORSTVU
I
SUGLASNOST ZA JAVNU OBJAVU

Završni/diplomski rad isključivo je autorsko djelo studenta koji je isti izradio te student odgovara za istinitost, izvornost i ispravnost teksta rada. U radu se ne smiju koristiti dijelovi tuđih radova (knjiga, članaka, doktorskih disertacija, magistarskih radova, izvora s interneta, i drugih izvora) bez navođenja izvora i autora navedenih radova. Svi dijelovi tuđih radova moraju biti pravilno navedeni i citirani. Dijelovi tuđih radova koji nisu pravilno citirani, smatraju se plagijatom, odnosno nezakonitim prisvajanjem tuđeg znanstvenog ili stručnoga rada. Sukladno navedenom studenti su dužni potpisati izjavu o autorstvu rada.

Ja, KARLO TOMAŠIĆ *(ime i prezime)* pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor/ica završnog/diplomskog *(obrisati nepotrebno)* rada pod naslovom REACTJS JAVASCRIPT PROGRAMSKA ZBIRKA *(upisati naslov)* te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.

Student/ica:
(upisati ime i prezime)

Karlo Tomašić
(vlastoručni potpis)

Sukladno Zakonu o znanstvenoj djelatnost i visokom obrazovanju završne/diplomske radove sveučilišta su dužna trajno objaviti na javnoj internetskoj bazi sveučilišne knjižnice u sastavu sveučilišta te kopirati u javnu internetsku bazu završnih/diplomskih radova Nacionalne i sveučilišne knjižnice. Završni radovi istovrsnih umjetničkih studija koji se realiziraju kroz umjetnička ostvarenja objavljuju se na odgovarajući način.

Ja, KARLO TOMAŠIĆ *(ime i prezime)* neopozivo izjavljujem da sam suglasan/na s javnom objavom završnog/diplomskog *(obrisati nepotrebno)* rada pod naslovom REACTJS JAVASCRIPT PROGRAMSKA ZBIRKA *(upisati naslov)* čiji sam autor/ica.

Student/ica:
(upisati ime i prezime)

Karlo Tomašić
(vlastoručni potpis)

8. Literatura

1) https://www.reactenlightenment.com/react-jsx/5.1.html, dostupno 02.07.2022.
2) https://reactjs.org/community/support.html, dostupno 05.07.2022.
3) https://www.topsinfosolutions.com/blog/choose-reactjs-front-end-development/, dostupno 07.07.2022.
4) https://www.simform.com/blog/angular-vs-react/#:~:text=Angular%20is%20a%20Javascript%20framework,app%20with%20frequently%20variable%20data, dostupno 11.07.2022.
5) https://www.npmjs.com/package/react-router-dom, dostupno 15.07.2022.
6) https://docs.mockend.com/, dostupno 28.07.2022.
7) https://reactjs.org/docs/create-a-new-react-app.html, dostupno 20.08.2022.
8) https://www.freecodecamp.org/news/why-use-react-for-web-development/, dostupno 22.08.2022.
9) https://www.freecodecamp.org/news/all-the-fundamental-react-js-concepts-jammed-into-this-single-medium-article-c83f9b53eac2/, dostupno 25.08.2022.
10) http://blog.thefirehoseproject.com/posts/reactjs-101/, dostupno 26.08.2022.
11) https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs, dostupno 26.08.2022.
12) https://medium.com/the-research-nest/everything-you-need-to-know-about-react-ab24da4275ea, dostupno 26.08.2022.
13) https://www.popwebdesign.net/popart_blog/2021/12/zasto-angazovati-najbolje-react-js-developere-za-svoju-veb-aplikaciju/, dostupno 06.09.2022.
14) https://www.royalcyber.com/blog/ecommerce/react-spa-for-increased-performance-and-customer-experience/, dostupno 09.09.2022.

9. Popis slika