Izrada grafičkog sučelja web aplikacije koristeći programski okvir React na studijskom slučaju TODO aplikacije

Sudar, Josip

VELEUČILIŠTE U ŠIBENIKU

ODJEL POSLOVNE INFORMATIKE

PREDDIPLOMSKI STRUČNI STUDIJ POSLOVNE INFORMATIKE

Josip Sudar

Izrada grafičkog sučelja web aplikacije koristeći programski okvir React na studijskom slučaju TODO aplikacije

Završni rad

Šibenik, 2023.

VELEUČILIŠTE U ŠIBENIKU

ODJEL POSLOVNE INFORMATIKE

PREDDIPLOMSKI STRUČNI STUDIJ POSLOVNE INFORMATIKE

Izrada grafičkog sučelja web aplikacije koristeći programski okvir React na studijskom slučaju TODO aplikacije

Završni rad

Kolegij:

Mentor: Marko Pavelić, mag. ing. inf. et comm. techn., pred.

Student: Josip Sudar

Matični broj studenta: 0243104715

Šibenik, rujan 2023.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, Josip Sudar , student Veleučilišta u Šibeniku, JMBAG 0243104715 izjavljujem pod materijalnom i kaznenom odgovornošću i svojim potpisom potvrđujem da je moj završni rad na preddiplomskom/specijalističkom diplomskom stručnom studiju Poslovna informatika pod naslovom: Izrada grafičkog sučelja web aplikacije koristeći programski okvir React na studijskom slučaju TODO aplikacije

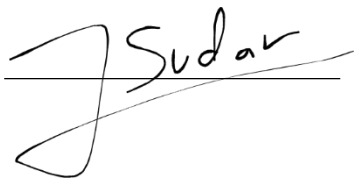
isključivo rezultat mog vlastitog rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu, a što pokazuju korištene bilješke i bibliografija.

Izjavljujem da nijedan dio rada nije napisan na nedozvoljen način, odnosno da je prepisan iz necitiranog rada, te da nijedan dio rada ne krši bilo čija autorska prava.

Izjavljujem, također, da nijedan dio rada nije iskorišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

U Šibeniku, 19.09.2023

Student/ica:



TEMELJNA DOKUMENTACIJSKA KARTICA

Veleučilište u Šibeniku Završni rad

Odjel Poslovne informatike

Preddiplomski/Specijalistički diplomski stručni studij Poslovne Informatike

Izrada grafičkog sučelja web aplikacije koristeći programski okvir React na studijskom slučaju TODO aplikacije

Josip Sudar

Danilska 40, 22000 Šibenik, josip.sudar@gmail.com

Sažetak rada

Ovaj rad opisuje izradu grafičkog sučelja TODO aplikacije koristeći programski okvir React.js. Izrada se temelji na kreiranju vlastitih komponenti kao i korištenju javno dostupnih komponenti. Aplikacija je povezana na poslužitelj te ima mogućnost registracije, prijave, kreiranja zadatka, prikaza zadataka, filtriranja zadataka prema imenu, završavanju zadatka i brisanja istog.

(62 stranica / 33 slika / 25 tablica / 28 literaturnih navoda / jezik izvornika: hrvatski)

Rad je pohranjen u digitalnom repozitoriju Knjižnice Veleučilišta u Šibeniku

Ključne riječi: Todo, React, Axios, Klijentska Strana, Poslužitelj

Mentor: Marko Pavelić, mag. ing. inf. et comm. techn., pred.

Rad je prihvaćen za obranu dana:

BASIC DOCUMENTATION CARD

Polytechnic of Šibenik Batchelor/Graduation Thesis

Department of Business Informatics

Professional Undergraduate/Graduate Studies of Business Informatics

Creation of the graphic interface of Todo application using React framework.

Josip Sudar

Danilska 40, 22000 Šibenik, josip.sudar@gmail.com

Abstract

This paper describes the creation of the graphical interface of the TODO application using the React.js framework. The development is based on the creation of own components as well as the use of publicly available components. The application is connected to the server and has the ability to register, log in, create a task, display tasks, filter tasks by name, complete a task and delete it.

(62 pages / 33 figures / 25 tables / 28 references / original in Croatian language)

Thesis deposited in Polytechnic of Šibenik Library digital repository

Keywords: Todo, React, Client side, Server side, Axios

Supervisor: Marko Pavelić, mag. ing. inf. et comm. techn., pred.

Paper accepted:

SADRŽAJ

1. UVOD

Kao uvod u izradu same web aplikacije počinje se od samih početaka web stranica kao takvih. Prema navodima internetske stranice businessinsider prva web stranica pokrenuta je 06. ožujka. 1991. godine. Služila je kao informativna stranica za World Wide Web projekt a njeni autor bio je Tim Berners-Lee. Stranica je kao osnovu za jezik koristila prvu verziju HTML-a koja je imala mogućnost pisati samo tekst bez dodavanja slika ili uređivanja same stranice (bussinesinsider, 2011). Kako je vrijeme prolazio tehnologija se jako mnogo razvijala pa tako danas imamo internetske trgovine, stranice s mogućnosti gledanja videozapisa, stranice s animacijama, i dr. Sve je to omogućeno uz pomoć raznih programskih okvira programskog jezika JavaScript kao što su React.js, Next.js, Vue.js, Vite.js, i dr. Ono što oni omogućuju je kreiranje bržih, sigurnijih, jednostavnijih web stranica jer HTML kao takav nema napredne mogućnosti bez korištenja JavaScripta. Ovaj rad usmjerava pažnju na programski okvir React.js koji se koristi za izradu TODO web aplikacije u kojoj korisnik ima mogućnost kreiranja, promjene i brisanja zadataka te prijavu i registraciju u samu stranicu. Razlog biranja ove teme je taj što izrada TODO web aplikacije prikazuje osnovne principe i načela na kojima se programski okvir React.js zasniva.

2. OPIS TEHNOLOGIJA KORIŠTENIH U IZRADI

2.1 HTML

HyperText Markup Language odnosno HTML je prezentacijski jezik koji se koristi u izradi web stranica. Sastoji se od niza elemenata i atributa koji definiraju strukturu i sadržaj web stranice. Ostale tehnologije koje se često koriste uz HTML su stilski jezik CSS i jezik funkcionalnosti JavaScript koji će kasnije biti detaljno objašnjeni.

HyperText se odnosi na poveznice koje povezuju web stranice međusobno budući da su poveznice sastavni dio interneta kakvog poznajemo. Markup se odnosi na način koji HTML obilježava naslove, tekst, slike i ostale elemente za prikaz u internetskom pregledniku. Neki od najpoznatijih elemenata koji se najviše koriste su `<title>`, `<body>`, `<p>`, `<div>`, a svi elementi se mogu pronaći na stranici developer.mozilla.org/en-US/docs/Learn/HTML/Cheatsheet. Primjećujemo da se svaki element sastoji od oznaka (engl. *tags*) koji su zapravo znakovi jednakosti (< , >) (devdocs, 2023). Slika 1 prikazuje način zapisivanja paragrafa koristeći HTML.

Slika 1: Prikaz HTML oznake

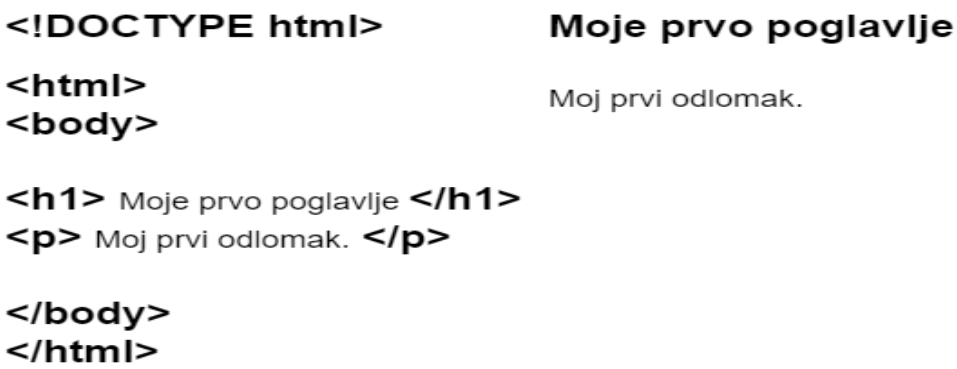
Izvor: (devdocs, 2023)

Na početku se nalazi početna oznaka (engl. *opening tag*) koja označava određeni element počinje vrijediti što je u ovom slučaju paragraf. U sredini se nalazi sadržaj (engl. *content*) kojega element prikazuje što je u ovom slučaju obični tekst. Na kraju se nalazi završna oznaka (engl. *closing tag*) koja se od početne razlikuje u tome što sadrži kosu crtu i na taj način HTML zna da tu određeni element koji je započeo završava. Svi ovi ranije spomenuti pojmovi zajedno tvore HTML element.

HTML kao takav nastao je 1991.godine a njegov izumitelj je bio engleski informatičar Timothy Berners-Lee koji je u CERN-u stvorio novi način prikazivanja informacija putem računala pod imenom HTML 1.0. koji je imao mogućnosti osnovnog prikaza teksta. Tokom godina se stalno unaprjeđivao i poboljšavao te 2014. nastaje HTML 5.0 koji je danas standard u web industriji i koristi se širom svijeta (bu.edu, 2020).

Slika 2 prikazuje generički HTML dokument u programu za uređivanje koda i kako to izgleda na internetskom pregledniku.

Slika 2: Prikaz HTML dokumenta



Izvor: (w3schools, 2023)

`<!DOCTYPE html>` označava deklaracijski element koji pomaže pregledniku da web stranicu prikaže kako treba. `<html> </html>` označava sami dokument dok `<body> </body>` označava vidljivi dio dokumenta. Ostali elementi su `<h1> </h1>` što označava naslov i `<p> </p>` što označava paragraf.

2.2 CSS

Cascading Style Sheets odnosno CSS je jezik temeljen na pravilima koji omogućava uređivanje web stranica. Pod jezikom temeljenim na pravilima se misli da se prilikom izrade moraju definirati određeni stilovi koji se koriste na određenom elementu što se može vidjeti u Kod 1.

Kod 1: Prikaz CSS koda

```
h1 {
  color: red;
  font-size: 5em;
}
p {
  color: black;
}
```

Izvor: (developer.mozilla, 2023)

Kod 1 sadrži element za **naslov (h1)** i za **paragraf (p)**. Svaka naredba u CSS-u započinje da se definira element odnosno selektor kojega se dizajnira. Unutar vitičastih zagrada **{ }**[1] unose se vrijednosti i naredbe za pojedini element. Lista kompletnih CSS naredbi se može pronaći ovdje http://cheatsheets.shecodes.io/css (developer.mozilla, 2023).

2.2.1 Tailwind CSS

U ovom radu će se koristiti Tailwind CSS. Tailwind CSS je stilski jezik koji omogućuje pisanje CSS-a u samim HTML elementima bez potrebe sa stvaranje još jedne datoteke u koju se inače piše CSS čime se štedi jako mnogo vremena. Još jedna velika prednost je ta što posjeduje vlastitu sintaksu koja je mnogo skraćenija i preglednija u odnosu na klasični CSS što se može vidjeti u Kod 2.

[1] Oznaka CSS naredbe

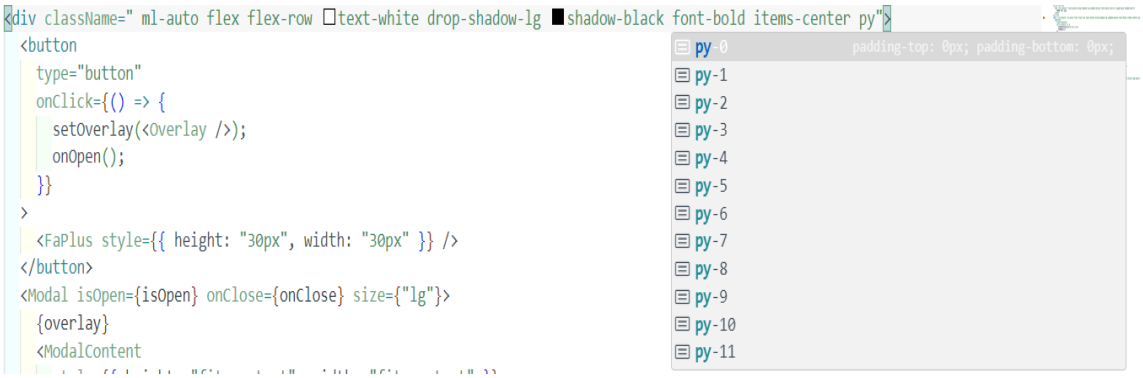
Kod 2: Usporedba Tailwind CSS-a s klasičnim CSS-om

.btn{@apply text-white py-2 px-4 rounded;} .btn-purple{@apply bg-purple-500;} .btn-purple:hover{@apply bg-purple-700;}	.btn{color:#fff; padding:.5rem 1rem; broder-radius:.25rem;} .btn-purple{background-color:#9f7aea;} .btn-purple:hover{background-color:#6b46c1;}

Izvor: (marketplace.visualstudio, 2023)

Ono što je još važno je to da posjeduje vlastiti InteliSense odnosno algoritam koji programerima omogućava inteligentno dovršavanje koda ili teksta čime se smanjuju tipografske ili pogreške u sintaksi prikazano na Slika 3.

Slika 3: Tailwind.css IntelliSense



Sami postupak instalacije je sličan sa sve platforme dostupne dok će u ovom radu biti objašnjena instalacija u programskom okviru React.js. Kao prvi korak u terminal se upisuju naredbe `npm install -D tailwindcss` i `npx tailwindcss init`. Prva naredba instalira pakete odnosno module potrebne za rad Tailwind-a dok druga naredba kreira konfiguracijsku datoteku pod imenom tailwind.config.js prikazanu u Kod 3.

Kod 3: Tailwind konfiguracijska datoteka

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: ["./src/**/*.{html,js}"],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

Izvor: (tailwindcss/installation, 2023)

Ono što je zadnje preostalo je u index.css datoteku odnosno glavnu CSS datoteku cijelog projekta dodati ove tri naredbe koje omogućuju pravilno funkcioniranje CSS-a na svim elementima i komponentama kako bi instalacija bila uspješna kao što je prikazano u Kod 4 (tailwindcss/installation, 2023).

Kod 4: Tailwind konfiguracijske naredbe

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

Izvor: (tailwindcss/installation, 2023)

2.3 JAVASCRIPT

Kako bi se mogao objasniti programski okvir React.js prvo je potrebno objasniti programski jezik na kojem se on zasniva. Programski jezik u pitanju je JavaScript koji omogućava implementaciju kompleksnih značajki u web stranicu. Kada web stranica sadrži animirane objekte, interaktivne elemente ili nešto dinamično na njoj velika vjerojatnost je ta da je to rađeno preko JavaScript-a.

Slika 4: JavaScript skripta

Izvor: *(geekboots, 2022)*

JavaScript je programski jezik koji se koristi za kreiranje dinamičkih interakcija web stranica s korisnikom. Koristi se za izradu mobilnih i web aplikacija, web servera i serverskih aplikacija[2], dinamičkog izgleda web stranice i razvoj mobilnih igrica.

Prednosti JavaScript-a:

- Jednostavnost: lagana struktura omogućuje brzo učenje i implementaciju
- Brzina: skripte se pokreću odmah bez potrebe za kompiliranjem
- Prilagodljivost: jako dobra kompatibilnost s ostalim programskim jezicima
- Popularnost: veliki broj resursa za početnike s niskom razinom znanja
- Smanjeno opterećenje na server: omogućuje smanjeni broj zahtjeva na server iz razloga što se učitavanja sadržaja odnose samo na dijelove stranice a ne na cijelu stranicu
- Nadogradnje: razvojni tim stalno nadograđuje i kreira nova razvojna okruženja

Nedostaci JavaScript-a:

- Kompatibilnost različitih internetskih preglednika: svaki internetski preglednik ima drugačije načine kako interpretira JavaScript kod
- Sigurnost: kod koji se izvršava na klijentskoj strani odnosno na samoj web stranici je podložan eksploataciji od strane neodgovornih korisnika
- Otklanjanje pogrešaka: često razvojna okruženja nema ugrađene alate za otklanjanje pogrešaka što programiranje može znatno otežati

[2] Tu se pretežito koristi Node.js koji će kasnije biti detaljno objašnjen

Ono što je još bitno napomenuti je to da se JavaScript kod može kreirati unutar HTML datoteke pod tagom odnosno elementom `<script> </script>` ili u posebnoj datoteci koja se zatim poziva unutar HTML datoteke (hostinger, 2023).

2.4 NODE.JS

Razvojno okruženje Node.js je razvojno okruženje za JavaScript programski jezik otvorenog tipa koje omogućuje pokretanje web aplikacija izvan internetskog preglednika korisnika te je pogodan za aplikacije s velikom količinom podataka budući da koristi asinkrone modele. Ono što to zapravo znači je to da sve radnje kao što su učitavanje, renderiranje i slično obavlja Node.js umjesto internetskog preglednika čime se znatno rasterećuje korisnik. U ovom konkretnom slučaju koristi se kao lokalni server i za kreiranje predložaka React aplikacije[3].

Razlozi zbog kojega je tako popularan u svijetu web programiranja je taj što je izrađen na Google V8 engine[4] koji omogućava brzo pokretanje web aplikacija, postoji široka paleta paketa dostupnih na Node Package Manager (NPM) koje developeri mogu jednostavno uvesti u kod ovisno o potrebi, pogodan je za kreiranje web aplikacija s velikom količinom podataka jer je asinkron što znači da je učitavanje komponenti neovisno jedna o drugoj, omogućava bolju sinkronizaciju audio i video zapisa jer se koristi isti bazi kod između klijenta i servera i ono najbitnije je to da je javno dostupan (simplilearn.com/nodejs, 2023).

Značajke Node.js-a:

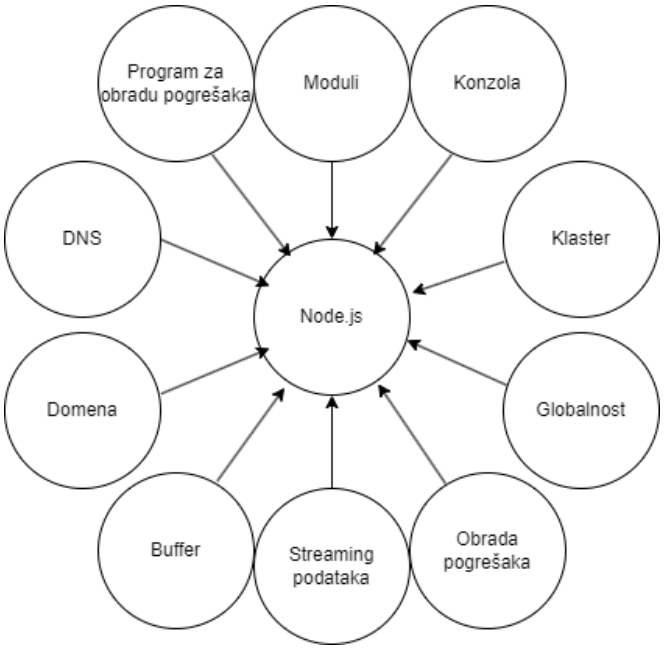
- Asinkron i pogonjen događajima: Kada je u procesu renderiranja web aplikacije te dođe do API-a u kojem ne može doći do podataka prelazi na drugoga čime korisniku ne blokira učitavanje aplikacije. Kako bi primio podatke API-a koje je preskočio koristi mehanizam događaja što znači da prilikom pokretanja skenira kod za sve varijable i funkcije.
- Single-Threaded arhitektura: Za razliku od ostalih servera koji su sinkroni jer sve zahtjeve procesiraju odjednom Node.js kao što smo ranije rekli je asinkron te može procesirati veliku količinu podataka čime se postiže visoka razina skalabilnosti
- Skalabilnost: Rješava jedan od najvećih problema razvoja softvera jer može

[3] U sljedećem poglavlju sve naredbe i komponente predloška React aplikacije će biti detaljno objašnjene
[4] Engine korišten za pokretanje i održavanje JavaScript-a na Google Chrome internetskom pregledniku

istovremeno upravljati zahtjevima pomoću klastera koji upravlja opterećenjem procesorskih jezgara putem particija. Na ovaj način različite verzije aplikacije prenose se ciljanoj publici te se omogućuje prilagodba željama klijenata.

- Brzo vrijeme izvršavanja koda: Koristi Google Chrome V8 engine kojeg smo ranije spomenuli.
- Kompatibilnost na različitim platformama: Različite vrste sustava kao što su Windows, Linux ili MacOS kao i mobilni sustavi mogu koristiti Node.js.
- Koristi JavaScript: Za developera velika je prednost kada koristi razvojno okruženje u programskom jeziku s kojim već dobro barata.
- Brzi prijenos podataka: Istovremeno obrađuje i učitava datoteke čime poboljšava ukupnu brzinu protoka podataka.
- Ne koristi među spremnik: Podaci se nikad ne spremaju u među spremnik.

Slika 5: Node.js komponente



Izvor: *(simplilearn.com/nodejs, 2023)*

Slika 5 prikazuje različite komponente Node.js razvojnog okruženja:

- Moduli: Moduli su JavaScript biblioteke koje uključuju set funkcija. Da bi se koristio u aplikaciji koristi se „require( )“ funkcija.
- Konzola: Modul koji omogućuje otklanjanje pogrešaka u internetskom pregledniku.
- Klaster: Modul koji omogućuje multi-threading kreiranje pod procesa koji se vrte

istovremeno.

- Globalnost: Funkcije, moduli, stringovi, … su dostupni globalno na razini koda.
- Obrada pogreške: Pogreške se obrađuju kroz tzv. „exception“. Ono što to zapravo znači je to da ukoliko kompajler naiđe na grešku u kodu aplikacija se ne ruši već nam daje mogućnost obrade greške prije nastavka.
- Streaming podataka: Streaming omogućuje kontinuirano čitanje ili pisanje podataka. Postoje četiri vrste: podaci koji se mogu samo čitati, podaci koji se mogu samo pisati, podaci koji mogu oboje te podaci kojima se može manipulirati tokom čitanja i pisanja.
- Buffer: Modul koji omogućuje obradu tokova.
- Domena: Modul koji presreće greške koje nisu obrađene.
- DNS: Modul za povezivanje na DNS[5] server.
- Program za otklanjanje pogrešaka: omogućuje pregledavanje koda.

2.5 REACT.JS

Nakon što su objašnjene tehnologije na kojima se programski okvir React.js zasniva, u ovom podpoglavlju biti će objašnjen programski okvir React.js. React.js je besplatno okruženje za razvoj aplikacija otvorenog tipa koje za bazu koristi JavaScript programski jezik te služi u svrhu kreiranja korisničkih sučelja putem komponenti. U ovom poglavlju programski okvir će biti teorijski objašnjen dok će se u samu srž programskog okvira ulaziti u procesu same aplikacije te će cijeli proces biti potkrepljen slikama. Kreirala ga je kompanija Facebook u svibnju 2013-te godine, te je jedno od najraširenijih i najbrže rastućih okruženja u razvoju web aplikacija.

[5] Domain Name System: sustav za dobivanje domena za internetske stranice.

Slika 6: React.js logo

Izvor: *(bairesdev, 2023)*

Razlozi zbog kojih bih netko koristio React.js umjesto klasičnog HTML-a i JavaScript-a je taj što React: omogućuje lakšu izradu dinamičkih web aplikacija jer zahtjeva manje kodiranja uz više funkcionalnosti, poboljšava izvedbu web aplikacije korištenjem virtualnog DOM-a[6] čime uspoređuje stanja komponenti i ažurira samo one u kojima se dogodila promjena, koristi komponente s višekratnom upotrebom koje su osnova kreiranja React aplikacije te imaju svoju logiku i mogu se ponovno koristiti u cijeloj aplikaciji što smanjuje vrijeme razvoja aplikacije, koristi jednosmjerni tok podataka što znači da programeri mogu ugnijezditi pojedine komponente za lakše praćenje pogrešaka, vrlo je lagan za naučiti jer uglavnom kombinira klasični HTML, CSS i JavaScript s dodacima, sadrži alat za otklanjanje pogrešaka (simplilearn/reactjs, 2023).

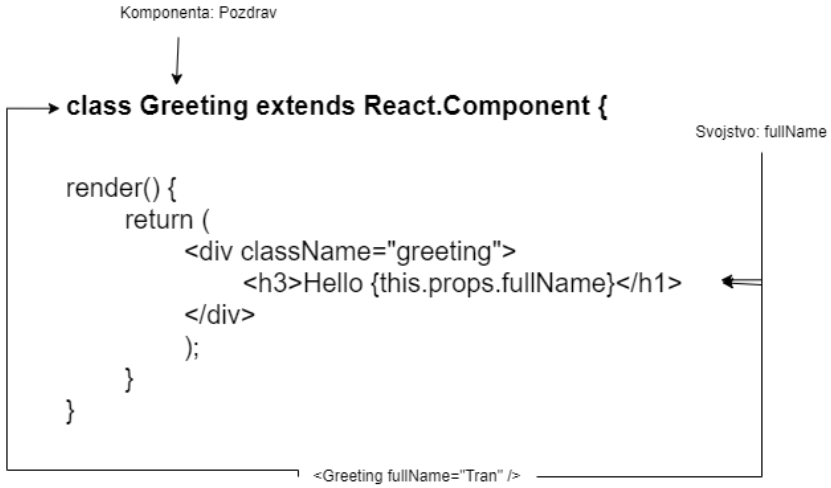
Značajke React.js-a:

- JSX: JSX odnosno JavaScript syntatic extension vrsta datoteke u kojoj možemo kombinirati HTML strukturu s dinamičnosti JavaScript-a.
- Virutal DOM: Kao što smo već ranije naveli React koristi strukturu grananja HTML dokumenta te mijenja stanje samo onih elemenata u kojima se promjena dogodi.
- Paketi i ekstenzije: Programeri imaju mogućnost implementacije raznih modula koristeći ranije objašnjeni NPM.
- Jednosmjerni tok podataka za brzinu i modularnost podataka u aplikaciji.
- Otklanjanje pogrešaka kroz Google Chrome ekstenziju.[7]
- Komponente koje smo već ranije objasnili.

[6] Document Object Model – struktura grananja HTML dokumenta
[7] https://react.dev/learn/react-developer-tools

- Props: Props odnosno Properties (svojstva) omogućavaju korisniku da proslijedi argumente ili podatke s komponente na komponentu prikazano na Slika 7. Bitno je napomenuti kako se svojstva mogu samo čitati te se ne mogu mijenjati.

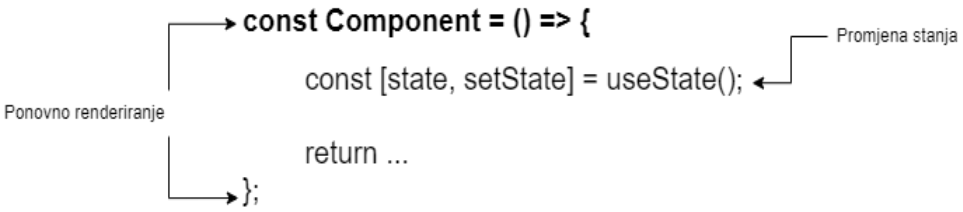
Slika 7: React.js props primjer



Izvor: *(07planning, 2014)*

- State odnosno stanje je objekt koji sprema vrijednosti komponenti kojima se kako samo ime nalaže promijenilo stanje. Stanje se može promijeniti kao rezultat akcija korisnika ili prilikom ponovnog renderiranja same aplikacije.

Slika 8: Primjer React.js state varijable



Izvor: *(developerway, 2022)*

2.6 DOCKER

Docker je platforma koja služi u svrhu izrade i lokalnog testiranja aplikacija. Bazirana je na tzv. Kontejnerima odnosno paketu softvera u kojem su sadržani svi elementi kao i kod potrebni za pokretanje. Ovdje se koristi za spajanje korisničkog sučelja s poslužiteljem.

Naravno postoji još mnogo načina povezivanja sučelja s poslužiteljem ali je odlučeno raditi preko Dockera jer ne zahtjeva nikakvo plaćanje te omogućava lokalno testiranje. Prvo što se treba napraviti je zapakirati cijeli server u kontejner u što je tema drugog rada. Ono što je bitno je zapravo kako preuzeti taj kontejner i namjestiti ga da funkcionira s web aplikacijom. Način na koji se to postiže je kreiranjem konfiguracijske datoteke pod imenom Docker-compose čija sintaksa se nalazi u Kod 5 (docker, 2023).

Kod 5: Docker konfiguracijska datoteka

```
version: "3.8"
services:
  backend:
    image: luka789/daily_todo:v1.0.0
    ports:
      - "8080:8080"
    depends_on:
      - database
    networks:
      - network-backend
    environment:
      -
SPRING_DATASOURCE_URL=jdbc:postgresql://database:5432/postgres
      - SPRING_DATASOURCE_USERNAME=postgres
      - SPRING_DATASOURCE_PASSWORD=postgres
  database:
    image: postgres:14.1-alpine
    environment:
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
      - POSTGRES_DB=postgres
      - POSTGRES_INITDB_ARGS="--encoding=UTF-8"
    ports:
      - "5432:5432"
    networks:
      - network-backend
    volumes:
      - ./db/db.sql:/docker-entrypoint-initdb.d/init.sql
```

```
networks:
  network-backend:
```

Prvo je definirana verzija Docker-a koja se koristi pa servisi. Pod servise se ubraja kontejner koji se pokreće, port poslužitelja na kojem se aplikacija vrti, ovisnost o bazi podataka, poslužiteljsko okruženje, parametre baze podataka koji su isti kao i za poslužitelja.

Nakon uspješne konfiguracije svih parametra virtualnog poslužitelja isti se pokreće s naredbom `docker-compose up` te se na taj način može lokalo povezati korisničko sučelje s poslužiteljem i bazom podataka.

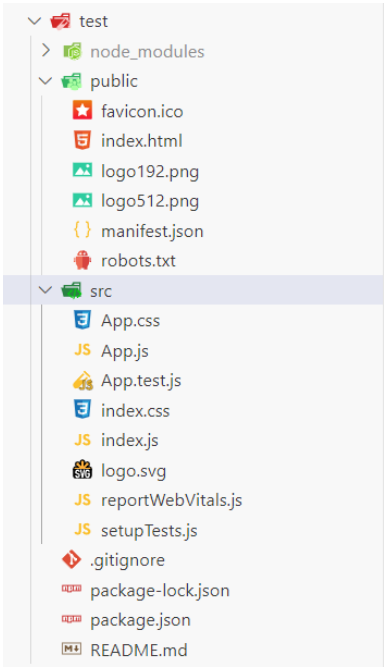
3. IZRADA TODO WEB APLIKACIJE

3.1 DIZAJN WEB APLIKACIJE

Nakon opisa svih potrebnih tehnologija na kojima će se ova web aplikacija bazirat može se početi s samim procesom izrade. Prije svega potrebno je instalirati program za razvoj koda. Jedino što je bitno je to da podržava JavaScript sintaksu. U ovom slučaju se koristi Visual Studio Code koji je već sadrži podršku za JavaScript (code.visualstudio, 2023). Iduće što je potrebno je instalirati Node.js koji će omogućiti korištenje lokalnog servera za razvoj web aplikacije, instalaciju raznih paketa za bolju funkcionalnost te kreiranje predloška React aplikacije (nodejs, 2023).

Nakon završetka instalacije potrebnih programa kreira se mapa u direktoriju po želji. Zatim se ta mapa otvara u programu za razvoj koda i u terminalu se upisuje naredbu `npx create-react-app ./` koji kreira React aplikaciju u tom direktoriju te ima isto ime kao ranije kreirana mapa. Po završetku kreiranja predloška aplikacije dobiva se direktorij prikazan na Slika 9.

Slika 9: Direktorij React.js aplikacije



Prva je mapa pod imenom `node_modules` u kojoj se nalaze svi instalirani paketi. Ta se mapa u praksi skoro pa nikada ne mijenja jer se prilikom instalacije paketa kodovi sami tu spremaju. Iduće je mapa `public` u kojoj se često nalaze svi audio/vizualni sadržaji korišteni u izradi. Ono na što treba obratiti posebnu pažnju je datoteka pod imenom `index.html`. Ona potkrepljuje zapravo ono ranije rečeno a to je da React zapravo kao samu srž koristi osnovni HTML gdje u tijelu dokumenta poziva skriptu što znači da se cijela React aplikacija pretvara u jednu JavaScript skriptu koja se poziva unutar HTML dokumenta prikazano u Kod 6.

Kod 6: Sintaksa Index.html datoteke

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-
scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta name="description" content="Web site created using
create-react-app" />
```

```
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png"
/>
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this
app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

Iduća je `src` mapa. U njoj se nalaze sve komponente i stranice koje jedna web aplikacija sadrži. Ono što je bitno naglasiti je to da jedino što ostaje su `App` i `Index.js` te `App` i `Index.css`. `App.js` je datoteka koju kompajler odnosno `Node.js` u ovom primjeru uzima kao polazišnu točku te se tu obično nalaze navigacijske rute ili komponente koje se nalaze globalno na razini same aplikacije prikazano na slici ispod. `Index.js` služi kao premosnica između `App.js-a` i `Index.html-a` na način da cijelu `App.js` datoteku renderira u jedan HTML element pod imenom `root` te taj element prosljeđuje u `Index.html` datoteku prikazano u **Error! Reference source not found.**. Ostale datoteke s e kstenzijom `.css` su CSS datoteke koje se inače nalaze u običnim HTML stranicama.

Kod 7: Sintaksa Index.js datoteke

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode> <App /> </React.StrictMode>
);

reportWebVitals();
```

Kod 7 prikazuje sintaksu `Index.js` datoteke. Na vrhu se nalaze import naredbe kojima React uvozi različite module, komponente i stranice. Zadnja je `root` varijabla u koja kreira `'root'` HTML element kojim zapravo renderira App datoteku.

Kod 8: Sintaksa App.js datoteke

```
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>Edit <code>src/App.js</code> and save to reload.</p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        > Learn React </a>
      </header>
    </div>
  );
}

export default App;
```

Kod 8 prikazuje sintaksu `App.js` datoteke. Na vrhu kao i na prethodnoj slici se nalaze import naredbe koje se zapravo nalaze u svakoj `.js` datoteci. Iduće je funkcija koja vraća HTML elemente. Time se potkrepljuje ranije rečena izjava da je React zapravo spojnica HTML-a i JavaScript-a jer povezuje ono najbolje od oba svijeta. I ono zadnje je naredba za izvoz (engl. *export*) funkcije koja omogućava da se ta ista funkcija uveze u ostale datoteke.

Od ostalih datoteka koje se nalaze u direktoriju predloška jedine koje je bitno spomenuti

su `package-lock.json` i `package.json` koje prikazuju verzije modula i paketa koji se koristi a čija je sintaksa prikazana u Kod 9.

Kod 9: Sintaksa package.json datoteke

```
{
  "name": "test",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.17.0",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
```

```
        "last 1 chrome version",
        "last 1 firefox version",
        "last 1 safari version"
      ]
    }
}
```

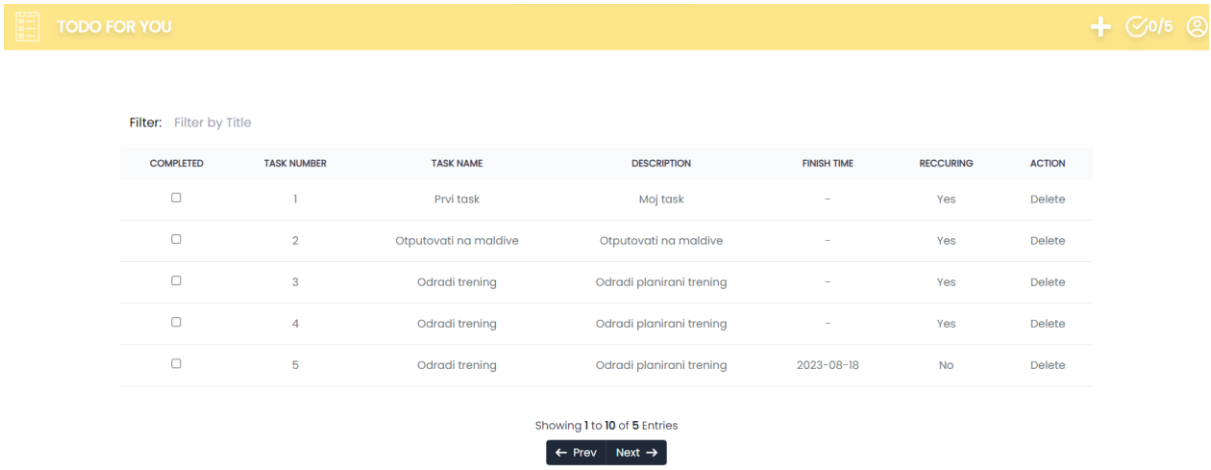
Slika 10 prikazuje zapravo cijeli taj direktorij na internetskom pregledniku.

Slika 10: Testna aplikacija u internetskom pregledniku

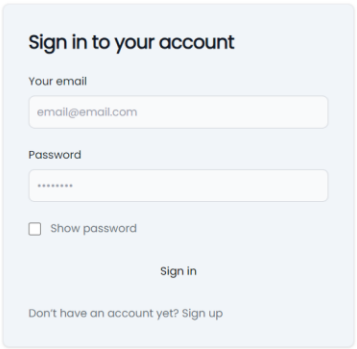
Napokon se može krenuti u izradu web aplikacije. Prvo što se treba je izbrisati sve datoteke i mape koje nisu potrebne tako da internetski preglednik prikazuje bijeli ekran. Zatim treba pronaći dizajn koji je javno dostupan i besplatan ili osmisliti vlastiti. Za nekog tko se prvi put susreće s razvojem web aplikacija kao takvim izrazito je teško osmisliti dizajn pa će stoga dizajn biti preuzet s neke stranice i modificiran sukladno potrebama. Dizajn će se temeljiti na dizajnu pronađenom na stranici dribbble.com (dribbble, 2023). Za razliku od ovog dizajn na kojem će se temeljiti ova TODO web aplikacija sadržavat se zaglavlje stranice u kojem će biti logo aplikacije, gumb za dodavanje zadataka gdje će se na klik otvarati prozor u kojem ćemo definirati parametre pojedinog zadatka, prikaz koliko zadataka je napravljeno u odnosu koliko ih trebamo napraviti i ikona profila gdje će korisnik klikom na nju otvoriti padajući izbornik. Za glavni dio web aplikacije nalazit će se tablica s pregledom svih zadataka i odgovarajućih svojstava (da li je izvršen, redni broj, ime zadatka, kategorija, završno vrijeme i gumb za brisanje zadatka) i tražilicu gdje će korisnik imati mogućnost pretrage zadataka prema imenu te mogućnost prelaska na druge stranice kao što je prikazano na Slika 11. Razlog toga je taj da se poslužitelj puno ne opterećuje prilikom dohvaćanja podataka i to što korisniku nisu odmah

potrebni svi podaci kao i stranice za registraciju prikazanu na Slika 12 i za prijavu na Slika 13.

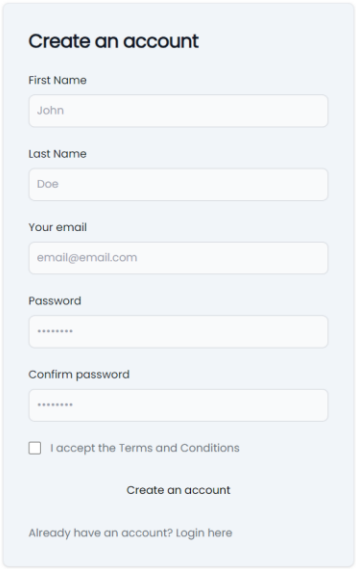
Slika 11: Početna stranica web aplikacije



Slika 12: Ekran za prijavu na web aplikaciju



Slika 13: Stranica za registraciju na web aplikaciju



Nakon opisa elemenata dizajna može se krenuti u izradu istih. Za jednostavnost izrade koristiti će se komponente koje su besplatne i javno dostupne. Za svaku komponentu će se iz službene dokumentacije opisati kako radi[8] te kako je ukomponirati u dizajn.

Započinje se s zaglavljem aplikacije. U lijevom kutu se nalazi logo aplikacije koji sadrži sliku i tekst dok a na desnoj gumb za dodavanje zadataka, prikaz broja riješenih zadataka u odnosu na one koje treba napraviti i profilna slika korisnika. Ikone su iz paketa React Icons koji sadrži široku paletu ikona svih dizajna i čiji će način rada biti također kasnije objašnjen.

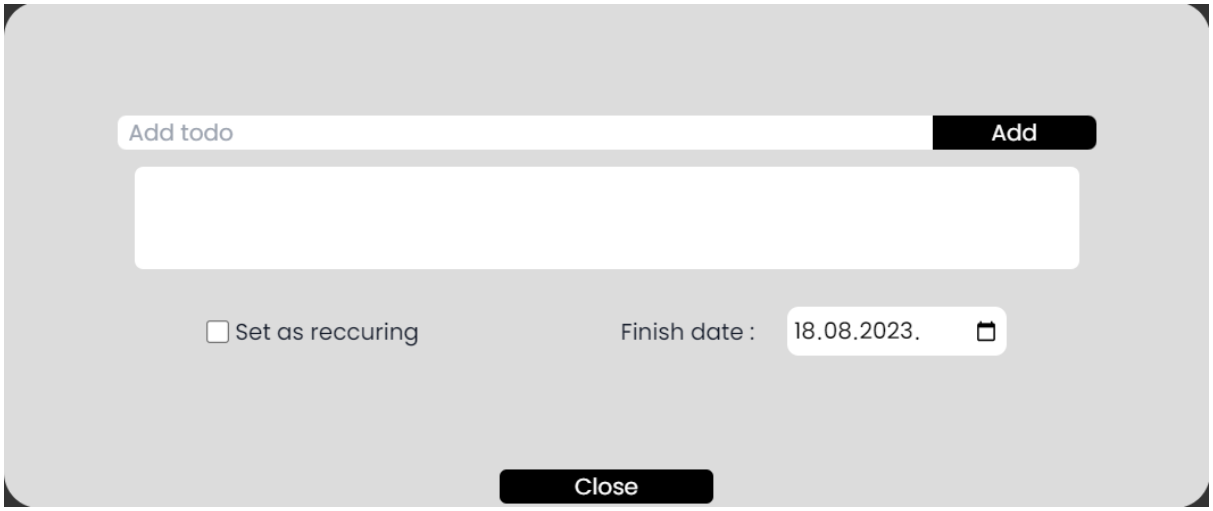
Slika 14: Zaglavlje web aplikacije

Što se tiče logoa to se dodaje na klasični način kao što bi u HTML-u što nije potrebno detaljno objašnjavati. Ono na što treba obratiti posebnu pozornost su ostali elementi zaglavlja. Gumb za dodavanja zadatka otvara modal u kojem korisnik upisuje svojstva zadatka. Modal je rađen koristeći ChakraUI komponentu. Što se tiče elemenata koje modal sadrži, a to su forma za unos teksta i opis zadatka, gumb gdje se zadatak može postaviti kao ponavljajući i datum

[8] Njihova dokumentacija će se nalaziti u idućem poglavlju

kada se će se zadatak biti završen. Za datum se koristi osnovni HTML input element s tipom date odnosno datum a za ponavljajući zadatak HTML input s tipom potvrdni okvir (engl. *checkbox*).

Slika 15: Modal za dodavanje zadatka



Primjer koda na kojem se element za dodavanje zadataka zasniva nalazi se u Kod 10.

Kod 10: Chakra UI modal

```
function BackdropExample() {
  const OverlayOne = () => (
    <ModalOverlay
      bg='blackAlpha.300'
      backdropFilter='blur(10px) hue-rotate(90deg)'/>
  )

  const OverlayTwo = () => (
    <ModalOverlay
      bg='none'
      backdropFilter='auto'
      backdropInvert='80%'
      backdropBlur='2px'/>
  )

  const { isOpen, onOpen, onClose } = useDisclosure()
```

```
    const [overlay, setOverlay] = React.useState(<OverlayOne />)

    return (
      <>
        <Button
          onClick={() => {
            setOverlay(<OverlayOne />)
            onOpen()
          }}> Use Overlay one </Button>
        <Button
          ml='4'
          onClick={() => {
            setOverlay(<OverlayTwo />)
            onOpen()
          }}> Use Overlay two </Button>
        <Modal isCentered isOpen={isOpen} onClose={onClose}>
          {overlay}
          <ModalContent>
            <ModalHeader>Modal Title</ModalHeader>
            <ModalCloseButton />
            <ModalBody>
              <Text>Custom backdrop filters!</Text>
            </ModalBody>
            <ModalFooter>
              <Button onClick={onClose}>Close</Button>
            </ModalFooter>
          </ModalContent>
        </Modal>
      </>
    )
}
```

Izvor: (chakra-ui/modal, 2023)

Kao što se može vidjeti u tablici CharkaUI komponenta već sadrži neke svoje elemente čime se znatno smanjuje vrijeme potrebno za izradu projekta. Ono što je potrebno je samo kopirati kod s interneta i prilagoditi ga prema potrebi (chakra-ui/modal, 2023).

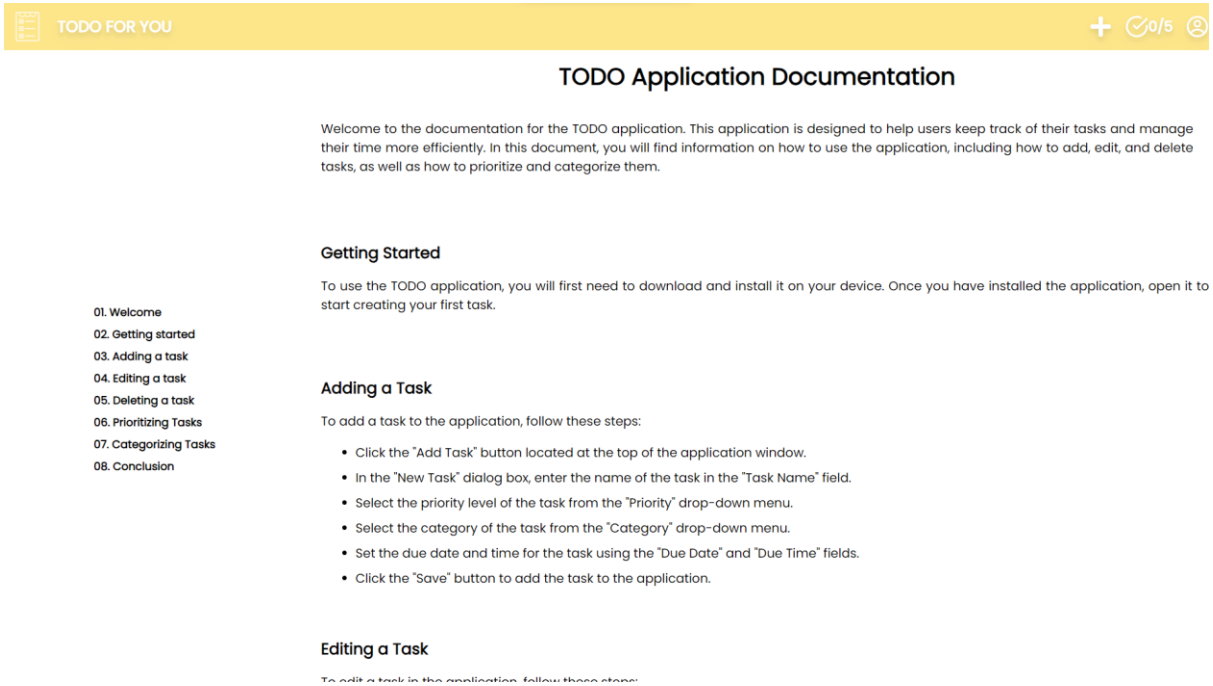
Iduće je pregled koliko je odrađenih zadataka u odnosu na stvorene. Što se toga tiče zasad su tu samo testni podatci koji će kasnije nakon spajanja na poslužitelj dobiti svoju srž. Kreiraju se pomoću običnih HTML oznaka te na njih ne treba obratiti neku posebnu pozornost.

I ono zadnje u zaglavlju je ikona profilne slike gdje se opet klikom na nju otvara padajući izbornik. U padajućem izborniku koji je napravljen koristeći ChakraUI korisnik ima mogućnost odjaviti se ili otići u pomoćni dio stranice koji sadrži često postavljana pitanja kao i dokumentaciju web aplikacije.

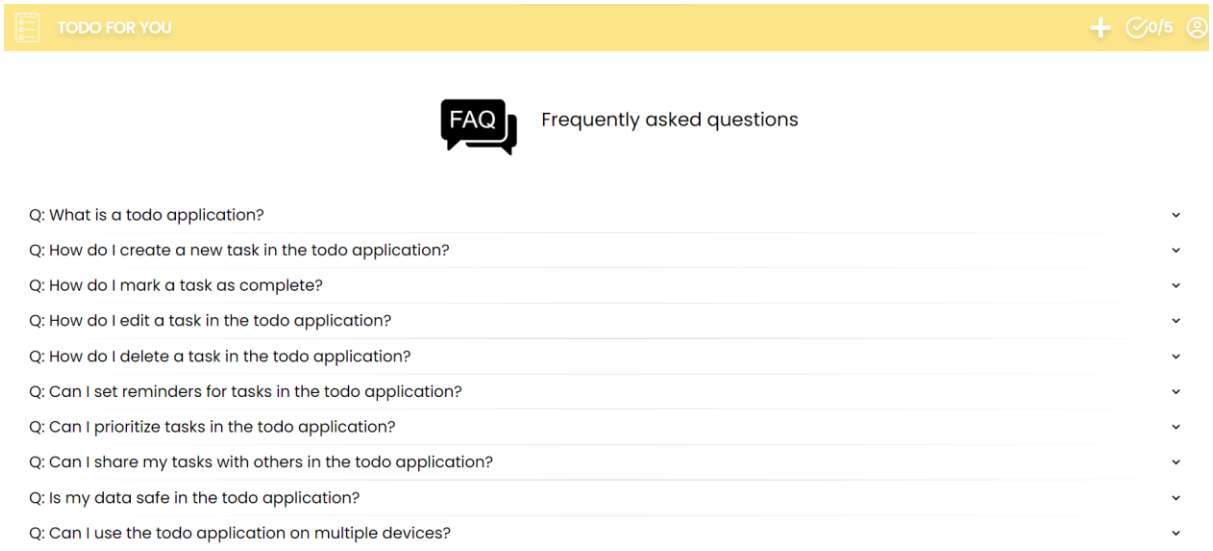
Slika 16: Padajući izbornik web aplikacije

Za tekst često postavljanih pitanja i dokumentacije korišten je ChatGPT iz razlog što postoje stranice koje izrađuju stranice s često postavljanim pitanjima i dokumentaciju ali često zahtijevaju punu domenu stranice, ime pravnog subjekta i ostale pravne informacije tako da u ovom slučaju to nije moguće koristiti. Dizajn ovih dviju stranica je napravljen proizvoljno bez korištenja ostalih dizajna kao referenci (chat.openai.com/često postavljana pitanja, 2023) (chat.openai.com/dokumentacija, 2023).

Slika 17: Dokumentacija web aplikacije



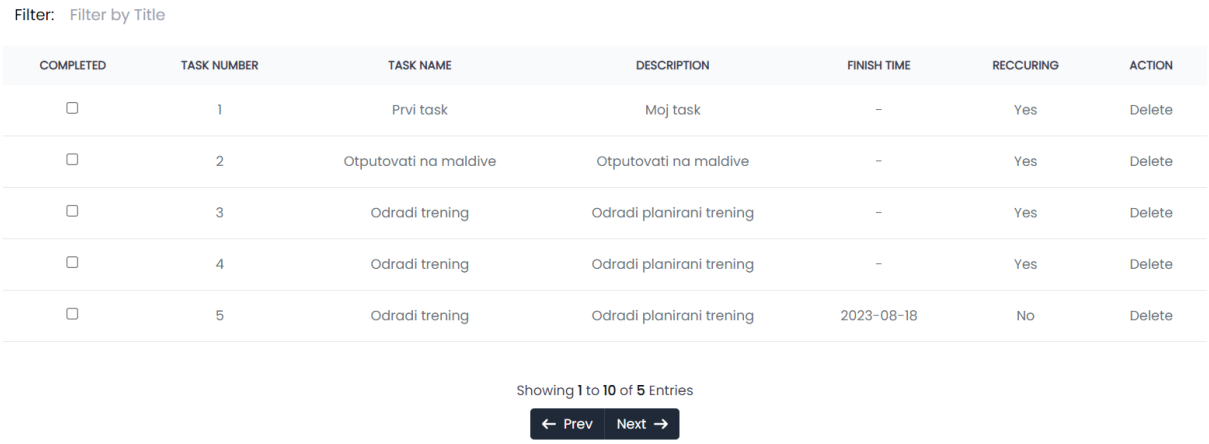
Slika 18: Često postavljana pitanja



Može se preći na glavni dio stranice u kojem kao što je već ranije spomenuto sadrži tražilicu i tablicu s popisom svih zadataka. Tablica sadrži sve zadatke koji su spremljeni u bazi podataka te tražilicu za pretragu po imenu zadatka. Dizajn tablice kao i tražilice su opet proizvoljni. Tablica osim što sadrži svojstva zadataka ima mogućnost brisanja pojedinog

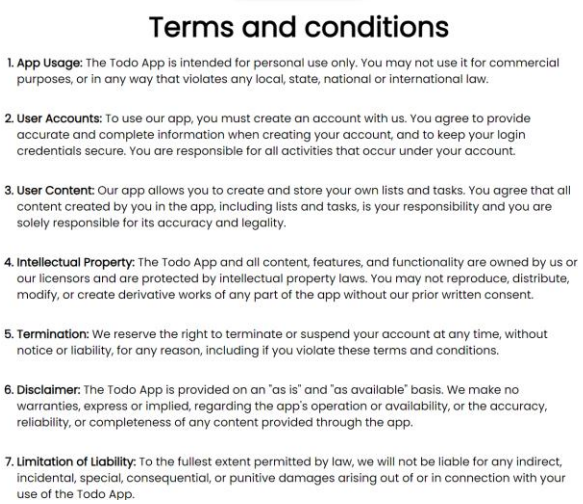
zadatka i završavanja istog.

Slika 19: Tablica s zadacima



Ono što je još ostalo je pojasniti stranicu za prijavu i registraciju. Dizajni su opet proizvoljni te jedino što je bitno napomenuti je to da na stranici za prijavu korisnik ima mogućnost prikazati ili sakriti svoju lozinku za lakšu prijavu dok na stranici za registraciju korisnik može vidjeti odredbe i uvjete. Za stranicu za odredbe i uvjete korišten je ChatGPT iz istih razloga koji su navedeni prethodno (chat.openai.com/odredbe i uvijeti, 2023).

Slika 20: Odredbe i uvjeti



Terms and conditions

1. **App Usage:** The Todo App is intended for personal use only. You may not use it for commercial purposes, or in any way that violates any local, state, national or international law.
2. **User Accounts:** To use our app, you must create an account with us. You agree to provide accurate and complete information when creating your account, and to keep your login credentials secure. You are responsible for all activities that occur under your account.
3. **User Content:** Our app allows you to create and store your own lists and tasks. You agree that all content created by you in the app, including lists and tasks, is your responsibility and you are solely responsible for its accuracy and legality.
4. **Intellectual Property:** The Todo App and all content, features, and functionality are owned by us or our licensors and are protected by intellectual property laws. You may not reproduce, distribute, modify, or create derivative works of any part of the app without our prior written consent.
5. **Termination:** We reserve the right to terminate or suspend your account at any time, without notice or liability, for any reason, including if you violate these terms and conditions.
6. **Disclaimer:** The Todo App is provided on an "as is" and "as available" basis. We make no warranties, express or implied, regarding the app's operation or availability, or the accuracy, reliability, or completeness of any content provided through the app.
7. **Limitation of Liability:** To the fullest extent permitted by law, we will not be liable for any indirect, incidental, special, consequential, or punitive damages arising out of or in connection with your use of the Todo App.

3.1.1 RESPONZIVNI DIZAJN

Budući da u današnje vrijeme ljudi sve više koriste mobilne uređaje do te mjere da čak

zamijene u potpunosti računalo potrebno je web aplikaciju učiniti preglednu na uređajima različite veličine. Način na koji se to radi je preko CSS-a preko naredbe `@media max-width`. Ono što ta naredba označava je set stilova koji vrijede samo do navedene širine ekrana. Kao što je već ranije rečeno ovaj rad za CSS koristi Tailwind.css pa je stoga način izrade malo drugačiji. Tailwind.css je CSS koji je namijenjen za mobilne uređaje (engl. *Mobile-first*) što znači da svi stilovi koje napišemo vrijede za mobilne uređaje dok za uređaje veće širine možda neće vrijediti. Način na kojima se postiže pravilan dizajn na uređajima veće širine je preko prijelomne točke (engl. *breakpoint*) (tailwindcss/responsive-design, 2023). Postoji pet prijelomnih točki:

Tablica 1: Responzivni dizajn tailwind.css-a

Prijelomna točka	Širina zaslona(px)	Minimalna širina
sm	640px	@media(min-width: 640px)
md	768px	@media(min-width: 768px)
lg	1024px	@media(min-width:1024px)
xl	1280px	@media(min-width:1280px)
2xl	1536px	@media(min-width:1536px)

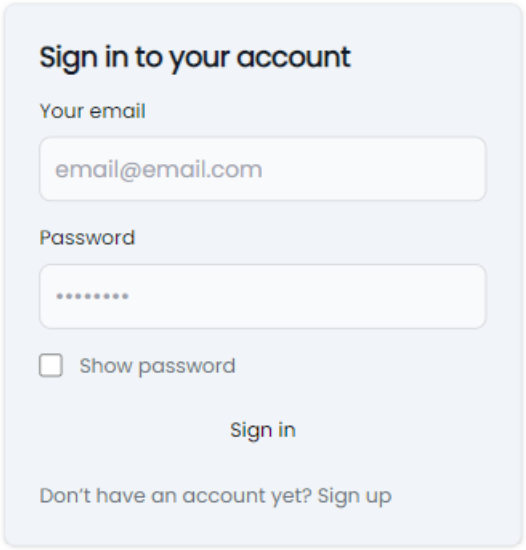
Izvor: (tailwindcss/responsive-design, 2023)

Kod 11 prikazuje prijelomne točke u kodu dok Slika 21 prikazuje stranicu za prijavljivanje na web aplikaciju na ekranu mobilnog uređaja.

Kod 11: CSS mobilni dizajn

```
<div className="inline-flex mt-2 xs:mt-0">
```

Slika 21: Ekran za prijavljivanje na mobilnom uređaju



3.2.1 KORIŠTENE KOMPONENTE

3.2.1.1 CHAKRA UI

Chakra UI je jednostavna, modularna i prilagodljiva biblioteka komponenti osnovne dijelove potrebne za izradu React web aplikacije. Posjeduje veliku prilagodljivost te je pogodna za svijetlu ili tamnu temu web stranice.

Sami proces instalacije je vrlo jednostavan. Prvo treba u terminalu upisati naredbu `npm`

`i @chakra-ui/react @emotion/react @emotion/styled framer-motion`. Tom naredbom instaliraju se svi paketi potrebni za pravilno funkcioniranje elemenata i komponenti. Zatim u `index.js` datoteci treba uvesti komponentu kojom se omogućuje korištenje Chakra UI komponenti kao što je prikazano u Kod 12 (chakra-ui/getting-started, 2023).

Kod 12: Chakra UI instalacija

```
import * as React from 'react'

import { ChakraProvider } from '@chakra-ui/react'

function App() {
  return (
    <ChakraProvider>
      <TheRestOfYourApplication />
    </ChakraProvider>
  )
}
```

Izvor: (chakra-ui/getting-started, 2023)

3.2.1.2 REACT ICONS

React Icons je biblioteka s širokom paletom popularnih ikona. Kako bi se mogla koristit prvo je treba putem terminala instalirati naredbom `npm install react-icons`. Sami proces implementiranja ikona u web aplikaciju je jednostavan. Sve što je potrebno je uvesti ikonu po želji te je u kodu pozvati kao zasebnu komponentu kao što je prikazano u Kod 13 (react-icons.github, 2023).

Kod 13: Instalacija React Icons paketa

```
import { FaBeer } from 'react-icons/fa';

class Question extends React.Component {
  render() {
    return <h3> Lets go for a <FaBeer />? </h3>
  }
```

```
}
```

Izvor: (react-icons.github, 2023)

3.2 FUNKCIONALNOSTI WEB APLIKACIJE

U ovom podpoglavlju biti će opisane funkcionalnosti web aplikacije. Kako bi se ostvarile funkcionalnosti potrebno je ovu aplikaciju povezati na poslužitelja. Budući da je tema ovog rada izrada klijentske strane web aplikacije ovdje će se samo opisati povezivanje na poslužitelja.

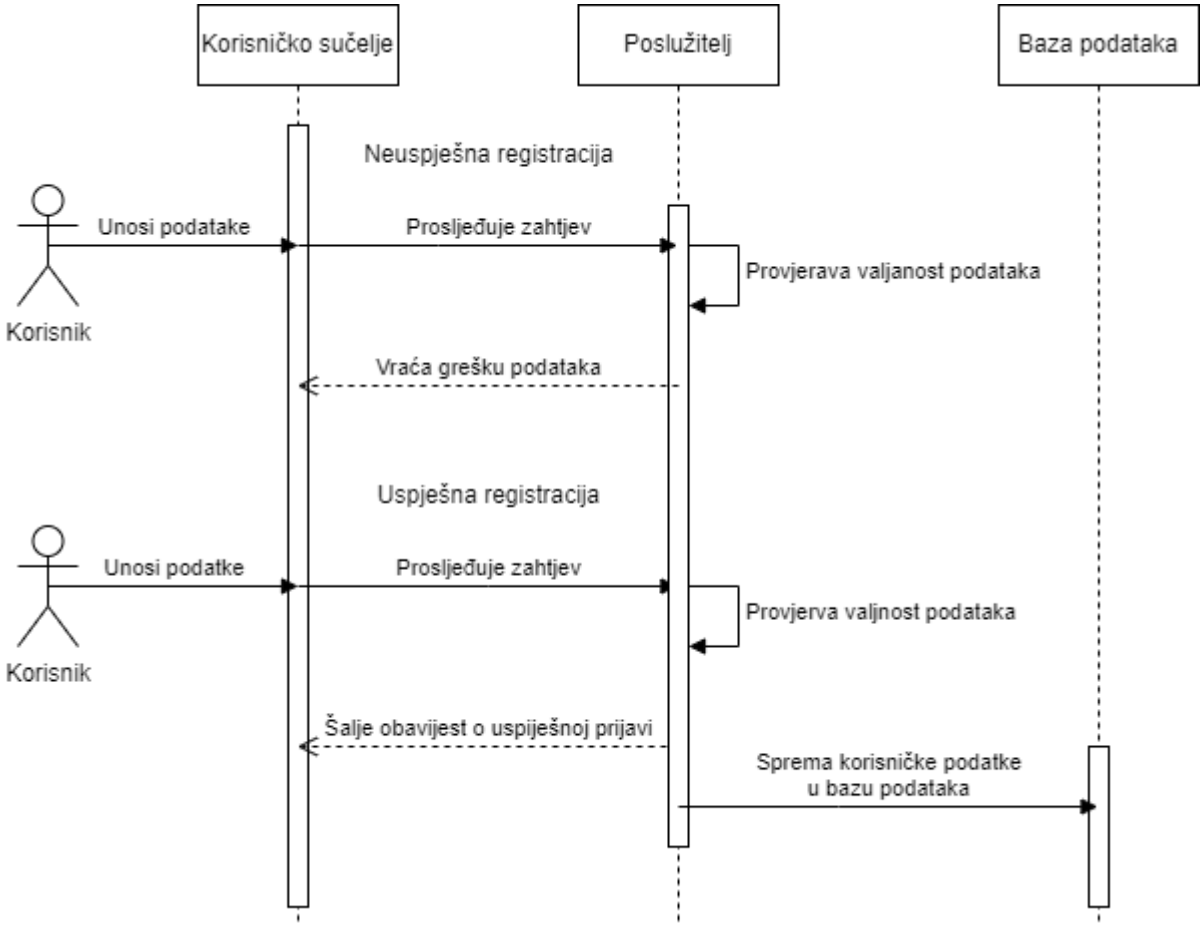
Za spajanje na poslužitelj koristi se programsko sučelje aplikacije odnosno API. API je mehanizam koji omogućava komunikacija između dvije softverske komponente (klijentska i serverska strana u ovom slučaju) pomoću skupa definicija i protokola. Način na koje rade je da aplikacija koja šalje zahtjev naziva se klijent, a aplikacija koja šalje odgovor poslužitelj. Postoje četiri vrste API-a ovisno o potrebi: SOAP API koji koristi XML za komunikaciju, RPC API za udaljenje procedure, Websocket API koji koristi JSON za prijenos podataka i REST API gdje se šalju samo podaci kao takvi. REST je kratica za Representational State Transfer te je to API kojeg ćemo koristiti u ovom radu. Definira skup funkcija kao što su GET, PUT, DELETE itd. koje klijenti mogu koristiti za pristup podacima poslužitelja. Klijenti i poslužitelji razmjenjuju podatke koristeći HTTP protokol (axios-http, 2023). Glavna značajka REST API-a je što nema stanja što znači da poslužitelji ne spremaju klijentske podatke između zahtjeva. Zahtjevi klijenta poslužitelju slični su URL-ovima koji se upisuju u preglednik za posjetu bilo koje web stranice. Odgovor s poslužitelja su obični podaci, bez tipičnog grafičkog prikaza web stranice (blog.hubspot, 2023).

Sada će se sve funkcionalisti programskog sučelja aplikacije prikazati na web aplikaciji. Prvo se kreće s registracijom korisnika pa prijava korisnika, prikaz svih zadataka, kreiranje zadataka, završavanje kao i brisanje istog.

3.2.1 REGISTRACIJA KORISNIKA

Registracija korisnika može se prikazati sekvencijskim dijagramom prikazanim na Slika 22**Error! Reference source not found.** prije uvoda u sami kod i objašnjavanja tehničkih podataka.

Slika 22: Sekvencijski dijagram registracije korisnika



Sekvencijski dijagram sadrži dva dijela odnosno neuspješnu i uspješnu registraciju na web aplikaciju. Neuspješna registracija se odnosi na to kada korisnik nije pravilno unio podatke ili već postoji takav korisnik. U tom slučaju poslužitelj vraća grešku na korisničko sučelje. Što se tiče uspješne registracije proces je isti osim što kada poslužitelj provjeri valjanost podataka ukoliko ne pronađe nikakve probleme šalje obavijest na korisničko sučelje kako je registracija uspješno obavljena te sprema te iste podatke u bazu podataka.

Kod 14: Funkcija za registraciju korisnika

```
const [firstName, setFirstName] = useState("");

const [lastName, setLastName] = useState("");

const [email, setEmail] = useState("");

const [password, setPassword] = useState("");

const registerUser = (event) => {

  event.preventDefault();

  const params = {

    firstName: firstName,

    lastName: lastName,

    email: email,

    password: password,

  };

  axios

    .post("/user/register", params)

    .then((res) => {

      console.log(res.data);

      alert("User succesfully registered!");

    })

    .catch((e) => {

      console.log(e);

    });

};
```

Kod 14 prikazuje funkciju za registraciju korisnika. Prije kreiranja same funkcije potrebno je prvo deklarirati varijable stanja kojima zapisujemo trenutno stanje u bazu podataka. Zatim se kreira funkcija koja uzima parametre početnih vrijednosti te ih zajedno s rutom koju je dodijelio poslužitelj i `POST` zahtjevom sprema u bazu podataka.

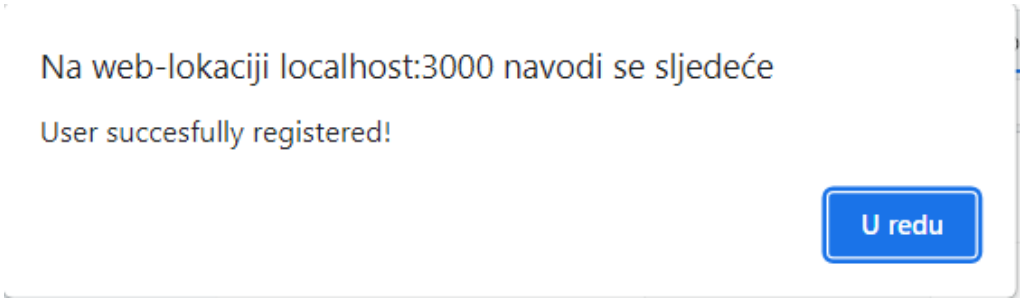
Kod 15: HTML forma za registraciju korisnika

```
<form className="space-y-4 md:space-y-6" onSubmit={registerUser}>
            <div>
              <input
                type="text"
                className="bg-gray-50 border border-gray-300 text-
gray-900     sm:text-sm     rounded-lg     focus:ring-primary-600
focus:border-primary-600 block w-full p-2.5"
                placeholder="John"
                required
                value={firstName}
                onChange={(e) => setFirstName(e.target.value)}
              />
            </div>
            <div>
              <input
                type="Last Name"
                className="bg-gray-50 border border-gray-300 text-
gray-900     sm:text-sm     rounded-lg     focus:ring-primary-600
focus:border-primary-600 block w-full p-2.5"
                placeholder="Doe"
                required
                value={lastName}
                onChange={(e) => setLastName(e.target.value)}
              />
            </div>
            <div>
              <input
                type="email"
                className="bg-gray-50 border border-gray-300 text-
gray-900     sm:text-sm     rounded-lg     focus:ring-primary-600
```

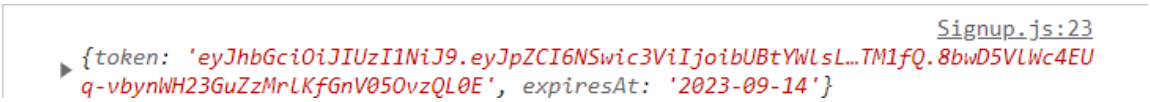
```
focus:border-primary-600 block w-full p-2.5"
                placeholder="email@email.com"
                required
                value={email}
                onChange={(e) => setEmail(e.target.value)}
              />
            </div>
            <div>
              <input
                type="password"
                placeholder="••••••••"
                className="bg-gray-50 border border-gray-300 text-
gray-900     sm:text-sm     rounded-lg     focus:ring-primary-600
focus:border-primary-600 block w-full p-2.5"
                required
                value={password}
                onChange={(e) => setPassword(e.target.value)}
              />
            </div>
            <div>
              <input
                type="password"
                placeholder="••••••••"
                className="bg-gray-50 border border-gray-300 text-
gray-900     sm:text-sm     rounded-lg     focus:ring-primary-600
focus:border-primary-600 block w-full p-2.5"
                required
                autoComplete="confirm-password"
              />
            </div>
            <button
              type="submit"
              className="w-full     text-black     bg-primary-600
hover:bg-primary-700  focus:ring-4  focus:outline-none  focus:ring-
primary-300 font-medium rounded-lg text-sm px-5 py-2.5 text-center"
            > Create an account </button>
</form>
```

Kod 15 prikazuje HTML formu za registraciju korisnika. Forma se kreira `<form>` elementom koji `onSubmit` događajem šalje zahtjev na poslužitelj. Također, forma sadrži `<input>` elemente koji su zapravo tekstualni okviri koji za početnu vrijednost uzimaju početne vrijednosti varijabli stanja te ih `onChange` događaj pretvaraju u završne vrijednosti. Ono što taj događaj zapravo radi je pretvara korisnikov unos u završnu vrijednost. Ono što je još bitno je gumb koji za je `submit` tipa. To se radi iz razloga da se gumb može povezati s formom. Ukoliko je zahtjev uspješan i poslužitelj nije naišao na nikakve probleme korisnik dobiva obavijest kako je registracija uspješna prikazano na Slika 23 te se generira novi objekt prikazan u konzoli prikazano na Slika 24.

Slika 23: Obavijest prilikom uspješne registracije



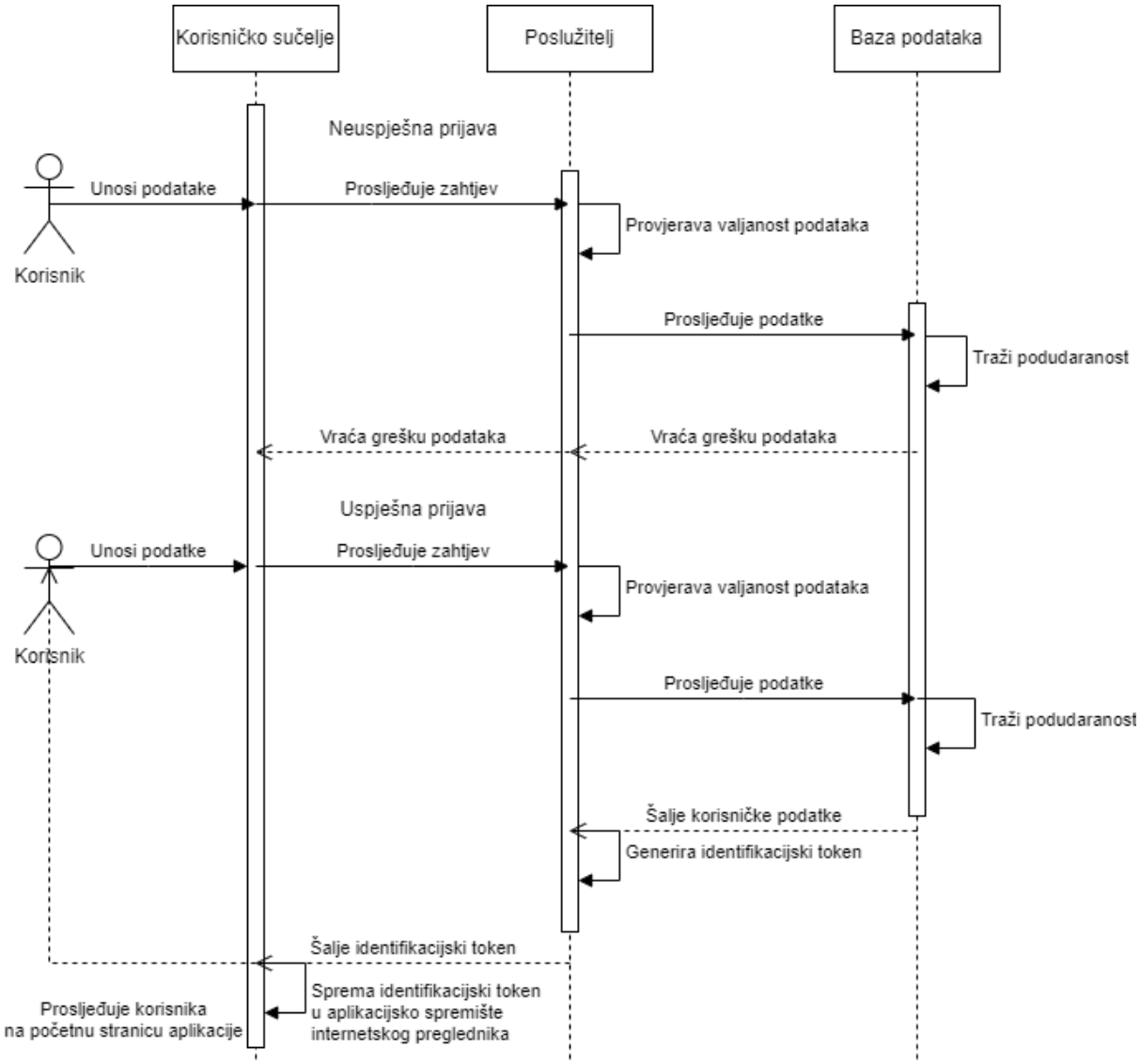
Slika 24: Novonastali objekt u bazi podataka



3.2.2 PRIJAVA KORISNIKA

Proces prijave korisnika je sličan registraciji korisnika samo što ovdje poslužitelj uspoređuje unesene podatke s podatcima iz baze podataka.

Slika 25: Sekvencijski dijagram prijave na web aplikaciju



Sekvencijski dijagram kao i prethodni prikazuje neuspješnu i uspješnu prijavu. Neuspješna prijava se događa kada korisnik unese krivo korisničko ime ili lozinku ili korisnik jednostavno ne postoji te tada dobije obavijest kako se ne može izvršiti prijava. Kod uspješne prijave korisnik unosi podatke koji se prosljeđuju na poslužitelj. Poslužitelj zatim provjerava valjanost tih podataka te pregledava bazu podataka tražeći podudaranja. Kada pronađe podudaranost generira identifikacijski token koji prosljeđuje korisničkom sučelju koji ga zatim sprema u aplikacijsko spremište internetskog preglednika.

Kod 16: Funkcija za prijavu korisnika

```
const [passwordShown, setPasswordShown] = useState(false);
const togglePassword = () => {
```

```
    setPasswordShown(!passwordShown);
  };
  const navigate = useNavigate();
  const [email, setEmail] = useState();
  const [password, setPassword] = useState();
  const login = (e) => {
    e.preventDefault();
    const params = {
      email: email,
      password: password,
    };
    axios
      .post("/user/login", params)
      .then((res) => {
        if (res.data.token) {
          localStorage.setItem("token", res.data.token);
          navigate("/");
          window.location.reload();
        } else {
          console.log(res.data);
        }
      })
      .catch((e) => {
        console.log(e);
      });
  };
```

Kod 16 prikazuje funkciju za prijavu korisnika na web aplikaciju. Funkcija je u pravilu ista kao i za registraciju osim što je sada potrebno samo deklarirati samo varijable za email i lozinku budući da se prilikom slanja zahtjeva samo pregledava baza podataka za podacima koji su unijeti. Jedina novost je ta što prilikom slanja zahtjeva generira identifikacijski token koji se sprema u aplikacijsko spremište internetskog preglednika koji će biti kasnije potreban.

Kod 17: HTML forma za prijavu korisnika

```
<form className="space-y-4 md:space-y-6" onSubmit={login}>
            <div>
```

```
              <input
                type="email"
                className="bg-gray-50 border border-gray-300 text-
gray-900     sm:text-sm     rounded-lg     focus:ring-primary-600
focus:border-primary-600 block w-full p-2.5"
                placeholder="email@email.com"
                required
                onChange={(e) => setEmail(e.target.value)}
              />
            </div>
            <div>
              <input
                type={passwordShown ? "text" : "password"}
                placeholder="••••••••"
                className="bg-gray-50 border border-gray-300 text-
gray-900     sm:text-sm     rounded-lg     focus:ring-primary-600
focus:border-primary-600 block w-full p-2.5"
                required
                onChange={(e) => setPassword(e.target.value)} />
            </div>
            <div className="flex items-center justify-between">
              <div className="flex items-start">
                <div className="flex flex-row items-center h-5">
                  <input
                    type="checkbox"
                    className="w-4   h-4   border   border-gray-300
rounded bg-gray-50 focus:ring-3 focus:ring-primary-300"
                    onClick={togglePassword} />
                  <div className="ml-3 text-sm">
                    <label   htmlFor="remember"   className="text-
gray-500"> Show password </label>
                  </div>
                </div>
              </div>
            </div>
            <button
              type="submit"
```

```
              className="w-full bg-primary-600 hover:bg-primary-
700 focus:ring-4 focus:outline-none focus:ring-primary-300 font-
medium rounded-lg text-sm px-5 py-2.5 text-center"
            > Sign in </button>
          </form>
```

Kod 17 prikazuje HTML formu za prijavu korisnika koja je skoro pa identična onoj za registraciju korisnika. Ukoliko je zahtjev uspješan i poslužitelj je pronašao korisničke podatke u bazi podataka korisnik se preusmjerava na početnu stranicu aplikacije te se generira identifikacijski token koji se sprema u aplikacijsko spremište internetskog preglednika prikazano na Slika 26.

Slika 26: Identifikacijski token u aplikacijskom spremištu internetskog preglednika

3.2.3 KREIRANJE ZADATKA

Kod 18: Funkcija za kreiranje zadataka

```
const [isCompleted] = useState(false);
const [finishTime, setFinishTime] = useState(moment(new Date()));
const [description, setDescription] = useState("");
const [taskGroup] = useState(null);
const [title, setTitle] = useState("");
const [recurring, setReccuring] = useState(false);
  const createTask = (event) => {
    event.preventDefault();
    const token = localStorage.getItem("token");
    const headers = {
      Authorization: `Bearer ${token}`,
    };
    const params = {
      title: title,
      isCompleted: isCompleted,
      description: description,
      taskGroup: taskGroup,
```

```
      recurring: recurring,
      finishTime: finishTime,
    };
    axios
      .post("/task/create", params, { headers })
      .then((res) => {
        console.log(res);
      })
      .catch((e) => {
        console.log(e);
      });
  };
  const handleCheckboxChange = () => {
    setReccuring(!recurring);
 };
```

Kod 18 prikazuje funkciju za kreiranje zadatka koje se vrši na isti način kao i registracija i prijava. Jedina novosti koja se uvodi je provjera autentifikacije korisnika. To se izvršava na način da se u funkciju uvodi naredba koja preuzima identifikacijski token iz aplikacijskog spremišta internetskog preglednika.

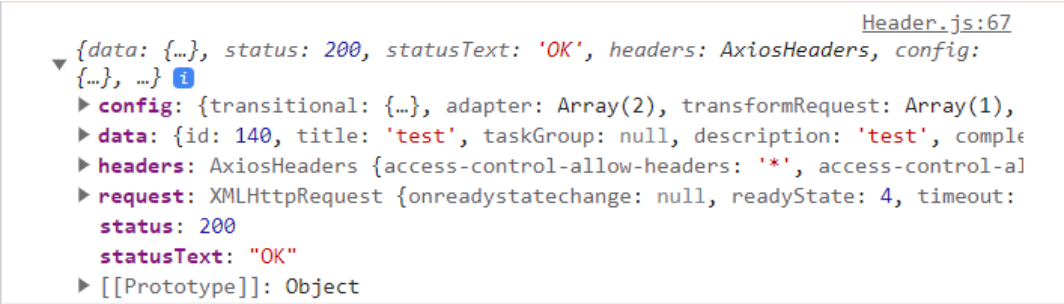
Kod 19: HTML forma za kreiranje zadatka

```
<form className="add-todo" onSubmit={createTask}>
                  <div className=" flex flex-row items-center">
                    <input
                      className=" w-10/12 rounded-tl-md rounded-
bl-md focus:outline-none pl-2"
                      placeholder="Add todo"
                      value={title}
                      onChange={(e) => setTitle(e.target.value)}
/>
                    <button className=" bg-black text-white w-2/12
rounded-tr-md rounded-br-md hover:bg-neutral-400"> Add </button>
                  </div>
                  <div className=" rounded-lg m-3">
                    <textarea
```

```
                        cols="80"
                        rows="3"
                        className=" pl-2 resize-none focus:outline-
none rounded-md"
                        value={description}
                        onChange={(e)                                  =>
setDescription(e.target.value)}
                      > Add description </textarea>
                    </div>
                    <div  className="  flex  flex-row  justify-around
items-center mt-5">
                      <div class="flex items-center mr-2">
                        <input
                          type="checkbox"
                          className="w-4 h-4 text-blue-600 bg-gray-
100 border-gray-300"
                          checked={recurring}
                          onChange={handleCheckboxChange} />
                      </div>
                      <p  className="ml-2  text-md  font-medium  text-
gray-900"> Finish date : </p>
                      <input
                        type="date"
                        onChange={(e)                                  =>
setFinishTime(e.target.value)}
                        className=" ml-[-15%] rounded-lg p-1"
                        min={formattedDate} />
                    </div>
                  </form>
```

Kod 19 prikazuje HTML formu za kreiranje zadatka koja je ista kao i one za registraciju i prijavu. Ukoliko su svi parametri zadovoljeni u konzoli vraća objekt s podacima zadatka prikazano na Slika 27.

Slika 27: Kreirani zadatak u konzoli



3.2.4 PRIKAZ ZADATKA

Kod 20: Funkcija za prikaz zadataka

```
const [data, setData] = useState([]);
const [currentPage, setCurrentPage] = useState(0);
const [filterTitle, setFilterTitle] = useState("");
 const pageSize = 10;
  const fetchData = async () => {
    const token = localStorage.getItem("token");
    const headers = {
      Authorization: `Bearer ${token}`,
    };
    try {
      const response = await axios.post(
        "/task/getTaskPage",
        {
          pageRequestData: {
            pageNumber: currentPage,
            pageSize: pageSize,
            sortBy: "title",
            sortDirection: 0,
          },
          taskFilterData: {
            category: null,
            status: null,
            finishTime: null,
          },
        },
```

```
        { headers }
      );
      const tasks = response.data.taskList;
      setData(tasks);
    } catch (error) {
      console.error("Error fetching data:", error);
    }
  };
  const handlePageChange = (newPage) => {
    setCurrentPage(newPage);
  };
  const filteredData = data.filter((task) =>
    task.title.toLowerCase().includes(filterTitle.toLowerCase())
 );
```

Kod 20 prikazuje funkciju za prikaz zadataka koji se odvija pomoću paginacije te funkcije za promjenu stranice paginacije i filtriranje zadataka prema imenu. Paginacija se vrši na način da se u `POST` zahtjev stavljaju parametri koje je dodjelo poslužitelj. Po obavljanju funkcije podaci se spremaju u prazno polje podataka. Iduće se funkcija za promjenu stranica koja u varijablu stanja ranije deklariranu sprema trenutni broj stranice s poslužitelja i funkcija za filtriranje prema imenu koja putem `.filter` metode pretražuje sve podatke sadržane u novonastalom polju.

Kod 21: Tablica za prikaz zadataka

```
<div className=" flex flex-row gap-4 ml-3 mb-5 mt-10">
            <p>Filter:</p>
            <input
              type="text"
              placeholder="Filter by Title"
              value={filterTitle}
              onChange={(event)                                    =>
setFilterTitle(event.target.value)}/>
 </div>
<table className=" w-full text-sm text-center text-gray-500">
            <thead className=" text-xs text-gray-700 uppercase bg-
```

```
gray-50">
              <tr>
                <th scope="col" className="px-6 py-3"> Completed
</th>
                <th scope="col" className="px-6 py-3"> Task Number
</th>
                <th scope="col" className="px-6 py-3"> Task Name
</th>
                <th scope="col" className="px-6 py-3"> Description
</th>
                <th scope="col" className="px-6 py-3"> Finish Time
</th>
                <th scope="col" className="px-6 py-3"> Reccuring
</th>
                <th scope="col" className="px-6 py-3"> Action </th>
              </tr>
            </thead>
            <tbody>
              {filteredData.map((task, index) => (
                <tr
                  key={task.id}
                  className={`bg-white border-b ${
                    checkedTasks.includes(task.id) ? "text-green-
300" : "" }`} >
                  <td className="px-6 py-4">
                    <input
                      className="w-4 h-4"
                      type="checkbox"
                      checked={task.isCompleted}
                      onChange={() => { markTaskCompleted(task,
task.isCompleted); }}
                      disabled={task.completed} /> </td>
                  <td className="px-6 py-4">{index + 1}</td>
                  <td className="px-6 py-4">{task.title}</td>
                  <td              className="px-6              py-
4">{task.description}</td>
                  <td className="px-6 py-4"> {task.recurring ? "-"
```

```
: formatDate(task.finishTime)} </td>
                  <td   className="px-6   py-4">{task.recurring   ?
"Yes" : "No"}</td>
                  <td className="px-6 py-4">
                    <button    type="button"    onClick={()    =>
handleDelete(task.id)}> Delete </button> </td>
                </tr>
              ))}
            </tbody>
          </table>
```

Kod 21 prikazuje tablicu za prikaz zadataka. Prvi dio koda sadrži tekstualni okvir za filtriranje koji je isti kao onaj za prijavu korisnika. Iduća je tablica koja prikazuje zadatke dohvaćene s poslužitelja. Dohvaćanje se vrši putem `.map` metode koja prikazuje elemente onoliko puta koliko ih pronađe u bazi podataka. Kako bi se pravilno prikazali potrebno je postaviti `key` atribut koji sadrži `id` zadatka te na mjestu prikazivanja podataka pozvati te iste podatke. Sintaksa za to je `{ ime podatka }`.

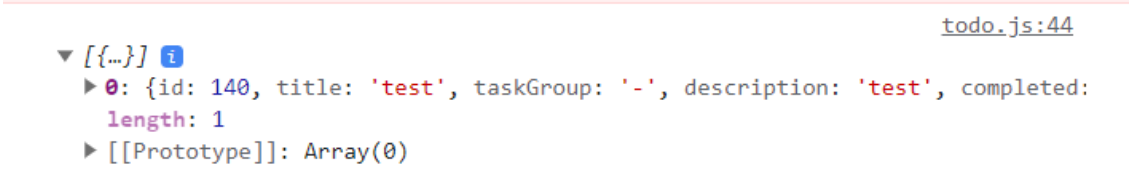
Kod 22: Paginacija web aplikacije

```
          <div className="flex flex-col items-center mt-10">
            <span className="text-sm text-gray-700">
              Showing <span className="font-semibold text-gray-
900">1</span> to{" "}
              <span className="font-semibold text-gray-
900">{pageSize}</span> of{" "}
              <span className="font-semibold text-gray-
900">{data.length}</span>{" "} Entries </span>
            <div className="inline-flex mt-2 xs:mt-0">
              <button className="flex items-center justify-center
px-3 h-8 text-sm font-medium text-white bg-gray-800 rounded-l
hover:bg-gray-900"
                onClick={() => handlePageChange(currentPage - 1)}
                disabled={currentPage === 0}
              >
                <svg
```

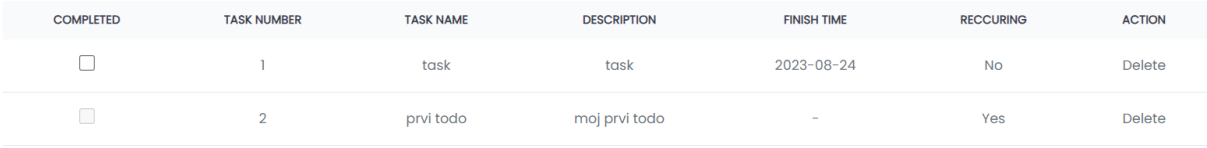
```
                    className="w-3.5 h-3.5 mr-2"
                    aria-hidden="true"
                    xmlns="http://www.w3.org/2000/svg"
                    fill="none"
                    viewBox="0 0 14 10" >
                    <path
                      stroke="currentColor"
                      strokeLinecap="round"
                      strokeLinejoin="round"
                      strokeWidth="2"
                      d="M13 5H1m0 0 4 4M1 5l4-4" /> </svg> Prev
</button>
              <button className="flex items-center justify-center
px-3 h-8 text-sm font-medium text-white bg-gray-800 border-0
border-l border-gray-700 rounded-r hover:bg-gray-900"
                onClick={() => handlePageChange(currentPage + 1)}
> Next
                <svg
                  className="w-3.5 h-3.5 ml-2"
                  aria-hidden="true"
                  xmlns="http://www.w3.org/2000/svg"
                  fill="none"
                  viewBox="0 0 14 10" >
                  <path
                    stroke="currentColor"
                    strokeLinecap="round"
                    strokeLinejoin="round"
                    strokeWidth="2"
                    d="M1 5h12m0 0L9 1m4 4L9 9" /> </svg>
</button>
            </div>
          </div>
```

Kod 22 prikazuje paginaciju web aplikaciju. Paginacija na web aplikaciji osim što sadrži funkciju sadrži i gumbe za povratak na prošlu odnosno odlazak na iduću stranicu.

Slika 28: Svi zadaci u konzoli



Slika 29: Zadaci u tablici



COMPLETED	TASK NUMBER	TASK NAME	DESCRIPTION	FINISH TIME	RECCURING	ACTION
☐	1	task	task	2023-08-24	No	Delete
☐	2	prvi todo	moj prvi todo	-	Yes	Delete

Slika 28 i Slika 29 prikazuju kako ovaj cijeli kod izgleda u konzoli internetskog preglednika i u tablici na sučelju web aplikacije.

3.2.5 OZNAČAVANJE ZADATKA KAO ZAVRŠENI

Kod 23: Kod za označavanje zadatka kao završeni

```
const markTaskCompleted = async (taskToUpdate) => {
  const token = localStorage.getItem("token");
  const headers = {
    Authorization: `Bearer ${token}`,
  };
  try {
    await axios.post(
      "/task/updateTask",
      {
        id: taskToUpdate.id,
        isCompleted: true,
        completed: true,
        recurring: taskToUpdate.recurring,
        taskGroup: null,
        description: null,
        finishTime: null,
        title: null,
      },
```

```
          { headers }
        );
        fetchData();
      } catch (error) {
        console.error("Error marking task as completed:", error);
      }
  };

//…ostatak koda
 <td className="px-6 py-4">
                  <input
                    className="w-4 h-4"
                    type="checkbox"
                    checked={task.isCompleted}
                    onChange={() => {
                      markTaskCompleted(task, task.isCompleted);
                    }}
                    disabled={task.completed}/> </td>
```

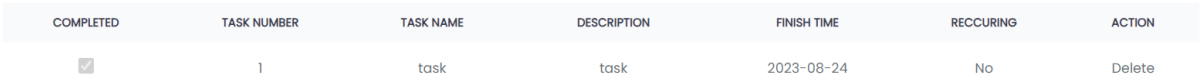
Kako bi se zadatak moga označiti kao završeni potrebni su parametri koje je zadao poslužitelj. Jedan od bitnih parametara je `id`. Razlog toga je taj što taj isti `id` nije zadan u nikakvoj ruti koju nam zadaje poslužitelj već je potrebno iz izvučenih podataka odnosno onih podataka iz prikaza zadataka pronaći `id`. To se radi da se putem `propsa` pristupa funkciji za prikaz zadataka te se iz njega izvlači `id` pojedinog zadatka. Kada cijeli zahtjev prođe potrebno je ažurirati prikazane zadatke jer ako se to ne napravi klijent će izbaciti grešku.

HTML element koji se koristi za ostvarivanje ove mogućnosti je input s tipom potvrdni okvir (engl. *checkbox*) koji za polje vrijednosti sadrži novu `isCompleted` vrijednost odnosno `true`, prilikom promjene uzima cijelu funkciju za označavanje te u nju stavlja zadataka i vrijednost iz polja vrijednost te mogućnost zabrane klikanja kada se funkcija izvrši.

Kada se to sve odradi dobiva se prikaz u internetskom pregledniku na Slika 30 te ažurirani objekt na Slika 31.

Slika 30: Završeni zadatak u konzoli

Slika 31: Završeni zadatak



3.2.6 BRISANJE ZADATKA

Kod 24: Kod za brisanje zadataka

```
  const handleDelete = async (id) => {
    const token = localStorage.getItem("token");
    const headers = {
      Authorization: `Bearer ${token}`,
    };
    try {
      await axios.delete(`/task/delete/${id}`, { headers });
    } catch (error) {
      console.error("Error deleting task:", error);
    }
  };

//…ostatak koda
 <td className="px-6 py-4">
                 <button      type="button"      onClick={()      =>
 handleDelete(task.id)}> Delete </button> </td>
```

Kreira se funkcija za brisanje pojedinog zadatka. Način na koji se kreira je da tražimo pojedini `id` zadatka te kao i u prethodnim slučajevima tražimo identifikacijski token radi sigurnosti. Prilikom klika na gumb briše se zadatak gdje se taj gumb nalazi te se opet poziva funkcija za prikaz zadataka kako bi se ažurirala početna stranica.

3.2.7 PRIKAZ OMJERA ZADATAKA

Kod 25: Kod za prikaz omjera zadataka

```
<p className="text-lg">
            <span className=" ml-1 mr-1">
              {data.filter((task) => task.completed ===
true).length} </span> /<span className=" ml-
1">{data.length}</span> </p>
```

Prikaz omjera završenih u odnosu na sve zadatke se radi na način da se za odrađene zadatke svi podaci filtriraju tako da ostanu svi oni kojima je parametar `completed === true` te se iz njih putem metode `.length` dobiva broj dok se za ukupne zadatke traži broj svih podataka što je prikazano na Slika 32.

Slika 32: Omjer odrađenih i svih zadataka

4. ZAKLJUČAK

Ovaj radi prikazuje izradu grafičkog sučelja TODO web aplikacije u programskom okviru React.js. React.js kao programski okvir je pogodan za početnike što je bio i jedan od razloga odabira baš njega za ovu web aplikaciju. Što je tiče samo procesa učenja na internetu je dostupno jako mnogo tečajeva i interaktivnih kvizova kao i github repozitorija tako je jedino što je zapravo potrebno je volja i želja. Cilj rada je ostvaren te aplikacija u potpunosti komunicira s poslužiteljem te bi se čak moglo reći da je ostvareno i malo više od zadanog što je uvijek jedan veliki plus. Nekakve smjernice za budućnost je naravno učiniti kod što čitkijim i modularnijim te stavljati još više funkcionalnosti u samu web aplikacije kako na samom sučelju tako i na poslužitelju. Moje mišljenje je to da je ovo bila dobra ali i pomalo zahtjevna tema za nekoga tko se prvi put susreće s ovim ali mi je drago što sam ovo odabrao i nastavit ću raditi i graditi ovu web aplikaciju i dalje

LITERATURA

(29. Lipanj 2011). Preuzeto 15. Kolovoz 2023 iz bussinesinsider: https://www.businessinsider.com/flashback-this-is-what-the-first-website-ever-looked-like-2011-6

(Ožujak 2014). Preuzeto 18. Kolovoz 2023 iz 07planning: https://o7planning.org/12125/react-props-state

(2020). Preuzeto 16. Kolovoz 2023 iz bu.edu: https://www.bu.edu/lernet/artemis/years/2020/projects/FinalPresentations/HTML/historyofhtml.html

(12. srpanj 2020). Preuzeto 17. Kolovoz 2023 iz medium: https://medium.com/@annguyenhieuduc/3-tailwindcss-vscode-extension-makes-your-life-easier-9067b065c4f0

(22. Srpanj 2022). Preuzeto 17. Kolovoz 2023 iz geekboots: https://www.geekboots.com/story/importance-of-javascript-in-web-development

(2. Kolovoz 2022). Preuzeto 18. Kolovoz 2023 iz developerway: https://www.developerway.com/posts/react-re-renders-guide

(18. Srpanj 2023). Preuzeto 16. Kolovoz 2023 iz devdocs: https://devdocs.io/html/

(2023). Preuzeto 16. Kolovoz 2023 iz w3schools: https://www.w3schools.com/html/tryit.asp?filename=tryhtml_basic_document

(18. Srpanj 2023). Preuzeto 17. Kolovoz 2023 iz developer.mozilla: https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS

(2023). Preuzeto 17. Kolovoz 2023 iz marketplace.visualstudio: https://marketplace.visualstudio.com/items?itemName=sudoaugustin.tailwindcss-transpiler

(2023). Preuzeto 17. Kolovoz 2023 iz tailwindcss/installation: https://tailwindcss.com/docs/installation

(1. Kolovoz 2023). Preuzeto 17. Kolovoz 2023 iz hostinger: https://www.hostinger.com/tutorials/what-is-javascript

(16. Svibanj 2023). Preuzeto 18. Kolovoz 2023 iz simplilearn.com/nodejs: https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-nodejs#what_is_nodejs

(2023). Preuzeto 18. Kolovoz 2023 iz bairesdev: https://www.bairesdev.com/react/react-js/

(7. Veljača 2023). Preuzeto 18. Kolovoz 2023 iz simplilearn/reactjs: https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs

(2023). Preuzeto 20. Kolovoz 2023 iz docker: https://www.docker.com/resources/what-container/

(2023). Preuzeto 20. Kolovoz 2023 iz code.visualstudio: https://code.visualstudio.com/

(2023). Preuzeto 20. Kolovoz 2023 iz nodejs: https://nodejs.org/en

(2023). Preuzeto 20. Kolovoz 2023 iz dribbble: https://dribbble.com/shots/19626767-Table-Component

(2023). Preuzeto 21. Kolovoz 2023 iz chakra-ui/modal: https://chakra-ui.com/docs/components/modal/usage

(2023). Preuzeto 21. Kolovoz 2023 iz chat.openai.com/često postavljana pitanja: https://chat.openai.com/c/1ef7a65f-893a-495c-97d6-04603fcfc6cb

(2023). Preuzeto 21. Kolovoz 2023 iz chat.openai.com/dokumentacija: https://chat.openai.com/c/8f5f1861-1a13-4cf2-8dda-4dae761f984e

(2023). Preuzeto 21. Kolovoz 2023 iz chat.openai.com/odredbe i uvijeti: https://chat.openai.com/c/9c8fd554-bebb-4abc-8e93-459c5d3487c3

(2023). Preuzeto 22. Kolovoz 2023 iz tailwindcss/responsive-design:

https://tailwindcss.com/docs/responsive-design
(2023). Preuzeto 23. Kolovoz 2023 iz chakra-ui/getting-started: https://chakra-ui.com/getting-started/cra-guide
(2023). Preuzeto 22. Kolovoz 2023 iz react-icons.github: https://react-icons.github.io/react-icons/
(2023). Preuzeto 23. Kolovoz 2023 iz axios-http: https://axios-http.com/docs/intro
(9. Kolovoz 2023). Preuzeto 23. Kolovoz 2023 iz blog.hubspot: https://blog.hubspot.com/website/what-is-rest-api

PRILOZI

Popis tablica

Popis slika

Popis kodova