Razvoj web aplikacije za rezervaciju sportskih termina

Kreuzer, Alen

VELEUČILIŠTE U BJELOVARU
STRUČNI PRIJEDIPLOMSKI STUDIJ RAČUNARSTVO

RAZVOJ WEB APLIKACIJE ZA REZERVACIJU SPORTSKIH TERMINA

Završni rad br. 11/RAČ/2023

Alen Kreuzer

Bjelovar, listopad 2023.

obrazac ZR - 001

Veleučilište u Bjelovaru

Trg E. Kvaternika 4, Bjelovar

1. DEFINIRANJE TEME ZAVRŠNOG RADA I POVJERENSTVA

Student: **Alen Kreuzer** JMBAG: **0314023098**

Naslov rada (tema): **Razvoj web aplikacije za rezervaciju sportskih termina**

Područje: **Tehničke znanosti** Polje: **Računarstvo**

Grana: **Informacijski sustavi**

Mentor: **Tomislav Adamović, mag. ing. el.** zvanje: **viši predavač**

Članovi Povjerenstva za ocjenjivanje i obranu završnog rada:

1. **Krunoslav Husak, dipl. ing. rač., predsjednik**
2. **Tomislav Adamović, mag. ing. el., mentor**
3. **Krešimir Markota, mag. ing. comp., član**

2. ZADATAK ZAVRŠNOG RADA BROJ: 11/RAČ/2023

U sklopu završnog rada potrebno je:

1. Dizajnirati korisničko sučelje u React programskom okviru namijenjeno rezervaciji sportskih termina na računalnoj platformi.
2. Implementirati obradu zahtjeva u Nest.js-u za upravljanje rezervacijama.
3. Integrirati Prisma ORM za interakciju s bazom podataka vezanom uz sportske termine.
4. Strukturirati model podataka za sportske termine unutar Postgres baze podataka.
5. Uvesti sustav autentifikacije i autorizacije korisnika unutar aplikacije za rezervaciju sportskih termina.
6. Postaviti aplikaciju za rezervaciju sportskih termina kao Docker kontejner.

Datum: 29.08.2023. godine

Mentor: **Tomislav Adamović, mag. ing. el.**

Zahvala

Zahvaljujem se svim profesorima sa veleučilišta u Bjelovaru koji su mi prenijeli svoje znanje kako bih bio u mogućnosti izrađivati ovakve aplikacije. Posebno se zahvaljujem svom mentoru Tomislavu Adamoviću.

Sadržaj

1. Uvod

U današnjem vremenu ne može se bez tehnologije, upotreba papira se u potpunosti smanjuje i pretvara u digitalni oblik. Velike koristi od tehnologije dobio je i sport. Programeri traže rješenja kako bi povezali tehnologiju i sport u jednu cjelinu. Web aplikacija za rezervaciju sportskih termina može biti korisna ljudima koji žele riješiti svoju rezervaciju u svega nekoliko pritisaka u aplikaciji.

Proces razvoja web aplikacije za rezervaciju sportskih termina će biti istražen u završnome radu. Istražit će proces projektiranja i implementaciju ideja pa sve do puštanja u rad aplikacije. Cilj ovog rada je pružiti dublji uvid u korištenje tehnologija i alata za razvoj aplikacije.

Drugo poglavlje ovog završnog rada odnosi se na opis tehnologija, rečeno je nešto malo više o svakoj tehnologiji se koristila u izradi aplikacije. Treće poglavlje opisuje o funkcionalnostima aplikacije, koje funkcionalnosti ima administrator, a koje sami korisnici. Sljedeće poglavlje će prikazati dizajn baze podataka koji je korišten u izradu istog. Peto poglavlje obrađuje planiranje dizajna aplikacije. Šesto poglavlje objašnjava klijentsku stranu aplikacije (engl. Frontend), kako je kreiran i detaljno ga opisati, dok zadnje poglavlje istražuje istu stvar, ali vezano uz pozadinsku stranu aplikacije (engl. Backend).

2. TEHNOLOGIJE

Jedan od izazova u razvoju je bila odluka koju tehnologiju koristiti. Svaka tehnologija ima svoje prednosti i nedostatke. Na temelju istraživanja različitih tehnologija, odabrana je kombinacija Reacta, TypeScripta i CSS-a za razvoj *frontenda*.

Za *backend* je odlučeno da će se koristiti NestJS s Prismom, a za pohranu podataka korištena je PostgreSQL baza podataka.

Za funkcionalnost i pokretanje ovog rješenja korišten je Docker.

2.1 React

React je trenutno predvodnik među programerima i dosta poznat JavaScript programski okvir (eng Framework). Stvoren je od strane FaceBook-a 2011. godine. React omogućuje laki i brzi prijelaz između stranica bez potrebnog ponovnog učitavanja, zbog toga dolazi naziv „React“. Također, React pruža izrazito dobre performanse te s time omogućuje programerima stvarati kvalitetne i brze web aplikacije. U nastavku je prikazan logo tehnologije.

Slika 2.1 React logo

2.2 CSS

CSS je široko prepoznat među onima koji se bave razvojem web aplikacija. Kratica CSS se primarno koristi za stilizaciju web stranica. Većina današnjih web stranica koristi CSS za svoj vizualni izgled. Slika 2.2 prikazuje logo tehnologije.

Slika 2.2 CSS logo

2.3 TypeScript

TypeScript je objektno orijentirani programski jezik. Napisani kod ne izvršava se izravno u svom izvornom obliku, već se prevođenjem pretvara u JavaScript koji se zatim prikazuje na webu. Prednost korištenja TypeScripta je u tome što odmah nakon pisanja koda otkriva potencijalne greške, za razliku od JavaScripta gdje to nije izravno moguće. Sljedeća slika prikazuje logo TypeScripta.

Slika 2.3 TypeScript logo

2.4 NestJS

NestJS je programski jezik za razvoj aplikacija na serverskoj strani, baziran je na Node.js platformi. Također, koristi se u kombinaciji s TypeScriptom kao i s Reactom. Prikladan je izbor za izradu serverske strane aplikacije, dokumentacija je opširna, a zajednica programera koji koriste je mnogobrojna. Slika 2.4 prikazuje logo tehnologije.

Slika 2.4 NestJS logo

2.5 Prisma

Prisma je ORM alat za razvoj aplikacija. Omogućuje interakciju s bazom podataka i lakšu manipulaciju s podatcima i upravljanje bazom. Podržava relacijski model te pruža podršku za različite baze.

2.6 Docker

Docker predstavlja kontejnersku platformu koja olakšava razvoj aplikacija, pružajući funkcionalnosti za upravljanje aplikacijom i njezinu implementaciju u produkciji. Aplikacije se pokreću unutar virtualnih okruženja poznatih kao kontejneri

2.7 PostgreSql

PostgreSql je objektno-relacijski sustav koji služi za upravljanje bazom podataka. Podržava relacijsku bazu podataka. PostgreSql je jedan od najpoznatijih sustava za upravljanje bazom podataka s velikom zajednicom, što pruža programerima pristup različitim resursima. Slika 2.5 prikazuje logo tehnologije.

Slika 2.5 PostgreSQL logo

3. FUNKCIONALNOSTI APLIKACIJE

Aplikacija sadrži niz različitih funkcionalnosti. Korisnik i administrator dobivaju različite funkcionalnosti kako bi mogli upravljati s rezervacijama, terenima i ostalo. Sljedeća slika prikazuje početnu stranicu aplikacije.

Može se reći da je aplikacija sigurna za upotrebu zbog svog sustava za autentifikaciju i autorizaciju.

Slika 3.1 Početna stranica aplikacije

3.1 Autorizacija i autentifikacija

U svakoj web aplikaciji, autorizacija i autentifikacija su jedne od najbitnijih stavki. Može se reći da aplikacije ne bi bile upotrebljive bez toga, no to sve ovisi o tome koliko je aplikacija složena. Na *backendu* je korišten JWT. JWT je web token koji se koristi za zaštitu podataka, odnosno ako korisnik nema dodijeljen JWT token ne može pristupiti aplikaciji. Također, nakon što se korisnik prijavi, njegova lozinka se kriptira radi dodatne zaštite, tako da lozinku zna samo osoba koja ju je osmislila.

Programski kod 3.1 Funkcija za prijavu

```
async signin(dtoo: AuthDtoLogin, req: Request, res: Response) {
    const { email, lozinka } = dtoo;

    const foundUser = await this.prisma.korisnik.findUnique({
      where: {
        email,
      },
    });

    if (!foundUser) {
      throw new BadRequestException('Korisnik već postoji');
    }

    const isMatch = await this.comparePassword({
      password: lozinka,
      hash: foundUser.lozinka,
    });

    if (!isMatch) {
      throw new BadRequestException('Pogrešna lozinka');
    }

    const rola = foundUser.IDInstitucija === null ? 0 : 1;

    const token = await this.singToken({
      id: foundUser.ID as any,
      email: foundUser.email,
    });

    if (!token) {
      throw new ForbiddenException();
    }

    res.cookie('token', token);
```

```
    return res.send({
      message: 'Korisnik uspješno prijavljen',
      rola,
      id: foundUser.ID,
    });
  }
```

Ovaj programski kod prikazuje funkciju koja služi za prijavu korisnika. Kao što se vidi, korisniku se dodjeljuje token i sprema se u kolačić. Odabrano je spremanje JWT tokena u kolačić kako bi bilo manje posla na samom *frontendu*. Ovo rješenje omogućuje jednostavniju razmjenu podataka između klijenta i servera jer se JWT token automatski šalje u svakom zahtjevu.

Programski kod 3.2 Funkcije za autentifikaciju

```
async hashPassword(password: string) {
  const saltOrRounds = 10;
  const hash = await bcrypt.hash(password, saltOrRounds);
  return hash;
}

async comparePassword(args: { password: string; hash: string }) {
  return await bcrypt.compare(args.password, args.hash);
}

async singToken(args: { id: string; email: string }) {
  const payload = args;

  return this.jwt.signAsync(payload, { secret: jwtSecret });
}
```

Ove tri funkcije također se koriste za autentifikaciju. Prva funkcija je za kriptiranje lozinke radi zaštite korisnika i privatnih informacija. Sljedeća funkcija uspoređuje lozinke kako bi se korisnik prijavio, a zadnja funkcija dodjeljuje JWT token korisniku.

Nakon što je autentifikacija odrađena na *backendu*, slijedi autorizacija korisnika na *frontendu*. U sustavu postoje dvije uloge, administrator i korisnik. Administrator pripada pod institucije koje iznajmljuju svoje terene, a korisnici ih rezerviraju. *Frontend* dobiva od

backenda odgovor koje je uloge korisnik. U sljedećem programskom kodu može se vidjeti kako je spomenuti postupak izveden u programskom jeziku React.

Programski kod 3.3 Autentifikacija i autorizacija Frontend

```
const App = () => {
  const { isLogged, rola } = useContext(AuthContext);
  return (
    <div>
      <BrowserRouter>
        <Header />
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/pregledTermina" element={<PrikazRezervacija />} />

          {isLogged ? (
            <>
              {rola === 1 ? (
                <>
                  <Route path="/admin/panel" element={<AdminPanel />} />
                  <Route
                    path="/admin/preplate"
                    element={<PregledRezervacija />}
                  />
                  <Route path="/admin/unosTerena" element={<UnosTerena />}
/>
                  <Route path="/admin/tereni" element={<PregledTerena />}
/>
                  <Route path="/admin/tereni/edit" element={<EditTerena
/>} />
                  <Route path="/admin/termini/unos" element={<UnosTermina
/>} />
                  <Route
                    path="/admin/odobravanje"
                    element={<OdobravanjeTermina />}
                  />
                  <Route
                    path="/admin/preplate/zauzete"
                    element={<PregledTerminaa />}
                  />
                </>
              ) : (
                <>
                  <Route path="/rezervacija" element={<PrikazRezervacija
/>} />
                  <Route path="/rezervacija/unos" element={<Rezervacija
/>} />
```

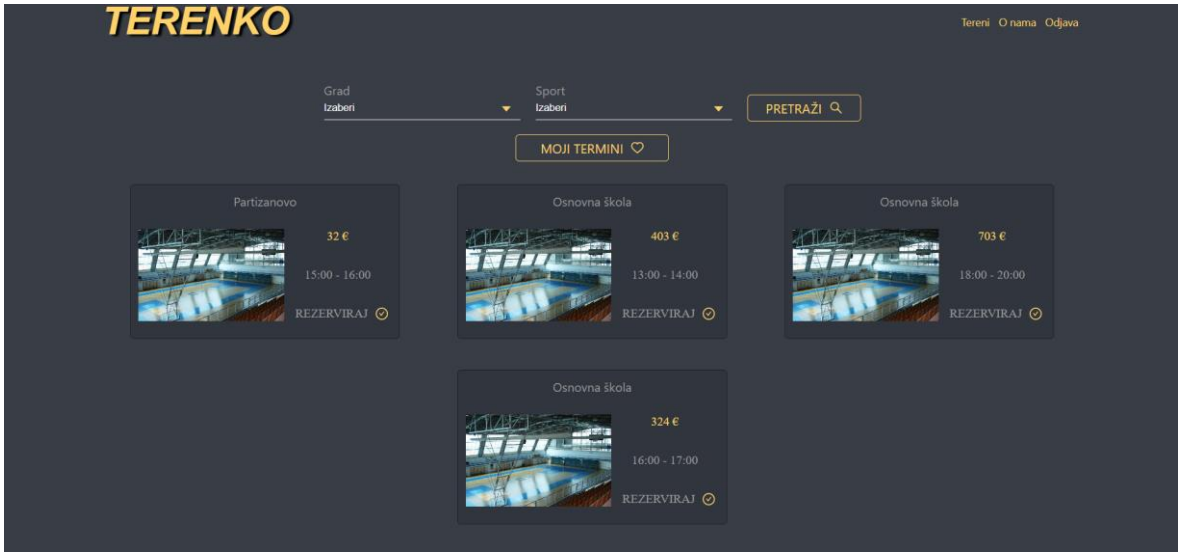
```
                    <Route path="/rezervacija/placanje" element={<Placanje
/>} />
                    <Route path="/mojiTermini" element={<UserTermini />} />
                  </>
                )}
              </>
            ) : (
              <>
                <Route path="*" element={<h1>404</h1>} />
              </>
            )}
          </Routes>
        </BrowserRouter>
      </div>
    );
};

export default App;
```

Programski kod provjerava prvo da li je korisnik prijavljen, a zatim provjerava na koje putanje odnosno stranice korisnik ili administrator ima prava posjetiti.

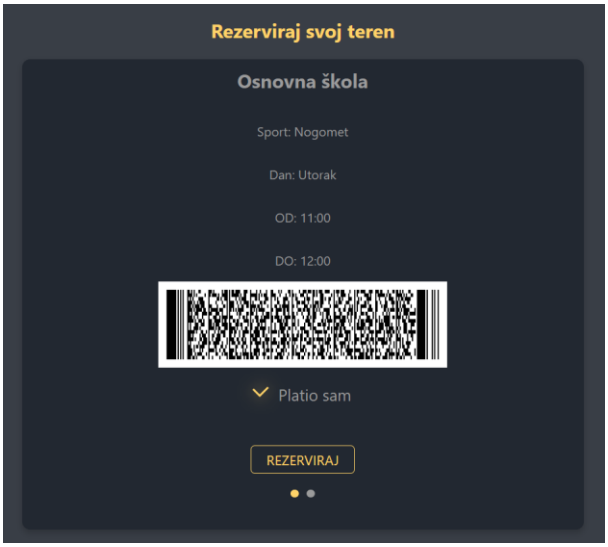
3.2 Rezervacija termina

Glavna funkcionalnost i svrha ove aplikacije je rezervacija sportskih termina. Prilikom ulaska u aplikaciju korisnik dobije uvid u sve termine koji su slobodni za rezervaciju, što se može vidjeti na sljedećoj slici.



Slika 3.2 Prikaz rezervacija

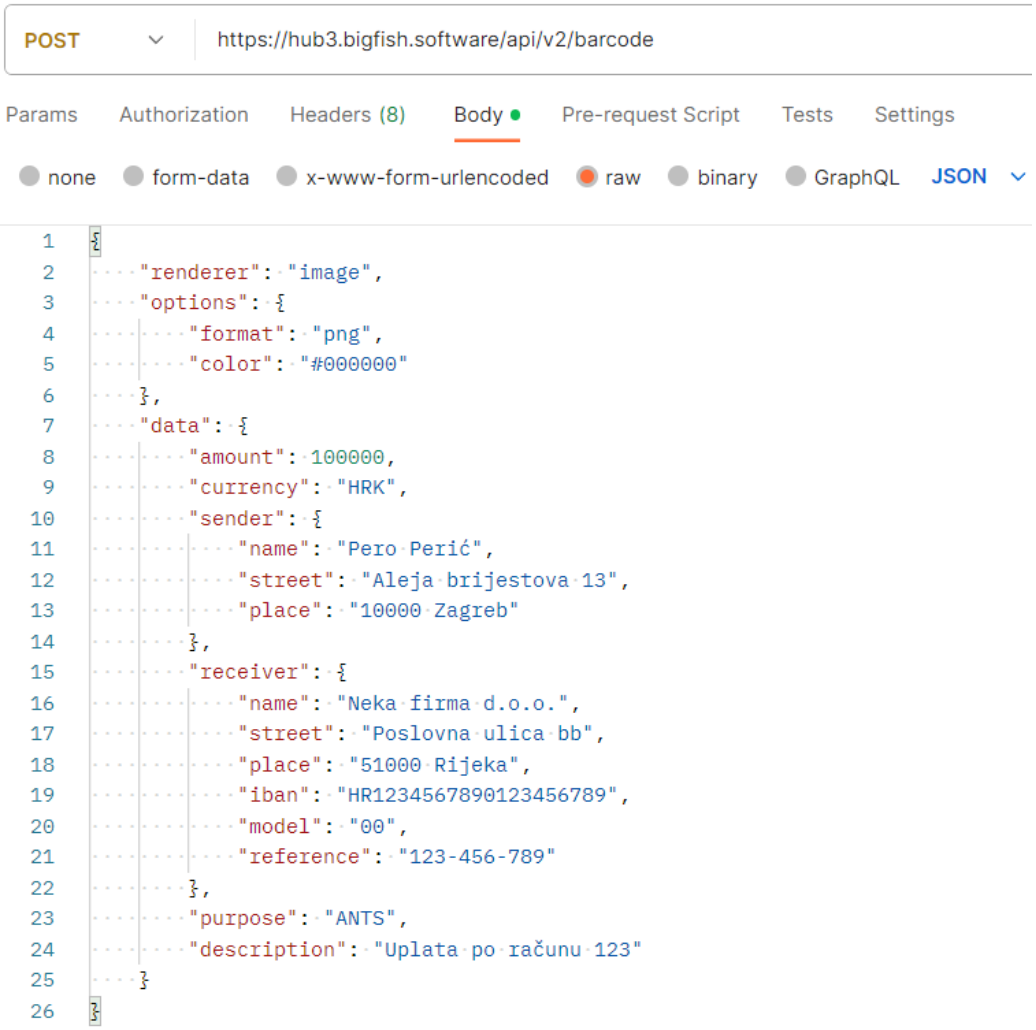
Nakon što korisnik odabere željeni termin dolazi na ekran gdje mu se prikazuju detalji samog termina i barkod koji je potrebno uslikati sa svojom aplikacijom za internet bankarstvo, što prikazuje slika 3.3. Ako je korisnik platio svoj termin stavlja se na listu čekanja sve dok ga administrator ne odobri.



Slika 3.3 Rezervacija termina

3.2.1 *Hub3 API*

Kako bi se izvršilo plaćanje termina, korišten je HUB3 API koji je stvorila Big Fish Software firma. API zahtjeva JSON objekt koji se može vidjeti na slici.



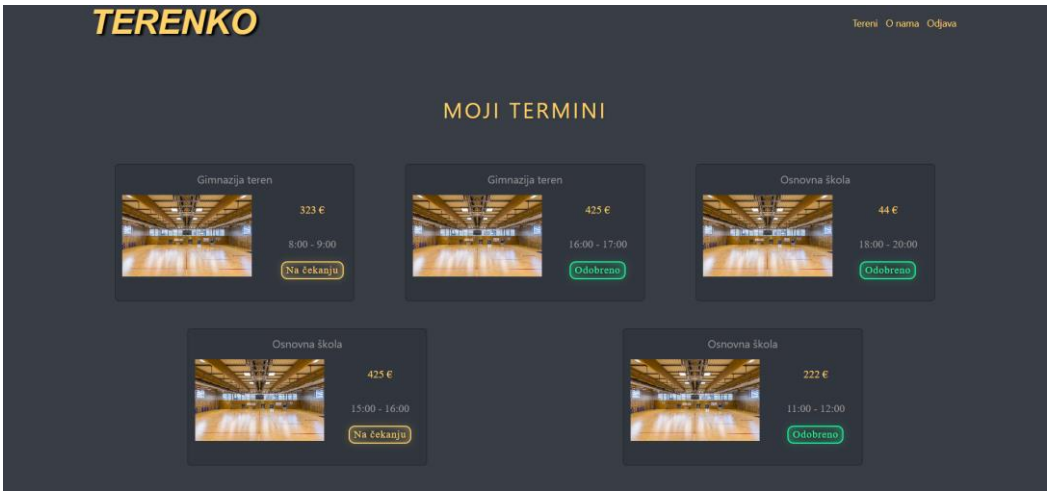
Slika 3.4 Poziv za Hub3

Nakon što API zaprimi potrebne i odgovarajuće podatke, generira se barkod koji korisnik skenira i tako plati svoj termin.

Na slici 3.3 prikazano je kako izgleda barkod koji API generira. API pruža razne mogućnosti, generiranje barkoda u drugim formatima, mijenjanje boje barkoda i tako dalje.

3.3 Moji termini

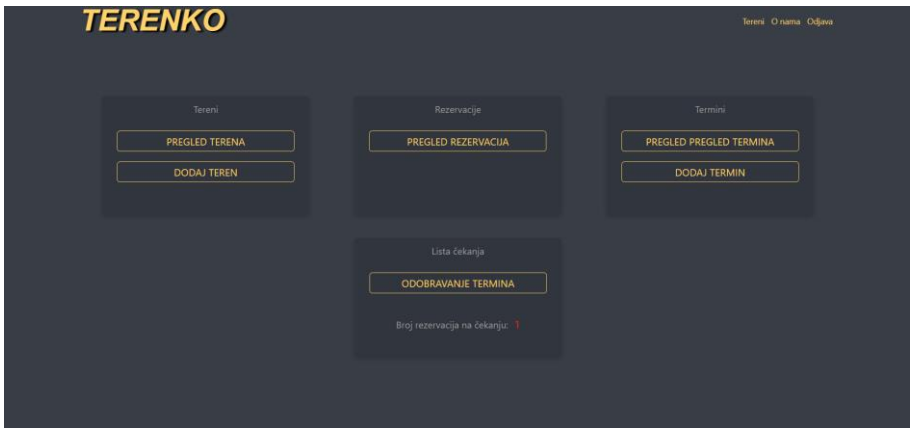
Nakon što korisnik rezervira termin, ima mogućnost pregleda svojih termina te također može vidjeti koji je status rezerviranog termina. Postoje statusi „odobreno“ i „na čekanju“. Sljedeća slika prikazuje ekran za prikaz korisnikovih termina.



Slika 3.5 Ekran moji termini

3.4 Administratorski panel

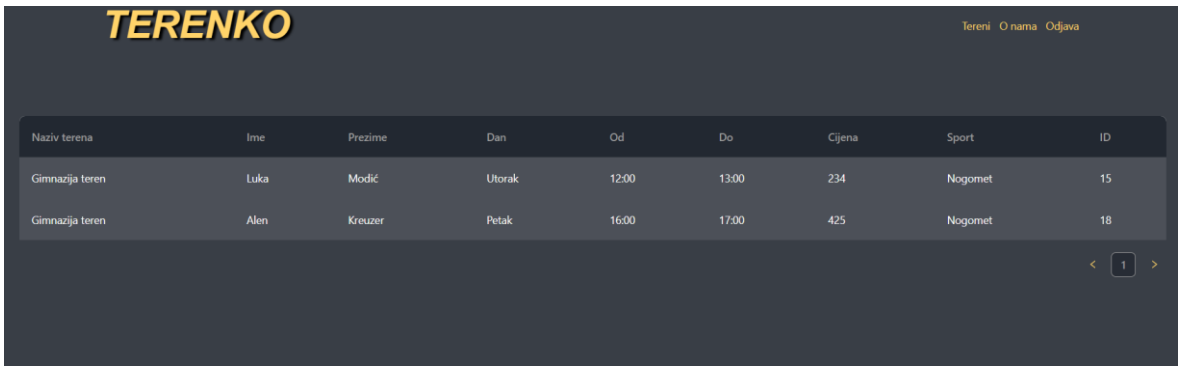
Kako bi administratoru bilo olakšano da ima uvid u sve funkcije aplikacije napravljen je administratorski panel koji je jednostavan za korištenje. Sadrži funkcije kao što su pregled rezervacija, terena i termina. Također, tamo se nalaze paneli za unos terena i termina. Administrator mora imati mogućnost odobravanja rezervacije te je dodana još jedna kartica pod nazivom Odobravanje rezervacija“. Administrator odnosno institucija može izmjenjivati podatke vezane za termine i terene, a isto tako ih može i brisati. Na slici koja slijedi može se vidjeti kako izgleda panel za administratore.



Slika 3.6 Administratorki panel

3.4.1 *Pregled rezervacija*

Na ovom ekranu koji prikazuje slika 3.7, prijavljena institucija može vidjeti sve termine koji su rezervirani, ima uvid u termine i korisnike koji su rezervirali termin.



Slika 3.7 Prikaz rezervacija

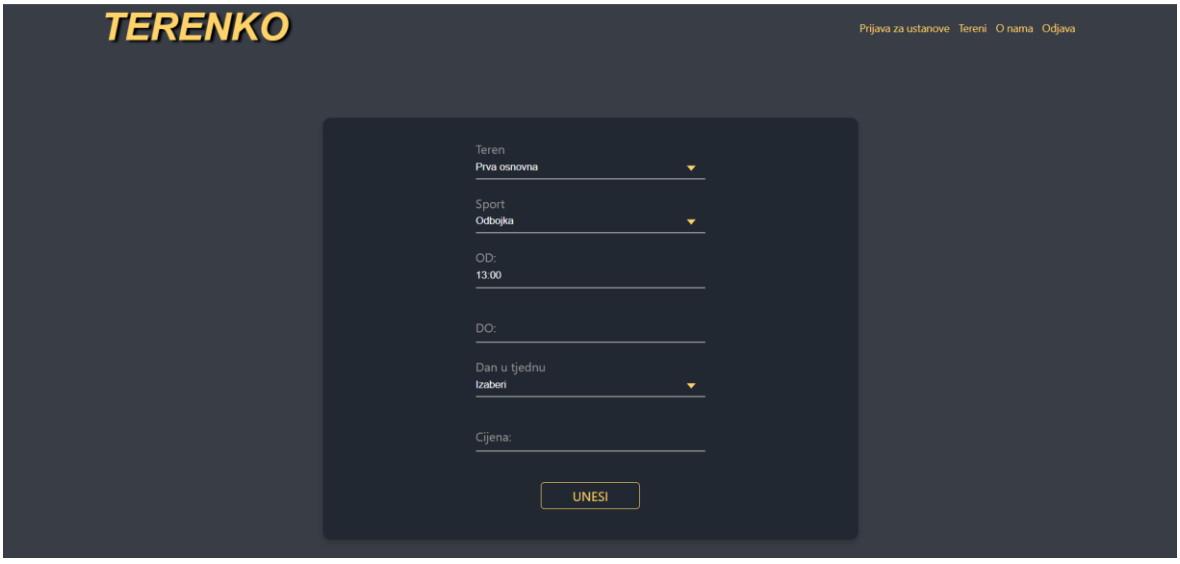
3.4.2 *Odobravanje rezervacije*

Nakon što je korisnik obavio rezervaciju, institucija tog termina dobiva obavijest na svom administratorskom panelu da je netko rezervirao termin. Zatim institucija treba provjeriti je li je uplata prošla i ako je sve u redu, institucija odobrava rezervaciju. Ovaj korak je uveden radi dodatne provjere te kako bi institucija i korisnik bili sigurni da je sve u redu s rezervacijom te kako ne bi došlo do nepotrebnih komplikacija s rezervacijom termina.

3.4.3 *Unos, brisanje i ažuriranje terena i termina*

Institucija mora imati mogućnosti takozvanih CRUD operacija nad terenima i terminima kako bi mogla dodavati iste i izmijeniti podatke ako nešto nije uredu. Na sljedeće dvije slike prikazane su CRUD operacije.

Slika 3.8 CRUD operacije



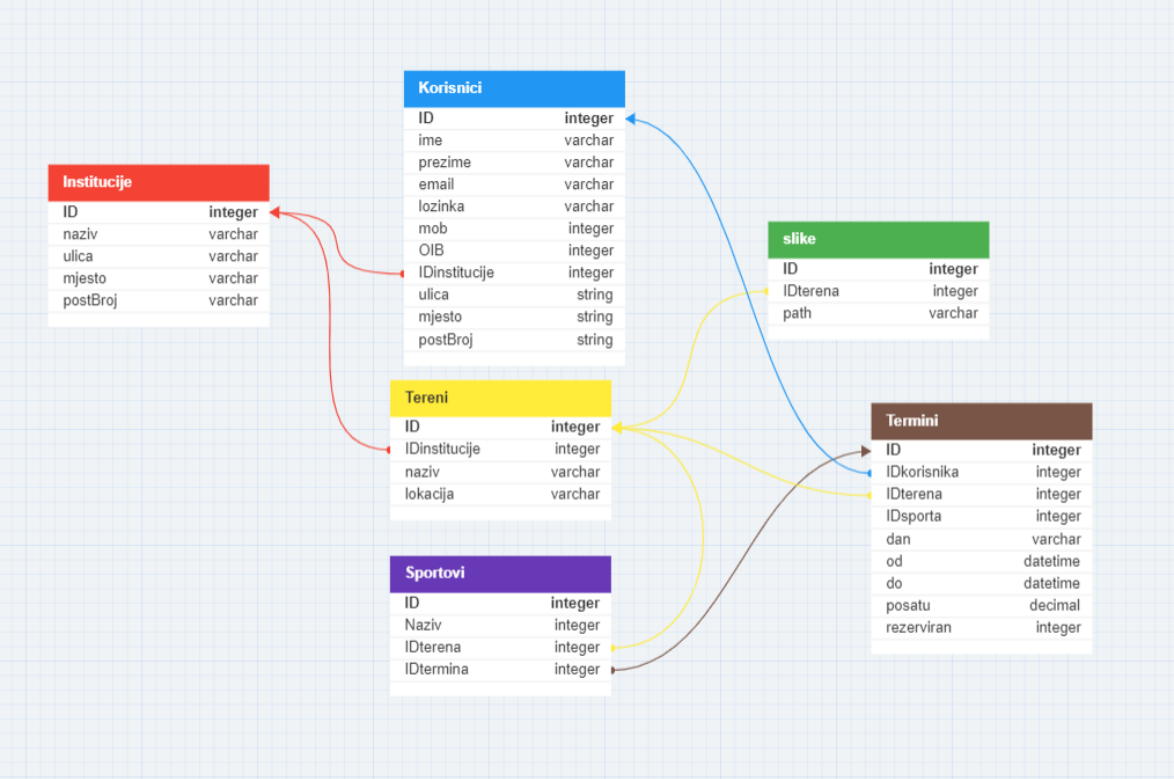
Slika 3.9 Unos termina

4. DIZAJN BAZE

Može se reći da je dizajn baze ključna stavka u izradi web aplikacije. Ako je model baze krivo napravljen cijela aplikacija je neupotrebljiva. Dosta pažnje posvećeno je izradi modela kako bi cijela aplikacija radila na ispravan način.

4.1 DB designer

Za izradu modela baze podataka korišten je DB designer. Alat je djelomično besplatan i jednostavan za korištenje. Na sljedećoj slici može se vidjeti model baze podataka za aplikaciju „Terenko“.



Slika 4.1 Model baze podataka

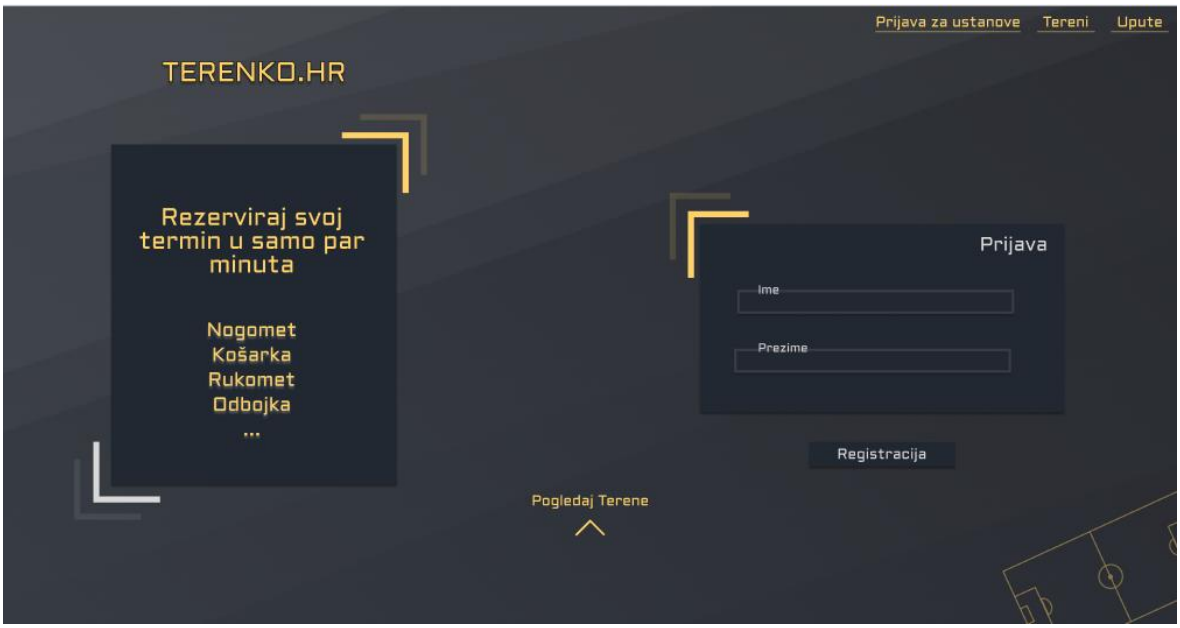
5. PLANIRANJE DIZAJNA

Ono što ima važnost u stvaranju sustava je izrada dizajna aplikacije. Svakom *frontend* programeru je ova stavka bitna kako bi bio efikasniji u kreiranju sučelja web aplikacije. Izrada dizajna aplikacije ima ključnu ulogu u stvaranju sustava kako bi došlo do manje grešaka prilikom izrade *frontenda*. Planiranjem i izradom dizajna dobije se bolje korisničko iskustvo i korisničko sučelje.

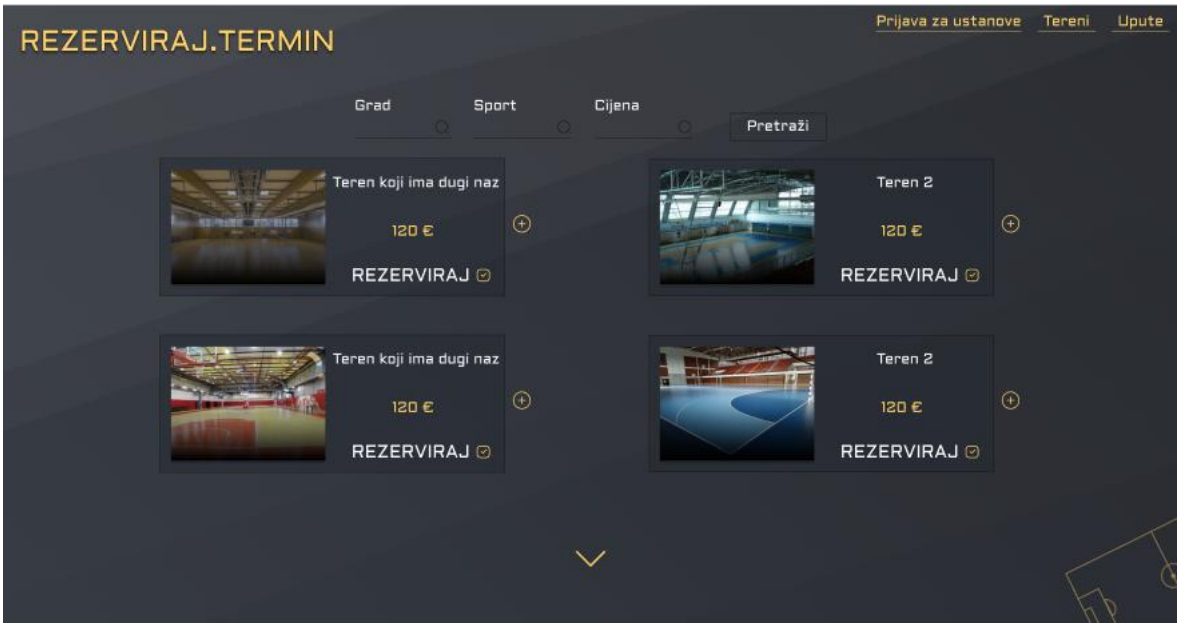
5.1 Figma

Za izradu dizajna aplikacije korišten je besplatan alat Figma koji nudi niz funkcionalnosti za kreiranje prototipa aplikacije. U sljedećim slikama možemo vidjeti kako izgleda izrada dizajna za ovu aplikaciju.

Dizajn iz Figme razlikuje se od konačnog proizvoda, ali bio je značajan prilikom snalaženja gdje će se koja komponenta nalaziti na ekranu.



Slika 5.1 Prototip dizajna početne stranice



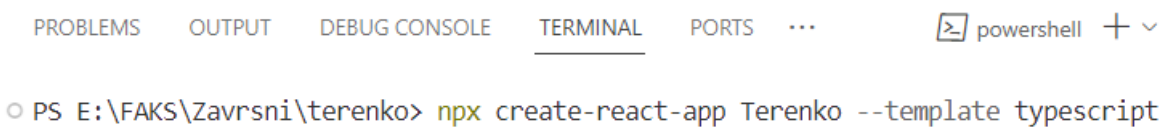
Slika 5.2 Prototip dizajna za prikaz termina

6. FRONTEND

Kao što je navedeno, korišten je programski okvir React. Prvo je bilo potrebno postaviti okruženje i instalirati sve potrebne pakete. Korištene su različite biblioteke, animacije te stilovi za dizajn kako bi aplikacija izgledala što privlačnije i bila zanimljivija korisnicima.

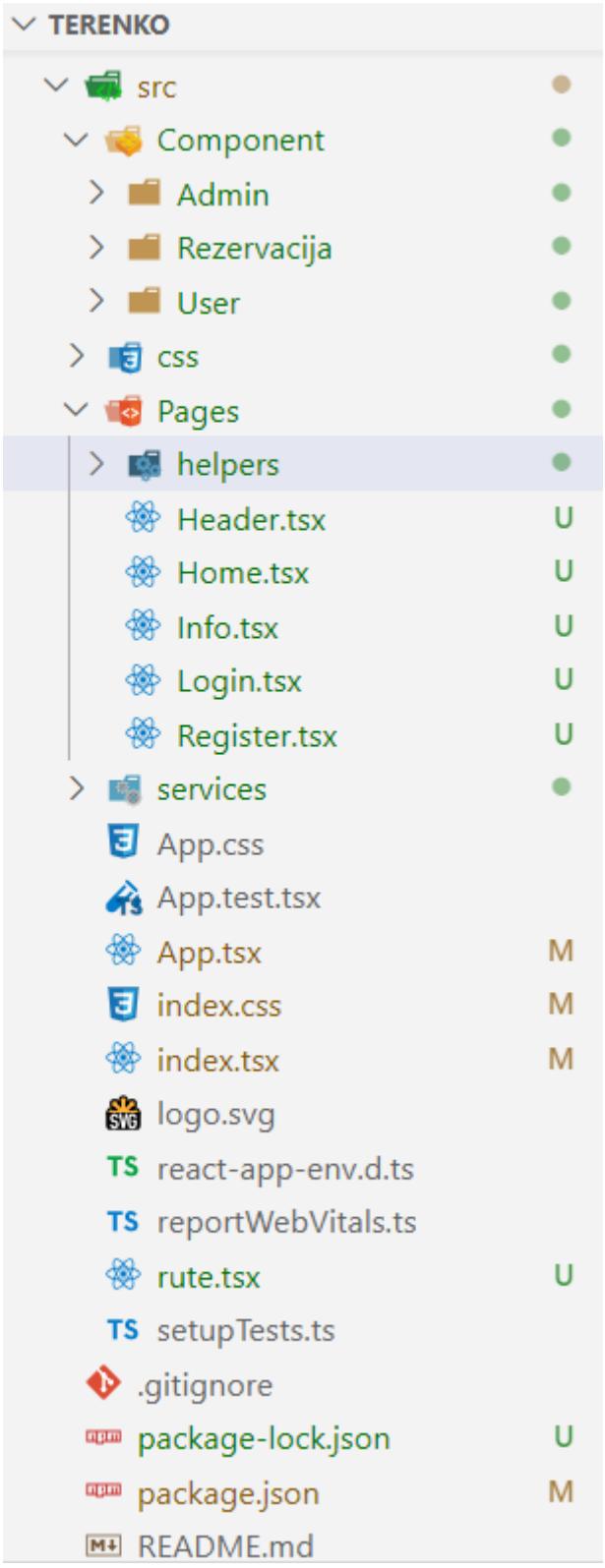
6.1 Postavljanje okruženja

Postavljanje okruženja za React je vrlo jednostavno. Praćena je dokumentacija na web stranici koju nudi React. [1]



Slika 6.1 Postavljanje frontend okruženja

Na ovoj slici možemo vidjeti kako izgleda kreiranje React projekta s TypeScriptom. Nakon što se izvrši naredba, dobije se projekt i njegova struktura. Na sljedećoj slici je prikazana struktura već gotove aplikacije Terenko.



Slika 6.2 Struktura projekta

6.2 Korištene biblioteke

U projektu su korištene biblioteke kao što je Ant Design. Ant Design sadrži komponente koje olakšavaju stvari programerima kako ne bi morali raditi dodatne prilagođene komponente. Za primjer, korištena je komponenta koja šalje obavijesti.

Slika 6.3 Ant Design komponenta

6.3 Dizajn

Na izradu dizajna potrošeno je od prilike 10 sati. Glavni cilj bio je postići da aplikacija izgleda moderno te da je dinamična. Korišten je CSS i *framer-motion* za animacije. Cijela aplikacija je puna originalnog CSS koda, te je animirana kako bi bila ugodna oku.

6.4 Pozivi i ostale tehnike

Prilikom izrade aplikacije korišteno je veliki broj poziva i tehnika koje React kao programski jezik pruža.

Za dohvat HTTP zahtjeva bila je potrebna biblioteka Axios. Axios se koristi za komunikaciju sa serverom, u ovom slučaju NestJS-om. Programski kod ispod prikazuje Axios poziv za rezervaciju termina.

Programski kod 6.1 Axios poziv

```
const rezervirajTermin = async () => {
    try {
      const response = await axios.post(rezService.rezerviraj(), rez, {
        headers: {
          "Content-Type": "application/json",
        },
      });

      if (response.status === 201) {
        navigate("/rezervacija/placanje");
        message.success("Uspješno ste rezervirali termin");
      }
    } catch (err: any) {
      console.log(err);
```

```
    }
  };
```

Ostale tehnike koje su korištene u izradi aplikacije su takozvane „*custom-hook*“ koje služe za odvajanje poslovne (engl. *buisness*) logike. Korisna je tehnika koja omogućava programerima da izdvoje i organiziraju logiku programskog koda. Ovime se olakšava održavanje, testiranje i ponovna uporaba koda.

Također, bitna tehnika u Reactu je kontekst (engl. Context). *Context* je mehanizam za globalnu razmjenu podataka između komponenata, bez obzira na njihovu dubinu u komponentnoj hijerarhiji. U nastavku je prikazan primjer korištenja, odnosno postavljanje *contexta*.

Programski kod 6.2 Context

```
interface AuthContextValue {
  isLogged: boolean;
  setIsLogged: React.Dispatch<React.SetStateAction<boolean>>;

  rola: number | undefined;
  setRola: React.Dispatch<React.SetStateAction<number | undefined>>;

  korisnikID: string;
  setKorisnikID: React.Dispatch<React.SetStateAction<string>>;
}

export const AuthContext = createContext<AuthContextValue>(
  {} as AuthContextValue
);

interface AuthProviderProps {
  children: ReactNode;
}

export const AuthProvider = ({ children }: AuthProviderProps) => {
  const [isLogged, setIsLogged] = useState<boolean>(() => {
    const storedIsLogged = localStorage.getItem("isLogged");
    return storedIsLogged ? JSON.parse(storedIsLogged) : false;
```

```
  });

  const [rola, setRola] = useState<number>();
  const [korisnikID, setKorisnikID] = useState<string>("");
  const contextValue: AuthContextValue = {
    isLogged,
    setIsLogged,
    rola,
    setRola,
    korisnikID,
    setKorisnikID,
  };

  return (
    <AuthContext.Provider
value={contextValue}>{children}</AuthContext.Provider>
  );
};
```

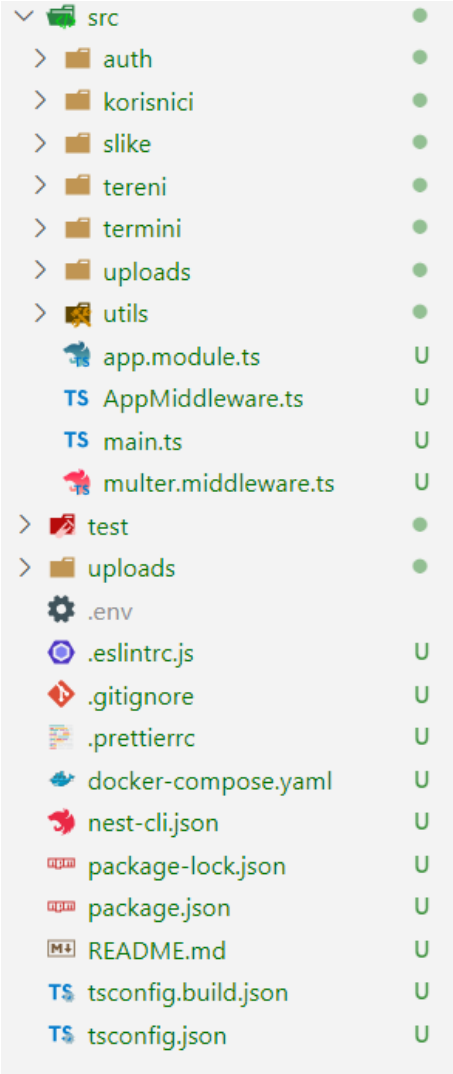
7. BACKEND

Na *backendu*, aplikacija koristi NestJS i Prismu. Da bi NestJS i Prisma mogli zajedno raditi potrebno je postaviti okruženje za te tehnologije.

7.1 Postavljanje okruženja

7.1.1 *NestJS*

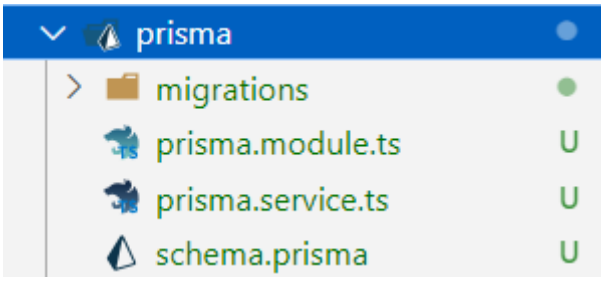
Prije nego što se započne s instalacijom NestJS-a bilo je potrebno instalirati NodeJS. NodeJS besplatan je serverski okvir koji izvršava JavaScript kod na poslužiteljskoj strani. Nakon što je NodeJS instaliran, slijedi instalacija NestJS. Na slici ispod prikazana je struktura projekta kada se postavi okruženje za NestJS. Za detaljnije postupke prilikom instalacije dostupna je dokumentacija na službenoj web stranici. [2]



Slika 7.1 Struktura backenda

7.1.2 *Prisma*

Prisma je korištena kao moderni ORM koji služi za lakše upravljanje s bazom putem programskog koda. Nakon instalacije NestJS-a slijedi instalacija Prisme koja se također može provjeriti detaljno na njihovoj službenoj stranici. Na sljedećoj slici može se vidjeti kako izgleda struktura Prisme unutar NestJS aplikacije. [3]



Slika 7.2 Struktura Prisme

Kao što je prikazano na slici, Prisma sadrži migracije, modul, servise i shemu baze podataka. Migracije se provode kada dođe do izmjena u shemi baze podataka.

7.1.3 *Docker*

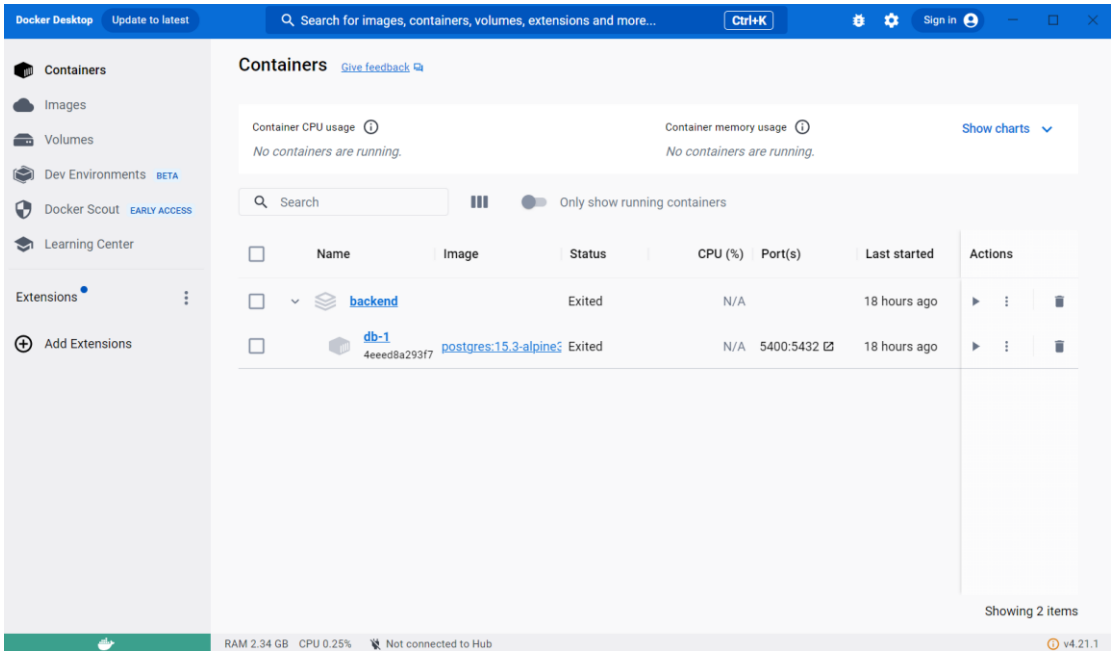
Docker pokreće cijelu aplikaciju. Jednostavan je za postavljanje. Prvo što je bilo potrebno napraviti za Docker je „docker.compose“ (programski kod 7.1) datoteka koja u sebi sadrži detalje o bazi podataka koju Docker pokreće. Za detaljnije postavljanje Dockera dostupna je dokumentacija. [4]

Programski kod 7.1 Docker.compose datoteka

```
version: '3'

services:
  db:
    image: postgres:15.3-alpine3.18
    restart: always
    environment:
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
    ports:
      - '5400:5432'
```

Za upravljanje kontejnerom koji pokreće bazu podataka korištena je besplatna aplikacija Docker Desktop. Slika 7.3 prikazuje Docker Desktop.



Slika 7.3 Docker Desktop

7.2 Kreiranje poziva

Frontend zahtjeva podatke koje može prikazivati na svom UI. NestJS je zadužen za kreiranje poziva, odnosno generiranje API-a. Na sljedećim primjerima koda prikazan je jednostavan kod kontrolera i servisa. Kod prikazuje dohvat termina koje je korisnik rezervirao.

Programski kod 7.2 Prikaz kontrolera

```
@Post('/terminiByUser')
  async terminiByUser(@Body() body: { korisnikId: number }):
Promise<Termin[]> {
    const { korisnikId } = body;
    return
this.terminiService.getAllReservedTerminiByUserId(korisnikId);
  }
```

Programski kod 7.3 Prikaz servisa

```
async getAllReservedTerminiByUserId(userId: number):
Promise<any[]> {
    const termini = await this.prisma.termin.findMany({
      where: {
        IDkorisnika: {
          equals: userId,
        },
      },
      select: {
        IDterena: true,
        ID: true,
        dan: true,
        od: true,
        do: true,
        rezeraviran: true,
        posatu: true,
        Tereni: {
          select: {
            naziv: true,
            lokacija: true,
            Sport: {
              select: {
                Naziv: true,
              },
            },
          },
        },
      },
    });

    const filteredTermini = termini
      .map((termin) => ({
        IDterena: termin.IDterena,
        IDtermina: termin.ID,
        dan: termin.dan,
        od: termin.od,
        do: termin.do,
        posatu: termin.posatu,
        imeTerena: termin.Tereni?.naziv,
        sport: termin.Tereni.Sport[0]?.Naziv,
        sport2: termin.Tereni.Sport[1]?.Naziv || null,
        lokacija: termin.Tereni?.lokacija,
        status: termin.rezeraviran,
      }))
      .filter((termin) =>
        Object.values(termin).some((value) => value !==
null),
```

```
      );

    return filteredTermini;
  }
```

Pomoću Prisme, NestJS dohvaća sve potrebne podatke kako bi prikazao rezervirane termine korisnika. Nakon što dohvati podatke oni se filtriraju kako bi se dobio uređen JSON koji se prikazuje na *frontendu*.

8. ZAKLJUČAK

Sport je važan dio našeg života, kako bi potakli ljude na sportski način života, aplikacija za rezervaciju termina im omogućuje da odaberu teren na kojem žele rezervirati svoj termin, a to mogu napraviti iz udobnosti svoga doma. Aplikacija omogućuje pregled termina, rezervaciju i plaćanje istog.

Osim praktičnih koristi za korisnike koji rezerviraju svoj termin prednost imaju i sportski klubovi ili institucije koje pružaju uslugu rezervacije termina. Aplikacija omogućuje bolje upravljanje resursima, praćenje korisnika, te donosi određeni oblik reklame, a s time dolazi veći profit.

9. LITERATURA

[1] React. [Online]. 2023. Dostupno na: https://react.dev. (1.10.2023.)

[2] NestJS Documentation. 2023. [Online]. Dostupno na: https://docs.nestjs.com (1.10.2023.)

[3] Prisma. [Online]. 2023. Dostupno na: https://www.prisma.io (1.10.2023.)

[4] Docker Documentation. [Online]. 2023. Dostupno na: https://docs.docker.com (1.10.2023.

[5] BigFish Software. 2023. [Online]. Dostupno na: https://hub3.bigfish.software/api/v2 (1.10.2023.)

10. OZNAKE I KRATICE

API – Application Programming Interface (Aplikacijsko-programsko sučelje)

CSS – Cascading Style Sheets (kaskadni listovi stilova)

HUB – Hrvatska udruga banaka

HTTP – Hyper text transfer protocol (Hipermedijski protokol za prijenos)

JSON – JavaScript Oriented Notation (JavaScript objektna notacija)

JWT – JSON WEB token

ORM - Oject-relation mapping (Objektno-relacijsko preslikavanje)

UI – User Interface (Korisničko sučelje)

11. SAŽETAK

Naslov:

Cilj ovog završnog rada je bio predstaviti WEB aplikaciju za rezervaciju sportskih termina. Većina ljudi ne želi imati posla s mnoštvom papira kada se radi o rezervaciji termina, zbog toga je osmišljena ova aplikacija koja omogućuje korisnicima rezervaciju termina, a institucijama da imaju uvid u svoje korisnike. Za izradu ove aplikacije korišten je React, TypeScript, CSS, NestJS, Prisma, Docker i PostgreSql. U radu su objašnjeni koraci kako je napravljena aplikacija. Također, rad opisuje postavljanje okruženja za rad na *frontendu* i *bakcendu*, planiranje dizajna aplikacije i dizajna baze te način na koji je napravljena autorizacija i autentifikacija.

Ključne riječi: WEB aplikacija, sportski termini, React, NestJS, Prisma, PostgreSQL

12. ABSTRACT

Title:

The goal of this paper was to present WEB application for booking sports appointments. Most of the pople don't want to deal with paperwork when it comes to booking appointments, that's why this application was designed to allow users to book appointments and institutions to have insight into their users. React, TypeScript, CSS, NestJS, Prisma, Docker and PostgreSQL were used to create this application. The paper explains the steps of how the application was made. Also, the paper describes setting up the environment for working on the frontend and backend, planning the design of the application and the design of the database, and the way in which authorization and authentication were made.

Keywords: WEB application, sports appointments, React, NestJS, Prisma, PostgreSQL

IZJAVA O AUTORSTVU ZAVRŠNOG RADA

Pod punom odgovornošću izjavljujem da sam ovaj rad izradio/la samostalno, poštujući načela akademske čestitosti, pravila struke te pravila i norme standardnog hrvatskog jezika. Rad je moje autorsko djelo i svi su preuzeti citati i parafraze u njemu primjereno označeni.

Mjesto i datum	Ime i prezime studenta/ice	Potpis studenta/ice
U Bjelovaru, 19. 10. 2023	Alen Kreuzer	Alen Kreuzer

U skladu s čl. 58, st. 5 Zakona o visokom obrazovanju i znanstvenoj djelatnosti, Veleučilište u Bjelovaru dužno je u roku od 30 dana od dana obrane završnog rada objaviti elektroničke inačice završnih radova studenata Veleučilišta u Bjelovaru u nacionalnom repozitoriju.

Suglasnost za pravo pristupa elektroničkoj inačici završnog rada u nacionalnom repozitoriju

Alen Kreuzer
ime i prezime studenta/ice

Dajem suglasnost da tekst mojeg završnog rada u repozitorij Nacionalne i sveučilišne knjižnice u Zagrebu bude pohranjen s pravom pristupa (zaokružiti jedno od ponuđenog):

a) Rad javno dostupan
b) Rad javno dostupan nakon _____ (upisati datum)
c) Rad dostupan svim korisnicima iz sustava znanosti i visokog obrazovanja RH
d) Rad dostupan samo korisnicima matične ustanove (Veleučilište u Bjelovaru)
e) Rad nije dostupan.

Svojim potpisom potvrđujem istovjetnost tiskane i elektroničke inačice završnog rada.

U Bjelovaru, 19. 10. 2023.

Alen Kreuzer
potpis studenta/ice