Izrada web aplikacije koristeći programski okvir Tailwind CSS na studijskom slučaju aplikacije za rezervaciju malonogometnih terena

Knežević, Ante

VELEUČILIŠTE U ŠIBENIKU
ODJEL RAČUNARSTVA
PRIJEDIPLOMSKI STRUČNI STUDIJ POSLOVNE INFORMATIKE

Ante Knežević

IZRADA WEB APLIKACIJE KORISTEĆI PROGRAMSKI OKVIR TAILWIND CSS NA STUDIJSKOM SLUČAJU APLIKACIJE ZA REZERVACIJU MALONOGOMETNIH TERENA

Završni rad

Šibenik, 2024.

VELEUČILIŠTE U ŠIBENIKU
ODJEL RAČUNARSTVA
PRIJEDIPLOMSKI STRUČNI STUDIJ POSLOVNE INFORMATIKE

IZRADA WEB APLIKACIJE KORISTEĆI PROGRAMSKI OKVIR TAILWIND CSS NA STUDIJSKOM SLUČAJU APLIKACIJE ZA REZERVACIJU MALONOGOMETNIH TERENA

Završni rad

Kolegij: Izrada web aplikacija
Mentor: Marko Pavelić
Student: Ante Knežević
Matični broj studenta: 1219064610

Šibenik, rujan 2024.

TEMELJNA DOKUMENTACIJSKA KARTICA

Veleučilište u Šibeniku
Odjel Poslovne informatike
Preddiplomski stručni studij poslovne informatike

Završni rad

IZRADA WEB APLIKACIJE KORISTEĆI PROGRAMSKI OKVIR TAILWIND CSS NA STUDIJSKOM SLUČAJU APLIKACIJE ZA REZERVACIJU MALONOGOMETNIH TERENA

ANTE KNEŽEVIĆ
Boraja 12, ante.knezevic23@gmail.com

Ovaj završni rad opisuje razvoj web aplikacije za rezervaciju malonogometnih terena. Aplikacija omogućuje korisnicima pregled i rezervaciju dostupnih terena putem jednostavnog i intuitivnog sučelja. Razvijena je korištenjem tehnologija React za korisničko sučelje (engl. *Frontend*) i Firebase kao pozadinski poslužitelj (engl. *Backend)*, dok je programski okvir Tailwind CSS korišten za stilizaciju korisničkog sučelja. Korisnici se mogu registrirati i prijaviti kako bi pristupili svojim profilima i upravljali rezervacijama, uključujući mogućnost brisanja rezervacija. Aplikacija nudi pregled informacija o terenu, uključujući termine i cijene, te koristi platformu Firebase za autentifikaciju korisnika i pohranu podataka o rezervacijama. Fokus je stavljen na korisničko iskustvo, osiguravajući brz i jednostavan proces rezervacije. Implementirani su responzivni dizajn i funkcionalnosti poput *ScrollToTopButtona* za bolju navigaciju. Aplikacija je dizajnirana tako da omogućuje jednostavno proširenje funkcionalnosti u budućnosti.

(62 stranice / 28 slika /22 koda/ jezik izvornika: hrvatski)
Rad je pohranjen u digitalnom repozitoriju Knjižnice Veleučilišta u Šibeniku
Ključne riječi: web aplikacija, korisničko sučelje, pozadinski poslužitelj
Mentor: Marko Pavelić mag. ing. inf. et comm. tech., predavač
Rad je prihvaćen za obranu dana:

BASIC DOCUMENTATION CARD

Polytechnic of Šibenik
Department of Business informatics
Professional Undergraduate of business informatics

Batchelor Thesis

WEB APPLICATION DEVELOPMENT USING TAILWIND CSS FRAMEWORK ON CASE STUDY OF RESERVATIONS FOOTBAL COURTS

ANTE KNEŽEVIĆ
Boraja 12, ante.knezevic23@gmail.com

This thesis describes the development of a web application for booking small football fields. The application allows users to browse and reserve available fields through a simple and intuitive interface. It was developed using technologies such as React for the frontend and Firebase for the backend, with Tailwind CSS used for styling the interface. Users can register and log in to access their profiles and manage reservations, including the ability to delete bookings. The application provides an overview of field information, including available time slots and prices, and uses Firebase for user authentication and storage of reservation data. The focus was on user experience, ensuring a fast and straightforward booking process. Features like responsive design and functionalities such as the *ScrollToTopButton* were implemented to enhance navigation. The application is designed to allow easy future expansion of functionalities.

(62 pages / 28 figures /22 codes/ original in Croatian language)
Thesis deposited in Polytechnic of Šibenik Library digital repository
Keywords: web app, frontend, backend
Supervisor: Marko Pavelić mag. ing. inf. et comm. tech., predavač
Paper accepted:

Sadržaj

1. UVOD

U suvremenom svijetu digitalizacije i automatizacije, web aplikacije postale su ključan pomagalo za olakšavanje svakodnevnih aktivnosti, uključujući upravljanje različitim vrstama usluga. Industrija sportskih objekata, posebice najam malonogometnih terena, također osjeća potrebu za modernizacijom kako bi se zadovoljili rastući zahtjevi korisnika. Cilj ovog rada je predstaviti proces razvoja web aplikacije koja omogućuje jednostavno rezerviranje malonogometnih terena putem Interneta, uz optimizirano korisničko iskustvo. Aplikacija pruža korisnicima mogućnost pretrage dostupnih terena, rezervacije termina te upravljanje svojim rezervacijama, sve kroz jedno integrirano sučelje. Dodatno, vlasnicima terena omogućava bolju organizaciju i transparentan sustav upravljanja rezervacijama.

Tradicionalni načini rezervacije sportskih terena, poput telefonskih poziva ili osobnih dolazaka, često su nepraktični i neefikasni, kako za korisnike, tako i za vlasnike terena. Problemi kao što su dvostruke rezervacije, manjak informacija o dostupnosti terena u stvarnom vremenu te nedostatak centraliziranog sustava za praćenje i upravljanje rezervacijama predstavljaju velike izazove. Ovi izazovi ne samo da narušavaju korisničko iskustvo, već stvaraju dodatne operativne troškove i složenost za upravitelje terena. Stoga je potrebno moderno digitalno rješenje koje će omogućiti brži, pouzdaniji i transparentniji proces rezervacija.

Za razvoj web aplikacije korišteno je programsko okruženje (IDE) Visual Studio Code zbog fleksibilnosti, jednostavnosti i podrške za moderne web tehnologije.Programski okvir React je primijenjen za izradu interaktivnog korisničkog sučelja, omogućujući lako upravljanje komponentama i brze promjene. Tailwind CSS odabran je za stilizaciju radi ubrzanja dizajna i osiguravanja responzivnosti putem unaprijed definiranih klasa.

Kao pozadinski poslužitelj, *Google Firebase* je korišten za upravljanje podatcima i autentikaciju. *Firestore* pruža stabilnu NoSQL bazu podataka za pohranu informacija o terenima i rezervacijam, dok *Firebase Authentication* omogućava sigurno prijavljivanje korisnika. *Firebase Cloud Functions* koristi se za implementaciju poslužateljskih funkcionalnosti, poput slanja potvrda o rezervacijama.

2. TEHNOLOŠKA RJEŠENJA ZA RAZVOJ WEB APLIKACIJE

Za razvoj web aplikacije korišten je niz modernih tehnologija koje omogućuju fleksibilan, skalabilan i siguran sustav za rezervaciju malonogometnih terena. Ove tehnologije odabrane su zbog svojih performansi, jednostavnosti korištenja i mogućnosti brze prilagodbe različitim poslovnim potrebama. U nastavku su detaljno opisane glavne tehnologije koje su korištene u razvoju aplikacije, uključujući razvojno okruženje, biblioteke za prikaz, reprezentaciju te izgled aplikacije, kao i poslužiteljske usluge.

2.1.Visual Studio Code

Visual Studio Code (VS Code) je besplatno razvojno okruženje (IDE)[1] otvorenog koda koje je razvila tvrtka Microsoft. Korištenje VS Code-a tijekom razvoja ove web aplikacije omogućilo je efikasan rad zbog niza ključnih značajki:

- **Podrška na više jezika i proširenja**: VS Code nativno podržava mnoge jezike, uključujući JavaScript, HTML, CSS i JSON što je od presudne važnosti za razvoj web aplikacija. Uz to, dostupne su tisuće ekstenzija koje dodatno proširuju funkcionalnost, kao što su ES7+ React/Redux ekstenzija za bolju podršku u radu sa Reactom, ili Tailwind CSS IntelliSense za automatske prijedloge Tailwind klasa.
- **IntelliSense**: Ova značajka omogućava autokompletiranje koda i automatski prijedlog sintakse na temelju postojeće strukture projekta. IntelliSense ubrzava proces razvoja i smanjuje mogućnost pojave sintatičkih pogrešaka.
- **Linter i Formaters**: Integrirani alat za linting[2] i formatiranje koda pomažu u održavanju čistoće i konzistentnosti koda. U ovom projektu korišten je ESLint za osiguranje ispravnog JavaScript koda, te Prettier za automatsko formatiranje koda prema određenim stilskim pravilima.
- **Debugging alat**: VS Code nudi ugrađene alate za otklanjanje pogrešaka koje omogućavaju testiranje aplikacije direktno iz okruženja, što uključuje postavljanje točaka prekida (breakpoints), pregled varijabli i praćenje toka programa u stvarnom vremenu.

[1] IDE je skraćenica za Integrated Development Environment (integrirano razvojno okruženje). To je softverski alat koji programerima omogućuje razvoj softverskih aplikacija na učinkovit način pružajući niz alata na jednom mjestu.
[2] Linting je proces analize izvornog koda kako bi se otkrile potencijalne greške, stilističke nepravilnosti i probleme s kodom.

- **Git integracija**: Ugrađena podrška za Git omogućila je jednostavno praćenje verzija koda, brzu sinkronizaciju sa repozitorijem i rad na različitim granama (branchovima) projekta. Ova funkcionalnost osigurava bolju kontrolu nad promjenama i pojednostavljuje suradnju.

2.2. React

React je JavaScript biblioteka za izradu korisničkih sučelja koja se temelji na komponentama, a razvijena je od strane Facebooka. React je izabran zbog svoje modularnosti, brzine i fleksibilnosti. U ovom projektu korišten je za izradu cijelog korisničkog sučelja aplikacije.

- **Komponentna arhitektura**: React omogućuje podjelu korisničkog sučelja u manje, izolirane komponente koje se mogu ponovno koristiti. Na primjer, komponente za prikaz informacija o terenu, formu za rezervaciju terena i navigaciju su odvojene, čime se omogućava njihova lakša kontrola i ponovno korištenje u različitim dijelovima aplikacije.
- **Virtual DOM**: Virtualni DOM omogućava React-u da efikasno upravlja promjenama u korisničkom sučelju. Umjesto direktnog manipuliranja stvarnim DOM-om (koji može biti spor), React koristi virtualni prikaz DOM-a, uspoređuje ga s prethodnom verzijom i zatim optimizirano primjenjuje samo one promjene koje su potrebne.
- **React Hooks**: Funkcionalni hooks, kao što su *useState* i *useEffect*, korišteni su za upravljanje lokalnim stanjem komponenti i pozivanje API-ja tijekom životnog ciklusa aplikacije. Ovi hooks omogućuju bolje rukovanje asinkronim pozivima, poput onih za dohvaćanje dostupnih termina ili podatke o korisnicima.

2.3. JavaScript

JavaScript je dinamički programski jezik koji se koristi prvenstveno za razvoj interaktivnih elemenata na web stranicama. JavaScript omogućava manipulaciju sadržajem web stranica, rukovanje događajima, animacije, validaciju obrazaca, asinkronu komunikaciju s pozadinskim sučeljem i još mnogo toga.

- **Upotreba na strani klijenta (korisničko sučelje)**: JavaScript se koristi za interakciju s HTML-om i CSS-om, omogućavajući dinamičke promjene u korisničkom sučelju, kao što su dodavanje ili uklanjanje elemenata stranice, validacija obrazaca i rukovanje korisničkim događajima.
- **Asinkroni rad**: JavaScript podržava asinkrone operacije poput učitavanja podataka sa servera pomoću AJAX-a ili Fetch API-ja, što omogućava korisnicima neprekidnu interakciju s aplikacijom bez potrebe za osvježavanjem stranice.

- **Interoperabilnost**: JavaScript se savršeno integrira s drugim web tehnologijama (HTML i CSS), omogućujući potpunu kontrolu nad ponašanjem i izgledom web stranice.
- **Upotreba na strani servera (pozadinski poslužitelj)**: Korištenjem Node.js, JavaScript se može koristiti i na strani servera za izradu pozadinskog sučelja sustava, rukovanje HTTP zahtjevima, rad s bazama podataka i drugo.

2.4. Tailwind CSS

Tailwind CSS je programski okvir za stiliziranje koji omogućava kreiranje dizajna pomoću korisničkih klasa direktno u HTML-u ili JSX-u. Programski okvir Tailwind CSS je odabran zbog svoje fleksibilnosti i brzine u izradi responzivnih sučelja.

- **Utility-first pristup**: Tailwind CSS koristi pristup u kojem su sve stilizacije definirane kroz korisničke klase, kao što su `text-center`, `bg-blue-500`, ili `mt-4`. Ovaj pristup ubrzava proces izrade sučelja jer eliminira potrebu za pisanjem vlastitih CSS pravila.
- **Responzivni dizajn**: Tailwind CSS podržava mobile-first dizajn, što znači da je fokusiran na optimizaciju aplikacije za mobilne uređaje, ali istovremeno omogućava jednostavno prilagođavanje za veće ekrane pomoću klasa poput `md:` ili `lg:` za definiranje stilova specifičnih za veće rezolucije.
- **Customizabilnost**: Tailwind CSS nudi visok stupanj prilagodljivosti. Iako dolazi s unaprijed definiranim stilovima, omogućuje i jednostavnu modifikaciju pomoću *tailwind.config.js* datoteke, gdje se mogu prilagoditi boje, fontovi, razmaci i druge vrijednosti prema specifičnim potrebama projekta.

2.5. Google Firebase

Google Firebase je platforma koja nudi cjelovita rješenja za razvoj web i mobilnih aplikacija, posebno korisna za aplikacije u realnom vremenu. Firebase je korišten kao rješenje za pozadinsko sučelje u ovom projektu zbog jednostavne integracije s korisničkim sučeljem i bogatstva funkcionalnosti koje pruža.

- **Firebase Firestore**: Firestore je *NoSQL* baza podataka u oblaku koja omogućava pohranu podataka o korisnicima, terenima i rezervacijama. Firestore podržava real-time sinkronizaciju, što znači da se promjene u bazi podataka automatski ažuriraju na strani korisnika. U aplikaciji, Firestore se koristi za pohranu informacija o svim terenima dostupnim za rezervaciju, kao i praćenje korisničkih rezervacija.

- **Firebase Authentication**: Firebase Authentication je korišten za upravljanje korisničkim prijavama i registracijama. Podržava razne metode autentifikacije, uključujući prijavu putem e-maila i lozinke, Google računa i drugih društvenih mreža. Ova funkcionalnost je ključna za osiguravanje privatnosti i sigurnosti korisničkih podataka, kao i za personalizaciju korisničkog iskustva.

3. TEHNIČKA STRUKTURA I FUNKCIONALNOST

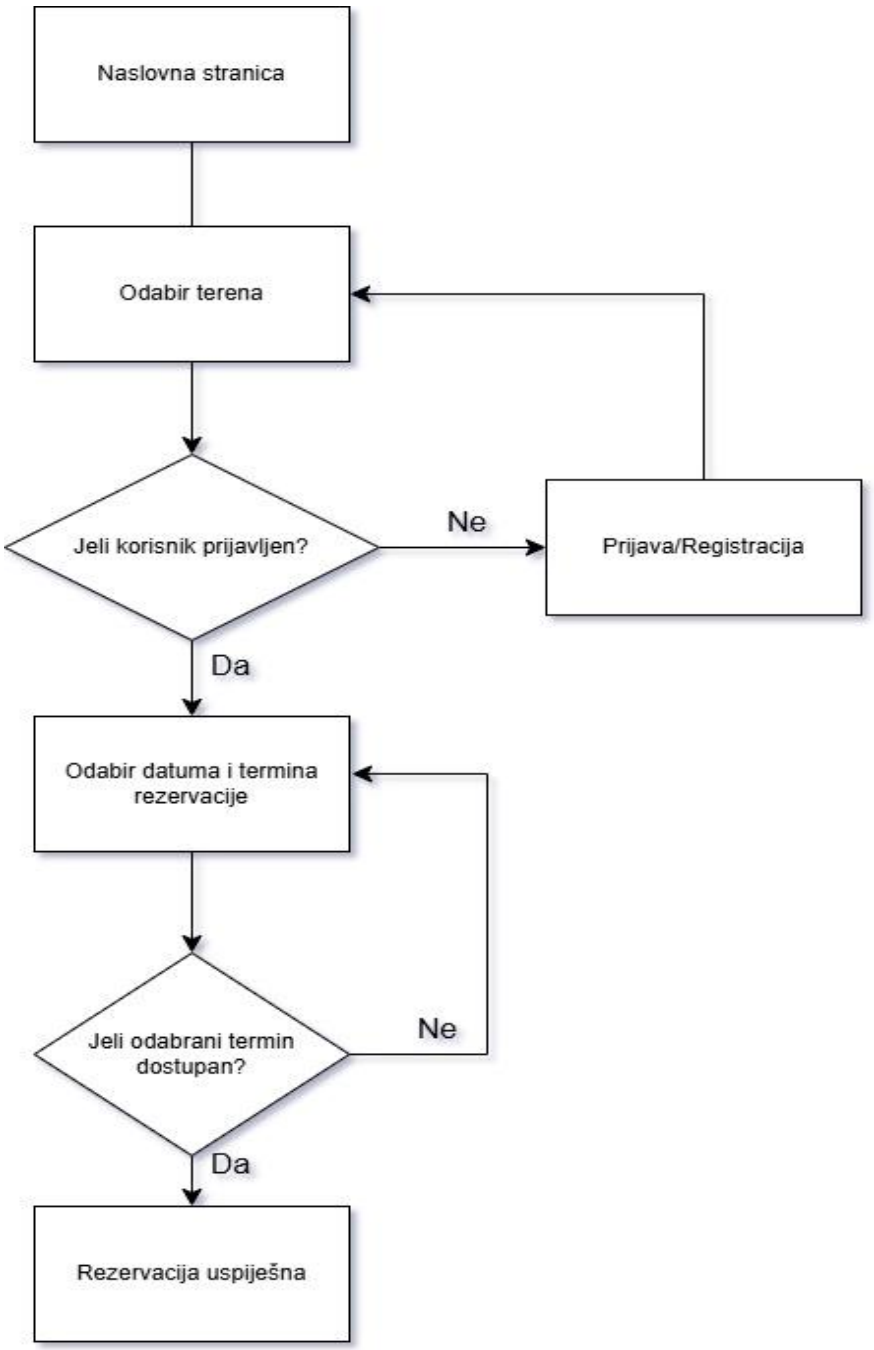
U ovom poglavlju detaljno je objašnjeno kako ova web aplikacija za rezervaciju malonogometnih terena funkcionira, uključujući arhitekturu sustava, ključne funkcionalnosti, dijagrame i primjere implementacije. Aplikacija koristi klijent-poslužitelj arhitekturu, gdje korisničko sučelje i pozadinski poslužitelj komuniciraju putem API poziva. Korisničko sučelje je dio aplikacije koji je izrađen pomoću programskog okvira React, dok je pozadinsko sučelje implementirano koristeći Google Firebase, uključujući Firestore bazu podataka za pohranu i upravljanje podacima.

3.1. Tijek rada aplikacije

Korisnici aplikacije mogu pretraživati dostupne malonogometne terene na temelju lokacije i termina. Aplikacija omogućuje pregled detalja o svakom terenu, uključujući cijenu najma, broj igrača za koje je teren pogodan, te slobodne termine. Nakon odabira terena, korisnik može rezervirati željeni termin, pod uvjetom da je dostupan, birajući datum i vrijeme. Uz mogućnost pretrage i rezervacije, aplikacija omogućuje korisnicima registraciju i prijavu, nakon čega mogu pregledavati i upravljati vlastitim rezervacijama. Korisnici također imaju pristup rezervacijama, gdje mogu pregledavati nadolazeće termine, kao i otkazivati buduće rezervacije prema potrebi.

3.1.1. Dijagram toka

Na dijagramu toka na slici 1, koja se nalazi ispod, prikazani su ključni koraci koje korisnik prolazi unutar aplikacije za rezervaciju malonogometnih terena. Na temelju dijagrama, možemo pratiti cijeli proces od početne stranice pa sve do uspješne rezervacije.



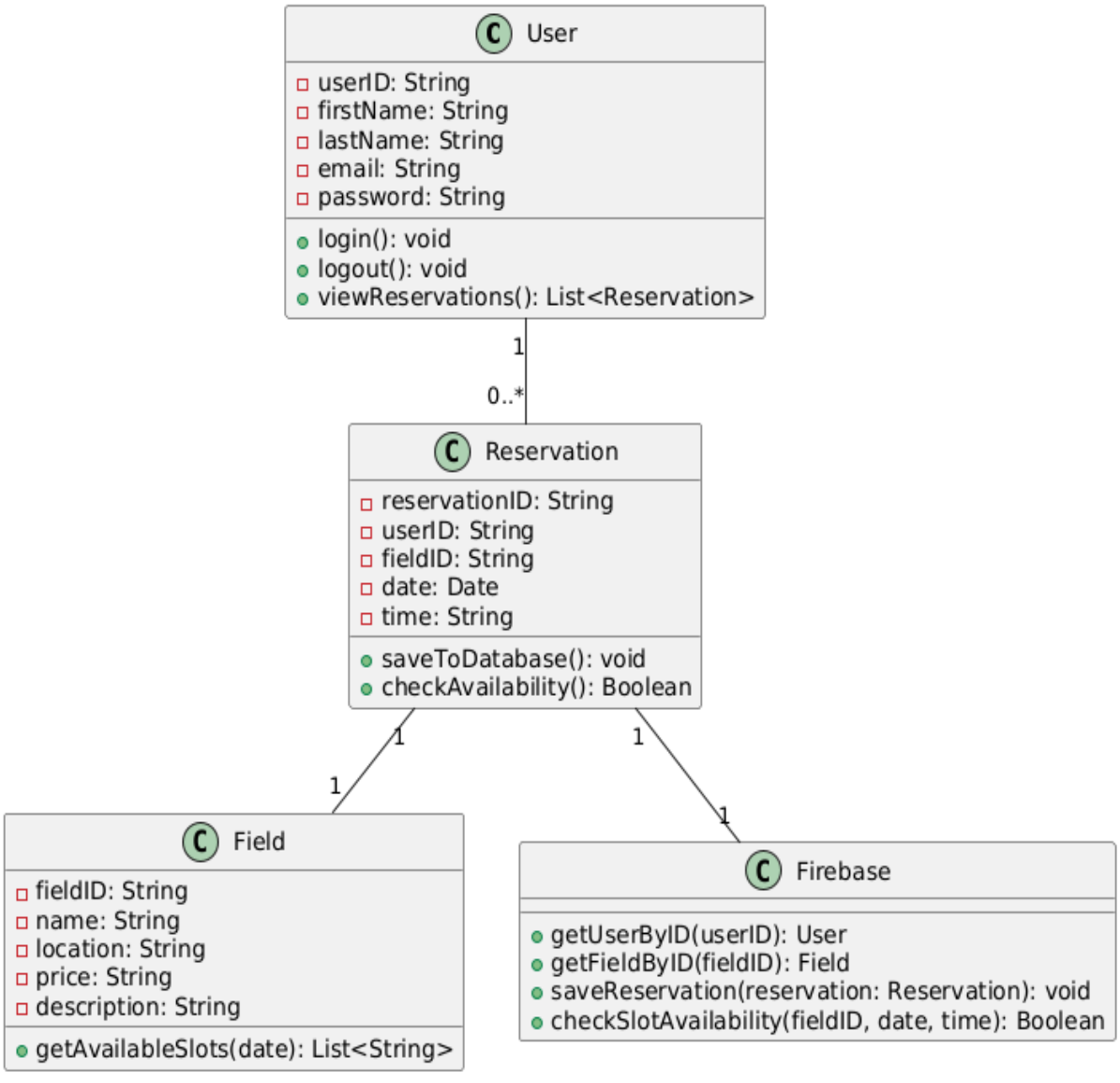
Slika 1. Dijagram toka rada aplikacije

Korisnik započinje na početnoj stranici aplikacije. Na ovoj stranici korisniku su dostupne osnovne informacije o aplikaciji, kao što su stranice "O nama" i stranica "Kontakt", te opcije za prijavu/registraciju. Kada korisnik odabere opciju za pregled terena, aplikacija prvo provjerava je li korisnik registriran ili prijavljen. Ako korisnik nije prijavljen, aplikacija ga preusmjerava na stranicu za prijavu ili registraciju. U slučaju da je korisnik već prijavljen, nastavlja se s pregledom dostupnih terena. Nakon toga, korisnik odabire željeni teren, a aplikacija ga vodi na stranicu za odabir termina. Korisniku se omogućuje odabir termina prema

datumu i vremenu, ali samo ako je željeni termin dostupan. Ukoliko je odabrani termin nedostupan, rezervacija nije moguća i korisnik mora odabrati neki od drugih dostupnih termina. Ako korisnik odabere dostupan termin, rezervacija se uspješno bilježi u bazi podataka te korisnik dobiva obavijest o uspješnoj rezervaciji.

3.1.2. Dijagram klasa aplikacije

Dijagram klasa prikazuje ključne entitete unutar aplikacije za rezervaciju malonogometnih terena i njihove međusobne odnose.



Slika 2. Dijagram klasa aplikacije

Dijagram uključuje četiri glavne klase: *User*, *Reservation*, *Field*, i *Firebase*, koje prikazuju različite aspekte aplikacije, uključujući korisnike, terene, rezervacije i interakciju s bazom podataka.

Klasa *User* predstavlja korisnike aplikacije, pri čemu svaki korisnik ima jedinstveni identifikator (`userID`), ime, prezime, email adresu i lozinku. Korisnik može izvršiti određene radnje, kao što su prijava u aplikaciju, odjava te pregled svojih rezervacija. Na ovaj način korisnik može upravljati vlastitim rezervacijama i pregledavati svoje aktivnosti unutar aplikacije.

Klasa *Reservation* služi za pohranu podataka o rezervacijama koje korisnik izvršava. Svaka rezervacija povezana je s jednim korisnikom i jednim terenom. Rezervacija sadrži informacije poput jedinstvenog identifikatora rezervacije (`reservationID`), korisničkog ID-a (`userID`), ID-a terena (`fieldID`), datuma i vremena rezervacije. Metode unutar klase omogućuju spremanje rezervacije u bazu podataka te provjeru dostupnosti termina za određeni teren. Na taj način, aplikacija osigurava da korisnik može rezervirati termin samo ako je on slobodan.

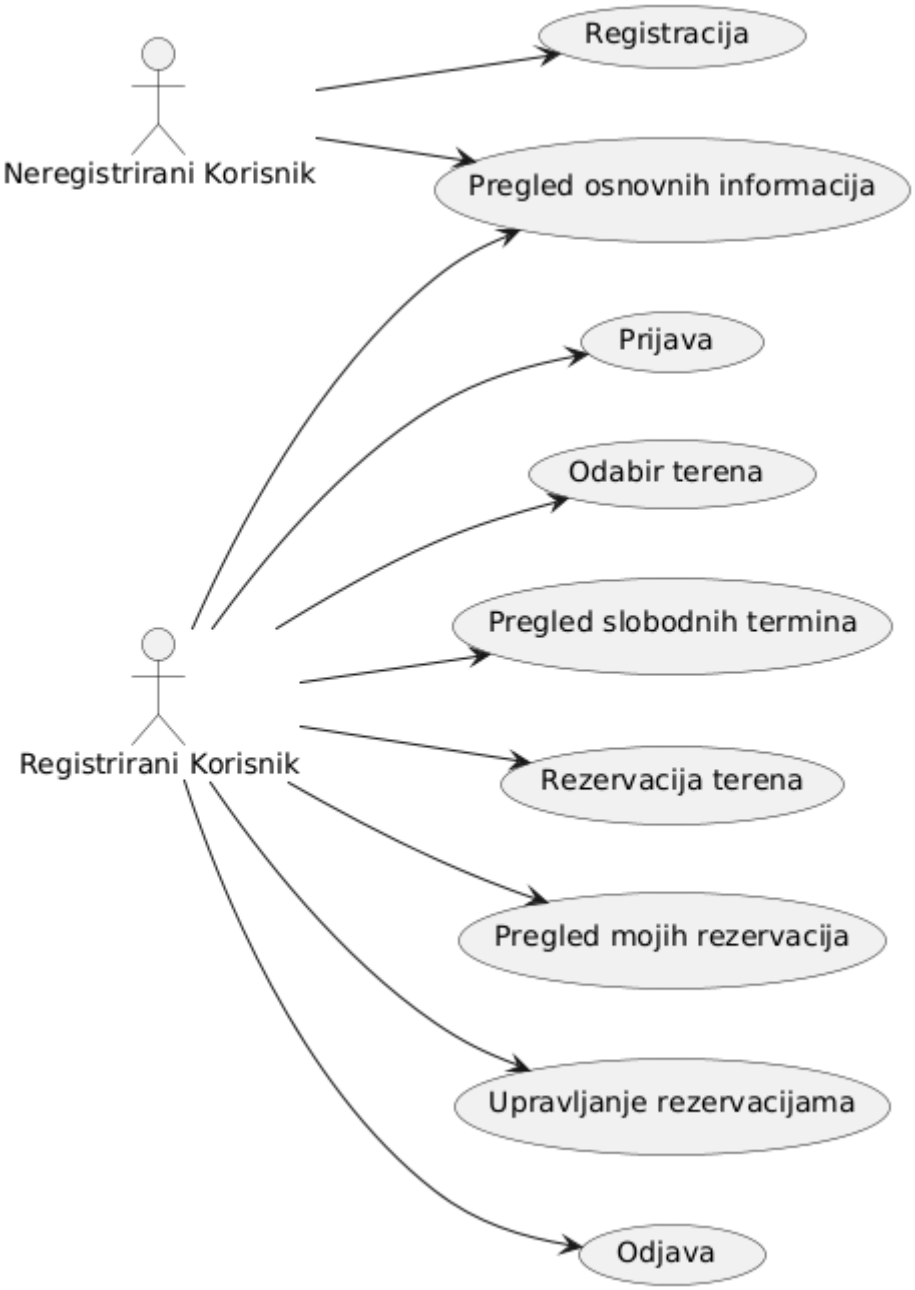
Klasa *Field* predstavlja terene dostupne za rezervaciju. Svaki teren ima svoj jedinstveni identifikator, naziv, lokaciju, cijenu najma te opis koji daje korisnicima osnovne informacije o terenu. Osim toga, metoda unutar klase omogućuje dohvaćanje slobodnih termina za određeni datum, što korisniku olakšava planiranje i izbor.

Klasa *Firebase* služi kao veza između aplikacije i baze podataka. Ova klasa omogućava dohvaćanje podataka o korisnicima, terenima i rezervacijama iz Firebase Firestore baze podataka te spremanje novih rezervacija u bazu. Također, omogućuje provjeru dostupnosti termina za odabrani teren i određeni datum. Firebase klasa pruža ključne funkcionalnosti za upravljanje podacima unutar aplikacije te omogućuje da svi podaci budu pohranjeni i ažurirani u stvarnom vremenu.

Sve četiri klase međusobno su povezane u složenom sustavu rezervacija. Korisnik može izvršiti više rezervacija, a svaka rezervacija povezana je s jednim terenom. Klasa Firebase upravlja pohranom i dohvaćanjem podataka u stvarnom vremenu, čime osigurava pravilan rad aplikacije i dostupnost svih potrebnih informacija korisnicima. Ovaj dijagram klasa detaljno prikazuje kako aplikacija funkcionira i kako su ključni entiteti međusobno povezani kako bi omogućili jednostavnu i sigurnu rezervaciju terena.

3.1.3. Dijagram slučaja uporabe

Dijagram slučaja uporabe prikazuje funkcionalnosti koje su dostupne neregistriranim i registriranim korisnicima u aplikaciji za rezervaciju malonogometnih terena.



Slika 3. Dijagram slučaja uporabe aplikacije

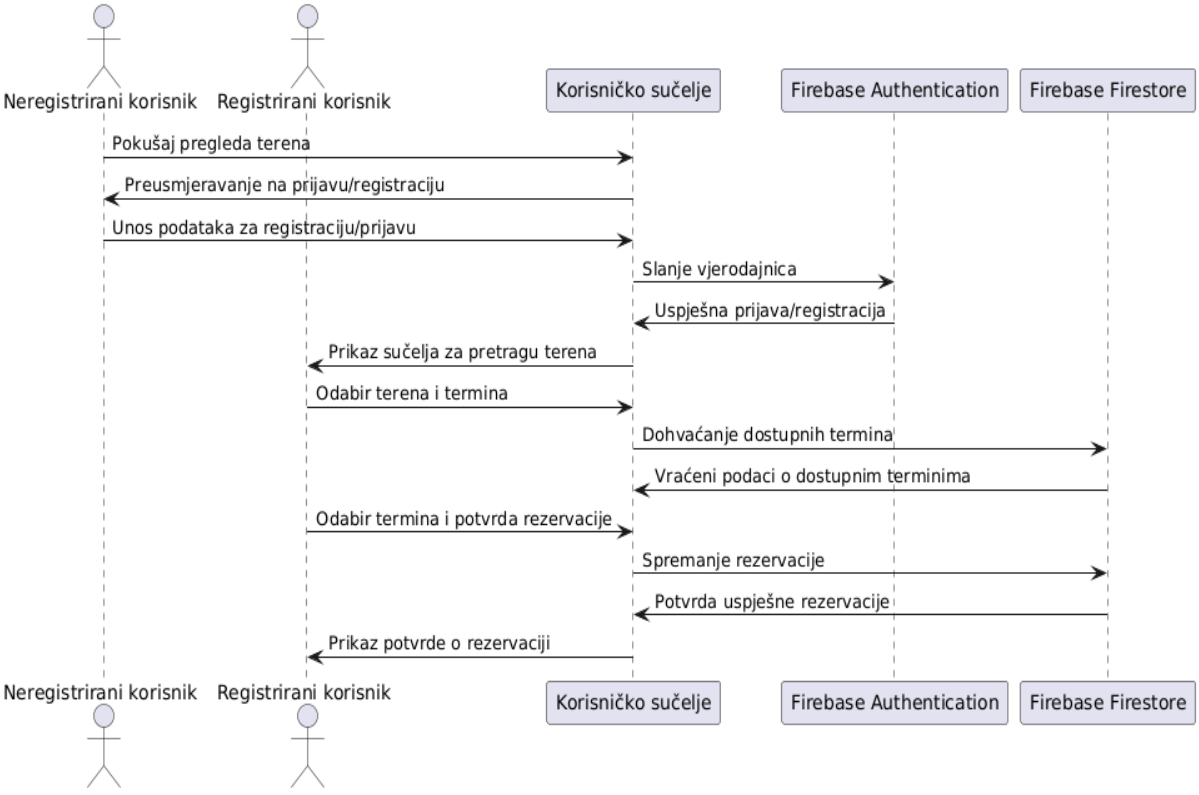
Na vrhu dijagrama prikazanog na slici 3, neregistrirani korisnik ima dvije glavne funkcije. Prvo, može pregledati osnovne informacije o aplikaciji putem funkcionalnosti "Pregled osnovnih informacija", gdje mu je omogućeno da vidi javno dostupne podatke. Također, neregistrirani korisnik može izvršiti registraciju kako bi dobio pristup dodatnim funkcionalnostima unutar aplikacije. Nakon registracije i prijave, korisnik postaje registrirani korisnik, kojem je dostupno znatno više funkcionalnosti. Najprije, registrirani korisnik se može prijaviti u aplikaciju. Nakon prijave, korisniku se pružaju razne mogućnosti, poput odabira terena, gdje može pretraživati dostupne sportske terene. Nakon odabira terena, korisnik može

pregledavati slobodne termine i odabrati odgovarajući termin za rezervaciju. Sljedeći korak je rezervacija terena, gdje korisnik može rezervirati odabrani teren i termin. Nakon što rezervira teren, registrirani korisnik može koristiti funkcionalnosti za pregled svojih rezervacija te može upravljati rezervacijama, što uključuje izmjene ili otkazivanje rezervacija. Kada završi s korištenjem aplikacije, korisnik se može odjaviti iz sustava. Ovaj dijagram jasno prikazuje funkcionalnosti koje su specifične za registrirane i neregistrirane korisnike, te kako se mogućnosti proširuju nakon uspješne registracije i prijave u aplikaciju.

3.1.4. Sekvencijski dijagram

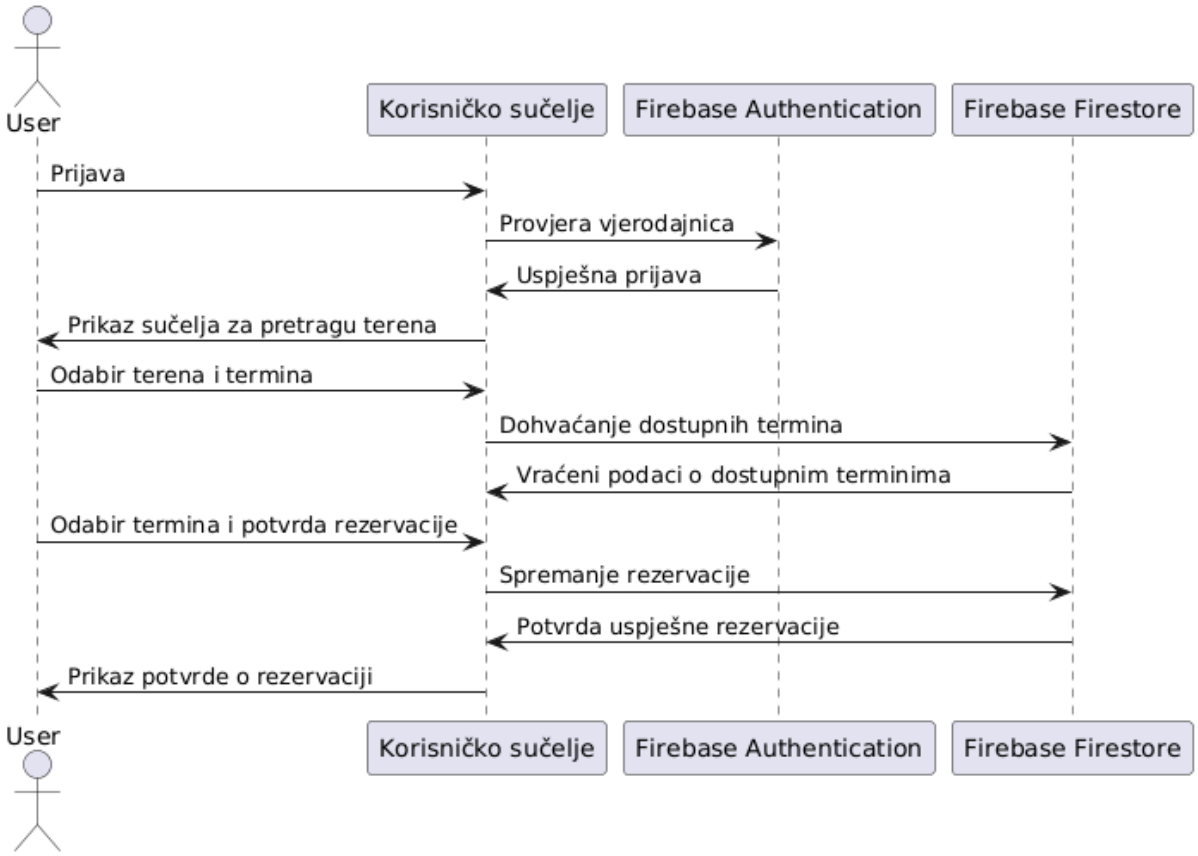
Kod sekvencijskog dijagrama imamo dvije različite situacije u kojima se korisnici mogu naći: jedna prikazuje proces za neregistriranog korisnika, dok druga prikazuje tijek radnji za registriranog korisnika tijekom procesa rezervacije malonogometnog terena.

U prvom scenariju, neregistrirani korisnik pokušava pregledavati terene ili izvršiti rezervaciju, ali aplikacija prepoznaje da korisnik nije prijavljen. Kao rezultat toga, aplikacija preusmjerava korisnika na stranicu za prijavu ili registraciju. Korisnik unosi svoje podatke za prijavu ili registraciju, a aplikacija šalje te podatke Firebase Authentication servisu radi provjere autentičnosti. Nakon što Firebase potvrdi vjerodajnice, korisnik se uspješno prijavljuje i aplikacija mu prikazuje sučelje za pretragu terena. Korisnik zatim može nastaviti s pregledom terena i odabirom termina, kao što bi to učinio registrirani korisnik.



Slika 4. Sekvencijski dijagram rezervacije termina kod neregistriranog korisnika

U drugom scenariju, registrirani korisnik ima direktan pristup pregledu terena. Nakon prijave, korisnik bira željeni teren i termin te aplikacija šalje upit Firebase Firestore bazi podataka kako bi provjerila dostupnost termina za odabrani datum. Baza vraća podatke o slobodnim i zauzetim terminima, a korisnik zatim odabire jedan od dostupnih termina. Nakon potvrde termina, aplikacija pohranjuje podatke o rezervaciji u Firebase Firestore. Nakon uspješno izvršene rezervacije, aplikacija vraća potvrdu korisniku da je rezervacija uspješno zabilježena.



Slika 5. Sekvencijski dijagram rezervacije termina kod registriranog korisnika

Dijagramima na slikama 4 i 5 jasno su prikazane razlike između neregistriranih i registriranih korisnika u aplikaciji. Neregistrirani korisnici najprije moraju proći proces prijave ili registracije kako bi dobili pristup punim funkcionalnostima aplikacije, dok registrirani korisnici odmah mogu pregledavati terene, odabrati termin i izvršiti rezervaciju.

3.2. Arhitektura sustava

Aplikacija koristi klijent-poslužitelj arhitekturu, pri čemu su korisničko sučelje i pozadinsko sučelje odvojeni, ali međusobno komuniciraju putem REST API ili Firebase SDK-a. Korisničko sučelje je izrađeno pomoću programskog okvira React, dok je za pozadinsko

sučelje korišten Google Firebase, uključujući Firestore bazu podataka za pohranu podataka o korisnicima, terenima i rezervacijama.

3.2.1. Arhitektura korisničkog sučelja

Korisničko sučelje je izrađeno pomoću programskog okvira React, koji omogućuje modularnost putem komponenti. Korištenjem React Hooks-a kao što si *useState* i *useEffect*, aplikacija upravlja lokalnim stanjem i asinkronim dohvaćanjem podataka iz baze. Korisničko sučelje (UI) stilizirano je pomoću programskog okvira Tailwind CSS-a, što omogućuje responzivnost i konzistentnost sučelja na svim uređajima.

Korištenjem programskog okvira React korisničko sučelje organizirano je u jasno definirane direktorije, što omogućuje lako održavanje, proširivanje i skalabilnost. Komponente su grupirane prema funkcionalnosti unutar aplikacije, čime se osigurava logički slijed u razvoju i olakšava suradnja unutar tima. Svaka glavna funkcionalnost ima svoj direktorij, a unutar njih se nalaze potrebne komponente i stilovi. Prikaz strukture aplikacije:

```
MY-PROJECT
|
├── .firebase/            # Firebase konfiguracijski podaci
├── .vscode/              # Postavke Visual Studio Code-a
├── build/                # Build datoteke za produkciju
├── functions/            # Firebase funkcije
├── node_modules/                       # Moduli koji se koriste u aplikaciji
├── public/                             # Javno dostupne statične datoteke
|
├── src/                                # Glavni izvorni kod aplikacije
|   ├── components/                      # Zajedničke komponente aplikacije
|   ├── firebase/                        # Firebase konfiguracija i usluge
|   ├── helpers/                         # Pomoćne funkcije i datoteke
|   ├── pages/                           # Razne stranice aplikacije
|   |   ├── about/                      # Stranica "O nama"
|   |   |   └── About.js                # Komponenta stranice "O nama"
|   |   ├── contact/                    # Stranica za kontakt
|   |   |   └── Contact.js              # Komponenta za kontakt stranicu
|   |   ├── head-foot/                  # Zaglavlje i podnožje aplikacije
|   |   |   ├── Footer.js               # Komponenta podnožja
|   |   |   └── Header.js               # Komponenta zaglavlja
|   |   ├── home/                       # Stranica naslovnice
|   |   |   ├── components/             # Dodatne komponente naslovnice
|   |   |   └── Home.js                 # Glavna komponenta za prikaz naslovnice
|   |   ├── login/                      # Stranica za prijavu
|   |   |   └── Login.js                # Komponenta za prijavu
|   |   ├── player/                     # Stranica profila korisnika
|   |   |   └── Player.js               # Komponenta za korisnički profil
|   |   ├── registration/               # Stranica za registraciju
|   |   |   └── Registration.js         # Komponenta za registraciju
|   |   └── reservation/                # Stranica za rezervacije
|   |       └── components/
|   |           └── Reservation.js      # Komponenta za upravljanje rezervacijama
|
├── App.css               # Glavni stilovi aplikacije
├── App.test.js           # Testovi za aplikaciju
└── index.css             # Globalni stilovi aplikacije
└── index.js              # Glavna ulazna točka aplikacije
```

Slika 6. Struktura aplikacije koristeći programski okvir React

Kao što je vidljivo iz strukture, aplikacija je modularno organizirana putem komponenti. Svaka komponenta ima jasnu ulogu, što omogućava njihovo ponovno korištenje u različitim dijelovima aplikacije. Na primjer *ScrollToTopButton.js* je komponenta koja se može koristiti na različitim stranicama kako bi korisnicima omogućila povratak na vrh stranice. *Header.js* i *Footer.js* su višekratno upotrebljive komponente koje se prikazuju na svim stranicama aplikacije.

```
const About = () => {
  return (
    <div>
      <Header />
      <div className="mt-2 font-teachers">
        <div className="bg-white p-8 rounded-xl shadow-md w-full h-full
mb-2">
          <div className="flex justify-center mb-8">
            <img
              src={logo}
              className="w-1/2 md:w-1/3 rounded-lg shadow-sm"
              alt="About Us"
            />
          </div>
          <div className="text-center">
            <h1 className="text-4xl md:text-6xl font-bold
text-primary-0">
              O Nama
            </h1>
```

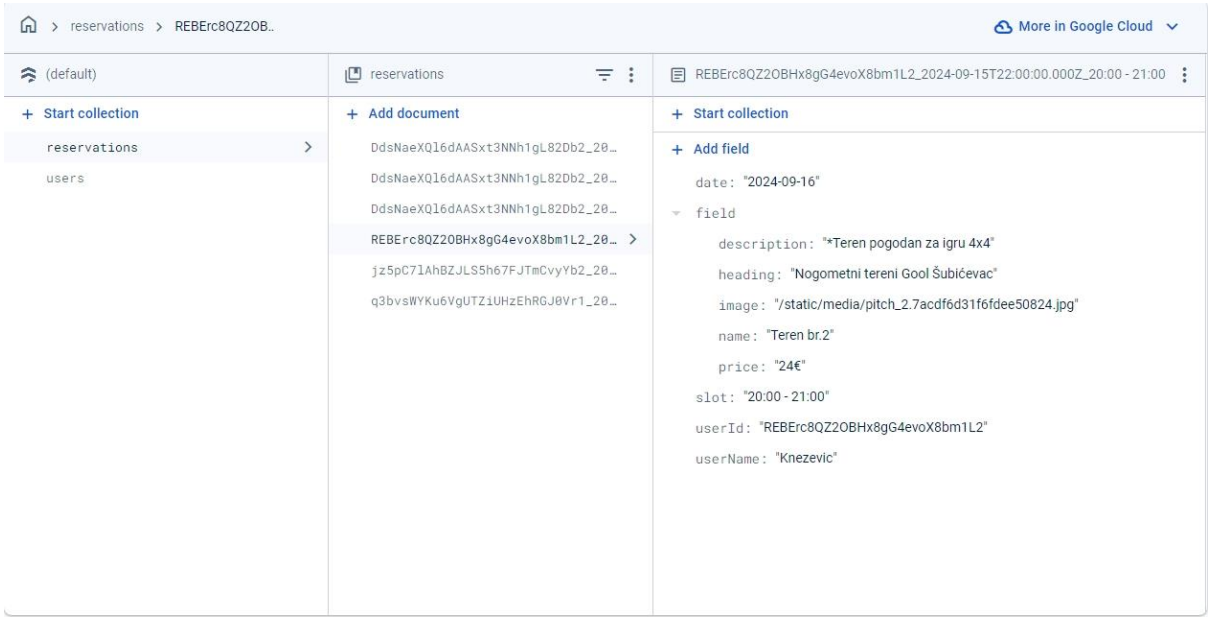
Kod 1. Primjer korištenja programskog okvira Tailwind CSS unutar koda

Koristeći programski okvir Tailwind CSS omogućujemo definiranje stilova putem utility klasa, čime se eliminira potreba za pisanjem vlastitih CSS pravila. U primjeru stranice *About.js*, vidimo korištenje Tailwind CSS-a za stilizaciju elemenata unutar komponente. Korištenje utility klasa omogućuje brzo definiranje stilova bez potrebe za pisanjem vlastitih CSS pravila. Tailwind CSS pruža mnoštvo klasa koje olakšavaju postavljanje rasporeda, boja, dimenzija i drugih stilskih aspekata elemenata. Neke od korištenih Tailwind CSS klasa su:

- **`mt-2`**: Postavlja marginu na vrhu elementa s vrijednošću od 0.5rem (*margin-top*).
- **`bg-white`**: Postavlja pozadinsku boju elementa na bijelu.
- **`rounded-xl`**: Stvara zaobljene rubove velikog radijusa (*extra-large*).
- **`justify-center`**: Koristi se u Flexbox rasporedu kako bi centriralo elemente horizontalno.
- **`text-4xl`**: Povećava veličinu teksta na 4xl (32px u osnovnoj konfiguraciji).

3.2.2. Arhitektura i funkcionalnosti pozadinskog sučelja

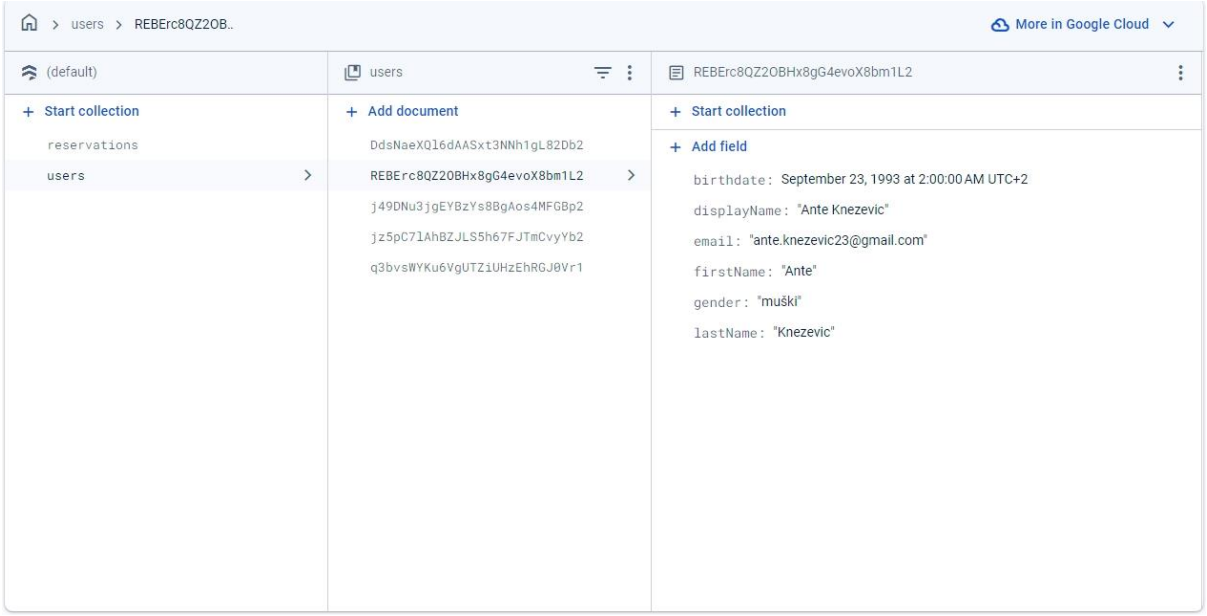
Firebase je Googleova platforma koja nudi niz alata i usluga za razvoj web i mobilnih aplikacija, a u ovoj aplikaciji koristi se za funkcionalnosti pozadinskog sučelja , uključujući pohranu podataka i autentifikaciju korisnika. Jedan od ključnih dijelova Firebasea koji se koristi u ovoj aplikaciji je Firestore, *NoSQL* baza podataka koja omogućuje pohranu podataka u obliku dokumenata unutar kolekcija. Firestore je skalabilan i omogućuje real-time sinkronizaciju podataka između klijenta i servera. Ova funkcionalnost omogućuje aplikaciji da trenutačno ažurira dostupne termine i rezervacije kada korisnik unese promjene ili izvrši novu rezervaciju. Podaci su organizirani u dvije glavne kolekcije: *users* i *reservations*. Kolekcija *users* pohranjuje osnovne informacije o korisnicima, uključujući korisnički ID (*userID*), ime, prezime i email adresu. Ovi podaci omogućuju povezivanje korisnika s njihovim rezervacijama i upravljanje pristupom unutar aplikacije. Rezervacije se pohranjuju u kolekciji *reservations*, gdje svaki dokument sadrži informacije o terenu, datumu i vremenu rezervacije, a povezan je s korisničkim ID-om. Firestore omogućuje jednostavno i brzo dohvaćanje i pohranu podataka, što je ključno za korisničko iskustvo. Real-time sinkronizacija podataka omogućuje trenutačno ažuriranje informacija o dostupnim terminima, tako da korisnici mogu odmah vidjeti slobodne termine i izvršiti rezervacije bez potrebe za osvježavanjem stranice.



Slika 7. Primjer uspješne rezervacije unutar Google Firebase Firestore-a

Slika 7 je primjer koji prikazuje dokument iz kolekcije *reservations* unutar *Firebase Firestore* baze podataka. Dokument sadrži detalje o rezervaciji, uključujući datum rezervacije (*date:* „*2024-08-21*“), kao i informacije o terenu. Podaci o terenu uključuju opis („*teren pogodan za igru 4x4“*), naziv terena („*Nogometni Tereni Rasadnik Vidici*“), cijenu najma („*24€“*), te

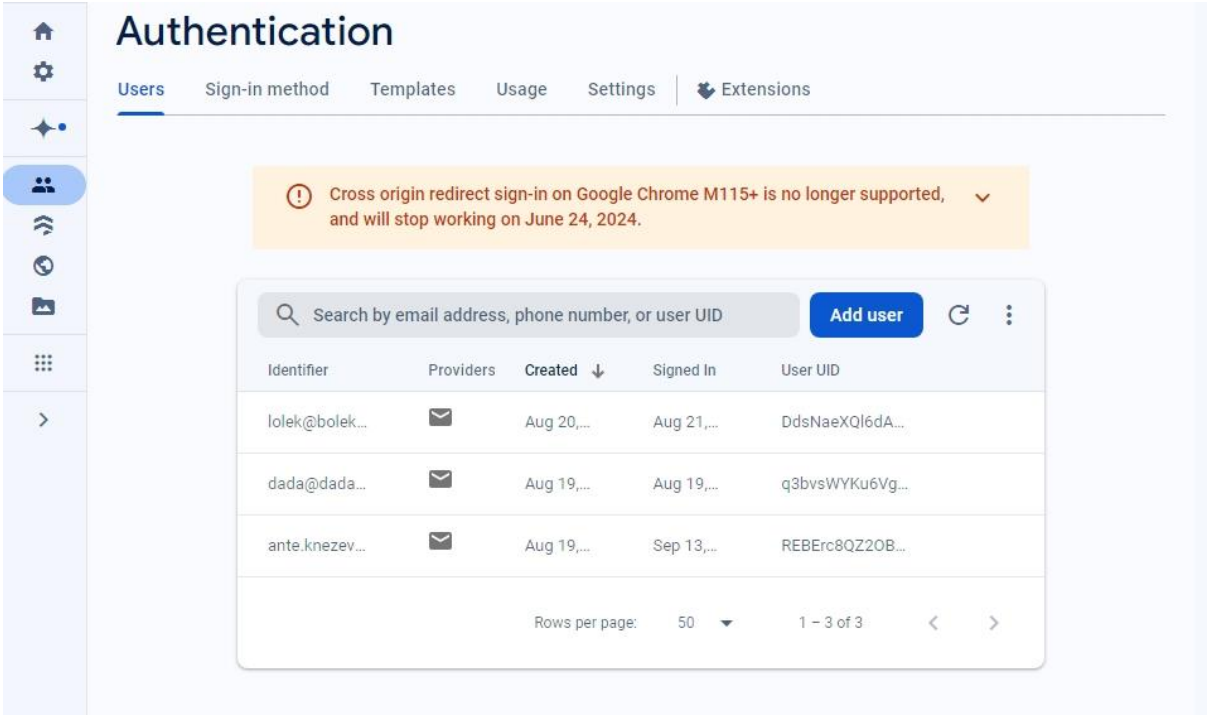
vremenski termin rezervacije („*20:00 - 21:00*“). Također su pohranjeni podaci o korisniku koji je izvršio rezervaciju, uključujući korisnički ID (*userId*) i ime korisnika (*userName*: „*Knezevic*“). Ovaj dokument omogućuje aplikaciji praćenje rezervacija i povezivanje svake rezervacije s određenim korisnikom i terminom.



Slika 8. Primjer pohrane podataka o korisnicima unutar Google Firebase Firestore-a

Na slici 8 je prikazan pregled Firebase Firestore baze podataka s kolekcijom *users*. U kolekciji se nalaze dokumenti koji predstavljaju korisnike aplikacije, svaki s jedinstvenim identifikatorom (*userID*). U ovom primjeru, odabrani korisnički dokument prikazuje detalje poput datuma rođenja (*birthdate*), prikazanog imena (*displayName*), email adrese (email), imena (*firstName*), prezimena (*lastName*) i spola (*gender*). Ova struktura omogućava spremanje i organiziranje korisničkih podataka u realnom vremenu, a podaci su dostupni aplikaciji za različite funkcionalnosti, uključujući prijavu i povezivanje s rezervacijama.

Još jedna od značajki Firebase-a a koja se koristi za izradu pozadinskog sučelja aplikacije je Firebase Authentication. Firebase Authentication pruža robustan sustav autentifikacije i autorizacije korisnika, čime osigurava sigurnost aplikacije i zaštitu korisničkih podataka. Na slici 10 koja se nalazi ispod je prikazan pregled korisnika unutar Firebase Authentication konzole, gdje se mogu vidjeti svi korisnički računi koji su registrirani u aplikaciji.



Slika 9. Prikaz Firebase Authentication konzole

Svaki korisnik ima svoj jedinstveni identifikator (*User UID*) koji se koristi za povezivanje korisničkih podataka s njihovim aktivnostima unutar aplikacije. Firebase Authentication omogućava aplikaciji da upravlja prijavom korisnika putem emaila i lozinke, a podržava i druge metode prijave, poput prijave putem društvenih mreža (Google, Facebook, itd.). Kada korisnik unese svoje vjerodajnice, Firebase provjerava autentičnost tih podataka i, ako su ispravni, dodjeljuje korisniku jedinstveni UID. Taj UID koristi se za autentifikaciju u svim narednim zahtjevima prema aplikaciji, čime se omogućava siguran pristup resursima kao što su rezervacije ili upravljanje korisničkim računom. Sigurnost podataka osigurava se nizom zaštitnih mjera. Firebase koristi enkripciju podataka u prijenosu (TLS) kako bi zaštitio informacije tijekom komunikacije između korisnika i servera. Osim toga, autentifikacija je integrirana s Firebase Firestoreom, što omogućuje restriktivan pristup bazama podataka na temelju korisničkih prava. Na primjer, neautentificirani korisnici nemaju pristup podacima rezervacija ili drugih korisničkih informacija. Firebase Authentication omogućava i kontrolu nad sigurnosnim postavkama, kao što su opcije za resetiranje lozinke, upravljanje prijavljenim sesijama te zaštita od neovlaštenih pristupa. Time se osigurava da samo ovlašteni korisnici mogu pristupiti osjetljivim podacima unutar aplikacije, a sve transakcije između korisnika i aplikacije pažljivo su zaštićene kako bi se spriječila bilo kakva sigurnosna prijetnja.

3.3. Detaljni opis funkcionalnosti

Web aplikacija pruža intuitivno korisničko iskustvo od trenutka registracije pa sve do uspješne rezervacije terena. Kroz ovaj vodič, opisat ću ključne korake koje korisnik prolazi, počevši od stvaranja računa.

3.3.1. Registracija i prijava korisnika

Prvenstveno je važno naglasiti da ništa u aplikaciji nije moguće koristiti bez prethodne registracije i prijave. Na primjer, ako korisnik pokušava pregledati terene ili izvršiti rezervaciju, aplikacija automatski provjerava je li korisnik već prijavljen. Ukoliko korisnik nije prijavljen ili registriran, klikom na bilo koje dugme, poput „*Pregled terena*“, aplikacija ga preusmjerava na stranicu za prijavu ili registraciju. Ovaj mehanizam osigurava da samo ovlašteni i prijavljeni korisnici imaju pristup naprednim funkcionalnostima aplikacije, poput pregleda slobodnih termina i rezervacije terena. Firebase Authentication omogućuje jednostavno praćenje statusa prijave, a React-router-dom se koristi za preusmjeravanje korisnika na odgovarajuću stranicu ovisno o njegovom statusu prijave. Stoga, prvi korak u korištenju aplikacije je registracija novog korisnika. Korisnik unosi osnovne podatke kao što su ime, prezime, email adresa i lozinka. Ovi podaci šalju se Firebase Authentication servisu, koji kreira korisnički račun i dodjeljuje korisniku jedinstveni UID. Ovaj UID koristi se za povezivanje korisnika s podacima u bazi podataka. Prikaz koda za registraciju je prikazan u slikama ispod.

```
const Registration = () => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [firstName, setFirstName] = useState("");
  const [lastName, setLastName] = useState("");
  const [gender, setGender] = useState("");
  const [birthdate, setBirthdate] = useState(new Date());
  const [error, setError] = useState(null);

  const navigate = useNavigate();

  const handleRegistration = async () => {
    try {
      const userCredential = await createUserWithEmailAndPassword(
        auth,
        email,
        password
      );
      const user = userCredential.user;

      await setDoc(doc(db, "users", user.uid), {
        firstName: firstName,
        lastName: lastName,
        email: email,
        gender: gender,
        birthdate: birthdate,
        displayName: `${firstName} ${lastName}`,
      });
```

Kod 2. Prikaz koda registracije korisnika

Dio koda 2 prikazuje funkciju registracije korisnika u aplikaciji, koristeći React i Firebase Authentication. Na početku funkcije definirani su React state hook-ovi koji se koriste za pohranjivanje vrijednosti koje korisnik unosi u formu. Email adresa, lozinka, ime, prezime, spol, datum rođenja i eventualne greške tijekom procesa registracije pohranjuju se u različite varijable stanja (`state`). Ove varijable omogućuju praćenje promjena u podacima koje korisnik unosi u realnom vremenu. Funkcija `handleRegistration` je asinkrona funkcija koja upravlja procesom registracije korisnika. Kada se pozove, najprije se pokušava kreirati novi korisnički račun putem Firebase Authentication servisa koristeći metodu `createUserWithEmailAndPassword`. Ova metoda prima email i lozinku koje je korisnik unio, a ako je uspješna, vraća `userCredential` objekt koji sadrži podatke o novom korisniku. Zatim se iz tog objekta izvlači korisnički ID (`user.uid`), koji je jedinstveni identifikator korisnika u Firebaseu. Nakon uspješne registracije, aplikacija koristi Firebase Firestore kako bi spremila dodatne podatke o korisniku u bazu podataka. Funkcija `setDoc`

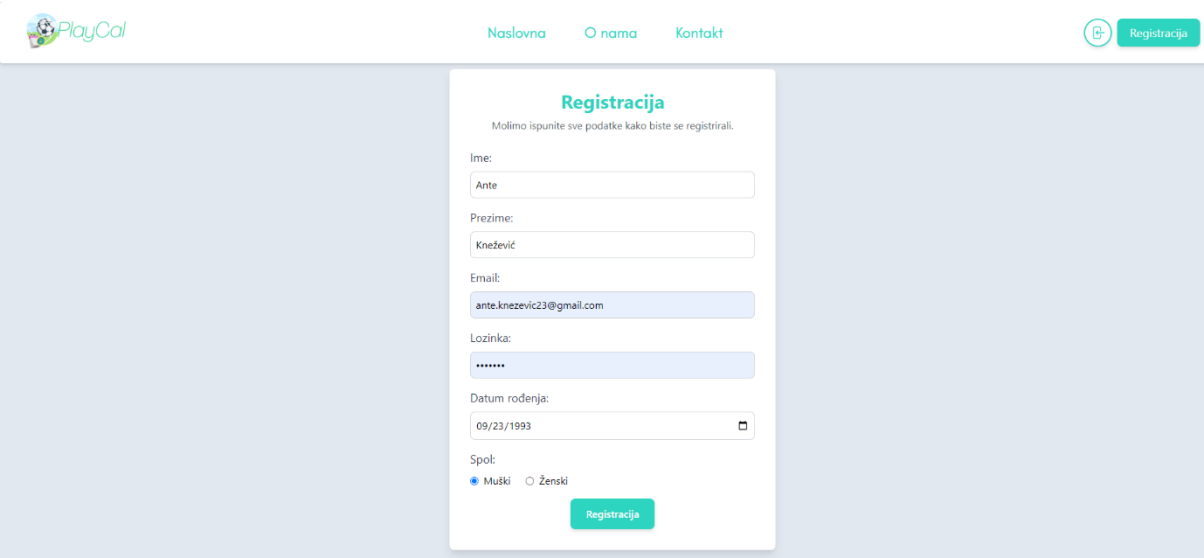
postavlja novi dokument unutar kolekcije „*users*“, koristeći korisnički ID kao identifikator dokumenta. U dokumentu se pohranjuju podaci kao što su ime, prezime, email, spol, datum rođenja, te prikazano ime koje se sastoji od imena i prezimena korisnika. Ovaj dio koda omogućuje jednostavno kreiranje novog korisničkog računa i spremanje dodatnih podataka o korisnicima u bazu podataka, čime se osigurava integracija korisničkih informacija unutar aplikacije.

```
console.log("Korisnik registriran i podaci spremljeni u
Firestore");

      navigate("/pitches");
    } catch (error) {
      setError(error.message);
    }
  };
```

Kod 3. Prikaz koda registracije korisnika – obavijest o uspiješnoj registraciji

Dio koda 3 prikazuje završni korak funkcije registracije. Nakon što je korisnik uspješno registriran i podaci su pohranjeni u Firebase Firestore, ispisuje se poruka u konzolu koja potvrđuje uspješnost iste. Funkcija `navigate` preusmjerava korisnika na stranicu „*pitches*“, gdje može pregledati dostupne terene. U slučaju da dođe do pogreške tijekom procesa registracije, catch-blok hvata grešku i postavlja njenu poruku u stanje `error`, kako bi se korisniku prikazala odgovarajuća poruka o problemu.

Ostatak koda odnosi se na uređenje same stranice, odnosno korisničkog sučelja, gdje se koristi programski okvir Tailwind CSS za stilizaciju elemenata. Tailwind CSS omogućuje brzu i jednostavnu primjenu stilova putem utility klasa, čime se osigurava responzivnost i moderan izgled stranice. Konačni prikaz stranice za registraciju korisnika je prikazan u slici 10 koja se nalazi ispod.



Slika 10. Prikaz izgleda stranice za registraciju korisnika

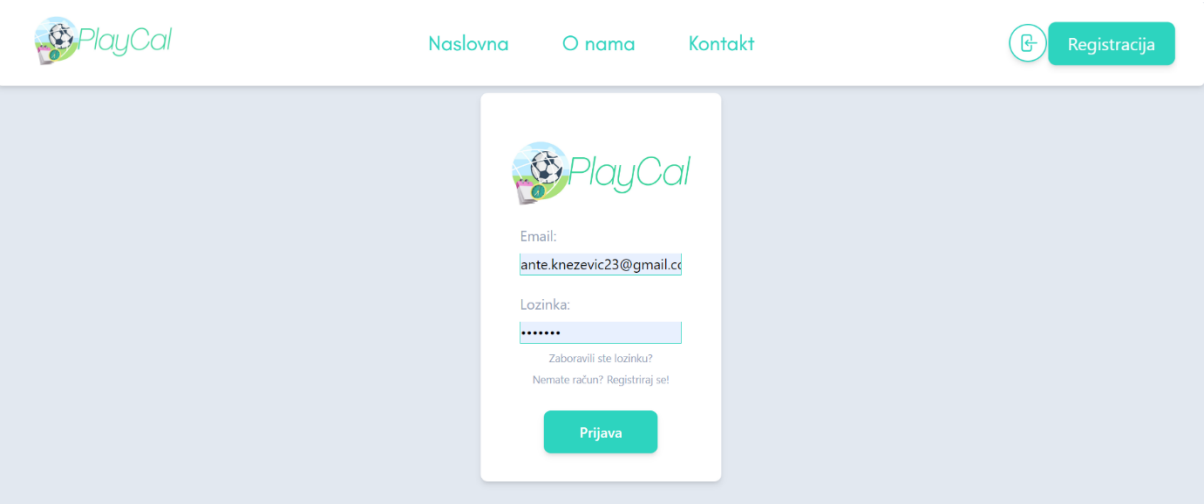
Nakon uspješne registracije, korisnik automatski ostaje prijavljen u aplikaciju i preusmjeren je na stranicu s pregledom dostupnih terena. Ovaj mehanizam omogućuje korisnicima da odmah počnu koristiti aplikaciju bez potrebe za dodatnom prijavom. No, u slučaju da korisnik već ima postojeći račun i dolazi na naslovnu stranicu, tada ćemo opisati postupak prijave, gdje će korisnik unijeti svoje vjerodajnice kako bi dobio pristup funkcionalnostima aplikacije.

Korisnik, kada odabere opciju prijave putem dugmeta u zaglavlju stranice (engl. *header*), biva preusmjeren na stranicu za prijavu. Na toj stranici korisnik unosi svoju email adresu i lozinku kako bi pristupio aplikaciji.

```
const LogIn = () => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [error, setError] = useState(null);
  const navigate = useNavigate();

  const handleLogin = async () => {
    try {
      const userCredential = await signInWithEmailAndPassword(
        auth,
        email,
        password
      );
      console.log("Korisnik prijavljen:", userCredential.user);

      // Preusmjeravanje nakon uspješne prijave
      navigate("/pitches");
    } catch (error) {
      setError(error.message);
    }
  };
```

Kod 4. Prikaz koda za stranicu prijave u aplikaciju

Prikazani kod 4 implementira funkcionalnost prijave koristeći programski okvir React i Firebase Authentication. Prvo, definiraju se dva stanja pomoću React Hooksa: `email` i `password`, koja pohranjuju unesene vrijednosti korisničke email adrese i lozinke. Također, koristi se *error* stanje za pohranu potencijalnih poruka o grešci prilikom prijave. Koristi se i `useNavigate` hook iz React-router-dom biblioteke kako bi se korisnika preusmjerilo na stranicu s terenima nakon uspješne prijave. Funkcija `handleLogin` je asinhrona i koristi Firebase Authentication za prijavu. Kada korisnik unese svoje vjerodajnice i pritisne dugme za prijavu, funkcija `signInWithEmailAndPassword` šalje te podatke Firebase Authentication servisu. Ako su podaci ispravni, korisnički podaci se pohranjuju u `userCredential` i ispisuje se poruka u konzoli koja potvrđuje uspješnu prijavu. Nakon toga, korisnik se preusmjerava na stranicu „*pitches*“ pomoću `navigate` funkcije. U slučaju greške tijekom prijave, catch-blok hvata grešku i postavlja odgovarajuću poruku u `error` stanje, kako bi korisniku bila prikazana poruka o neuspjeloj prijavi. Ostatak koda odnosi se na uređenje same stranice, odnosno korisničkog sučelja, gdje se koristi programski okvir Tailwind CSS za stilizaciju elemenata. Konačni prikaz stranice za prijavu je prikazan u slici 11 koja se nalazi ispod.



Slika 11. Prikaz stranice za prijavu

3.3.2. Odabir terena i izgled stranice za odabir terena

Na slici koda 5, prikazanoj ispod, je početak React komponente *Pitches*, koja se koristi za prikaz popisa malonogometnih terena. Komponenta vraća JSX kod unutar kojeg se koriste Tailwind CSS klase za stilizaciju elemenata. U gornjem dijelu komponente nalaze se važne komponente kao što su `Res_Heading` za prikazivanje naslova stranice, i `ScrollToTopButton`, koja omogućava korisnicima da se brzo vrate na vrh stranice. Ove komponente poboljšavaju korisničko iskustvo, a njihova funkcionalnost se može pretpostaviti iz naziva.

```
const Pitches = () => {
  return (
    <div className="mb-2">
      <Res_Heading />
      <div className="flex justify-center uppercase text-4xl
text-slate-600 italic bg-white px-10 py-4 my-2 rounded-lg
shadow-lg">
        <h1 className="border-b-4 border-slate-400 pb-2">Popis
terena</h1>
      </div>
      <div className="flex flex-col lg:flex-row lg:justify-between p-4">
        <div className="grid grid-cols-1 md:grid-cols-2
lg:grid-cols-4 gap-6">
          {[
            {
              name: "Tereni 'Gool' Godimento",
              imgSrc: godimento,
              hours: "Od 15h do 23h",
              price: "od 18€ do 24€",
              fields: 3,
              link: "/gool",
            },
            {
              name: "Tereni Rasadnik Vidici",
              imgSrc: rasadnik,
              hours: "Od 16h do 22h",
              price: "24€",
              fields: 1,
              link: "/rasadnik",
            },
```

Kod 5. Prikaz koda stranice za odabir terena

Nakon uvodnog dijela, komponenta prikazuje popis terena koji su organizirani unutar `grid` sustava kako bi bili responzivno prikazani na različitim veličinama zaslona. Koristeći Tailwind klase kao što su `grid-cols-1 md:grid-cols-2 lg:grid-cols-4`, sadržaj se prilagođava, tako da se na manjim zaslonima prikazuje jedan stupac, a na većim ekranima dva ili četiri stupca, čime se postiže optimalna responzivnost. Prikazani tereni uključuju važne informacije poput naziva, radnog vremena, cijene i broja dostupnih terena. Na primjer, teren „*Tereni 'Gool' Godimento*“ ima cijenu „*od 18€ do 24€*“, dok „*Tereni Rasadnik Vidici*“ ima cijenu „*od 24€*“. Ovi podaci su prikazani kao objekti koji sadrže atribute poput `name`,

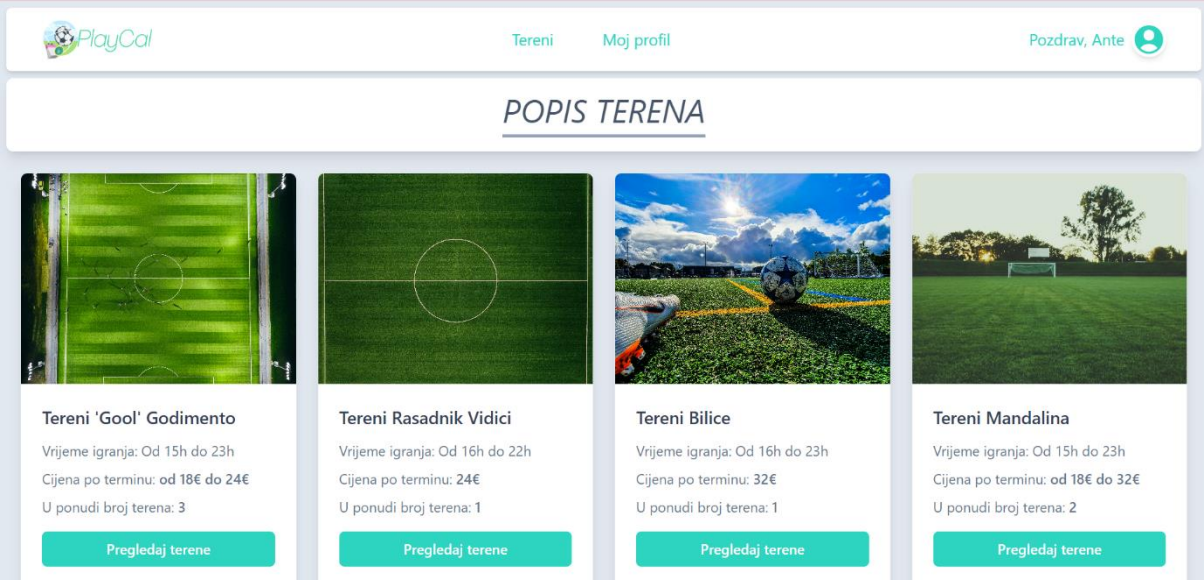
`imgSrc`, `hours`, `price`, `fields` i `link`, a kasnije će biti prikazani putem `map()` metode, oja iterira kroz svaki objekt i prikazuje podatke na stranici.

```
].map((pitch, index) => (
            <div
              key={index}
              className="bg-white shadow-lg rounded-lg
overflow-hidden border border-gray-200 transition-transform
transform hover:scale-105"
            >
              <img
                className="w-full h-60 object-cover"
                src={pitch.imgSrc}
                alt={pitch.name}
              />
              <div className="p-6">
                <h2 className="text-xl font-semibold
text-slate-700 mb-3">
                  {pitch.name}
                </h2>
                <p className="text-slate-500 mb-2">
                  Vrijeme igranja: {pitch.hours}
                </p>
                <p className="text-slate-500 mb-2">
                  Cijena po terminu:{" "}
                  <span className="font-semibold">{pitch.price}</span>
                </p>
                <p className="text-slate-500 mb-4">
                  U ponudi broj terena:{" "}
                  <span className="font-semibold">{pitch.fields}</span>
                </p>
                <Link
                  to={pitch.link}
                  className="block text-center bg-primary-0
text-white font-semibold py-2 px-4 rounded-md hover:bg-teal-300 transition
duration-300"
                >
                  Pregledaj terene
                </Link>
```

Kod 6. Prikaz koda stranice za odabir terena br. 2

Nakon što su podaci o terenima definirani u obliku objekata, prikazani dio koda koristi `.map()` metodu za iteraciju kroz svaki teren i dinamičko generiranje elemenata stranice. Svaki objekt `pitch` predstavlja pojedini teren, a metoda `.map()` uzima svaki teren iz niza i prikazuje ga unutar HTML strukture. Kroz div element s klasama `bg-white`, `shadow-lg`, `rounded-lg`, `overflow-hidden` stvaraju se kartice koje prikazuju pojedini teren.

Stilizacija kartica omogućena je putem Tailwind CSS klasa koje kartici dodaju bijelu pozadinu, zaobljene rubove i sjenu. Također, dodana je klasa `hover:scale-105`, koja omogućuje da se kartica malo poveća kada korisnik prijeđe mišem preko nje, čime se poboljšava interaktivnost korisničkog sučelja. Unutar svake kartice prvo se prikazuje slika terena, uz pomoć `img` elementa. Klase poput `w-full`, `h-60`, osiguravaju da slika zauzme cijelu širinu kartice, dok `object-cover` osigurava ispravno prilagođavanje slike unutar zadanih dimenzija bez gubitka proporcija. Konačni izgled stranice za odabir terena prikazan je u slici 12 koja se nalazi ispod.



Slika 12. Prikaz stranice za odabir terena

3.3.3. Odabir podterena i prikaz stranice podterena

U ovom poglavlju opisuje se koncept podterena unutar sportskih kompleksa, odnosno terena definiranih u prethodnom poglavlju. Svaki teren može biti kompleks koji se sastoji od jednog ili više podterena, što omogućuje veću fleksibilnost u korištenju i organizaciji sportskih objekata. Na primjer, kompleks „*Tereni Gool Godimento*“ sadrži tri podterena – *Teren 1*, *Teren 2* i *Teren 3*, dok kompleks „*Tereni Bilice*“ sadrži samo jedan teren, *Teren br. 1*. Ovaj koncept podterena važan je za korisnike jer im omogućuje precizniji odabir sportskog prostora, posebno u većim kompleksima s više opcija. Aplikacija automatski upravlja dostupnošću svakog podterena, olakšavajući korisnicima odabir željenog termina za rezervaciju. Kroz ovakvu organizaciju podterena unutar aplikacije, korisnici dobivaju jasniji pregled ponude i raspoloživosti unutar svakog sportskog kompleksa, što im omogućuje lakšu navigaciju i jednostavnije donošenje odluka pri rezervaciji. U ovom poglavlju prikazat ćemo koncept podterena kroz dva različita slučaja. Prvi slučaj odnosi se na terene koji imaju samo jedan

podteren, što znači da taj podteren predstavlja jedini teren u sklopu kompleksa. Drugi slučaj prikazuje terene koji imaju više podterena, pri čemu svaki podteren predstavlja zasebnu cjelinu unutar većeg kompleksa. Ova podjela omogućuje fleksibilnost prilikom organizacije terena te korisnicima pruža jasniji pregled mogućnosti prilikom rezervacije.

```
const Bilice = () => {
  const fieldData = {
    heading: "Nogometni Tereni Bilice",
    name: "Teren br. 1",
    price: "24€",
    description: "*Teren pogodan za igru 4x4",
    image: pitch,
  };

  return (
    <div className="font-teachers bg-slate-200 main-h-screen">
      <Res_Heading />
      <div className=" mx-auto p-8 bg-white rounded-lg shadow-lg">
        <div className="text-center mb-8">
          <h1 className="text-slate-700 font-bold text-4xl mb-4">
            {fieldData.heading}
          </h1>
          <img className="w-52 mx-auto mb-6" src={img_bilice} alt="Bilice" />
        </div>
        <div className="bg-slate-700 text-white rounded-lg overflow-hidden sha-
dow-lg transition-transform transform hover:scale-105">
          <img
            src={fieldData.image}
            alt={fieldData.name}
            className="w-full h-64 object-cover"
          />
          <div className="p-6">
            <h2 className="text-xl font-semibold mb-3">{fieldData.name}</h2>
            <p className="text-sm mb-4">{fieldData.description}</p>
            <p className="text-lg font-bold mb-4">
              Cijena termina:{" "}
```

Kod 7. Prikaz koda za podteren Bilice

Na prikazanom primjeru koda 7, definira se komponenta pod nazivom *Bilice* koja prikazuje detalje o terenu „*Nogometni Tereni Bilice*“ koji se sastoji od jednog podterena. Na početku koda definirana je varijabla `fieldData` koja sadrži osnovne informacije o terenu, uključujući naslov („*Nogometni Tereni Bilice*“), ime terena („*Teren br. 1*“), cijenu („*24€*“), opis („*Teren pogodan za igru 4x4*“) i sliku terena. Stranica je oblikovana pomoću programskog okvira Tailwind CSS utility klasa kako bi se osigurao moderan i responzivan dizajn. U povratnoj funkciji `return`, cijeli sadržaj prikazuje se unutar div elemenata koji su stilizirani pomoću

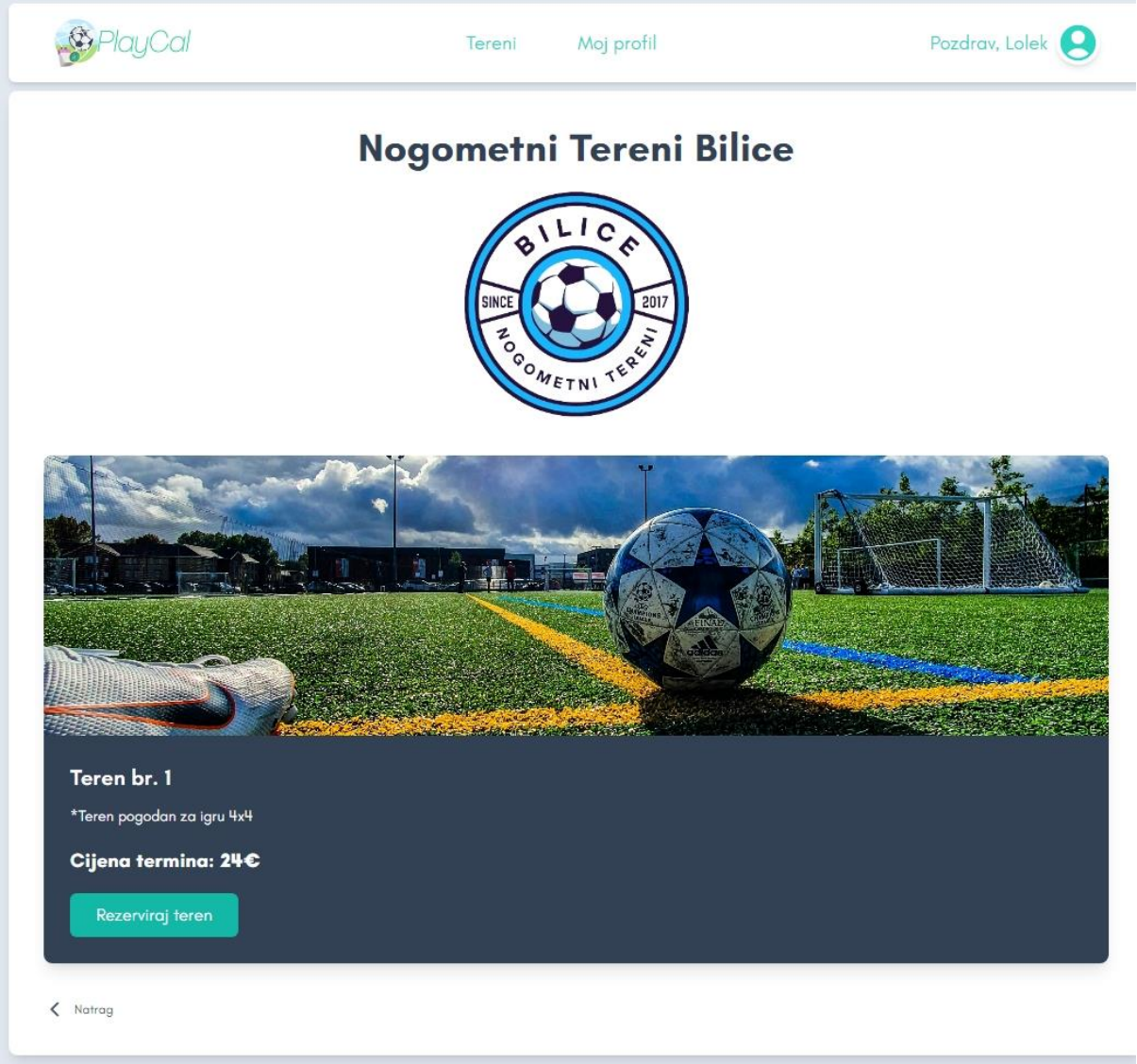
Tailwind klasa, kao što su `bg-slate-200`, `rounded-lg`, `shadow-lg`, što omogućuje pozicioniranje i izgled elemenata na stranici. U samoj strukturi HTML-a, naslov terena prikazan je pomoću `h1` elementa koji koristi Tailwind klasu `text-slate-700` i `text-4xl` za stilizaciju teksta. Ispod naslova nalazi se slika terena koja se učitava pomoću varijable `fieldData.image`. Ova slika koristi Tailwind klasu `w-full` za postavljanje širine slike i `object-cover` za ispravno skaliranje slike unutar zadanih dimenzija. Opis terena prikazuje se unutar *p* elementa, koristeći varijablu `fieldData.description`, te prikazuje osnovne informacije o samom terenu, kao i njegovu cijenu i broj podterena. U ovom primjeru prikazuje se informacija o jednom terenu u sklopu kompleksa „*Bilice*“.

```
<Link to="/reservation" state={{ field: fieldData }}>
              <button className="bg-teal-500 hover:bg-teal-400 text-white py-2
px-6 rounded-md transition-colors">
                Rezerviraj teren
              </button>
            </Link>
          </div>
        </div>
        <Link to="/pitches">
          <button className="flex items-center text-xs
text-slate-600 hover:text-red-500 mt-8">
            <svg
              className="w-5 h-5 mr-2"
              xmlns="http://www.w3.org/2000/svg"
              viewBox="0 0 512 512"
            >
              <path
                fill="none"
                stroke="currentColor"
                strokeLinecap="round"
                strokeLinejoin="round"
                strokeWidth="48"
                d="M328 112L184 256l144 144"
```

Kod 8. Prikaz koda za podteren Bilice br.2

Uz sve to, tu je i gumb za rezervaciju terena, koji omogućuje korisniku da klikom na njega pređe na stranicu za rezervaciju. Gumb je stiliziran pomoću Tailwind CSS klasa `bg-teal-500`, `hover`, `text-white`, a koristi `Link` komponentu iz React Router biblioteke za preusmjeravanje na stranicu za rezervaciju. Na kraju koda nalazi se i gumb „*Natrag*“, koji omogućuje korisniku povratak na popis svih terena. Ovaj gumb koristi *SVG* ikonu kao vizualni element, a implementiran je koristeći `Link` komponentu iz React Router biblioteke s

pripadajućim stilizacijama koje su definirane Tailwind CSS klasama. Konačan izgled stranice za podterene Bilice prikazan je na slici 13 koja se nalazi ispod.



Slika 13. Prikaz stranice podterena Bilice

Sljedeći je drugi slučaj, u kojem kompleks terena ima više podterena na raspolaganju. Ovdje ćemo analizirati kako korisnici mogu birati između različitih podterena unutar istog kompleksa, te kako je taj koncept implementiran unutar aplikacije. Ovakva struktura omogućuje korisnicima veću fleksibilnost u izboru termina, ovisno o dostupnosti i karakteristikama pojedinih podterena.

```
const Gool = () => {
  const fieldData = {
    field_1: {
      heading: "Nogometni tereni Gool Šubićevac",
      name: "Teren br.1",
      price: "24€",
      description: "*Teren pogodan za igru 4x4",
      image: pitch,
    },
    field_2: {
      heading: "Nogometni tereni Gool Šubićevac",
      name: "Teren br.2",
      price: "24€",
      description: "*Teren pogodan za igru 4x4",
      image: pitch_2,
    },
    field_3: {
      heading: "Nogometni tereni Gool Šubićevac",
      name: "Teren br.3",
      price: "18€",
      description: "*Teren pogodan za igru 3x3",
      image: pitch_3,
    },
  };
```

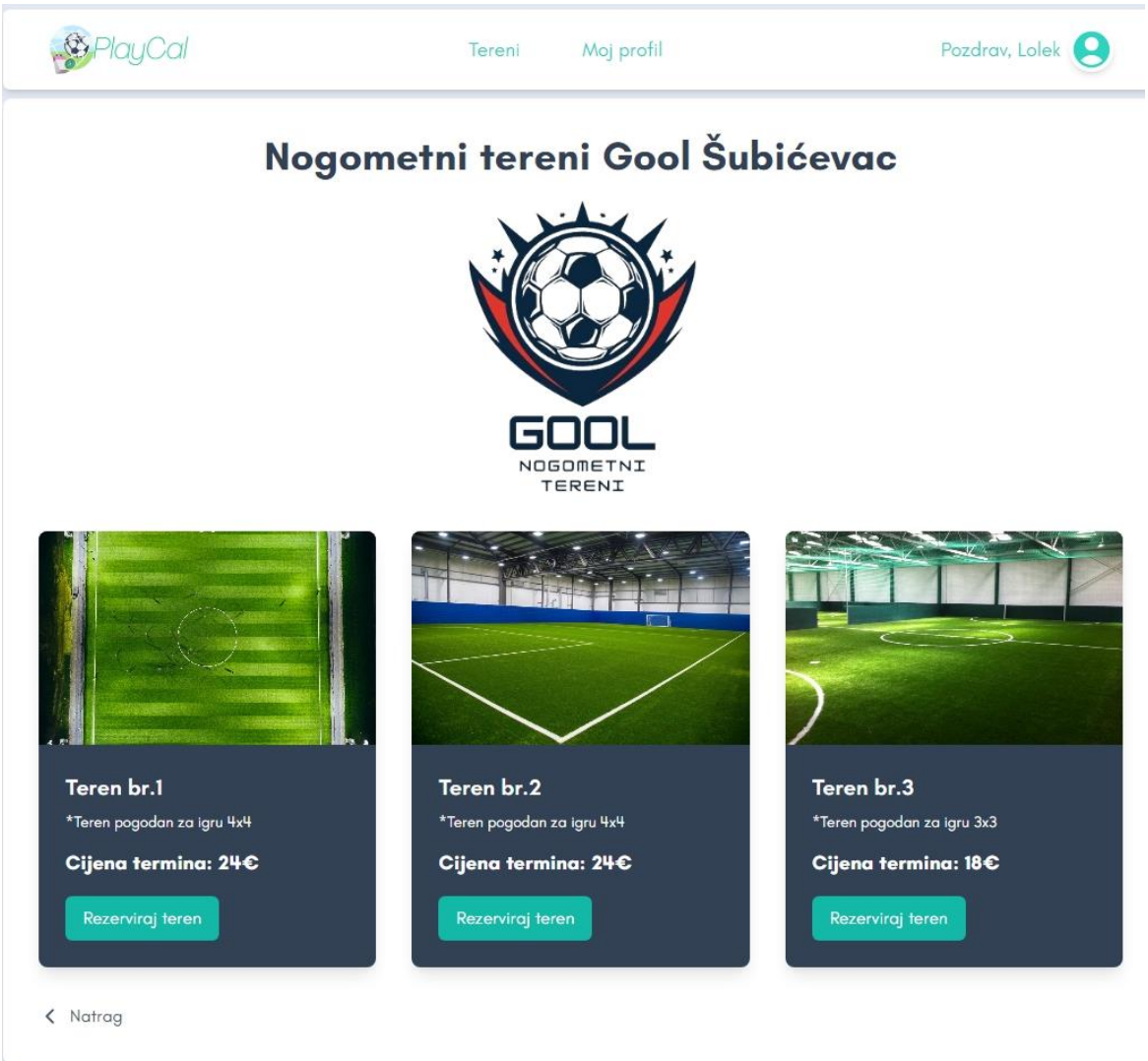
Kod 9. Prikaz koda stranice podterena Gool

Na kodu 9 prikazuje se definicija kompleksa pod imenom „*Nogometni tereni 'Gool' Šubićevac*", koji se sastoji od tri podterena. Svaki podteren je definiran unutar objekta `fieldData` s ključevima `field_1`, `field_2` i `field_3`, koji sadrže informacije o nazivu terena, cijeni, opisu i slici terena. Ovakva struktura omogućuje korisniku pregled i odabir između više podterena unutar istog kompleksa.

```
return (
    <div className="font-teachers bg-slate-200 min-h-screen">
      <Res_Heading />
      <div className="mx-auto h-full p-8 bg-white rounded-lg
shadow-lg">
        <div className="text-center mb-8">
          <h1 className="text-slate-700 font-bold text-4xl mb-4">
            Nogometni tereni Gool Šubićevac
          </h1>
          <img className="w-52 mx-auto" src={img_gool}
alt="Gool Logo" />
        </div>
        <div className="grid lg:grid-cols-3 md:grid-cols-2 sm:grid-cols-1
gap-8">
          {Object.keys(fieldData).map((key) => {
            const field = fieldData[key];
            return (
              <div
                key={key}
                className="bg-slate-700 text-white rounded-lg
overflow-hidden shadow-lg transition-transform transform hover:scale-105"
              >
                <img
                  src={field.image}
                  alt={field.name}
                  className="w-full h-48 object-cover"
                />
                <div className="p-6">
                  <h2 className="text-xl font-semibold mb-2">
{field.name}</h2>
                  <p className="text-sm mb-3">
{field.description}</p>
                  <p className="text-lg font-bold mb-4">
                    Cijena termina: <span>{field.price}</span>
                  </p>
                  <Link to="/reservation" state={{ field }}>
                    <button className="bg-teal-500 hover:bg-teal-400 text-white
py-2 px-4 rounded-md transition-colors">
                      Rezerviraj teren
                    </button>
                  </Link>
                </div>
              </div>
            );
```

Kod 10. Prikaz koda stranice podterena Gool br. 2

Na gore prikazanom kodu 10 se renderira stranica za prikaz podterena unutar kompleksa terena „*Nogometni tereni 'Gool' Šubićevac*“. Ovdje je implementiran prikaz podataka o svakom pojedinačnom terenu, uključujući naziv terena, cijenu po terminu, opis terena i sliku. Za svaki teren unutar kompleksa koristi se objekt `fieldData` koji sadrži informacije o tri različita podterena. Unutar React-ove funkcionalne komponente *Gool*, objekt `fieldData` koristi se za dinamičko generiranje podataka za svaki od terena kroz metodu `Object.keys().map()`. Ova metoda iterira kroz sve ključeve objekta `fieldData` te za svaki ključ dohvaća odgovarajuće podatke o podterenu. Svaki podteren ima definiranu sliku, cijenu, naziv i opis, koji se prikazuju pomoću Tailwind CSS klasa za stilizaciju. Također, na kraju svake kartice nalazi se gumb "*Rezerviraj teren*" koji korisnika preusmjerava na stranicu za rezervaciju, prenoseći podatke o odabranom terenu pomoću React-ovog `Link` komponenta i state parametra. Na kraju koda nalazi se i gumb "*Natrag*", koji omogućuje korisniku povratak na popis svih terena. Konačan izgled stranice prikazan je u slici 14 koja se nalazi ispod.



Slika 14. Prikaz izgleda stranice podterena Gool

3.3.4. Prikaz stranice rezervacije

U ovom poglavlju fokus je na funkcionalnost rezervacije terena unutar aplikacije. Ova funkcionalnost omogućava korisnicima da, nakon pregleda dostupnih terena, odaberu željeni termin i rezerviraju ga. Proces rezervacije obuhvaća dohvaćanje korisničkih podataka, provjeru dostupnosti termina te prikaz relevantnih informacija o terenu i korisniku. Detaljno će se opisati kako aplikacija upravlja rezervacijom terena, koristeći React-ove hookove za upravljanje stanjem i Firebase za dohvaćanje i spremanje podataka. Funkcija `Reservation` koristi React-ove hookove za rukovanje stanjem i efektima. Korištenjem `useLocation` hooka dohvaćaju se podaci o terenu koji su poslani s prethodne stranice. Ako nisu dostupni podaci o terenu, prikazuju se zadane vrijednosti (npr. „*Nepoznati tereni*“). Termini su prikazani pomoću niza `availableSlots`, koji sadrži unaprijed definirane vremenske okvire za rezervaciju. Korisnik može odabrati željeni termin, a komponente `useState` služe za upravljanje odabranim terminom i rezerviranim terminima. Metoda `fetchUserData` koristi `useEffect` hook za dohvaćanje korisničkih podataka iz Firebase Firestore baze podataka kada je korisnik prijavljen. Funkcija se poziva kada postoji aktivni korisnik, što se provjerava pomoću varijable `user`, koja koristi `useAuth` hook kako bi pristupila trenutnim podacima o prijavljenom korisniku. Opisano se može vidjeti u primjeru koda 11 koji se nalazi ispod.

```
const Reservation = () => {
  const location = useLocation();
  const navigate = useNavigate();
  const [date, setDate] = useState(null);
  const [selectedSlot, setSelectedSlot] = useState(null);
  const [userData, setUserData] = useState(null);
  const [reservedSlots, setReservedSlots] = useState([]);
  const field = location.state?.field || {
    heading: "Nepoznati tereni",
    name: "Nepoznati teren",
    price: "0€",
    description: "Opis nije dostupan",
    image: {},
  };
  const availableSlots = [
    "17:00 - 18:00",
    "18:00 - 19:00",
    "19:00 - 20:00",
    "20:00 - 21:00",
    "21:00 - 22:00",
  ];
  const { user } = useAuth();

  // Dohvaćanje korisničkih podataka
  useEffect(() => {
    const fetchUserData = async () => {
      if (user) {
        try {
          const userDoc = doc(db, "users", user.uid);
          const docSnap = await getDoc(userDoc);
          if (docSnap.exists()) {
            setUserData(docSnap.data());
          } else {
            console.log("Nema korisničkih podataka!");
          }
        } catch (error) {
          console.error("Greška pri dohvaćanju podataka:", error);
        }
      }
    };
    fetchUserData();
```

Kod 11. Prikaz koda stranice za rezervaciju

```
fetchUserData();
  }, [user]);
  // Dohvaćanje rezerviranih termina za odabrani datum
  useEffect(() => {
    const fetchReservedSlots = async () => {
      if (date) {
        const formattedDate = formatDate(date[0]);
        const q = query(
          collection(db, "reservations"),
          where("date", "==", formattedDate),
          where("field.heading", "==", field.heading)
        );
        try {
          const querySnapshot = await getDocs(q);
          const reserved = querySnapshot.docs.map((doc) => doc.data().slot);
          setReservedSlots(reserved);
        } catch (error) {
          console.error("Greška pri dohvaćanju rezerviranih termina:", error);
        }
      }
    };
    fetchReservedSlots();
  }, [date]);
  // Formatiranje datuma u obliku "Ponedjeljak, 2024-08-19"
  const formatDateWithDay = (date) => {
    const options = {
      weekday: "long",
      year: "numeric",
      month: "numeric",
      day: "numeric",
    };
    return new Intl.DateTimeFormat("hr-HR", options).format(date);
  };
```

Kod 12. Prikaz koda stranice za rezervaciju br. 2

Kod 12 prikazuje implementaciju funkcionalnosti rezervacije unutar aplikacije, posebno fokusirajući se na dohvaćanje već rezerviranih termina za odabrani datum te prikaz informacija o datumu i vremenu. Na početku, koristi se `useEffect` hook za dohvaćanje rezerviranih termina iz Firebase Firestore baze podataka, koristeći funkciju `formatDate` kako bi se osiguralo da se podaci o terminima pretražuju prema formatiranom datumu. Korištenjem `where` uvjeta, aplikacija filtrira podatke u bazi prema odabranom terenu i datumu. U slučaju uspješnog dohvaćanja podataka, popunjava se stanje s već rezerviranim terminima pomoću `setReservedSlots`. U suprotnom, prikazuje se greška u konzoli. Kod također formatira datum koristeći `Intl.DateTimeFormat` kako bi se osiguralo da korisnik dobije pregled

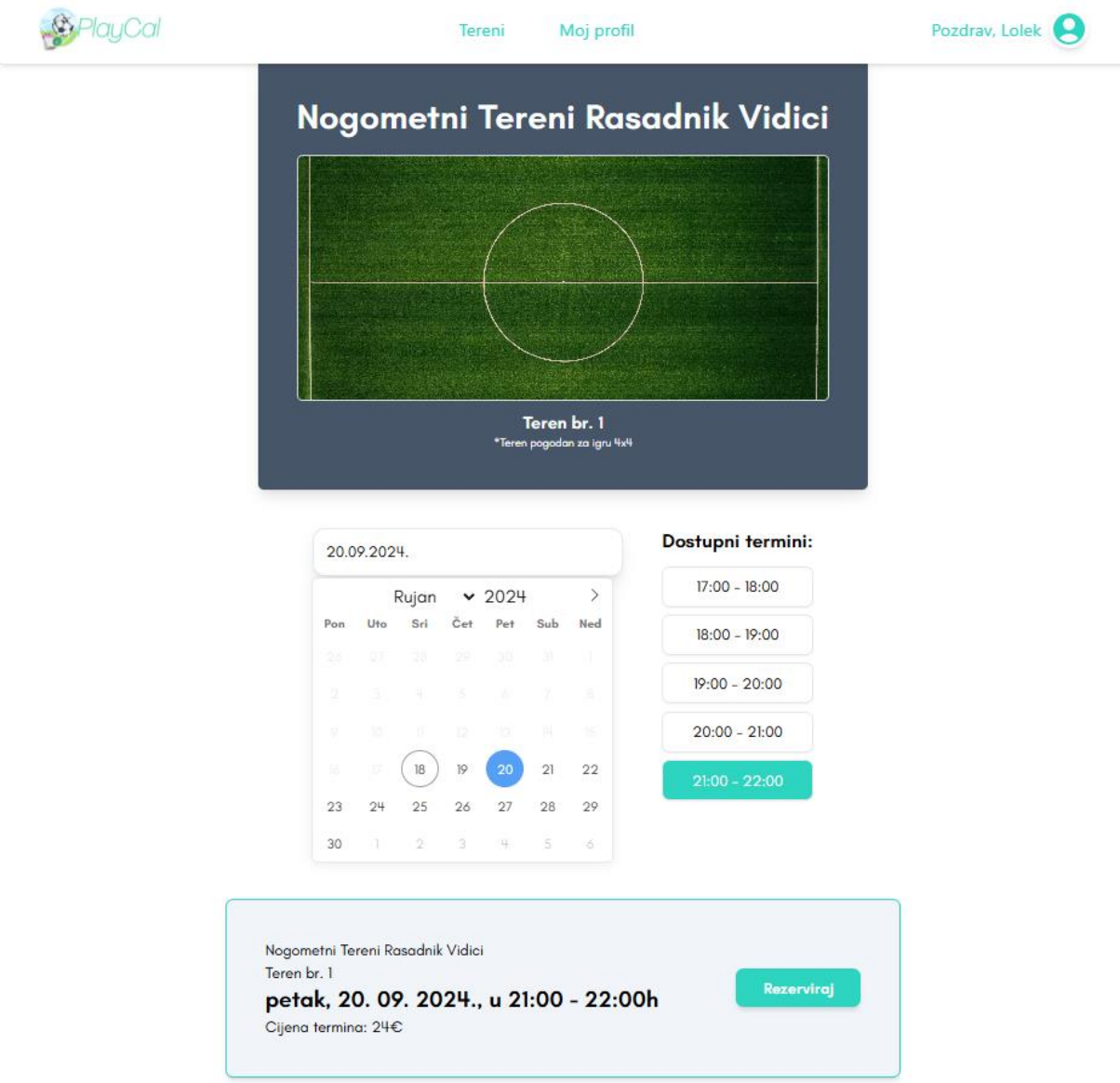
datuma u čitljivom formatu, npr. „*Ponedjeljak, 27.08.2024.*“, što olakšava korisnicima odabir željenog termina.

```
const handleReservation = async () => {
    if (!date || !selectedSlot) {
      alert("Molimo odaberite datum i termin.");
      return;
    }
    if (!userData) {
      alert("Niste prijavljeni ili nema korisničkih podataka.");
      return;
    }
    try {
      await setDoc(
        doc(
          db,
          "reservations",
          `${user.uid}_${date[0].toISOString()}_${selectedSlot}`
        ),
        {
          userId: user.uid,
          userName: userData.lastName,
          field: field,
          date: formatDate(date[0]), // Spremamo formatirani datum bez dana u
tjednu
          slot: selectedSlot,
        }
      );
      alert("Rezervacija je uspješno spremljena!");
      navigate("/pitches"); // Preusmjeravanje na pitches stranicu
    } catch (error) {
      console.error("Greška pri spremanju rezervacije:", error);
    }
  };
  // Filtriranje dostupnih termina
  const getSlotClass = (slot) => {
    if (reservedSlots.includes(slot)) {
      return "bg-red-400 text-white cursor-not-allowed"; // Stil za zauzete
termine
    }
    return selectedSlot === slot
```

Kod 13. Prikaz koda stranice za rezervaciju br. 3

Metoda `handleReservation` unutar prikazanog koda zadužena je za rukovanje rezervacijom termina. Ova funkcija koristi asinkrono rukovanje (`async` i `await`) kako bi se podaci o rezervaciji pohranili u Firebase Firestore bazu podataka. Na početku, funkcija provjerava je li korisnik odabrao datum (`date`) i termin (`selectedSlot`). Ako korisnik nije

odabrao ove ključne informacije, prikazuje se obavijest upozorenja koja traži od korisnika da odabere datum i termin, te funkcija prekida daljnje izvršavanje (`return`). Sljedeća provjera je jesu li korisnički podaci (`userData`) dostupni. Ako su svi uvjeti zadovoljeni, funkcija koristi `setDoc` za spremanje rezervacije u kolekciju „*reservations*“ unutar Firestore baze. Dokument se sprema s ključem koji uključuje korisnički ID (`user.uid`), odabrani datum (`date[0]`) i termin (`selectedSlot`). Podaci koji se spremaju uključuju korisnički ID, prezime korisnika (`userData.lastName`), informacije o terenu (`field`), formatirani datum te odabrani termin. Nakon uspješno spremljene rezervacije, korisnik dobiva obavijest o uspješnoj rezervaciji putem `alert` funkcije, a zatim ga funkcija preusmjerava na stranicu s popisom terena. Ako tijekom procesa dođe do greške, greška se hvata i ispisuje u konzoli kako bi se dijagnosticirali eventualni problemi. Ovaj dio koda osigurava da se podaci o rezervaciji točno pohranjuju u bazu i da korisnik bude pravilno obaviješten o uspjehu ili neuspjehu rezervacije. Ostatak koda posvećen je stiliziranju stranice koristeći programski okvir Tailwind CSS kako bi se osigurao responzivan i moderan dizajn. Korištenjem Tailwind CSS-a klasa definira se izgled elemenata, poput gumba, polja za unos i rasporeda elemenata na stranici. Konačan izgled stranice rezervacije prikazan je na slici 15 koja se nalazi ispod.



Slika 15. Prikaz izgleda stranice za rezervaciju

3.3.5. Prikaz stranice profila i upravljanja rezervacijama

U ovom poglavlju opisat će se rukovanje stranicom profila i upravljanje rezervacijama unutar web aplikacije. Kada se korisnik uspješno prijavi putem stranice za prijavu, automatski se preusmjerava na stranicu *pitches*, gdje se prikazuje zaglavlje specifično za ovu funkcionalnost. Ovo zaglavlje sadrži navigacijske opcije za odabir terena, kao i pregled informacija o profilu i rezervacijama korisnika. Korištenje zaglavlja omogućava lakši pristup tim funkcijama, a njegov izgled definiran je putem koda koji uključuje mogućnost odjave korisnika, pregled terena i upravljanje korisničkim profilom.

```
const Res_Heading = () => {
const [menuOpen, setMenuOpen] = useState(false); // State za otvaranje/
skrivanje izbornika
  const [firstName, setFirstName] = useState(""); // State za pohranu
korisničkog imena
  const { user } = useAuth(); // Dohvat trenutnog korisnika iz AuthContext
  const navigate = useNavigate(); // Inicijalizacija useNavigate za
preusmjeravanje
  useEffect(() => {
    // Funkcija za dohvaćanje korisničkog imena iz Firestore-a
    const fetchFirstName = async () => {
      if (user) {
        try {
          const userDocRef = doc(db, "users", user.uid);
          const userDoc = await getDoc(userDocRef);

          if (userDoc.exists()) {
            const userData = userDoc.data();
            setFirstName(userData.firstName);
          } else {
            console.log("Nema podataka za ovog korisnika.");
          }
        } catch (error) {
          console.error("Greška pri dohvaćanju korisničkog imena:", error);
        }
      }
    };
    fetchFirstName();
  }, [user]);
```

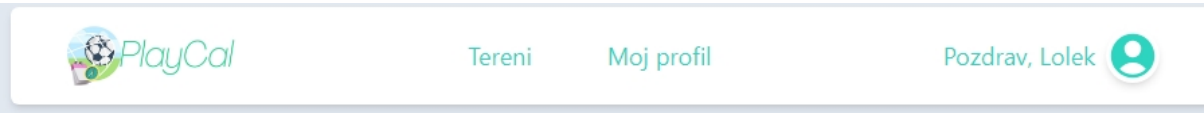
Kod 14. Prikaz koda zaglavlja profila

U kodu 14 se koristi `useEffect` kako bi dohvaćao ime korisnika iz Firestore-a pri učitavanju komponente. Ako je korisnik prijavljen, dohvaća se njegov dokument pomoću `user.uid`. Ako dokument postoji, postavlja se ime korisnika u `state` (`setFirstName`), inače se ispisuje poruka da nema podataka. U slučaju greške, ispisuje se poruka o grešci.

```
const toggleProfileMenu = () => {
    setMenuOpen(!menuOpen);
  };

  const handleSignOut = async () => {
    try {
      await signOut(auth);
      console.log("Korisnik je odjavljen");
      navigate("/login"); // Preusmjeravanje na stranicu za prijavu
    } catch (error) {
      console.error("Greška pri odjavi:", error);
    }
  };
```

Kod 15. Prikaz koda zaglavlja profila br. 2

Sljedeći dio, kod 15, sadrži dvije funkcije. Prva, `toggleProfileMenu`, omogućava otvaranje i zatvaranje izbornika za korisnički profil mijenjanjem stanja (`setMenuOpen`). Druga funkcija, `handleSignOut`, koristi se za odjavu korisnika iz aplikacije. Kada se korisnik odjavi, pomoću `signOut` funkcije iz Firebase Authentication-a, preusmjerava se na stranicu za prijavu koristeći `navigate("/login")`. U slučaju greške, ispisuje se poruka o grešci u konzoli. Ostatak koda posvećen je za stilizaciju Tailwind CSS-om te konačni izgled tog zaglavlja je prikazan u slici 16 koja se nalazi ispod.



Slika 16. Prikaz izgleda zaglavlja profila

Link „*Tereni*“ u zaglavlju već je prethodno obrađen, gdje se korisnicima omogućuje pregled i odabir dostupnih terena. Sada će biti obrađen link „*Moj profil*“, koji korisnicima pruža pristup njihovim osobnim podacima i upravljanje rezervacijama unutar aplikacije.

```
import {
  doc,
  getDoc,
  collection,
  query,
  where,
  getDocs,
  deleteDoc,
} from "firebase/firestore";

const Player = () => {
  const [activeTab, setActiveTab] = useState("profile");
  const [userData, setUserData] = useState(null);
  const [reservations, setReservations] = useState([]);
  const [showModal, setShowModal] = useState(false);
  const [reservationToDelete, setReservationToDelete] = useState(null);

  useEffect(() => {
    const fetchUserDataAndReservations = async () => {
      try {
        const user = auth.currentUser;

        if (user) {
          const userDocRef = doc(db, "users", user.uid);
          const userDoc = await getDoc(userDocRef);

          if (userDoc.exists()) {
            setUserData(userDoc.data());
          }
```

Kod 16. Prikaz koda stranice "Moj profil"

Prikazani kod 16 implementira stranicu profila korisnika i omogućava pregled i upravljanje rezervacijama unutar web aplikacije. Komponenta Player koristi `useState` i `useEffect` hookove kako bi se pratilo trenutno aktivni tab, dohvatili korisnički podaci i prikazale korisnikove rezervacije. Na početku, definiraju se state varijable:

- `activeTab` prati koji je tab trenutno aktivan („`profile`“ ili „`reservations`“),
- *`userData`* čuva podatke o korisniku,
- `reservations` pohranjuje korisnikove rezervacije,
- `showModal` kontrolira prikaz modala za potvrdu brisanja rezervacije,
- `reservationToDelete` čuva ID rezervacije koja će se izbrisati.

Unutar `useEffect` hooka, definirana je asinhrona funkcija `fetchUserDataAndReservations` koja dohvaća korisničke podatke i rezervacije iz Firestore baze podataka. Ako je korisnik prijavljen, prvo se dohvaćaju osobni podaci iz

kolekcije „*users*“, a zatim se iz kolekcije „*reservations*“ dohvaćaju sve rezervacije povezane s korisnikom na temelju njihovog `userId`.

```
const reservationsQuery = query(
            collection(db, "reservations"),
            where("userId", "==", user.uid)
          );
          const reservationsSnapshot = await getDocs(reservationsQuery);
          const reservationsData = reservationsSnapshot.docs.map((doc) => ({
            id: doc.id,
            ...doc.data(),
          }));

          setReservations(reservationsData);
        }
      } catch (error) {
        console.error("Error fetching user data or reservations:", error);
      }
    };

    fetchUserDataAndReservations();
  }, []);
```

Kod 17. Prikaz koda stranice "Moj profil" br. 2

Dohvaćene rezervacije se spremaju u `state`. Funkcija `handleDeleteReservation` koristi se za brisanje određene rezervacije iz baze podataka. Nakon što se rezervacija izbriše, ona se uklanja iz lokalnog `statea` rezervacija kako bi se korisniku prikazale samo preostale rezervacije. Ako korisnik želi izbrisati rezervaciju, otvara se modal s potvrdom brisanja pomoću funkcije `openDeleteModal`, a modal se zatvara bez brisanja pomoću funkcije `closeModal`. Glavna funkcija `renderContent` ovisi o trenutno aktivnom tabu i prikazuje ili komponentu `ProfileInfo`, koja prikazuje korisničke osobne podatke, ili komponentu `MyReservations`, koja prikazuje popis korisnikovih rezervacija s mogućnošću brisanja.

```
const handleDeleteReservation = async () => {
    try {
      await deleteDoc(doc(db, "reservations", reservationToDelete));

      setReservations(
        reservations.filter((res) => res.id !== reservationToDelete)
      );
      setShowModal(false); // Zatvori modal nakon brisanja
    } catch (error) {
      console.error("Greška pri brisanju rezervacije:", error);
    }
  };

  const openDeleteModal = (reservationId) => {
    setReservationToDelete(reservationId);
    setShowModal(true); // Otvori modal
  };

  const closeModal = () => {
    setShowModal(false); // Zatvori modal bez brisanja
  };

  const renderContent = () => {
    switch (activeTab) {
      case "profile":
        return <ProfileInfo userData={userData} />;
      case "reservations":
        return (
          <MyReservations
            reservations={reservations}
            openDeleteModal={openDeleteModal}
          />
        );
      default:
        return <ProfileInfo userData={userData} />;
    }
  };
```

Kod 18. Prikaz koda stranice "Moj profil" br. 3

Komponenta `ProfileInfo` prikazuje korisničke informacije poput *imena*, *prezimena*, *emaila*, *spola* i *datuma rođenja*. Datum rođenja formatira se u čitljiv oblik koristeći `toLocaleDateString`. Komponenta `MyReservations` prikazuje korisnikove rezervacije te omogućuje brisanje rezervacija klikom na odgovarajući gumb. Na dnu se nalazi modal za potvrdu brisanja rezervacije, koji se prikazuje kada korisnik klikne na gumb za brisanje rezervacije. Osim toga, kod uključuje funkciju `formatDate` koja formatira datume

rezervacija u čitljiv oblik koristeći lokalne postavke za hrvatski jezik. Konačan izgled stranice prikazan je na slici 17 i slici 18 koje se nalaze ispod.

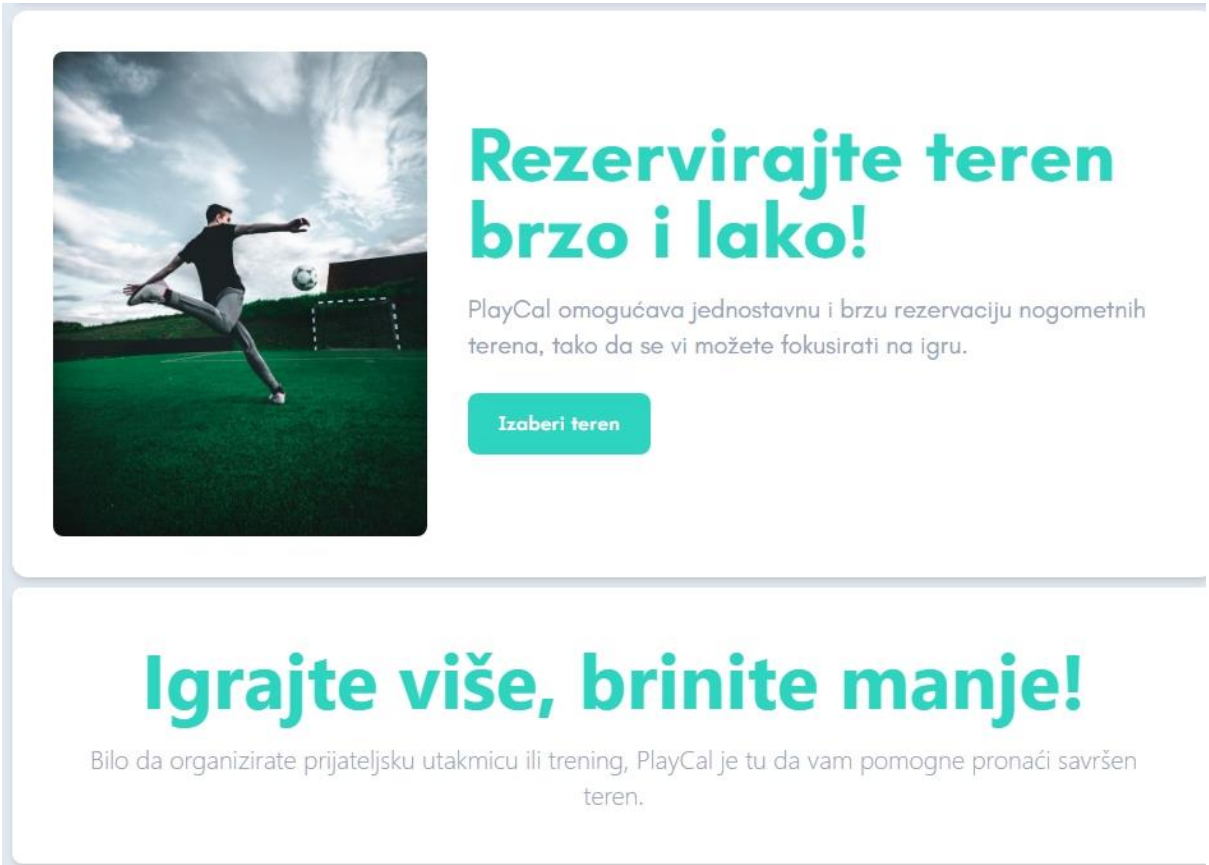
Slika 17. Prikaz osobnih podataka unutar stranice "Moj profil"

Slika 18. Prikaz rezervacija unutar stranice "Moj profil"

3.3.6. Prikaz naslovnice, stranice „O nama“ i „Kontakt“

Naslovna stranica web aplikacije sastavljena je od više međusobno povezanih komponenti koje zajedno oblikuju korisničko sučelje. Komponenta *Home* služi kao okvir u kojem su smještene sve ostale komponente, čineći je modularnom i fleksibilnom za proširenje. Svaka komponenta ima specifičnu ulogu, od prikaza ključnih informacija do poziva na akciju, što korisnicima omogućuje intuitivno kretanje kroz aplikaciju. *Home* komponenta je organizirana tako da se svaki njezin dio logički nadovezuje jedan na drugi.

Na početku, *HomeBody_1* pruža uvodni tekst i opciju brzog odabira terena, privlačeći korisnike s jasnom i izravnom porukom. Zatim, *HomeBody_2* ističe glavne prednosti aplikacije, dodatno motivirajući korisnike kroz jednostavne i korisnički usmjerene poruke. Središnji dio stranice čini *HomeBody_3*, koji koristi vizualne elemente, poput slidera s nogometnim terenima, i omogućuje korisnicima da se interaktivno povežu s aplikacijom. Na kraju, *HomeBody_6* predstavlja poziv na akciju, pozivajući korisnike da odmah započnu s korištenjem aplikacije. Cijeli dizajn naslovne stranice je responzivan i prilagođen svim vrstama uređaja zahvaljujući upotrebi Tailwind CSS-a, čime se osigurava jednostavna navigacija i preglednost na različitim veličinama ekrana. Konačan izgled naslovne stranice je prikazan u slici 19 i slici 20 koje se nalaze ispod.



Slika 19. Prikaz izgleda naslovne stranice

Slika 20. Prikaz izgleda naslovne stranice br. 2

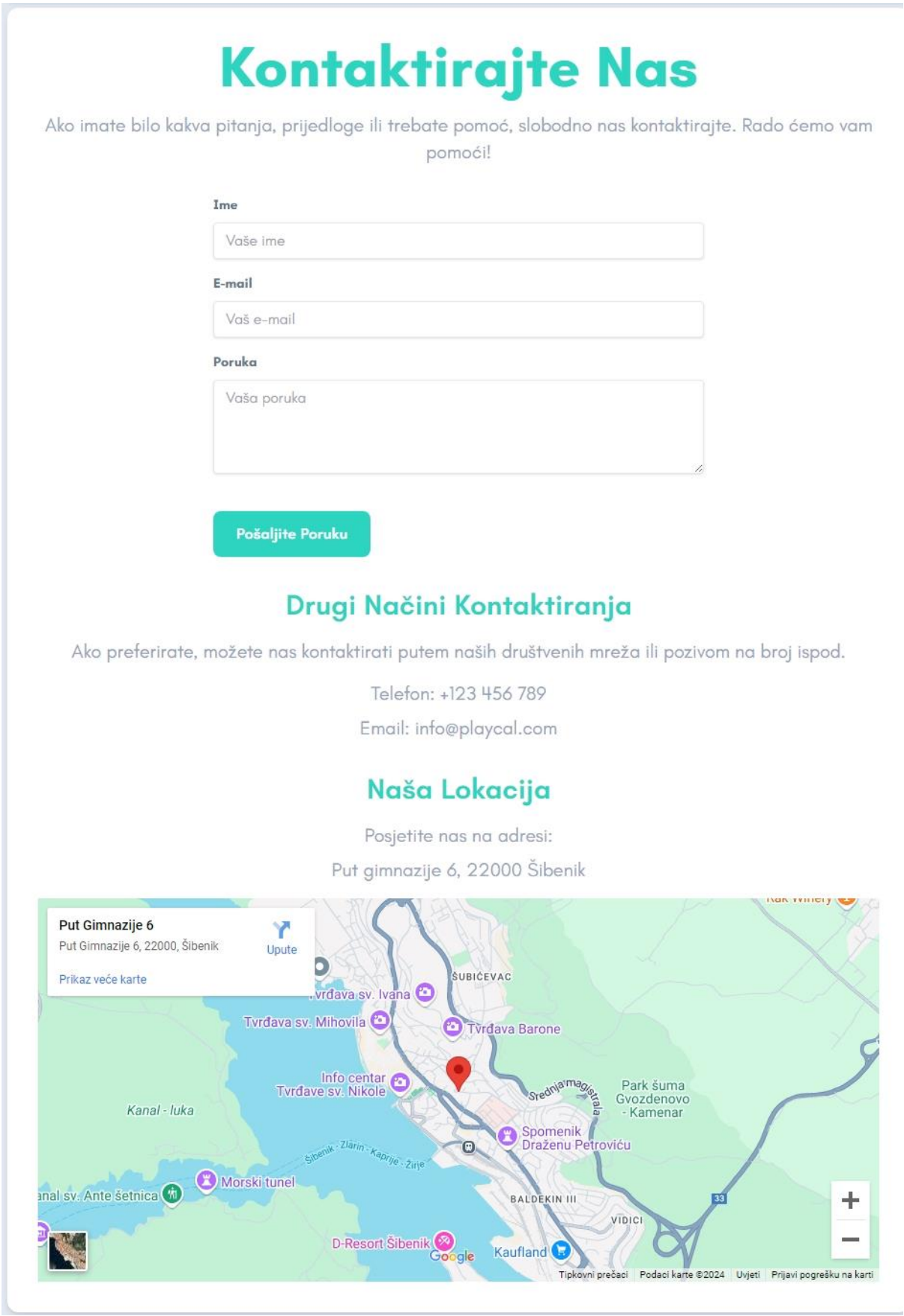
Što se tiče stranice "*O Nama*" pružene su ključne informacije o svrsi, misiji i viziji web aplikacije te o timu i postignućima. Stranica započinje logotipom aplikacije, koji se nalazi centralno i privlači pažnju korisnika. Ispod logotipa nalazi se veliki naslov „*O Nama*“, koji jasno identificira temu stranice. U prvom odlomku naglašava se posvećenost aplikacije pružanju najboljeg korisničkog iskustva u rezervaciji nogometnih terena, s posebnim naglaskom na jednostavnost organizacije igara i turnira. Dalje se opisuje kako je platforma dizajnirana da bude intuitivna i prilagođena korisnicima, omogućujući brz i jednostavan pristup rezervaciji terena. Sljedeći dio stranice fokusira se na „*Našu Misiju i Viziju*“. Ovdje se iznosi želja da aplikacija pruži najefikasniju i najpristupačniju uslugu rezervacije, istovremeno naglašavajući ulogu sporta u povezivanju ljudi. Vizija je postati lider u industriji kroz inovacije

i izvrsno korisničko iskustvo. Sekcija „*Naš Tim*“ posvećena je predstavljanju tima koji stoji iza aplikacije, gdje se ističe kako članovi tima dolaze iz različitih sektora, uključujući tehnologiju i korisničku podršku, a svojim jedinstvenim vještinama doprinose razvoju aplikacije. U zadnjoj sekciji, pod naslovom „*Naša Postignuća“*, opisana su dosadašnja postignuća aplikacije, uključujući veliki broj zadovoljnih korisnika i uspješno organizirane sportske događaje. Poziv korisnicima da stupe u kontakt nalazi se na dnu stranice, uz gumb „*Kontaktirajte nas*“, koji korisnike vodi do kontakt forme. Cijela stranica je dizajnirana s jednostavnim i čistim izgledom, koristeći Tailwind CSS za stilizaciju, što osigurava dosljedan vizualni identitet s ostatkom aplikacije. Konačni izgled stranice je prikazan u slici 21 koja se nalazi ispod.

Slika 21. Prikaz izgleda stranice "About"

Na stranici „*Kontakt*“ nalazi se obrazac koji omogućava korisnicima slanje poruka putem e-maila te pruža dodatne načine kontaktiranja. Stranica započinje velikim naslovom u središtu, jasno pozivajući korisnike da stupe u kontakt. Ispod naslova se nalaze upute korisnicima, koje naglašavaju kako je aplikacija dostupna za sva pitanja, prijedloge ili pomoć. Glavni dio stranice je obrazac za kontakt koji sadrži polja za unos imena, e-mail adrese i poruke, što omogućuje jednostavno i direktno slanje upita ili povratnih informacija. Obrazac je stiliziran pomoću Tailwind CSS-a i uključuje elegantna polja za unos te gumb "*Pošaljite Poruku*", koji

korisnicima omogućava slanje poruke. U drugom dijelu stranice, ispod obrasca, korisnicima su ponuđeni dodatni načini kontaktiranja. Ovdje se nalaze informacije o broju telefona i e-mail adresi koje su dostupne za direktni kontakt. Na kraju, stranica nudi pregled lokacije pomoću *Google Maps* integracije, koja omogućava korisnicima da vide točnu adresu u Šibeniku i eventualno dobivaju upute za posjet. Cijela stranica prati jednostavan i pregledan dizajn, prilagođen korisnicima, kako bi komunikacija bila što jednostavnija. Konačni izgled stranice je prikazan u slici 22 koja se nalazi ispod.



Slika 22. Prikaz izgleda stranice kontakta

3.3.7. Prikaz zaglavlja i podnožja stranica te ScrollToTopButton-a

U ovom poglavlju bit će obrađena struktura i funkcionalnost ključnih elemenata korisničkog sučelja koji su prisutni na skoro svim stranicama aplikacije. Fokusirat će se na opis zaglavlja (engl. *Header*), podnožja (engl. *Footer*) te gumba za povratak na vrh stranice (*ScrollToTopButton*). Ovi elementi imaju važnu ulogu u navigaciji unutar aplikacije, omogućavajući korisnicima lak pristup glavnim funkcijama i olakšavajući korisničko iskustvo prilikom pregledavanja sadržaja.

```
const Header = () => {
  const [isOpen, setIsOpen] = useState(false);
  const [menuOpen, setMenuOpen] = useState(false); // State za otvaranje/skri-
vanje izbornika
  const [firstName, setFirstName] = useState(""); // State za pohranu korisni-
čkog imena
  const { user } = useAuth(); // Dohvat trenutnog korisnika iz AuthContext

  useEffect(() => {
    const fetchFirstName = async () => {
      if (user) {
        try {
          const userDocRef = doc(db, "users", user.uid);
          const userDoc = await getDoc(userDocRef);

          if (userDoc.exists()) {
            const userData = userDoc.data();
            setFirstName(userData.firstName || "Korisnik");
          } else {
            setFirstName("Korisnik");
          }
        } catch (error) {
          console.error("Greška pri dohvaćanju korisničkog imena:", error);
          setFirstName("Korisnik");
        }
      } else {
        setFirstName("Korisnik");
      }
    };
```

Kod 19. Prikaz koda zaglavlja web aplikacije

U prikazanom kodu 19, `useEffect` funkcija koristi se za dohvaćanje imena korisnika iz Firestore baze podataka čim se komponenta *Header* učita. Prvo, provjerava se je li korisnik prijavljen (`if (user)`), koristeći podatke iz `useAuth()` hooka koji dohvaća trenutačnog korisnika iz Firebase AuthContexta. Ako je korisnik prijavljen, pokušava se dohvatiti dokument iz kolekcije "*users*" u Firestoreu, gdje je `user.uid` jedinstveni identifikator

korisnika. Za ovo se koristi funkcija `doc()` koja kreira referencu na dokument, i `getDoc()` koja dohvaća dokument iz baze. Ako dokument postoji (`userDoc.exists()`), dohvaćaju se podaci iz dokumenta i spremaju u varijablu `userData`. Zatim se koristi `setFirstName` za postavljanje imena korisnika, ako je ono dostupno; inače se postavlja `default` vrijednost "*Korisnik*". U slučaju greške tijekom dohvaćanja podataka, u catch-bloku hvata se greška i ispisuje u konzolu, a ime se postavlja na "*Korisnik*" kao sigurnosna mjera. Ako korisnik nije prijavljen (`else` dio logike), automatski se postavlja `default` vrijednost za ime na "*Korisnik*". Ova logika osigurava da se uvijek prikazuje ime prijavljenog korisnika, ili, ako nešto pođe po zlu ili korisnik nije prijavljen, prikazuje se defaultni naziv "*Korisnik*".

```
fetchFirstName();
  }, [user]);

  const toggleMenu = () => {
    setIsOpen(!isOpen);
  };

  const toggleProfileMenu = () => {
    setMenuOpen(!menuOpen);
  };

  const handleSignOut = async () => {
    try {
      await signOut(auth);
      console.log("Korisnik je odjavljen");
    } catch (error) {
      console.error("Greška pri odjavi:", error);
    }
  };
```

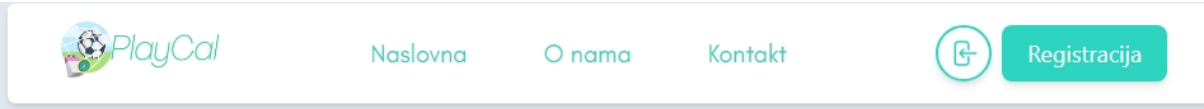
Kod 20. Prikaz koda zaglavlja web aplikacije br. 2

Gore prikazane funkcije, u kodu 20, upravljaju različitim aspektima korisničkog iskustva u aplikaciji. Funkcija `toggleMenu` koristi se za upravljanje stanjem izbornika. Promjenom vrijednosti `isOpen`, kontrolira se je li izbornik vidljiv ili skriven. Svakim pozivom funkcije prebacuje se stanje između otvorenog i zatvorenog izbornika. Slično tome, funkcija `toggleProfileMenu` kontrolira otvaranje i zatvaranje korisničkog izbornika u zaglavlju. Promjenom vrijednosti `menuOpen`, funkcija omogućava prikazivanje ili skrivanje opcija koje su dostupne korisnicima prilikom prijave, poput pregleda profila ili opcije za odjavu. Funkcija `handleSignOut` asinkrono upravlja procesom odjave korisnika. Koristi `signOut()` metodu iz Firebase Authentication biblioteke za odjavu korisnika. Ako je odjava uspješna, u

konzoli se ispisuje poruka „*Korisnik je odjavljen*“. U slučaju greške, hvata se pogreška u catch-bloku, koja se zatim ispisuje u konzoli.

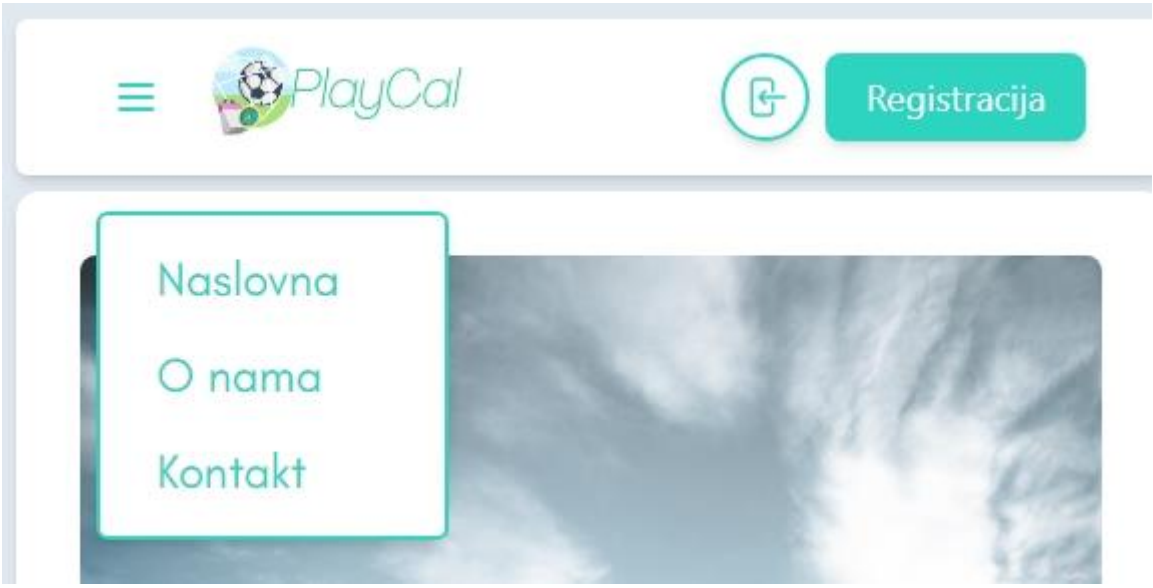
Sljedeći dio koda prikazuje strukturu zaglavlja pomoću Tailwind CSS klasa za stilizaciju i raspored elemenata. Unutar zaglavlja, prvo se nalazi gumb za izbornik koji se prikazuje samo na manjim ekranima. Klikom na taj gumb poziva se funkcija `toggleMenu`, koja otvara ili zatvara mobilni izbornik. *SVG* element unutar gumba prikazuje ikonu s tri horizontalne crte, poznatu kao "*hamburger izbornik*", stiliziranu zelenom bojom. Uz gumb za izbornik, postoji i logo web aplikacije, smješten unutar gumba koji vraća korisnika na početnu stranicu. Slika logotipa ima definiranu širinu koja se prilagođava veličini ekrana.

Desno od logotipa nalazi se navigacijski izbornik koji je sakriven na manjim ekranima, ali se prikazuje na većim zaslonima. Unutar navigacije nalaze se tri poveznice: "*Naslovna*", "*O nama*" i "*Kontakt*". Svaka poveznica vodi na odgovarajuću stranicu, a prilikom prelaska mišem preko njih, boja pozadine mijenja se u zelenu, dok se tekst mijenja u bijeli. Animacija prijelaza postiže se pomoću Tailwind CSS klase koja definira trajanje i kašnjenje efekta. Konačan izgled zaglavlja na velikim ekranima je prikazan na slici 23 koja se nalazi ispod.



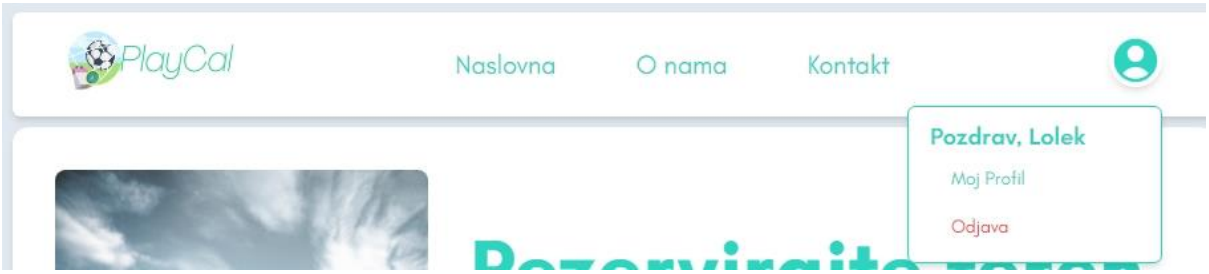
Slika 23. Izgled zaglavlja na velikim ekranima

A izgled zaglavlja na malim ekranima je prikazan na slici 24 koja se nalzi ispod.



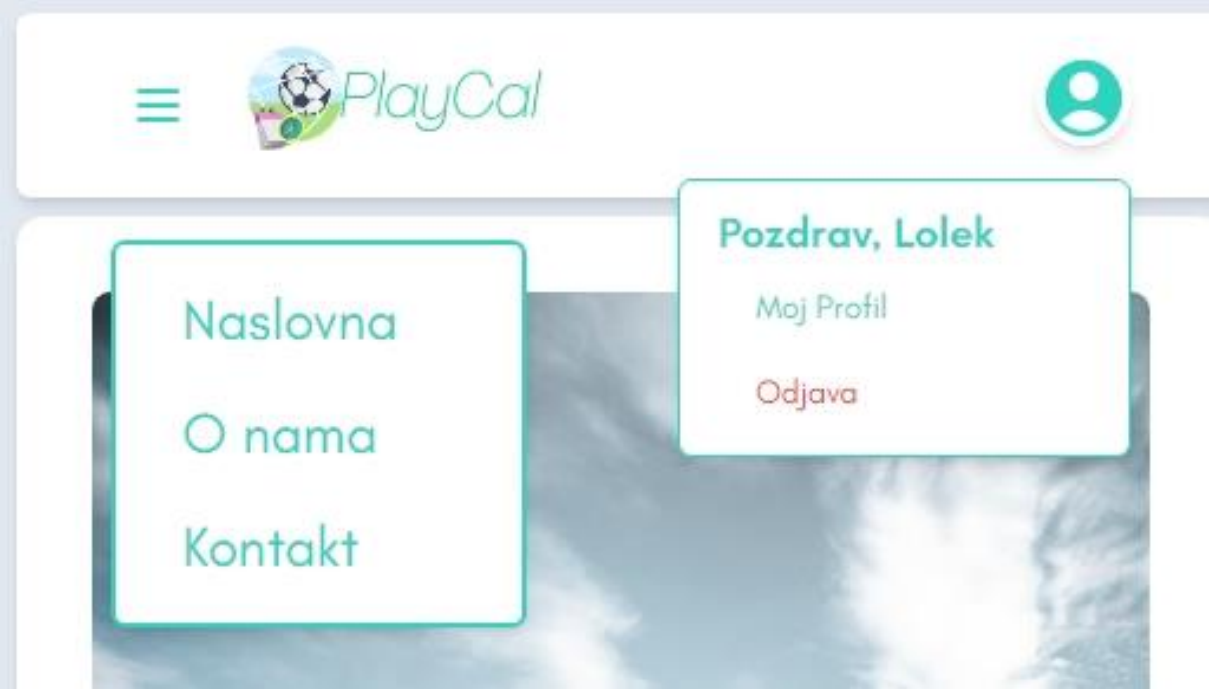
Slika 24. Izgled zaglavlja na malim ekranima

U nastavku koda prikazuju se elementi zaglavlja koji upravljaju korisničkim izbornikom i funkcionalnostima prijave i registracije. U slučaju da je korisnik prijavljen, prikazuje se gumb s ikonom korisničkog profila. Klikom na taj gumb poziva se funkcija `toggleProfileMenu`, koja otvara ili zatvara padajući izbornik. Ikona profila koristi SVG element, a ikona je stilizirana s Tailwind CSS klasama, koje omogućuju promjene u veličini, boji i animaciji. Kada je padajući izbornik otvoren, prikazuje korisničko ime i dvije opcije: „*Moj Profil*“ i „*Odjava*“. Klikom na „*Odjava*“ poziva se funkcija `handleSignOut`, koja odjavljuje korisnika koristeći Firebaseov `signOut` metod. Ako korisnik nije prijavljen, umjesto ikone profila, prikazuju se dva gumba: "*Prijava*" i "*Registracija*". Gumb za prijavu koristi *SVG* ikonu ključa, dok je gumb za registraciju stiliziran s Tailwind CSS klasama i omogućuje animaciju prilikom prelaska mišem. Ovi gumbi preusmjeravaju korisnika na odgovarajuće stranice za prijavu ili registraciju. Na manjim zaslonima, tu je i skriveni mobilni izbornik koji se otvara klikom na gumb izbornika. Kada se izbornik otvori, prikazuju se poveznice do glavnih stranica web aplikacije, a animacija izbornika postiže se promjenom skale i transparentnosti elemenata. Prikaz zaglavlja kod prijavljenog korisnika na velikim ekranima prikazan je na slici 25 koja se nalazi ispod.



Slika 25. Prikaz zaglavlja kod prijavljenog korisnika na velikim ekranima

A prikaz zaglavlja kod prijavljenog korisnika na malim ekranima je prikazan na slici 26 koja se nalazi ispod.



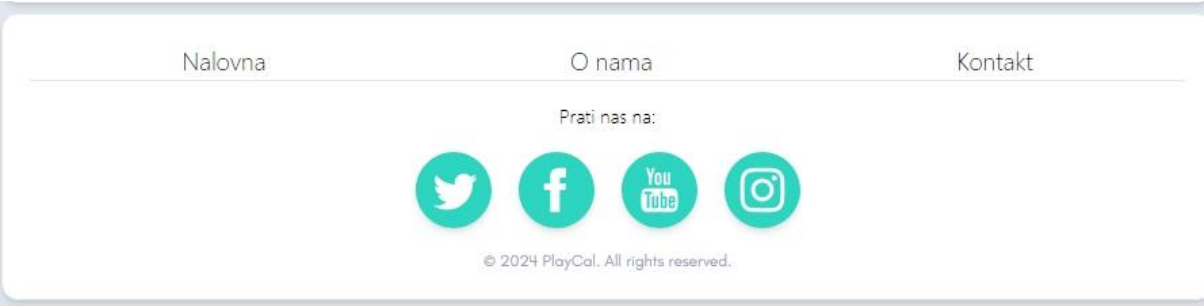
Slika 26. Prikaz zaglavlja kod prijavljenog korisnika na malim ekranima

```
const Footer = () => {
  return (
    <div className=" bg-white border p-5 rounded-xl shadow-md">
      <div className="flex justify-around mb-4 lg:space-x-44 text-md md:text-lg
font-thin border-b border-slate-200 ">
        <Link to="/" className="hover:text-primary-0" href="#">
          Nalovna
        </Link>
        <Link to="/about" className="hover:text-primary-0" href="#">
          O nama
        </Link>
        <Link to="/contact" className="hover:text-primary-0" href="#">
          Kontakt
        </Link>
      </div>
      <div className="flex justify-center mb-4 text-sm font-extralight">
        <span>Prati nas na:</span>
      </div>
      <div className=" flex justify-center items-center space-x-5">
          <button  className="  rounded-full  ease-in-out  duration-300  delay-100
shadow-md hover:scale-110">
```

Kod 21. Prikaz koda za podnožje web aplikacije

Kod 21 definira komponentu *Footer* koja se koristi za prikaz podnožja stranice u web aplikaciji. Podnožje se sastoji od nekoliko elemenata i koristi Tailwind CSS za stiliziranje. Prvi dio podnožja sadrži tri poveznice: "*Naslovna*", "O *nama*" i "*Kontakt*", koje vode na različite

stranice unutar aplikacije. Te su poveznice prikazane kao jednostavni tekstualni linkovi, a kada se korisnik pređe mišem preko njih, tekst mijenja boju na tirkiznu (definirano klasom `hover:text-primary-0`). U drugom dijelu podnožja nalazi se tekst "*Prati nas na:*" i ikone društvenih mreža. Svaka ikona je *SVG* grafika unutar gumba, a svaka od njih ima animaciju koja povećava gumb kada korisnik pređe mišem preko ikone. Ikone predstavljaju platforme poput Twittera, Facebooka i Instagrama. Na kraju, tu je i tekstualna oznaka koja prikazuje autorska prava na aplikaciju, zajedno s nazivom „*PlayCal*“ i napomenom da su sva prava zadržana. Tekst je stiliziran da bude diskretan i koristi sive tonove kako bi bio nenametljiv u odnosu na ostatak podnožja. Konačan izgled stranice je prikazan u slici 27 koja se nalazi ispod.



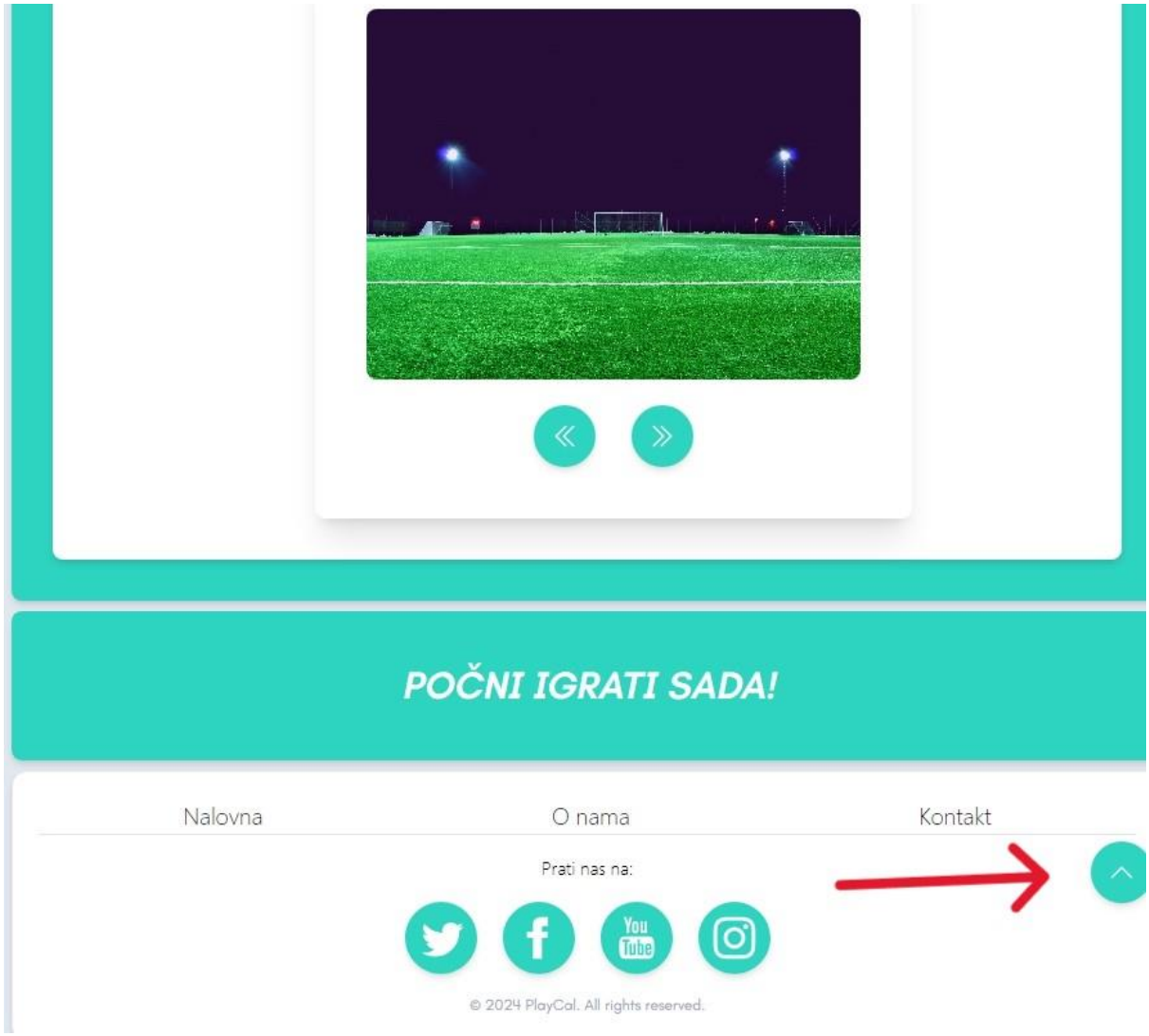
Slika 27. Prikaz izgleda zaglavlja web aplikacije

Komponenta `ScrollToTopButton` koristi se za prikaz gumba koji omogućuje korisnicima da se brzo vrate na vrh stranice kada se skrolira prema dolje. Gumb se pojavljuje samo kada korisnik skrolira više od 300 piksela od vrha stranice. Unutar komponente koristi se `useState` hook za praćenje vidljivosti gumba. `toggleVisibility` funkcija se koristi za postavljanje vidljivosti gumba na temelju trenutne pozicije skrola stranice. Ako je stranica skrolirana više od 300 piksela, gumb postaje vidljiv, a ako nije, gumb se skriva. `ScrollToTop` funkcija omogućuje glatko vraćanje korisnika na vrh stranice korištenjem `window.scrollTo` metode s opcijom za "*smooth*" ponašanje skrolanja. Unutar useEffect hooka dodaje se scroll event listener koji pokreće funkciju `toggleVisibility` svaki put kad korisnik skrola. Na kraju, ako je `isVisible` postavljen na true, gumb se prikazuje na fiksnoj poziciji u donjem desnom kutu stranice. Kada korisnik klikne na gumb, poziva se funkcija `scrollToTop` koja korisnika vraća na vrh stranice.

```
const ScrollToTopButton = () => {
  const [isVisible, setIsVisible] = useState(false);

  const toggleVisibility = () => {
    if (window.pageYOffset > 300) {
      setIsVisible(true);
    } else {
      setIsVisible(false);
    }
  };

  const scrollToTop = () => {
    window.scrollTo({
      top: 0,
      behavior: "smooth",
    });
  };

  useEffect(() => {
    window.addEventListener("scroll", toggleVisibility);
    return () => {
      window.removeEventListener("scroll", toggleVisibility);
    };
  }, []);

  return (
    <div className="fixed bottom-4 right-4">
      {isVisible && (
        <button
          onClick={scrollToTop}
```

Kod 22. Prikaz koda ScrollToTopButton-a

Konačan izgled prikazan je na slici 28 koja se nalazi ispod.



Slika 28. Prikaz izgleda ScrollToTopButton-a

4. ZAKLJUČAK

Ovaj rad prikazuje kako je uspješno razvijena web aplikacija za rezervaciju malonogometnih terena koja zadovoljava sve potrebe modernih korisnika i vlasnika sportskih objekata. Aplikacija je dizajnirana tako da olakša proces pretrage i rezervacije terena putem interneta, čime se eliminiraju problemi koji se javljaju kod tradicionalnih metoda rezervacije, poput telefonskih poziva ili osobnih dolazaka.

Razvoj aplikacije temeljio se na suvremenim tehnologijama kao što su programski okvir React za korisničko sučelje, programski okvir Tailwind CSS za stilizaciju i Firebase za pozadinsko poslužitelj funkcionalnosti. Ove tehnologije omogućile su brzu izradu responzivnog sučelja te stabilnu i sigurnu pohranu podataka. Korištenjem Firebase Authentication servisa omogućeno je jednostavno upravljanje korisničkim računima, dok Firestore baza podataka omogućuje real-time sinkronizaciju podataka o dostupnosti terena i korisničkim rezervacijama.

Tijekom rada posebna je pažnja posvećena korisničkom iskustvu, s fokusom na jednostavnost korištenja i optimizaciju procesa rezervacije. Implementirane su funkcionalnosti poput pregleda dostupnih termina, odabira željenih termina, upravljanja rezervacijama te jednostavne registracije i prijave korisnika. Dizajn aplikacije osigurava prilagodbu različitim veličinama ekrana, čime se postiže maksimalna dostupnost i funkcionalnost na svim uređajima.

Ovaj rad pokazuje kako se, uz korištenje suvremenih web tehnologija, mogu riješiti problemi koji prate tradicionalne metode rezervacije sportskih terena te kako se uz jednostavna, ali učinkovita rješenja može unaprijediti korisničko iskustvo. Aplikacija ne samo da korisnicima omogućuje brži i jednostavniji pristup sportskim terenima, nego također pomaže vlasnicima sportskih objekata u boljoj organizaciji i upravljanju terminima.

LITERATURA

ESLint Documentation, ESLint. (25. rujan 2024). Dohvaćeno iz ESLint.org: https://eslint.org/docs/latest

Firebase Documentation, Google. (25. rujan 2024). Dohvaćeno iz Google Firebase: https://tailwindcss.com/docs/installation

Git Documentation, Git. (25. rujan 2024). Dohvaćeno iz Git: https://git-scm.com/doc

JavaScript Documentation, Mozilla Developer Network. (25. rujan 2024). Dohvaćeno iz MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

JavaScript Documentation, Mozilla Developer Network. (25. rujan 2024). Dohvaćeno iz MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/JavaScript

Node.js Documentation, Node.js Foundation. (25. rujan 2024). Dohvaćeno iz Node.js: https://nodejs.org/en/docs

Prettier Documentation, Prettier. (25. rujan 2024). Dohvaćeno iz Prettier.io: https://prettier.io/docs/en/index.html

React (JavaScript library), Wikipedia. (23. rujan 2024). Dohvaćeno iz Wikipedia: https://en.wikipedia.org/wiki/React_(JavaScript_library)

React: A JavaScript library for building user interfaces, Meta. (23. rujan 2024). Dohvaćeno iz React.dev: https://react.dev/

Tailwind CSS Documentation, Tailwind Labs. (25. rujan 2024). Dohvaćeno iz Tailwind CSS: https://tailwindcss.com/docs

Visual Studio Code Documentation, Microsoft. (25. rujan 2024). Dohvaćeno iz Microsoft: https://code.visualstudio.com/docs

PRILOZI

Popis slika

Popis koda