Web aplikacija za upravljanje nogometnim klubom

Trbara, Jakov

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

Web aplikacija za upravljanje nogometnim klubom

Završni rad

Jakov Trbara

Osijek, 2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P: Obrazac za ocjenu završnog rada na sveučilišnom prijediplomskom studiju

Ocjena završnog rada na sveučilišnom prijediplomskom studiju

Ime i prezime pristupnika:	Jakov Trbara
Studij, smjer:	Sveučilišni prijediplomski studij Elektrotehnika i informacijska tehnologija
Mat. br. pristupnika, god.	4742, 23.07.2019.
JMBAG:	0165083614
Mentor:	doc. dr. sc. Krešimir Romić
Sumentor:	
Sumentor iz tvrtke:	Željko Brdarić
Naslov završnog rada:	Web aplikacija za upravljanje nogometnim klubom
Znanstvena grana završnog rada:	**Informacijski sustavi (zn. polje računarstvo)**
Zadatak završnog rada:	U ovom radu potrebno je izraditi responzivni frontend web aplikacije za upravljanje nogometnog kluba. Korisnička strana aplikacije treba omogućiti registraciju, prijavu korisnika, unošenje i prikaz igrača, izradu upitnika za igrače, pohranu tih podataka u bazu, te dohvaćanje podataka o performansama igrača s poslužiteljske strane aplikacije i njihov analitički prikaz. Naglasak ovog rada je na implementaciji responzivnog frontenda, spajanja na postojeću poslužiteljsku stranu i na ostvarivanju komunikacije između njih. Programsko rješenje treba ostvariti koristeći odgovarajuće programske jezike, tehnologije i razvojne okoline.

Datum prijedloga ocjene završnog rada od strane mentora:	17.09.2024.
Prijedlog ocjene završnog rada od strane mentora:	Dobar (3)
Datum potvrde ocjene završnog rada od strane Odbora:	27.09.2024.
Ocjena završnog rada nakon obrane:	Dobar (3)
Datum potvrde mentora o predaji konačne verzije završnog rada čime je pristupnik završio sveučilišni prijediplomski studij:	30.09.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 30.09.2024.

Ime i prezime Pristupnika:	Jakov Trbara
Studij:	Sveučilišni prijediplomski studij Elektrotehnika i informacijska tehnologija
Mat. br. Pristupnika, godina upisa:	4742, 23.07.2019.
Turnitin podudaranje [%]:	9

Ovom izjavom izjavljujem da je rad pod nazivom**: Web aplikacija za upravljanje nogometnim klubom**

izrađen pod vodstvom mentora doc. dr. sc. Krešimir Romić

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

SADRŽAJ

1. UVOD

Nogomet kakvog ga danas svi danas znamo, kao moderni organizirani sport, star je otprilike 150 godina. U rasponu od tih 150 godina se u jednu ruku malo toga promijenilo, a u drugu ruku mnogo toga se promijenilo. Nogomet je i dalje sport sastavljen od dvije protivničke ekipe koje se natječu međusobno, udarajući loptu nogom, pokušavajući tu loptu smjestiti u mrežu protivničkog gola. S vremenom, nogomet je postao jedan od najpopularnijih i najomiljenijih svjetski poznatih sportova. Rastom popularnosti i napretkom nogometa među obožavateljima diljem svijeta porasla je i potreba za drugim parametrima i potrebama koji prate taj sport. Većom gledanosti i većim brojem obožavatelja nastala je potreba za pomicanje granica u organizaciji nogometnih klubova koji se natječu.

Danas svaki moderni, uspješni i visoko produktivni nogometni klub ima, barem u nekakvom obliku, sistem praćenja napretka kluba u svakom mogućem pogledu. Od performansi svakog pojedinog igrača na treninzima i utakmicama, međusobne usklađenosti pojedinih parova igrača, postotak uspješnosti kombinacija pri izvođenju prekida tijekom utakmice do nebrojenih ostalih parametara. Stoga, uspješni klubovi koriste navedene servise ili *web* stranice koje im omogućuju povratnu vezu za njihove metode upravljanja svojim nogometnim klubovima koje im pružaju jasniju sliku uspješnosti ili neuspješnosti svojih klubova.

Tema ovog završnog rada upravo je izrada *web* aplikacije za upravljanje nogometnim klubom. Navedena *web* aplikacija za upravljanje nogometnim klubom namijenjena je osoblju nogometnog kluba koji ju koristi radi uvida u analitiku svog kluba. Aplikacija omogućuje registraciju, prijavu korisnika, unošenje i prikaz igrača, izradu upitnika za igrače, pohranu tih podataka u bazu, te dohvaćanje podataka o performansama igrača s poslužiteljske strane aplikacije i njihov analitički prikaz. Naglasak završnog rada je na implementaciji responzivnog dizajna, spajanje na postojeću poslužiteljsku stranu i na ostvarivanju komunikacije između njih.

1.1. Zadatak završnog rada

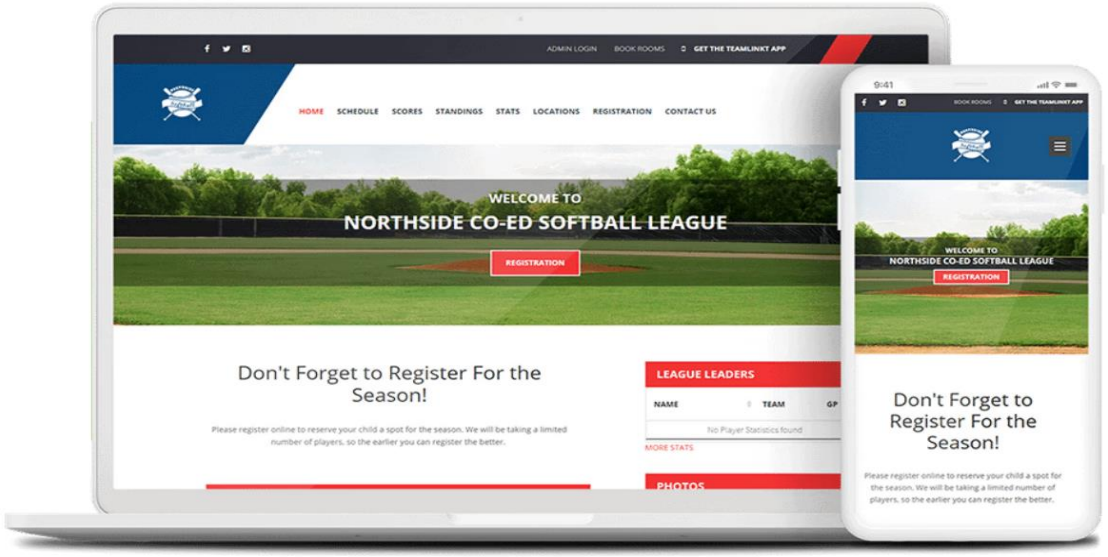
Zadatak ovog završnog rada je izrađivanje *web* stranice za upravljanje nogometnim klubom koja omogućuje registraciju, prijavu korisnika, unošenje i prikaz igrača, izradu upitnika za igrače, pohranu tih podataka u bazu, te dohvaćanje podataka o performansama igrača s poslužiteljske strane aplikacije i njihov analitički prikaz. Naglasak ovog rada je na implementaciji responzivnog dizajna, spajanja na postojeću poslužiteljsku stranu i na ostvarivanju komunikacije između njih. Programsko rješenje aplikacije ostvareno je koristeći programske jezike, tehnologije i razvojne okoline navedene u nastavku završnog rada.

2. POSTOJEĆA SLIČNA RJEŠENJA

Prije početka procesa razvoja *web* aplikacije za upravljanje nogometnim klubom potrebno je istražiti te analizirati postojeća rješenja koja pružaju slične funkcionalnosti. Istraživanjem i analizom postojećih sličnih rješenja ostvaruje se potpunija slika prednosti i nedostataka svakog pojedinačnog promatranog rješenja te se time pronalaze mogućnosti za unapređenje. Postojeće aplikacije i platforme uzete u obzir koriste se za vođenje sportskih timova, s posebnim naglaskom na nogometne klubove. Analizirane su aplikacije: *TeamLinkt* [1], *Director11* [2], *Vensica Football Club Manager* [3], *Wyscout* [4] te *SportEasy* [5].

2.1. TeamLinkt

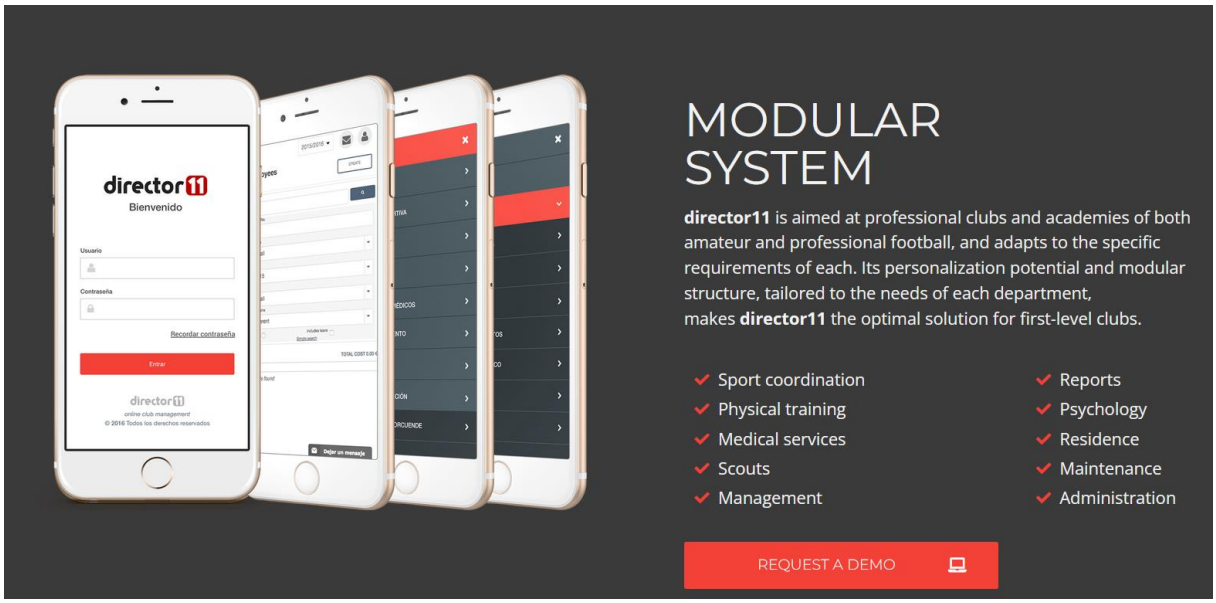
TeamLinkt je *web* platforma koja omogućava razna rješenja, kao što su: pristupačno upravljanje sportskim klubovima, komunikacija među članovima tima, prilagodljive *web* stranice i mobilne aplikacije te statistika sportskih klubova (Slika 2.1). Značajka upravljanja statistike igrača poslužila je kao inspiracija za razvoj analitičkog dijela aplikacije prikazane u ovom radu, u kojem se podaci o igračima prikupljaju, analiziraju te uređuju. Temeljna razlika *TeamLinkt* i aplikacije razvijene u ovome radu očituje se u tome da *TeamLinkt* ima fokus na komunikacijske alate i organizaciju sportskog tima, dok aplikacija u ovome radu ima cilj detaljnu analizu performansi igrača te responzivni dizajn koji omogućava korištenje aplikacije na velikom broju uređaja raznih veličina zaslona.



Sl. 2.1. Primjer ***TeamLinkt*** aplikacije

2.2. director11

Director11 je *web* platforma s glavnom primjenom kod nogometnih menadžera i trenera (Slika 2.2). Platforma omogućuje detaljne analize performansi sportskog tima, upravljanje strategijama te planiranje utakmica. Analiza performansi tima te pojedinačnih igrača tima omogućuje menadžerima i trenerima uvid u učinak tima i pojedinačnih igrača tijekom sezona, uspoređivanje prethodnih sezona te planiranje budućih strategija. *Director11* je dizajniran za nogometne klubove svih razina, ali osobito onih nogometnih klubova kojima je potrebna detaljna analitika i uvid u performanse igrača nogometnog kluba.



Sl. 2.2. Prikaz ***director11*** aplikacije

2.3. Vensica football club manager

Vensica Football Club Manager je *WordPress* tema koja je namijenjena za upotrebu u izradi *web* stranica za nogometne klubove (Slika 2.3). Svojim dizajnom, tema prezentira nogometni klub na profesionalan način. Tema sadrži prikaz rasporeda utakmica i rezultata, te slikama i videozapisima pridonosi privlačnom dizajnu.

Sl. 2.3. Prikaz ***Vensica football club manager*** aplikacije

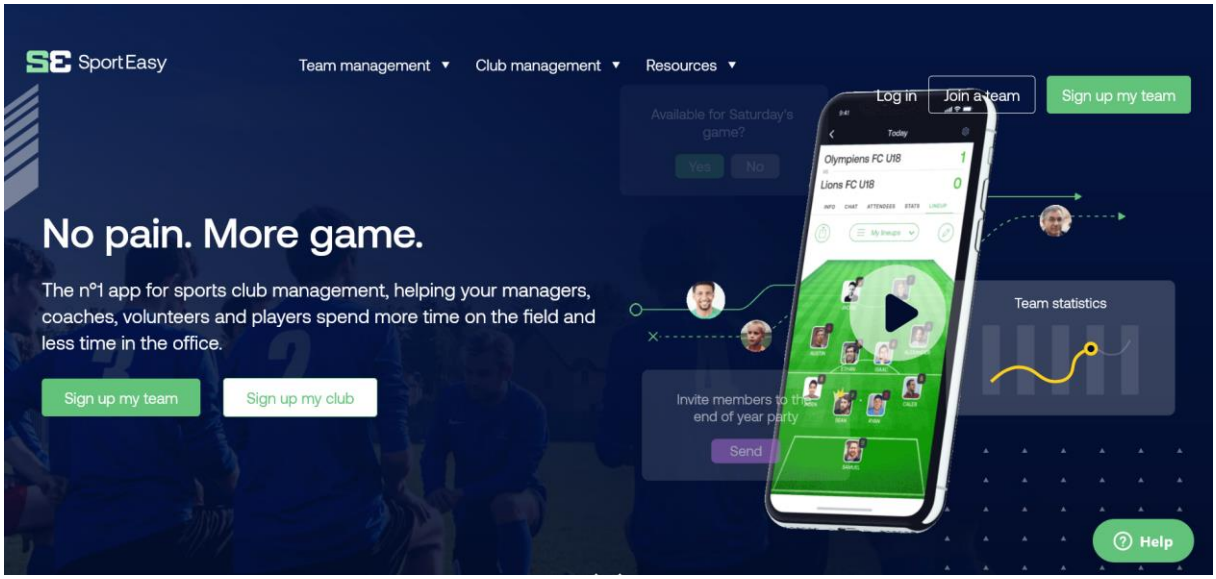
2.4. WYSCOUT

Wyscout je kompleksna platforma koja se primjenjuje za analizu nogometnih utakmica i igrača (Slika 2.4). Popularna je među profesionalnim klubovima, skautima i trenerima. Neke od naprednih mogućnosti koje ova platforma ima su: detaljne video analize i statistički podaci o igračima i timovima širom svijeta, praćenje performansi igrača, analize odigranih utakmica i kreiranje izvještaja. Prednost *Wyscout*-a je mogućnost proučavanja protivničkih timova, analiziranje njihovih slabosti te prepoznavanje potencijalnih talenata.

Sl. 2.4. Prikaz ***Wyscout*** aplikacije

2.5. SportEasy

SportEasy je platforma za upravljanje sportskim timovima koja se primarno fokusira na jednostavnost korištenja platforme te korištenje intuitivnog sučelja (Slika 2.5). Neke mogućnosti koje *SportEasy* omogućava su: upravljanje rasporedima, utakmicama, statističkim podacima i članovima. Zbog svoje jednostavnosti, ova platforma je popularna među amaterskim i polu-profesionalnim sportskim timovima.



Sl. 2.5. Prikaz ***SportEasy*** aplikacije

3. TEHNOLOGIJE KORIŠTENE U IZRADI APLIKACIJE

U procesu razvoja *web* aplikacije za upravljanje nogometnim klubom korištene su tehnologije koje omogućuju funkcionalnost, responzivnost te integraciju korisničkog sučelja (engl. *frontend*) s poslužiteljskom (engl. *backend*) [6] stranom *web* aplikacije. Tehnologije koje su korištene u *web* aplikaciji za upravljanje nogometnim klubom su: *Angular, TypeScript, HTML, CSS, Spring Boot, REST API te IntelliJ IDEA.*

3.1. HTML i CSS

HTML (engl. *HyperText Markup Language*) i *CSS* (engl. *Cascading Style Sheets*) su temelji većine *web* aplikacija na internetu. *„Hypertext"* se odnosi na poveznice koje spajaju *web* stranice međusobno ili unutar samih *web* stranica. *CSS* je stilski jezik koji se upotrebljava za opisivanje prezentacije dokumenta napisanog u *HTML*-u. Uloga *CSS*-a je da opisuje kako se elementi trebaju prikazivati na zaslonu. Može se slikovito reći da je *HTML* kostur *web* stranica, a *CSS* je vanjski izgled toga kostura sa svim fizičkim i vidljivim karakteristikama koje ga opisuju.

3.2. TypeScript

TypeScript je programski jezik koji je nastao na temeljima *JavaScripta*. Stoga je sintaksa *JavaScripta* važeća u *TypeScriptu*. *TypeScript* je proširenje *JavaScripta* i za razliku od *JavaScripta*, omogućava veću kontrolu nad tipovima podataka u kôdu te ima strožu kontrolu tipova podataka u kôdu. U projektima se koristi za stvaranje klasa, modula i sučelja te se koristi za definiranje strukture aplikacije i za upravljanje tokom informacija koje se prikazuju korisnicima [9].

3.3. Angular

Angular [7] je okvir (engl. *framework*) za razvijanje aplikacija i platformi za razvoj koji omogućuje stvaranje visoko učinkovitih i naprednih jednostraničnih (engl. *Single Page Application - SPA*) aplikacija. Izgrađen je na *TypeScript*-u, a kao platforma uključuje:

- Okvir za izgradnju skalabilnih *web* aplikacija zasnovan na konceptu komponenata
- Zbirku vrlo kvalitetno integriranih biblioteka koje pokrivaju razne funkcionalnosti, kao što su npr. *routing*, upravljanje formama, komunikacija između klijenta i servera i mnoge druge funkcionalnosti
- Skup alata za razvoj koji vam pomažu pri razvijanju, izgradnji, testiranju i ažuriranju svog kôda

Angular je platforma koja ima mogućnost prilagodbe projektima s jednim razvojnim timom sve do aplikacija na razini poduzeća. *Angular* je dizajniran da omogući što jednostavnija i neprimjetnija ažuriranja, kako bi mogli imati koristi od najnovijih verzija s minimalnim uloženim trudom.

3.3.1. Komponente

Komponente su konstrukcijski blokovi koji čine temelj svake *Angular* aplikacije. Svaka komponenta sastoji se od: TypeScript klase sa *@Component()* dekoratorom kôda, *HTML* predloškom te stilovima. Dekorator *@Component()* definira sljedeće informacije specifične za svaku *Angular* aplikaciju (slika 3.1):

- *CSS* selektor koji određuje kako će se komponenta koristiti u predlošku. *HTML* elementi u predlošku koji odgovaraju ovom selektoru postaju instance komponente.
- *HTML* predložak koji naređuje Angular-u kako da prikaže komponentu.
- Neobavezan set *CSS* stilova koji definira izgled *HTML* elemenata u predlošku

Slijedi prikaz minimalne *Angular* komponente:

```
import { Component } from '@angular/core';

@Component({
  selector: 'hello-world',
  template: `
    <h2>Hello World</h2>
    <p>This is my first component!</p>
  `
})
export class HelloWorldComponent {
  // The code in this class drives the component's behavior.
}
```

Sl. 3.1. Kôd primjer *typescript* klase

Kako bi mogli koristiti komponentu, kôd na slici 3.2 se upisuje u *HTML* predložak.

```
<hello-world></hello-world>
```

Sl. 3.2. Kôd *HTML* predloška

Kad *Angular* učita komponentu, za rezultat dobijemo *DOM* [8] koji je prikazan na slici 3.3.

```
<hello-world>
    <h2>Hello World</h2>
    <p>This is my first component!</p>
</hello-world>
```

Sl. 3.3. Kôd rezultat-a u *DOM*-u

3.3.2. HTML Predlošci

Svaka komponenta ima *HTML* predložak koji deklarira kako će se komponenta prikazati. Programer definira predložak *inline* načinom ili zasebnim *HTML* dokumentom. *Angular* dodaje sintaksne elemente koji proširuju mogućnosti *HTML*-a na način da se mogu unositi dinamične vrijednosti koje dolaze iz komponente.

3.4. Spring Boot

Spring Boot je *framework* kojemu je glavni zadatak maksimalno pojednostaviti *back-end* razvoj aplikacije [10], [11]. *Spring Boot* upotrebljava *JPA (*engl. *Java Persistence API)* kako bi programeru pojednostavio i olakšao implementaciju slojeva pristupa podacima. Na primjer, stvarajući sučelje repozitorija bilo kojim načinom, *Spring* će sve automatski spojiti umjesto programera. U projektu ovoga završnog rada, za izradu poslužiteljske strane aplikacije rada koristi se *H2* koji omogućava manipulaciju podataka o igračima i korisnicima aplikacije, te obavlja povezivanje s bazom podataka [12].

3.5. REST API

Kako bi se omogućila komunikacija između korisničkog sučelja i poslužiteljskog dijela aplikacije, ključno je upotrijebiti *REST API*. *REST API* (također nazvan *RESTful API* ili *RESTful web API*) je

sučelje za programiranje aplikacija (engl. *Application Programming Interface - API*) koje je u skladu s načelima dizajna arhitektonskog stila reprezentativnog prijenosa stanja (*REST*). *REST API*-ji pružaju fleksibilan, lagan način za integraciju aplikacija i povezivanje komponenti u arhitekturama mikro servisa [13]. *REST API* omogućava aplikaciji vrlo bitne operacije kao što su slanje zahtjeva serveru za dohvaćanje i ažuriranje podataka aplikacije koristeći *HTTP* metode kao što su *GET, POST, PUT* i *DELETE* [14]. Kako bi se omogućio prijenos podataka o igračima, kao i autorizacija korisnika aplikacije upotrijebljen je *REST API* pomoću *HTTP* metoda.

4. IMPLEMENTACIJA

U ovom poglavlju biti će detaljno objašnjena implementacija projekta *web* aplikacije za upravljanje nogometnim klubom. Temeljni koncepti ove *Angular* aplikacije su responzivan dizajn uz korištenje komponenti iz *Angular Material* biblioteke, upravljanje igračima nogometnog kluba, pohranjivanje i dohvaćanje igrača sa poslužiteljske strane aplikacije, prikazivanje performansi pojedinačnih igrača nogometnog kluba te komunikacija između korisničkog sučelja i poslužiteljske strane aplikacije pomoću *REST API*-ja putem *HTTP* zahtjeva. Za korisničko sučelje je korišten *Angular* dok je za poslužiteljsku stranu upotrijebljen *Spring Boot*.

4.1. Početno postavljanje aplikacije

4.1.1. Instalacija Angular framework-a

Za projekt je upotrijebljen razvojni okvir (engl. *framework*) *Angular* jer ima svojstvo modularnosti i skalabilnosti. Projekt se početno inicijalizira korištenjem *Angular CLI* alata, koji omogućuje efikasno i brzo postavljanje te upravljanje projektom. Početni koraci u razvoju aplikacije projekta su sljedeći:

1. **Instalacija *Angular CLI* alata:**

- Naredba za instalaciju *Angular CLI* nalazi se na slici 4.1.

Kod

```
npm install -g @angular/cli
```

Sl. 4.1. Kôd instalacije *Angular CLI*

- Nakon uspješne instalacije kreira se novi projekt naredbom na slici 4.2.

Kod

```
ng new final-paper
```

Sl. 4.2. Kôd stvaranja novog projekta

Tijekom procesa stvaranja novog projekta odabire se opcija *SCSS* za stilizaciju. Zatim *Angular* procesom automatizma stvara osnovnu strukturu aplikacije.

2. **Pokretanje aplikacije**:

- Aplikacija se pokreće sljedećom naredbom na slici 4.3.

Kod

```
ng serve
```

Sl. 4.3. Kôd pokretanja projekta

- Naredba „*ng serve*“ pokreće lokalni server na kojemu se učitava aplikacija, a na pregledniku se nalazi na adresi *http://localhost:4200*.

3. ***Angular Material***

- Instalacija se obavlja naredbom na slici 4.4:

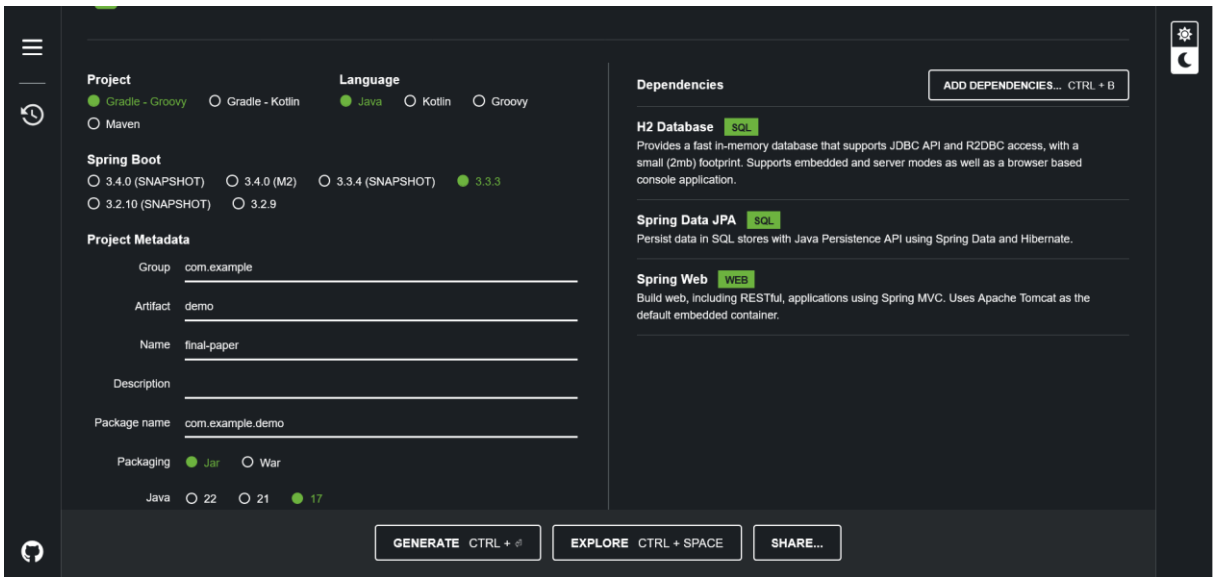
Kod

```
ng add @angular/material
```

Sl. 4.4. Kôd instaliranja *Angular Material*-a

4.1.2. Postavljanje poslužiteljske strane aplikacije

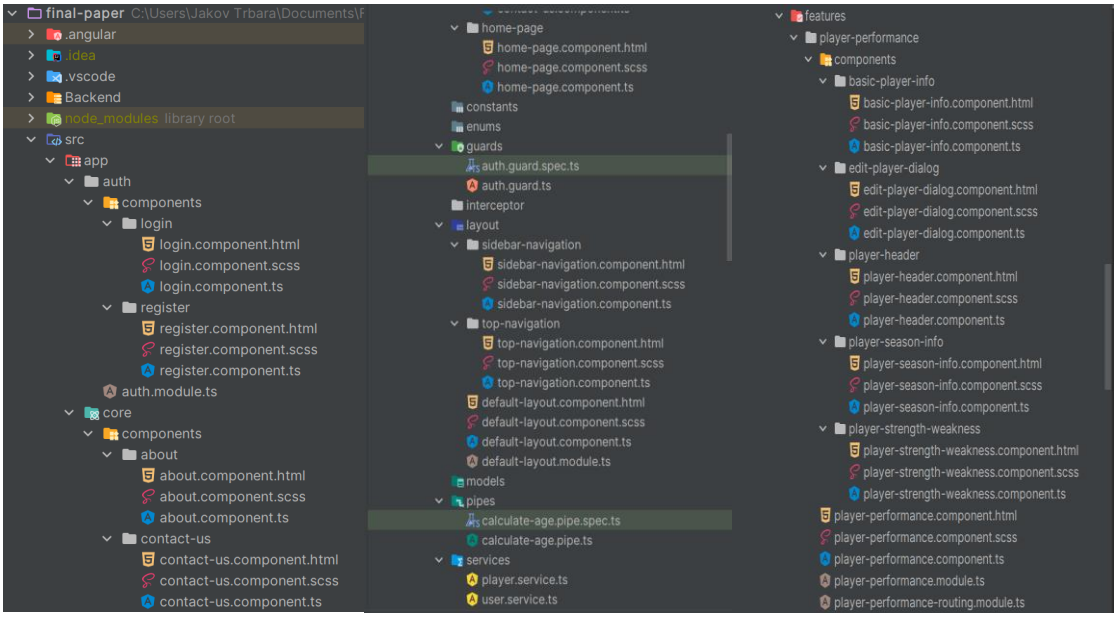
Za razvoj poslužiteljske strane aplikacije korišten je *Spring Boot* zbog jednostavosti postavljanja i mnoštva ugrađenih funkcionalnosti poput vrlo bitne podrške za *REST API* i *HTTP* protokola što je ključno za komunikaciju korisničkog sučelja i poslužiteljske strane. Generiranje projekta sa svim potrebnim ovisnostima obavlja se putem *Spring Initializrera* (Slika 4.5).



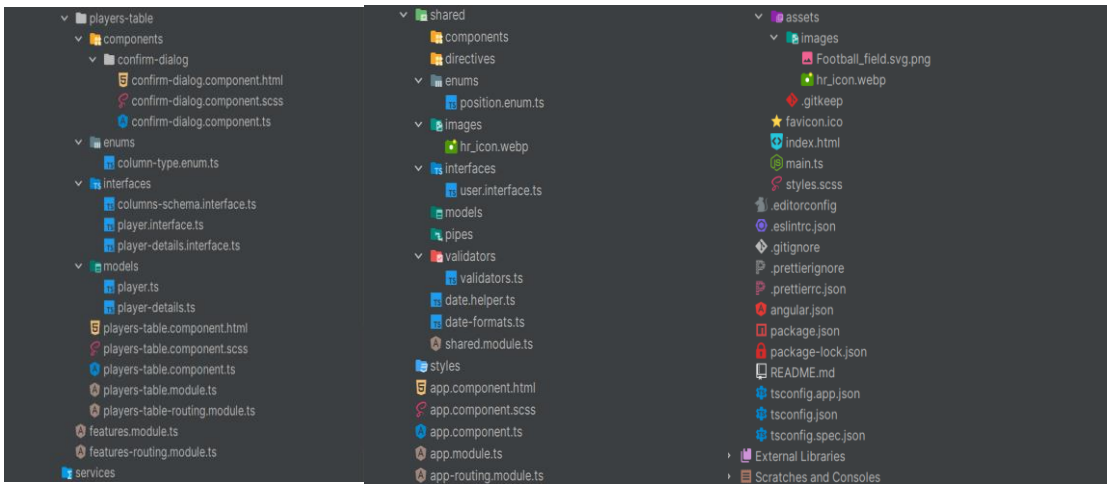
Sl. 4.5. *Spring Initializr*

4.2. Struktura aplikacije i komponenti

Aplikacija je strukturirana tako da je podijeljena na module i komponente kako bi se ostvarila modularnost aplikacije. Cjelokupna struktura aplikacije je prikazana na slikama 4.6 i 4.7.



Sl. 4.6. Kod strukture foldera aplikacije prvi dio



Sl. 4.7. Kod strukture foldera aplikacije drugi dio

4.2.1. Moduli aplikacije

Kao što je već prethodno navedeno, aplikacija je strukturirana pomoću *Angular* modula. Glavni moduli aplikacije su sljedeći: ***App Module***, ***App Routing Module***, ***Shared Module***, ***Features Module*** i ***Auth Module***. ***App module*** je glavni modul *Angular aplikacije* i on je središnje mjesto na kojem se povezuju svi ostali moduli od drugih komponenti, komponente i servisi iz aplikacije. Na slici 4.8 i slici 4.9 se nalazi kôd iz *App Module*-a.

```
import {NgModule}      from '@angular/core';
import {BrowserModule} from '@angular/platform-browser';

import {AppRoutingModule}                from './app-routing.module';
import {AppComponent}                    from './app.component';
import {BrowserAnimationsModule}         from '@angular/platform-browser/animations';
import {FormsModule, ReactiveFormsModule} from '@angular/forms';
import {FeaturesModule}                  from './features/features.module';
import {DefaultLayoutModule}             from './core/layout/default-layout.module';
import {HttpClientModule}                from '@angular/common/http';
import {AuthModule}                      from './auth/auth.module';
import {HomePageComponent}               from './core/components/home-page/home-page.component';
import {AboutComponent}                  from './core/components/about/about.component';
import {SharedModule}                    from './shared/shared.module';
import {ContactUsComponent}              from './core/components/contact-us/contact-us.component';

2 usages  Jakov Trbara
@NgModule ({
            declarations: [
              AppComponent,
              HomePageComponent,
              AboutComponent,
              ContactUsComponent
            ],
            imports: [
              BrowserModule,
              AppRoutingModule,
              BrowserAnimationsModule,
              FormsModule,
              FeaturesModule,
              DefaultLayoutModule,
              HttpClientModule,
              ReactiveFormsModule,
              AuthModule,
```

Sl. 4.8. Kôd *App Module* prvi dio

Sl. 4.9. Kôd *App Module* drugi dio

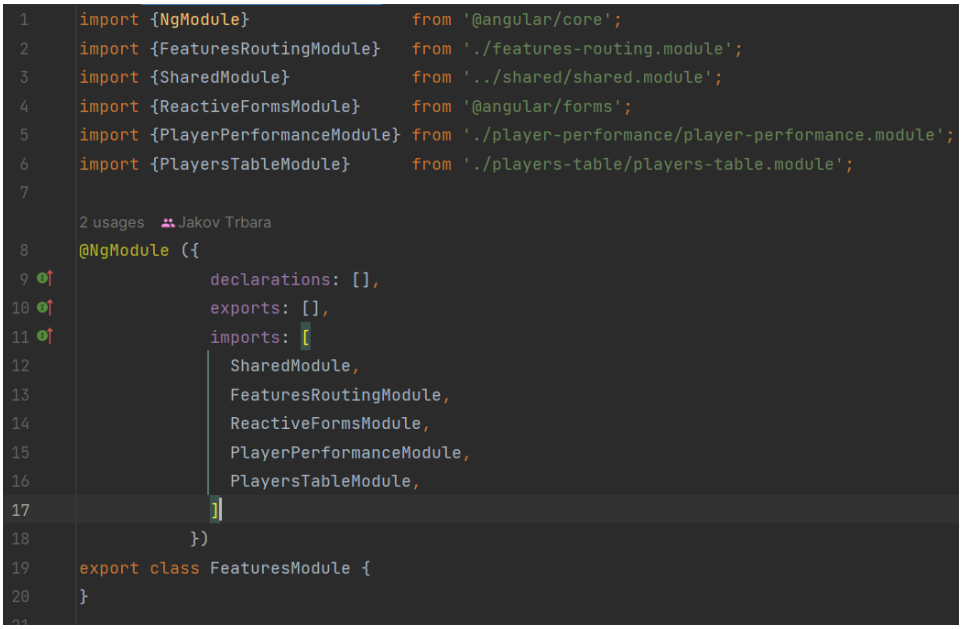
App Routing Module je modul zaslužan za ispravnu navigaciju aplikacije. Unutar *App Routing Module*-a nalaze se definicije putanja između različitih dijelova aplikacije. Putanje se nalaze na slici 4.10 i na slici 4.6.

```
import {RegisterComponent}     from './auth/components/register/register.component';
import {LoginComponent}        from './auth/components/login/login.component';
import {PlayersTableComponent} from './features/players-table/players-table.component';
import {AuthGuard}             from './core/guards/auth.guard';
import {HomePageComponent}     from './core/components/home-page/home-page.component';
import {AboutComponent}        from './core/components/about/about.component';
import {ContactUsComponent}    from './core/components/contact-us/contact-us.component';

const routes: Routes = [
  {
    path: '',
    redirectTo: '/home',
    pathMatch: 'full'
  },
  {
    path: 'home',
    component: HomePageComponent,
    canActivate: [AuthGuard],
  },
  {
    path: 'about',
    component: AboutComponent,
    canActivate: [AuthGuard],
  },
  {
    path: 'contact-us',
    component: ContactUsComponent,
    // canActivate: [AuthGuard],
  },
```

Sl. 4.10. Kôd *App Routing Module* prvi dio

Sl. 4.6. Kôd *App Routing Module* drugi dio

Putanje se definiraju stvaranjem polja koje sadrži objekte, pri čemu svaki objekt predstavlja jednu putanju. Svaka putanja u objektu određuje odgovarajući URL (engl. *Uniform Resource Locator*) i komponentu koja će se prikazati kada se ta putanja posjeti. *Lazy Load* je metoda koja omogućava da se moduli učitavaju samo kada su potrebni. Umjesto učitavanja svih modula pri pokretanju aplikacije, upotrebljava se svojstvo *loadChildren* u putanjama gdje je potreban *Lazy Load*. Time se modul, u kojemu se nalazi komponenta određene putanje, učitava samo kada korisnik pristupi toj putanji. Na taj način se značajno poboljšava učinkovitost aplikacije time što se smanjuje početno vrijeme učitavanja. U ***Shared Module*** se nalaze svi moduli koji bi se trebali moći koristiti bilo gdje u aplikaciji gdje je potrebno koristiti neki od često korištenih modula na način da se uveze *Shared Module* u komponentu gdje je potreban jedan od modula koji se nalaze u njemu kako bi se mogao koristiti. Taj modul je praktičan iz razloga što se njime maksimalno smanjuje dupliciranje koda u različitim dijelovima aplikacije jer se komponente iz *Shared Modulea* mogu koristiti na više mjesta u aplikaciji tako što učitamo željenu komponentu iz *Shared Modulea* na mjesto gdje je potrebna ta komponenta. ***Features Module*** sadržava u sebi module svih „*feature*“ komponenata. Te komponente su odgovorne za prikaz i upravljanje podacima o igračima nogometnog kluba unutar web aplikacije za upravljanje nogometnim klubom. Moduli koji se nalaze u *Features Moduleu* su prikazani na slici 4.7.



Sl. 4.7. Kôd *Features Module*-a

Auth Module je modul koji upravlja funkcionalnostima autentifikacije korisnika u aplikaciji. Sadrži *Login* i *Register* module koji u svojoj logici imaju implementaciju registriranja i prijave korisnika aplikacije.

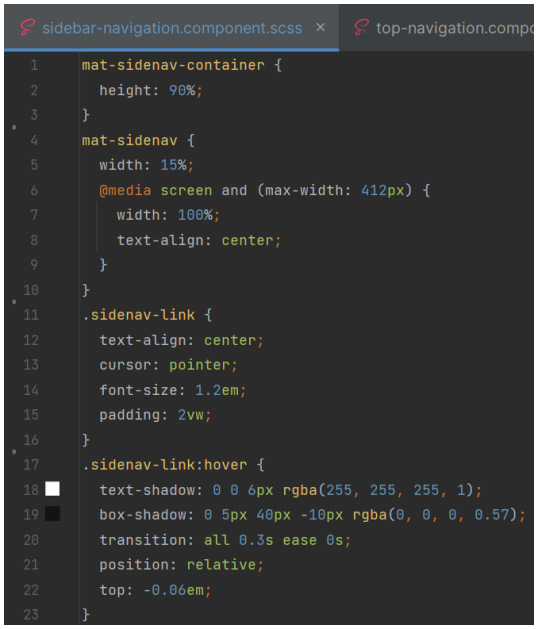
4.3. Implementacija tehnologija za responzivni dizajn

Responzivni *web* dizajn je proces u kojem se *web* aplikacija razvija s primarnim ciljem da bude optimalno dostupna za prikaz na različitim zaslonima raznih uređaja. S razvojem tehnologije dolaze nove dimenzije zaslona pametnih uređaja, a time se stvara potreba za optimizacijom *web* dizajna koji odgovara tim zahtjevima. Responzivni dizajn *web* stranica prilagođava izgled stranica , bilo da se radi o računalu, tabletu ili pametnom telefonu.

4.3.1. Primjena responzivnog dizajna u projektu

Primjena responzivnog *web* dizajna u aplikaciji za upravljanje nogometnim klubom bit će prezentirana na primjerima komponenti: ***Sidebar Navigation Component*** i ***Top Navigation Component***. *Sidebar Navigation Component* je komponenta koja služi kao navigacijsko okno, koje je prvotno skriveno, a zatim otkriveno s lijeve strane ukoliko se klikne padajući izbornik koji se nalazi na *Top Navigation* komponenti. Na *Sidebar Navigation* komponenti nalaze se poveznice na druge dijelove *web* stranice. Responzivnim dizajnom, omogućeno je da se komponenta automatski prilagođava raznim veličinama ekrana, te ukoliko korisnik koristi dovoljno malen ekran komponenta mijenja svoje ponašanje i izgled. Na manjim se ekranima aktiviranjem *Sidebar Navigation* komponente prikazuje preko cijele širine zaslona umjesto samo 15% sa lijeve strane

zaslona. Na slici 4.8 može se uočiti medijski upit označen sa *@media* koji označava da ukoliko je zaslon širine 412 piksela ili manje, širina elementa *mat-sidenav* biti će postavljena na cijelu širinu zaslona i tekstualni elementi unutar tog elementa će se centrirati.



Sl. 4.8. *SCSS* Kôd *Sidebar Navigation Component*

Sljedeća na redu je komponenta *Top Navigation Component* koja se nalazi na gornjem rubu zaslona aplikacije i ima ulogu prikazivanja elemenata koji su zaslužni za *Sidebar Navigation* komponentu, navigaciju do početne stranice aplikacije te funkciju odjavljivanja korisnika iz aplikacije. Kod *Top Navigation* komponente bitno je naglasiti nekoliko značajki koje joj omogućavaju responzivan dizajn i ponašanje. U *Angular Material* klasi *mat-toolbar* visina je namještena na 10% što omogućava da taj element ne bude fiksne veličine (npr. 100 piksela), već uvijek zauzima 10% zaslona te se automatski prilagođava raznim veličinama zaslona uređaja. U *SCSS* klasi *mat-toolbar-row* koristi se *flexbox* koji služi za fleksibilan dizajn te da elementi na koje se primjeni *flexbox* automatski zauzimaju slobodan prostor na responzivan način. Svojstvo *justify-content* unutar te klase ima vrijednost *space-between* što primjenjuje oblikovanje na elemente te klase da budu ravnomjerno raspoređeni po horizontalnoj liniji sa jednakim slobodnim prostorom između tih elemenata. Kôd za *Top Navigation* komponentu nalazi se na slici 4.9.

```scss
mat-toolbar {
  height: 10%;
  padding: 0;
}

.clickable-title {
  cursor: pointer;
}

.clickable-title:hover {
  text-shadow: 0 0 6px rgba(255, 255, 255, 1);
  box-shadow: 0 5px 40px -10px rgba(0, 0, 0, 0.57);
  transition: all 0.3s ease 0s;
  position: relative;
  top: -0.06em;
}

mat-toolbar-row {
  display: flex;
  justify-content: space-between;
}
```

Sl. 4.9. *SCSS* Kôd *Top Navigation Component.*

4.4. Poslužiteljska strana aplikacije

Poslužiteljska strana web aplikacije izrađena je pomoću *Spring Boot* razvojnog okvira koji omogućava jednostavnu i brzu implementaciju.

4.4.1. Glavna klasa

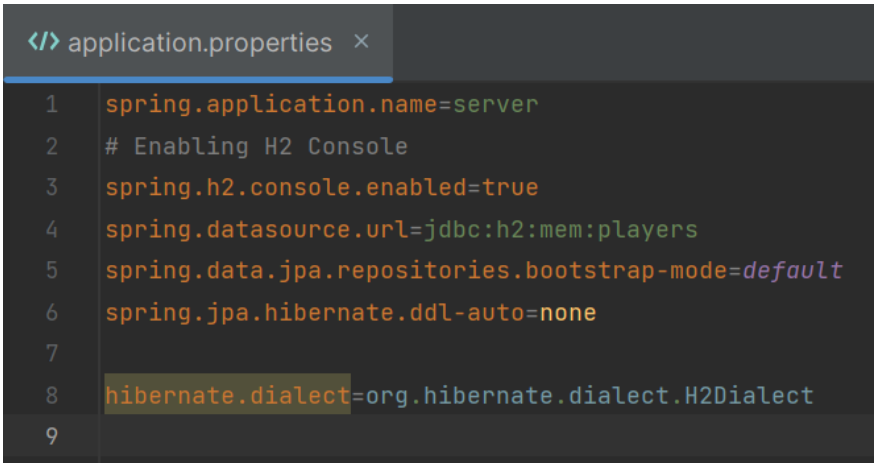
Glavna klasa aplikacije nalazi se u dokumentu „*ServerApplication.java*“ pod nazivom *ServerApplication*, a označena je anotacijom *@SpringBootApplication*. Anotacijom kojom je označena glavna klasa označava se početna točka aplikacije koja uključuje svu potrebnu konfiguraciju za pokretanje *Spring Boot* aplikacije (Slika 4.10).

```java
package com.example.server;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class ServerApplication {

  public static void main(String[] args) {
    SpringApplication.run(ServerApplication.class, args);
  }

}
```

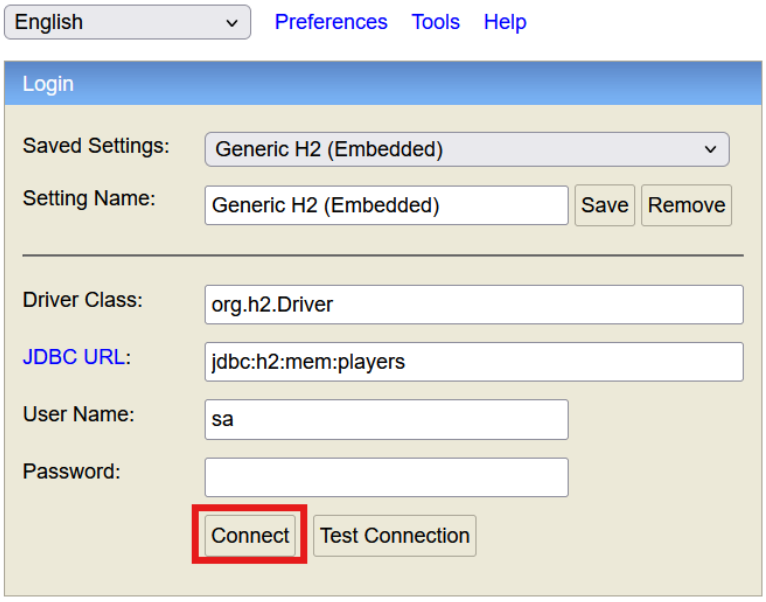
Sl. 4.10. Kôd glavne klase.

4.4.2. Konfiguracija baze podataka

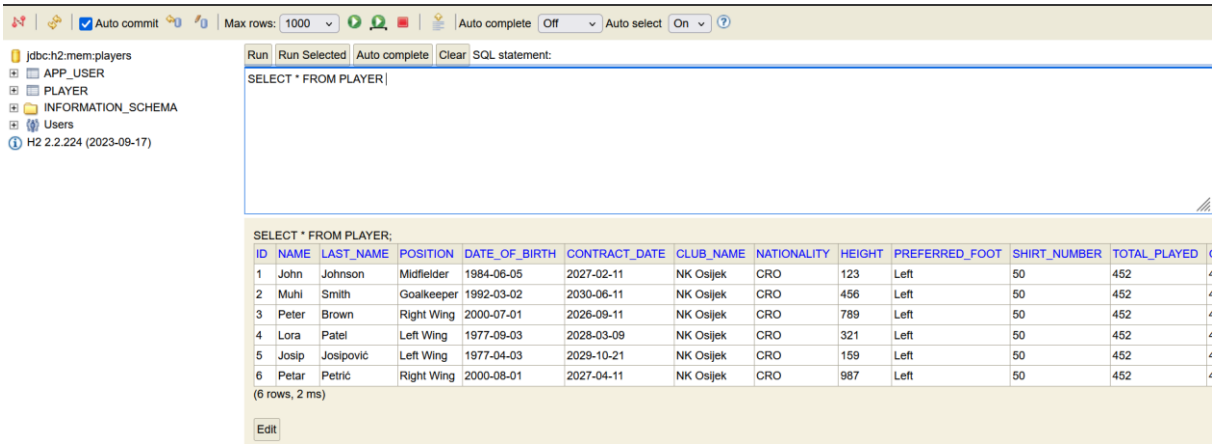
Kao što je već prethodno navedeno u ovom radu se za bazu podataka koristi *H2 in-memory* baza podataka. Definicija baze podataka je postavljena u „*application.properties*“ dokumentu (Slika 4.11). Korištenjem *H2* baze podataka oduzima se potreba za vanjskom bazom podataka, već je napravljena u aplikaciji. Također moguć je pristup *H2* konzoli putem preglednika gdje se mogu na pregledniji način nadzirati podaci unutar baze podataka (Slika 4.12 , Slika 4.13).



Sl. 4.11. Kôd „*application.properties*“.



Sl. 4.12 Ulaz u *H2* konzolu.



Sl. 4.13.*H2* konzola.

4.4.3. Entiteti

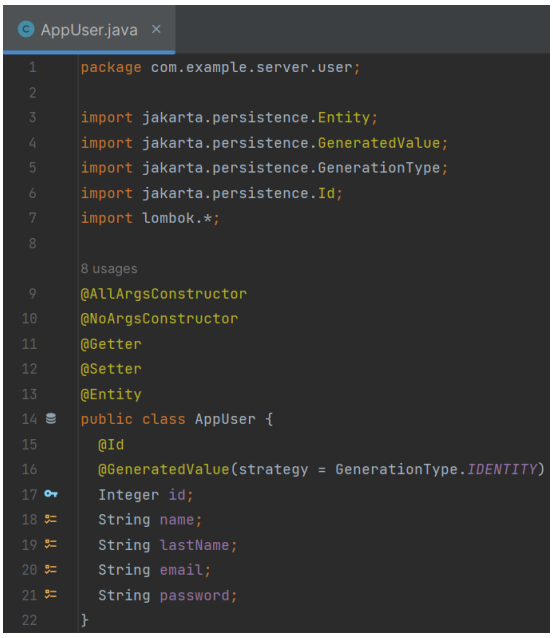
Entiteti su *Java* klase koje su anotirane s *@Entity* oznakom te oni predstavljaju tablice u bazi podataka. U poslužiteljskoj strani ovoga rada egzistiraju dva entiteta, a to su entiteti *Player* i *User*. Entitet *Player* predstavlja igrača u bazi podataka, a svakog pojedinačnog igrača definiraju svojstva poput *ID*-ja, imena, prezimena, pozicije i slično (Slika 4.14).

Player.java

```java
package com.example.server.player;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

import java.time.LocalDate;

15 usages
@AllArgsConstructor
@NoArgsConstructor
@Getter
@Setter
@Entity
public class Player {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  Integer id;
  String name;
  String lastName;
  String position;
  LocalDate dateOfBirth;
  LocalDate contractDate;
  String clubName;
  String nationality;
  Integer height;
  String preferredFoot;
  Integer shirtNumber;
  Integer totalPlayed;
  Integer gameStarted;
  Double minutesPerGame;
  Integer teamOfWeek;
  Integer goals;
  Integer expectedGoals;
  Integer scoringFrequency;
  Integer goalsPerGame;
  Double shotsPerGame;
  Double shotsOnTargetPerGame;
  Integer bigChancesMissed;
  String firstStrength;
  String secondStrength;
  String thirdStrength;
  String firstWeakness;
  String secondWeakness;
  String thirdWeakness;
  Integer totalPasses;
  Integer keyPasses;
  Integer successfulPasses;
  Integer interceptions;
  Integer tackles;
  Integer ballRecovered;
  Integer ballWon;
  Integer redCard;
  Integer yellowCard;
}
```

Sl. 4.14. Kôd *player* entiteta.

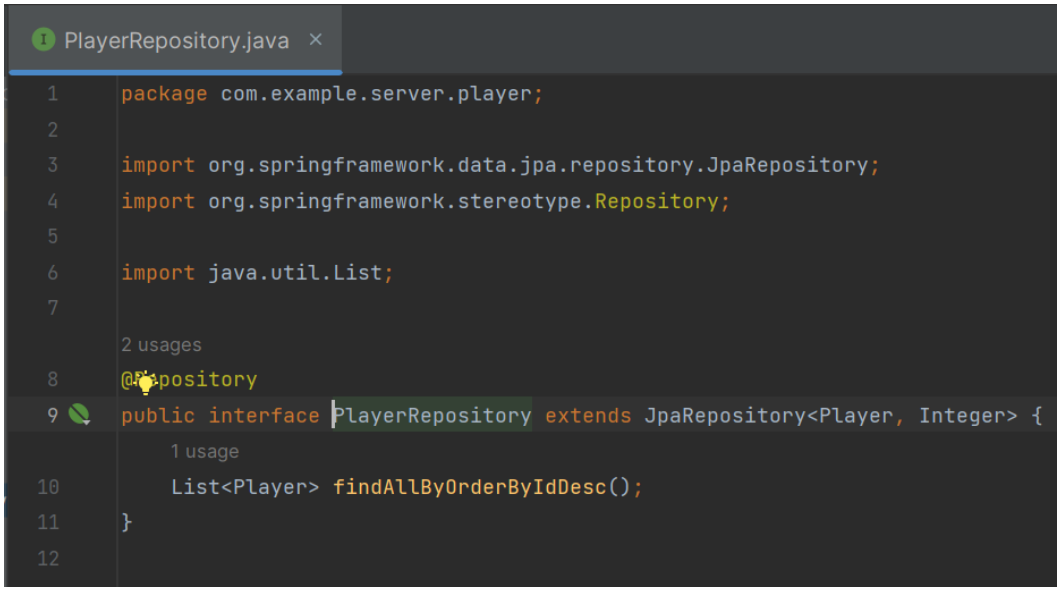
App User entitet predstavlja korisnika aplikacije te sadrži informacije o tome korisniku koje su bitne za njegovu registraciju ili prijavljivanje u aplikaciju (Slika 4.15).



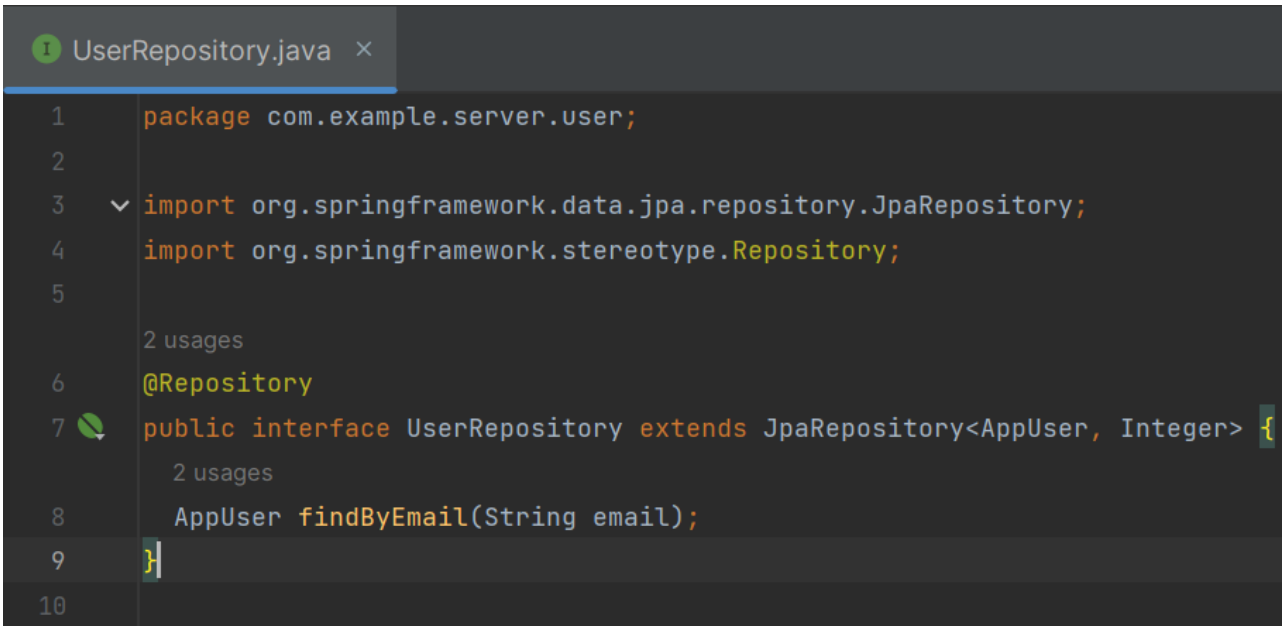
Sl. 4.15. Kôd *user* entiteta.

4.4.4. Repozitoriji

Repozitorij je sučelje koji daje kontrolu nad podacima baze podataka. Aplikacija koristi *Spring Data JPA* koja generira sve ključne metode za rad s bazom podataka. *Player Repository* proširuje funkcionalnost *Spring Dana JPA* te omogućuje operacije poput spremanja, dohvaćanja, ažuriranja podataka i brisanje igrača (Slika 4.16). *User Repository* jednako tako proširuje funkcionalnost *JpaRepository*-ja te pruža mogućnost upravljanja podacima korisnika aplikacije (Slika 4.17).



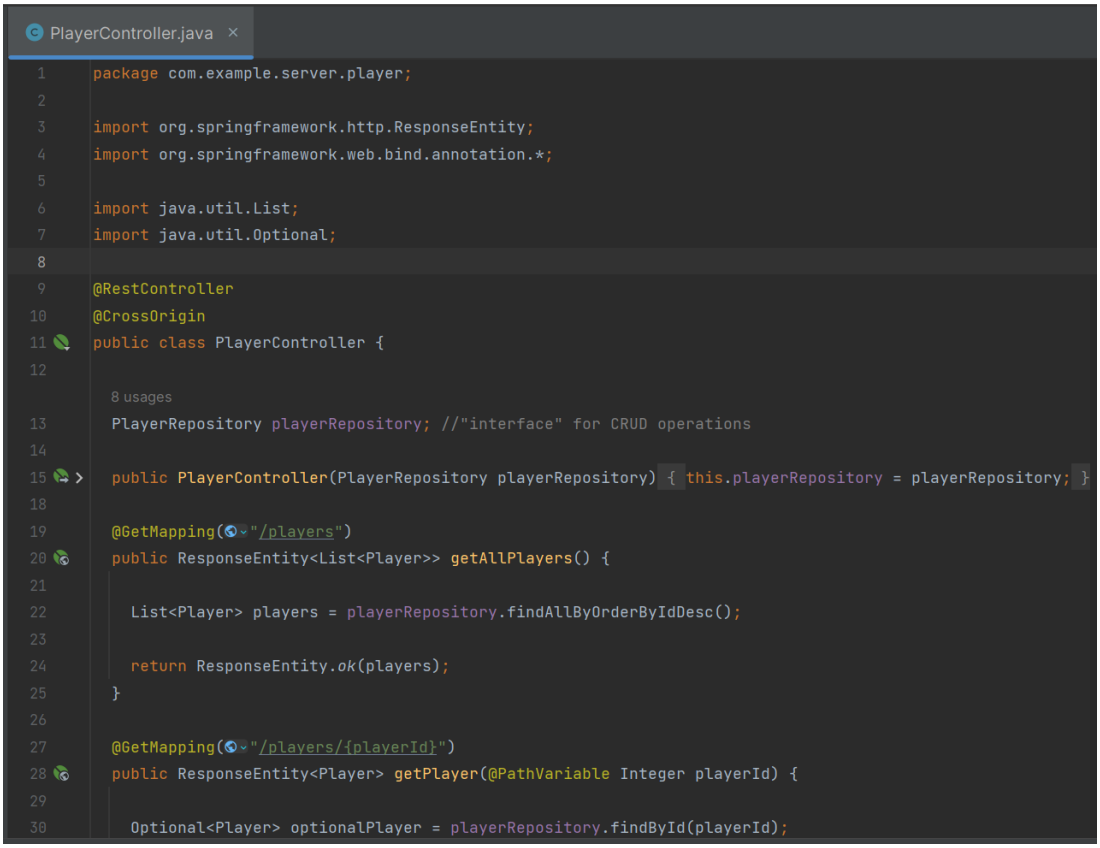
Sl. 4.16. Kôd *Player Repository*.



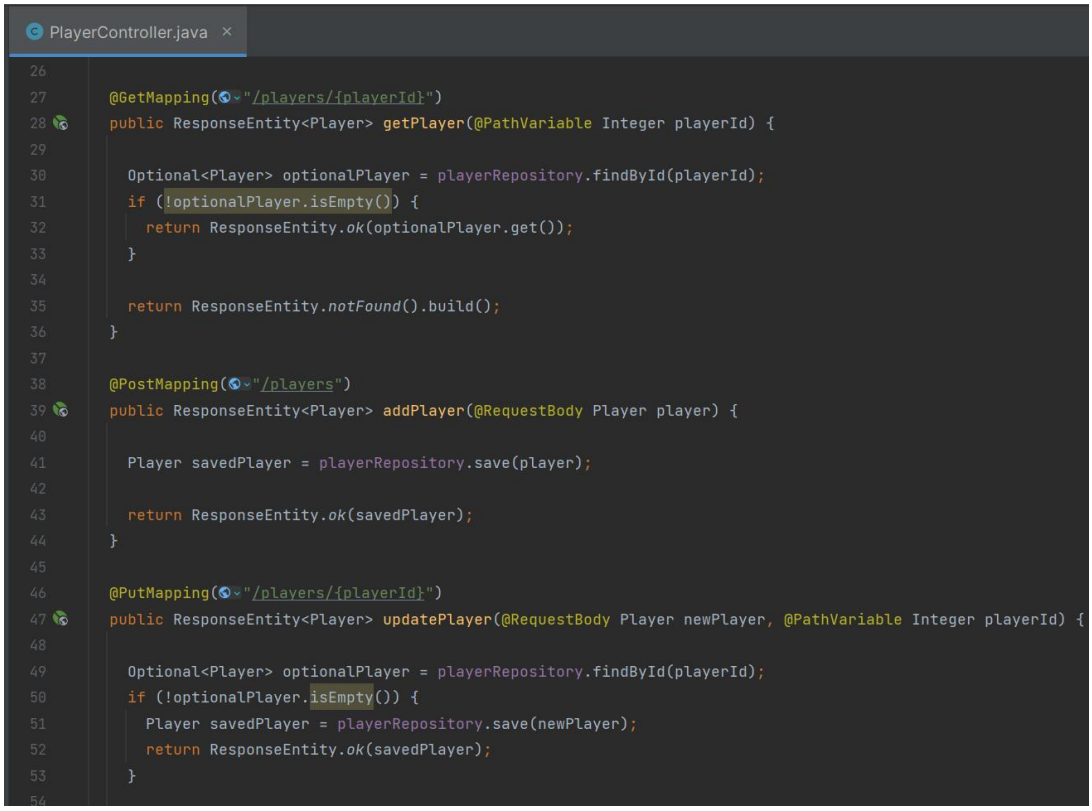
Sl. 4.17. Kôd *User Repository*.

4.4.5. Kontroleri

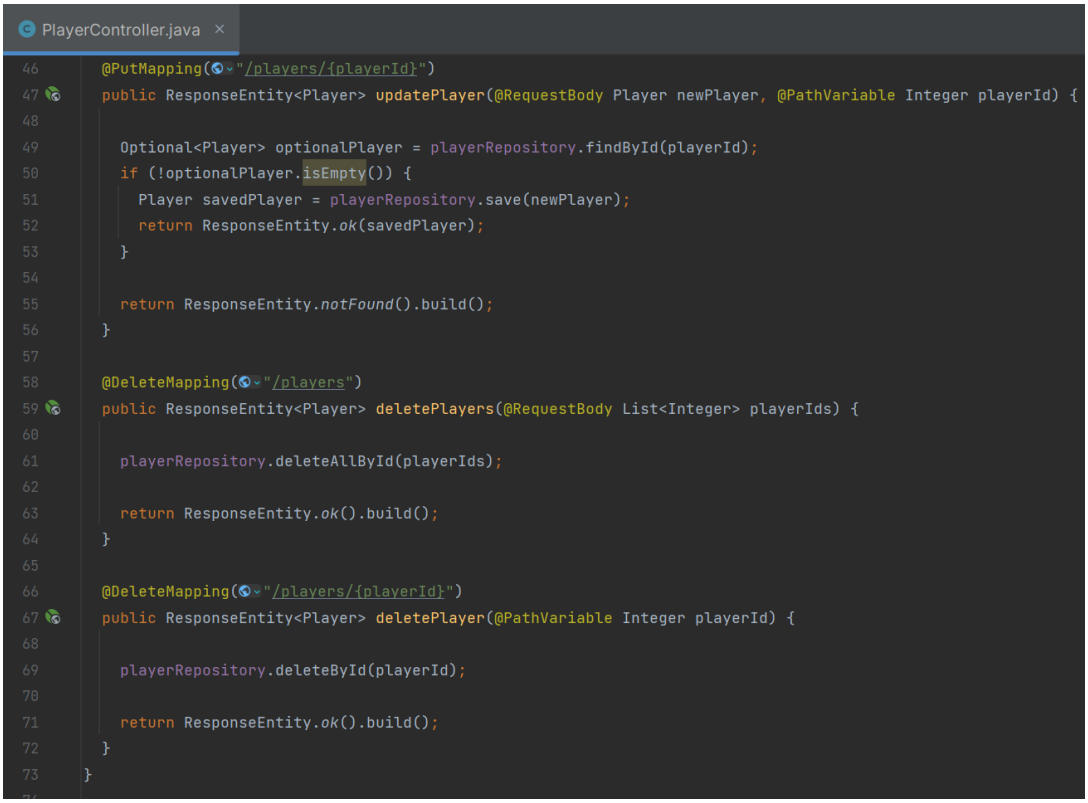
Kontroleri su *Java* klase koje se koriste za upravljanje *HTTP* zahtjevima i definiranju krajnjih točaka *API*-ja. U *web* aplikaciji za upravljanje nogometnim klubom se koriste dva kontrolera, a to su *Player Controller* i *User Controller*. *Player Controller* je zaslužan za upravljanje dohvaćanje svih igrača, dodavanje novog igrača, ažuriranje igrača i njihovih podataka te brisanje postojećih igrača. U metodama kontrolera koriste se anotacije poput: *@GetMapping*, *@PostMapping*, *@PutMapping i @DeleteMapping*, koje označavaju i definiraju koja vrsta *HTTP* zahtjeva se obrađuje tom metodom što je vidljivo na slikama 4.18- 4.20 [15].



Sl. 4.18. Prvi dio kôd-a *Player Contoller*-a.



Sl. 4.19. Drugi dio kôd-a *Player Contoller*-a.



Sl. 4.20. Treći dio kôd-a *Player Contoller*-a.

User Controller je zaslužan za upravljanje *HTTP* zahtjevima vezanim uz korisnike aplikacije. Omogućuje funkcionalnosti poput registracije novih korisnika, dohvaćanje podataka o korisnicima te provjeru korisničkih podataka (Slika 4.21).

Sl. 4.21.Kôd *User Contoller*.

4.4.6. Inicijalizacija podataka

Podaci poslužiteljske strane se inicijaliziraju pomoću datoteka „*data.sql*“ i „*schema.sql*“. „Schema.sql“ je dokument zaslužan za definiranje strukture tablica korisnika i igrača u bazi podataka (Slika 4.22), dok „*data.sql*“ inicijalizira početne podatke unutar tablica u bazi podataka (Slika 4.23).

```sql
insert into app_user (name, last_name, email, password)
values ('Name', 'Last name', 'email@address.com', 'password');
insert into app_user (name, last_name, email, password)
values ('Name2', 'Last name2', 'email2@address.com', 'password2');


insert into player (name, last_name, position, date_of_birth, contract_date, club_name, nationality, height,
                    preferred_foot, shirt_number, total_played, game_started, minutes_per_game, team_of_week, goals,
                    expected_goals, scoring_frequency, goals_per_game, shots_per_game, shots_on_target_per_game,
                    big_chances_missed, first_strength, second_strength, third_strength, first_weakness,
                    second_weakness, third_weakness, total_passes, key_passes, successful_passes, interceptions,
                    tackles, ball_recovered, ball_won, red_card, yellow_card)
values ('John', 'Johnson', 'Midfielder', '1984-06-05', '2027-02-11', 'NK Osijek', 'CRO', 123, 'Left', 50, 452, 401,
        79.8, 7, 32, 38, 14, 2, 7.1, 4.6, 16, 'strength_1', 'strength_2', 'strength_3', 'weakness_1', 'weakness_2',
        'weakness_3', 148, 109, 79, 19, 10, 14, 15, 1, 7);
insert into player (name, last_name, position, date_of_birth, contract_date, club_name, nationality, height,
                    preferred_foot, shirt_number, total_played, game_started, minutes_per_game, team_of_week, goals,
                    expected_goals, scoring_frequency, goals_per_game, shots_per_game, shots_on_target_per_game,
                    big_chances_missed, first_strength, second_strength, third_strength, first_weakness,
                    second_weakness, third_weakness, total_passes, key_passes, successful_passes, interceptions,
                    tackles, ball_recovered, ball_won, red_card, yellow_card)
values ('Muhi', 'Smith', 'Goalkeeper', '1992-03-02', '2030-06-11', 'NK Osijek', 'CRO', 456, 'Left', 50, 452, 401, 79.8,
        7, 32, 38, 14, 2, 7.1, 4.6, 16, 'strength_1', 'strength_2', 'strength_3', 'weakness_1', 'weakness_2',
        'weakness_3', 148, 109, 79, 19, 10, 14, 15, 1, 7);
```

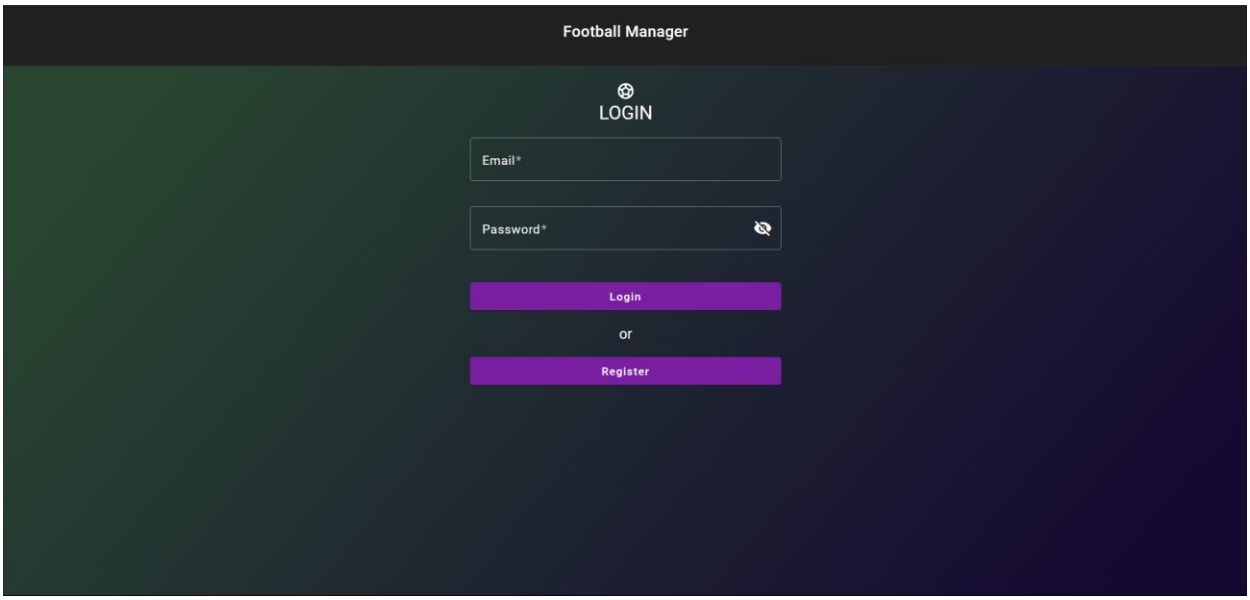
Sl. 4.22.Kôd *data.sql*.

```sql
create table app_user
(
    id        integer auto_increment,
    name      varchar(255) not null,
    last_name varchar(255) not null,
    email     varchar(255) not null,
    password  varchar(255) not null,
    primary key (id)
);

create table player
(
    id                    integer auto_increment,
    name                  varchar(255) not null,
    last_name             varchar(255) not null,
    position              varchar(255) not null,
    date_of_birth         date         not null,
    contract_date         date,
    club_name             varchar(255),
    nationality           varchar(255),
    height                integer,
    preferred_foot        varchar(255),
    shirt_number          integer,
    total_played          integer,
    game_started          integer,
    minutes_per_game      double precision,
    team_of_week          integer,
    goals                 integer,
    expected_goals        integer,
    scoring_frequency     integer
```

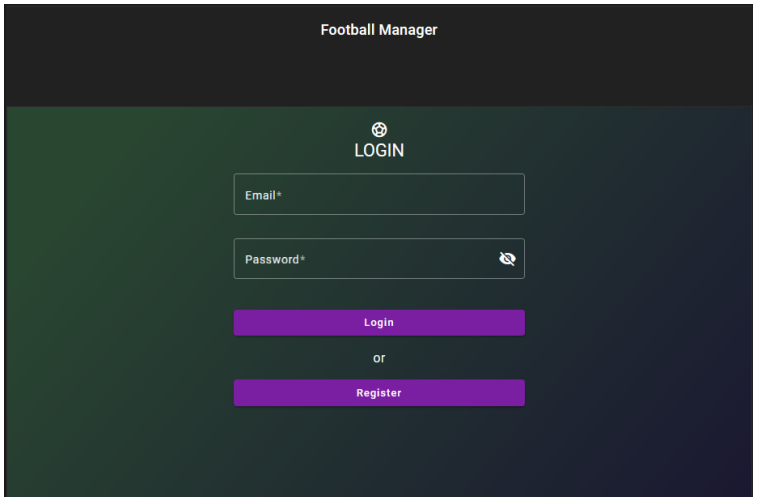
Sl. 4.23.Kôd *schema.sql*.

5. TESTIRANJE APLIKACIJE

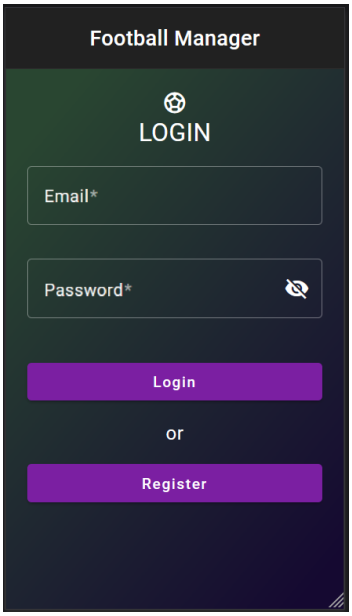
U ovom poglavlju predstavljena je funkcionalnost, *web* dizajn, logičan slijed protoka aplikacije ovog projekta te prikaz na tri različite veličine zaslona gdje se prikazuje svojstvo responzivnosti aplikacije. Ulaskom u *web* aplikaciju prikazuje se *login* komponenta gdje se postojeći korisnik može prijaviti na *web* stranicu ili može odabrati opciju *register*, ukoliko nije postojeći korisnik *web* stranice, gdje se može registrirati i spremiti u bazu podataka korisnika aplikacije što je vidljivo na slikama 5.1 - 5.7.



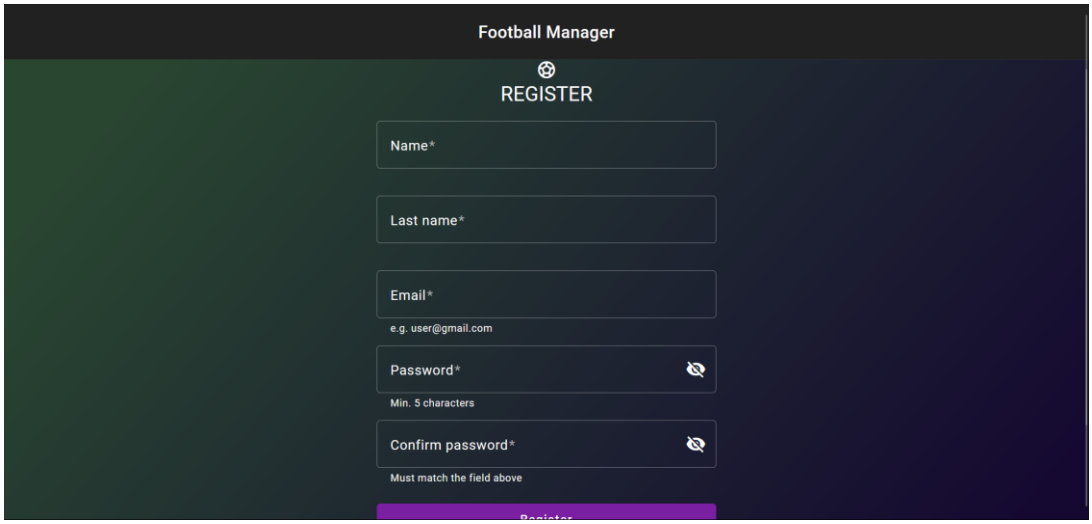
Sl. 5.1. *Login* komponenta, *PC* zaslon.



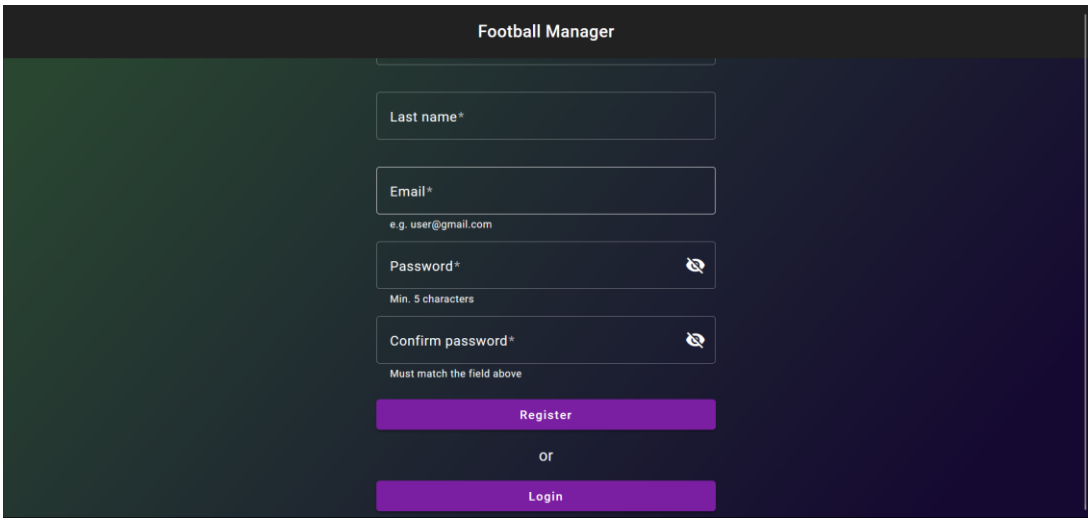
Sl. 5.2. *Login* komponenta, *iPad PRO* 12,9 inča zaslon.



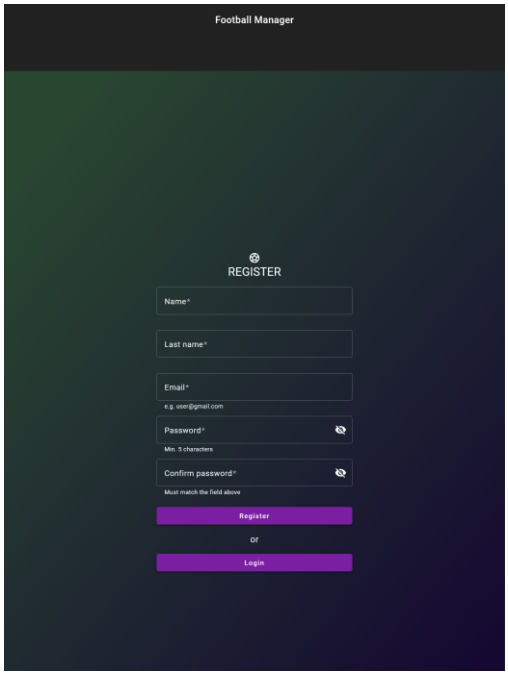
Sl. 5.3. *Login* komponenta, *iPhone 5* zaslon.



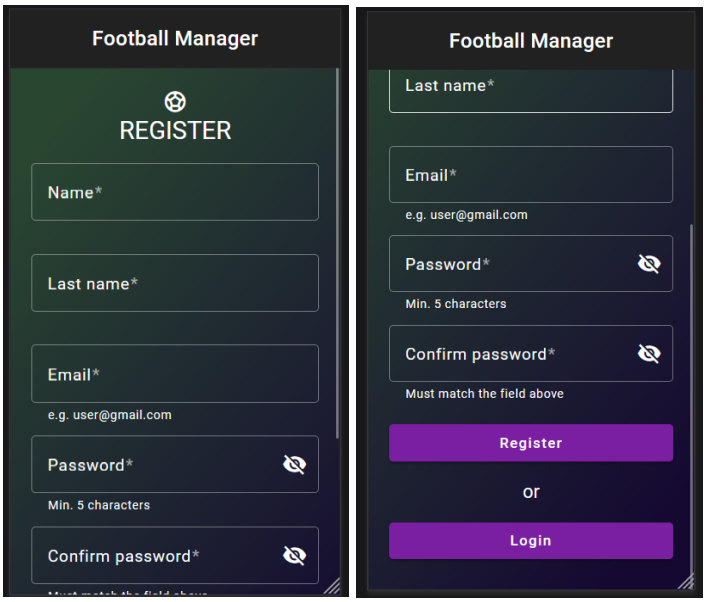
Sl. 5.4. *Register* komponenta, *PC* zaslon jedan.



Sl. 5.5. *Register* komponenta, *PC* zaslon dva.



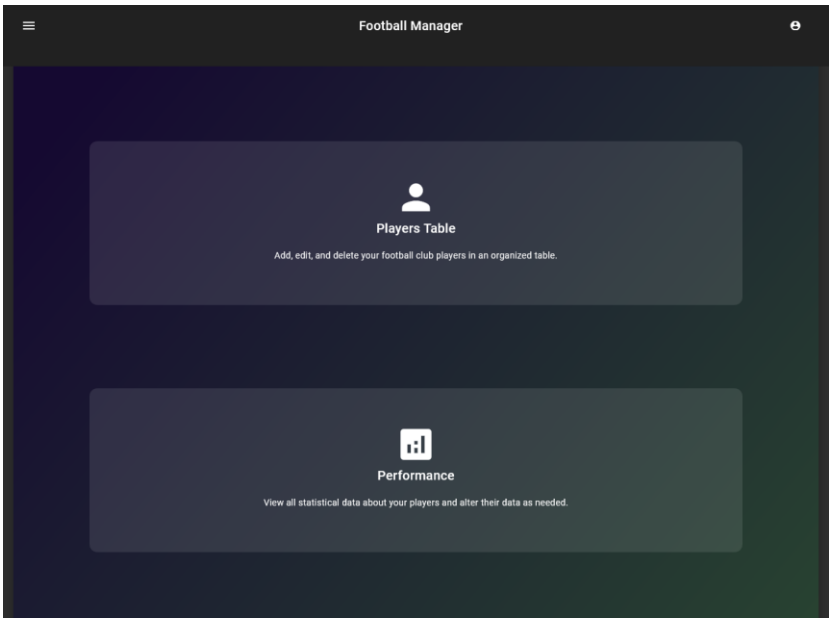
Sl. 5.6. *Register* komponenta, *iPad PRO* 12,9 inča zaslon.



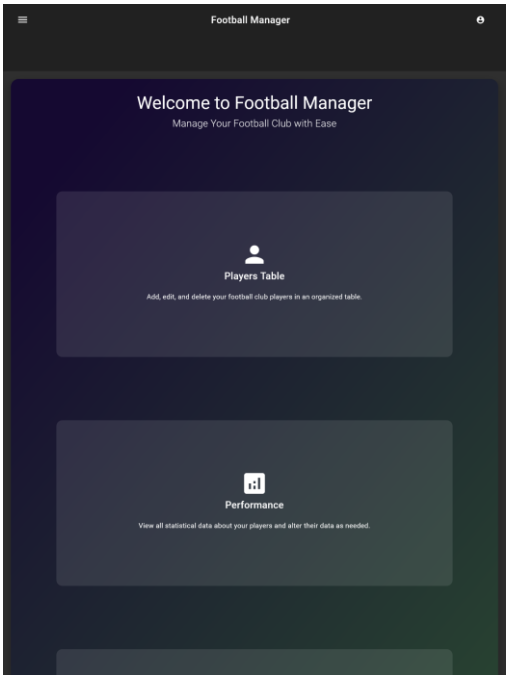
Sl. 5.7. *Register* komponenta, *iPhone 5* zaslon.

Nakon uspješne prijave u aplikaciju, *Angular Router* korisnika upućuje na *Home Page* komponentu prikazanu na slikama 5.1-5.2. Na *Home Page* komponenti nalaze se opisi glavnih funkcija aplikacije gdje se *Players Table* opis može kliknuti i korisnik je tada upućen na *Players Table* komponentu. Također može se primijetiti da se na gornjoj navigacijskoj traci nalazi gumb za padajući izbornik koji otvara *Sidebar Navigation* komponentu prikazanu na slikama 5.2i 5.11,

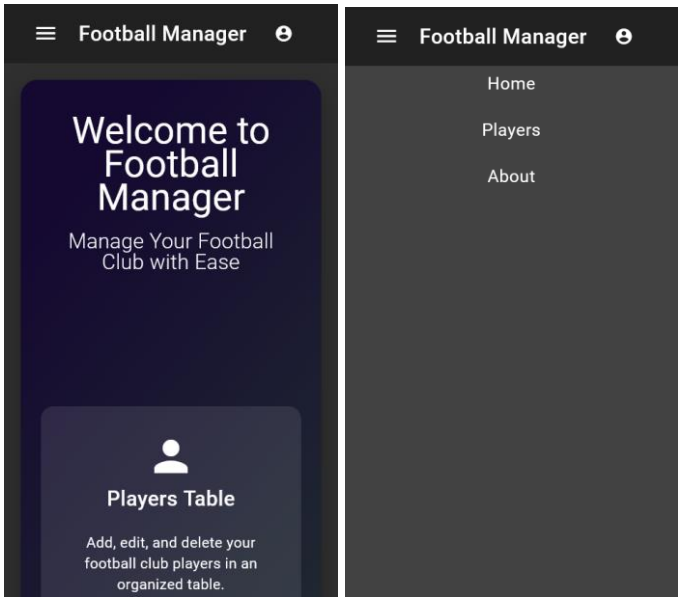
koja se povlači s lijeve strane, a u desnom kutu navigacijske trake nalazi se ikona pomoću koje se korisnik može odjaviti iz aplikacije.



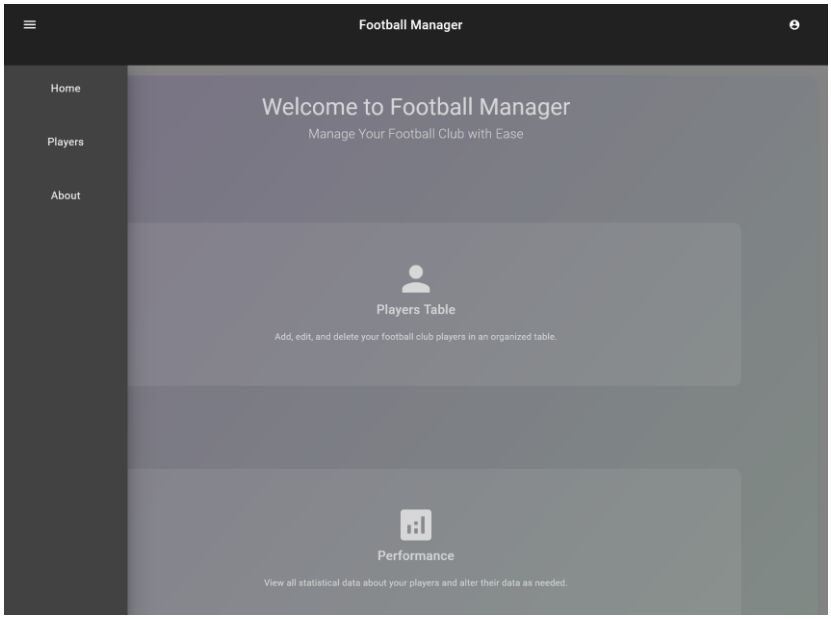
Sl. 5.1. *Home Page* komponenta, *PC*.



Sl. 5.9. *Home Page* komponenta, *iPad PRO* 12,9 inča zaslon.

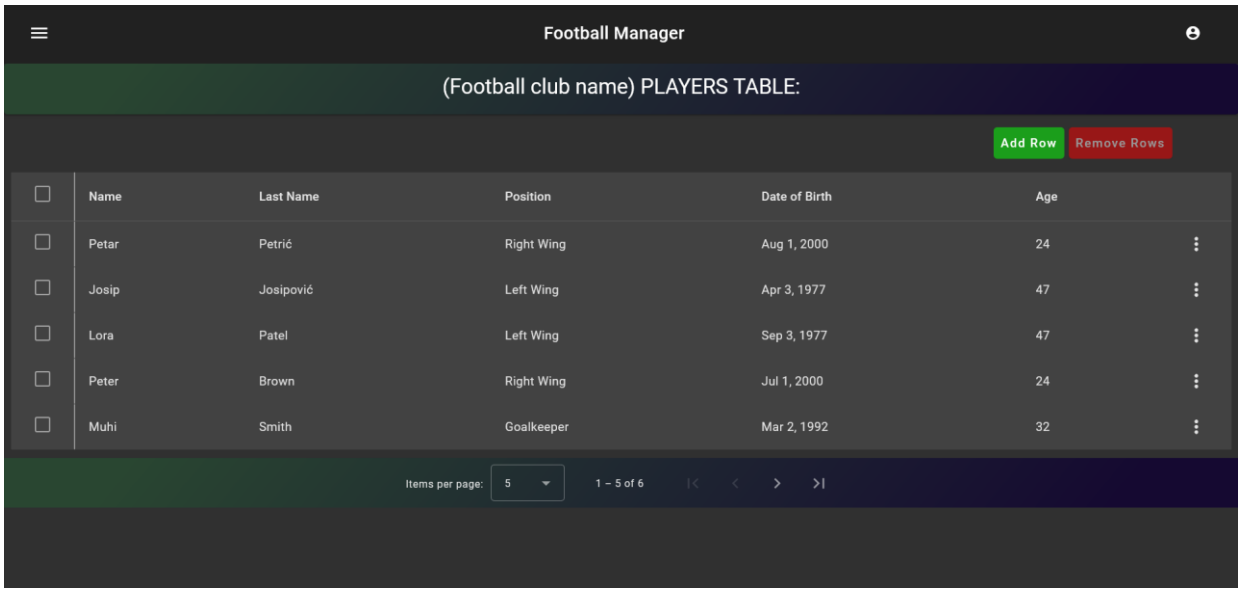


Sl. 5.2. *Home Page* i *Sidebar Navigation* komponente, *iPhone 5* zaslon.

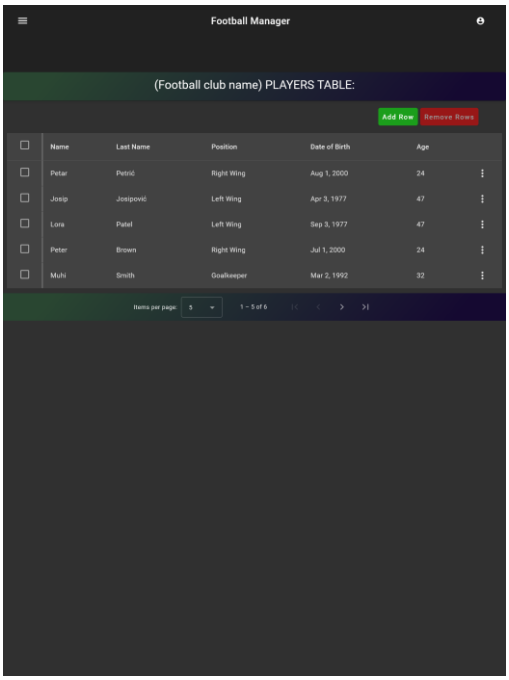


Sl. 5.11. *Sidebar Naviagtion* komponenta, *PC*.

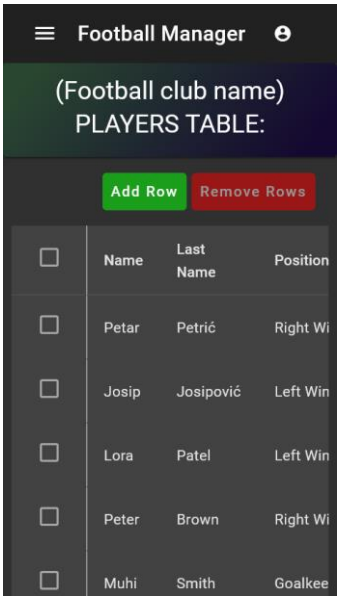
Nakon *Home Page* komponente logičan slijed je da korisnik nastavi na *Players Table* komponentu prikazanu na slikama 5.3 – 5.5. *Players Table* komponenta ima razne mogućnosti od kojih su neke: dodavanje i brisanje igrača, uređivanje informacija postojećih igrača, odabir jednog ili više igrača odjednom te navigiranje stranicama tablice putem *mat-pagination* komponente u podnožju tablice.



Sl. 5.3. *Players Table* komponenta, *PC* zaslon.

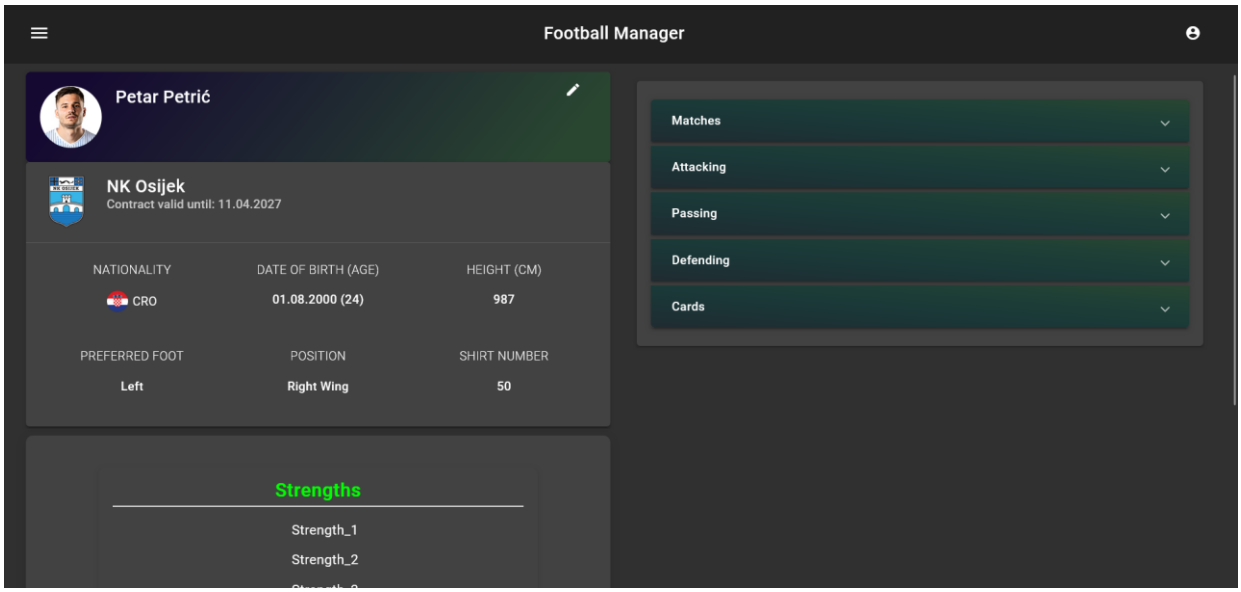


Sl. 5.4. *Players Table* komponenta, *iPad PRO* 12,9 inča zaslon.

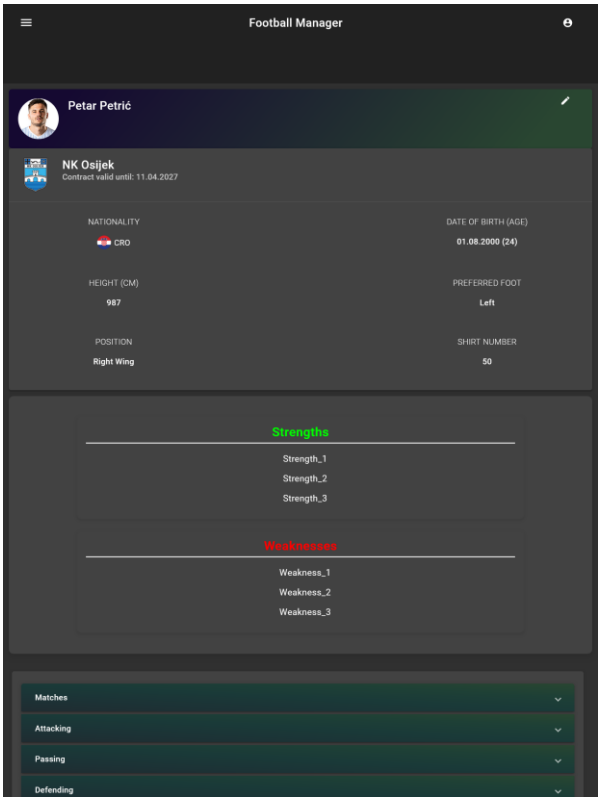


Sl. 5.5. *Players Table* komponenta, *iPhone 5* zaslon.

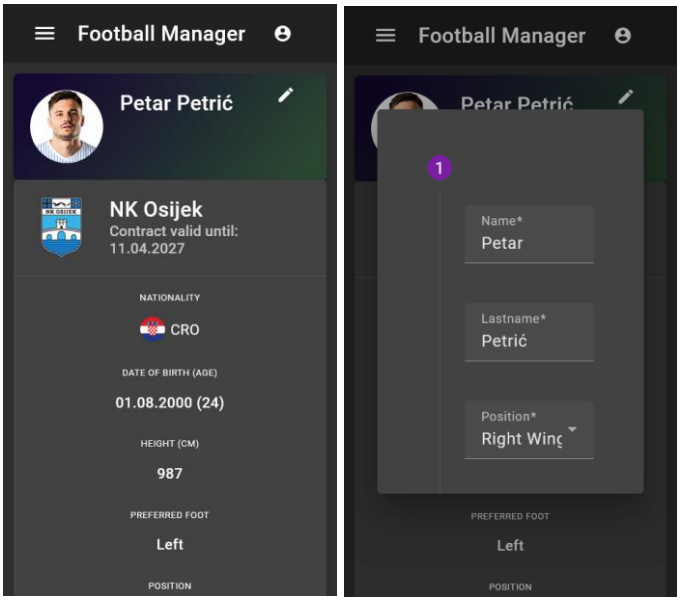
Nakon što korisnik odabere jednog igrača, može kliknuti na *details* gumb u tablici koji se nalazi unutar gumba s tri okomite točke te ga taj gumb vodi na *Player Performance* komponentu koja prikazuje sve detaljne informacije o odabranom nogometnom igraču (Slike 5.6 – 5.17).



Sl. 5.6. *Player Performance* komponenta, *PC* zaslon.

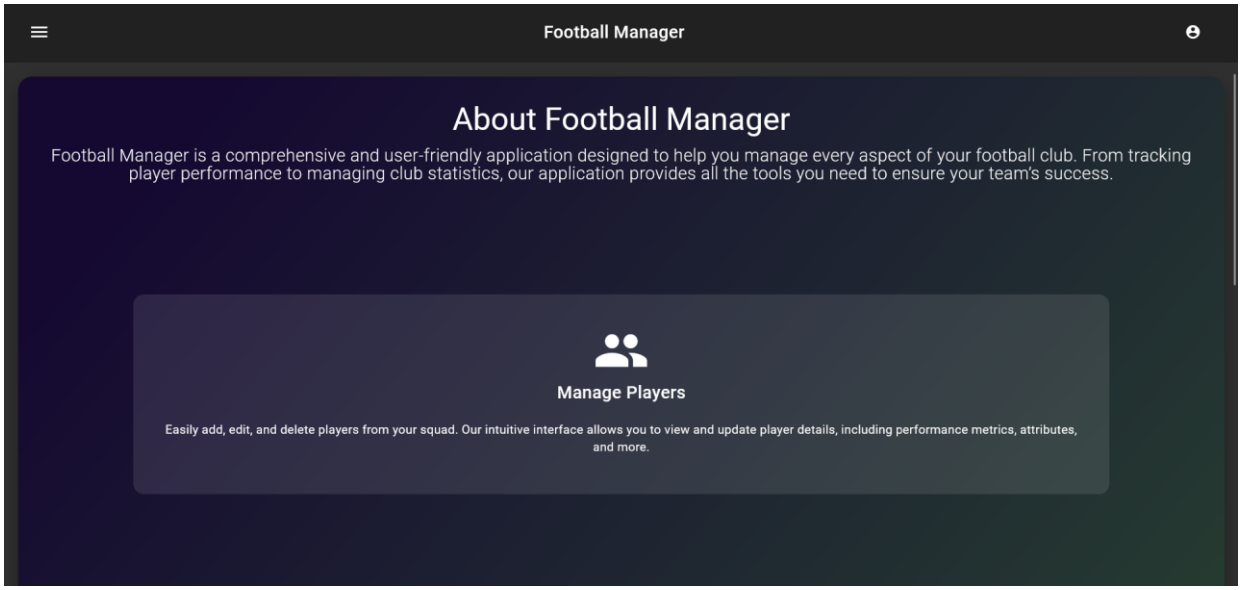


Sl. 5.16. *Player Performance* komponenta, *iPad PRO* 12,9 inča zaslon.

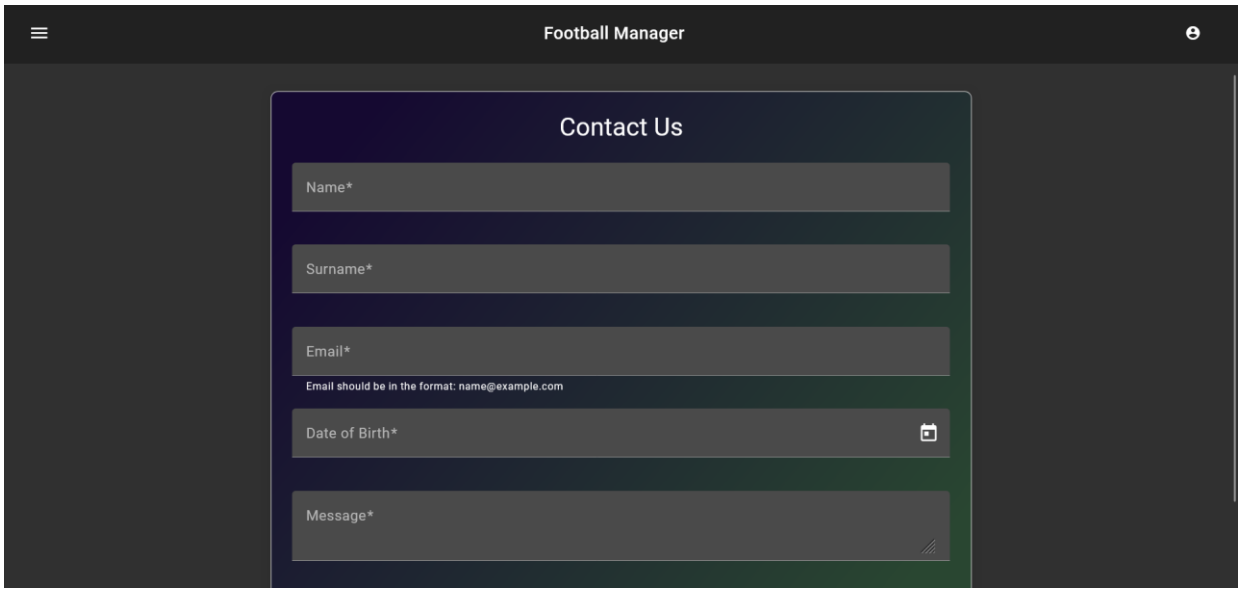


Sl. 5.17. *Player Performance* komponenta, *iPhone 5* zaslon.

Korisnik u bilo kojem trenutku, pod uvjetom da je prijavljen na aplikaciju, može pomoću poveznice iz *Sidebar Navigation* komponente otići na *About* (Slika 5.18) i *Contact Us* (Slika 5.19) komponente.



Sl. 5.18. *About* komponenta, *PC* zaslon.



Sl. 5.19. *Contact Us* komponenta, *PC* zaslon.

6. ZAKLJUČAK

Zadatak završnog rada bio je razviti *web* aplikaciju za upravljanje nogometnim klubom. Sastavljena od korisničkog sučelja napravljenog s *Angularom* te posužiteljske strane napravljene uz pomoć *Spring Boota*, aplikacija je rezultat znanja stečenog na fakultetu i kroz praksu. Aplikacija nudi značajne funkcionalnosti kao što su registracija i prijava korisnika, upravljanje podacima igrača nogometnog kluba, komunikaciju između klijentske i poslužiteljske strane te responzivan dizajn. U usporedbi s ostalim *web* aplikacijama i rješenjima ova aplikacija nema određeni broj funkcionalnosti kao što su na primjer službeni kalendar nogometnog kluba, službeni *chat* te integracija sa ostalim platformama.

Jedan od fokusa aplikacije bio je responzivan dizajn koji je uspješno ostvaren upotrebljavajući komponente i alate iz *Angular Material* biblioteke te *SCSS*. Alati korišteni u izradi *web* aplikacije zaslužni su za njen responzivan dizajn, modularnu strukturu i jednostavno korištenje aplikacije. Prednost *Angular* aplikacije je njena modularna struktura koja omogućuje daljnji razvoj na veću razinu.

Završetkom razvoja aplikacije identificirane su prilike za daljnji razvoj i područja koja se mogu poboljšati. Primjeri prilike za poboljšanje su: detaljniji grafički prikaz statističkih podataka o performansama pojedinog igrača, brojnije i strože validacije unosa podataka, korištenje vanjske baze podataka te dodavanje strožih mjera za registraciju i prijavu korisnika. Izradom ove aplikacije prikupljeno je vrijedno iskustvo u razvoju *web* aplikacija upotrebom modernih tehnologija. Unatoč uspješnom razvoju *web* aplikacije, identificirane su prednosti, mane te prilike za daljnji napredak ove aplikacije.

LITERATURA

[1] TeamLinkt, *Football Management Software* [online], TeamLinkt, 2024, dostupno na: https://teamlinkt.com/sports/football [11.09.2024.]

[2] Director11, *director11 - Football club management software revolution* [online], Director11, 2024, dostupno na: https://www.director11.com/en/home/ [15.09.2024.]

[3] ThemeForest, *Vensica - Football Club Manager Elementor Theme Preview* [online], ThemeForest, 2024, dostupno na: https://preview.themeforest.net/item/vensica-football-club-manager-elementor-theme/full_screen_preview/44324910 [11.09.2024.]

[4] Hudl, *Wyscout - The world's biggest library of football* [online], Hudl, 2024, dostupno na: https://www.hudl.com/products/wyscout [11.09.2024.]

[5] Amphibee, *SportEasy: The #1 App for Managing Your Sports Team or Club* [online], Amphibee, 2024, dostupno na: https://www.sporteasy.net/ [11.09.2024.]

[6] Cambridge Dictionary, *back end* [online], Cambridge University Press, 2024, dostupno na: https://dictionary.cambridge.org/dictionary/english/back-end [11.09.2024.]

[7] Angular, *What is Angular?* [online], Google LLC, 2024, dostupno na: https://v15.angular.io/guide/what-is-angular [11.09.2024.]

[8] MDN Web Docs, *Introduction to the DOM - Web APIs* [online], Mozilla Foundation, 2024, dostupno na: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction [11.09.2024.]

[9] TypeScript, *Documentation - TypeScript for the New Programmer* [online], Microsoft, 2024, dostupno na: https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html [11.09.2024.]

[10] GeeksforGeeks, *Introduction to Spring Boot* [online], GeeksforGeeks, 2024, dostupno na: https://www.geeksforgeeks.org/introduction-to-spring-boot/ [11.09.2024.]

[11] Spring, *Spring Boot* [online], Pivotal Software, 2024, dostupno na: https://spring.io/projects/spring-boot [11.09.2024.]

[12] H2 Database, *H2 Database Engine* [online], H2 Database, 2024, dostupno na: https://www.h2database.com/html/main.html [11.09.2024.]

[13] IBM, *What is a REST API?* [online], IBM, 2024, dostupno na: https://www.ibm.com/topics/rest-apis [12.09.2024.]

[14] REST API Tutorial, *What is REST?* [online], REST API Tutorial, 2024, dostupno na: https://restfulapi.net/ [12.09.2024.]

[15] Symflower, *What is the difference between @Controller vs @RestController in Spring Boot?* [online], Symflower, 2024, dostupno na: https://symflower.com/en/company/blog/2024/controller-restcontroller-spring-boot/ [14.09.2024.]

POPIS KRATICA

HTML = prezentacijski jezik za izradu web stranica (engl. *HyperText Markup Language*)

HTTP = glavna metoda prijenosa informacija na internetu (engl. *Hyper Text Transfer Protocol*)

CSS = stilski jezik, koji se rabi za opis prezentacije dokumenta (engl. *Cascading Style Sheets*)

SCSS = CSS pretprocesor koji dodaje dodatnu funkcionalnost i sintaksu CSS jeziku (akronim *Sassy CSS*)

REST API = arhitektonski stil koji definira skup principa za izgradnju web usluga (engl. *Representational State Transfer Application Programming Interface*)

DOM = predstavlja strukturu web stranice (engl. *Document Object Model*)

CLI = sučelje komandne linije (engl. *Command-line interface*)

SAŽETAK

U ovom završnom radu, napravljena je *web* aplikacija za upravljanje nogometnim klubom. Cilj ove *web* aplikacije je omogućiti korisniku aplikacije upravljanje nogometnim klubom time što ima mogućnost pregleda igrača nogometnog kluba, dodavanje i brisanje igrača, pregleda statistike pojedinačnog igrača te sve to uz optimalan responzivan dizajn. Teorijski dio završnog rada se temelji na objašnjenju svih tehnologija i programskih jezika koji su bili korišteni u izradi. U praktičnom dijelu je prikazan i opisan izgled te funkcionalnost web aplikacije. Za izradu aplikacije upotrijebljeni su: *Angular*, *Angular Material*, *Spring Boot*, *TypeScript*, *SCSS*, *HTML* te razvojno okruženje *IntelliJ IDEA*.

Ključne riječi: *Angular*, *Angular Material,* baza podataka, korisničko sučelje, responzivni dizajn

ABSTRACT

Title: Web application for football club management.

In this final paper, a web application for managing a football club was developed. The goal of this web application is to allow the user to manage the football club by providing the ability to view players of the club, add and delete players, and review individual player statistics, all with an optimal responsive design. The theoretical part of the final paper is based on explaining all the technologies and programming languages used in the development. The practical part presents and describes the appearance and functionality of the web application. The technologies used for developing the application include Angular, Angular Material, Spring Boot, TypeScript, SCSS, HTML, and the integrated development environment called IntelliJ IDEA.

Keywords: *Angular*, *Angular Material*, database, frontend, responsive design