Izrada i testiranje Angular aplikacije za nogometnu statistiku

Faletar, Josip

Master's thesis / Diplomski rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:064142

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-11-27**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

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

Sveučilišni studij

IZRADA I TESTIRANJE ANGULAR APLIKACIJE ZA NOGOMETNU STATISTIKU

Diplomski rad

Josip Faletar

Osijek, 2020.

SADRŽAJ

1. UVOD 1
1.1. Zadatak diplomskog rada 1
2. VRSTE TESTIRANJA 2
3. TEHNOLOGIJE I ALATI KORIŠTENI PRI IZRADI APLIKACIJE 4
3.1. Visual Studio Code 4
3.2. Angular 5
3.3. TypeScript 5
3.4. HTML 5
3.5. CSS 6
3.6. Firebase 6
3.7. NodeJS 7
4. IZRADA APLIKACIJE 8
4.1. Programski dio aplikacije 8
5. KORISNIČKO SUČELJE APLIKACIJE 15
5.1. Dodavanje igrača 15
5.2. Prikaz statistike igrača 18
5.3. Poredak 20
5.4. Utakmice 21
5.5. Ekipe 22
6. TESTIRANJE APLIKACIJE 27
6.1. Postavke testiranja 27
6.2. Testiranje jedinice 27
6.3. Cjelovito testiranje 30
7. ZAKLJUČAK 34
LITERATURA 35
SAŽETAK 36

ABSTRACT 37
ŽIVOTOPIS 38

1. UVOD

Korištenje modernih tehnologija u današnjem svijetu olakšava čovjekovu svakodnevicu i obavljanje određenih poslova. Razvoj tehnologije ima veliki utjecaj na nogometni svijet i na njegove djelatnike. Problem ovog diplomskog rada je omogućiti nogometnim trenerima praćenje određenih parametara i podataka vezanih za nogometaše pomoću aplikacije. Upravo web aplikacija za nogometnu statistiku bi riješila problem diplomskog rada jer bi olakšala trenerima praćenje statistike. Pomoću aplikacije trenerima bi bilo omogućeno praćenje rezultata utakmica, stanja na ljestvici, popis igrača, unos igrača te detaljan prikaz statistike za svakog igrača.

Ovaj rad se temelji na izradi i testiranju web aplikacije. Mogućnosti i korištenje aplikacije su objašnjeni u nekoliko poglavlja. U drugom poglavlju će biti objašnjena važnost testiranja aplikacije te vrste testiranja Angular aplikacija dok će u trećem poglavlju će biti opisani programski jezici i alati koji su korišteni pri izradi web aplikacije. Programski dio aplikacije će biti objašnjen u četvrtom poglavlju, a način na koji se aplikacija za nogometnu statistiku koristi te koje su sve njene mogućnosti će biti objašnjeni u petom poglavlju. Aplikacija će se testirati pomoću testova koji će biti objašnjeni i čiji će rezultati biti interpretirani u šestom poglavlju.

1.1. Zadatak diplomskog rada

Zadatak diplomskog rada je napraviti i testirati web aplikaciju koja će omogućiti trenerima praćenje nogometne statistike. U radu treba objasniti programske alate koji su se koristili pri izradi aplikacije te pojasniti na koji način aplikacija radi. Potrebno je testirati aplikaciju te pokazati i interpretirati dobivene rezultate.

2. VRSTE TESTIRANJA

Uz izradu aplikacije glavni dio ovog rada je testiranje aplikacije. Testiranje Angular aplikacije je važno kako bi se spriječila oštećenja odnosno greške u aplikaciji. Glavni razlog zbog čega se aplikacija testira je taj da se provjeri da li se aplikacija ponaša onako kako se očekuje, da li je funkcionalna te da li ispravno radi. Postoje različite vrste Angular testova kojima se aplikacija može testirati, a neki od njih su:

- Testiranje jedinice (eng. *Unit test*)
- Cjelovito testiranje (eng. *End-to-End test*)
- Integracijsko testiranje (eng. *Integration test*)
- Regresijski test (eng. *Regression test*)
- Test performansi (eng. *Performance test*)
- Test opterećenja (eng. *Load test*)
- Korisnički test (eng. *User acceptance test*)

U nastavku će biti objašnjeni gore navedeni testovi. Najkorišteniji testovi za provjeru ispravnosti rada aplikacije su: : testiranje jedinice, integracijsko testiranje i cjelovito testiranje. Testiranje jedinice je testiranje koje se temelji na testiranju samo jednog dijela aplikacije odnosno testiranje klase ili metode. Cilj ovog načina testiranja je izolirati dio aplikacije te ga zasebno testirati.

Integracijski test je način testiranja u kojem se provjerava da li više dijelova aplikacije međusobno dobro rade odnosno da li se upotpunjuju. Prednost integracijskog testa je da se njime može provjeriti komunikacija među pojedinim dijelovima aplikacije. Integracijski test se najčešće koristi nakon što se aplikacija testira testom jedinice te prije cjelovitog testiranja.

Treći najčešće korišteni test za provjeru funkcionalnosti aplikacije je cjelovito testiranje. Cjelovito testiranje je testiranje koje omogućuje provjeru da li aplikacija radi ispravno od početka do kraja. Ovaj način testiranja se temelji na simulaciji stvarnog korisnika u pregledniku.

Regresijsko testiranje je vrsta testiranja aplikacije koja služi kako bi se provjerilo da li nedavna promjena koja je unesena u program aplikacije utječe negativno na rad same aplikacije. Regresijski test omogućuje provjeru da li dijelovi aplikacije i dalje dobro funkcioniraju međusobno nakon izmjene koda. Sljedeća vrsta testiranja aplikacije je test performansi.

Test performansi omogućuje provjeru brzine, opterećenja te skalabilnosti neke aplikacije. Ova vrsta testiranja je korisna jer se njome može ispitati učinkovitost pojedinih dijelova aplikacije odnosno koliko brzo se izvršavaju pojedini dijelovi aplikacije.

Test opterećenja omogućuje provjeru brzine aplikacije odnosno njenog opterećenja kada se njome služi više korisnika istovremeno. Glavni zadatak ove vrste testiranja je simulirati veći broj korisnika i određenih aktivnosti kako bi se provjerila izdržljivost aplikacije. Razlika između testa performansi i testa opterećenja je taj što se test performansi temelji na provjeri brzine samo jednog dijela aplikacije, dok s druge strane test opterećenja kao što je ranije rečeno se temelji na brzini aplikacije kada se njome koristi više korisnika istovremeno.

Korisnički test je test kojim se provjerava da li aplikacija ispunjava sve korisnikove zahtjeve. Glavni zadatak korisničkog testa je da ju obavlja krajnji korisnik kako bi se provjerio rad i ispravnost aplikacije prije njenog objavljivanja. Korisnički test se izvodi nakon provođenja integracijskog testiranja, testiranja jedinica te cjelovitog testiranja.

3. TEHNOLOGIJE I ALATI KORIŠTENI PRI IZRADI APLIKACIJE

Web aplikacija za nogometnu statistiku izrađena je u razvojnom sučelju Microsoft Visual Studio Code, dok je sučelje (eng. *front-end*) izrađeno u razvojnom okviru Angular koji se koristi za razvoj aplikacija. Programski kod aplikacije je pisan u programskom jeziku TypeScript.

Za bazu podataka je korištena Google-ova platforma Firebase koja predstavlja bazu podataka u stvarnom vremenu. Za izradu aplikacije su korišteni HTML i CSS. HTML je prezentacijski jezik koji se koristi za stvaranje hipertekstualnih datoteka, dok je CSS stilski jezik kojim se definira kako će izgledati HTML element. Pri izradi aplikacije korišten je i NodeJS koji predstavlja pozadinsku platformu koja je temeljena na programskom jeziku JavaScript.

3.1. Visual Studio Code

Microsoft Visual Studio Code je razvojno okruženje koje je razvio Microsoft. Visual studio Code služi za izradu aplikacija, te je on dizajniran za jednostavniji rad odnosno izradu web aplikacija.

Prema [1], neke od važnijih značajki Visual Studio Code-a su:

- Uređivač izvornog koda (eng. *Editor*) – označavanje sintaksi, uklanjanje grešaka, dovršavanje koda, brzina, jednostavnost.
- Pronalaženje pogrešaka u kodu pomoću programa (eng. *Debugger*).
- Otvorenog koda (eng. *Open Source*).
- Podržava većinu programskih jezika.
- IDE (eng. *Integrated development environment*) – omogućuje inteligentni pristup pisanja kodu, organizaciju koda.
- Git podrška - povezanost s Git-om.
- Terminal – omogućuje korisniku jednostavnije korištenje unutar Visual Studio Code-a.

Visual Studio Code je iznimno jednostavan za korištenje te je vrlo praktičan za izradu web aplikacija posebno za programske jezike TypeScript i C#.

3.2. Angular

Angular je razvojni okvir koji je napravljen od strane Google-a te služi za razvoj aplikacija. Angular je napisan u programskom jeziku TypeScript te je otvorenog koda (eng. *Open Source*) [2].

Glavni zadatak Angular-a je razvoj dinamičnih web aplikacija, ali osim razvoja web aplikacija, može se koristiti i za izradu mobilnih aplikacija te za razvoj aplikacija namijenjenih za radnu površinu računala. Prednost Angulara je i jednostavno održavanje aplikacija te njihovo testiranje.

3.3. TypeScript

TypeScript je programski jezik u kojem je napisan razvojni okvir Angular. Razvijen je od strane Microsoft-a te je njegova glavna zamisao da nadomjesti nedostatke JavaScript-a. TypeScript funkcionira tako da se programski kod prevodi (eng. *compile*) u JavaScript kod.

Razlog zašto se TypeScript kod prevodi u JavaScript je taj što TypeScript nema mogućnost da ga web preglednici interpretiraju. Prednosti TypeScripta su lakše čitanje koda, sprječavanje problema te stvaranje objekata temeljenih na klasama. Prema [3], glavna razlika između JavaScript i TypeScript jezika je taj što je JavaScript na strani klijenta dok je TypeScript objektno orijentirani jezik koji se temelji na klasama.

3.4. HTML

HTML je prezentacijski jezik čija kratica znači HyperText Markup Language. Služi prvenstveno za izradu web stranica te za stvaranje hipertekstualnih datoteka. HTML je jezik koji je rasprostranjen diljem svijeta zbog svoje jednostavnosti te zbog toga što je besplatan i lako dostupan svima. Sastoji se od poveznica i oznaka kojima se definira izgled web stranice odnosno web aplikacije. Prema [4] , neke od prednosti korištenja HTML-a su :

- Jednostavan za korištenje
- Jednostavna sintaksa
- Omogućuje upotrebu predložaka
- Podržan od gotovo svih web preglednika

3.5. CSS

CSS (Cascading Style Sheets) je stilski jezik koji služi za uređivanje HTML elemenata. CSS-om se definira na koji način će se HTML element prikazati. Prednosti CSS-a su lakše, brže i jednostavnije uređivanje web stranica [5].

Zahvaljujući CSS-u HTML je postao jednostavniji i pregledniji i manji što omogućuje jednostavniju izradu aplikacija odnosno stranica. Razlog tomu je što se CSS kod sprema u posebne datoteke izvan HTML dokumenata te više nije potrebno definirati stilove unutar HTML dokumenta.

3.6. Firebase

Firebase je platforma koja je razvijena 2011. godine, koristi se za razvoj mobilnih i web aplikacija. Ova razvojna platforma omogućuje lakšu i bržu izradu kvalitetnih aplikacija.

Prema [6], nekoliko značajki koje omogućuje platforma Firebase su:

- Google Analitika
- Baza podataka u stvarnom vremenu (eng. *Realtime Database)*
- Autentifikacija korisnika – provjera identiteta.
- Spremanje podataka u „oblak“ (eng. *Cloud Firestore)*
- Firebase spremište (eng. *Firebase Storage*) – služi za spremanje slika, videozapisa te audio zapisa na Firebase.
- Performanse aplikacije (eng. *Performance Monitoring*).

U aplikaciji za nogometnu statistiku se koristi Firebase NoSQL baza podataka. NoSQL je baza podataka u stvarnom vremenu odnosno u oblaku kod koje se podaci spremaju u JSON formatu. Baza podataka u stvarnom vremenu znači da kada korisnik napravi neku promjenu u bazi da se ono odmah prenosi na uređaj. Prednost ovakve vrste baze podataka je ta što se podaci sinkroniziraju u realnom vremenu sa svim klijentima te im je moguće pristupiti i kada aplikacija nije spojena na Internet. Firebase daje mogućnost da se definira kada i koji podaci se mogu čitati te se može odrediti tko može pristupiti određenim podacima.

3.7. NodeJS

NodeJS je pozadinska (eng. *back end*) platforma koja se temelji na JavaScriptu. NodeJS svoj kod izvršava na serveru što je korisno pri izradi kompliciranijih aplikacija u programskom jeziku JavaScript.

Prema [7], NodeJS ima nekoliko važnijih značajki, a neke od njih su:

- Brzina – zbog Google V8 JavaScript *engine* ima veliku brzinu izvršavanja. V8 JavaScript *engine* je razvijen za Google Chrome kako bi poboljšao performanse JavaScript izvođenja.
- Otvorenog je koda (eng. *Open Source*) – Korisnici imaju pravo na korištenje, proučavanje i izmjenu koda.
- Podržan je na svim platformama (MacOS, Microsoft Windows, Linux, FreeBSD, OpenBSD).

ExpressJS je jedan od najkorištenijih okvira NodeJS-a (eng. *Framework*) koji omogućuje brži prikaz promjena koje je izvršio korisnik. Kombinacija ExpressJS te Angular-a pruža programeru lakšu i kvalitetniju izradu aplikacije.

4. IZRADA APLIKACIJE

U ovom poglavlju bit će objašnjen postupak izrade aplikacije za nogometnu statistiku za englesku nogometnu ligu. Aplikacija je izrađena u Angular-u te će u sljedećem potpoglavlju biti opisan programski dio aplikacije.

Aplikacija se sastoji od sljedećih dijelova:

- „Dodaj igrača“
- „Statistika igrača“
- „Uređivanje igrača“
- „Poredak“
- „Utakmice“
- „Ekipe“

4.1. Programski dio aplikacije

Nakon kreiranja Angular projekta potrebno je bilo instalirati komponente koje će sadržavati aplikacija. Na slici 4.1. nalazi se naredba kojom se instalira komponenta u ovome slučaju komponenta za dodavanje igrača.

```
ng g component components/dodaj-igrača
```

Sl. 4.1. *Prikaz koda za instaliranje Angular komponente*

Nakon dodavanje komponenti treba instalirati Angular materijal koji služi za dizajn odnosno izgled aplikacije. Materijal se instalira pomoću sljedeće naredbe (slika 4.2.).

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

Sl. 4.2. *Prikaz koda za instaliranje Angular materijala*

Na slici 4.3. su prikazane sve vrste Angular materijala koji se dobiju nakon unosa naredbe koja je prikazana na slici 4.2. Za izradu aplikacije za nogometnu statistiku odabrana je tema *Indigo/Pink.*

```
? Choose a prebuilt theme name, or "custom" for a custom theme: (Use arrow keys)
> Indigo/Pink        [ Preview: https://material.angular.io?theme=indigo-pink ]
  Deep Purple/Amber  [ Preview: https://material.angular.io?theme=deeppurple-amber ]
  Pink/Blue Grey     [ Preview: https://material.angular.io?theme=pink-bluegrey ]
  Purple/Green       [ Preview: https://material.angular.io?theme=purple-green ]
  Custom
```

Sl. 4.3. *Prikaz koda za instaliranje Angular komponente*

Poslije odabiranja Angular materijala te dodavanja određenih modula u program potrebno je povezati Firebase bazu podataka s aplikacijom. Nakon instaliranja Firebase-a u aplikaciju potrebno je napraviti projekt na Firebase-u, pri kraju izrade projekta dobit će se sučelje prikazano na slici 4.4.

```
<!-- The core Firebase JS SDK is always required and must be listed
<script src="https://www.gstatic.com/firebasejs/7.14.5/firebase-app.

<!-- TODO: Add SDKs for Firebase products that you want to use
     https://firebase.google.com/docs/web/setup#available-libraries

<script>
  // Your web app's Firebase configuration
  var firebaseConfig = {
    apiKey: "AIzaSyAI4KYIqZPymcnGGXlpDo-BJ-KRxVK9UjQ",
    authDomain: "book-ba711.firebaseapp.com",
    databaseURL: "https://book-ba711.firebaseio.com",
    projectId: "book-ba711",
    storageBucket: "book-ba711.appspot.com",
    messagingSenderId: "375510970340",
    appId: "1:375510970340:web:240de37a91e6dd97a9a1f1"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
</script>
```

Sl. 3.4. *Postavke Firebase-a*

Kao što je prikazano na slici 4.4. postavke Firebase-a sadrže API ključ, domenu, URL, ID projekta i ostale stavke. Navedene stavke je potrebno ubaciti u kod aplikacije i to u datoteku *environment.ts* (slika 4.5.).

```
export const environment = {
  production: false,
  firebaseConfig: {
    apiKey: "AIzaSyAI4KYIqZPymcnGGX1pDo-BJ-KRxVK9UjQ",
    authDomain: "book-ba711.firebaseapp.com",
    databaseURL: "https://book-ba711.firebaseio.com",
    projectId: "book-ba711",
    storageBucket: "book-ba711.appspot.com",
    messagingSenderId: "375510970340",
    appId: "1:375510970340:web:240de37a91e6dd97a9a1f1"
  }
};
```

Sl. 4.5. *Implementacija Firebase postavki*

Na slici 4.5. je prikazana baza podataka na Firebase-u gdje se igrači spremaju pod točno određenim ključem nakon dodavanja od strane korisnika.

Slika 4.6. prikazuje na koji način se sprema svaki igrač u bazu podataka na Firebase-u. Svaki igrač prilikom spremanja u bazu dobije svoj jedinstveni ključ.

Sl. 4.6. *Izgled baze podataka Firebase*

S dodavanjem koda u aplikaciju te dodavanjem određenih Firebase modula aplikacija je povezana s Firebase bazom podataka. U aplikaciji se koristi i AngularFire koji omogućava CRUD operacije odnosno kreiranje, čitanje, izmjene te brisanje podataka igrača. Slika 4.7. prikazuje programski kod kojim se omogućuju CRUD operacije.

```
/* Create player */
AddPlayer(player: Player) {
  this.playersRef.push({
    first_name: player.first_name,
    goals: player.goals,
    asist: player.asist,
    player_name: player.player_name,
    club: player.club
  })

}

/* Get player */
GetPlayer(id: string) {
  this.playerRef = this.db.object('players-list/' + id);
  return this.playerRef;
}

/* Get player list */
GetPlayerList() {
  this.playersRef = this.db.list('players-list');
  return this.playersRef;
}

/* Update  */
UpdatePlayer(id, player: Player) {
  this.playerRef.update({
    first_name: player.first_name,
    goals: player.goals,
    asist: player.asist,
    player_name: player.player_name,
    club: player.club,
  })

}
/* Delete  */
DeletePlayer(id: string) {
  this.playerRef = this.db.object('players-list/' + id);
  this.playerRef.remove()
}
```

Sl. 4.7. *CRUD operacije*

Dodavanje igrača u bazu aplikacije omogućava sljedeći dio koda (slika 4.8.).

```
submitPlayerForm() {
  this.playerForm = this.fb.group({
    first_name: ['', [Validators.required]],
    goals: ['', [Validators.required]],
    asist: ['', [Validators.required]],
    player_name: ['', [Validators.required]],
    club: ['', [Validators.required]]
  })
}
```

Sl. 4.8. *Dio koda koji omogućuje pohranu igrača*

Funkcija *submitPlayerForm()* traži od korisnika da unese sljedeće stavke: ime i prezime igrača, broj golova, broj asistencija te klub za koji nastupa igrač.

Dio aplikacije *Statistika igrača* omogućuje prikaz glavnih podataka o igračima a to su:

- Ime igrača
- Prezime igrača
- Klub
- Broj golova
- Broj asistencija
- Opcije

Stavka *opcije* daje mogućnost korisniku aplikacije da unese izmjene o nekome igraču ili da ga obriše s liste.

Za dijelove aplikacije *poredak* i *utakmice* korišteni su API-ja koji omogućuju automatsku izmjenu podataka bez da korisnik samostalno mora unositi promjene. Primjer API za dohvaćanje podataka o trenutnom stanju na ljestvici se nalazi na slici 4.9.

```
export class StandingService {
  private _teams: Subject<Array<Team>> = new BehaviorSubject<Array<Team>>([]);
  private readonly API = "https://api.football-data.org/v2/competitions/";
  private readonly leagues = {
    premierLeague: "PL"
  };

  public readonly teams: Observable<Array<Team>> = this._teams.asObservable();

  constructor(private http: HttpClient) { }

  fetchStandings(league: string) {
    this.http.get(`${this.API}${this.leagues[league]}/standings`).subscribe((response: ApiRespone) => {
      this._teams.next(response.standings[0].table);
    });
  }
}
```

Sl. 4.9. *Prikaz koda kojim se dohvaća API za poredak*

Na slici 4.10. se nalazi dio koda kojim je definirano koje stavke sadrži tablica odnosno poredak klubova:

```
columnsToDisplay = [
  "position",
  "name",
  "points",
  "playedGames",
  "won",
  "draw",
  "lost",
  "goalsFor",
  "goalsAgainst",
  "goalDifference"
];
```

Sl. 4.10. *Dio koda kojim se određuju stavke koje sadrži tablica*

Zadnji dio aplikacije čini prikaz ekipa odnosno igrača koji sudjeluju u engleskoj nogometnoj ligi. Popis igrača se dohvaća preko API-ja (slika 4.11.).

```
getGeneralData(cb: (value: any) => void): void {
  this.http.jsonp('https://json2jsonp.com/?url=https://fantasy.premierleague.com/api/bootstrap-static/', 'callback').subscribe(cb);
}
```

Sl. 4.11. *API kojim se dohvaća popis igrača klubova*

Korisniku je omogućeno da filtrira igrače po pozicijama i klubovima a to je omogućeno sljedećim dijelom koda (slika 4.12.) :

```
filterPlayers(): void {
    let filtered = [];
    if(!this.generalData || !this.generalData.elements) return;
    this.generalData.elements.forEach(player => {
        const s = this.op_players;
        let include = true;
        if(s.name &&
            player.web_name.toLowerCase().indexOf(s.name.toLowerCase()) === -1)
            include = false;
        if(!s.teams.all && !s.teams[player.team])
            include = false;
        if(!s.poss.all && !s.poss[player.element_type])
            include = false;
        this.filters.some(filter => {
            if(filter.show && !this.between(player[filter.field], filter.low, filter.high)) {
                include = false;
                return true;
            }
            return false;
        });
        if(include) filtered.push(player);
    });
    this.filteredPlayers = filtered.sort((a, b) => {
        let propA = a[this.op_players.sort];
        let propB = b[this.op_players.sort];
        let valueA = isNaN(+propA) ? propA : +propA;
        let valueB = isNaN(+propB) ? propB : +propB;
        return (valueA < valueB ? -1 : 1) * (this.op_players.reverse ? -1 : 1);
    });
}
```

Sl. 4.12. *Prikaz koda kojim se omogućuje filtriranje igrača*

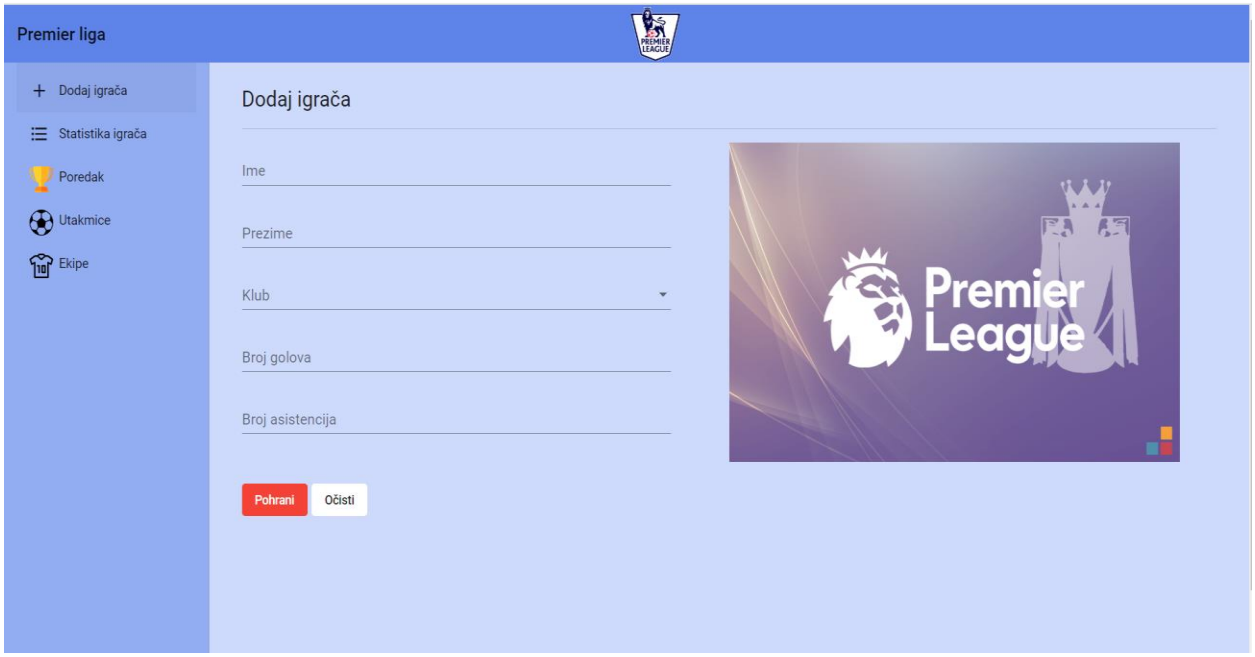
5. KORISNIČKO SUČELJE APLIKACIJE

U ovom poglavlju će biti objašnjeno na koji način radi aplikacija za nogometnu statistiku. Aplikacija je namijenjena za praćenje statistike za englesku nogometnu ligu. U nastavku će biti prikazani dijelovi aplikacije te način na koji korisnik koristi aplikaciju.

5.1. Dodavanje igrača

Kada korisnik pokrene aplikaciju pojavit će mu se sučelje prikazano na slici 5.1.. Kao što je prikazano na slici, aplikacija sadrži izbornik s lijeve strane gdje može odabrati što želi otvoriti. Izbornik sadrži sljedeće stavke:

- „Dodaj igrača“
- „Statistika igrača“
- „Poredak“
- „Utakmice“
- „Ekipe“



Sl. 5.1. *Početno sučelje aplikacije*

Kada korisnik odabere u izborniku stavku „Dodaj igrača“, pojaviti će mu se sučelje u kojem ima mogućnost dodati igrača. Pri dodavanju igrača korisnik mora ispuniti sljedeće stavke:

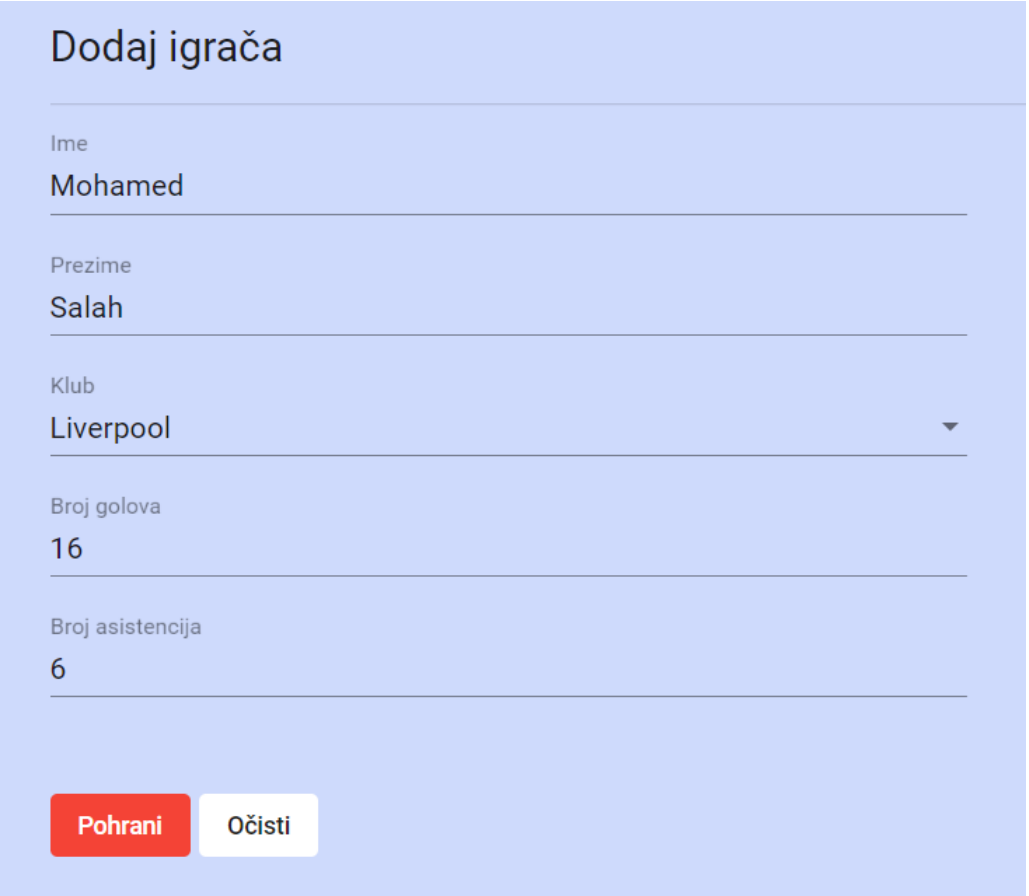
- Ime
- Prezime
- Klub
- Broj golova
- Broj asistencija

Na slici 5.2. je prikazano sučelje koje korisnik treba popuniti za dodavanje igrača. Da bi korisnik dodao igrača na listu treba pritisnuti na „*Pohrani*“, pritiskom na gumb „*Pohrani*“ igrač se automatski sprema u bazu podataka koja se nalazi na Firebase-u.

Sl. 5.2. *Sučelje za dodavanje igrača*

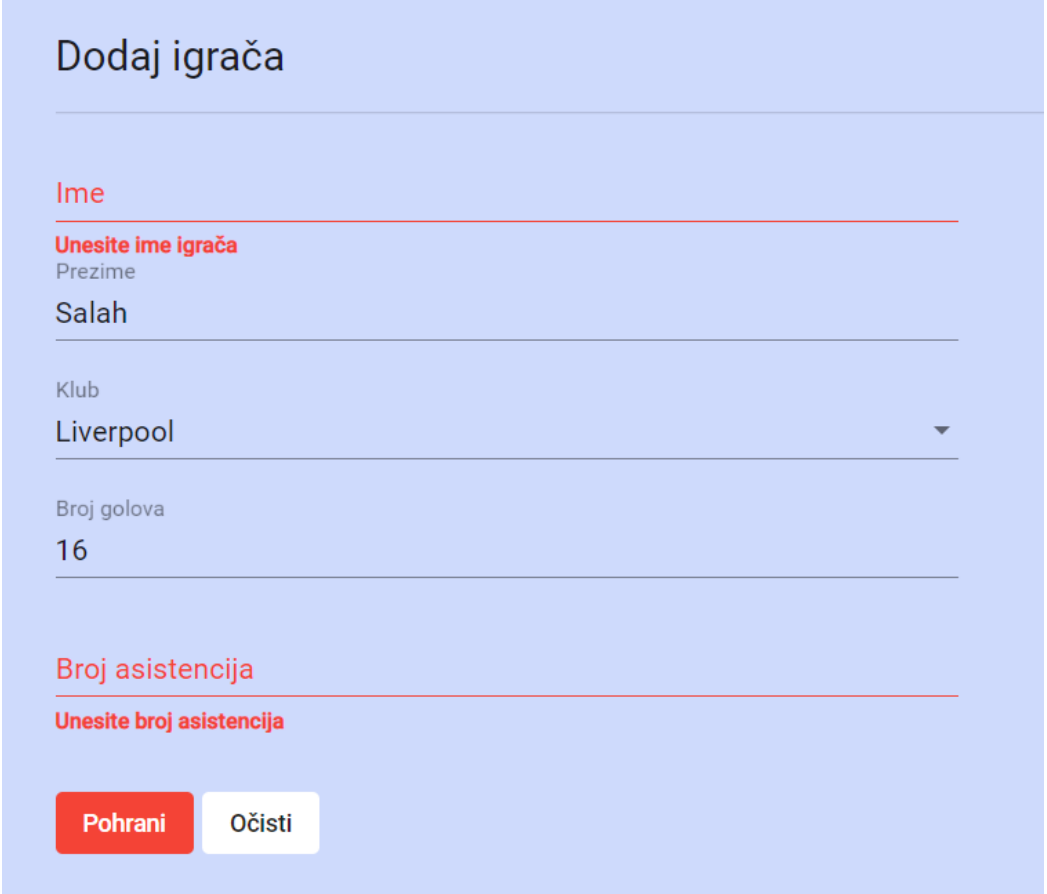
Pri odabiru kluba za igrača pojaviti će se padajući izbornik u kojem se nalazi popis klubova koji se natječu u prvoj engleskoj nogometnoj ligi.

Slika 5.3. prikazuje primjer na koji korisnik dodaje odnosno unosi igrača na listu te ga sprema u bazu podataka koja se nalazi na Firebase-u. Korisnik nakon što unese sve podatke treba pritisnuti na gumb „Pohrani“ odnosno ako želi obrisati podatke odabere gumb „Očisti“.



Sl. 5.3. *Primjer dodavanja igrača*

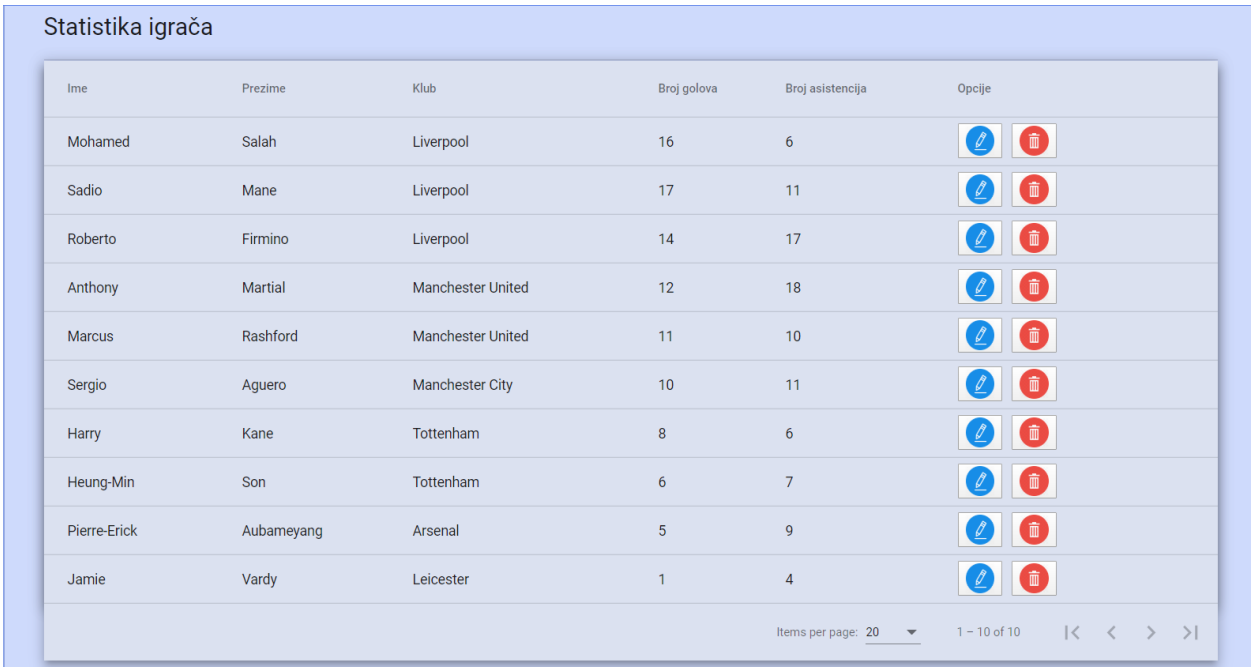
U programskom dijelu je napravljeno da korisnik ne može unijet broj koji je veći od dvoznamenkastog te ako korisnik zaboraviti ispuniti neku stavku pojavit će mu se upozorenje kao što je prikazano na slici 5.4. Da bi korisnik uspješno unio igrača na listu treba ispuniti sve podatke.



Sl. 5.4. *Upozorenja u slučaju izostavljanja podatka*

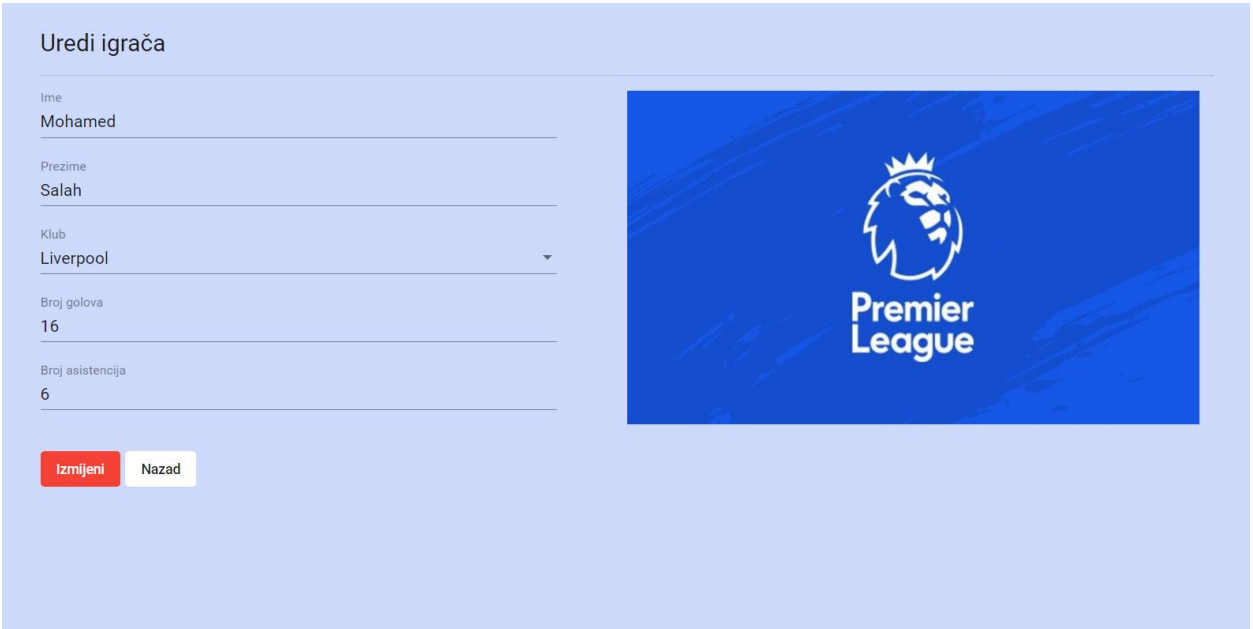
5.2. Prikaz statistike igrača

Kada korisnik odabere u izborniku opciju „*Statistika igrača*“ pojavit će mu se sučelje prikazano na slici 5.5. Na slici se vidi da statistika sadrži sljedeće stavke: ime, prezime, broj golova, broj asistencija te dvije opcije: brisanje igrača te izmjena podataka o igraču.



Sl. 5.5. *Lista igrača*

Nakon pritiska na ikonicu „Izmijeni“ korisniku će se pojaviti sučelje (slika 5.6.) na kojoj ima mogućnost izmijeniti podatak o igraču.



Sl. 5.6. *Sučelje „Uredi igrača“*

Kada korisnik završi s izmjenom podatka o igraču pojavit će mu se prozorčić u kojem će se korisnika pitati da li je siguran za izmjenu podataka.

Ako korisnik želi obrisati igrača s liste, odabrat će ikonicu „Obriši“ kod toga igrača te će se nakon toga pojaviti prozorčić koji će pitati korisnika da li je siguran da želi obrisati igrača.

5.3. Poredak

Opcija u izborniku „*Poredak*“ omogućuje korisniku da provjeri stanje na ljestvici. Podaci u tablici se dohvaćaju pomoću API-ja, podaci u tablici se ažuriraju nakon svakog odigranog kola, a detaljnije o tome na koji način API radi te na koji način je implementiran u program odnosno aplikaciju pojašnjeno je u prethodnom poglavlju.

Tablica sadrži sljedeće stavke:

- Pozicija kluba
- Naziv ekipe
- Broj bodova
- Odigrani susreti
- Broj pobjeda
- Broj neriješenih rezultata
- Broj izgubljenih utakmica
- Zabijeni golovi
- Primljeni golovi
- Gol razlika

Na slici 5.7. prikazan je primjer kako izgleda sučelje opcije „*Poredak*“ u aplikaciji za nogometnu statistiku za englesku nogometnu ligu.

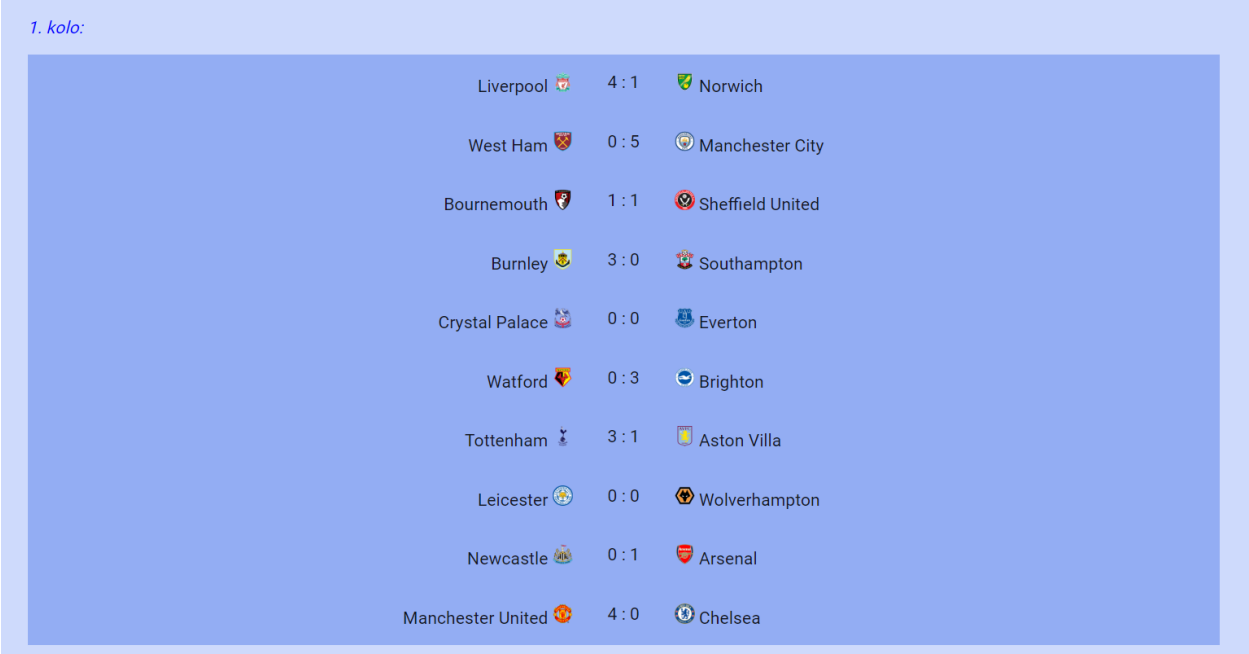
#	Ekipa	B	OS	P	N	I	GZ	GP	GR
1	Liverpool FC	82	29	27	1	1	66	21	45
2	Manchester City FC	57	28	18	3	7	68	31	37
3	Leicester City FC	53	29	16	5	8	58	28	30
4	Chelsea FC	48	29	14	6	9	51	39	12
5	Manchester United FC	45	29	12	9	8	44	30	14
6	Wolverhampton Wanderers FC	43	29	10	13	6	41	34	7
7	Sheffield United FC	43	28	11	10	7	30	25	5
8	Tottenham Hotspur FC	41	29	11	8	10	47	40	7
9	Arsenal FC	40	28	9	13	6	40	36	4
10	Burnley FC	39	29	11	6	12	34	40	-6
11	Crystal Palace FC	39	29	10	9	10	26	32	-6
12	Everton FC	37	29	10	7	12	37	46	-9
13	Newcastle United FC	35	29	9	8	12	25	41	-16
14	Southampton FC	34	29	10	4	15	35	52	-17
15	Brighton & Hove Albion FC	29	29	6	11	12	32	40	-8
16	West Ham United FC	27	29	7	6	16	35	50	-15
17	Watford FC	27	29	6	9	14	27	44	-17
18	AFC Bournemouth	27	29	7	6	16	29	47	-18
19	Aston Villa FC	25	28	7	4	17	34	56	-22
20	Norwich City FC	21	29	5	6	18	25	52	-27

Sl. 5.7. *Poredak*

5.4. Utakmice

Četvrta opcija koju korisnik može izabrati u izborniku je opcija pod nazivom „Utakmice“. Rezultati utakmica se dohvaćaju kao i poredak klubova pomoću API-ja. Opcija „Utakmice“ sadrži sljedeće stavke: broj kola, ime kluba domaćina, ime gostujućeg kluba te konačni rezultat utakmice.

Na slici 5.8. je prikazan primjer kako izgleda sučelje opcije „Utakmice“.

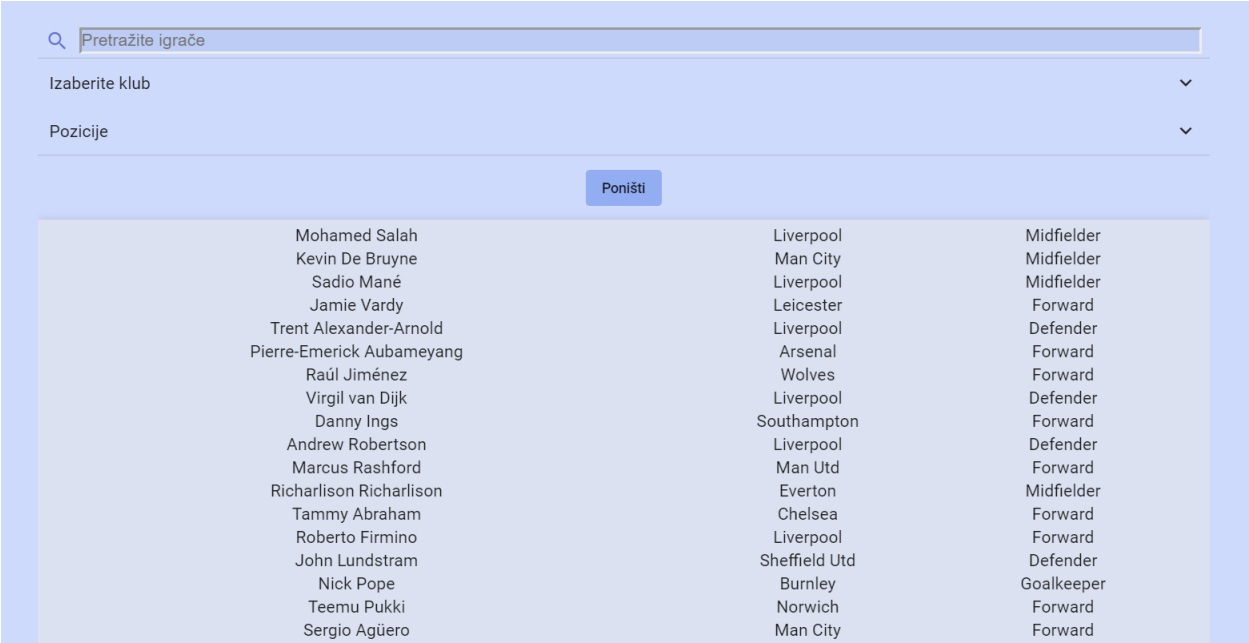


Sl. 5.8. *Primjer prikaza rezultata utakmica (1.kolo)*

5.5. Ekipe

Zadnja opcija koju sadrži izbornik aplikacije je „Ekipe“. Opcija „Ekipe“ nudi više mogućnosti korisniku u pogledu pretraživanja igrača. Mogućnosti po kojima korisnik može pretražiti igrače bit će navedeni u nastavku ovog potpoglavlja.

Korisnik kada odabere opciju „Ekipe“ pojavit će mu se popis svih igrača koji nastupaju u aktualnoj sezoni engleske nogometne lige. Slika 5.9. prikazuje početnu stranicu koja se prikaže kada korisnik odabere navedenu opciju. Popis igrača sadrži naziv kluba za koji nastupa igrač te koju poziciju igra. Korisnik ima mogućnost pritiskom na gumb „Poništi“ vratiti postavke popisa igrača na početno stanje.



Sl. 5.9. *Sučelje opcije „Ekipe“*

Korisnik ima tri načina na koja može pretražiti igrača a to su:

- Tražilica
- Po klubu
- Po pozicijama

Na slici 5.10. se nalazi primjer prema kojem korisnik može pretraživati igrače.

Sl. 5.10. *Primjer pretraživanja igrača*

Slika 5.11. prikazuje padajući izbornik koji korisnik otvara pritiskom na opciju „Izaberite klub“ te nakon što se otvori izbornik korisnik odabire klub za koji želi da mu se prikažu igrači.

Sl. 5.11. *Padajući izbornik „Izaberite klub“*

U programskom dijelu aplikacije je omogućeno da korisnik može prikazati popis igrača za jedan ili više klubova (slika 5.12.).

Sl. 5.12. *Primjer korištenja izbornika „Izaberite klub“*

Korisnik može pretražiti igrače i po pozicijama tako da se pritiskom na opciju „Pozicije“ otvara padajući izbornik (slika 5.13.) koji sadrži sljedeće stavke:

- Sve pozicije
- Golmani
- Obrambeni igrači
- Vezni igrači
- Napadači

Sl. 5.13. *Padajući izbornik „Pozicije“*

Pri odabiru pozicija moguće je odabrati da se na popisu igrača prikažu npr. samo golmani ili da se prikažu samo vezni igrači i napadači (slika 5.14.).

Sl. 5.14. *Prikaz odabira veznih igrača i napadača*

Na slici 5.15. je prikazan primjer pretraživanja igrača. Primjer prikazuje popis veznih i obrambenih igrača Liverpoola i Manchester City-a.

Sl. 5.15. *Primjer pretraživanja igrača*

6. TESTIRANJE APLIKACIJE

Pri testiranju aplikacije su se koristila dva testa: testiranje jedinice (eng. *Unit test*) i cjelovito testiranje (eng. *end-to-end*). Navedena testiranja će biti objašnjena u sljedećim potpoglavljima. Testiranje aplikacije je važno provesti kako bi se provjerila ispravnost, funkcionalnost te rad odnosno povezanost između različitih dijelova aplikacije.

6.1. Postavke testiranja

Testiranje aplikacije se provodi pomoću okvira Jasmine te alata Karma. Jasmine je okvir koji se koristi za testiranje programa napisanih JavaScript kodom. Jasmine sadrži veliki broj funkcija pomoću kojeg je korisniku omogućeno pisanje više vrsta testova. Jasmine je dobar razvojni okvir za testiranje jer omogućava pisanje testova koji će biti lako čitljiv i jednostavni te će biti svima jasni [8]. Okvir Jasmine pri testiranju koristi sljedeće funkcije:

- *beforeEach*
- *afterEach*
- *beforeAll*
- *afterAll*

Karma je alat koji omogućuje pokretanje web preglednika te pokretanje Jasmine testova. Rezultati testiranja se prikazuju u naredbenom retku [8]. Karma alat se instalira pomoću naredbe *npm install -g karma* te se pokreće automatski nakon unosa naredbe *ng test.*

6.2. Testiranje jedinice

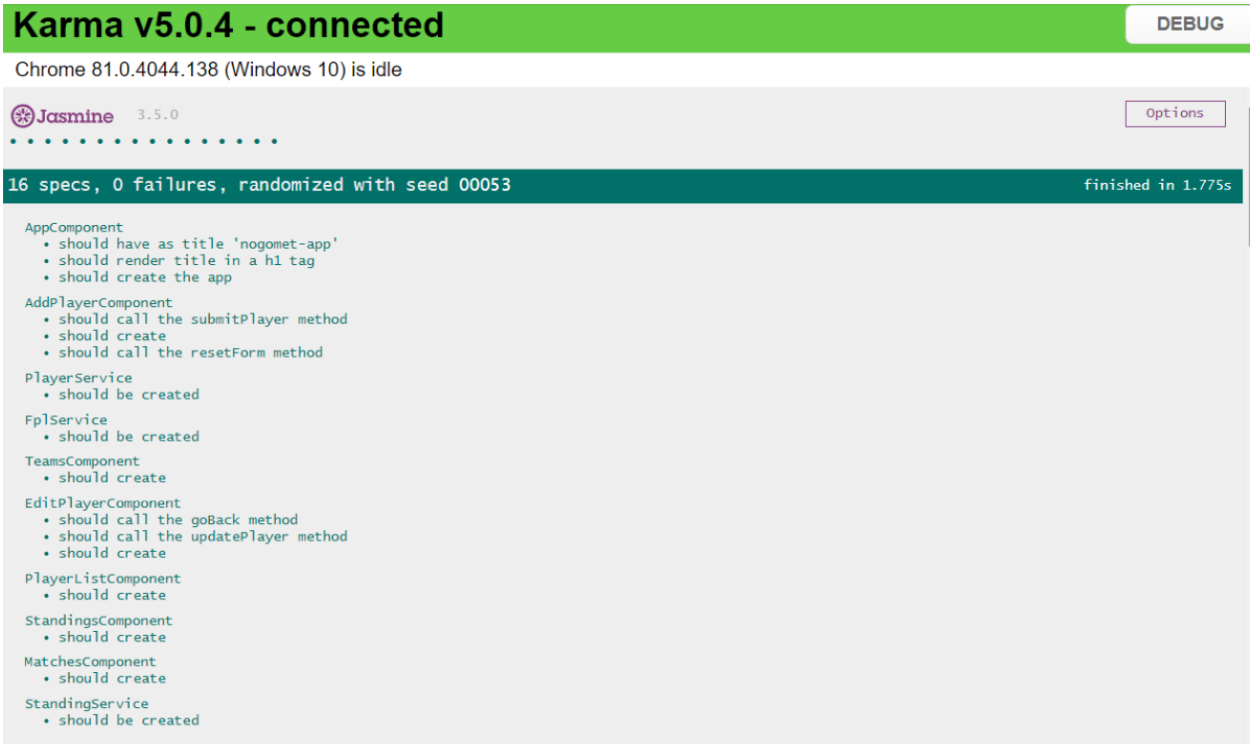
Prva vrsta testiranja koja je provedena je testiranje jedinice koja testira samo jedan dio aplikacije. Testiranje jedinice koristi okvir Jasmine te alat Karma koji su objašnjeni u prethodnom potpoglavlju. Primjer testiranja jedinice se nalazi na slici 6.1.

```
it('should create the app', () => {
  const fixture = TestBed.createComponent(AppComponent);
  const app = fixture.componentInstance;
  expect(app).toBeTruthy();
});

it(`should have as title 'nogomet-app'`, () => {
  const fixture = TestBed.createComponent(AppComponent);
  const app = fixture.componentInstance;
  expect(app.title).toEqual('nogomet-app');
});

it('should render title in a h1 tag', async (() => {
  const fixture = TestBed.createComponent(AppComponent);
  fixture.detectChanges();
  const compiled = fixture.debugElement.nativeElement;
  expect(compiled.querySelector('h1').textContent).toContain('Premier liga');
}));

});
```

Sl. 6.1. *Primjer koda za testiranje*

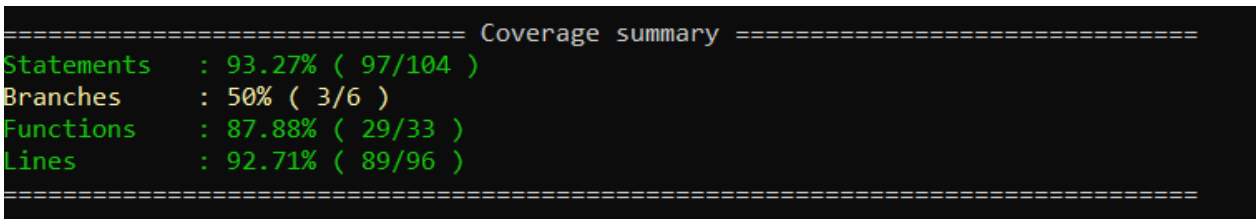
Slika 6.1. prikazuje primjer u kojem se testiraju tri stvari. Prvo se testira hoće li se aplikacija učitati, zatim da li je naziv aplikacije „*nogomet-app*“ te treća stvar je hoće li se učitati naslov „*Premier liga*“.

Prema [9], testiranje jedinice se obavlja na sljedeći način: prvo je potrebno u naredbeni redak NodeJS unijeti naredbu „*ng test*“ nakon toga će se pokrenuti Karma te će se pojaviti sučelje prikazano na slici 6.2.



Sl. 6.2. *Testiranje jedinice - rezultati*

Kako bi se provjerila pokrivenost koda koji je obuhvaćen testiranjem jedinice koristi se sljedeća naredba: „*ng test –code-coverage"*. Pokrivenost koda (eng. *Code coverage*) zapravo provjerava koliki je dio koda obuhvaćen testiranjem jedinice. Prednosti provedbe pokrivenosti koda su sljedeće: provjera učinkovitosti, izvornosti koda te pokrivenosti koda testiranjem. Rezultati pokrivenosti koda aplikacije za nogometnu statistiku se nalazi na slici 6.3.

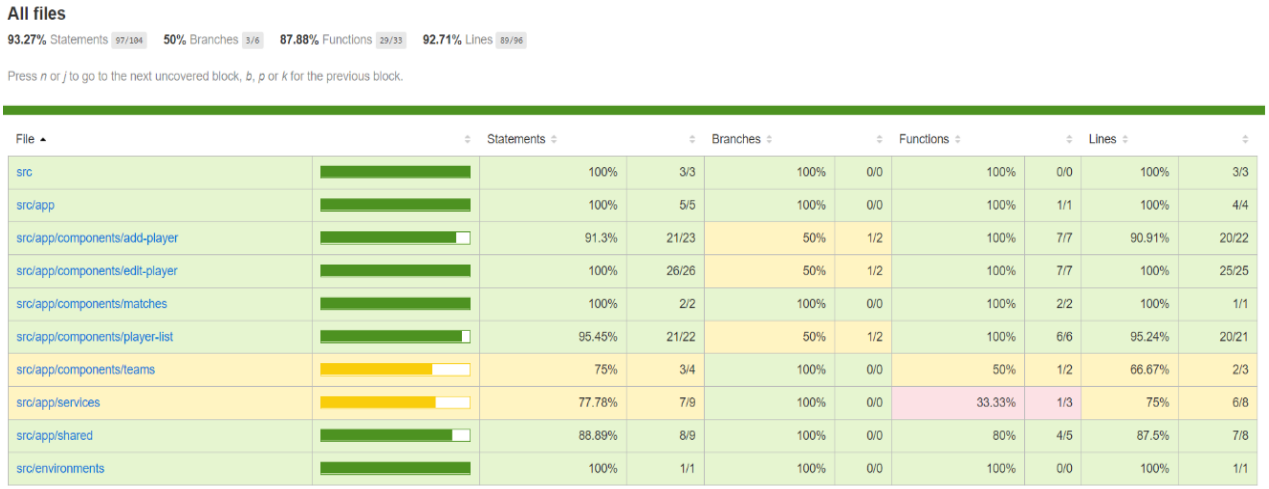


Sl. 6.3. *Rezultati za pokrivenost koda testiranjem*

Pokrivenost koda pokazuje koliki je dio koda aplikacije obuhvaćen testiranjem. Rezultati pokrivenosti koda sadržavaju sljedeće stavke:

- Datoteka (eng. File) – dijelovi aplikacije koji se testiraju.
- Izvještaji (eng. Statements) – koliko je programskih dijelova u aplikaciji uspješno izvršeno. Računa se kao omjer uspješno izvršenih dijelova aplikacije i ukupnog broja dijelova aplikacije.
- Pokrivenost grana (eng. Branches) – koliko je grana koda uspješno izvršeno.
- Pokrivenost funkcija (eng. Functions) – koliko je funkcija u kodu pozvano te uspješno izvršeno.
- Pokrivenost linija koda (eng. Lines)– koliko je linija koda obuhvaćeno testiranjem.

Pokrivenost koda se koristi kako bi se provjerila kvaliteta programskog koda aplikacije. Kao što je prikazano na slici 6.4. ukupna pokrivenost koda iznosi 93.27%. Uz ukupnu pokrivenost izvješće pokrivenosti koda prikazuje i pokrivenost funkcija, grana te linija koda. Osim ukupne pokrivenosti izvješća, moguće je vidjeti i kolika je pokrivenost izvješća po pojedinim dijelovima aplikacije. Rezultati za pokrivenost koda aplikacije za nogometnu statistiku pokazuju kako nekoliko datoteka kao što su *add-player, edit-player* te *matches* imaju pokrivenost izvješća koja iznosi 100%, to znači da su pri testiranju aplikacije obuhvaćeni svi dijelovi tih datoteka. Druge datoteke imaju pokrivenost između 75% te 96%, što znači da nisu svi dijelovi obuhvaćeni testiranjem, ali se smatra da je pokrivenost izvješća koda uspješna ako ona iznosi više od 70%.



All files

93.27% Statements 97/104 50% Branches 3/6 87.88% Functions 29/33 92.71% Lines 89/96

Press n or j to go to the next uncovered block, b, p or k for the previous block.

File	Statements		Branches		Functions		Lines	
src	100%	3/3	100%	0/0	100%	0/0	100%	3/3
src/app	100%	5/5	100%	0/0	100%	1/1	100%	4/4
src/app/components/add-player	91.3%	21/23	50%	1/2	100%	7/7	90.91%	20/22
src/app/components/edit-player	100%	26/26	50%	1/2	100%	7/7	100%	25/25
src/app/components/matches	100%	2/2	100%	0/0	100%	2/2	100%	1/1
src/app/components/player-list	95.45%	21/22	50%	1/2	100%	6/6	95.24%	20/21
src/app/components/teams	75%	3/4	100%	0/0	50%	1/2	66.67%	2/3
src/app/services	77.78%	7/9	100%	0/0	33.33%	1/3	75%	6/8
src/app/shared	88.89%	8/9	100%	0/0	80%	4/5	87.5%	7/8
src/environments	100%	1/1	100%	0/0	100%	0/0	100%	1/1

Sl. 6.4. *Cjelokupni rezultati za pokrivenost koda*

6.3. Cjelovito testiranje

Cjelovito testiranje je testiranje koje obuhvaća cijelu aplikaciju odnosno obuhvaća testiranje aplikacije od početka do kraja. Glavni zadatak ovog testiranja je simulirati stvarnog korisnika u web pregledniku [10]. Cilj je utvrditi da li aplikacija ispravno radi od početka do kraja. Da bi se aplikacija testirala pomoću cjelovitog testa potrebno je napisati testne kodove u programski dio aplikacije.

Faze cjelovitog testiranja su:

- Planiranje testa
- Izrada testnih slučajeva
- Izvršenje testa
- Analiza rezultata

Primjeri koda kojim se ostvaruje testiranje dijela aplikacije koji je zadužen za dodavanje igrača u aplikaciju se nalazi na slikama 6.5. i 6.6..

```
import { browser, by, element } from 'protractor';
export class AddPlayerPage {
    navigateTo(){
        return  browser.get('http://localhost:4200/add-player');
    }
    getNameTextbox(){
        return element(by.name('Ime'));
    }
    getPrezimeTextbox(){
        return element(by.name('Prezime'));
    }
    getBrojTextbox(){
        return element(by.name('BrGol'));
    }
    getAsistTextbox(){
        return element(by.name('BrAsist'));
    }
    getKlub(){
        return element(by.name('selected'));
    }
    getBinding(){
        return element.all(by.css('.hej')).first();
    }
    getOčistiButton(){
        return element(by.css('#btnClear'));
    }
    getPohraniButton(){
        return element(by.css('#btnSubmit'));
    }
    getplayerForm(){
        return element(by.css('.inner-wrapper'));
    }
```

Sl. 6.5. *Definiranje funkcija za cjelovito testiranje*

Na slici 6.5. je prikazano definiranje funkcija koje se koriste pozivanje određene metode u aplikaciji, dok se na slici 6.6. pozivaju te iste funkcije kako bi se izvršio zadatak.

```
import { AddPlayerPage } from './add-player.po';
import { browser, element, by } from 'protractor';

describe('AddPlayer tests', () => {
    let page: AddPlayerPage;

    beforeEach(() => {
        page = new AddPlayerPage();
        page.navigateTo();
    });
    it('Player form should be valid', () => {
        page.getNameTextbox().sendKeys('Harry');
        browser.sleep(600);
        page.getPrezimeTextbox().sendKeys('Kane');
        browser.sleep(600);
        page.getKlub().click();
        browser.sleep(1000);
        page.getBinding().click();
        browser.sleep(600);
        page.getBrojTextbox().sendKeys('14');
        browser.sleep(600);
        page.getAsistTextbox().sendKeys('3');
        browser.sleep(600);
       // page.getOčistiButton().click();
        //browser.sleep(1000);
        page.getPohraniButton().click();
        browser.sleep(1100);
       });


});
```

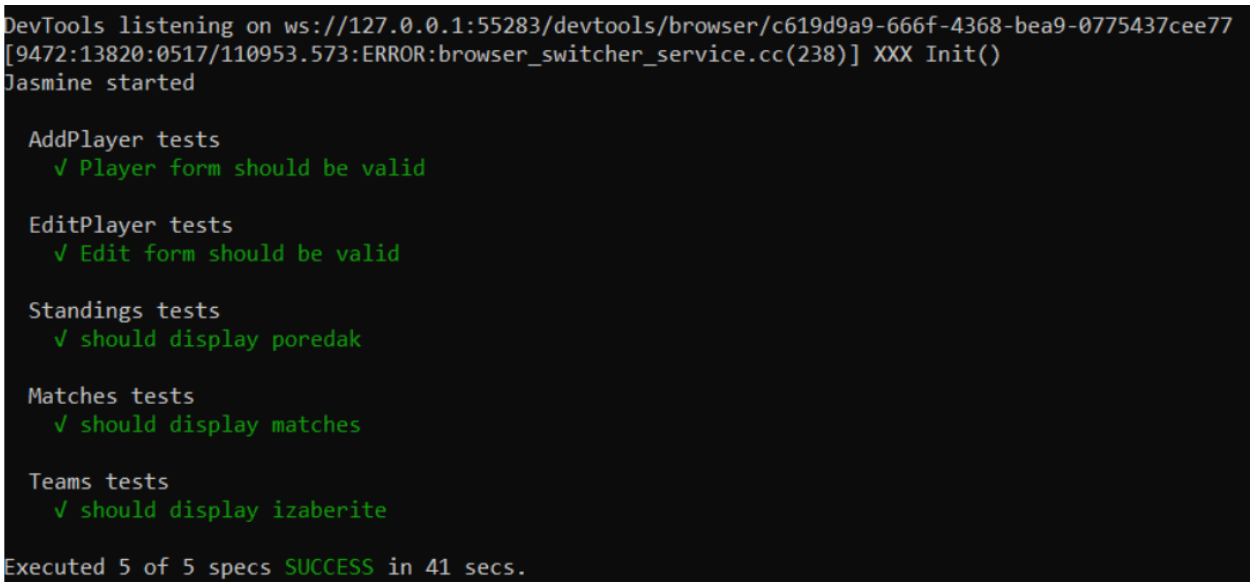
Sl. 6.6. *Pozivanje funkcija kod cjelovitog testiranja*

Prednosti cjelovitog testiranja aplikacije su:

- Detaljnije od pokrivenosti koda.
- Provjeriti ispravnost aplikacije od početka do kraja.
- Otkriti pogreške
- Smanjiti troškove
- Smanjiti vrijeme

Kako bi se provelo cjelovito testiranje potrebno je napisati programske kodove unutar pojedinih dijelova aplikacije. Kao što je prikazano na slici 6.7., cjelovito testiranje aplikacije za nogometnu statistiku se sastoji od pet testova: *AddPlayer*, *EditPlayer*, *Standings*, *Matches* te *Teams* test. U svakom od navedenih dijelova je bilo potrebno napisati programski kod kojim se određuje što se treba izvršiti u određenom trenutku i kojim redom.

Programskim kodom u *AddPlayer* testu (slika 6.6.) je omogućena simulacija dodavanja igrača u bazu podataka unošenjem određenih podataka kao što su: ime, prezime, klub, broj golova te broj asistencija. *EditPlayer* testom se provjerava popis igrača na listi te mogućnosti izmjene podataka o igraču te brisanje igrača s liste. *Standings* test omogućuje provjeru učitavanja i prikaza tablice, dok test *Matches* provjerava da li se prikazuju rezultati utakmica po kolima. Posljednji test je *Teams* test koji provjerava mogućnost sortiranja i pretraživanja igrača po imenu, pozicijama te klubovima engleske nogometne lige.



Sl. 6.7. *Rezultati za cjelovito testiranje*

7. ZAKLJUČAK

Problem diplomskog rada je bio izraditi web aplikaciju za praćenje nogometne statistike te ju testirati. Aplikacija je izrađena u razvojnom okviru Angular, te je za bazu podataka korištena Google-ova platforma Firebase. Cilj aplikacije je bio omogućiti unos podataka o nogometašima koji sudjeluju u engleskoj nogometnoj ligi. Kao što je ranije objašnjeno u radu, izradom aplikacije korisniku je omogućen unos podataka o igračima, a podaci koje korisnik mora unijeti su ime, prezime, klub za koji igrač nastupa te broj golova i asistencija. Osim unosa podataka korisnik ima mogućnost provjeriti i ostale stavke kao što su: stanje na ljestvici, popis igrača te rezultati utakmica u aktualnoj sezoni.

Kako bi se provjerila ispravnost i funkcionalnost aplikacije korišteni su Angular testovi. Aplikacija je testirana pomoću dva testa, a to su: testiranje jedinice (eng. *Unit test*) i cjelovito testiranje (eng. *End-to-End test*). Prvi test, testiranje jedinice omogućuje testiranje pojedinačnih dijelova aplikacije te provjeru na koji način radi neki dio aplikacije. S druge strane, cjelovito testiranje je vrsta testiranja u kojem je glavni cilj simulirati stvarnog korisnika u pregledniku te provjeriti rad aplikacije od početka do kraja. U radu su detaljnije opisani testovi pomoću kojih se aplikacija testirala te su objašnjeni i rezultati samog testiranja.

LITERATURA

[1] EDUCBA [online], dostupno na: https://www.educba.com/what-is-visual-studio-code/ [zadnji pristup 22.5.2020.]

[2] Angular [online], dostupno na: https://angular.io/, [zadnji pristup 22.5.2020.]

[3] Bccrwp.org [online], dostupno na: https://hr.bccrwp.org/compare/difference-between-javascript-and-typescript/, [zadnji pristup 22.5.2020.]

[4] Vinod Kumar, Ezine Articles [online], dostupno na https://ezinearticles.com/?What-Is-HTML?-Advantage-and-Disadvantage-of-HTML&id=9963710, [zadnji pristup 22.5.2020.]

[5] Web Tech [online], dostupno na: http://www.webtech.com.hr/css.php, [zadnji pristup 22.5.2020.]

[6] Firebase [online], dostupno na: https://firebase.google.com/docs/database, [zadnji pristup 22.5.2020.]

[7] Altexsoft [online], dostupno na: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-node-js-web-app-development/, [zadnji pristup 22.5.2020.]

[8] CodeCraft [online], dostupno na: https://codecraft.tv/courses/angular/unit-testing/jasmine-and-karma/, [zadnji pristup 25.5.2020.]

[9] AngularJS [online], dostupno na: https://docs.angularjs.org/guide/unit-testing, [zadnji pristup 25.5.2020.]

[10] Katalon [online], dostupno na: https://www.katalon.com/resources-center/blog/end-to-end-e2e-testing/, [zadnji pristup 25.5.2020.]

SAŽETAK

Tema ovog diplomskog rada je izrada i testiranje aplikacije za nogometnu statistiku. Aplikacija je izrađena u razvojnom okviru Angular koji se koristi za razvoj web i mobilnih aplikacija, dok se za bazu podataka koristio Firebase. Aplikacija za nogometnu statistiku omogućuje korisniku prikaz statistike za englesku nogometnu ligu. Osim prikaza statistike, korisnik ima mogućnost unijeti određene podatke za igrače kao što su: ime, prezime, broj golova i asistencija te klub za koji nastupa. Drugi dio rada je testiranje aplikacije, testiranje omogućuje provjeru da li aplikacija ispravno radi. Aplikacija za nogometnu statistiku je testirana pomoću dva testa: Testiranje jedinice (*eng. Unit test*) i Cjelovito testiranje (*eng. End-to-End test*). Testiranje jedinice omogućuje da se testira samo jedan dio aplikacije, dok se cjelovitim testiranjem aplikacija testira od početka do kraja, odnosno kod cjelovitog testiranja je potrebno simulirati stvarnog korisnika u pregledniku.

Ključne riječi: Angular, Firebase, statistika, testiranje, web aplikacija

ABSTRACT

Creating and testing an Angular application for football statistics

The theme of this thesis of graduates is to develop and test applications for football statistics. The application is made in the next window, Angular, which is used for developing web and mobile applications and databases I have used Firebase. App for football statistics allows the user to view the statistics for the English football league. In addition to displaying statistics, the user can enter some data for players, such as name, surname, number of goals and assists, and the club, which is. The second part of testing applications, testing allows you to check whether the application is properly working. App for football statistics was tested using two tests: Unit test and End-to-End test. Unit testing allows you to check only part of the application, and the integrity of the application testing the testing from beginning to end, or the full code test needs to simulate a real user in the browser.

Keywords: Angular, Firebase, statistics, testing, web application

ŽIVOTOPIS

Josip Faletar je rođen 14.3.1996. u Vinkovcima. Student je 2. godine sveučilišnog diplomskog studija računarstvo. 2003. godine upisuje Osnovnu školu Ivan Goran Kovačić u Vinkovcima. Nakon završene osnovne škole upisuje Gimnaziju Matije Antuna Reljkovića gdje je pohađao Matematičku gimnaziju. 2015. godine, nakon položene Državne mature, upisuje Fakultet Elektrotehnike, Računarstva i Informacijskih Tehnologija u Osijeku. 2018. godine nakon završetka preddiplomskog studija računarstvo stječe titulu sveučilišnog prvostupnika inženjera računarstva, te nakon toga upisuje diplomski sveučilišni studij računarstva, smjer informacijske i podatkovne znanosti. 2019. godine odrađuje stručnu praksu u tvrtki Informatika Fortuno kao razvojni programer web aplikacija u Angular-u.

_____

Josip Faletar