Web stranica plesnog studija

Montibeler, Benjamin

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Stručni studij

Web stranica plesnog studija

Završni rad

Benjamin Montibeler

Osijek, 2023.

SADRŽAJ

1. UVOD

Najbitnije stavke prilikom vođenja plesnog studija su reklamiranje, upisi i kontakt. Pomoću modernih tehnologija za razvoj web stranica, moguće je kreirati mjesto na kojem će plesači moći pronaći najbitnije informacije o određenom plesnom studiju, mogućnost upisa i ispisa s određenog plesnog programa te praćenje upisanih programa. Također, vlasnici imaju mjesto za promociju novosti vezanih za plesni studio.

Cilj ovog završnog rada je opisati tehnologije i metode korištene pri izradu web stranice plesnog studija. Funkcionalnosti koje web stranica plesnog studija treba imati su mogućnost registracije ili prijave korisnika, upis ili ispis na određeni plesni stil, pregled naslovne stranice gdje se nalaze novosti i poveznice na ostale stranice. Na vrhu stranice nalazi se navigacijska traka pomoću koje korisnik može pristupiti ostalim podstranicama poput galerije, podstranice s ponuđenim plesnim programima, podstranice za registraciju i tako dalje. Nakon korištenja web stranice, korisnik se može odjaviti. Prijava na web stranicu nije nužna, osim u slučaju upisa.

U prvom dijelu rada su predstavljena slična rješenja zadatka, izrađena u drugim tehnologijama. Zatim su predstavljene i detaljno opisane korištene tehnologije i programski jezici, njihova svrha i implementacija. Naposljetku, objašnjen je proces izrade same aplikacije, njene funkcionalnosti i način korištenja.

1.1. Zadatak završnog rada

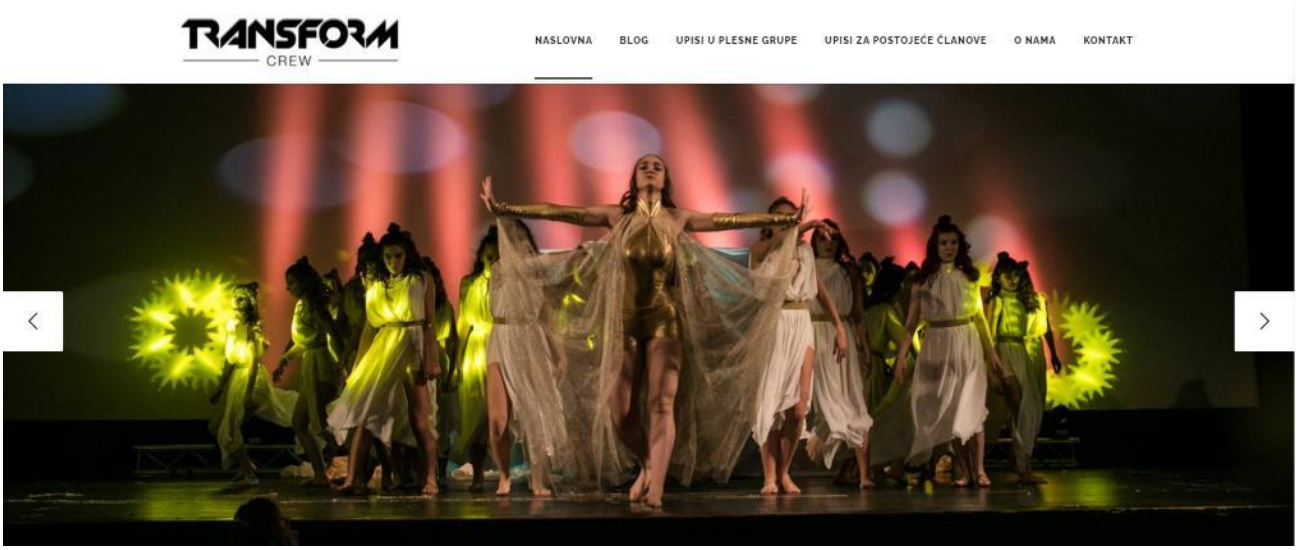
Izraditi web stranicu plesnog studija koristeći se React razvojnom okvirom pomoću koje korisnici mogu pratiti najnovije događaje u studiju, kao i mogućnost upisa u plesni studio, pregled plesnih programa i trenera, angažiranje plesača te pristup detaljnijim informacijama o studiju. Opisati postupak izrade web stranice i izrađene funkcionalnosti.

2. PREGLED PROBLEMATIKE I POSTOJEĆIH RJEŠENJA

Ubrzani razvoj interneta i web programiranja omogućio je mnogim poduzećima da otvore svoje digitalno sjedište pa tako i plesnim studijima. Jedno mjesto sa svim nužnim informacijama i mogućnosti digitalnog upisa na plesne programe uvelike je olakšao plesačima pronalazak idealnog plesnog doma, kao što je olakšao vlasnicima lakši pronalazak klijenata. U nastavku su predstavljena slična rješenja opisanog zadatka.

2.1. Plesni studio TransForm

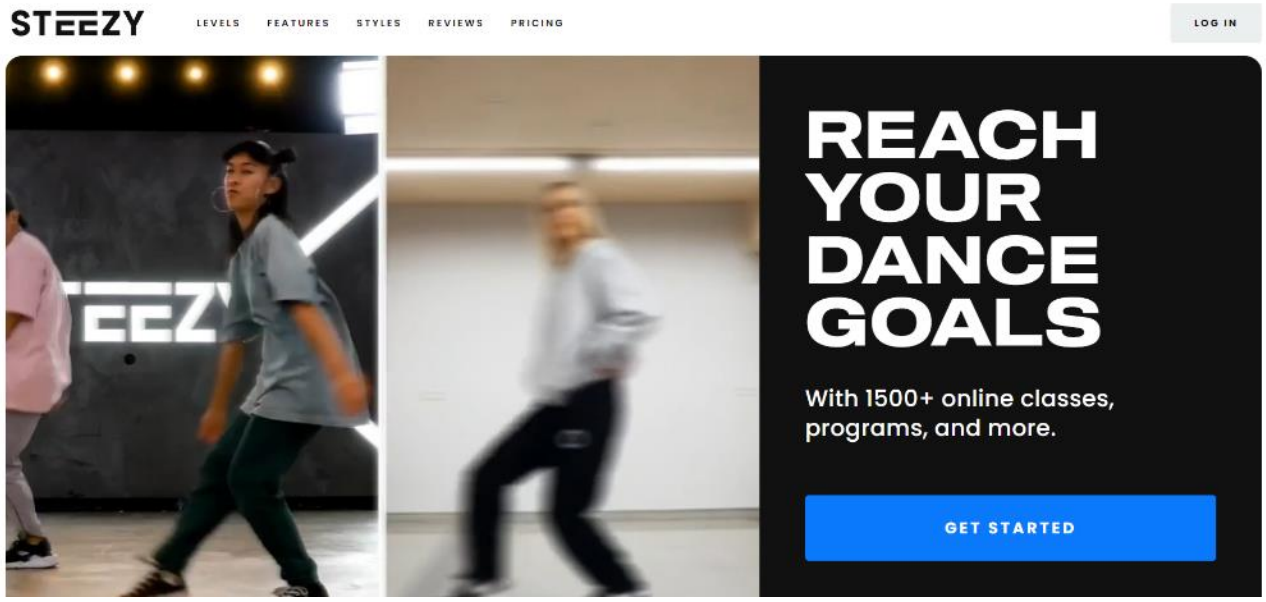
Web stranica plesnog studija TransForm odličan je primjer rješenja opisanog problema. Na naslovnoj stranici se vrte slike s događaja na kojima je plesni studio učestovao. Na vrhu stranice nalazi se navigacijska traka pomoću koje korisnik može pristupiti podstranicama za upis u plesne grupe, podstranici „O nama" s detaljnijim informacijama studija kao i podstranici za kontakt [1]. Prilikom prelaska pokazivačem preko hiperveza navigacijske trake, aktivira se animacija. Zastupljenost animacija na web stranicama pridonosi profesionalnijem izgledu što uvelike privlači pažnju korisnika. Navigacijska traka i naslovna stranica prikazani su na slici 2.1. (https://transform.hr/) u nastavku.



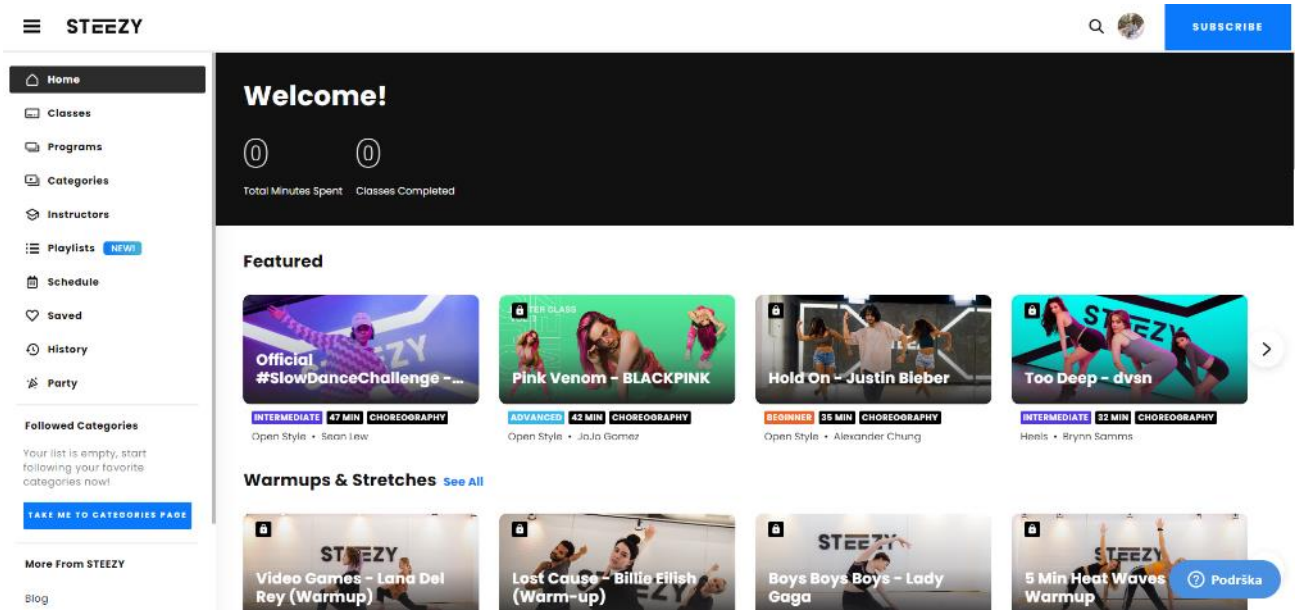
Sl. 2.1. *Prikaz naslovne stranice plesnog studija TransForm*

2.2. Plesni studio STEEZY

Naslovna stranica plesnog studija STEEZY sadrži video plesača koji se vrti u beskonaćnoj petlji. S desne strane videa nalazi se gumb koji pritiskom vodi na stranicu za upis. Kao kod web stranice plesnog studija TransForm, na vrhu se nalazi navigacijska traka. S krajnje desne strane trake nalazi se gumb za prijavu, odnosno registraciju na stranicu. Nakon prijave korisnik je preusmjeren na podstranicu za korisnički profil gdje je moguće pregledati razne plesne stilove i najvažnije informacije o svakom plesnom kursu. S lijeve strane prikazan je izbornik za lakše snalaženje i pretraživanje plesnih tečaja. Prelaskom pokazivača preko određenog tečaja, pokreće se video prikaz plesa podučavanog na tom tečaju [2]. Na slikama 2.2. i 2.3. prikazan je opisan izgled stranice (https://www.steezy.co/) .



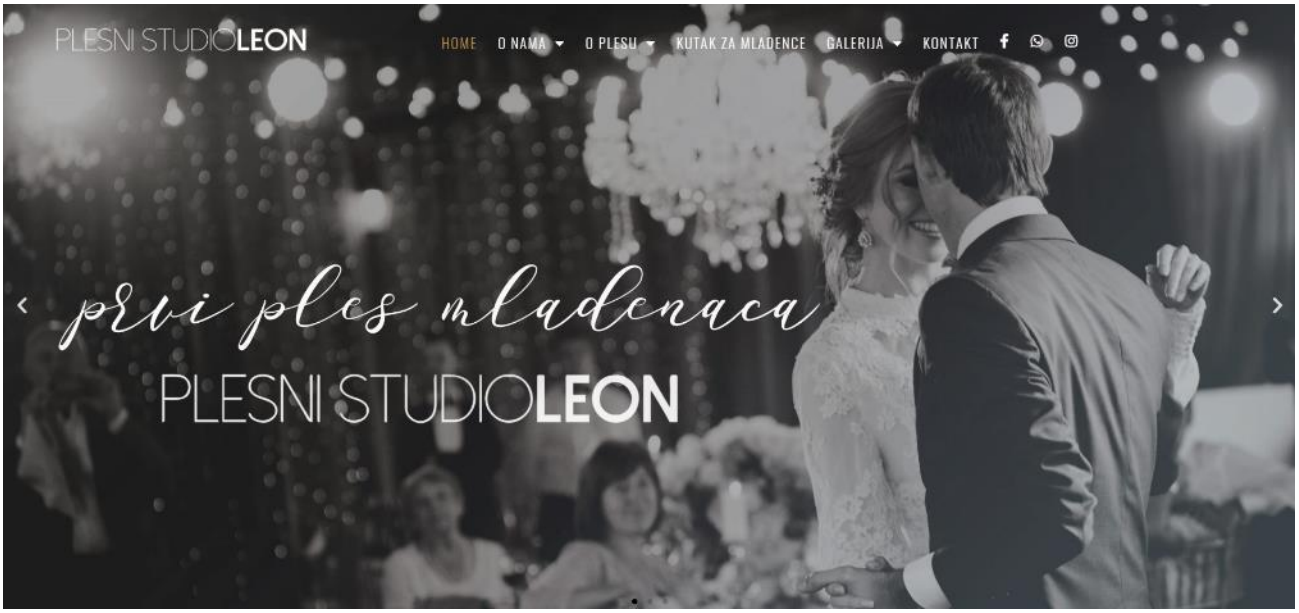
Sl. 2.2. *Prikaz naslovne stranice plesnog studija STEEZY*



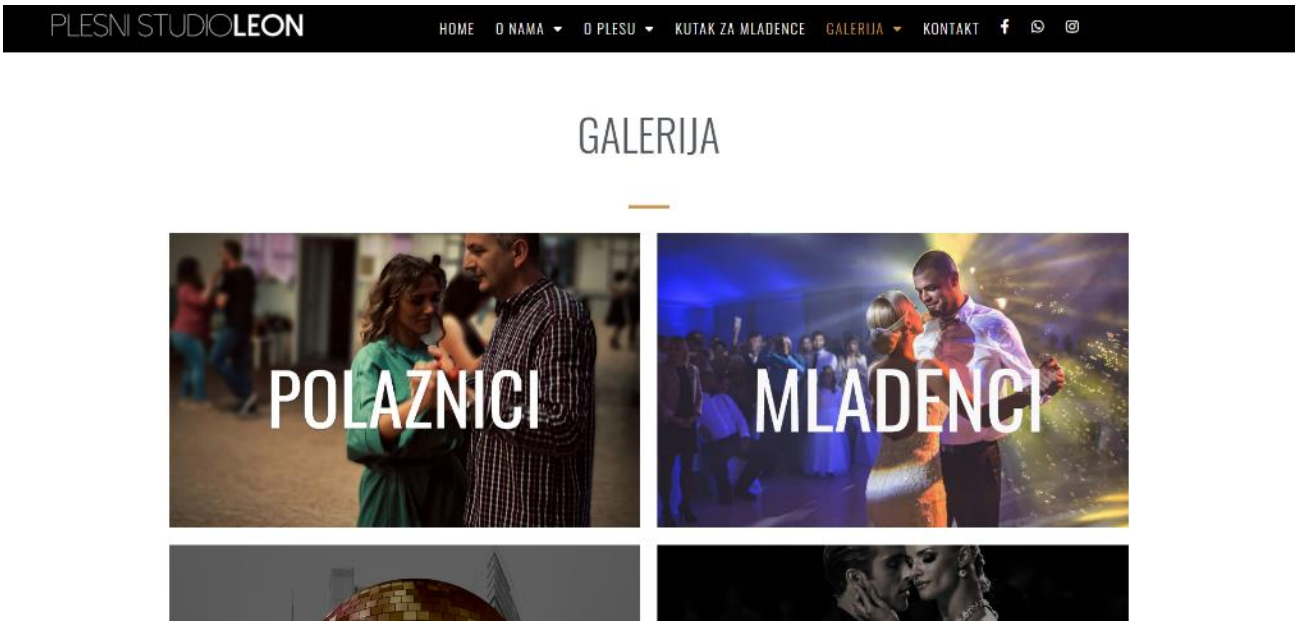
Sl. 2.3. *Prikaz stranice korisničkog profila plesnog kluba STEEZY*

2.3. Plesni studio Leon

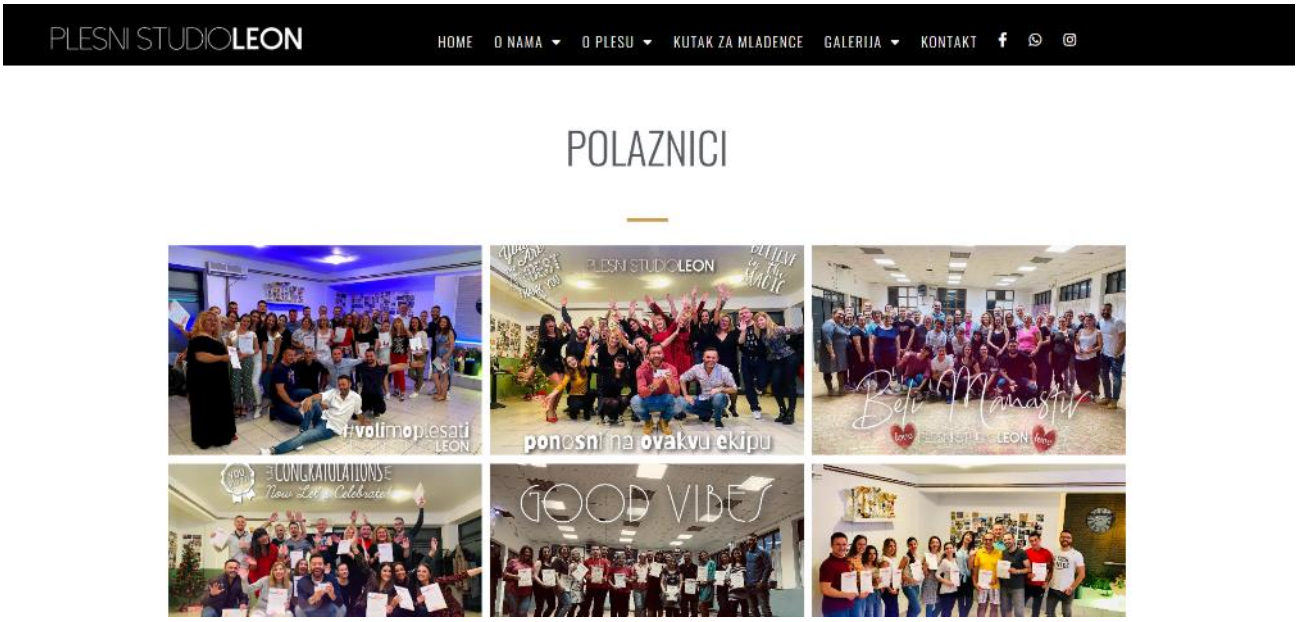
Na naslovnoj stranici plesnog studija Leon moguće je listati prikazane slike plesnih stilova kojima se studio bavi. Poveznice na ostale podstranice s određenim informacijama također su vidljive na naslovnoj stranici. Navigacijska traka sastoji se od poveznica na ostale podstranice te poveznica za društvene mreže studija. Na podstranici galerije nalazi se izbornik albuma. Odabirom određenog albuma, kroz animaciju, prikazuju se slike plesača [3]. Na slici 2.4. nalazi se naslovna stranica, dok je na slikama 2.5. i 2.6. prikazana galerija plesnog studija. (https://plesni.studio/) prikazana je u nastavku.



Sl. 2.4. *Prikaz naslovne stranice plesnog studija Leon*



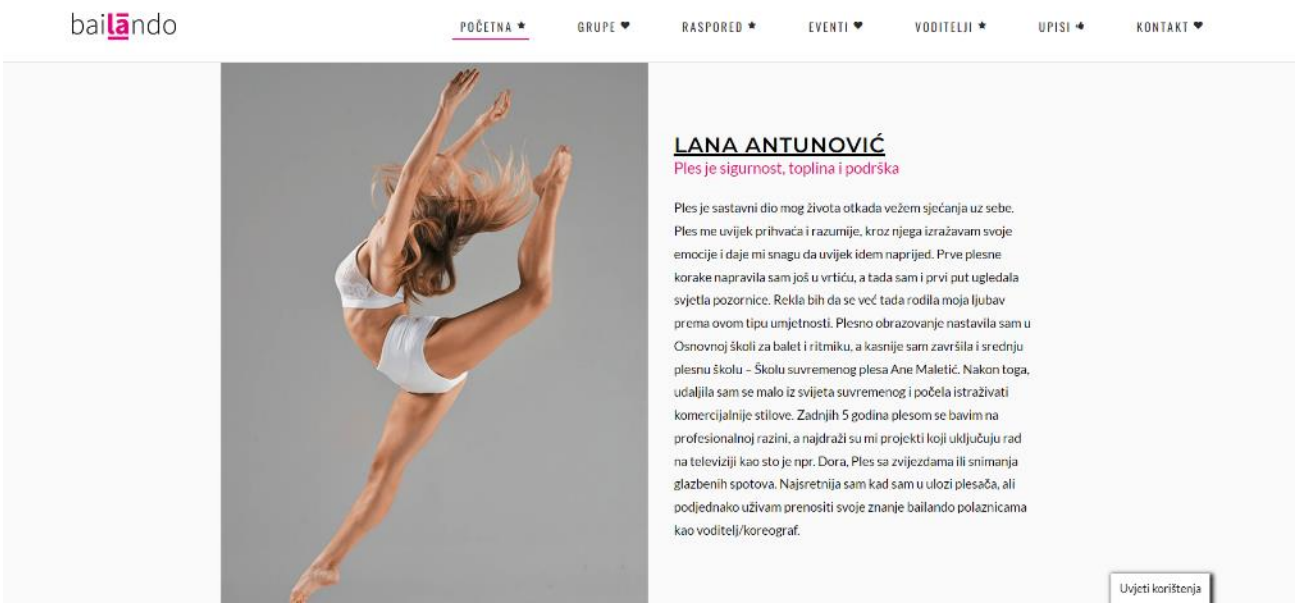
Sl. 2.5. *Prikaz podstranice galerije plesnog studija Leon*



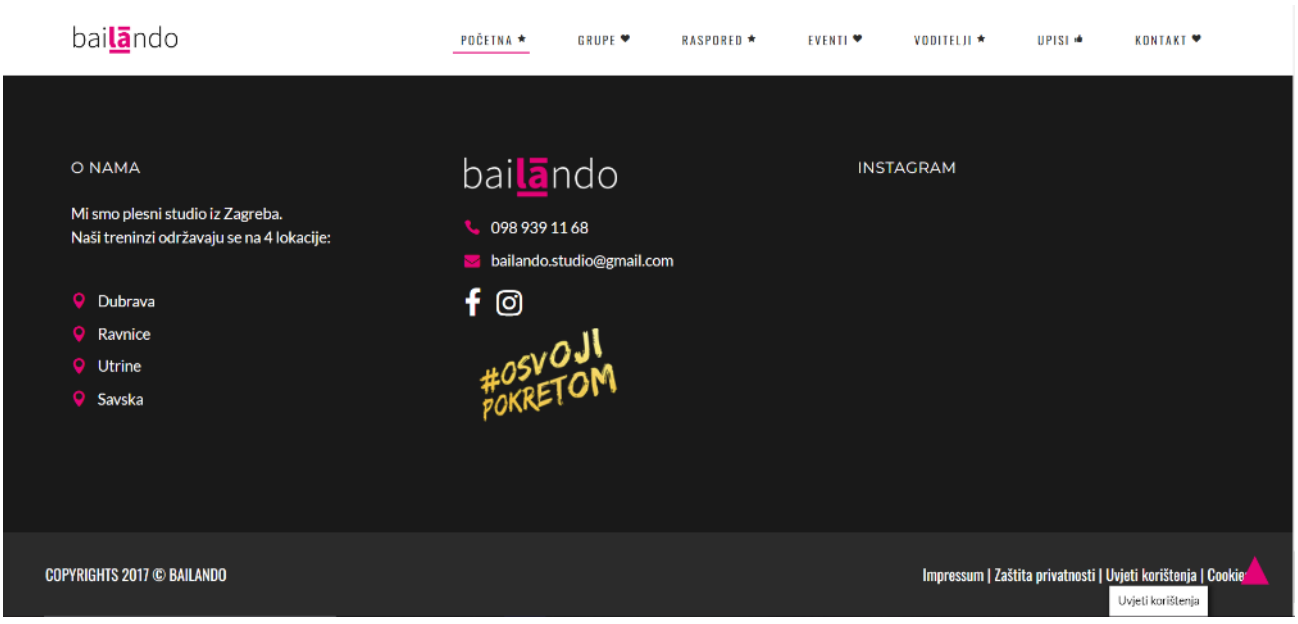
Sl. 2.6. *Prikaz podstranice određenog albuma galerije*

2.4. Plesni studio Bailando

Naslovna stranica plesnog studija Bailando sastoji se od dijaprojekcije slika plesača i gumbom za kontakt. Ispod dijaprojekcije slika nalazi se popis svih plesnih stilova kluba s detaljnim opisom. Također, na stranici se nalazi gumb koji vodi na podstranicu za izbor plesne grupe te slike i biografija svakog od trenera. Sve navedene funkcije pojavljuju se kroz razne animacije. Na dnu stranice nalazi se *footer* s najosnovnijim informacijama i vezama, a na vrhu navigacijska traka s vezama na podstranice „Grupe", „Raspored", „Eventi", „Voditelji", „Upisi" i „Kontakt" [4]. Na slici 2.7. prikazan je izgled biografije trenera dok je na slici 2.8. prikazan footer stranice. (https://bailando.hr/) u nastavku.



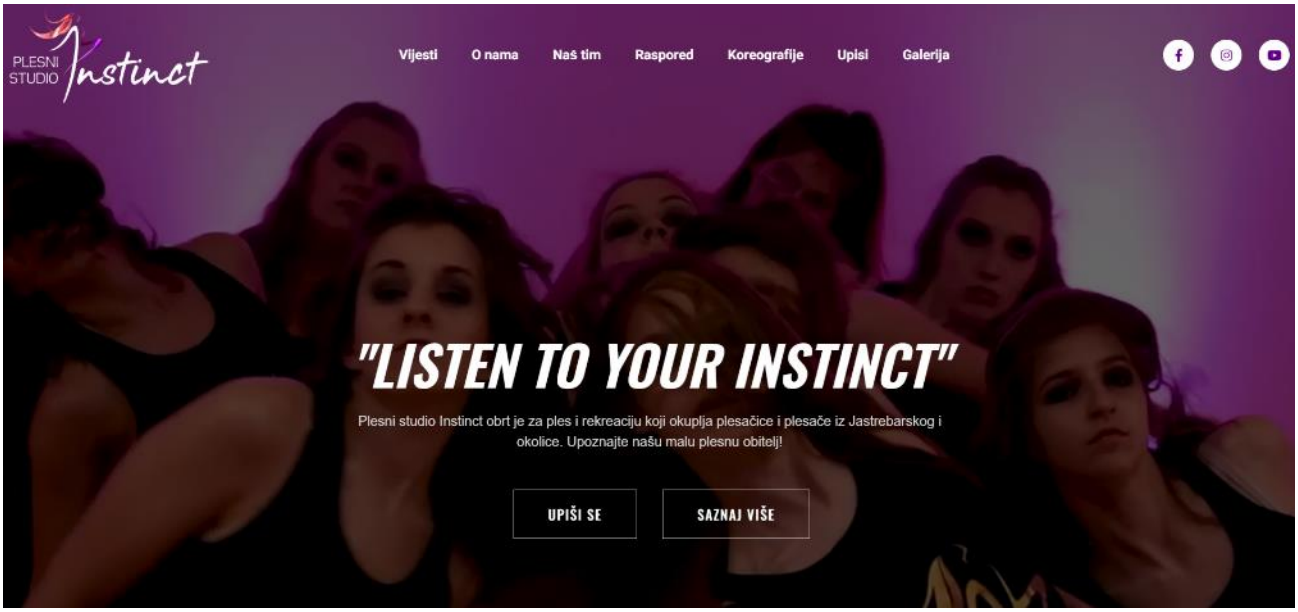
Sl. 2.7. *Prikaz biografije trenera na naslovnoj stranici plesnog studija Bailando*



Sl. 2.8. *Prikaz footer-a plesnog studija Bailando*

2.5. Plesni studio Instinct

Kao i kod plesnog studija STEEZY, na web stranici plesnog studija Instinct (https://plesnistudioinstinct.com/) također se nalazi video plesača. Na navigacijskoj traci, uz logo, nalaze se slične poveznice kao i kod drugih plesnih studija. Pomoću gumba „Upiši se", centriranog na sredini videa, možemo se preusmjeriti na podstranicu za upis u plesni klub, a s gumbom „Saznaj više" automatski smo pomaknuti na sekciju naslovne stranice s detaljnim opisom plesnog studija [5]. Kod ove web aplikacije, kao i kod već spomenutih, korištene su animacije pri listanju kako bi se dodatno obuhvatila pažnja korisnika. Na slici 2.9. prikazan je izgled navigacijske trake i vrha naslovne stranice, a slikom 2.10. opis studija.



Sl. 2.9. *Prikaz navigacijske trake i naslovne stranice plesnog studija Instinct*

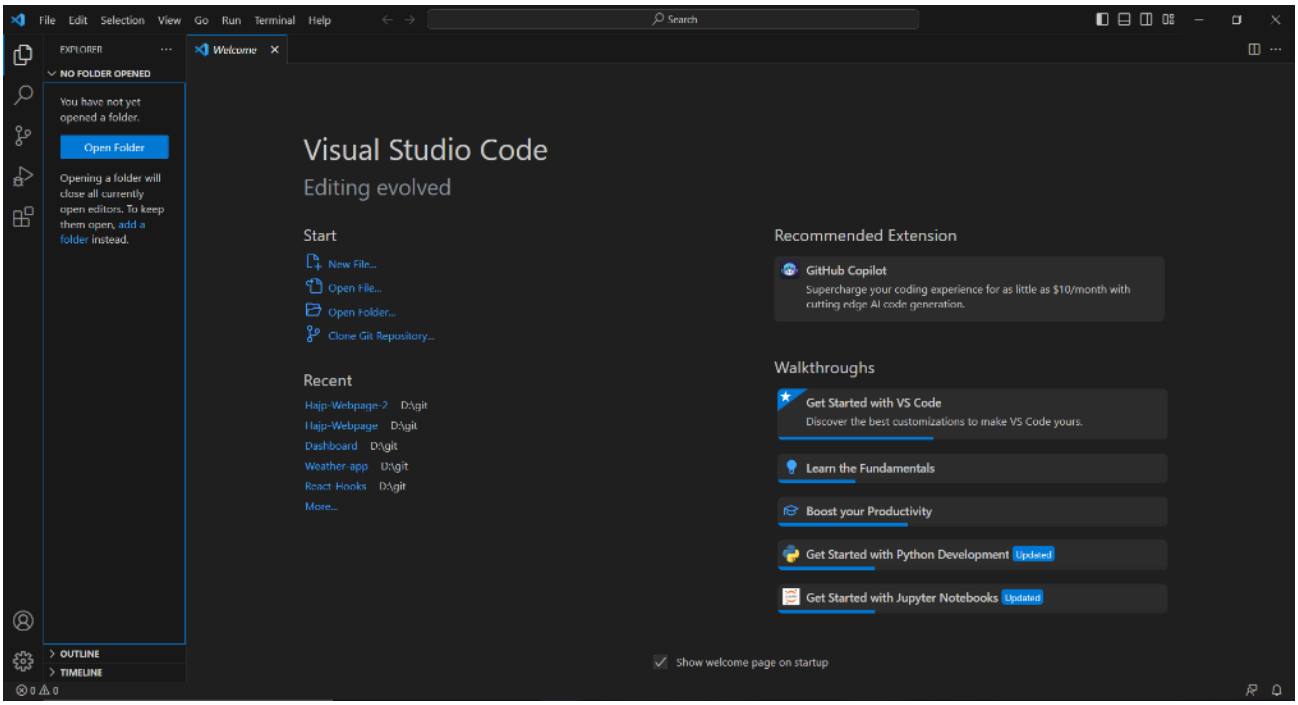
Sl. 2.10. *Prikaz sekcije ,,O nama" naslovne stranice plesnog studija Instinct*

3. TEHNOLOGIJE KORIŠTENE ZA IZRADU WEB APLIKACIJE

Za izradu funkcionalne web aplikacije potrebni su razni alati i tehnologije, počevši od programa za uređivanje teksta do programskih jezika i *framework*-a. Web programiranje je kompleksno te se sastoji od razvoja korisničkog sučelja tj. *front-end* dijela aplikacije, i izgradnje baze podataka, odnosno *back-end* dijela. U nastavku su opisane sve tehnologije korištene za realizaciju projekta .

3.1. Visual Studio Code

Visual Studio Code je besplatan uređivač izvornog koda dostupan za Windows, macOS, Linux i Raspberry Pi OS, razvijen od strane Microsoft-a. Podržava proširenja za mnoge programske jezike poput C++, C#, Java, JavaScript, Python, Typescript, PHP, Go, CSS te omogućuje rad raznih tehnologija kao što su React i Node.js. VS code ima *IntelliSense* dovršavanje koda za varijable i metode, otklanjanje pogrešaka unutar koda i druge moćne funkcije uređivanja. Također je ugrađena Git podrška. Pri izradi softvera, Microsoft se koristio Electron shell-om, Node.js-om i TypeScript-om te je ažuriran na mjesečnoj bazi. Slikom 3.1. prikazano je sučelje VS Code-a [6].



Sl. 3.1. *Prikaz Visual Studio Code sučelja*

3.2. HTML – HyperText Markup Language

HTML ili HyperText Markup Language je najosnovniji građevni element svake web stranice. Definira strukturu i značenje stranice koristeći *markup* za označavanje teksta, slika, videa i ostalih sadržaja. Markup se sasatoji od posebnih elemenata kao *<head>*, *<title>*, *<body>*, *<header>*, *<footer>*, *<article>*, *<section>*, *<p>*, * *, **, *<img>*, *<aside>*, *<audio>*, *<canvas>*, *<datalist>*, *<details>*, *<embed>*, *<nav>*, *<output>*, *<progress>*, *<video>*, *<ul>*, *<ol>*, *<li>* i mnogih drugih. Elementi su omeđeni izlomljenim zagradama kako bi ih se odvojilo od ostatka teksta. Naziv elementa unutar zagrada nije osjetljiv na veliko i malo slovo, što znaći da element * * možemo napisati kao *<Div>* i *<DIV>* ili bilo koji drugi način. Međutim, konvencija i preporučena praksa je pisanje oznaka malim slovima [7].

3.3. CSS - Cascading Style Sheets

CSS je stilski programski jezik korišten za definiranje izgleda elemenata napisanih u HTML-u. Standardiziran za sve web preglednike, postao je jedan od najvažnijih i najosnovnijih jezika web programiranja [8]. Sintaksa CSS-a sastoji se od tri dijela prikazanih na slici 3.2.

```
selektor {
    svojstvo: vrijednost;
}
```

Sl. 3.2. *Prikaz sintakse CSS programskog jezika*

Selektor je HTML element ili klasa kojem želimo dodijeliti određenu vrijednost svojstva. Svojstvo je postojeća, predefinirana oznaka elementa koja se mijenja u odnosu na zadanu vrijednost. Selektor se može sastojati od više svojstava, svojstvo i vrijednost odvajamo dvotočkom, a iza vrijednosti nalazi se točka-zarez koja međusobno odvaja svojstva. Višestruke vrijednosti jednog svojstva odvajaju se zarezom. Svojstva svakog selektora nalaze se u vitičastim zagradama, kao što je prikazano na slici 3.2. Također, možemo dodijeliti iste vrijednosti svojstava više selektora tako da selektore napišemo jedne iza drugih te ih odvojimo zarezom [9].

3.4. JavaScript

JavaScript ili JS je skriptni programski jezik pomoću kojeg programer statičnu stranicu pretvara u dinamičnu. Upravo se JS-om gradi interaktivnost, odnosno međudjelovanje web aplikacije s korisnikom. Može se izvoditi na klijentskom računalu i svim modernim web preglednicima. JavaScript smanjuje potrebu komunikacije aplikacije sa serverom jer može provjeriti ispravnost podataka prije njihovog slanja, tako smanjujući promet i potrošnju resursa. Također je moguće izraditi animacije pridonoseći bogatije sučelje. Implementacija JavaScript-a moguća je na tri načina: interne skripte, vanjski JS kod i kod unutar HTML-a. JavaScriptom je moguće definirati varijable, polja, funkcije, petlje, objekte, klase i tako dalje [10]. Slikom 3.3. predstavljena je sintakta jednostavne petlje [11].

```
const cars = ["BMW", "Volvo", "Saab", "Ford"];
let i = 0;
let text = "";

while (cars[i]) {
  text += cars[i];
  i++;
}
```

Sl. 3.3. *Primjer petlje napisane JavaScript-om*

3.5. Bootstrap

Bootstrap je besplatan *front-end* razvojni framework, dizajniran kako bi omogućio lakšu implementaciju responzivnosti web aplikacijama. Izrađen je uz pomoć HTML-a, CSS-a i JavaScripta te pomaže programerima pri bržem pisanju koda, eliminirajući potrebu za ručnim unosom osnovnih CSS i JavaScript funkcija [12]. Bootstrap je odličan za početnike koji se prvi put susreću s web programiranjem. Postoje brojne komponente unutar Bootstrap-a kao što su navigacijske trake, trake napretka, upozorenja, padajući izbornici i slično. Na slici 3.4. prikazana je implementacija Bootstrap-a unutar HTML koda [13].

```
<div class="container">
  <div class="row">
    <div class="col">
      Column
    </div>
    <div class="col">
      Column
    </div>
    <div class="col">
      Column
    </div>
  </div>
</div>
```

Sl. 3.4. *Prikaz implementacije Bootstrap-a unutar HTML koda*

3.6. React

React je biblioteka za izgradnju korisničkog sučelja web stranice, uz limitirane *back-end* funkcionalnosti. U kombinaciji s ostalim bibliotekama, može se koristiti i za razvoj u drugim okruženjima, kao što je React Native za mobilne aplikacije. Glavni cilj React-a je minimizirati pogreške koje se javljaju kada programeri kreiraju korisnička sučelja. To se postiže upotrebom komponenti. Komponente su samostalni, logični dijelovi koda koji opisuju dio korisničkog sučelja te se mogu kombinirati kako bi se stvorilo cjelovito korisničko sučelje. React koristi moderne JavaScript funkcije u mnogim svojim oblicima. Glavno odstupanje od JavaScripta je korištenje JSX sintakse. JSX proširuje sintaksu JavaScripta tako da kôd HTML-a može koegzistirati. Primjer JSX koda prikazan je slikom 3.5.

```jsx
const header = (
  <header>
    <h1>Mozilla Developer Network</h1>
  </header>
);
```

Sl. 3.5. *Prikaz JSX React koda*

Postoji mnogo načina za instalaciju React-a, a jedan od njih je korištenje *comand-line interface-a* (CLI) to jest sučelja naredbenog retka. CLI alat *create-react-app* ubrzava proces razvoja React aplikacije instaliranjem paketa i stvaranjem datoteka umjesto programera. Nakon procesa kreiranja aplikacije, naredbom *npm start* aplikacija se pokreće u web pregledniku. Moguće je implementirati kodove iz drugih datoteka pomoću *import* funkcije, tako primjerice možemo uređivati JSX kod pomoću CSS programskog jezika. React aplikacija može imati više datoteka s kodom, ali glavna datoteka, koja treba imati poveznicu na ostale, je App.js. Na slici 3.6. predočen je jednostavan izgled App.js datoteke [14].

```jsx
function App() {
  const subject = "React";
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>Hello, World!</p>
      </header>
    </div>
  );
}
```

Sl. 3.6. *Jednostavan primjer koda App.js datoteke*

3.7. Firebase

Firebase je *back-end* usluga (BaaS) koja programerima pruža niz alata i usluga za stvaranje visokokvalitetnih aplikacija i korisničkih baza. Temelji se na Google-ovoj infrastrukturi. Firebase je klasificiran kao NoSQL program baze podataka koji pohranjuje podatke u dokumente slične JSON-u [15]. Na slici 3.7. vidljivo je sučelje Firebase web stranice.

3.7.1. Autentifikacija

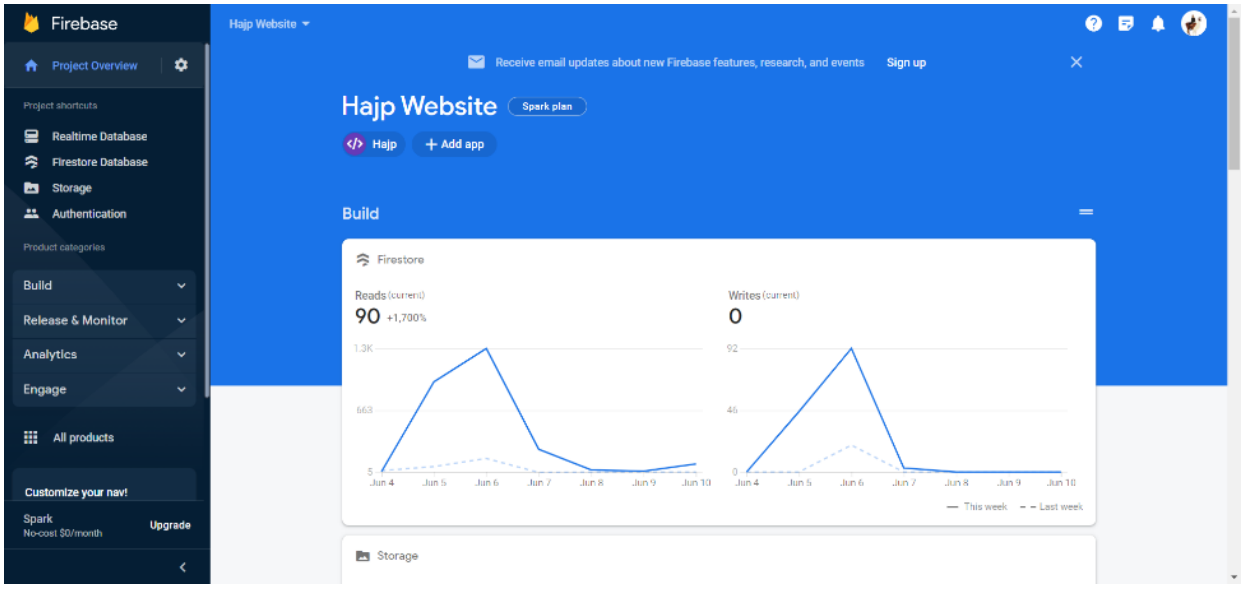
Firebase podržava autentifikaciju pomoću lozinki, telefonskih brojeva, Googlea, Facebooka, Twittera i više. Firebase Authentication (SDK) omogućuje ručno integriranje jedne ili više metoda prijave u aplikaciju [15].

3.7.2. Storage

Firebase Storage je spremnik za pohranu datoteka, poput slika i videa, što daje mogućnost prijenosa i preuzimanja datoteka kroz mobilne i web aplikacije [16].

3.7.3. Firestore baza podataka

Firestore je NoSQL baza podataka dizajnirana za jednostavan razvoj aplikacija. Iako Firestore sučelje dijeli mnoge iste karakteristike kao tradicionalne baze podataka, kao NoSQL baza podataka razlikuje se od njih po načinu na koji opisuje odnos između podatkovnih objekata [17].



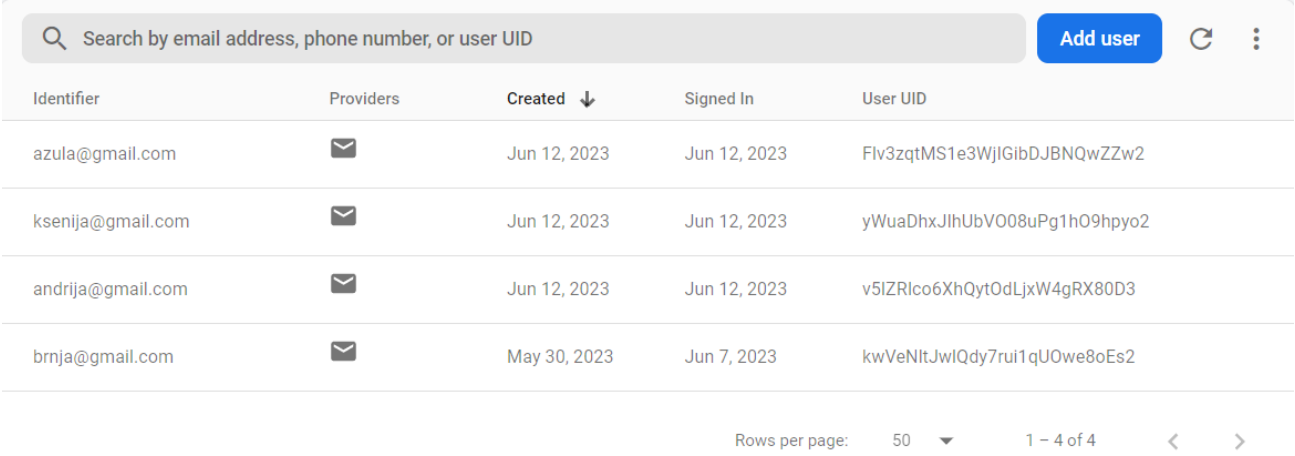
Sl. 3.7. *Prikaz Firebase sučelja*

4. IZRADA APLIKACIJE

Izradu aplikacije web aplikacije sastoji se od dva dijela, baze podataka i dizajna stranice. U ovom poglavlju, najprije je opisana struktura Firebase baze korištene za projekt. Nadalje, detaljno je objašnjen kod svake podstranice projekta.

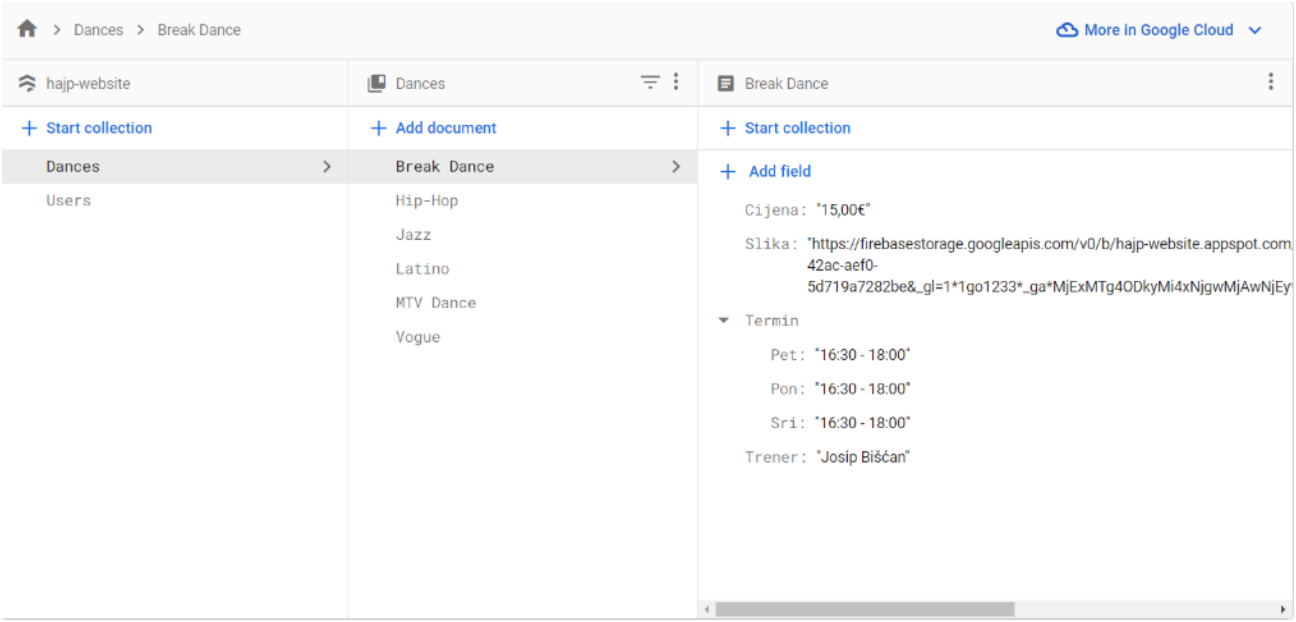
4.1. Struktura korištenih Firebase funkcionalnosti

Pri registraciji i prijavi korisnika korištena je takozvana Authentication funkcionalnost Firebase sustava. Pri izradi baze, ponuđen je izbornik s opcijama registracije poput registracija korisnika putem Facebook-a, Twittera i slično, no odabrana je jednostavna registracija putem e-pošte. Nakon aktivacije, potrebno je samo spojiti React aplikaciju s Firebase-om. Slikom 4.1. prikazan je izgled registriranih korisnika.



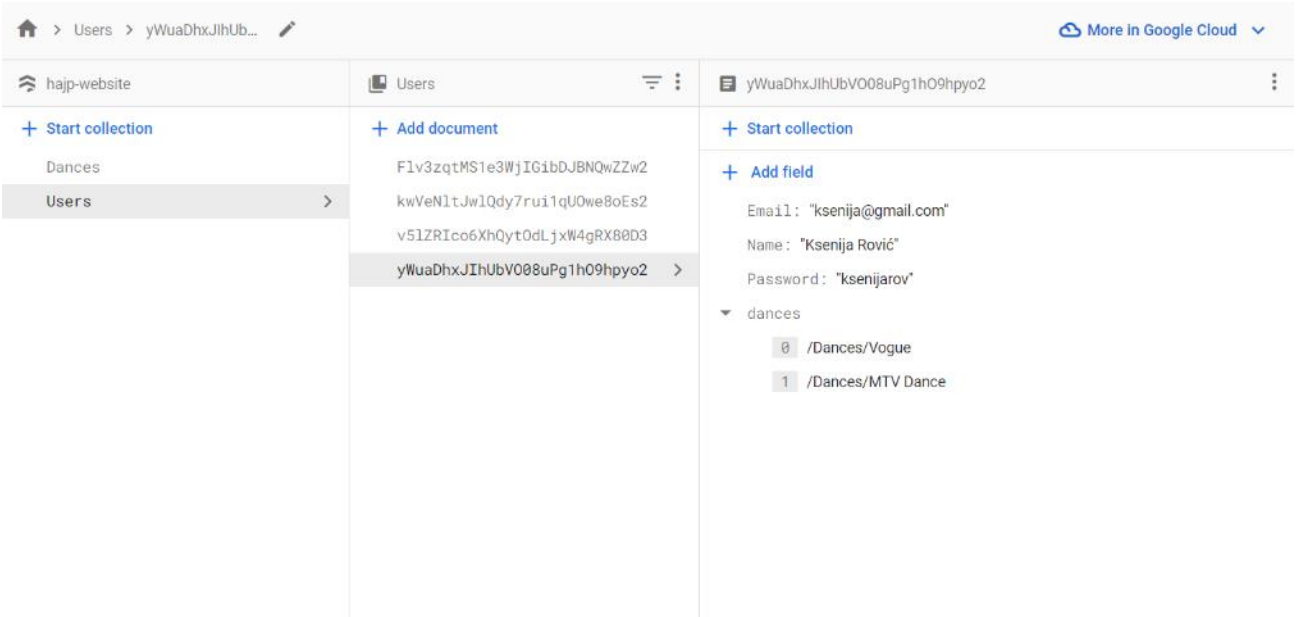
Sl. 4.1. *Prikaz registriranih korisnika u Firebase Authentication bazu podataka*

Kako bi bilo moguće stvoriti referencu na upisani plesni program, potrebno je te iste korisnike spremiti u Firestore bazu podataka, zajedno sa svim dostupnim plesnim stilovima. Stvorene su dvije kolekcije: *Users* i *Dances*. Unutar *Dances* kolekcije nalaze se dokumenti plesnih programa s varijablama Cijena, Trener, Slika i poljem Termin. Varijabla Slika sadrži link slike kojom je prikazan plesni stil, a slika je pohranjena u Firebase Storage spremniku. Na slici 4.2. nalazi se izgled opisane baze.



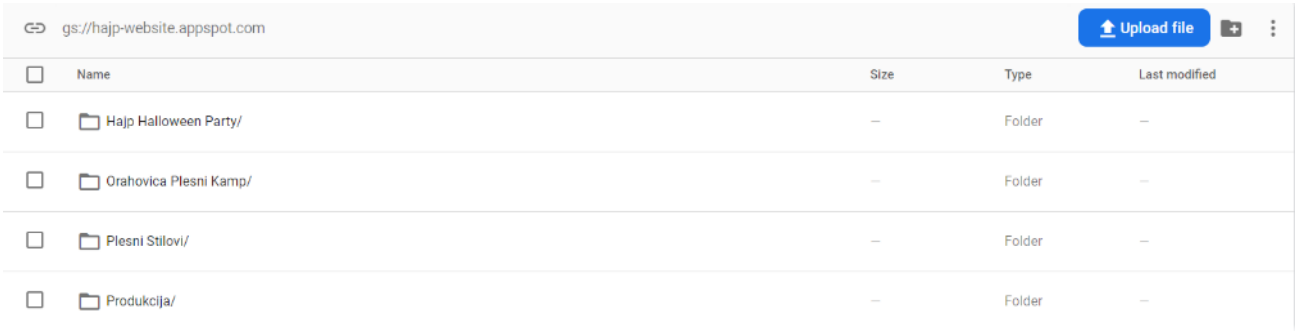
Sl. 4.2. *Prikaz plesnih programa unutar Firebase Firestore baze podataka*

U kolekciji *Users* pohranjeni su svi registrirani korisnici aplikacije. Naziv svakog korisničkog dokumenta unutar kolekcije je identifikacijski ključ tog korisnika. Varijablama *Email*, *Name* i *Password* pohranjeni su e-pošta, ime i lozinka korisnika. Prilikom upisa na određeni program, kreira se polje *dances* u kojem je pohranjena putanja do određenog plesnog stila u sljedećem obliku: *Dances/Break Dance*. Slikom 4.3. prikazana je opisana kolekcija.



Sl. 4.3. *Prikaz korisnika unutar Firebase Firestore baze podataka*

Firebase Storage spremnik služi za spremanje datoteka, idealan za pohranu slika. Kako bi slike preglednije prikazali na stranici, podijelili smo ih u albume. Slikom 4.4. prikazan je izgled albuma unutar spremnika.



Sl. 4.4. *Prikaz albuma unutar Firebase Storage spremnika*

4.1.1. Implementacija Firebase-a u React aplikaciju

Prije svega, potrebno je instalirati Firebase programsko sučelje za aplikacije ili *API*, koristeći se npm naredbom *npm install firebase*. Zatim treba inicijalizirati Firebase, što je napravljeno u posebnoj JavaScript datoteci, prikazano slikom 4.5. Većinski kod automatski je formiran na službenoj Firebase stranici.

```
import firebase from 'firebase/compat/app';
import 'firebase/compat/firestore';
import 'firebase/compat/auth'
import 'firebase/compat/storage';


const firebaseConfig = {
    apiKey: "AIzaSyDLkLcuyMx_KHHyA16XZHClTw4YeJCXNuA",
    authDomain: "hajp-website.firebaseapp.com",
    projectId: "hajp-website",
    storageBucket: "hajp-website.appspot.com",
    messagingSenderId: "162467470616",
    appId: "1:162467470616:web:6e46bbe1272bb6b205526d",
    measurementId: "G-82RWBC9RTJ",
    storageBucket: 'hajp-website.appspot.com'
  };


firebase.initializeApp(firebaseConfig);

const projectFirestore = firebase.firestore();

const projectStorage = firebase.storage();

const projectAuth = firebase.auth();

export { projectFirestore, projectStorage, projectAuth };
```

Sl. 4.5. *Prikaz koda unutar firebase.js datoteke*

Varijablama *projectFirestore*, *projectStorage* i *projectAuth* inicijalizirane su baza podataka, spremnik i autentifikacija.

4.2. Izrada navigacijske trake

Navigacijska traka uvjerljivo je najbitniji dio svake web aplikacije jer omogućuje korisniku pregled svih podstranica. Navigacijska traka plesnog studija sastojat će se od loga, koji je ujedno i poveznica na naslovnu stranicu, podstranice za plesne programe i upise, podstranicu za trenere, galeriju, booking, podstranicu s informacijama o studiju kao i poveznicu na registraciju. Dodane su i određene, manje bitne stranice, čija je funkcija pružiti što više detalja o studiju te zainteresirati korisnika. Slikom 4.6. predstavljen je React JSX kod za usmjeravanje na podstranice.

```
<Link to="/Programi"><span>Programi i Upisi</span></Link>
```

Sl. 4.6. *Prikaz JSX koda za poveznicu na određenu podstranicu*

Komponenta <*Link*> dio je React Router biblioteke, koja se obično koristi za upravljanje navigacijom u React aplikacijama. Omogućuje deklarativan način stvaranja veza između različitih ruta. Komponenta <*Link*> renderira poveznicu <*a*>. Kada korisnik klikne na poveznicu, umjesto da pokrene ponovno učitavanje cijele stranice, React Router presreće događaj i ažurira URL aplikacije bez osvježavanja stranice. To omogućuje lakšu primjenu na jednoj stranici. Komponenta <*Link*> prihvaća *to* atribut, koji navodi ciljni URL ili rutu do koje bi poveznica trebala voditi. Može se dati kao niz koji predstavlja URL stazu ili kao objekt s dodatnim svojstvima.

S obzirom da se navigacijska traka mijenja ukoliko je korisnik prijavljen, potrebno je implementirati kod koji će provjeriti to stanje. Kod se nalazi na slici 4.7.

```
const [korisnik, postaviKorisnik] = useState(null);
const usmjeri = useNavigate();
useEffect(() => {
    const provjeriKorisnika = projectAuth.onAuthStateChanged((korisnik) => {
        if (korisnik) {
            postaviKorisnik(korisnik);
        } else {
            postaviKorisnik(null);
        }
    });

    return () => provjeriKorisnika();
}, []);

const odjavi = () => {
    projectAuth.signOut().then(() => {
        usmjeri('/Login');
    });
};
```

Sl. 4.7. *Prikaz koda za provjeru stanja korisnika i odjavu*

Varijabla korisnik te funkcija kojom se postavlja vrijednost varijable deklarirane su metodom *useState*. React metoda *useEffect* se koristi za opažanje promjena, a u slučaju prijave koristi se za promatranje promjena Firebase autentifikacije. Izvršava se kada je komponenta montirana i ima prazan niz ovisnosti, označen s "[]", što ukazuje da se pokreće samo jednom. Unutar *useEffect* metode, postavljen je slušatelj za događaj *onAuthStateChanged*. Taj se događaj pokreće kad god se promijeni stanje autentifikacije korisnika (npr. prijava ili odjava). Funkcija povratnog poziva *(korisnik) => {},* poziva se kada se pokrene događaj *onAuthStateChanged*. Prima objekt korisnika koji predstavlja trenutno autentificiranog korisnika ili *null* vrijednost ako korisnik nije autentificiran. Unutar funkcije nalazi se izjava *if else*, koja provjerava postoji li prijavljen korisnik. Ako je korisnik prijavljen, poziva se funkcija *postaviKorisnika(korisnik)* koja ažurira varijablu *korisnik* s autentificiranim korisničkim objektom. Ako korisnik nije prijavljen, varijabla se postavlja na null. Linija koda *return () => provjeriKorisnika();* koristi se kao funkcija čišćenja za *useEffect*. Poziva se kada se komponenta demontira. Unutar funkcije čišćenja se poziva *provjeriKorisnika()* da odvoji slušatelja za događaj *onAuthStateChanged*, osiguravajući da se ne primaju daljnja ažuriranja.

Mogućnost odjave će se isto nalaziti na navigacijskoj traci. Funkcija odjave pozvat će se na pritisak gumba. Unutar funkcije odjave pozvana je Firebase metoda odjave *signOut()*. Nakon što se korisnik uspješno odjavi biti će usmjeren na podstranicu korisničkog profila koju je omogućila varijabla

usmjeri, deklarirana pomoću *useNavigate* metode iz React biblioteke. Sve što je sada potrebno je dodati JSX kod za provjeru stanja korisnika, kao što je prikazano slikom 4.8.

```
{!korisnik &&
    <Link to="/Login"><span>Prijavi se</span></Link>
}
{korisnik &&
    <div className='logged-div'>
        <Link to="/Profile"><FaUser /></Link>
        <a className='logout-btn' onClick={odjavi}><span>Odjavi se</span></a>
    </div>
}
```

Sl. 4.8. *Prikaz JSX koda kojim je provjereno stanje korisnika*

4.3. Izrada funkcionalnosti registracije i prijave

Kako bi se korisnici mogli upisati na plesne programe, potrebno je izraditi formu za registraciju, odnosno prijavu. Najprije treba deklarirati varijable koje će pohraniti korisničke podatke u bazu podataka. Metodom *useState*, deklarirana je varijabla i pripadajuća joj funkcija koja će tu varijablu postaviti na određenu vrijednost. Za početak, varijable su postavljene na prazan *string*. Kod se nalazi na slici 4.9.

```
const usmjeri = useNavigate();
const [ime, postaviIme] = useState('');
const [email, postaviEmail] = useState('');
const [lozinka, postaviLozinka] = useState('');
const [greskaRegistracija, postaviGreskaRegistracija] = useState('');
const [greskaPrijava, postaviGreskaPrijava] = useState('');
```

Sl. 4.9. *Prikaz deklaracije varijabli potrebnih za izradu registracijske forme*

Varijabla *usmjeri*, deklariranu pomoću *useNavigate* funkcije iz React biblioteke, potrebna je za preusmjeravanje korisnika na podstranicu profila nakon registracije ili prijave. U slučaju da dođe do pogrešaka pri registraciji i prijavi, varijablama *greskaRegistracija* i *greskaPrijava* biti će prikazane.

Slikom 4.10. prikazana je funkcija za registraciju korisnika.

```
const Registracija = (e) => {
    e.preventDefault();
    projectAuth.createUserWithEmailAndPassword(email, lozinka).then((cred) => {
        projectFirestore.collection('Users').doc(cred.user.uid).set({
            Name: ime,
            Email: email,
            Password: lozinka
        }).then(() => {
            postaviIme('');
            postaviEmail('');
            postaviLozinka('');
            postaviGreskaRegistracija('');
            usmjeri('/Profile');
        }).catch(err => setErrorReg(err.message));
    }).catch(err => setErrorReg(err.message));
}
```

Sl. 4.10. *Prikaz funkcije za registraciju korisnika*

Funkcija *e.preventDefault();,* pozvana na vrhu funkcije, sprječava klasično ponašanje podnošenja obrasca, što bi uzrokovalo ponovno učitavanje stranice. Linija koda ispod pozvane funkcije, koristi *createUserWithEmailAndPassword* metodu Firebase autentifikacije za stvaranje novog korisničkog računa pomoću navedene e-pošte i lozinke. Vraća obećanje koje se rješava s objektom vjerodajnice, *cred,* nakon uspješne registracije. Metodom *collection(),* dohvaća se željena kolekcija unutar baze podataka te se u njoj kreira novi dokument s jednistvenim identifikatorom korisnika i njegovim podacima. Ako je kreiranje Firestore dokumenta uspješno, kod unutar metode *then* se izvršava. Njime se poništava vrijednosti obrasca i poruke o pogrešci. Zatim se korisnik preusmjerava na podstranicu koriničkog računa.

Sličnom logikom napisana je funkcija prijave već registriranih korisnika. Razlika je što pri prijavi korisnika nije potrebno arhivirati podatke u bazu, nego ih samo provjeriti. To je učinjeno Firebase funkcijom *signInWithEmailAndPassword.* Funkcija provjerava podudaranje unesene e-pošte i zaporke s onima u bazi podataka. Kod je prikazan slikom 4.11.

```
const Prijava = (e) => {
  e.preventDefault();
  projectAuth.signInWithEmailAndPassword(email, lozinka).then(() => {
    postaviEmail('');
    postaviLozinka('');
    postaviGreskaPrijava('');
    usmjeri('/Profile');
  }).catch(err => postaviGreskaPrijava(err.message));
}
```

Sl. 4.11. *Prikaz funkcije za prijavu korisnika*

Implementiranje funkcija u JSX obrazac poprilično je jednostavno. Pritiskom gumba za slanje podataka obrasca poziva se svojstvo *onSubmit* koje nadalje poziva željenu funkciju. Kako bi korisniku bilo jasno da sva polja moraju biti unesena, svako polje za unos ima *required* atribut. Rukovatelj događajima *onChange* poziva funkciju za postavljanje vrijednosti varijable unesenom vrijednosti. Na dnu obrasca nalazi se red koda koji uvjetno prikazuje *div* element ako je varijabla *greskaRegistracija* (poruka o pogrešci povezana s registracijom) istinita. To omogućuje prikaz poruke o pogrešci ako dođe do problema tijekom postupka registracije. Primjer opisanog koda nalazi se na slici 4.12.

```
<form className='form-wrapper mx-auto w-50' onSubmit={Registracija}>
  <input type='Email' placeholder='E-mail' className='reg-input row reg-animation' required
    onChange={(e) => postaviEmail(e.target.value)}></input>
  <input type='text' placeholder='Korisničko ime' className='reg-input row reg-animation' required
    onChange={(e) => postaviIme(e.target.value)}></input>
  <input type='password' placeholder='Lozinka' className='reg-input row reg-animation' required
    onChange={(e) => postaviLozinka(e.target.value)}></input>
  <button type="submit" class="reg-button reg-animation row" >REGISTRIRAJ SE</button>
  {greskaRegistracija && <div className='error-register'>{greskaRegistracija}</div>}
</form>
```

Sl. 4.12. *Prikaz JSX obrasca za registraciju*

4.4. Izrada podstranice za upis na plesne programe

Kako bi se prijavljeni korisnici mogli upisati na određene plesne programe, potrebno je izraditi podstranicu s navedenim mogućnostima. Na slici 4.13. nalazi se kod za provjeru stanja korisnika.

```
const [korisnik, postaviKorisnika] = useState(null);
useEffect(() => {
    const provjeriKorisnika = projectAuth.onAuthStateChanged((korisnik) => {
        postaviKorisnika(korisnik);
    });

    return () => provjeriKorisnika();
}, []);
```

Sl. 4.13. *Prikaz koda za provjeru stanja korisnika*

Podaci plesnih programa spremljeni su u Firestore bazu podataka, stoga ih prvo treba prikazati na stranici, a to je postignuto *useEffect* metodom. Unutar *useEffect* metode nalazi se asinkrona funkcija *prikupiPodatke*. Asinkrone funkcije obavljaju asinkrone radnje kao što su dohvaćanje podataka i postavljanje mrežnih zahtjeva. Asinkrone funkcije ne blokiraju izvođenje ostatka koda odnosno dopuštaju programu izvršavanje drugih radnji. Prikupljanje podataka iz kolekcije Firestore baze podataka učinjeno je sljedećim kodom: *const upit = await getDocs(collection(projectFirestore, 'Dances'));*. Funkcija *getDocs* pohranjuje sve dokuemnte unutar *Dances* kolekcije, a ključna riječ *await* pauzira izvršavanje funkcije dok se pristup podacima ne dobije ili odbije. Petljom *forEach* su, u polje *plesniPodaci*, spremljeni svi podaci, svih dokumenata. Podaci koji se nalaze u Firestore bazi podataka su abecedno poslagani, što znači da dani u tjednu neće biti kronološki prikazani. Kako bi se to izbjeglo, poslagani su na sljedeći način. *Object.entries()* je JavaScript metoda koja pretvara objekt u niz njegovih parova. Za svaki ples koristi se *Object.entries()* za pretvaranje objekta *Termin* u niz parova „ključ-vrijednost". Funkcija sortiranja poziva se na rezultirajućem nizu parova „ključ-vrijednost". Uspoređuje dane u tjednu (*prvi[0] i drugi[0]*) i određuje njihov redoslijed na temelju niza *daniUTjednu*. Razvrstani niz parova zatim se pretvara natrag u objekt pomoću *Object.fromEntries* funkcije. Izvorni plesni objekt se širi (*{ ...ples}*) kako bi se stvorio novi objekt sa svim izvornim svojstvima, osim što je polje *Termin* zamijenjeno sortiranim. Opisan kod nalazi se na slici 4.14.

```
const [ples, postaviPles] = useState([]);
const usmjeri = useNavigate();
useEffect(() => {
    const prikupiPodatke = async () => {
        const dokumenti = await getDocs(collection(projectFirestore, 'Dances'));
        const plesoviPodaci = [];
        dokumenti.forEach((dokument) => {
            plesoviPodaci.push({ id: dokument.id, ...dokument.data() });
        });

        const sortiraniPles = plesoviPodaci.map((ples) => {
            const sortiraniTermini = Object.entries(ples.Termin).sort((prvi, drugi) => {
                const daniUTjednu = ['Pon', 'Uto', 'Sri', 'Čet', 'Pet', 'Sub'];
                const danPrvi = prvi[0];
                const danDrugi = drugi[0];
                return daniUTjednu.indexOf(danPrvi) - daniUTjednu.indexOf(danDrugi);
            });

            return { ...ples, Termin: Object.fromEntries(sortiraniTermini) };
        });
        postaviPles(sortiraniPles);
    };

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

Sl. 4.14. *Prikaz koda za prikupljanje podataka iz Firestore baze podataka*

Svaki plesni stil ima odgovarajući gumb za upis. Kôd funkcije pokrenute pritiskom gumba nalazi se na slici 4.15. Pozvana funkcija je također asinkrona, a vrijednost koju prima je identifikacijski broj odabranog plesa. Uvjetom *if* provjereno je korisničko stanje. Varijablom *korisnikReferenca* pohranjuje se prijavljen korisnik, dok se varijablom *plesReferenca* sprema putanja do odabranog plesnog stila. Izjavom *try catch* hvataju se moguće pogreške pri pohrani putanje u bazu podataka te su prikazane u konzoli aplikacije. Pohrana je izvedena *update* funkcijom. Kako bi se plesač mogao upisati u više programa, podaci se spremaju u polje pod nazivom *dances*.

```
const naStisakGumba = async (plesID) => {

    if (!korisnik) {
        alert('Morate se registrirati kako bi se upisali na određeni ples.');
        return;
    }

    const korisnikReferenca = projectFirestore.collection('Users').doc(korisnik.uid);
    const plesReferenca = projectFirestore.doc(`Dances/${plesID}`);

    try {
        await korisnikReferenca.update({
            dances: arrayUnion(plesReferenca),
        });
        usmjeri('/Profile');
    } catch (error) {
        console.log('Error adding dance:', error);
    }
};
```

Sl. 4.15. *Prikaz funkcije za upis korisnika na željeni plesni program*

Funkcijom *map()*, moguće je unutar JSX-a izlistati sve vrijednosti polja te ih na taj način prikazati na zaslonu. JSX kod vidljiv je na slici 4.16.

```
{ples.map((ples) => (
    <div className="col-md-6 col-lg-6 col-xl-4" key={ples.id}>
        <section>
            <div className='container py-5'>
                <div className='card' data-aos="fade-right" data-aos-offset="200" data-aos-once="true">
                    <img src={ples.Slika} className='card-img-top' alt='Plesni stil' />
                    <div className='card-body'>
                        <div className='text-center'>
                            <h5 className='card-title'>{ples.id}</h5>
                            <p className='mb-4'>{ples.Trener}</p>
                        </div>
                        <div>
                            <h5>Termin</h5>
                            {ples.Termin && Object.entries(ples.Termin).map(([nazivVarijable, vrijednost]) => (
                                <div className='d-flex justify-content-between' key={nazivVarijable}>
                                    <span>{nazivVarijable}</span>
                                    <span>{vrijednost}</span>
                                </div>
                            ))}
                        </div>
                        <div className='d-flex justify-content-between total font-weight-bold mt-4'>
                            <span>Cijena</span>
                            <span>{ples.Cijena}</span>
                        </div>
                    </div>
                    <button className='mb-0 w-100 rounded' onClick={() => naStisakGumba(ples.id)}>Upiši</button>
                </div>
            </div>
        </section>
    </div>
))}
```

Sl. 4.16. *Dio koda za prikaz svih plesnih programa*

4.5. App.js

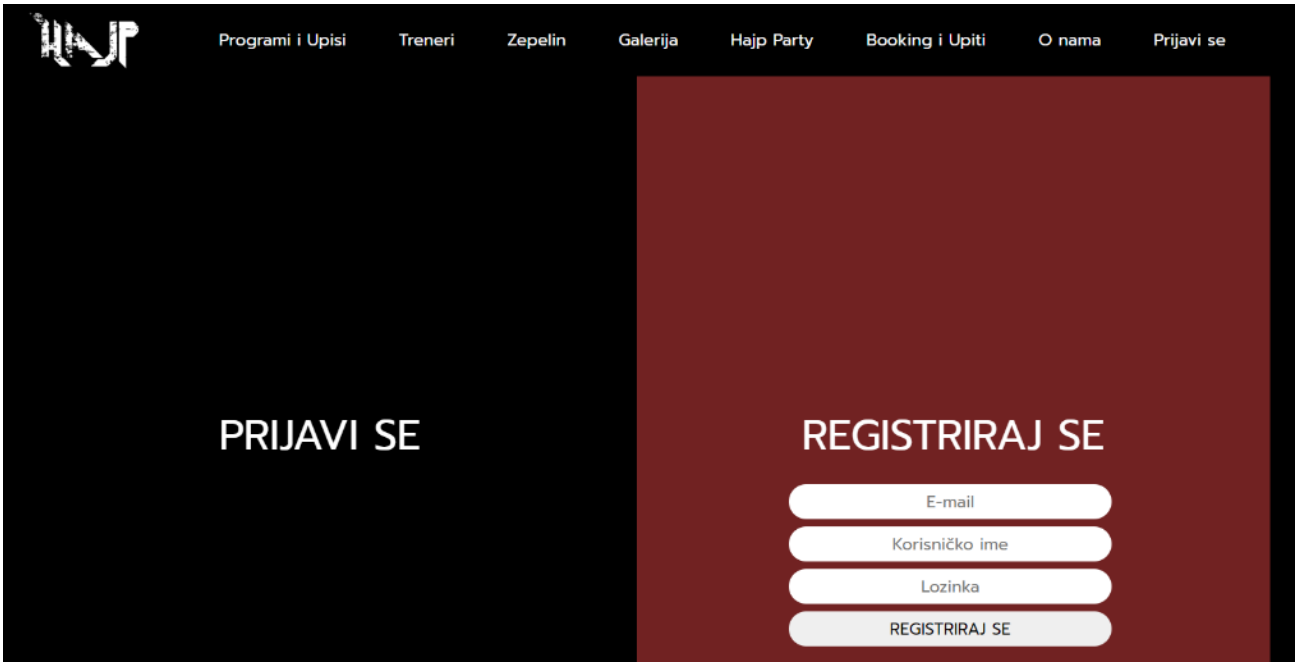
Kako bi stranice i navigacijska traka bili ispravno prikazani, potrebno ih je prikazati u App.js datoteci. Kod datoteke nalazi se na slici 4.17.

```
export class App extends Component {
  render() {
    return (
      <div className="App">
        <Navigation />
        <Routes>
          <Route path="/" element={<Pocetna />} />
          <Route path="/Programi" element={<Programi  />} />
          <Route path="/Booking" element={<Booking />} />
          <Route path="/Treneri" element={<Treneri />} />
          <Route path="/Zepelin" element={<Zepelin />} />
          <Route path="/Galerija" element={<Galerija />} />
          <Route path="/Party" element={<Party />} />
          <Route path="/About" element={<About />} />
          <Route path="/Login" element={<Login />} />
          <Route path="/Profile" element={<Profile  />} />
        </Routes>
        <Footer />
      </div>
    );
  }
}

export default App;
```

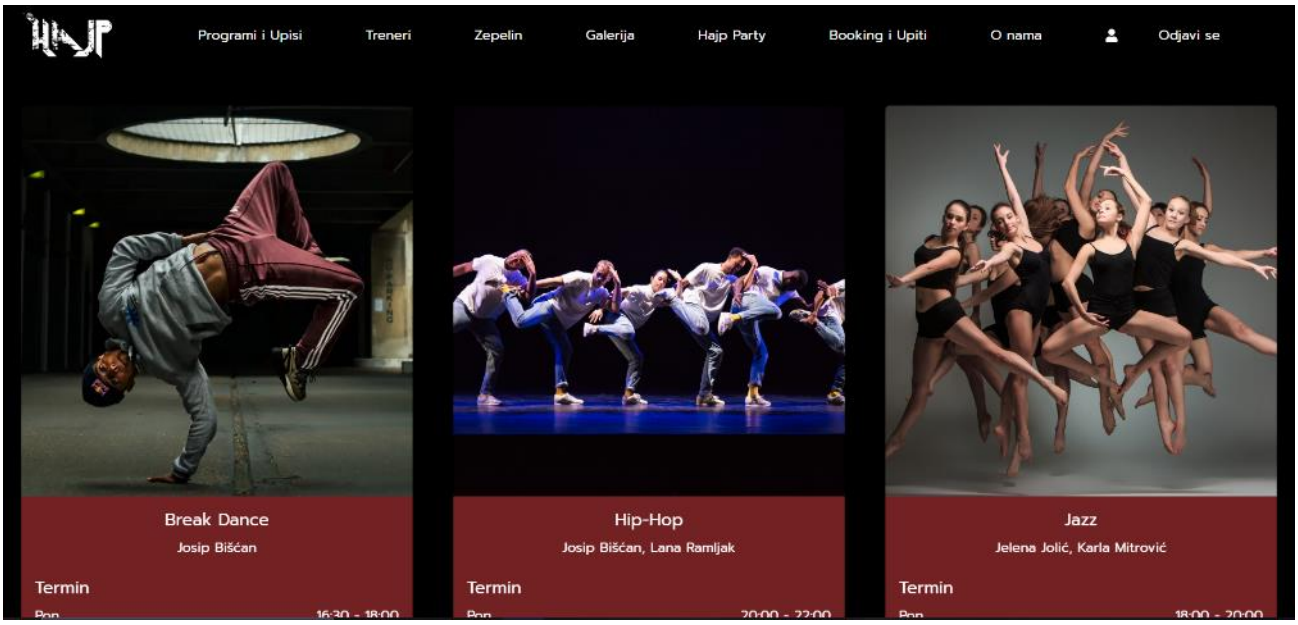
Sl. 4.17. *Dio koda App.js datoteke*

Metoda *render()* odgovorna je za prikaz JSX komponenti. Komponenta <*Routes*> koristi se za definiranje konfiguracije usmjeravanja. Sadrži više komponenti tj. ruta (<*Route*>), od kojih svaka preslikava određeni put do odgovarajuće komponente koja se prikazuje. Atribut *path* definira putanju URL-a, a atribut *element* specificira komponentu koja se prikazuje kada se staza podudara s trenutnim URL-om.

Slikom 4.18. prikazana je stranica za prijavu i registraciju, a slikom 4.19. stranica sa svim plesnim programima. Na slici 4.18. korisnik je ulogiran dok na slici 4.19 nije, što se da primijetiti na navigacijskoj traci.



Sl. 4.18. *Prikaz stranice za registraciju i prijavu korisnika*



Sl. 4.19. *Prikaz stranice s plesnim programima.*

4.6. Podstranica korisničkog profila

Na podstranici korisničkog profila prikazani su upisani plesni programi, ukoliko ih ima. Ispod svakog plesnog programa stoji gumb s kojim se korisnik može ispisati. Ukoliko korisnik nije upisan ni na jedan program ili nije prijavljen, biti će ispisana odgovarajuća poruka.

Prije svega, potrebno je provjeriti stanje korisnika. Kod za provjeru prikazan je slikom 4.20.

```
useEffect(() => {
  const provjeriKorisnika = projectAuth.onAuthStateChanged((korisnik) => {
    if (korisnik) {
      postaviKorisnika(korisnik);
      dohvatiKorisnickePodatke();
      setRemovalStatus(false);
    } else {
      postaviKorisnika(null);
    }
  });

  return () => provjeriKorisnika();
}, [korisnik, removalStatus]);
```

Sl. 4.20. *Prikaz koda za provjeru korisničkog stanja*

Metoda *useEffect* će se pokrenuti pri promijeni stanja korisnika ili ispisom iz plesnog programa. Funkcija *setRemovalStatus* poziva se s *false* kako bi se označilo da se status uklanjanja (ako postoji) treba poništiti. Ako korisnik ne postoji (tj. korisnik nije autentificiran), funkcija *postaviKorisnika* poziva se s *null* da bi se korisnik izbrisao iz stanja.

Nakon provjere stanja korisnika, potrebno je dohvatiti njegove podatke. Dohvaćanje podataka korisnika koristi kod sličan dohvaćanju podataka o plesnim programima i prikazan je slikom 4.21.

```
const dohvatiKorisnickePodatke = async () => {
  try {
    const korisnikReferenca = await projectFirestore.collection('Users').doc(korisnik.uid).get();
    const korisnikPodaci = korisnikReferenca.data();
    postaviIme(korisnikPodaci.Name);

    if (korisnikPodaci.dances && korisnikPodaci.dances.length > 0) {
      const plesReferenca = korisnikPodaci.dances;
      const plesPodaci = await Promise.all(
        plesReferenca.map(async (plesRef) => {
          const snapshot = await plesRef.get();
          return snapshot.exists ? { id: snapshot.id, ...snapshot.data() } : null;
        })
      );

      const provjereniPlesniPodaci = plesPodaci.filter((podatak) => podatak !== null);
      const sortiraniPlesniPodaci = provjereniPlesniPodaci.map((ples) => {
        const sortiraniTermini = Object.entries(ples.Termin).sort((a, b) => {
          const daniUTjednu = ['Pon', 'Uto', 'Sri', 'Čet', 'Pet', 'Sub'];
          const danPrvi = a[0]
          const danDrugi = b[0]
          return daniUTjednu.indexOf(danPrvi) - daniUTjednu.indexOf(danDrugi);
        });

        return { ...ples, Termin: Object.fromEntries(sortiraniTermini) };
      });

      postaviPlesove(sortiraniPlesniPodaci);
    } else {
      postaviPlesove([]);
    }
  } catch (error) {
    console.log('Error retrieving user data:', error);
  }
};
```

Sl. 4.21. *Prikaz funkcije za dohvaćanje korisničkih podataka*

U konstantnoj varijabli *korisnikReferenca* nalazi se referenca prijavljenog korisnika. Kako bi se locirao dokument prijavljenog korisnika, koristi se nastavak *uid*. Podaci iz reference spremljeni su u varijablu *korisnikPodaci*. Zatim, *if* izjava provjerava ima li korisnik upisanih plesnih programa. Ako korisnik ima referencu na plesne programe, nastavlja se s dohvaćanjem podataka o plesu koristeći plesne reference pohranjene u *korisnikPodaci.dances*. Funkcijom *Promise.all* istovremeno se izvršavaju višestruke asinkrone operacije, preslikavajući svaku referencu plesa za dohvaćanje odgovarajućih podataka o plesu. Dohvaćeni plesni podaci zatim se filtriraju kako bi se uklonile sve *null* vrijednosti.

Funkcionalnost ispisa s plesnog stila prikazana je slikom 4.22.

```
const ispisiSe = async (plesID) => {
  const korisnikReferenca = projectFirestore.collection('Users').doc(korisnik.uid);
  const korisnikSnapshot = await korisnikReferenca.get();
  const korisnikPodaci = korisnikSnapshot.data();
  const azuriraniPlesovi = korisnikPodaci.dances.filter(
    (plesRef) => plesRef.id !== plesID
  );
  await korisnikReferenca.update({ dances: azuriraniPlesovi });
  setRemovalStatus(true);
};
```

Sl. 4.22. *Prikaz koda za ispis s plesnog stila*

Definirana je asinkrona funkcija *ispisiSe* koja uzima *plesID* kao parametar. Najprije je definirana referenca na korisnički dokument u kolekciji. Funkcija zatim koristi *await* za dohvaćanje snimke (*snapshot*) korisničkog dokumenta iz Firestore-a. Metoda *korisnikSnapshot.data()* poziva se za izdvajanje korisničkih podataka iz snimke. Niz *korisnikPodaci.dances* filtrira se pomoću metode *filter()* za uklanjanje reference plesa s navedenim identifikacijskim ključem. Linija koda *korisnikReferenca.update({ dances: azuriraniPlesovi })* ažurira polje za plesove u korisničkom dokumentu s poljem *azuriraniPlesovi*, uklanjajući navedeni ples s korisničkog popisa plesova. Konačno, poziva se funkcija *setRemovalStatus(true)* da se varijabla stanja *RemoveStatus* postavi na true, što pokazuje da je ples uspješno uklonjen.

U JSX kodu, izjavom *{dances.length > 0 ? (* provjerava se postojanje upisanih plesova te ih se prikazuje istim načinom kao na podstranici za upis. Ukoliko korisnik nije upisan ni na jedan ples, pokreće se kod prikazan slikom 4.23.

```
) : (
  <div className='no-dances-message'>
    <p className='text-center'>
      Niste upisani ni na jedan plesni stil, upišite se{' '}
      <a href="/Programi">ovdje</a>.
    </p>
  </div>
)}
```

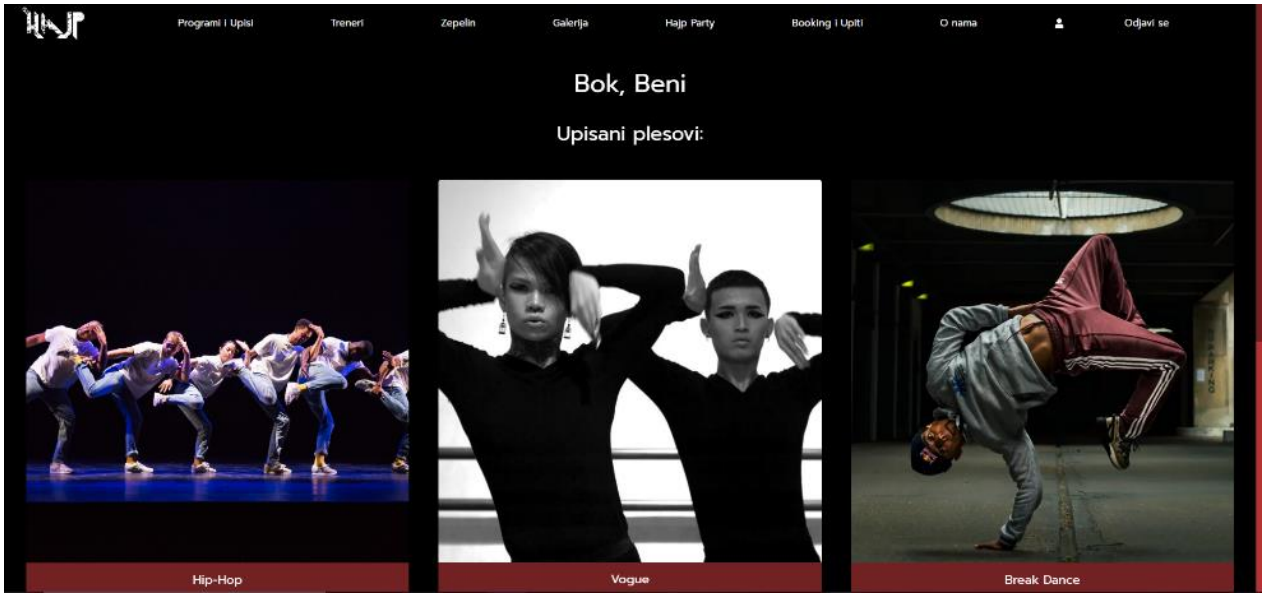
Sl. 4.23. *Prikaz JSX koda u slučaju da korisnik nije upisan ni na jedan plesni program*

Preostaje samo generirati JSX kod ukoliko korisnik nije prijavljen. Kod je prikazan slikom 4.24, a slikama 4.25., 4.26. i 4.27. izgled stranice korisničkog profila.

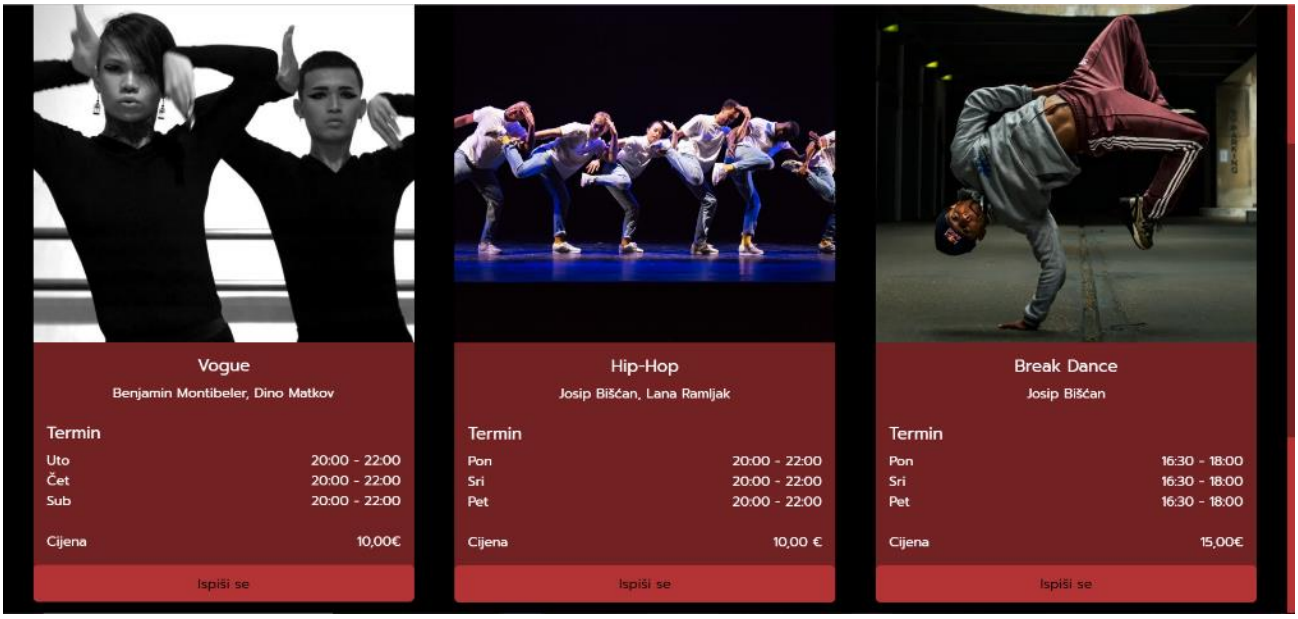
```jsx
if (!korisnik) {
  return (
    <div className='profile-container not-logged container-fluid d-flex justify-content-center'>
      <div className='row not-logged-container justify-content-center'>
        <h1 className='text-center'>Prijavite se kako bi vidjeli vaš profil.</h1>
        <button onClick={usmjeravanjeKorisnika}>Log in</button>
      </div>
    </div>
  );
}
```

Sl. 4.24. *Prikaz JSX koda u slučaju da korisnik nije prijavljen*

Sl. 4.25. *Prikaz stranice korisničkog profila bez upisanih plesnih programa*



Sl. 4.26. *Prikaz stranice korisničkog profila s upisanim plesnim programima*



Sl. 4.27. *Prikaz mogućnosti ispisa s plesnog programa*

4.7. Galerija

Za razliku od prijašnje objašnjenih podstranica, galerija ne učitava podatke iz baze podatak, nego iz Firebase spremišta. Slikom 4.28. prikazan je kod za pohranu putanja slika iz spremišta.

```
const [slike, postaviSlike] = useState([]);
useEffect(() => {
    const storageReferenca = projectStorage.ref();

    storageReferenca.listAll().then((storageRef) => {
        const datoteka = storageRef.prefixes.map((datotekaRef) => {
            const imeDatoteke = datotekaRef.name;

            return datotekaRef.listAll().then((slika) => {
                const slikaPromise = slika.items.map((slikaRef) =>
                    slikaRef.getDownloadURL()
                );

                return Promise.all(slikaPromise).then((slikaURL) => ({
                    imeDatoteke,
                    slikaURL,
                }));
            });
        });

        Promise.all(datoteka).then((slikaPodaci) => {
            postaviSlike(slikaPodaci);
        });
    });
}, []);
```

Sl. 4.28. *Prikaz koda za pohranu URL vrijednosti slika iz Firebase spremišta*

Unutar *useEffect* metode deklrarirana je varijabla *storageReferenca* iz *projectStorage.ref()* te ona predstavlja referencu na mjesto pohrane. Metoda *listAll()* poziva se na *storageReferenca* za dohvaćanje popisa svih stavki (datoteka i direktorija) te vraća obećanje koje se rješava objektom *storageRef*. Za svaki direktorij (*datotekaRef*) u *storageRef*-u, izvodi se operacija mapiranja. Naziv direktorija ekstrahira se u varijablu *imeDatoteke*. Metoda *listAll()* poziva se na svaku referencu direktorija (*datotekaRef*) za dohvaćanje popisa svih stavki unutar tog direktorija. Stvoren je lanac obećanja za pohranu URL-ova svake datoteke (*slikaRef*). Metoda *getDownloadURL*() služi za dohvaćanje URL-a. Rezultati se prikupljaju pomoću *Promise.all()* da bi se dobio niz URL-ova. Naposlijetku se vraća objekt koji sadrži *imeDatoteke* (naziv direktorija) i *slikaURL* (niz URL-ova za preuzimanje fotografija). Rezultirajući niz objekata (*slikaPodaci*) prosljeđuje se *postaviSlike* funkciji što ažurira varijablu *slike* s dohvaćenim podacima.

Svaka učitana slika ima svoj modular tj. moguće ju je uvećati pritiskom pokazivača. Kod za modulare nalazi se na slici 4.29.

```
const [odabranaSlika, postaviOdabranuSliku] = useState(null);
const otvorenModal = (slikaUrl) => {
    postaviOdabranuSliku(slikaUrl);
  };

  const zatvorenModal = () => {
    postaviOdabranuSliku(null);
  };

  const upravljajModalKlikom = (e) => {
    if (e.target.classList.contains('modal-overlay')) {
      zatvorenModal();
    }
  };

  const upravljajSlikaKlikom = (e) => {
    e.stopPropagation();
  };
```

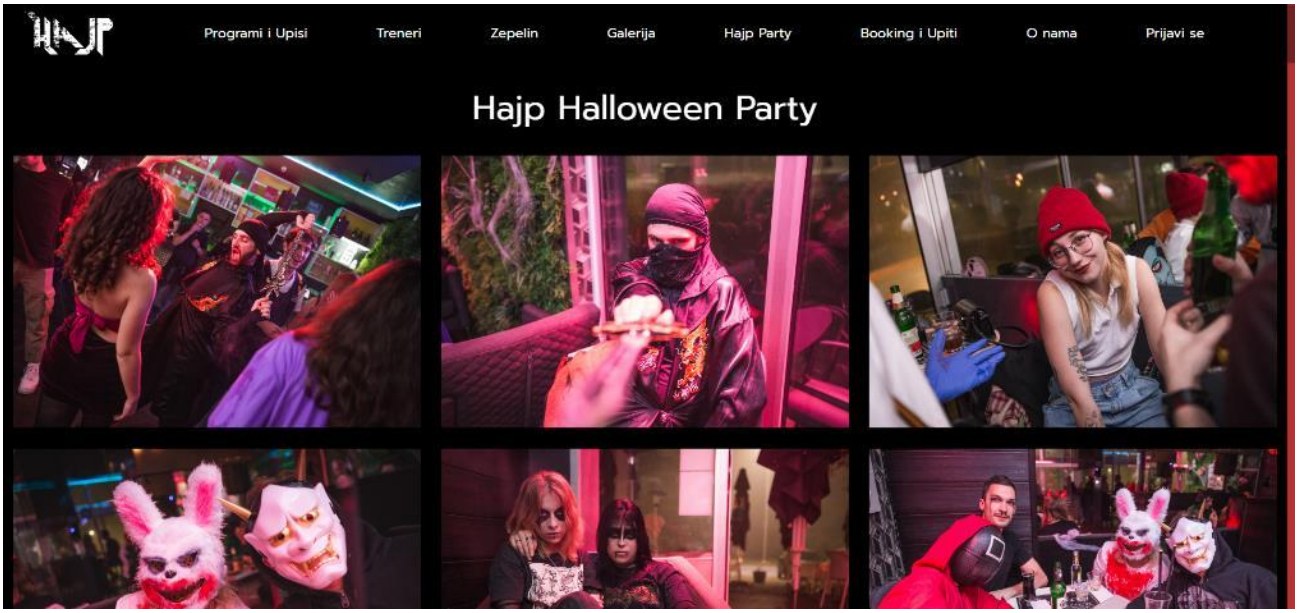
Sl. 4.29. *Prikaz koda za modular slike*

JSX kod za prikaz učitanih slika i otvaranje modulara nalazi se na slici 4.30.

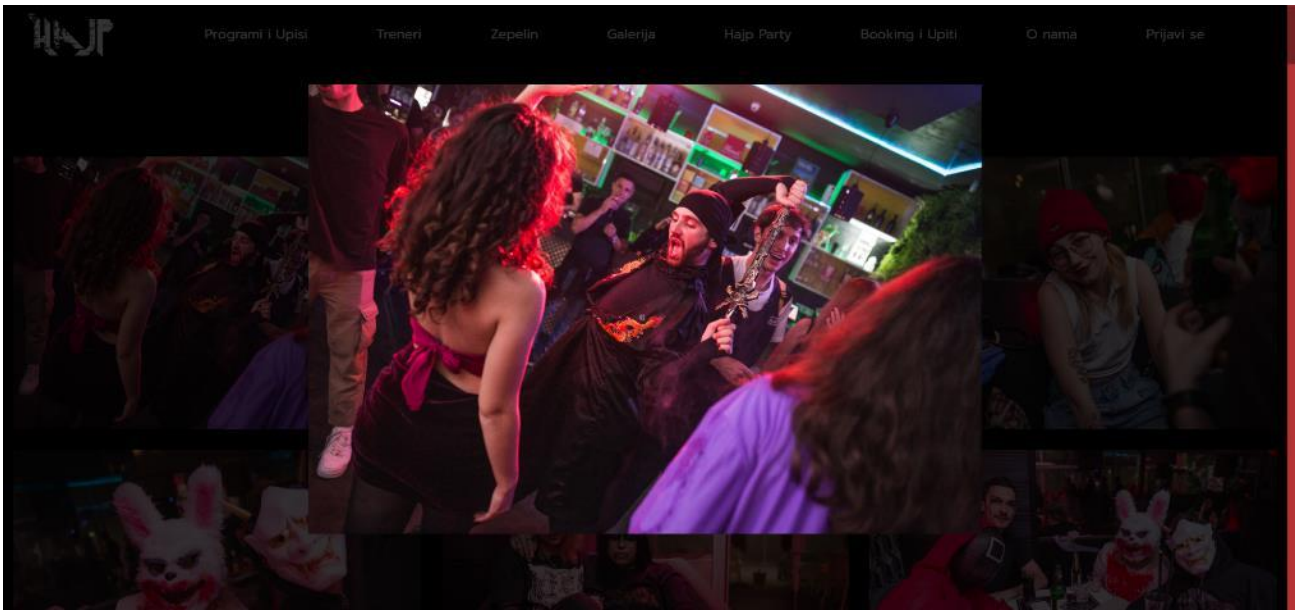
```
<div className="container-fluid gallery-container">
  {slike.map((slika) => (
    <div className="row justify-content-around" key={slika.imeDatoteke}>
      <h1 className="album-title text-center" data-aos="fade-right" data-aos-offset="200" data-aos-once="true">
        {slika.imeDatoteke}
      </h1>
      {slika.slikaURL.map((url, indeks) => (
        <div
          className="col-4 img-column"
          key={url}
          data-aos="fade-right"
          data-aos-delay={indeks * 100}
          data-aos-offset="200" data-aos-once="true"
          onClick={() => otvorenModal(url)}
        >
          <img className="img-fluid" src={url} alt="Firebase Storage" />
        </div>
      ))}
    </div>
  ))}
  {odabranaSlika && (
    <div className="modal-overlay" onClick={upravljajModalKlikom}>
      <div className="modal-content" data-aos="fade" onClick={zatvorenModal}>
        <img
          className="modal-image"
          src={odabranaSlika}
          alt="Modal"
          onClick={upravljajSlikaKlikom}
        />
      </div>
    </div>
  )}
</div>
```

Sl. 4.30. *Prikaz JSX koda galerije*

Izgled galerije pokazan je slikom 4.31., a slikom 4.32. prikazan je otvoren modular tj. uvećana slika.



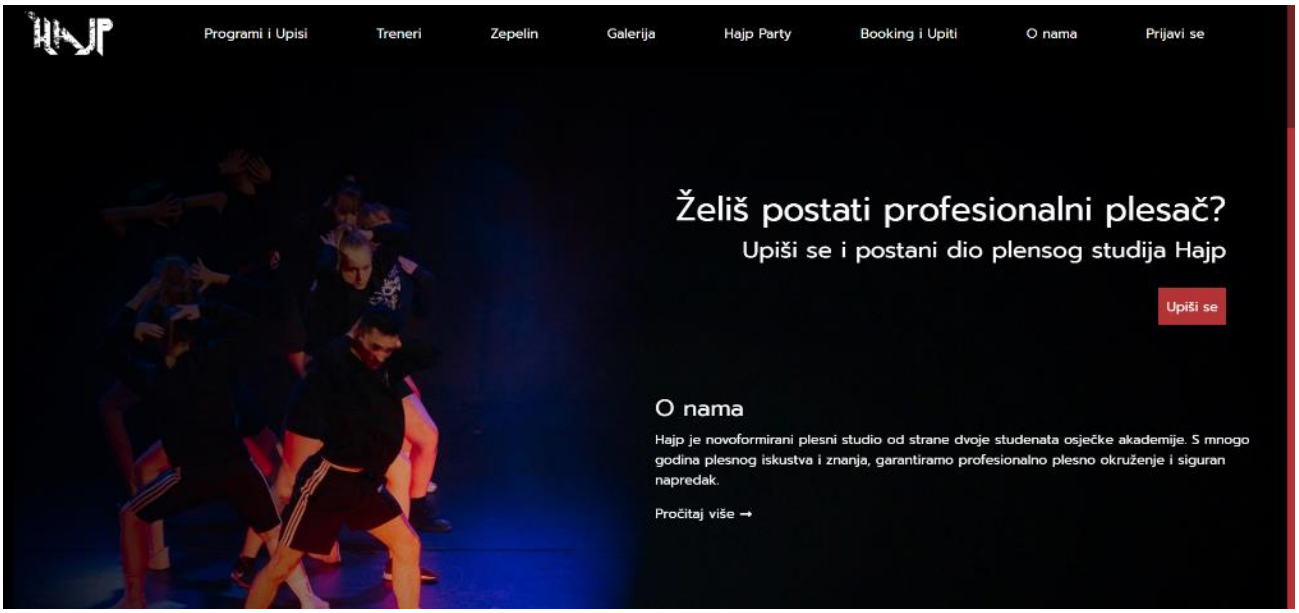
Sl. 4.31. *Prikaz podstranice galerije*



Sl. 4.32. *Prikaz podstranice galerije s uvećanom slikom*

4.8. Naslovna stranica

Naslovna stranica plesnog studija sadrži najbitnije informacije i poveznice kao što je vidljivo na slici 4.33.



Sl. 4.33. *Prikaz naslovne stranice plesnog studija*

Kako bi se zaokupila što veća pažnja korisnika, na elementima vidljivim na slici 4.32. korištene su animacije. U React-u postoje različiti pristupi za dodavanje animacija komponentama, no izabrana metoda za ovaj rad zove se AOS. AOS je kratica za „Animate On Scroll" što znači da će se animacije pokrenuti prilikom listanja stranice. Instalacija AOS biblioteke poprilično je lagana. Najprije je potrebno pokrenuti naredbu *npm install aos –save* u konzoli Visual Studio Code-a, koja dodaje AOS trenutnom projektu. Zatim treba ručno uvesti AOS biblioteke linijama koda *import AOS from 'aos';* i *import 'aos/dist/aos.css';*. AOS se inicijalizira na sljedeći način: slika 4.34.

```
useEffect(() => {
  const initAOS = setTimeout(() => {
    AOS.init({
      duration: 2000
    });
  }, 1000);

  return () => clearTimeout(initAOS);
}, []);
```

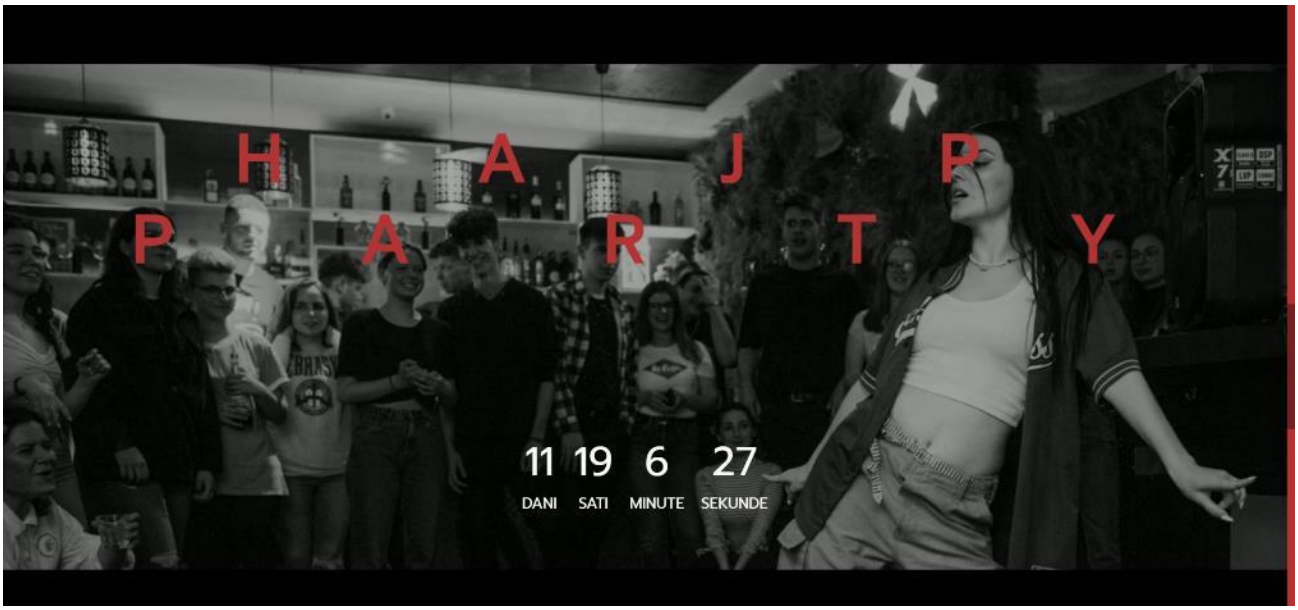
Sl. 4.34. *Prikaz inicijaliziranja AOS animacija*

Potom se animacije, na željene elemente, dodaju u obliku atributa: *data-aos="fade-left" data-aos-offset="200" data-aos-once="true"*. Atributom *data-aos* definirana je vrsta animacije, *data-aos-offset* definira ofset animacije u milisekundama, a atributom *data-aos-once=”true”* se jednom učitana animacija neće ponoviti. Slikom 4.35. prikazan je dio koda naslovne stranice s implementiranim animacijama.

```
return (
  <div className='main-page-container container-fluid'>
    <div className='container-1'>
      <div className='row justify-content-end h-100'>
        <div className='col-6 h-100'>
          <div className='row'>
            <div className='main-title h-100' data-aos="fade-left" data-aos-offset="200" data-aos-once="true">
              <div className='text-end padding-div'>
                <h1 className=''>Želiš postati profesionalni plesač?</h1>
                <h3 className=''>Upiši se i postani dio plensog studija Hajp</h3>
                <button onClick={usmjeravanjeKorisnika}>Upiši se</button>
              </div>
              <br /><br /><br />
              <h3>O nama</h3>
              <p>Hajp je novoformirani plesni studio od strane dvoje studenata osječke akademije. S mnogo godina plesnog iskustva i znanja,
                garantiramo profesionalno plesno okruženje i siguran napredak.
              </p>
              <a onClick={usmjeravanjeKorisnika2}>Pročitaj više <FaLongArrowAltRight /></a>
            </div>
          </div>
        </div>
      </div>
    </div>
    <div className='container-fluid d-flex justify-content-center row main-content-1'>
      <img src={link1} className='image-title-main' data-aos="fade-right" data-aos-offset="200" data-aos-once="true" />
      <img src={link2} className='image-title-main' data-aos="fade-down" data-aos-offset="200" data-aos-once="true" />
      <img src={link3} className='image-title-main' data-aos="fade-left" data-aos-offset="200" data-aos-once="true" />
      <h2 className='text-center w-100 h-100'>
        <a onClick={usmjeravanjeKorisnika3}>Plesni Programi <FaLongArrowAltRight /></a>
      </h2>
```

Sl. 4.35. *Prikaz dijela koda naslovne stranice*

Na naslovnoj stranici nalazi se brojač koji odbrojava do određenog događaja – slika 4.36.



Sl. 4.36. *Prikaz brojača na naslovnoj stranici*

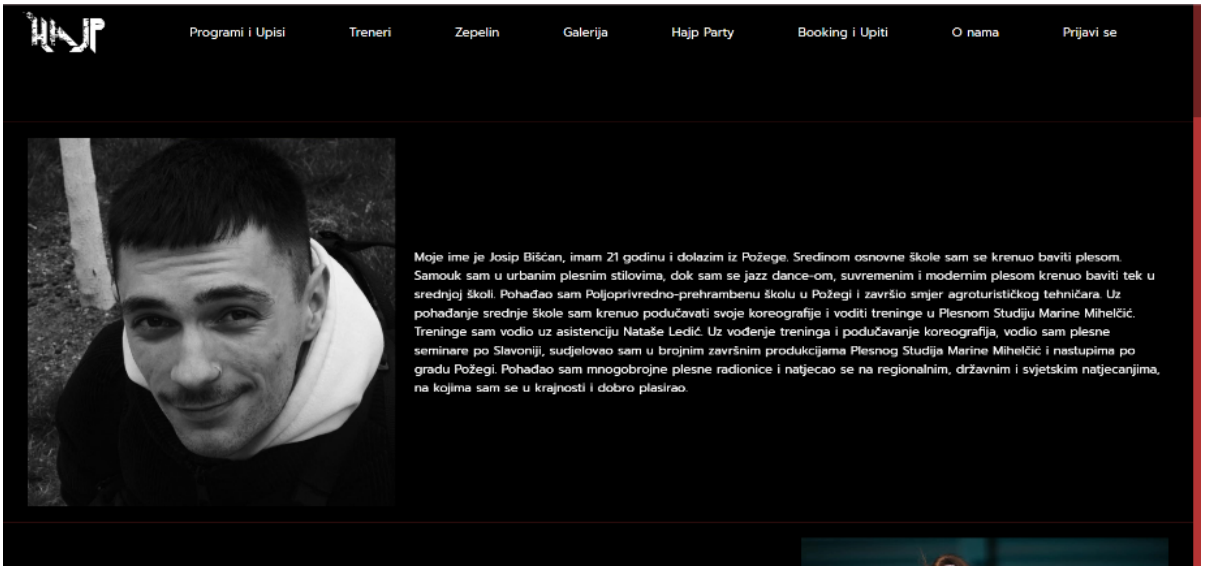
Brojač je kreiran JavaScipt konstruktorom *new Date()*. Pomoću njega moguće je kreirati varijablu s pohranjenim datumom, a metodom *getTime()* moguće je dohvatiti trenutni datum. Razlikom današnjeg i očekivanog datuma, najlakše je kreirati brojač. Kod brojača s naslovne stranice prikazan je slikom 4.37.

```
useEffect(() => {
  const datum = new Date('2023-06-26');
  const interval = setInterval(() => {
    const danasnjiDatum = new Date().getTime();
    const razlika = datum - danasnjiDatum;

    const dani = Math.floor(razlika / (1000 * 60 * 60 * 24));
    const sati = Math.floor((razlika % (1000 * 60 * 60 * 24)) / (1000 * 60 * 60));
    const minute = Math.floor((razlika % (1000 * 60 * 60)) / (1000 * 60));
    const sekunde = Math.floor((razlika % (1000 * 60)) / 1000);

    setCountdown({ dani, sati, minute, sekunde });

    if (razlika < 0) {
      clearInterval(interval);
      setCountdown({ days: 0, hours: 0, minutes: 0, seconds: 0 });
    }
  }, 1000);

  return () => clearInterval(interval);
}, []);
```

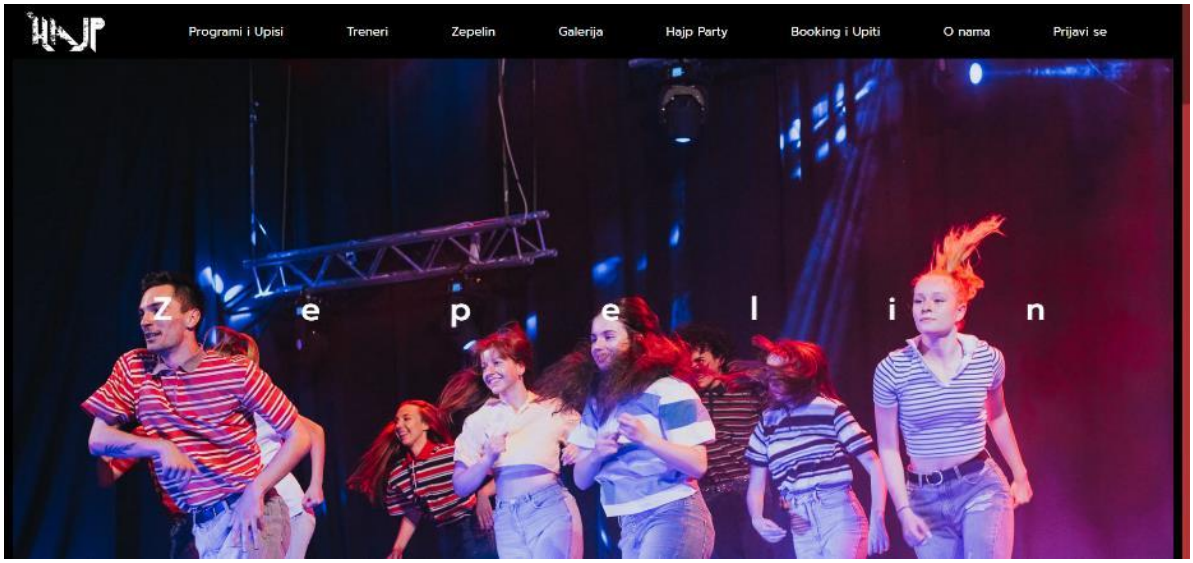
Sl. 4.37. *Prikaz koda brojača*

4.9. Ostale podstranice

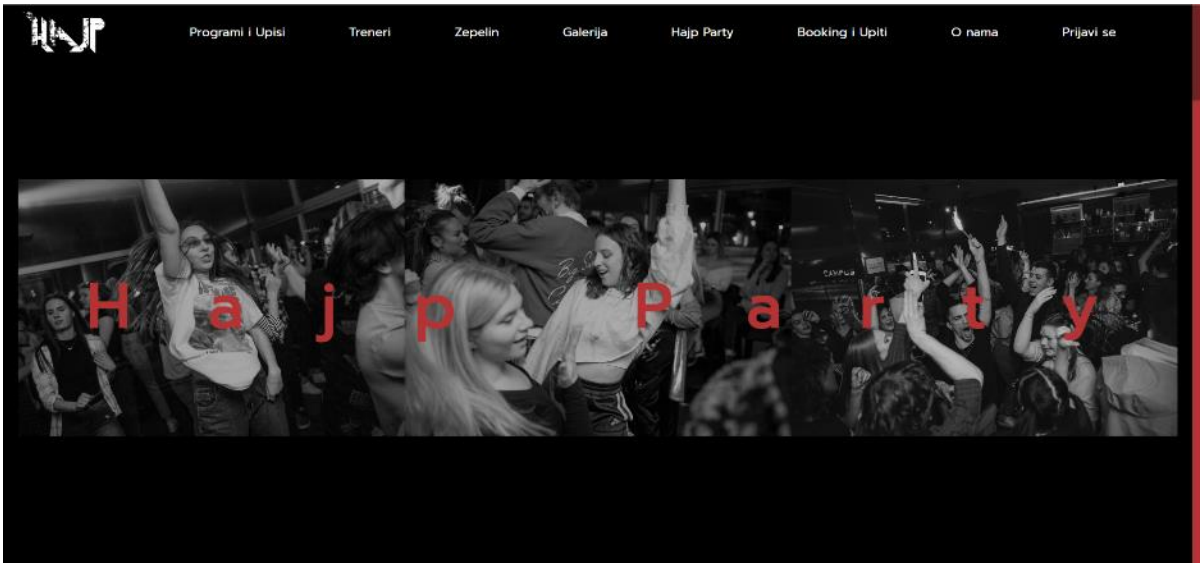
Ostale podstranice aplikacije ne zahtjevaju korištenje kompleksnih *back-end* elemenata jer služe kao informativne stranice. Slikama 4.38., 4.39., 4.40., 4.41., i 4.42. prikazane su podstranice „Treneri”, „Zepelin”, „Hajp Party”, „Booking i Upiti” i „O nama”.



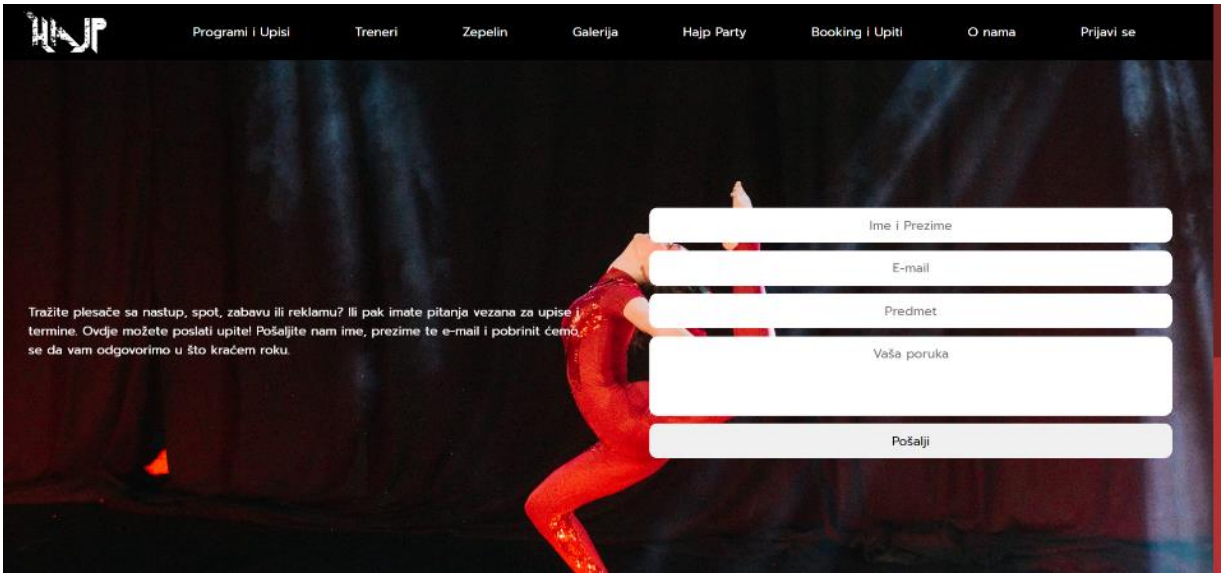
Sl. 4.38 *Prikaz podstranice ,,Treneri"*



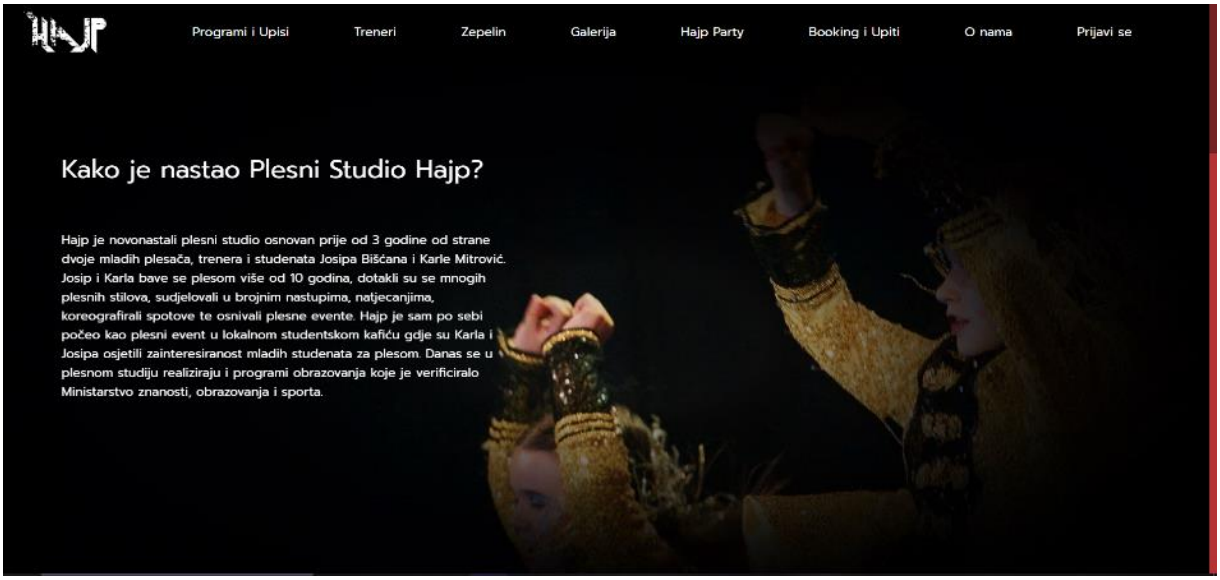
Sl. 4.39. *Prikaz podstranice ,,Zepelin"*



Sl. 4.40. *Prikaz podstranice ,,Hajp Party"*



Sl. 4.41. *Prikaz podstranice ,,Booking i Upiti"*



Sl. 4.42. *Prikaz podstranice „O nama”*

Sve navedene podstranice izrađene su istim ili sličnim načinom, stoga nije potrebno prikazati kôd svake. Na slici 4.43. nalazi se dio koda podstranice „O nama”.

```
return (
  <div className='about-container container-fluid'>

    <div className='info-container container-fluid'>
      <div className='info-text' data-aos="fade-right" data-aos-offset="200" data-aos-once="true">
        <h2>Kako je nastao Plesni Studio Hajp?</h2>
        <p>Hajp je novonastali plesni studio osnovan prije od 3 godine od strane dvoje mladih plesača, trenera i studenata Josipa Bišćana
          i Karle Mitrović. Josip i Karla bave se plesom više od 10 godina, dotakli su se mnogih plesnih stilova, sudjelovali u brojnim
          nastupima, natjecanjima, koreografirali spotove te osnivali plesne evente. Hajp je sam po sebi počeo kao plesni event u
          lokalnom studentskom kafiću gdje su Karla i Josipa osjetili zainteresiranost mladih studenata za plesom. Danas se u plesnom studiju
          realiziraju i programi obrazovanja koje je verificiralo Ministarstvo znanosti, obrazovanja i sporta.</p>
      </div>
    </div>

    <div className='info-container2 d-flex justify-content-center'>
      <div className='info-text2' data-aos="fade-down" data-aos-offset="200" data-aos-once="true">
        <h2>Plesni Stilovi</h2>
        <p>Iako se hajp pretežito bavi urbanim plesnim stilovima poput Hip-hopa, Voguea, MTV-a i Break Dancea,
          imamo dvije grupe koje se bave isključivo suvremenim tj. Jazz danceom i Latino plesovima.</p>
      </div>
    </div>

    <div className='info-container3 d-flex justify-content-between'>
      <div className='row w-100 info-padding'>
        <h2 data-aos="fade-right" data-aos-offset="200" data-aos-once="true">Dvorana</h2>
        <div className='row w-100'>
          <div className='col-6' data-aos="fade-right" data-aos-offset="200" data-aos-once="true">
            <img src={require('../images/lokacija.JPG')} className='img-fluid'></img>
          </div>
          <div className='col-6 d-flex my-auto location-info h-100' data-aos="fade-left" data-aos-offset="200" data-aos-once="true">
            <div className='row h-100 justify-content-evenly'>
              <div>
                <p>Nalazimo se u ul. Ivana Gundulića 5, Osijek gdje raspolažemo s jednom velikom te 3 manje dvorane; 2 urbane i baletnom</p>
              </div>
              <div>
                <h3>Kontakt</h3>
```

Sl. 4.43. *Dio koda podstranice „O nama”*

4.10. RAD APLIKACIJE

Web aplikacija plesnog studija Hajp je interaktivno web sjedište za sa svim novostima i informacijama o studiju te mogućnosti online upisa. Neregistriran korisnik odnosno gost može pregledati galeriju, plesne programe, naslovnu stranicu i ostale informativne podstranice, no nema mogućnost online upisa. Registriani korisnik spremljen je u bazu podataka aplikacije te, osim pristupa informacijama, ima sposobnost online upisu na određeni plesni program. Također, samo prijavljeni korisnici mogu vidjeti podstranicu korisničkog profila, na kojoj se nalaze upisani programi s mogućnosti ispisa. Prijavljenom korisniku, na navigacijskoj traci nalazi se funkcija odjave. Nakon odjave, korisnik je preusmjeren na podstranicu za prijavu.

5. ZAKLJUČAK

Zadatak ovog završnog rada je izraditi web sjedište plesnog studija. U uvodnom dijelu rada, predstavljena je problematika zadatka te je opisano pet sličnih rješenja. Aplikacija je dizajnirana pomoću modernih tehnologija web programiranja, koje su detaljno opisane u trećem poglavlju. Baze podataka stvorene su pomoću Firebase sustava Firestore, Authentication i Storage, dok su pohranjeni podaci prikazani koristeći se React *framework*-om i JavaScript metodama. Elementi unutar React JSX koda modelirani su CSS-om i Bootstrap klasama. Postupak izrade koda, dijelovi koda podstranica i njihov krajnji izgled prikazani su u četvrtom poglavlju rada, kao i samo korištenje i funkcionalnosti gotovog proizvoda. Krajnji produkt je interaktivna aplikacija na kojoj se korisnik ima mogućnost registrirati ili nastaviti pretraživanje kao gost. Web lokacija plesnog studija sastoji se od naslovne stranice, podstranice za plesne programe, podstranicu galerije, stranice za registraciju te stranicu korisničkog profila kao i ostalim informativnim stranicama.

LITERATURA

[1] TransForm, dostupno na: https://transform.hr/ (Pristupljeno 3.6.2023.)

[2] STEEZY, dostupno na: https://www.steezy.co/ (Pristupljeno 3.6.2023.)

[3] Leon, dostupno na: https://plesni.studio/ (Pristupljeno 3.6.2023.)

[4] Bailando, dostupno na: https://bailando.hr/ (Pristupljeno 3.6.2023.)

[5] Instinct, dostupno na: https://plesnistudioinstinct.com/ (Pristupljeno 3.6.2023.)

[6] InfoWorld, dostupno na: https://www.infoworld.com/article/3666488/what-is-visual-studio-code-microsofts-extensible-code-editor.html (Pristupljeno 5.6.2023.)

[7] MDN, dostupno na: https://developer.mozilla.org/en-US/docs/Web/HTML (Pristupljeno 5.6.2023.)

[8] MDN, dostupno na: https://developer.mozilla.org/en-US/docs/Web/CSS (Pristupljeno 5.6.2023.)

[9] Mojwebdizajn, dostupno na: https://www.mojwebdizajn.net/skriptni-jezici/vodic/css/css-syntax.php (Pristupljeno 5.6.2023.)

[10] Mojwebdizajn, dostupno na: https://www.mojwebdizajn.net/web-programiranje/vodic/javascript/uvod-u-javascript.php (Pristupljeno 6.6.2023.)

[11] W3Schools, dostupno na: https://www.w3schools.com/js/js_loop_while.asp (Pristupljeno 6.6.2023.)

[12] WhatIs, dostupno na https://www.techtarget.com/whatis/definition/bootstrap#:~:text=Bootstrap%20is%20a%20free%2C%20open,of%20syntax%20for%20template%20designs. (Pristupljeno 6.6.2023.)

[13] Get Bootstrap, dostupno na: https://getbootstrap.com/docs/5.0/layout/grid/ (Pristupljeno 6.6.2023.)

[14] MDN, dostupno na: https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_getting_started (Pristpuljeno 6.6.2023.)

[15] Edacutive, dostupno na: https://www.educative.io/answers/what-is-firebase (Pristupljeno 7.6.2023.)

[16] Firebase, dostupno na https://firebase.google.com/docs/storage/web/start#:~:text=Cloud%20Storage%20for%20Firebase%20lets,high%20availability%20and%20global%20redundancy. (Pristupljeno 7.6.2023.)

[17] Google Cloud, dostupno na https://cloud.google.com/firestore/docs#:~:text=Firestore%20is%20a%20NoSQL%20document,describes%20relationships%20between%20data%20objects. (Pristupljeno 7.6.2023.)

SAŽETAK

Web aplikacija pod nazivom „Web stranica plesnog studija Hajp” izrađena je alatima React *framework*-a, JavaScript metodama i Firebase sustavom, a izgled prikazanih elemenata dizajniran pomoću Bootstrap-a i CSS-a. Ovisno o stanju korisnika, web stranica nudi razne mogućnosti. Gost može listati gotovo sve podstranice, no nije mu dostupna funkcionalnost upisa na određene plesne programe, dok se registriran korisnik može upisati i ispisati sa željenog programa.

Ključne riječi: Bootstrap, CSS, Firebase, JavaScript, Ples, React, Web aplikacija

ABSTRACT

Web application called “dance studio Hajp website” was created using React framework tools, JavaScript methods and Firebase functions. The appearance of the displayed elements was designed using Bootstrap and CSS. Whether or not the user is logged in, the website offers various options. A guest can browse almost all subpages, but the functionality of enrolling in specific dance programs is not available to him. On the other side, a registered user can enroll and withdraw from a desired program.

Keywords: Bootstrap, CSS, Dance, Firebase, JavaScript, React, Web application

ŽIVOTOPIS

Benjamin Montibeler rođen je 26.6.1999. u Vinkovcima. Pohađao je osnovnu školu „August Cesarec” u Ivankovu. 2018. godine završio je prirodoslovno-matematički smijer gimnazije Matije Antuna Reljkovića u Vinkovcima. 2019. Godine upisuje stručni studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku.

Potpis autora