Društvena mreža za ljubitelje glazbe

Crnogorac, Bernarda

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

Sveučilišni studij

DRUŠTVENA MREŽA ZA LJUBITELJE GLAZBE

Završni rad

Bernarda Crnogorac

Osijek, 2021.

SADRŽAJ

1. UVOD

Glazba je jedna od glavnih tema današnjih web stranica koje privlače mnoge korisnike. Recenzija glazbe je jedna od bitnih stavki kod odabira slušanja, slično kao i kod filmova. Mnoge društvene mreže pružaju mogućnost ocjenjivanja filmova i serija te komunikaciju među korisnicima o pogledanom sadržaju. Korisnik će uvijek odabrati film s boljom recenzijom te se paralela može povući i s glazbom.

Društvena mreža s ciljem recenzije glazbe je cilj ovog završnog rada. Web aplikacija nudi mogućnost kreiranja korisničkog računa na brz i efikasan način kako bi se omogućio pristup društvenoj mreži bez prevelikog napora. Također, moguće je anonimno dati ocjenu albumu i ostaviti komentar na određeni album. Omogućeno je i dodavanje novih albuma putem admina. Kako bi korisničko iskustvo bilo unaprijeđeno, moguć je pristup trima listama albuma: nova izdanja, najviše ocjenjeni albumi i nadolazeći albumi. Shodno tome, korisnik u svakome trenu može provjeriti je li album izašao, kada će izaći, kako se drugim korisnicima svidio album te koje albume su korisnici označili najboljima. Naposlijetku, glavna prednost web aplikacije je interakcija korisnika međusobno s ciljem razmijenjivanja mišljenja o najdražim albumima.

Rad je predstavljen u pet poglavlja, u kojem su prva dva poglavlja kratak uvod o web aplikaciji i zadatku završnog rada te opisi svih potrebnih tehnologija (HTML, CSS, Angular, JavaScript, Google Firebase). Nadalje, treće poglavlje je sastavljeno od objašnjenja razvojne okoline Angular framework-a, svih potrebnih datoteka te plan izrade aplikacije. Naposlijetku je predstavljeno programsko rješenje, izrada sučelja i potrebne logike za rad aplikacije. Četvrto poglavlje daje izgled napravljenje web aplikacije naziva *BEAts* te sve mogućnosti izrađene u prethodnom poglavlju. Na kraju, peto poglavlje sažima rad u kratkom zaključku.

1.1. Zadatak završnog rada

U ovom završnom radu potrebno je analizirati očekivanja i zahtjeve korisnika društvenih mreža usmjerene glazbi. Nužno je osmisliti poboljšanje korisničkog iskustva prilikom korištenja web stranice omogućavanjem kreiranja korisničkog računa te njegovo uređivanje, reprodukcija glazbe, kao i ostavljanje povratnih informacija i ocjena vidljivih drugim korisnicima. Uz sve navedeno, potrebno je osmisliti bazu podataka te opisati programsko rješenje potrebnim programskim tehnologijama. Nadalje, potrebno je ispitati ostvareno programsko rješenje slučajevima korištenja te analizirati njegovu uspješnost prema korisničkom iskustvu.

2. PREGLED PODRUČJA TEME

2.1. Pregled sličnih rješenja

Recenzija glazbe je već postojana tema na internetu. Web stranice i aplikacije kao YouTube, Pitchfork i Genius sve imaju mogućnost recenzije glazbe, no na drugačije načine [1][2][3]. YouTube funkcionira na način ostavljanja komentara ispod glazbenih videa i ostavljanja povratne informacije u smislu „sviđa mi se“ i „ne sviđa mi se“, no nedostaje sveobuhvatna ocjena albuma i/ili pjesama. Pitchfork se više može opisati kao portal za vijesti iz svijeta glazbe, koji piše članke o albumima te tako formira recenziju samih albuma. Nedostatak je nemogućnost interakcije korisnika sa stranicom i što korisnik sam ne može ostaviti recenziju glazbe, već ju vrše stručnjaci. Genius je web aplikacija koja nudi tekst pjesama na više jezika i ostavljanja bilježaka o značenjima istih. Nema mogućnost ostavljanja konkretne ocjene te se to smatra nedostatkom.

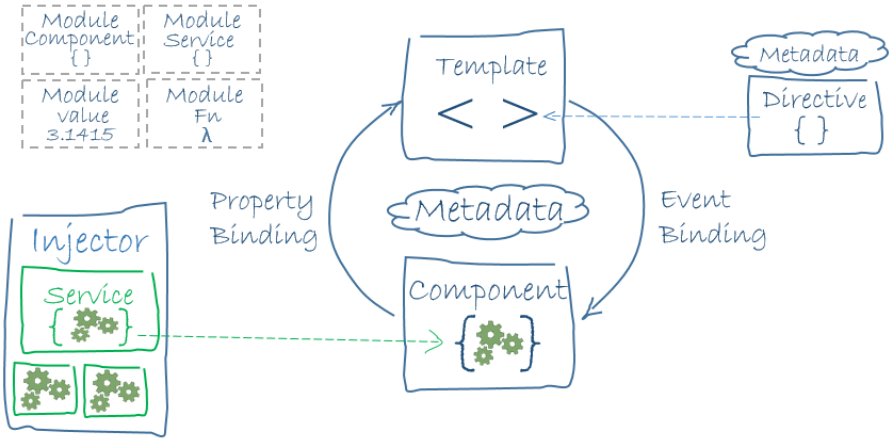
Kao slično rješenje, može se spomenuti i web aplikacija IMDb koja nudi recenziju filmova [4]. Korisnik ima mogućnost ostaviti ocjenu i komentar na film, koja na kraju služi kao recenzija samog filma. IMDb je jedan od glavnih web mjesta koji omogućava recenziju od strane samih korisnika koja zapravo utječe na ostale korisnike, odnosno utjeće na to hoće li drugi korisnik pogledati film ili ne.

2.2. Pregled korištenih tehnologija

2.2.1. Angular

Angular, također zvan Angular 2+, je besplatan softver otvorenog koda (engl. *open source*) i razvijajuća platforma za web aplikacije zasnovana na TypeScript-u. TypeScript je jezik otvorenog koda (engl. *open-source language*) koji je građen na JavaScriptu-u, te je svaki TypeScript kod naposljetku preveden u JavaScript kod [5]. Angular nudi framework zasnovan na komponentama za građenje web aplikacija koje imaju daljnu mogućnost rasta i razvoja, zbirku dobro integriranih biblioteka koje sadržavaju mnogo značajki (uključujući klijent-server rukovodstvo, usmjeravanje (engl. *routing*) i dr.) te mnoge razvojne alate pogodne za razvijanje, građenje (engl. *build*), testiranje i ažuriranje koda. Kvaliteta i kompleksnost web aplikacija postaju sve veće, isto kao i očekivanja korisnika. Angular nudi programerima ostvarenje sve većih zahtijeva korisnika bez previše napora [6]. Prednost Angulara kao razvijajuće platforme je mogućnost razvijanja malih projekata napravljenje od strane samo jednog programera do većih projekata razvijenih u tvrtkama.

Cijela arhitektura Angular aplikacije je prikazana na slici 2.1.:



***Slika 2.1.**. Arhitektura Angular aplikacije*

Angular koristi ideju komponenti kao gradivni blokovi od koje se sastoji web aplikacija. Svaka komponenta sadrži svoj HTML predložak (engl. *template*), te isto tako i CSS predložak koji definiraju ponašanje i izgled komponente. Prednost ideje komponenti je inkapsulacija i intuitivna struktura same aplikacije. Također, aplikacije su lakše pri *unit testing*-u te je poboljšana čistljivost koda i samo snalaženje u kodu [7].

Angular koristi module, što su mjesta gdje se mogu grupirati komponente, direktive i dr. koji su vezani uz aplikaciju. Kao primjer, ako se razvija web aplikacija, zaglavlje, podnožje te lijeva, desna i centralna strana postaju dio modula. Također, spominju se *data-binding* kao tehnika povezivanja podataka između komponente i pogleda, te *property-binding* kao metoda povezivanja koja pomaže pri postavljanju vrijednosti za svojstva HTML elemenata ili direktiva. Naposljetku, Angular nudi mogućnost direktiva, tj. klasa koje dodaju novo ponašanje već postojećem elementu. Glavna prednost je mogućnost kreiranja vlastite direktive te definiranje vlastitih ponašanja za određeni element.

2.2.2. JavaScript

JavaScript je objektno-orijentirani jezik kreiran za proširenje web stranica kodom koji se izvodi na klijentskoj strani. JavaScript nema klase, kao većina tradicionalnih jezika poput Jave ili C#, te ne potiče inkapsulaciju, već dovodi fleksibilnost do najviše točke. Zasniva se na tehnici objekata koji sadrži svojstva [8]. Cilj JavaScript jezika je dodavanje interaktivnosti HTML stranicama. Točnije, JavaScript je interpreter, što predstavlja izvršavanje naredbe po naredbu skripte bez prethodnog prevođenja cijelog programa i kreiranje izršne datoteke. Jedna od velikih prednosti ovog skriptnog jezika je javna raspoloživost, odnosno nepotreba za licencom za korištenje.

JavaScript pruža mnoge mogućnosti, poput pretvaranja dinamičkog teksta u HTML stranicu, reagiranje na događaje, provjera ispravnosti i vjerodostojnosti koda, detektiranje preglednika kojeg korisnik koristi te kreiranje kolačića (engl. *cookies*).

2.2.3. HTML

HTML, ili HyperText Markup Language, je najosnovniji gradivni dio današnjeg Weba. To je prezentacijski označni jezik za kreiranje hipertekstualnih datoteka (datoteke koje sadržavaju poveznice). Strogo gledajući, HTML nije programski jezik, već jednostavan jezik za oblikovanje web stranica. Koristi se u kombinaciji s CSS-om, odnosno Cascading Style Sheet, koji web stranici pruža dizajn i oblikovanje. Web preglednik omogućuje prikaz HTML dokumenta koji se sastoji od oznaka (engl. *tags*). Oznake definiraju kako će web preglednik prikazivati tekst, slike ili bilo koji drugi sadržaj sadržan u HTML dokumentu [9]. Nedostatak čistih HTML stranica je statičnost, te se danas koriste razne tehnologije poput JavaScripta radi dodavanja dinamičnosti web stranici.

2.2.4. Bootstrap CSS

Bootstrap CSS je besplatan framework otvorenog koda (engl. *open source*). Sadrži CSS (Cascading Style Sheet) i, izborno, dizajnerske predloške zasnovane na JavaScriptu za tipografiju, forme, gumbe, navigaciju i ostale komponente. Svrha korištenja Bootstrapa je pojednostavljenje razvoja web stranica jer se koriste Bootstrap-ovi izbori boja, veličine fonta te izgled projekta. Također, pružene su osnovne definicije svih HTML elemenata što dovodi do ujednačenog izgleda formi, tablica i ostalih elemenata na svim web preglednicima. Kao dodatak, Bootstrap nudi predefinirane tamne i svijetle teme za tablice, zaglavlja stranica, istaknuti tekst itd.

2.2.5. Google Firebase

Firebase je platforma za kreiranje mobilnih i web aplikacija. Najpoznatiji alat je *Realtime Database*, odnosno API koji sinkronizira podatke aplikacije širom iOS-a, Androida i Web uređaja te ih sprema na Firebase-ov oblak. Ostali alati koje platforma nudi pokrivaju veliku količinu usluga koju bi programer inače morao sam razviti, kao što su analitika, ovjeravanje autentičnosti, konfiguracija, pohrana podataka, usluge poslužitelja itd. Svi servisi su pruženi iz oblaka, što znači da imaju *backend* komponentu koja je u potpunosti održavana i vođena od strane Google-a. Razlog tomu je olakšanje samog razvojnog procesa programeru koji se tada može usredotočiti na uskustvo aplikacije i zahtjeve korisnika. Razvojni proces je drugačiji nego tradicionalni, koji tipično uključuje razvoj i *frontend* i *backend* dijela, te se to nudi kao jedna od puno prednosti Firebase platforme. Također, skalira se po potrebi, dramatično smanjuje vrijeme razvijanja programeru te možda najprivlačniji razlog korištenja, besplatan je [10].

Za završni rad su korištene tri usluge Firebase platforme, *Realtime Database, Firebase Hosting* i *Authentification*. *Realtime Database* je zapravo NoSQL baza podataka koja ima različite optimizacije i funkcionalnosti u usporedbi s relacijskom bazom te je pružena iz oblaka. Ključne sposobnosti uključuju sinkronizaciju podataka umjesto slanje HTTP zahtjeva, pristup bazi bez potrebe za aplikacijom poslužitelja, očuvanje podataka pri gubitku veze zbog lokalnog spremanja podataka te ovjeravanje i sigurnost podataka [11].

Firebase Hosting je usluga namijenjena za modernije web programere jer nudi cijelu infrastrukturu, značajke i alate napravljene po mjeri za uspostavljanje za posluživanje i održavanje web stranica i aplikacija. Svi podaci se šalju s računala programera na *Firebase Hosting* poslužitelje. Sadržaj je uvijek siguran prilikom prosluživanja web stranica i aplikacija te pruža potporu za posluživanje različitih sadržaja, od CSS i HTML datoteka do samih API-ja. [12]

Authentification usluga nudi razne alate od prijavljivanja korisnika do upravljanja prijavljenim korisnicima. Postoje razni oblici prijave, kao što su prijava email adresom, Google, Facebook, Twitter ili Yahoo računom te mobilnim brojem. Također, Firebase nudi mogućnost ugradnje svog sustava potvrde autentičnosti [13].

3. MODEL RJEŠENJA WEB APLIKACIJE

Pošto je Angular vrlo raznolik framework koji nudi puno mogućnosti, u ovom poglavlju je opisan proces postavljanja razvojne okoline te sam plan aplikacije. Prikazane su sve naredbe potrebne za instalaciju i kreiranje novog projekta te konfiguracija projekta. Shodno tome, prikazan je plan po kojemu će se aplikacija završnog rada kreirati. Kako bi se pobliže objasnilo kako Angular funkcionira, također su opisane sve korištene datoteke i razvojni blokovi. Nadalje, predstavljen je kod potreban za rad aplikacije s potrebnim objašnjenjima.

3.1. Postavljanje razvojne okoline

Kako bi se kreirao i uspješno radio novi Angular projekt, potrebno je instalirati Angular CLI (engl. *Command-line interface*) i Node.js koji će CLI koristiti u pozadini. Koristi se paket *npm* (engl. *Node Package Manager*) koji održava različite ovisnosti Angular projekta, odnosno sve biblioteke koji Angular koristi.

Nakon instalacije *npm* paketa preko interneta i Angular CLI narednom *npm install -g @angular/cli*, u terminal se unesu tri naredbe za kreiranje novog projekta (programski kod 3.1.).

```
ng new my-first-project
cd my-first-project
ng serve
```

Programski kod 3.1. *Naredbe za kreiranje novog projekta*

Zadnja naredba, *ng serve*, predstavlja naredbu za uspostavljanje veze s lokalnim serverom te posluživanje stranice. U ovome projektu se koristi localhost:4200.

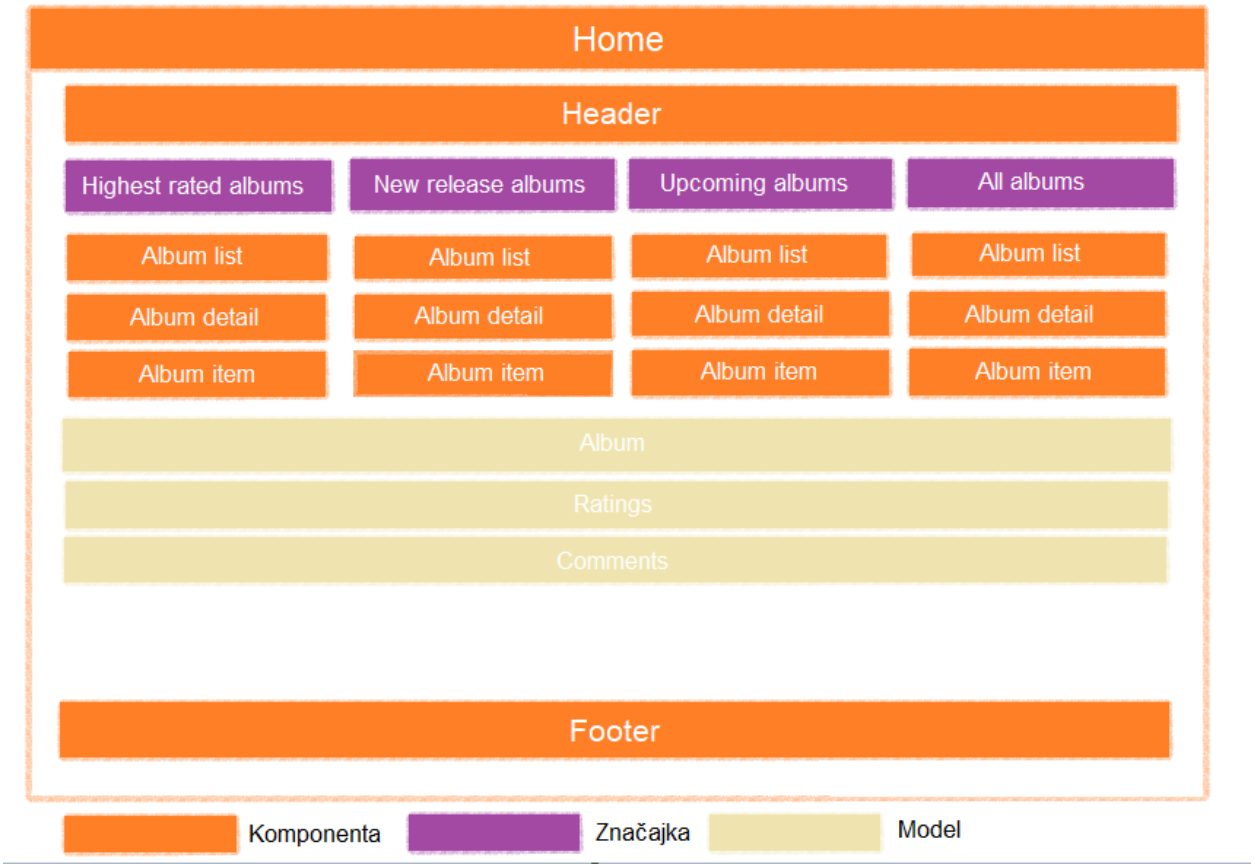
Kako bi se mogao koristiti Bootstrap u projektu, unese se naredba *npm install –save bootstrap@4* pošto se koristi verzija 4. Naredba instalira Boostrap, no ne informira CLI da Bootstrap treba biti ukjučen u paket prema zadanim postavkama. Stoga, u *angular.json* pod *styles* polje, potrebno je dodati Bootstrap datoteku kako bi CLI znao da se koristi novi stilski obrazac (programski kod 3.2.).

```
25      "styles": [
26        "node_modules/bootstrap/dist/css/bootstrap.min.css"
27      ],
```

Programksi kod 3.2. *Dodavanje Bootstrap stilskog obrasca*

3.2. Planiranje aplikacije i pojašnjenje korištenih datoteka

Angular je framework koji radi na temelju komponenti, koje međusobno komuniciraju i izmijenjuju se ovisno kako programer odluči. Potrebno je napraviti plan komponenti s obzirom na temu završnog rada, društvena mreža za ljubitelje glazbe usmjerena na recenziju albuma (slika 3.3.).



Slika 3.3. *Okvirni plan aplikacije*

Na početnoj *home* stranici će se nalaziti 4 glavne značajke: najviše ocijenjeni albumi (engl. *highest rated albums*), nova izdanja (engl. *new release albums*), nadolazeći albumi (engl. *upcoming albums*) i svi albumi (engl. *all albums*).

Za značajke je potrebno kreirati komponente naredbom *ng generate component ime_komponente* te odgovarajuće podkomponente. Pošto se neke komponente ponavljaju, moguće je kreirati jednu komponentu te ju koristiti gdje god bude potrebna. Jedna komponenta sadrži 3 važne datoteke, HTML predložak gdje se opisuje izgled komponente, CSS stilski obrazac i TypeScript datoteka

koja sadrži svu logiku ponašanja komponente. Kao i u cijelom Angular projektu, imenovanje je vrlo bitno te se komponente imenuju *ime_komponente.component.** [1]

Kako bi komponente znale kakve oblike podataka očekivati, kreiraju se modeli. Modeli su prvobitno obične klase koje pružaju definiciju kompleksnijih podataka koji se koriste u projektu. Modeli se imenuju *ime_modela.model.ts*. U slučaju završnog rada, koristi se model albuma, odnosno koja sva svojstva album treba sadržavati kako bi se pobliže opisao.

Pošto postoji pravilo da komonente ne bi trebale same upravljati podacima, poput dohvaćanja i spremanja, nego bi se trebale fokusirati na predstavljanje već obrađenih podataka, odgovornost upravljanja se prebacuje na servise. Servisi su također klase koje raspolažu s podacima i upravljaju njima, primjerice dohvaćanje podataka sa servera, sortiranje polja, ispitivanje je li polje sortirano i sl. Servisi su odličan način za dijeljenje podataka među klasama koje „se ne poznaju“, odnosno nemaju direktan pristup jedna drugoj. U pravilu se kreiraju unutar datoteke gdje se koriste te s imenuju *ime_servisa.service.ts*. U završnom radu će biti potreban servis za albume koji će sadržavati logiku za sortiranje albuma, dohvaćanje albuma, dodavanje ocjena i komentara i dr.

Također, postoji jedan oblik servisa koji se naziva *resolver*. Sadrži istu logiku kao i obični servis, no koristi se u situaciji kada su podaci dohvaćeni tek nakon što je komponenta spremna, stoga podaci na postoje. *Resolver* tada dohvai podatke prije nego je komponenta spremna te je problem izbjegnut. Imenuje se *ime_servisa-resolver.service.ts.* U završnom radu bit će korišten *resolver* koji će dohvatiti sve albume sa servera prije nego se komponente učitaju, tako da možemo obraditi same podatke u običnom servisu.

Kada dolazi do problema ovjerenja autentičnosti te same prijave u web aplikaciju, postoji klasa koja riješava problem pristupa nekoj ruti ako korisnik nije prijavljen. Naziva se *guard* te, kao što i sam naziv predlaže, čuva određenu rutu osim ako određeni uvjet nije ispunjen. *Guard* je obična klasa je implementira sučelje *CanActivate* te sučelje nudi metodu istog naziva. Metoda prvobitno odlučuje hoće li određena ruta biti aktivirana ili ne. Imenuje se *ime.guard.ts*. U završnom radu se kreira *guard* koji odlučuje je li korisnik prijavljen, te ako je, dopušta mu pristup aplikaciji.

Naposlijetku, koriste se vrlo bitne značajke zvani moduli. Moduli su način kako Angular pakira sve blokove potrebne za aplikaciju zajedno, odnosno komponente, direktive, servise itd. Potrebno

[1] .* predstavlja .html, .ts ili .css ekstenzije

je grupirati sve blokove zajedno u modul kako bi Angular znao za sve značajke korištene u projektu, pošto se automatski ne skeniraju sve datoteke u projektu. U projektu uvijek mora postojati osnovni modul, nazvan *app.module.ts* gdje su definirane sve korištene komponente, servisi, direktive itd. Također, mogu se kreirati novi moduli za, primjerice, sve rute koje projekt sadrži. U završnom radu bit će kreiran *app-routing.module.ts* koji će sadržavati sve moguće rute i koja ruta zahtjeva učitavanje koje komponente. U ovome modulu će biti navedene sve *guard* i *resolver* klase potrebne za određene rute.

3.3. Korištene komponente i odgovarajuće značajke

U ovome potpoglavlju predstavljeno je programsko rješenje te kreiranje svih potrebnih komponenti i elemenata za rad aplikacije. Prikazane su slike kodova radi lakšeg objašnjenja istih. Nadalje, pružene su logike rada svih elemenata te završni izgled aplikacije.

3.3.1. *Header, search-bar* i *footer* komponente

Po planu izrade aplikacije (slika 3.3.), prvo se spominje početna *home* komponenta. Ta komponenta sadrži još dvije osnovne komponente: *header* i *footer*.

Header komponenta sadrži gumb za početnu stranicu, polje za pretraživanje svih albuma i gumb za odjavljivanje (programski kod 3.4.).

```
<nav class="navbar navbar-expand-lg navbar-light bg-light justify-content-between fixed-top ">
    <div class="container justify-content-center">
        <a class="navbar-brand font-weight-light display:inline;"
          style="font-family: 'Dosis', sans-serif; font-size: 25px;">
          <span style="font-weight: 600;">BEA</span>ts</a>

          <a style="margin-right: 20px;"></a>
       <div class="collapse navbar-collapse" id="navbarSupportedContent">
       <ul class="navbar-nav mr-auto">
            <li class="nav-item active" routerLinkActive="active" *ngIf="isAuthed">
            <a class="nav-link font-weight-light"
              style="font-size: large;"
              routerLink="/home">Home <span class="sr-only">(current) </span></a>
            </li>
            <li class="nav-item active" routerLinkActive="active" *ngIf="!isAuthed">
              <a class="nav-link font-weight-light" routerLink="/auth" >Authenticate</a>
            </li>
       </ul>
       </div>
    </div>

  <app-search-bar></app-search-bar>
```

Programski kod 3.4. *Header.component.html – 1.dio*

```
<div class="container d-flex justify-content-end">
    <ul class="navbar-nav" *ngIf="isAuthed">
        <li class="nav-item dropdown">
            <a class="nav-link dropdown-toggle font-weight-light"
               id="navbarDropdown"
               role="button"
               data-toggle="dropdown"
               aria-haspopup="true"
               aria-expanded="false"
               style="font-size: large;">Me</a>
            <div class="dropdown-menu dropdown-menu-right" aria-labelledby="navbarDropdown">
              <a class="dropdown-item" style="cursor: pointer; font-family: Dosis;"
               routerLink="/add-new-album"
               *ngIf="albumService.getUser() == 'admin@admin.com'">Add an album</a>
              <div class="dropdown-divider" *ngIf="albumService.getUser() == 'admin@admin.com'"></div>
              <a class="dropdown-item" style="cursor: pointer; font-family: Dosis;" (click)="onLogout()">Logout</a>
            </div>
          </li>
      </ul>
  </div>
</nav>
```

Programski kod 3.5. *Header.component.html – 2.dio*

Ovisno je li korisnik prijavljen ili ne, gumb *home* se mijenja u gumb *authenticate* (programski kod 3.5.). Slično tome, ako korisnik nije prijavljen, ne može pristupiti padajućem izborniku gdje se nalazi gumb za odjavljivanje. U slučaju da je prijavljeni korisnik admin (*admin@admin.com*), može pristupiti gumbu za kreiranje novog albuma. Logika za odjavu korisnika je smještena u servis za ovjeru autentičnosti te se poziva u *header* komponenti (programski kod 3.6.). Nadalje, pri pokretanju komponente, provjeravamo je li korisnik prijavljen te pozivamo metodu servisa za albume koja će dohvatiti sve albume.

```
  ngOnInit() {
    this.userSub = this.authService.user.subscribe(user => {
      this.isAuthed = !!user;
    });

    this.subscription = this.albumService.albumsChanged
      .subscribe(
        (albums: Album[]) => {
          this.albums = albums;
        }
      );
    this.albums = this.albumService.getAlbums()
  }

  onLogout() {
    this.authService.logout();
  }

  ngOnDestroy() {
    this.userSub.unsubscribe();
  }
}
```

Programksi kod 3.6. *Header.component.ts*

Kako bi smanjili kod i povećali preglednost, pretraživanje albuma je izdvojeno kao zasebna komponenta *search-bar* (programksi kod 3.7.). Cijela komponenta će se prikazati samo ako je korisnik prijavljen. Kako bi se uredio izgled liste albuma, dopušta se samo dvadeset znakova naslova albuma, u protivnom se nadodaju tri točkice nakon dvadesetog znaka. Nakon što se klikne na željeni album, prikaže se detaljan pregled tog albuma.

```
<div class="dropdown" *ngIf="isAuthed">
  <button style="display: inline;"
    class="btn btn-light font-weight-light dropdown-toggle"
    type="button" id="dropdownMenu2"
    data-toggle="dropdown"
    aria-haspopup="true"
    aria-expanded="false">
    <img src="https://icons-for-free.com/iconfiles/png/512/search+icon-1320186817137746225.png"
         alt="search icon"
         style="display: inline; width: 17px; height: 17px; margin-right: 5px;">
    Click to search
  </button>
  <div class="dropdown-menu" aria-labelledby="dropdownMenu2">

    <a class="dropdown-item">
      <form class="form-inline container justify-content-center" >
        <input class="form-control mr-sm-2 col-md-12"
               type="search"
               placeholder="Try 'The Wall'"
               [(ngModel)]="searchText"
               name="Search bar"
               aria-label="Search">
      </form>
    </a>
    <div class="kontenjer" style="max-height: 500px; width: 380px;">
      <a class="dropdown-item" type="button" *ngFor="let album of albums | filter: searchText">
          <a *ngIf="searchText" class="row container"
             routerLinkActive="true"
             [routerLink]= "'/all-albums/' + albumService.getAlbumIndex(album)" >

            <div class="col">
              <img [src]="album.imagePath" alt="album cover" style="height: 100px; width: 100px">
            </div>

            <div class="col">
              <div class="row" style="font-family: Dosis; color: black">
                {{ ( album.name.length > 20) ? (album.name | slice:0:20) + '..' : album.name }}
              </div>
              <div class="row" style="font-family: Dosis; color: gray;">
                {{ album.artist }}
              </div>
            </div>
          </a>
      </a>
    </div>
  </div>
</div>
```

Programksi kod 3.7. *Search-bar.component.html*

Kako bi došli do svih albuma, kreirana je posebna klasa *pipe* nazvana *filter.pipe.ts* (programski kod 3.8.) koja sadrži logiku pretraživanja i filtriranja albuma u metodi *transform()*.

```typescript
import { Pipe, PipeTransform } from '@angular/core';
import { Album } from './shared/album/album.model';

@Pipe({
  name: 'filter'
})
export class FilterPipe implements PipeTransform {

  transform(items: Album[], searchText: string): Album[] {

    searchText = searchText ? searchText.toLocaleLowerCase() : null;
    return searchText ? items.filter((album: Album) =>
        album.name.toLocaleLowerCase().indexOf(searchText) !== -1) : items;

  }
}
```

Programski kod 3.8. *Filter.pipe.ts*

Footer komponenta je vrlo jednostavnog izgleda. Sadrži naziv stranice, naziv programera koji je razvio stranicu i poveznicu na LinkedIn profil programera (programski kod 3.9.).

```html
<nav class="navbar fixed-bottom navbar-light bg-light">
  <div class="container">
    <div class="justify-content-center align-items-center">

      <a class="navbar-brand font-weight-light;" style="font-family: 'Dosis', sans-serif; font-size: 17px;">/
        <span style="font-weight: 600;">BEA</span>ts  /</a>

        <div class="row">
          <p class="font-weight-light"
             style="font-size: 13px; display:inline;">Copyright © 2021 Bernarda Crnogorac</p>
          <a style="margin-right: 5px;"></a>
          <a style="margin-right: 5px;"></a>
          <a href="https://hr.linkedin.com/in/bernardacrnogorac"
             style="display:inline;"
             class="linkedIn">
            <img src="https://eaes.eu/wp-content/uploads/2017/05/linkedin-logo.png"
                 alt="LinkedIn"
                 style="width: 17px; height: 17px; display:inline;"
                 class="align-baseline">
          </a>
        </div>
      </div>
    </div>
</nav>
```

Programski kod 3.9. *Footer.component.html*

3.3.2. *Album* model, *album* i svi vezani servisi

Prvo se izrađuje model albuma (programski kod 3.10.) koji sadržava svojstva imena albuma, imena pjevača/pjevačice, poveznica na sliku albuma, polje ocjena, polje komentara, datum izdavanja albuma, opis albuma, poveznicu na Spotify te YouTube gdje se mogu poslušati albumi.

```typescript
import { Comment } from "./comment.model";
import { Ratings } from "./rating.model";

export class Album {
    public name: string;
    public artist: string;
    public imagePath: string;
    public ratings: Ratings[];
    public comments: Comment[];
    public releaseDate: Date;
    public description: string;
    public linkSpotify: string;
    public linkYoutube: string;

    constructor(name: string,
                artist: string,
                imagePath: string,
                ratings: Ratings[],
                comments: Comment[],
                releaseDate: Date,
                description: string,
                linkSpotify: string,
                linkYoutube: string){
        this.name = name;
        this.artist = artist;
        this.imagePath = imagePath;
        this.ratings = ratings;
        this.comments = comments;
        this.releaseDate = releaseDate;
        this.description = description;
        this.linkSpotify = linkSpotify;
        this.linkYoutube = linkYoutube;
    }
}
```

Programski kod 3.10. *Album.model.ts*

Nadalje, kreirana su dodatna dva modela za ocjenu (programski kod 3.11.) i komentare (programski kod 3.12.) koje koristi osnovni model albuma.

```typescript
export class Ratings {
    constructor(public mail: string, public rating: number) {}
}
```

Programski kod 3.11. *Rating.model.ts*

```typescript
export class Comment {
    constructor(public mail: string, public text: string) {}
}
```

Programski kod 3.12. *Comment.model.ts*

Kako bi mogli dohvatiti podatke s Firebase baze podataka, kreira se *data-storage* servis (programski kod 3.13.). Preko klase *HttpClient* pružene od strane Angulara, mogu se dohvatiti albumi metodom *get()* koja prima poveznicu na Firebase bazu podataka. Operator map uzima

podatke i preoblikuje ih tako što ako album nema polje ocjena, neka se stavi prazno polje, te isto tako i za polje komentara. Operator tap poziva metodu postavljanja albuma iz osnovnog servisa za albume.

```
15      fetchAlbums(){
16          return this.http
17              .get<Album[]>('https://beats-5369e-default-rtdb.firebaseio.com/albums.json')
18              .pipe(
19                      map(albums => {
20                          return albums.map(album => {
21                              return {...album,
22                                      ratings: album.ratings ? album.ratings: [],
23                                      comments: album.comments ? album.comments: []
24                                  };
25                                  });
26                      }),
27                      tap(albums => {
28                          this.albumService.setAlbums(albums);
29                      })
30                  )
31      }
32  }
```

Programski kod 3.13. *Data-storage.service.ts*

Servis za albuma sadrži svu logiku za upravljanje albumima. Prvo, albumi se postavljaju nakon što su dohvaćeni sa servera. Pružene su metode za dohvaćanje trenutnih albuma, dohvaćanje albuma poredanih po nazivu od A-Z, dohvaćanje trenutnog albuma i dohvaćanje jednog sortiranog albuma (programski kod 3.14.).

```
18      setAlbums(albums: Album[]){
19          this.albums = albums;
20          this.albumsChanged.next(this.albums.slice());
21      }
22
23      getAlbums(){
24          return this.albums.slice();
25      }
26
27      getSortedAlbums(){
28          let albums: Album[] = this.getAlbums();
29
30          albums.sort( (a, b) =>
31              a.name > b.name ? 1 : -1
32          );
33
34          return albums.slice();
35      }
36
37      getOneAlbum(index: number){
38          return this.albums[index];
39      }
40
41      getOneSortedAlbum(index: number){
42          let allAlbums: Album[] = this.getSortedAlbums();
43
44          return allAlbums[index];
45
46      }
```

Programski kod 3.14. *Album.service.ts. – 1. dio*

Nadalje, dane su metode za dohvaćanje indeksa trenutnog albuma, dohvaćanje sveukupne ocjene određenog albuma i dohvaćanje svih komentara za određeni album (programski kod 3.15.). Kako bi izračunali sveukupnu ocjenu albuma, dohvaćaju se sve ocjene albuma, zbrajaju te dijele s brojem sveukupnih ocjena. Kako bi prikazali samo jednu decimalu, koristi se predefinirana *Math.round()* metoda. Dohvaćanje komentara funkcionira na sličan način, prolazi se kroz polje svih komentara i spremaju se u novo polje koje se na kraju prikazuje.

```
48      getAlbumIndex(album: Album){
49          return this.getSortedAlbums().indexOf(album);
50      }
51
52      getAlbumRating(album: Album){
53          let ratingAvg: number = 0;
54
55          if(album.ratings.length > 0){
56              for(let index = 0; index < album.ratings.length; index++){
57                  ratingAvg = ratingAvg + album.ratings[index].rating;
58              }
59
60              ratingAvg = ratingAvg / album.ratings.length;
61              return Math.round(ratingAvg*10)/10;
62          }
63          return 'Not rated yet';
64      }
65
66      getComments(album: Album){
67
68          let allComments: Comment[] = [];
69          console.log(allComments);
70
71         if(album.comments.length > 0){
72          for(let index = 0; index < album.comments.length; index++){
73              allComments.push(album.comments[index]);
74          }
75          return allComments;
76         }
77      }
```

Programski kod 3.15. *Album.service.ts. – 2. dio*

Sortiranje polja albuma da bi dobili albume s najvećom ocijenom (programski kod 3.16.) funkcionira tako što se prolazi kroz polje albuma i odabiru samo albumi s ocjenom 5. U slučaju da ih ima više, dohvaćaju se samo prva tri albuma s najvišom ocjenom. Također, pružena je i metoda za dohvaćanje samo određenog albuma s najvećom ocjenom.

```
79      getHighestRatedAlbums(){
80          let albumsRated: Album[] = [];
81          let albumDate: Date;
82          let oneAlbum: Album;
83
84          for (let index = 0; index < this.albums.length; index++) {
85              if(this.getAlbumRating(this.albums[index]) == 5 ){
86                  oneAlbum = this.albums[index];
87                  albumDate = this.albums[index].releaseDate;
88                  albumsRated.push(this.albums[index]);
89              }
90          }
91          return albumsRated.slice().slice(0, 3);
92      }
93
94      getOneHighest(index: number){
95          let albumsHighest: Album[] = this.getHighestRatedAlbums();
96
97          return albumsHighest[index];
98      }
```

Programski kod 3.16. *Album.service.ts. – 3. dio*

Dobivanje nadolazećih albuma (programksi kod 3.17.) radi na način što prolazeći kroz polje albuma, uspoređuju se datum izdanja albuma i današnji datum. Ako je današnji datum „manji“ od datuma izdanja albuma, album još nije izašao te se sprema u novo polje koje se na kraju i prikazuje. Također, pružena je i metoda za dohvaćanje samo određenog nadolazećeg albuma.

```
100     getUpcomingAlbums(){
101         let albumsUpcoming: Album[] = [];
102         let currentDate: Date = new Date();
103         let albumDate: Date;
104
105         for(let index = 0; index < this.albums.length; index++){
106             albumDate = new Date(this.albums[index].releaseDate)
107             if(albumDate > currentDate){
108                 albumsUpcoming.push(this.albums[index]);
109             }
110         }
111
112         return albumsUpcoming.slice();
113     }
114
115     getOneUpcoming(index: number){
116         let albumsUpcoming: Album[] = this.getUpcomingAlbums();
117
118         return albumsUpcoming[index];
119     }
```

Programski kod 3.17. *Album.service.ts. – 4. dio*

Kao novi album, uzeti su svi albumi izdani u posljednih četiri mjeseca. Prvo se treba provjeriti koji je mjesec u pitanju kada se oduzmu četiri mjeseca. Primjerice, ako je album izašao u studenom mjesecu 2020. godine, a današnji datum je veljača 2021. godine, taj album dolazi u obzir kod novih izdanja. Stoga, za taj slučaj se kreiraju novi indeksi umjesto mjeseca i godine te se tek tada kreira

novi granični datum – odnosno, datum prije kojeg se album ne smatra novim izdanjem. Nadalje, datumi izdanja albuma se provjeravaju s graničnim datumom i trenutnim datumom. Ako je datum albuma unutar spomenutog raspona, album se smatra novim izdanjem i sprema u novo polje koje se na kraju i prikazuje. Također, pružena je i metoda za dohvaćanje samo određenog novog izdanja (programski kod 3.18.).

```
121     getNewReleases(){
122         let albumsNewReleases: Album[] = [];
123         let currentDate: Date = new Date();
124         let razlika = currentDate.getMonth() - 4;
125         let date: Date;
126         let day: number = currentDate.getDate();
127         let month: number;
128         let year: number;
129         let albumDate: Date;
130 
131         if(razlika < 0){
132             month= 12 + razlika + 1;
133             year = currentDate.getFullYear() - 1;
134 
135         }
136         else{
137             month = razlika + 1;
138             year = currentDate.getFullYear();
139         }
140 
141         date = new Date(`${year}-${month}-${day}`);
142 
143         for(let index = 0; index < this.albums.length; index++){
144             albumDate = new Date(this.albums[index].releaseDate)
145             if(albumDate >= date && albumDate <= currentDate){
146                 albumsNewReleases.push(this.albums[index]);
147             }
148         }
149         return albumsNewReleases.slice();
150     }
151 
152     getOneNewRelease(index: number){
153         let albumsNewReleases: Album[] = this.getNewReleases();
154 
155         return albumsNewReleases[index];
156     }
```

Programski kod 3.18. *Album.service.ts. – 5. dio*

Dodavanje ocjene određenom albumu radi na način da se dohvati trenutni korisnik iz lokalnog skladišta (engl. *local storage*) Google Chrome-a i uzme njegova email adresa. Nakon, u polje *ratings* albuma nadodaje se novi *rating* s argumentima emaila korisnika i same ocjene. Zatim se metodom *put()* albumi spreme na Firebase bazu podataka. Dodavanje komentara na određeni album te samo dodavanje albuma funkcioniraju na isti način (programski kod 3.19. i programski kod 3.20.).

```
158     addRating(album: Album, rating: number){
159         const userData: {
160             email: string,
161             id: string,
162             _token: string,
163             _tokenExpDate: string
164         } = JSON.parse(localStorage.getItem('userData'));
165 
166         let email: string = userData.email;
167         let ratings = new Ratings(email, rating);
168 
169         album.ratings.push(ratings);
170 
171         this.http.put('https://beats-5369e-default-rtdb.firebaseio.com/albums.json',
172                       this.getAlbums()).subscribe(response =>
173                        console.log(response));
174     }
175 
176     addComment(album: Album, text: string){
177         const userData: {
178             email: string,
179             id: string,
180             _token: string,
181             _tokenExpDate: string
182         } = JSON.parse(localStorage.getItem('userData'));
183 
184         let email: string = userData.email;
185         let comment = new Comment(email, text);
186 
187         album.comments.push(comment);
188 
189         this.http.put('https://beats-5369e-default-rtdb.firebaseio.com/albums.json',
190                       this.getAlbums()).subscribe(response =>
191                        console.log(response));
192     }
```

Programski kod 3.19. *Album.service.ts. – 6. dio*

```
194     addAlbum(name: string, artist: string, description: string, releaseDate: Date,
195             imgURL: string, spotify?: string, youtube?: string){
196 
197         let newAlbum: Album = new Album(name, artist, imgURL, [], [],
198                                         releaseDate, description, spotify ? spotify : '',
199                                         youtube ? youtube: '' );
200 
201 
202         this.albums.push(newAlbum);
203         this.albumsChanged.next(this.albums.slice());
204 
205         this.http.put('https://beats-5369e-default-rtdb.firebaseio.com/albums.json',
206                       this.getAlbums()).subscribe(response =>
207                       console.log(response));
208 
209     }
```

Programski kod 3.20.. *Album.service.ts. – 7. dio*

Kako bi znali je li korisnik ocijenio album ili ne, dodana je metoda koja provjerava. Također, metoda dodana je i metoda koja, pri unosu novog albuma, provjerava postoji li taj album već u bazi podataka. Naposlijetku, imamo metodu koja dohvaća trenutnog korisnika (programski kod 3.21.).

```
    checkIfVoted(album: Album){
        const userData: {
            email: string,
            id: string,
            _token: string,
            _tokenExpDate: string
        } = JSON.parse(localStorage.getItem('userData'));

        let email: string = userData.email;

        for(let index = 0; index < album.ratings.length; index++){
            if(album.ratings[index].mail === email){
                return true;
            }
        }
        return false;
    }

    checkIfAlbumExists(name: string, artist: string){

        for(let index = 0; index < this.albums.length; index++){
            if((this.albums[index].name === name) && (this.albums[index].artist === artist)){
                return true;
            }
        }
        return false;
    }

    getUser(){
        const userData: {
            email: string,
            id: string,
            _token: string,
            _tokenExpDate: string
        } = JSON.parse(localStorage.getItem('userData'));

        let email: string = userData.email;

        return email;
    }
}
```

Programski kod 3.21. *Album.service.ts – 8. dio*

Kako bi dohvatili albume prije nego se komponenta uopće uspije pokrenuti, koristi se *album-resolver* servis. Implementira se sučelje *Resolve* koje nudi metodu *resolve()* u kojoj se ispituje postoje li albumi, te ako ne, dohvati ih (programski kod 3.22.).

```
export class AlbumResolverService implements Resolve<Album[]>{

    constructor(private dsService: DataStorageService,
                private albumService: AlbumService){}

    resolve(route: ActivatedRouteSnapshot, state: RouterStateSnapshot){
        const albums = this.albumService.getAlbums()

        if(albums.length === 0){
            return this.dsService.fetchAlbums();
        }
        else {
            return albums;
        }
    }
}
```

Programski kod 3.22. *Album-resolver.service.ts*

3.3.3. *Home* i sve vezane komponente

Prema planu izrade (slika 3.3.), *home* komponenta uključuje četiri značajke. Pošto prve tri značajke zapravo predstavljaju samo različito poredane albume, koristit će se jedna komponenta za značajke *highest rated albums, upcoming albums* i *new release albums.* Komponenta je nazvana *album-item* te sadrži jednostavan izgled jednog albuma (programski kod 3.23.).

```
<div class="d-flex flex-row">
    <a
        class="list-group-item list-group-item-action shadow-sm p-2 mb-5"
        style="pointer-events: none; border: 0px; ">


        <div class="container">
                <img
                    src="{{ album.imagePath }}"
                    alt="{{ album.name }}"
                    class="img-responsive"
                    style="width: 150px; height: 150px; margin-bottom: 10px;">
                    <h5 style="max-width: 150px; font-family: Arial;">{{ album.name }}</h5>
                    {{ album.artist }}
       </div>
    </a>
</div>
```

Programski kod 3.23. *Album-item.component.html*

Home komponenta tri *album-item* komponente koje drugačije filtriraju podatke. Prvo se dohvaćaju sva polja već sortiranih albuma iz album servisa (programski kod 3.24.) te se u HTML predlošku prikazuju dohvaćena polja (programski kod 3.25, programski kod 3.26. i programski kod 3.27.). Komponenti *all-albums* se pristupa putem gumba koji se nalazi nakon prethodno spomenutih značajki (programski kod 3.28.).

```
export class HomeComponent implements OnInit {


  constructor(private albumService: AlbumService) { }

  albumsRated: Album[];
  albumsUpcoming: Album[];
  albumsNewReleases: Album[];


  ngOnInit() {
    this.albumsRated = this.albumService.getHighestRatedAlbums();
    this.albumsUpcoming = this.albumService.getUpcomingAlbums();
    this.albumsNewReleases = this.albumService.getNewReleases();
  }
}
```

Programski kod 3.24. *Home.component.ts*

```
<div class="d-flex bd-highlight">
    <div class="p-2 flex-fill bd-highlight">
      <div class="d-flex flex-column bd-highlight mb-3">
        <div class="p-2 bd-highlight">

        <a
            style="cursor: pointer;"
            routerLinkActive="true"
            routerLink="/highest-rated-albums">

            <div class="container list-group-item" style="border: 0px;">
                <div class="row">
                    <div class="col">
                        <h3 class="col-md-5"
                            style="font-family: Dosis; pointer-events: none; color: ☐black"
                            >Highest Rated Albums</h3>
                        <hr>
                        <div class="row" style="margin-left: 20px;">
                                <app-album-item
                                *ngFor="let albumItem of albumsRated; let i = index"
                                [album]="albumItem"
                                [index]="i"></app-album-item>
                        </div>
                    </div>
                </div>
            </div>
        </a>
        </div>
```

Programski kod 3.25. *Home.component.html – 1. dio*

```
        <div class="p-2 bd-highlight">

            <a
                style="cursor: pointer;"
                routerLinkActive="true"
                routerLink="/upcoming-albums">

            <div class="container list-group-item" style="border: 0px;">
                <div class="row">
                    <div class="col">
                        <h3 class="col-md-5"
                            style="font-family: Dosis; pointer-events: none; color: ☐black"
                            >Upcoming Albums</h3>
                        <hr>
                        <div class="row" style="margin-left: 20px;">
                            <app-album-item
                            *ngFor="let albumItem of albumsUpcoming; let i = index"
                            [album]="albumItem"
                            [index]="i"></app-album-item>
                            </div>
                        </div>
                    </div>
                </div>
            </a>
        </div>
      </div>
    </div>
```

Programski kod 3.26. *Home.component.html – 2. dio*

```
58      <div class="p-2 flex-fill bd-highlight">
59          <div class="d-flex flex-column bd-highlight mb-3">
60              <div class="p-2 bd-highlight">
61
62                  <a
63                  style="cursor: pointer;"
64                  routerLinkActive="true"
65                  routerLink="/new-releases">
66
67                  <div class="container list-group-item" style="border: 0px; ">
68                          <div class="row">
69                              <div class="col">
70                                  <h3 class="col-md-5"
71                                      style="font-family: Dosis; pointer-events: none; color: black"
72                                      >New Releases</h3>
73                                  <hr>
74                                  <div class="row" style="margin-left: 20px;">
75                                      <app-album-item
76                                      *ngFor="let albumItem of albumsNewReleases; let i = index"
77                                      [album]="albumItem"
78                                      [index]="i"></app-album-item>
79
80                                  </div>
81                              </div>
82                          </div>
83                      </div>
84                  </a>
85              </div>
```

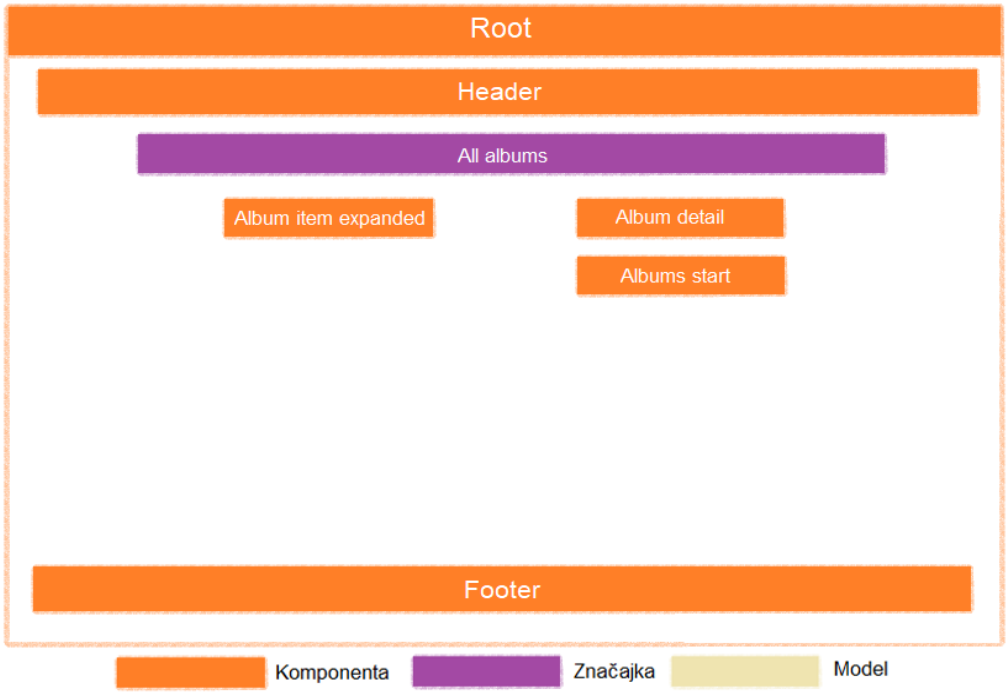
Programski kod 3.27. *Home.component.html – 3. dio*

```
87              <div class="p-2 bd-highlight">
88              <br><br><br><br><br><br>
89
90                  <div class="row justify-content-center">
91                      <button
92                          class="btn btn-light btn-lg font-weight-light shadow-sm p-3 mb-5 rounded"
93                          routerLinkActive="true"
94                          routerLink="/all-albums">Browse all albums!</button>
95                  </div>
96              </div>
97          </div>
98      </div>
99  </div>
```

Programski kod 3.28. *Home.component.html – 4. dio*

3.3.4. *All albums* i sve vezane komponente

Klikom na gumb *Browse all albums* u *home* komponenti, odlazi se na novu rutu. Plan izgleda *all albums* značajke (slika 3.29.) sastoji se od liste svih albuma s lijeve strane te prikaza odabranog albuma s desne strane. Kako bi to bilo moguće, koriste se tri nove komponente: *album-item-expanded*, *album-detail* i *albums-start*.



Slika 3.29. *Plan izgleda all albums značajke*

Album-item-expanded komponenta pruža prikaz jednog albuma u listi (programski kod 3.30.). Kako bi prikazali samo godinu datuma izdanja albuma, koristi se *pipe* date. Nadalje, kod prikaza sveukupne ocjene albuma poziva se metoda *album* servisa *getAlbumRating()* koja kao argument prima trenutni album. Tokom prikaza opisa albuma, smanjen je prikaz na samo 100 znakova *slice()* metodom.

```
<a
    class="list-group-item list-group-item-action "
    style="cursor: pointer; margin-left: 10px;"
    [routerLink]="[index]"
    routerLinkActive="active">

    <div class="media">
        <img
            class="align-self-start mr-3 img-responsive"
            src="{{ album.imagePath }}"
            alt="{{ album.name }}"
            style="width: 150px; height: 150px;">

        <div class="media-body">
        <h5 class="mt-0" style="font-family: Arial;">{{ album.name }}</h5>
        <h6 style="display:inline; font-family: Arial;">{{ album.artist }} · </h6>
        <h6 style="display:inline; color: rgb(180, 173, 173); font-family: Arial;">
            {{ album.releaseDate | date: 'yyyy'}}
        </h6>
        <p> <img src="https://freepngimg.com/thumb/star/19-star-png-image-thumb.png"
                 style="width: 17px; height: 17px;"
                 class="align-text-bottom">
            {{ albumService.getAlbumRating(album) }}</p>
        <p style="font-family: Arial;">
            {{ ((album.description).length > 100) ? (album.description | slice:0:100) + '..' : (album.description)  }}</p>
        </div>
    </div>
</a>
```

Programski kod 3.30. *Album-item-expanded.component.html*

Albums-start komponenta nudi jednostavan dizajn za početni zaslon kada nije odabran niti jedan album (programski kod 3.31.).

```
<div class="container h-100 ">
    <div class="row h-100 justify-content-center align-items-center">
        <div class="form-group bg-light shadow-sm p-3 mb-5 rounded">
            <h5 class="font-weight-light">Please select an album!</h5>
        </div>
    </div>
</div>
```

Programski kod 3.31. *Albums-start.component.html*

Album-detail komponenta pruža izgled odabranog albuma (programski kod 3.32. i programski kod 3.33.). Slično kao i u *album-item-expanded* komponenti, dohvaćaju se sva svojstva albuma. Prikazuje se koliko korisnika je ocijenilo album, kao i ukupna ocjena. Ako korisnik nije ocijenio album, prikazuje se mogućnost ocjenjivanja albuma. U protivnost, korisnik ne vidi opciju ocjenjivanja već tekst da je već ocijenio (programski kod 3.34.)

```
<div class="kontenjer">
  <div class="media">
    <img
    class="align-self-center mr-3 shadow-sm p-2 mb-5"
    [src]=" album.imagePath "
    alt="{{ album.name }}"
    style="width: 300px; height: 300px; background-color: rgb(255, 255, 255);">
    <div class="media-body">
      <hr>

      <h4 class="mt-0 " style="display:inline; ">{{ album.name }} </h4>
      <h5 style="display:inline;" class="align-baseline"></h5>
        <img src="https://freepngimg.com/thumb/star/19-star-png-image-thumb.png"
            style="width: 30px; height: 30px; cursor: pointer;"
            class="align-bottom ime">
      <a style="margin-right: 5px;"></a>

      <h5 style="display:inline;" class="align-text-bottom font-weight-light">
        {{ albumService.getAlbumRating(album) }}
      </h5>
      <a style="margin-right: 10px;"></a>
      <a [href]="album.linkSpotify" *ngIf="album.linkSpotify ? true : false">
        <img src="https://seeklogo.com/images/S/spotify-2015-logo-560E071CB7-seeklogo.com.png"
            alt="Spotify"
            style="width: 20px; height: 20px; display:inline;"
            class="align-text-bottom">
      </a>
      <a style="margin-right: 10px;"></a>
        <a [href]="album.linkYoutube" *ngIf="album.linkYoutube ? true : false">
          <img src="https://www.pinclipart.com/picdir/middle/530-5305220_youtube-icon-flat-red-play-button-logo-vector.png"
              alt="Youtube"
              style="width: 23px; height: 20px; display:inline;"
              class="align-text-bottom">
        </a>
```

Programski kod 3.32. *Album-detail.component.html – 1.dio*

```
<p style="display:inline; margin-left: 70px; color: gray;" class="font-weight-light" *ngIf="album.ratings.length > 0">
  Rated by {{album.ratings.length }} {{ album.ratings.length == 1 ? 'user' : 'users'}}
</p>
<h6>{{ album.artist }}</h6>

<p class="mb-0 font-weight-light" style="text-align: justify; margin-right: 20px;">{{ album.description }}</p>
<p style="display:inline;" >Released on: {{ album.releaseDate | date: 'mediumDate'}}</p>
<hr>
</div>
</div>
```

Slika 3.33. *Album-detail.component.html – 2.dio*

```
<div class="bg-light shadow-sm p-3 mb-5 rounded" style="margin-right: 20px; margin-left: 20px;">
  <br>
  <div class="row justify-content-center">
  <br>
    <div class="font-weight-light">{{ !isRated ? (album.linkSpotify ? 'Rate the album!' : 'Cannot rate yet!') : 'Album already rated!' }}</div>
  </div>
  <br>

  <div *ngIf="!isRated && album.linkSpotify ? true : false">

  <hr style="width: 5vh;">
    <div class="row justify-content-center">
      <button class="ratingDugme btn card-text btn-lg" (click)="onAddRating(album, 1)">1</button>
      <a style="margin-right: 5px;"></a>
      <button class="ratingDugme btn card-text btn-light btn-lg" (click)="onAddRating(album, 2)">2</button>
      <a style="margin-right: 5px;"></a>
      <button class="ratingDugme btn card-text btn-light btn-lg" (click)="onAddRating(album, 3)">3</button>
      <a style="margin-right: 5px;"></a>
      <button class="ratingDugme btn card-text btn-light btn-lg" (click)="onAddRating(album, 4)">4</button>
      <a style="margin-right: 5px;"></a>
      <button class="ratingDugme btn card-text btn-light btn-lg" (click)="onAddRating(album, 5)">5</button>
    </div>
    <br><br>
  </div>
```

Slika 3.34. *Album-detail.component.html – 3.dio*

Korisnik može ostaviti komentar na album, ocijenio album ili ne. U slučaju da nijedan korisnik nije komentirao, prikazuje se tekst da nema komentara. U protivnom, tekst se ne prikazuje. Kako bi dohvatili komentare određenog albuma, poziva se metoda album servisa *getComments()* koja kao argument prihvaća trenutni album (programski kod 3.35. i programski kod 3.36.).

```
<div *ngIf="album.linkSpotify.length > 0">
  <label style="font-family: Dosis; font-size: 30px; margin-left: 10px; display: inline;">Comments</label>
  <p style="font-style: italic; font-family: Dosis; display: inline; margin-left: 10px;"
    *ngIf="!albumService.getComments(album)">No comments yet!</p>
  <hr style="margin-top: 0px; margin-right: 20px;">

  <form #form="ngForm" (ngSubmit)="onAddComment(album, text.value, form)">
    <div class="form-group">

      <input
        style="margin-left: 60px; margin-bottom: 5px; width: 80%; display: inline;"
        placeholder="Add comment"
        type="text"
        class="form-control"
        ngModel
        name="text"
        required
        #text/>
```

Slika 3.35. *Album-detail.component.html – 4.dio*

```
        <button type="submit" class="btn btn-secondary align-baseline"
                style="display: inline; font-size: 15px; margin-left: 5px; width: 10%;">Add</button>
      </div>
      </form>

      <div class="komentari" style="margin-left: 60px; margin-right: 80px;">
        <div *ngFor="let comment of albumService.getComments(album)"
             style="margin-bottom: 5px; margin-bottom: 15px;"
             class="border-top-0 shadow-sm">
          <div class="col">
            <h4 style="font-family: Dosis; font-style: italic; display: inline; font-size: 17px; color: gray;">
              {{ comment.mail + ':'}}
            </h4>
            <p style="display: inline; margin-left: 5px; font-size: 18px;" class="text-break font-weight-light">
              {{ comment.text }}
            </p>
          </div>
        </div>
      </div>
    </div>
    <br>
  </div>
</div>
```

Slika 3.36. *Album-detail.component.html – 5.dio*

Kako bi korisnik mogao ostaviti komentar i ocjenu na album, u komponenti se poziva metoda album servisa *addRating()* i *addComment()* klikom na odgovarajuće gumbe (programski kod 3.37.). Nakon dodavanja komentara, unos se resetira.

```
  ngOnInit() {

    this.route.params
      .subscribe(
        (params: Params) => {
          this.id = +params['id'];
          this.album = this.albumService.getOneSortedAlbum(this.id);
          this.isRated = this.albumService.checkIfVoted(this.album);
        }
      );

      this.comments = this.albumService.getComments(this.album);
  }

  onAddRating(album: Album, rating: number){
    this.albumService.addRating(album, rating);
    this.isRated = this.albumService.checkIfVoted(album);
  }

  onAddComment(album: Album, text: string, form: NgForm){
    this.albumService.addComment(album, text);

    form.reset();
  }
}
```

Programski kod 3.37. *Album-detail.component.ts*

Naposlijetku, u roditeljskoj komponenti *all albums* poslažu se sve prethodno kreirane komponente (programski kod 3.38.). *Album-item-expanded* komponenta se prikazuje kao lista svih sortiranih albuma koji se dohvaćaju pri pokretanju iz album servisa metodom *getSortedAlbums()* (programski kod 3.39.)

```
<div class="row">
    <div class="col kontenjer">
        <app-album-item-expanded
        *ngFor="let albumItem of allAlbums; let i = index"
            [album]="albumItem"
            [index]="i"></app-album-item-expanded>
    </div>
  <div class="col kontenjer">
    <router-outlet></router-outlet>
  </div>
</div>
```

Programski kod 3.38. *All-albums.component.html*

```
import { Component, OnInit } from '@angular/core';
import { Subscription } from 'rxjs';
import { Album } from '../shared/album/album.model';
import { AlbumService } from '../shared/album/album.service';

@Component({
  selector: 'app-all-albums',
  templateUrl: './all-albums.component.html',
  styleUrls: ['./all-albums.component.css']
})
export class AllAlbumsComponent implements OnInit {

  constructor(private albumService: AlbumService) { }

  allAlbums: Album[];
  subscription: Subscription;

  ngOnInit() {
    this.subscription = this.albumService.albumsChanged
      .subscribe(
        (allAlbums: Album[]) => {
          this.allAlbums = allAlbums;
        }
      );
    this.allAlbums = this.albumService.getSortedAlbums();
  }
}
```

Programski kod 3.39. *All-albums.component.ts*

3.3.5. *Highest rated albums* i sve vezane komponente

Plan izgleda *highest rated albums* značajke isti je kao i prethodne *all albums* značajke (slika 3.29.). Kreiraju se tri nove komponente: *highest rated expanded, album-detail-highest* i *albums-start*. Zadnja komponenta već je kreirana te se ista ponovno koristi.

Highest rated expanded komponenta (programski kod 3.40.) sadrži listu svih sortiranih albuma, prikazanih preko već napravljene komponente *album-item-expanded* (programski kod 3.29.)

```
<div class="row">
    <div class="col">
        <h2 class="font-weight-light"
            style="margin-left: 20px; margin-right: 20px; font-family: Dosis; display: inline;">
            Highest Rated Albums
        </h2>
        <p style="display: inline; font-family: Dosis; font-style: italic;">
            Browse highest rated albums!
        </p><hr>
        <app-album-item-expanded
           *ngFor="let albumItem of albumsRated; let i = index"
            [album]="albumItem"
            [index]="i"></app-album-item-expanded>
    </div>
</div>
```

Slika 3.40. *Highest-rated-expanded.component.html*

Kako bi dobili listu najviše ocijenjenih albuma, poziva se metoda album servisa *getHighestRatedAlbums()* (programski kod 3.41.)

```
export class HighestRatedExpandedComponent implements OnInit, OnDestroy {

  constructor(private albumService: AlbumService) { }

  albumsRated: Album[];
  subscription: Subscription;

  ngOnInit() {
    this.subscription = this.albumService.albumsChanged
      .subscribe(
        (albumsRated: Album[]) => {
          this.albumsRated = albumsRated;
        }
      );
    this.albumsRated = this.albumService.getHighestRatedAlbums();
  }

  ngOnDestroy() {
    this.subscription.unsubscribe();
  }
}
```

Programski kod 3.41. *Highest-rated-expanded.component.ts*

Album-detail-highest koristi isti HTML predložak kao i *album-detail* komponenta (programski kod 3.32., programski kod 3.33., programski kod 3.34.). Razlika je u dohvaćanju podataka, pošto trenutna komponenta dohvaća sve albume koje su najviše ocijenjeni (programski kod 3.42.).

```
ngOnInit() {
  this.route.params
    .subscribe(
      (params: Params) => {
        this.id = +params['id'];
        this.album = this.albumService.getOneHighest(this.id);
        this.isRated = this.albumService.checkIfVoted(this.album);
      }
    );
}
```

Programski kod 3.42. *Album-detail-highest.component.ts*

Naposlijetku, u roditeljsku komponentu *highest-rated* se poslažu prethodno kreirane komponente (programski kod 3.43.).

```
<div class="row">
    <div class="col kontenjer1">
        <app-highest-rated-expanded></app-highest-rated-expanded>
    </div>
    <div class="col kontenjer2">
        <router-outlet></router-outlet>
    </div>
</div>
```

Programski kod 3.43. *Highest-rated.component.html*

3.3.6. *New releases* i sve vezane komponente

Plan izgleda *new releases* značajke isti je kao i od *all albums* značajke (slika 3.29.). Kreiraju se tri nove komponente: *new releases expanded, album-detail-new* i *albums-start*. Zadnja komponenta već je kreirana te se ista ponovno koristi. *New releases expanded* (programski kod 3.44.) sadrži listu svih sortiranih albuma, prikazanih preko već napravljene komponente *album-item-expanded* (programski kod 3.30.)

```
<div class="row">
    <div class="col">
        <h2 class="font-weight-light"
            style="margin-left: 20px; margin-right: 20px; font-family: Dosis; display: inline;">
            New releases
        </h2>
        <p style="display: inline; font-family: Dosis; font-style: italic;">
            Browse newly released albums!
        </p><hr>
        <app-album-item-expanded
           *ngFor="let albumItem of albumsNewReleases; let i = index"
            [album]="albumItem"
            [index]="i"></app-album-item-expanded>
    </div>
</div>
```

Programski kod 3.44. *New-releases-expanded.component.html*

Kako bi dobili listu najviše ocijenjenih albuma, poziva se metoda album servisa *getNewReleases()* (programski kod 3.45.).

```
export class NewReleasesExpandedComponent implements OnInit, OnDestroy {
  albumsNewReleases: Album[];
  subscription: Subscription;

  constructor(private albumService: AlbumService) { }


  ngOnInit() {
    this.subscription = this.albumService.albumsChanged
      .subscribe(
        (albumsNewReleases: Album[]) => {
          this.albumsNewReleases = albumsNewReleases;
        }
      );
    this.albumsNewReleases = this.albumService.getNewReleases();
  }

  ngOnDestroy() {
    this.subscription.unsubscribe();
  }

}
```

Programski kod 3.45. *New-releases-expanded.component.ts*

Album-detail-new koristi isti HTML predložak kao i *album-detail* komponenta (programski kod 3.32., programski kod 3.33., programski kod 3.34.). Razlika je u dohvaćanju podataka, pošto trenutna komponenta dohvaća sve albume koje su najviše ocijenjeni (programski kod 3.46.).

```
ngOnInit() {
  this.route.params
    .subscribe(
      (params: Params) => {
        this.id = +params['id'];
        this.album = this.albumService.getOneNewRelease(this.id);
        this.isRated = this.albumService.checkIfVoted(this.album);
      }
    );
}
```

Programski kod 3.46. *Album-detail-new.component.ts*

Naposlijetku, u roditeljsku komponentu *new-releases* se poslažu prethodno kreirane komponente (programski kod 3.47.).

```
<div class="row">
    <div class="col kontenjer">
        <app-new-releases-expanded></app-new-releases-expanded>
    </div>
    <div class="col kontenjer">
        <router-outlet></router-outlet>
    </div>
</div>
```

Programski kod 3.47. *New-releases.component.html*

3.3.7. *Upcoming albums* i sve vezane komponente

Plan izgleda *upcoming albums* značajke isti je kao i od *all albums* značajke (slika 3.29.). Kreiraju se tri nove komponente: *upcoming albums expanded*, *album-detail-upcoming* i *albums-start*. Zadnja komponenta već je kreirana te se ista ponovno koristi.

Upcoming albums expanded (programski kod 3.48.) sadrži listu svih sortiranih albuma, prikazanih preko već napravljene komponente *album-item-expanded* (slika 3.29.)

```
<div class="row">
    <div class="col">
        <h2 class="font-weight-light"
            style="margin-left: 20px; margin-right: 20px; font-family: Dosis; display: inline;">
            Upcoming Albums
        </h2>
            <p style="display: inline; font-family: Dosis; font-style: italic;">
                Browse upcoming albums!
            </p><hr>
        <app-album-item-expanded
            *ngFor="let albumItem of albumsUpcoming; let i = index"
            [album]="albumItem"
            [index]="i"></app-album-item-expanded>
    </div>
</div>
```

Programski kod 3.48. *Upcoming-albums-expanded.component.html*

Kako bi dobili listu najviše ocijenjenih albuma, poziva se metoda album servisa *getUpcomingAlbums()* (programski kod 3.49.).

```
export class UpcomingAlbumsExpandedComponent implements OnInit {

  constructor(private albumService: AlbumService) { }

  albumsUpcoming: Album[];
  subscription: Subscription;


  ngOnInit() {
    this.subscription = this.albumService.albumsChanged
      .subscribe(
        (albumsUpcoming: Album[]) => {
          this.albumsUpcoming = albumsUpcoming;
        }
      );
    this.albumsUpcoming = this.albumService.getUpcomingAlbums();
  }
}
```

Programski kod 3.49. *Upcoming-albums-expanded.component.ts*

Album-detail-upcoming koristi isti HTML predložak kao i *album-detail* komponenta (programski kod 3.32., programski kod 3.33., programski kod 3.34.). Razlika je u dohvaćanju podataka, pošto trenutna komponenta dohvaća sve albume koje su najviše ocijenjeni (programski kod 3.50.).

```
ngOnInit() {
  this.route.params
    .subscribe(
      (params: Params) => {
        this.id = +params['id'];
        this.album = this.albumService.getOneUpcoming(this.id);
      }
    );
}
```

Programski kod 3.50. *Album-detail-upcoming.component.ts*

Naposlijetku, u roditeljsku komponentu *upcoming-albums* se poslažu prethodno kreirane komponente (programski kod 3.51.).

```
<div class="row">
    <div class="col">
        <app-upcoming-albums-expanded></app-upcoming-albums-expanded>
    </div>
    <div class="col">
        <router-outlet></router-outlet>
    </div>
</div>
```

Programski kod 3.51. *Upcoming-albums.component.html*

3.3.8. *Add album* i sve vezane komponente

Kako bi admin mogao dodati novi album, kreira se *add album* komponenta (programski kod 3.52. i programski kod 3.53.). Pruža se *form* koji se puni podacima prepisanim album modelom (slika 3.9.). Kako admin ne bi unio album koji već postoji, prilikom unosa imena albuma i pjevača, poziva se metoda album servisa *checkIfAlbumExists()* koja ime albuma i pjevača kao argumente.

```
<div class="d-flex justify-content-center">

    <div class="d-flex flex-column">
        <div class="p-2">
        <h3 style="font-family: Dosis;">Add a new album</h3>
        <hr>
        </div>
        <div class="p-2">
            <form #myForm="ngForm" (ngSubmit)="onSubmit(name.value, artist.value, description.value,
                                               releaseDate.value, imgURL.value, myForm,
                                               spotify.value, youtube.value)">

                <div class="form-row">
                  <div class="form-group col-md-6">
                    <label for="inputAlbumName" style="font-family: Dosis;">Album Name</label>
                    <input name="name" type="text" class="form-control"
                           ngModel id="inputAlbumName" placeholder="The Wall"
                           #name required>
                  </div>

                  <div class="form-group col-md-6">
                    <label for="inputArtist" style="font-family: Dosis;">Artist</label>
                    <input name="artist" type="text" class="form-control"
                           ngModel id="inputArtist" placeholder="Pink Floyd"
                           #artist required>
                    <div class="text-danger" style="font-size: 13px;"
                        *ngIf="albumService.checkIfAlbumExists(name.value, artist.value)">
                        Album already exists!
                    </div>
                  </div>
                </div>

                <div class="form-group">
                  <label for="input">Description</label>
                  <textarea name="description" type="text" rows="5" class="form-control"
                           ngModel id="inputDescription" placeholder="The Wall is an album by..."
                           #description required></textarea>
                </div>
```

Programski kod 3.52. *Add-album.component.html – 1. dio*

```
                <div class="form-group">
                    <label for="inputAddress2">Release date</label>
                    <input name="date" type="text" class="form-control"
                           ngModel id="releaseDate" required placeholder="yyyy/dd/mm" #releaseDate>
                  </div>
                <div class="form-group">
                  <label for="inputAddress2">Image URL</label>
                  <input name="imgURL" type="text" class="form-control" ngModel id="inputImgURL" required #imgURL>
                </div>
                <div class="form-group">
                    <label for="inputAddress2">Spotify URL</label>
                    <input name="spotify" type="text" class="form-control" ngModel id="inputSpotifyURL" #spotify>
                  </div>
                  <div class="form-group">
                    <label for="inputAddress2">YouTube URL</label>
                    <input name="youtube" type="text" class="form-control" ngModel id="inputYouTubeURL" #youtube>
                  </div>
                  <button class="btn btn-secondary justify-content-center"
                          type="submit" [disabled]="!myForm.valid">
                          Submit
                  </button>
            </form>
        </div>
    </div>
</div>
```

Programski kod 3.53. *Add-album.component.html – 2. dio*

Klikom na gumb *Submit*, admin unosi album u Firebase bazu podataka. Omogućuje se pozivom funkcije album servisa *addAlbum()* koja prima sve potrebne argumente za kreiranje albuma (programski kod 3.54.)

```
export class AddAlbumComponent implements OnInit {


  constructor(public albumService: AlbumService) { }

  ngOnInit() {
  }

  onSubmit(name: string, artist: string, description: string,
           releaseDate: Date, imgURL: string, form: NgForm,
           spotify?: string, youtube?: string){

    this.albumService.addAlbum(name, artist, description,
                               releaseDate, imgURL, spotify, youtube);

    form.reset();
  }
}
```

Programski kod 3.54. *Add-album.component.ts*

Pošto samo admin može pristupiti dodavanju albuma, potrebno je kreirati *admin guard* koji će zabraniti navigaciju na rutu dodavanja albuma ako admin nije u pitanju (programski kod 3.55.).

```
export class AdminGuard implements CanActivate {

  constructor(private authService: AuthService, private router: Router) {}

  canActivate(
    route: ActivatedRouteSnapshot,
    router: RouterStateSnapshot
  ):
    | boolean
    | UrlTree
    | Promise<boolean | UrlTree>
    | Observable<boolean | UrlTree> {
    return this.authService.user.pipe(
      take(1),
      map(user => {
        const currentUser = user;
        if (currentUser.email == 'admin@admin.com') {
          return true;
        }
        return this.router.createUrlTree(['/home']);
      })
    );
  }
}
```

Programski kod 3.55. *Admin.guard.ts*

3.3.9. *Auth* i sve vezane komponente

Prije kreacije komponenti, prvo se mora saznati s kakvim će se podacima raditi. Riječ je o korisnicima te stoga se kreira novi *user* model. *User* posjeduje sva Firebase-om propisana svojstva koje korisnik pri prijavi mora sadržavati: email adresa, id korisnika, token i token do isteka prijave (programski kod 3.56.). Pružena je i funkcija koja vraća token korisnika u slučaju da korisnikov token do isteka prijave još vrijedi. Firebase nalaže da token do isteka prijave može vrijediti samo jedan sat. Nakon toga, korisnik se mora ponovno prijaviti.

```
export class User {

    constructor(public email: string,
                public id: string,
                private _token: string,
                private _tokenExpDate: Date){}

    get token(){
        if(!this._tokenExpDate || new Date() > this._tokenExpDate){
            return null;
        }
        return this._token;
    }
}
```

Programski kod 3.56. *User.model.ts*

Kako bi se korisnik mogao prijaviti u web aplikaciju, potrebno je kreirati *auth* komponent koji će držati logiku i izgled spomenutog. U HTML predlošku (programski kod 3.57. i programski kod 3.58) koristi se *form* element koji se popunjava podacima korisničkove email adrese i lozinke. Također, u slučaju greške s prijavom, ispisuje se poruka ovisno o kakvoj grešci koja je nastala. Pošto Firebase *Authentification* zahtjeva lozinku od minimalno 6 znakova, to postaje jednim od uvjeta za uspješnu prijavu. Ovisno o tome je li korisnik ispunio sve uvjete uspješne prijave, gumbovi postaje dostupni. Također, ovisno u kojem načinu rada je korisnik (prijava ili registracija), mijenja se natpis na dugmima (programski kod 3.59.).

```
<div class="view " style="background-image: url('https://uc92ba02e250007f40dbd4d5ce36.previews.d
  <div class="mask rgba-gradient align-items-center">

    <div class="container">
      <div class="row d-flex justify-content-center">
        <div class="col-md-5" style="margin-top: 250px;">

          <div class="alert alert-danger" *ngIf="error">
            <p>{{ error }}</p>
          </div>

        <div *ngIf="isLoading" style="text-align: center;">
          <app-loading-spinner></app-loading-spinner>
        </div>
```

Programski kod 3.57. *Auth.component.html – 1.dio*

```
<form #authForm="ngForm" (ngSubmit)="onSubmit(authForm)" *ngIf="!isLoading">
  <div class="form-group">
    <label for="email" style="font-family: Dosis; font-size: 20px;">E-Mail</label>
    <input
      type="email"
      id="email"
      class="form-control"
      ngModel
      name="email"
      required
      email/>
  </div>
  <div class="form-group">
    <label for="password" style="font-family: Dosis; font-size: 20px;">Password</label>
    <input
      type="password"
      id="password"
      class="form-control"
      ngModel
      name="password"
      required
      minlength="6"/>
  </div>
  <div>
```

Programski kod 3.58. *Auth.component.html – 2.dio*

```
      <button
        class="btn btn-secondary"
        type="submit"
        [disabled]="!authForm.valid">
        {{ isLoginMode ? 'Login' : 'Sign Up' }}
      </button>
      |
      <button class="btn btn-secondary" (click)="onSwitchMode()" type="button">
        Switch to {{ isLoginMode ? 'Sign Up' : 'Login' }}
      </button>
    </div>
  </form>
</div>
</div>
</div>
</div>
</div>
```

Programski kod 3.59. *Auth.component.html – 3.dio*

Klikom na gumbove registracija, korisnik se unosi u *Authentification* tablicu na Firebase-u. Također, pokreće se metoda *onSubmit()* koja kao argument prima *form* element u kojemu su svi podaci potrebni za registraciju ili prijavu. Prvo se provjerava u kojem načinu rada je korisnik, odnosno pokušava li se prijaviti ili registrirati, te ovisno o odgovoru, pozivaju se *login()* ili *signup()* funkcije (programski kod 3.59.).

```
24    onSwitchMode(){
25      this.isLoginMode = !this.isLoginMode;
26    }
27
28    onSubmit(form: NgForm){
29      if (!form.valid) {
30        return;
31      }
32
33      const email = form.value.email;
34      const password = form.value.password;
35      let authObs: Observable<AuthResponseData>;
36      this.isLoading = true;
37
38      if (this.isLoginMode) {
39        authObs = this.authService.login(email, password);
40      } else {
41        authObs = this.authService.signup(email, password);
42      }
43
44      authObs.subscribe(resData => {
45        this.isLoading = false;
46        this.router.navigate(['/home']);
47      },
48      errorMessage => {
49        this.error = errorMessage;
50        this.isLoading = false;
51    });
52      form.reset();
53    }
54    onHandleError() {
55      this.error = null;
56    }
57  }
```

Programski kod 3.60. *Auth.component.ts*

Radi odvajanja logike prikazivanja podataka i upravljanja podacima, spomenute funkcije su smještene u novi *auth* servis (programski kod 3.60.). Također, pružena je i definicija funkcije za odjavljivanje *logout()*. Kako bi trenutni korisnik ostao prijavljen u slučaju osvježavanja stranice, napisana je funcija *autoLogin()* koja provjerava posebni token koji svaki korisnik mora posjedovati. Pošto Firebase dopušta da je korisnik prijavljen maksimalno jedan sat, smanjuje se korisnikovo vrijeme do isteka prijave *expirationDate*. U slučaju da je korisnikov token istekao, odnosno da je prošao jedan sat od prijave, pokreće se funkcija *autoLogout()* koja provjerava token i pokreće funkciju *logout()* (programski kod 3.61. i programski kod 3.62.).

```
export interface AuthResponseData {
    idToken: string,
    email: string,
    refreshToken: string,
    expiresIn: string,
    localId: string ,
    registered?: boolean
}

@Injectable({
    providedIn: 'root'
})
export class AuthService {

    user = new BehaviorSubject<User>(null);
    private tokenExpTimer: any;

    constructor(private http: HttpClient,
                private router: Router){}

    signup(email: string, password: string){
        return this.http.post<AuthResponseData>('https://identitytoolkit.googleapis.com/v1/accounts:signUp?key='
                                                + environment.firebaseAPIKey,
        {
            email: email,
            password: password,
            returnSecureToken: true
        }).pipe(catchError(this.handleError),
            tap(resData => {
                this.handleAuth(resData.email, resData.localId, resData.idToken, +resData.expiresIn)
            })
        )
    }

    login(email: string, password: string){
        return this.http.post<AuthResponseData>('https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword?key='
                                                + environment.firebaseAPIKey,{
            email: email,
            password: password,
            returnSecureToken: true
        }
```

Programski kod 3.61. *Auth.service.ts – 1.dio*

```
    autoLogin(){
        const userData: {
            email: string,
            id: string,
            _token: string,
            _tokenExpDate: string
        } = JSON.parse(localStorage.getItem('userData'));
        if(!userData){
            return;
        }
        const loadedUser = new User(userData.email, userData.id,
                                    userData._token, new Date(userData._tokenExpDate));

        if(loadedUser.token){
            this.user.next(loadedUser);

            const expDuration = new Date(userData._tokenExpDate).getTime() - new Date().getTime();
            this.autoLogout(expDuration);
        }
    }
```

Programski kod 3.62. *Auth.service.ts – 2.dio*

```
56      autoLogin(){
57          const userData: {
58              email: string,
59              id: string,
60              _token: string,
61              _tokenExpDate: string
62          } = JSON.parse(localStorage.getItem('userData'));
63          if(!userData){
64              return;
65          }
66          const loadedUser = new User(userData.email, userData.id,
67                                      userData._token, new Date(userData._tokenExpDate));
68
69          if(loadedUser.token){
70              this.user.next(loadedUser);
71
72              const expDuration = new Date(userData._tokenExpDate).getTime() - new Date().getTime();
73              this.autoLogout(expDuration);
74          }
75      }
76
77      logout(){
78          this.user.next(null);
79          this.router.navigate(['/auth']);
80          localStorage.removeItem('userData');
81
82          if(this.tokenExpTimer){
83              clearTimeout(this.tokenExpTimer);
84          }
85
86          this.tokenExpTimer = null;
87      }
88
89      autoLogout(expDuration: number){
90          this.tokenExpTimer = setTimeout( () => {
91              this.logout();
92          }, expDuration);
93      }
```

Programski kod 3.63. *Auth.service.ts – 3.dio*

Funckija *handleAuth()* kao argumente prima sve argumente propisane *user* modelom te kreira novog korisnika i vrijeme do isteka prijave. Također, kako bi korisnik ostao prijavljen nakon osvježavanja stranice, funkcija sprema korisnikome podatke u *local storage* web preglednika. Naposlijetu, pružena je funkcija za poruke grešaka *handleError()*. Firebase nudi tri osnovne vrste grešaka: email adresa već postoji, email adresa nije pronađena i netočna lozinka. Sve tri greške su napisane u funkciji te ovisno koju grešku dobijemo nazad, određena poruka bude ispisana korisniku (programski kod 3.64.)

```
95      private handleAuth(email: string, userId: string, token: string, expiresIn: number){
96          const expDate = new Date(new Date().getTime() + expiresIn * 1000);
97          const user = new User(email, userId, token, expDate);
98          this.user.next(user);
99          this.autoLogout(expiresIn * 1000);
100         localStorage.setItem('userData', JSON.stringify(user));
101     }
102
103     private handleError(errorRes: HttpErrorResponse){
104         let errorMessage = 'An unknown error occurred.';
105
106             if(!errorRes.error || !errorRes.error.error){
107                 return throwError(errorMessage);
108             }
109             switch(errorRes.error.error.message){
110                 case 'EMAIL_EXISTS':
111                     errorMessage = 'This email is taken.';
112                     break;
113                 case 'EMAIL_NOT_FOUND':
114                     errorMessage = 'Email not found.';
115                     break;
116                 case 'INVALID_PASSWORD':
117                     errorMessage = 'Wrong password.';
118                     break;
119             }
120             return throwError(errorMessage);
121     }
122 }
```

Programski kod 3.64. *Auth.service.ts – 3.dio*

Slično kao kod dodavanja novog albuma, ne želimo da neprijavljeni korisnik može pristupiti ostatku aplikacije. U tom slučaju se kreira novi *auth guard* koji će „čuvati“ kod od nedopuštene navigacije (programski kod 3.64.).

```
@Injectable({ providedIn: 'root' })
export class AuthGuard implements CanActivate {
  constructor(private authService: AuthService, private router: Router) {}

  canActivate(
    route: ActivatedRouteSnapshot,
    router: RouterStateSnapshot
  ):
    | boolean
    | UrlTree
    | Promise<boolean | UrlTree>
    | Observable<boolean | UrlTree> {
    return this.authService.user.pipe(
      take(1),
      map(user => {
        const isAuth = !!user;
        if (isAuth) {
          return true;
        }
        return this.router.createUrlTree(['/auth']);
      })
    );
  }
}
```

Programski kod 3.65. *Auth.guard.ts*

Provjerom je li postoji trenutni prijavljeni korisnik, dobiva se odgovor je li navigacija na slijedeću komponentu moguća ili nije. U slučaju da nije, preusmjerava se na istu rutu prijave.

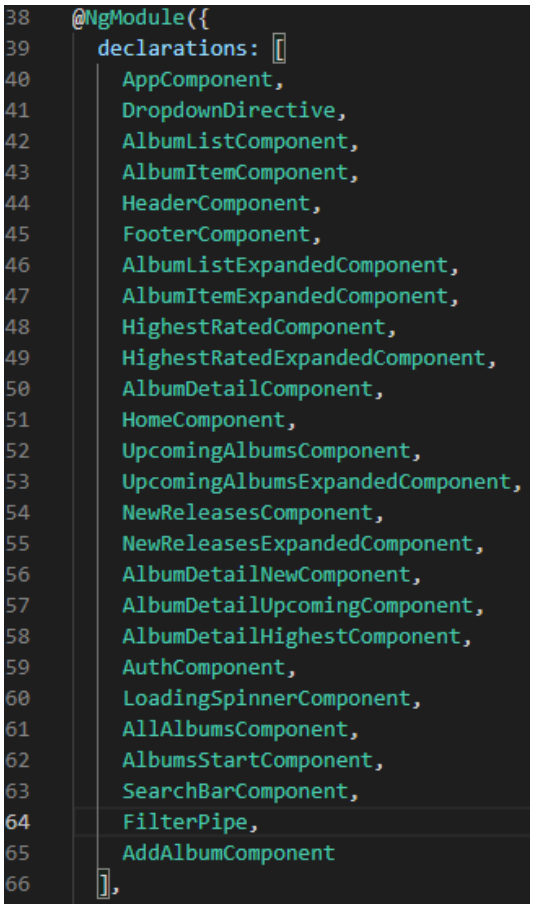
3.3.10. Moduli i osnovna komponenta

Kako bi Angular prepoznao sve dosad napravljenje komponente, direktive, *pipes*, servisi moraju se deklarirati u osnovnom modulu *app module*. Uz to, moraju se deklarirati svi korišteni moduli koji su pruženi od strane Angulara poput *FormsModule* (programski kod 3.66.)

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';

import { AppComponent } from './app.component';
import { DropdownDirective } from './shared/dropdown.directive';
import { AlbumListComponent } from './shared/album/album-list/album-list.component';
import { AlbumItemComponent } from './shared/album/album-list/album-item/album-item.component';
import { HeaderComponent } from './header/header.component';
import { FooterComponent } from './footer/footer.component';
import { AlbumListExpandedComponent } from './shared/album/album-list-expanded/album-list-expanded.component';
import { AlbumItemExpandedComponent } from './shared/album/album-list-expanded/album-item-expanded/album-item-expanded.component';
import { HighestRatedComponent } from './highest-rated/highest-rated.component';
import { HighestRatedExpandedComponent } from './highest-rated/highest-rated-expanded/highest-rated-expanded.component';
import { AppRoutingModule } from './app-routing.module';
import { AlbumDetailComponent } from './shared/album/album-detail/album-detail.component';
import { AlbumService } from './shared/album/album.service';
import { HomeComponent } from './home/home.component';
import { UpcomingAlbumsComponent } from './upcoming-albums/upcoming-albums.component';
import { UpcomingAlbumsExpandedComponent } from './upcoming-albums/upcoming-albums-expanded/upcoming-albums-expanded.component';
import { NewReleasesComponent } from './new-releases/new-releases.component';
import { NewReleasesExpandedComponent } from './new-releases/new-releases-expanded/new-releases-expanded.component';
import { HttpClientModule, HTTP_INTERCEPTORS } from '@angular/common/http';
import { AlbumDetailNewComponent } from './shared/album/album-detail/album-detail-new/album-detail-new.component';
import { AlbumDetailUpcomingComponent } from './shared/album/album-detail/album-detail-upcoming/album-detail-upcoming.component';
import { AlbumDetailHighestComponent } from './shared/album/album-detail/album-detail-highest/album-detail-highest.component';
import { AuthComponent } from './auth/auth.component';
import { LoadingSpinnerComponent } from './shared/loading-spinner/loading-spinner.component';
import { AuthInterceptorService } from './auth/auth-interceptor.service';
import { AllAlbumsComponent } from './all-albums/all-albums.component';
import { AlbumsStartComponent } from './shared/album/albums-start/albums-start.component';
import { SearchBarComponent } from './search-bar/search-bar.component';
import { FilterPipe } from './filter.pipe';
import { AddAlbumComponent } from './add-album/add-album.component';
```

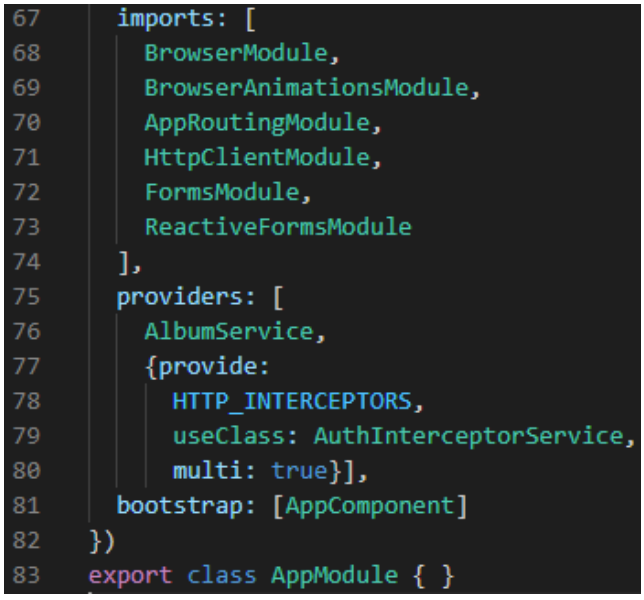
Programski kod 3.66. *App.module.ts*

U *declarations* listu sve pružene komponente se moraju navesti samo jednom kroz cijeli projekt. U *imports* listu dodaje su svi moduli koji se koriste u Angular projektu, moduli napravljeni od strane programera i oni pruženi od strane Angulara (programski kod 3.67. i programski kod 3.68.).



```
@NgModule({
  declarations: [
    AppComponent,
    DropdownDirective,
    AlbumListComponent,
    AlbumItemComponent,
    HeaderComponent,
    FooterComponent,
    AlbumListExpandedComponent,
    AlbumItemExpandedComponent,
    HighestRatedComponent,
    HighestRatedExpandedComponent,
    AlbumDetailComponent,
    HomeComponent,
    UpcomingAlbumsComponent,
    UpcomingAlbumsExpandedComponent,
    NewReleasesComponent,
    NewReleasesExpandedComponent,
    AlbumDetailNewComponent,
    AlbumDetailUpcomingComponent,
    AlbumDetailHighestComponent,
    AuthComponent,
    LoadingSpinnerComponent,
    AllAlbumsComponent,
    AlbumsStartComponent,
    SearchBarComponent,
    FilterPipe,
    AddAlbumComponent
  ],
```

Programski kod 3.67. *App.module.ts*



```
  imports: [
    BrowserModule,
    BrowserAnimationsModule,
    AppRoutingModule,
    HttpClientModule,
    FormsModule,
    ReactiveFormsModule
  ],
  providers: [
    AlbumService,
    {provide:
      HTTP_INTERCEPTORS,
      useClass: AuthInterceptorService,
      multi: true}],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Programski kod 3.68. *App.module.ts*

Kako bi znali koje se sve rute spominju u projektu i koja ruta zahtjeva koju komponentu, kreira se posebni modul *app-routing* (programski kod 3.69.). Sve moguće rute u projektu su opisane te su im dodane određene komponente. Kao dodatak, neke rute zahtijevaju *guard* klase i *resolver* servise

te je svakoj ruti pružena potrebna klasa. Primjerice, /*home* ruta učitava *HomeComponent*, koji zahtjeva *AuthGuard* koji čuva pristup ruti ako korisnik nije prijavljen te prima *AlbumResolverService* koji učitava sve albume prije kreiranja same komponente.

```
const appRoutes: Routes = [
    { path: '', redirectTo: '/home', pathMatch: 'full' },
    { path: 'home', component: HomeComponent,
        canActivate: [AuthGuard],
        resolve: [AlbumResolverService]
    },
    { path: 'highest-rated-albums', component: HighestRatedComponent,
        resolve: [AlbumResolverService],
    children: [
        { path: '', component: AlbumsStartComponent },
        { path: ':id', component: AlbumDetailHighestComponent,
            resolve: [AlbumResolverService] }
    ] },
    { path: 'upcoming-albums', component: UpcomingAlbumsComponent,
        resolve: [AlbumResolverService],
    children: [
        { path: '', component: AlbumsStartComponent },
        { path: ':id', component: AlbumDetailUpcomingComponent,
            resolve: [AlbumResolverService] }
    ] },
    { path: 'new-releases', component: NewReleasesComponent,
        resolve: [AlbumResolverService],
    children: [
        { path: '', component: AlbumsStartComponent },
        { path: ':id', component: AlbumDetailNewComponent,
            resolve: [AlbumResolverService] }
    ]},
    { path: 'all-albums', component: AllAlbumsComponent,
        resolve: [AlbumResolverService],
    children: [
        { path: '', component: AlbumsStartComponent, },
        { path: ':id', component: AlbumDetailComponent,
            resolve: [AlbumResolverService] }
    ] },
    { path: 'auth', component: AuthComponent },
    { path: 'add-new-album', component: AddAlbumComponent,
        resolve: [AlbumResolverService],
        canActivate: [AdminGuard]}
]

@NgModule({
    imports: [RouterModule.forRoot(appRoutes)],
    exports: [RouterModule]
})
export class AppRoutingModule {
}
```

Programski kod 3.69. *App-routing.module.ts*

Naposlijetku, popunjava se osnovna komponenta *app* (programski kod 3.70.). Selektor *router-outlet* predstavlja rezervirano mjesto koje Angular dinamički popunjav ovisno o trenutnoj ruti opisanoj u *app-routing* modulu.

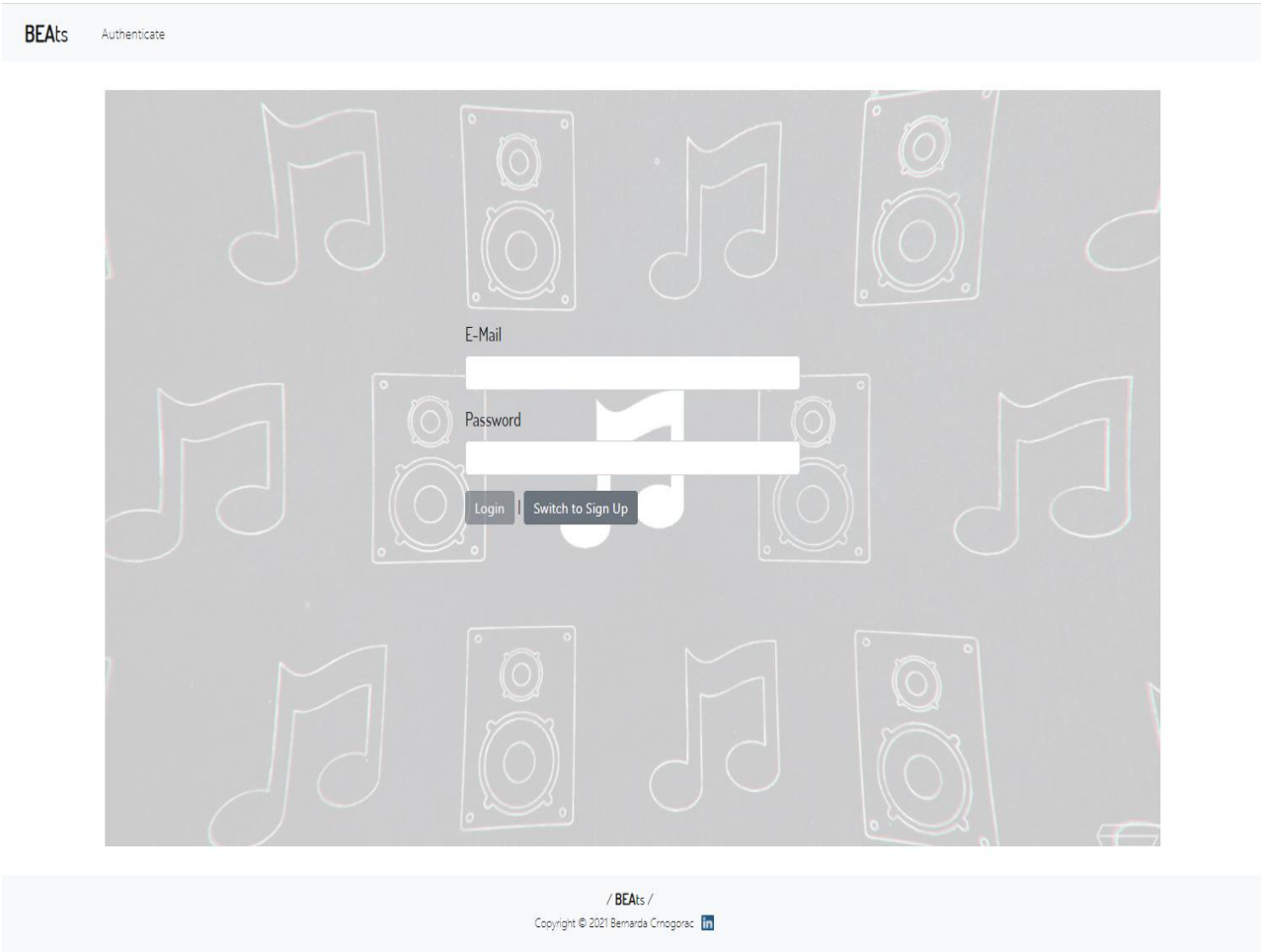
```
<app-header></app-header>
<br><br>
<div class="divizor">

    <br><br>

    <router-outlet></router-outlet>

    <br><br><br>

</div>
<app-footer></app-footer>
```

Programski kod 3.70. *App.component.html*

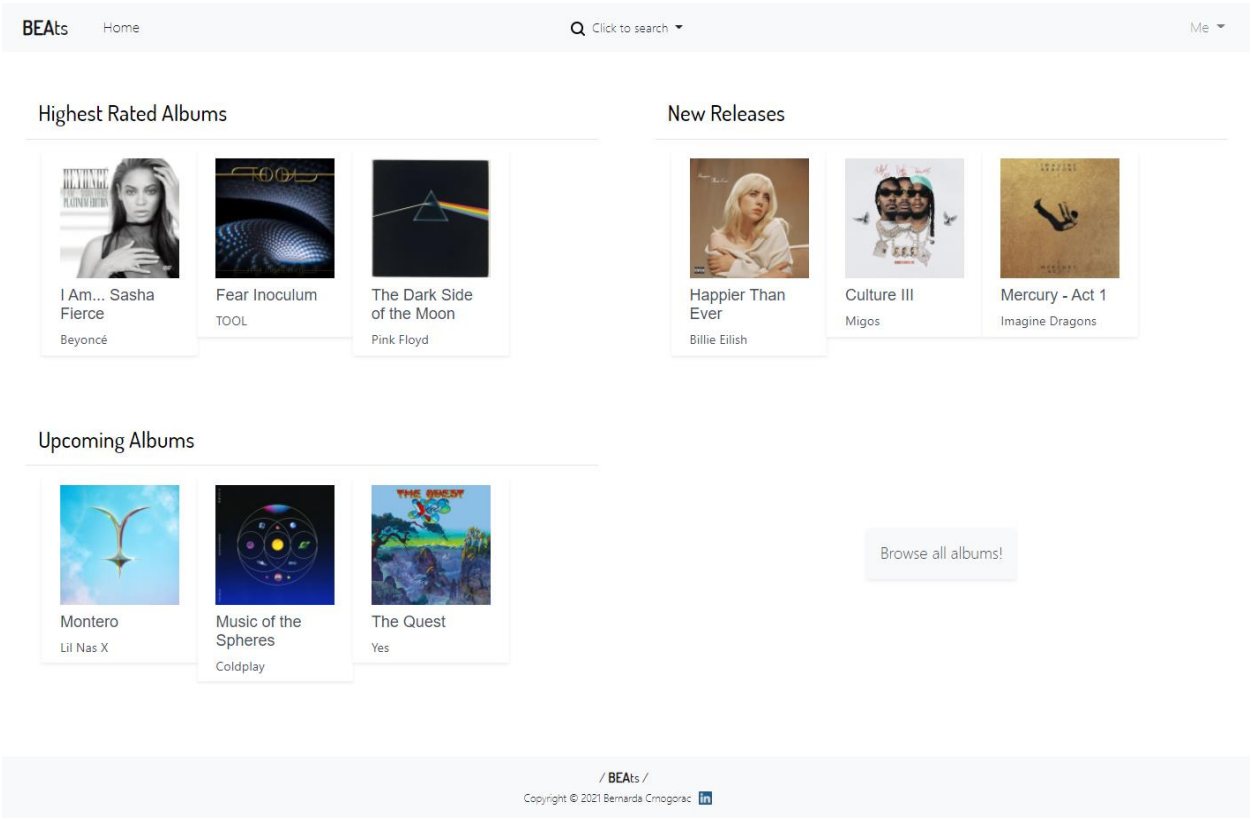
4. KORIŠTENJE WEB APLIKACIJE I ANALIZA PROGRAMSKOG RJEŠENJA

U ovome poglavlju prikazano je korištenje web aplikacije te sve moguće značajke prikazane planom izrade aplikacije. Ovisno o tome je li prijavljen admin, prikazana je i dodatna skrivena značajka, dodavanje novog albuma.

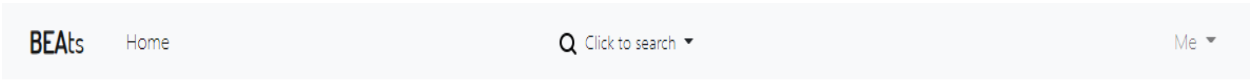
Pri otvaranju aplikacije s nazivom *BEAts*, učitava se *Auth* komponenta za prijavu ili registraciju korisnika u aplikaciju (slika 4.1.). Nakon uspješne prijave, odnosno registracije, učitava se *home* komponenta (slika 4.2.). *Header* (slika 4.3.) i *footer* (slika 4.4.) komponente su stalno prisutne, neovisno na kojoj se ruti korisnik trenutno nalazi.



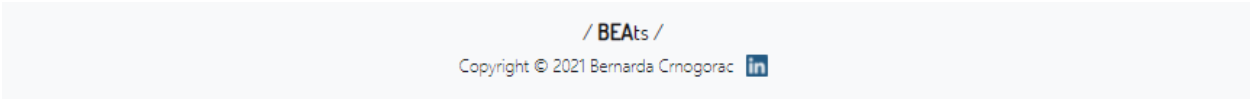
Slika 4.1. *Auth komponenta i prijava/registracija korisnika*



Slika 4.2. *Home komponenta i izgled početne stranice*

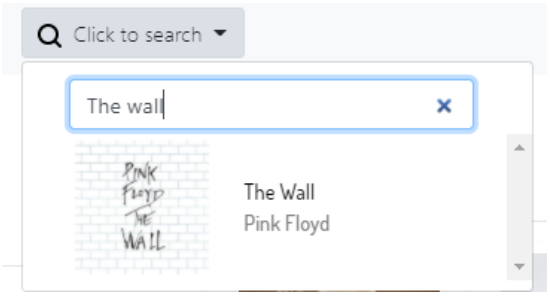


Slika 4.3. *Header komponenta*

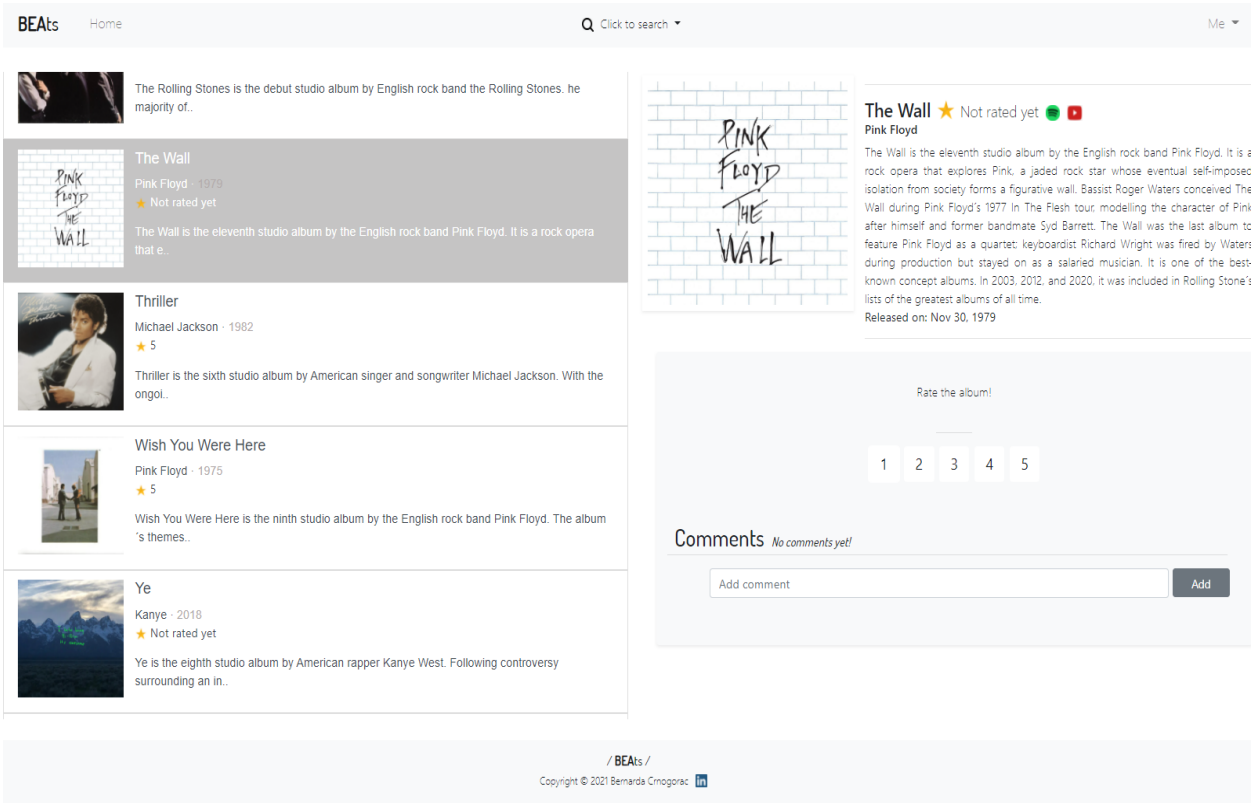


Slika 4.4. *Footer komponenta*

Search-bar komponenta u zaglavlju (slika 4.5.) omogućuje pretraživanje svih albuma te klikom na album, vodi na rutu detaljnog pregleda odabranog albuma (slika 4.6.)

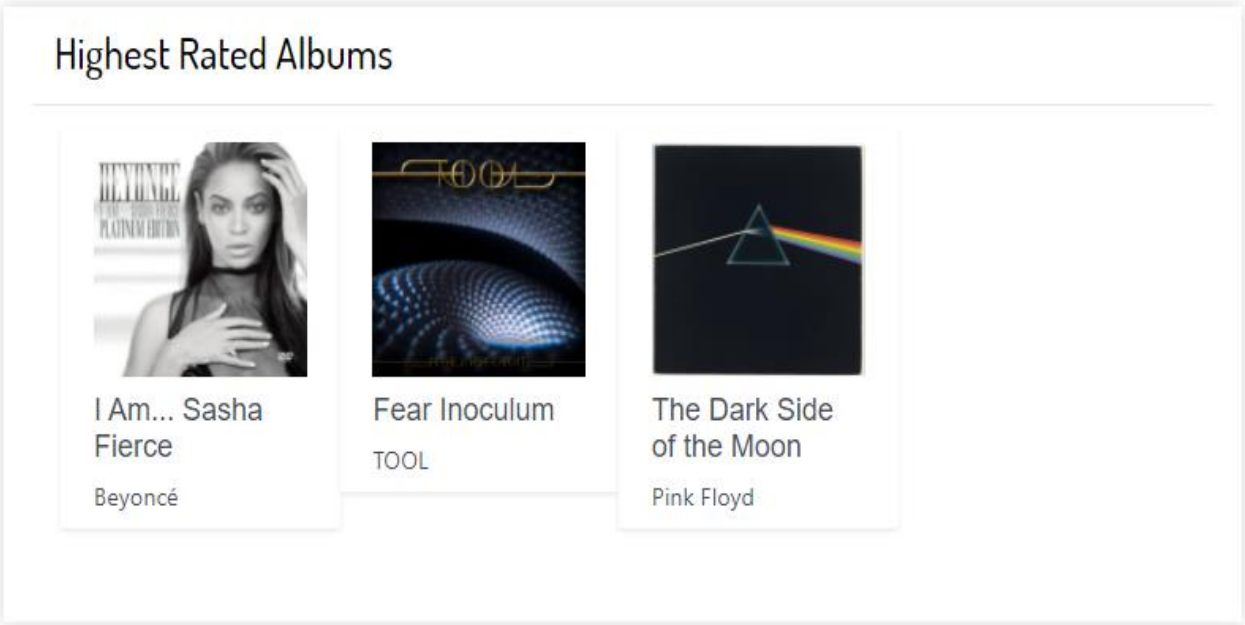


Slika 4.5. *Search-bar komponenta*

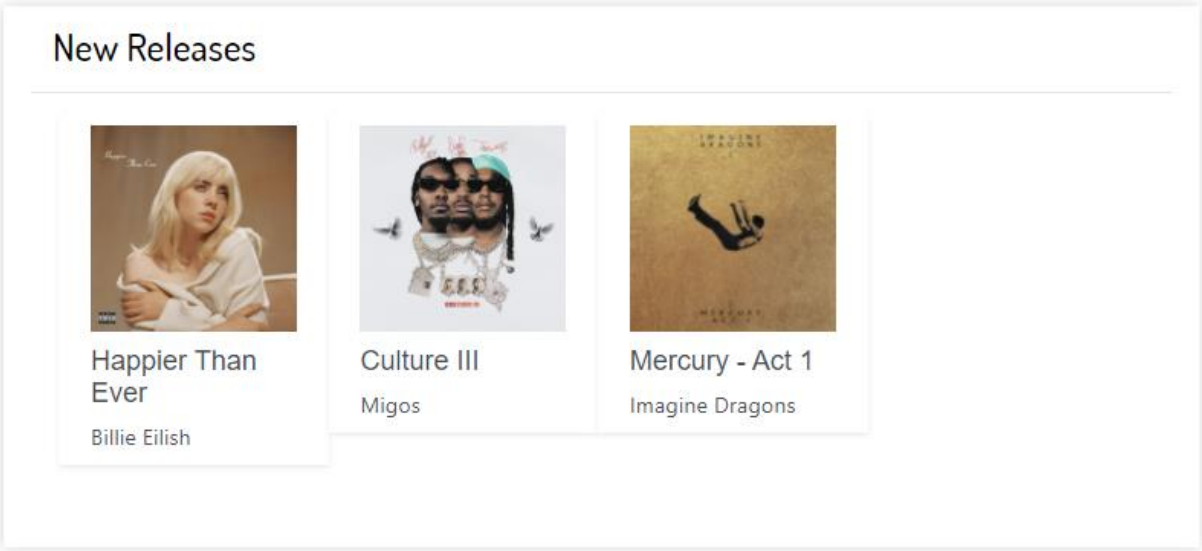


Slika 4.6. *Detaljni prikaz odabranog albuma (album-detail komponenta)*

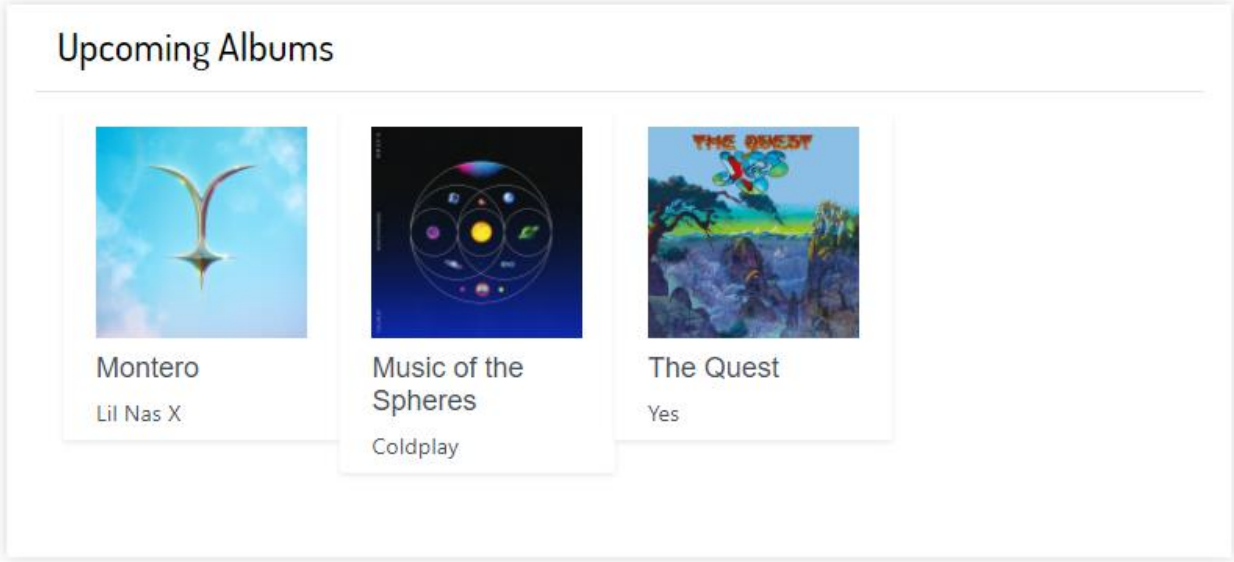
Propisano planom aplikacije (slika 3.3), na početnoj stranici se nalaze četiri značajke: *highest rated albums* (slika 4.7), *new releases* (slika 4.8.), *upcoming albums* (slika 4.9.) i *all albums* (slika 4.10.).



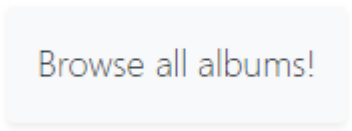
Slika 4.7. *Highest rated album značajka*



Slika 4.8. *New relaeases značajka*



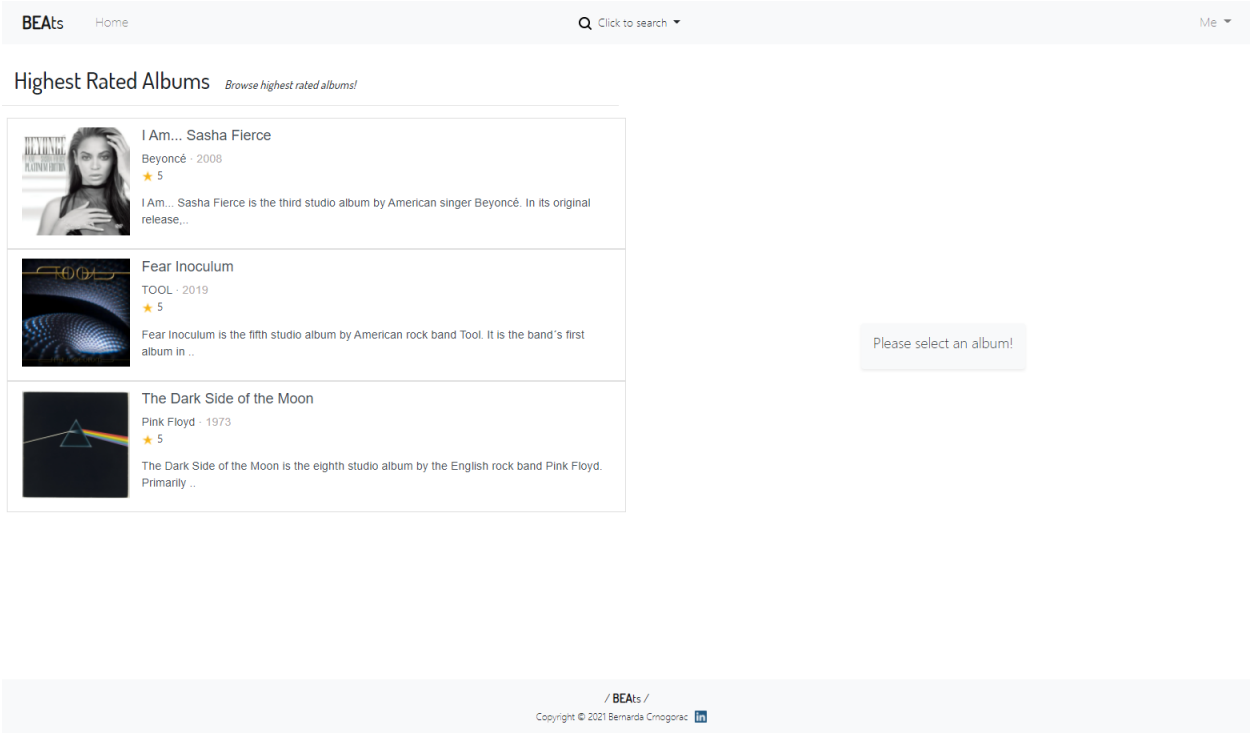
Slika 4.9. *Upcoming albums značajka*



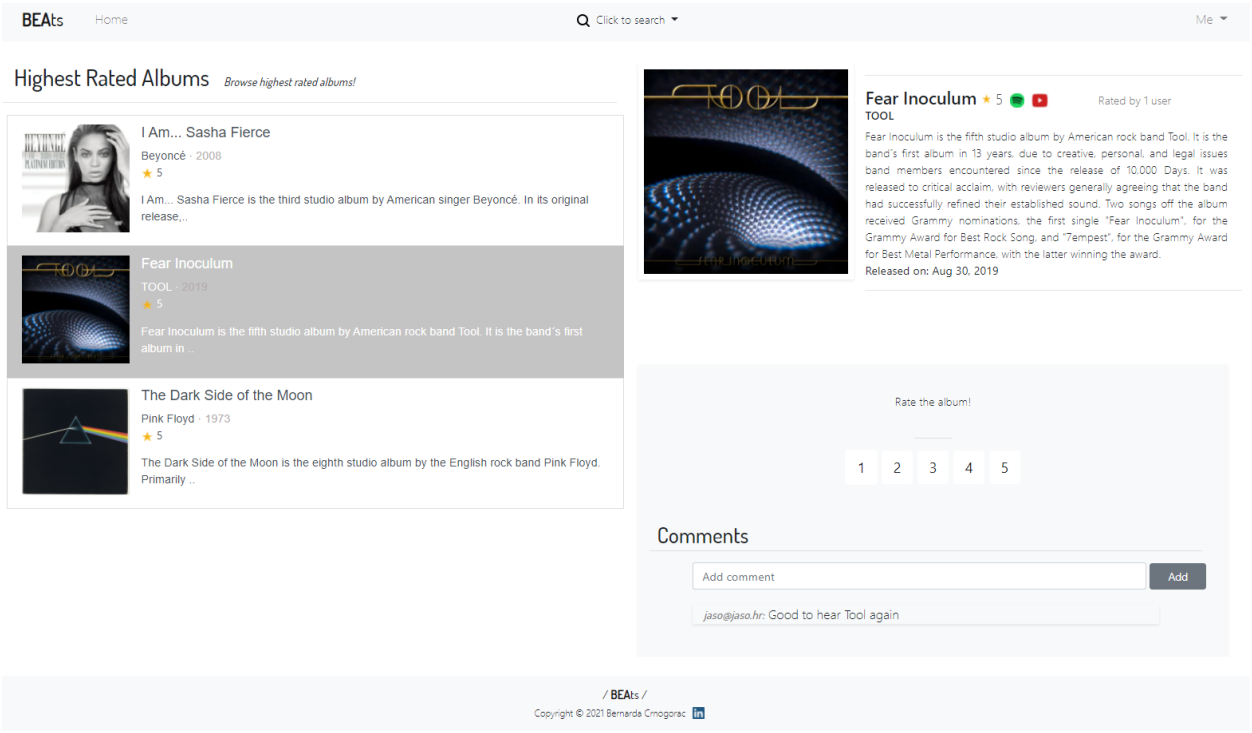
Slika 4.10. *All albums značajka*

Klikom na *highest rated* značajku, otvara se nova ruta */highest-rated-albums* (slika 4.11.). Kako nijedan album nije odabran, prvo se učitava *albums-start* komponenta koja korisnika navodi da odabere album. Nakon odabira albuma, učitava se *album-detail-highest* komponenta koja nudi

detaljni prikaz odabranog albuma s učitanim komentarima (slika 4.12.). Klikom na ikonice Spotifyja i Youtube-a, korisnik može poslušati album na spomenutim glazbenim servisima. Ostale tri značajke funkcioniraju na isti način.

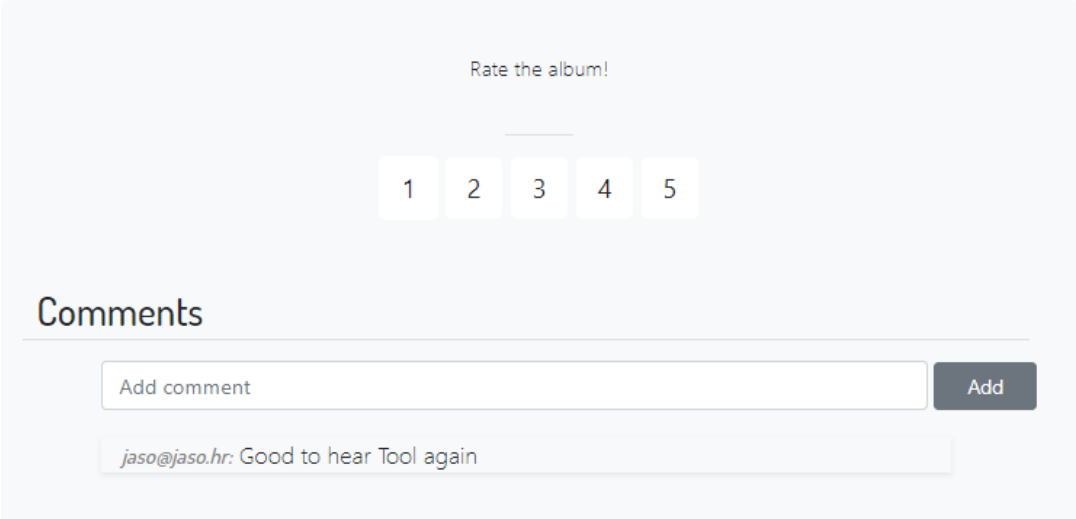


Slika 4.11. *Highest rated komponenta*

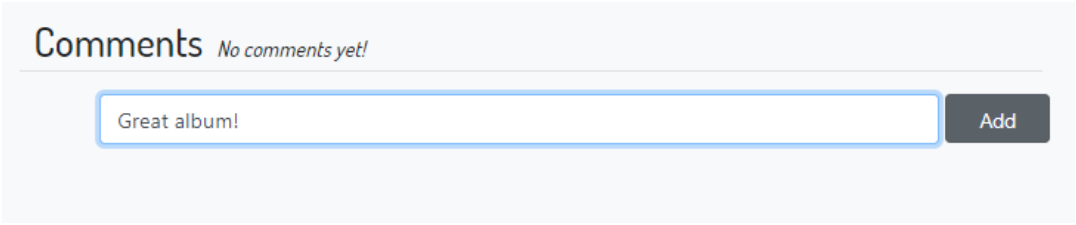


Slika 4.12. *Album-detail-highest komponenta*

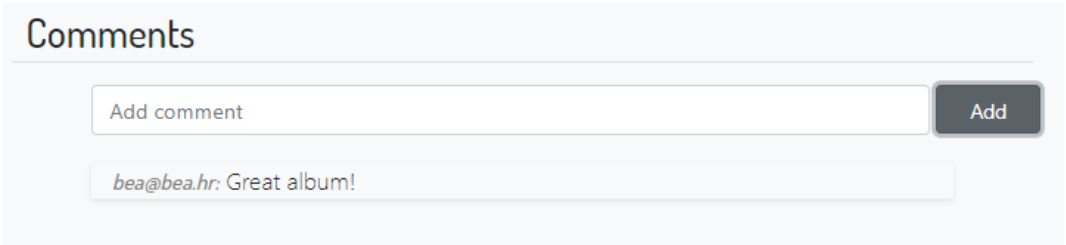
Korisnik koji nije ocijenio album ima pristup gumbima za ocjenjivanje (slika 4.13.). Također, komentari su vidljivi u svakome trenu (naravno, ako je korisnik prijavljen) te može po volji ostaviti komentar (slika 4.14. i slika 4.15.).



Slika 4.13. *Ocjenjivanje albuma i komentari*

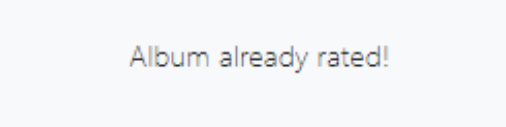


Slika 4.14. *Ostavljanje komentara – 1.dio*



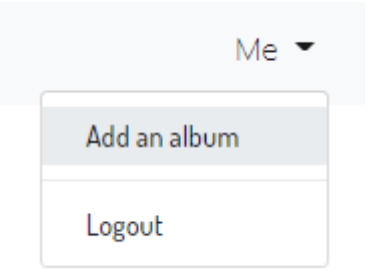
Slika 4.15. *Ostavljanje komentara – 2.dio*

Nakon ostavljanja ocjene, korisnik više nema mogućnost ocjenjivanja te se gumbi zamijene s porukom da je album već ocijenjen (slika 4.16.).



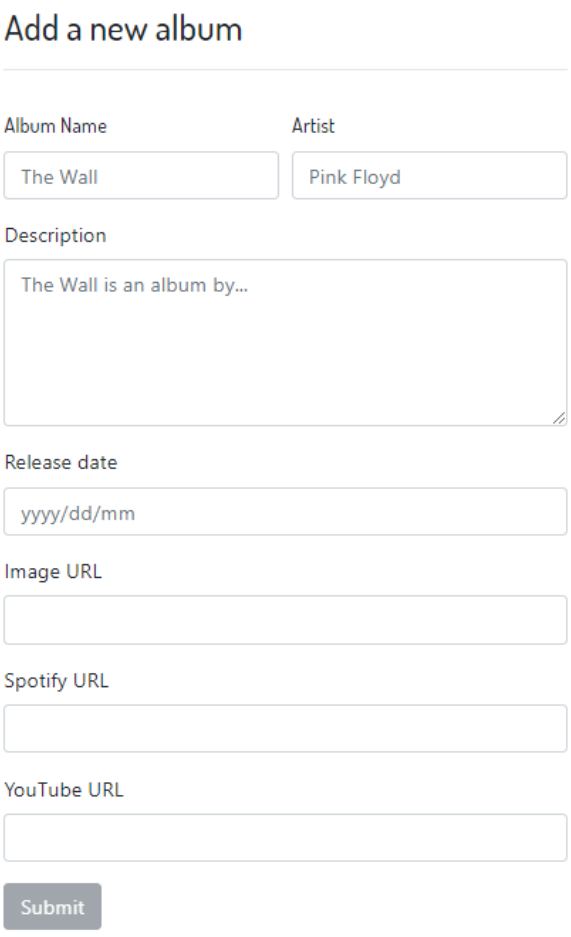
Slika 4.16. *Poruka nakon ocjenjivanja albuma*

U slučaju da je admin prijavljen, ima mogućnost dodavanja novog albuma klikom na gumb *Add an album* (slika 4.17.).



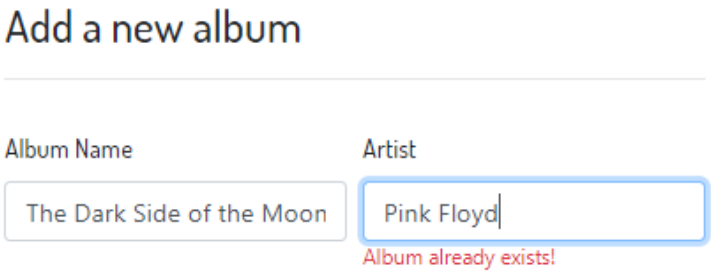
Slika 4.17. *Gumb za pristup dodavanju albuma*

Iduće, otvara se *add-album* komponenta zadužena za dodavanje novog albuma (slika 4.18.). Gumb *Submit* je onemogućen u slučaju da admin nije unio sve potrene podatke: ime albuma, ime pjevača, opis albuma, album izdanja i poveznicu na sliku albuma. Poveznice na Spotify i YouTube nisu potrebne u slučaju da se unosi album koji još nije izašao.



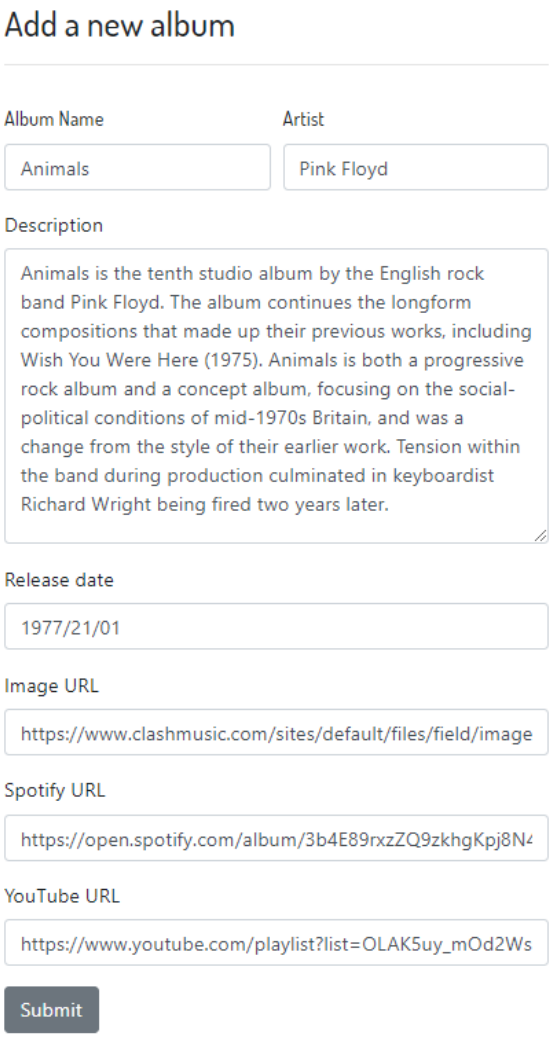
Slika 4.18. *Dodavanje novog albuma (add-album komponenta)*

Ako admin unese album koji već postoji, već kod imena pjevača javlja se greška, album već postoji (slika 4.19.).



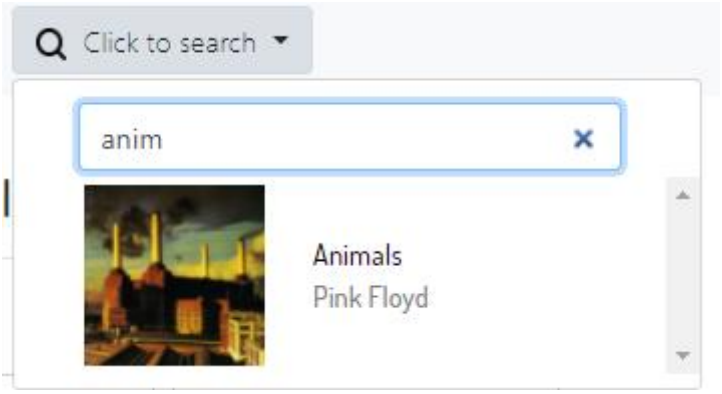
Slika 4.19. *Greška pri dodavanju novog albuma – album postoji*

Tek nakon unosa albuma koji ne postoji i unosa već prije spomenutih podataka, album se može unijeti (slika 4.20.).

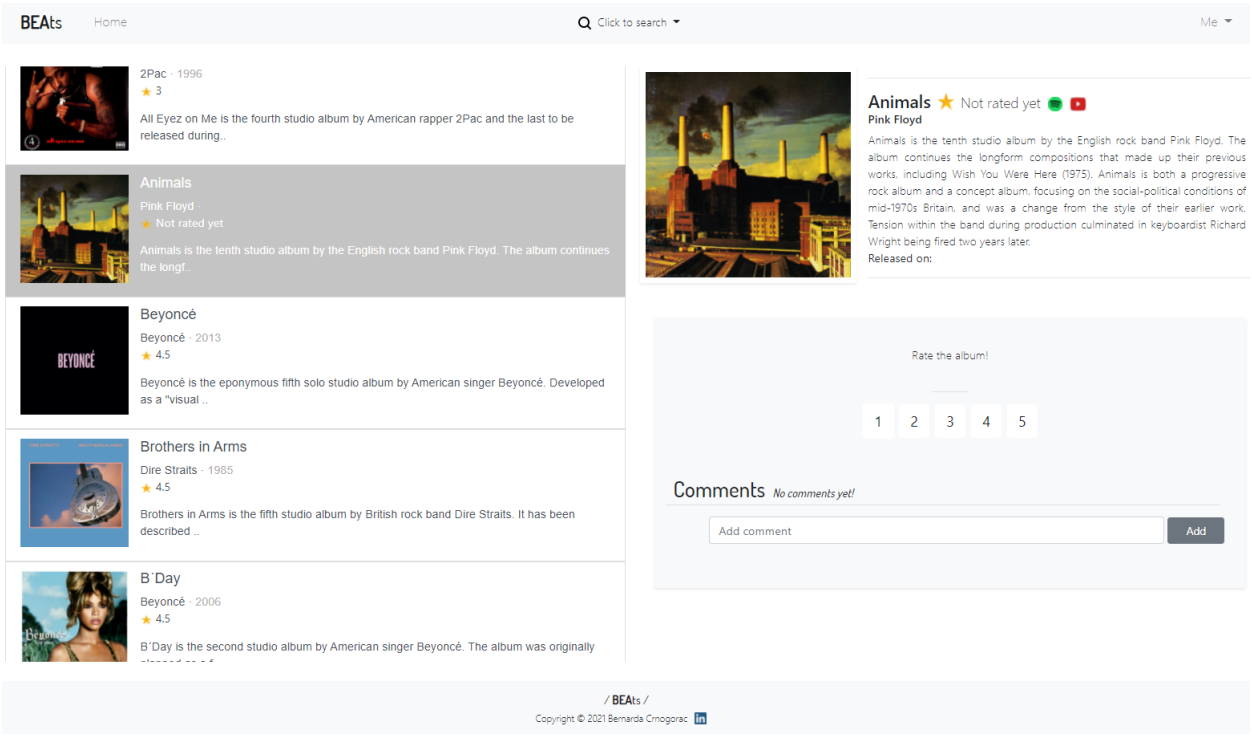


Slika 4.20. *Unos novog albuma – popunjen obrazac*

Odmah nakon dodavanja, album postaje dostupan putem pretraživanja (slika 4.21.) i preko detaljnog prikaza (slika 4.22.).



Slika 4.19. *Pretraživanje novog albuma*



Slika 4.19. *Detaljni prikaz novog albuma*

5. ZAKLJUČAK

Unatoč brojnosti društvenih mreža, ne postoji društvena mreža točno za recenziju glazbe od strane korisnika te izmjena dojmova. Ovaj završni rad pruža takvu društvenu mrežu s mogućnosti ostavljanja ocjena na određene albume, ostavljanja komentara na iste i vidljivost komentara ostalih korisnika. Korisnik također ima pristup posebnim listama albuma, albumi najviše ocijenjeni od strane ostalih korisnika, nova izdanja te albumi koji će uskoro izaći. Shodno tome, postoji mogućnost dodavanja novih albume od strane admina. Rješenje je ostvareno u Angular framework-u, koristeći HTML, Bootstrap i TypeScript (odnosno JavaScript). Front-end framework Angular nudi razne mogućnosti, poput stvaranje komponenata te dinamičko povezivanje istih, koje jednostavno nisu moguće s običnim tehnologijama HTML-a i CSS-a. Korištenje web aplikacije je vrlo jednostavno i maksimalno prilagođeno korisniku, da se korisnik može fokusirati na zašto koristi web aplikaciju, recenziju glazbe.

LITERATURA

[1] Youtube.com, https://www.youtube.com/ [stranica posjećena 11.07.2021.]

[2] Pitchfork.com, https://pitchfork.com/ [stranica posjećena 11.07.2021.]

[3] Genius.com, https://genius.com/ [stranica posjećena 11.07.2021.]

[4] IMDb.com, https://www.imdb.com/ [stranica posjećena 11.07.2021.]

[5] What is TypeScript?, https://www.typescriptlang.org/ [stranica posjećena 11.07.2021.]

[6] J. Wilken, **Angular in Action**, Manning Publications, March 2018

[7] What is Angular?, https://angular.io/guide/what-is-angular [stranica posjećena 11.07.2021.]

[8] G. Richards, S. Lebresne, B. Burg, J. Vitek, **An Analysis of the Dynamic Behaviour of JavaScript Programs,** S3 Lab, Department of Computer Science, Purdue University, West Lafayette, IN, June 2010

[9] HTML: HyperText Markup Language, https://developer.mozilla.org/en-US/docs/Web/HTML [stranica posjećena 12.07.2021.]

[10] J. Delaney, **Angular Firebase survival guide**, Lean Publishing, July 2017

[11] Firebase Realtime Database, https://firebase.google.com/docs/database [stranica posjećena 12.09.2021.]

[12] Firebase Hosting, https://firebase.google.com/docs/hosting [stranica posjećena 12.09.2021.]

[13] Firebase Authentification, https://firebase.google.com/docs/auth [stranica posjećena 12.09.2021.]

SAŽETAK

Cilj ovog završnog rada je kreiranje web aplikacije za recenziju glazbe koja nudi korisnicima posebne liste albuma, kao i anonimno ostavljanje ocjene i komentara. Korištenjem Angular framework-a, kreirane su komponente opisane planom izrade aplikacije. Također, kreirane su sve dodatke klase i datoteke potrebne za rad i povezivanje napravljenih komponenti. Nadalje, kreirano je sučelje u kojemu su smještene sve komponente i napisana logika ponašanja web aplikacije. Korištenjem Bootstrap paketa, dodan je dizajn stranici bez prevelike muke, a s odličnim rezultatima, poput responzivnosti na *desktop* verzijama. Spremanje svih albuma te ovjera autentičnosti korisnika su odrađeni preko Google Firebase alata: *Realtime Database* i *Authentification.*

Ključne riječi: Angular, Authentification, Bootstrap, Google Firebase, Realtime Database

ABSTRACT

Social network for music enthusiasts

Main purpose of the final paper is creating a web application for reviewing music which offers users special album lists, as well as anonimous music reviews and commenting. Using the Angular framework, components were created based on the application development plan. Also, all the necessary classes and files were created that are needed for components to work as intended. Further, an user interface was provided where previously made components have been placed and all the logic for the web application was given. Using the Bootstrap package, design was added to the web application without much hard effort, but with great outcome, such as desktop versions responsiveness. Storing all albums and authentification was enabled through Google Firebase tools: *Realtime Database* i *Authentification.*

Keywords: Angular, Authentification, Bootstrap, Google Firebase, Realtime Database

ŽIVOTOPIS

Bernarda Crnogorac rođena je 11. travnja 1999. godine u Slavonskome Brodu. Osnovnu školu je pohađala u rodnom gradu te također tamo upisuje prirodoslovno-matematičku gimnaziju Matija Mesić 2014. godine. Gimnaziju završava 2018. godine te iste godine upisuje preddiplomski smjer računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku.

_____

Bernarda Crnogorac

PRILOZI

CD:

- Elektronička verzija rada (dokument u .docx i .pdf formatu)
- Angular projekt