IZRADA APLIKACIJE ZA OCJENJIVANJE GLAZBENIH SPOTOVA

Ćubić, Marko

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Preddiplomski stručni studij Informacijska tehnologija

MARKO ĆUBIĆ

Z A V R Š N I R A D

IZRADA APLIKACIJE ZA OCJENJIVANJE GLAZBENIH SPOTOVA

Split, rujan 2022.

SVEUČILIŠTE U SPLITU
SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Preddiplomski stručni studij Informacijske tehnologije

Predmet: Baze podataka

Z A V R Š N I R A D

Kandidat: Marko Ćubić

Naslov rada: Aplikacija za ocjenjivanje glazbenih spotova

Mentor: mr. sc. Ivica Ružić, viši predavač

Split, rujan 2022.

Sadržaj

Sažetak

Temeljna svrha ovog završnog rada je stvoriti *web* aplikaciju "MusicVideos.com" koja će ljudima omogućiti jednostavan i brz pregled osnovnih informacija glazbenih videa. Osim standardnih informacija o videu, korisnici će uz pomoć korisničkih osvrta i ocjena, moći istražiti i generalno mišljenje ljudi o pojedinom glazbenom videu.

Web aplikacija izrađena je uz pomoć Python i Javascript programskih jezika. MySQL sistem korišten je za spajanje na bazu podataka. Dodatne biblioteke omogućili su React i Django.
Uređivač koji je korišten za rad s Reactom i Djangom je Visual Studio Code, dok je MySQL Workbench korišten za pregled i manipulaciju podataka u bazi.

Ključne riječi: *React, Django, MySQL, web aplikacija, glazbeni video*

Summary

Music video rating and reviewing application

Main purpose of this final paper is creating a web application "MusicVideos.com" that will enable a quick and simple overview of basic information of a music video.
Because of user reviews and ratings, along with the basic music video information, users will be able to see a general opinion of people on a certain music video.

The web application is created with the use of Python and Javascript programming languages. MySQL system was used for connecting to the database. Additional libraries were provided by React and Django.

Code editor used for work with React and Django is Visual Studio Code, while MySQL Workbench is used for inspection and manipulation of the database data.

Keywords: *React, Django, MySQL, web application, music video*

1. Uvod

Popularnost glazbenih videa dosegla je vrhunac kroz 80-e i 90-e, uz pomoć televizijskih programa kao što su MTV i VH1. Promocijski i umjetnički potencijal glazbenih videa postao je očit te je za svaki produkcijski studio i glazbenika postalo logično da za svoje najpopularnije pjesme snime barem jedan video. Tehnologija je imala jako velik utjecaj na popularnost i kvalitetu glazbenih videa i nastavlja imati jednak utjecaj i dan danas.

Budući da je danas popularnost kabelske televizije u padu, glazbena videa, naizgled po rastućim brojevima pregleda, dosežu novi vrhunac na internetskoj video platformi Youtube. No unatoč velikom broju pregleda na Youtubeu, poznato je da određeni broj ljudi koristi youtube kako bi samo slušali muziku, bez da provode mnogo vremena gledajući video, dok ga većina pogleda i ne razmišlja previše o njemu nakon toga.

S obzirom na rastući broj pregleda na Youtubeu i sličnim platformama i generalne rasprave na internetu i ostalim medijima, lako je zaključiti kako postoji ipak dovoljno velik broj ljudi koji stvarno cijene takvu formu umjetnosti i gaje interes za dodatnim informacijama i komunikacijom s drugim ljubiteljima glazbenih videa. Unatoč tome, trenutno ne postoji dovoljno kvalitetna *web* aplikacija za takve aktivnosti. Zaključak je da je razlog tome što postojeće aplikacije nisu dovoljno dobro dizajnirane ni promovirane. *Web* aplikacija "MusicVideos.com" omogućuje pregled informacija, ocjenjivanje, kreiranje lista, ostavljanje osvrta na glazbenim videima. Uz to korisnici s korisničkom ulogom osoblja, imaju mogućnost stvaranja novih videa i ažuriranja postojećih videa na *web* aplikaciji.

Sve to omogućeno je kroz jednostavno i intuitivno korisničko sučelje (engl. *user interface*), te pouzdanu funkcionalnost. To je postignuto tehnologijama koje su opširnije opisane u drugom poglavlju ovog završnog rada.

U trećem poglavlju opisuje se planiranje i sastav baze podataka "MusicVideos.com" *web* aplikacije. Svaki entitet baze je objašnjen i prikazan u obliku tablice.

Četvrto poglavlje prolazi kroz svaku bitnu funkcionalnost i neke od njihovih implementacija. Uz pomoć teksta i slika opisano je kako se funkcionalnost koristi, a uz to je pored nekih opisan i način implementacije uz primjere programskog kôda.

U petom poglavlju zaključuje se završni rad. Ukratko se opisuje sve što je obrađeno do tog poglavlja i nude se ideje mogućih budućih proširenja *web* aplikacije.

Posljednje poglavlje predstavlja listu s linkovima literature koja je omogućila izradu ovog završnog rada.

2. Tehnologije

Ovo poglavlje opisuje sve tehnologije korištene za izradu *web* aplikacije "MusicVideos.com".

2.1. Python

Python je objektno orijentiran programski jezik opće namjene kojeg je 1991. pod prvom 0.9.0 verzijom, stvorio nizozemski programer Guido van Rossum.

Verzije prekretnice u razvoju Pythona su Python 2.0, objavljena 2000. godine i Python 3.0 objavljena 2008. godine, kojom je napravljena značajna revizija koja nije više bila kompletno kompatibilna s prethodnim verzijama. Zadnja stabilna verzija je 3.10.6, objavljena 02.08.2022. godine[1].

Python koristi dinamične tipove (engl. *dynamic types*) i sakupljač smeća (engl. *garbage collector*) za upravljanje memorijom (engl. *memory management*), te podržava korištenje različitih programskih paradigmi (engl. *programming paradigms*) što znači da su objektno orijentirano, strukturirano i funkcionalno programiranje podržani.

Python je jedan od najpopularnijih programskih jezika i ima jednu od najvećih internetskih zajednica za jedan programski jezik. Zbog toga i zbog svoje jednostavnosti korištenja, često se preporučuje novim programerima za učenje osnova programiranja.

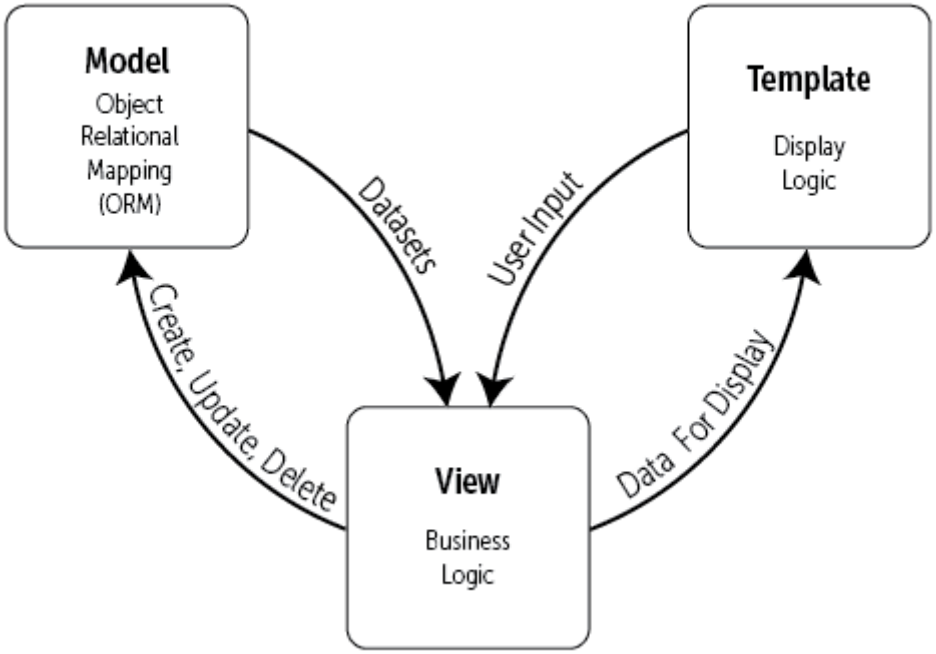
2.2. Django

Django je Python okvir (engl. *framework*) visoke razine (engl. *high-leve*l) koji prati MTV (engl. *Model Template View*) arhitekturu (Slika 1). Razvili su *web* programeri Adrian Holovaty i Simon Willison 2003. godine. Zadnja stabilna verzija Djanga je 4.1, objavljena 03.08.2022.

Primarna svrha Djanga je omogućavanje brze izrade aplikacija s fokusom na sigurnost i skalabilnost.

U MVT arhitekturi *Model* upravlja podacima i služi kao reprezentacija baze podataka. Na model se može gledati kao na tablicu u bazi podataka. *View* prihvaća i vraća HTTP (engl. *Hyper Text Transfer Protocol*) zahtjeve (engl. *request*) s preglednika (engl.

browser). Django nudi dvije opcije pisanja *viewova*: klasni (engl. *Class Based View*) i funkcijski (engl. *Function Based View*). U ovoj *web* aplikaciji koristi se *view* temeljen na klasama zbog toga što ima već ugrađene generične *viewove* temeljene na klasama. *Template* predstavlja *frontend* sloj i dinamičnu HTML (engl. *HyperText Markup Language*). On nije relevantan u ovom projektu budući da se za *frontend* koristi React.



Slika 1: *Django MTV dijagram*

Django ima alat zvan Django REST framework koji služi za izradu *web* API-ova (engl. *Application Programming Interface*)[2]. Korištenjem API poziva povezuje se i omogućuje se direktna komunikacija *frontenda* s *backendom*. Zbog korištenja REST API-a između modela i *viewa* potrebno je postaviti serijalizator (engl. *serializer*) koji omogućuje pretvaranje kompleksnih podataka kao što su *querysetovi* i instance modela u nativne Python tipove podataka (engl. *datatypes*) koji se mogu jednostavno prikazati u JSON (engl. *Javascript Object Notation*) formatu.

2.3. JavaScript

JavaScript je skriptni ili programski jezik koji omogućuje implementiranje jednostavnih ili kompleksnih značajki (engl. *features*) na *web* stranicama. Razvio ga je Brendan Eich pod tvrtkom Netscape, 1995. godine. Od tad se razvio u najpopularniji programski jezik današnjice i jednu od temeljnih tehnologija *World Wide Weba*.

JavaScript baziran je na ECMAScript standardu i razlikuje se od ostalih programskih jezika po tome što se može izvršavati na pregledniku. Koristi se za izradu različitih funkcionalnosti koje mogu uključivati ažuraciju HTML-a i CSS-a (engl. *Cascading Style Sheets*) ili izvršavanje različitih kalkulacija, manipulacija i validacija podataka.

Popularnost JavaScripta može se prepisati njegovoj jednostavnosti i intuitivnoj strukturi koja odgovara programerima početnicima. Uz to, budući da je najpopularniji programski jezik na svijetu, na internetu ima jako mnogo materijala za učenje i rješavanje različitih problema, te ima iznimno veliku internetsku zajednicu.

2.4. HTML

HTML (engl. *HyperText Markup Language*) je standardni označiteljski jezik (engl. *markup language*) koji služi za stvaranje *web* stranica. Razvio ga je Tim Berners-Lee,a prva službena verzija, HTML 2, objavljena je 1995. godine. HTML se sastoji od serija elemenata kojima se gradi struktura HTML stranice te koji govore kako browser kako da prikaže sadržaj. HTML se isto tako može dinamički generirati korištenjem JavaScripta.

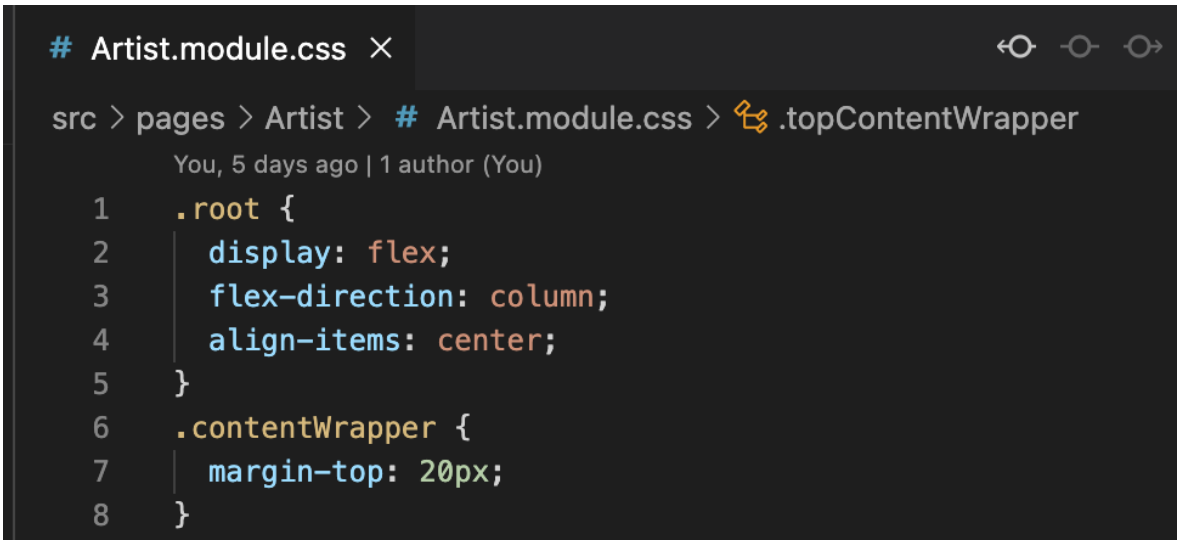
2.5. CSS

CSS (engl. *Cascading Style Sheets*) je stilski jezik (engl. *style sheet language*) koji služi za stiliziranje elemenata označiteljskog jezika kao što je HTML. CSS razdvaja strukturni sadržaj stranice od njene vizualne prezentacije.

U svojim počecima, ranih devedesetih, HTML nije imao opciju stiliziranja elemenata što ga je činilo nepreglednim za širu upotrebu. To je bila temeljna motivacija

Håkon Wium Liea i Ber Bosa da razviju svoj stilski jezik i tako poboljšaju korisničko iskustvo (engl. *user experience*) na *web* stranicama baziranim na HTML-u[3]. CSS je objavljen u prosincu 1996. godine. CSS specifikacije i dalje održava W3C (engl. *World Wide Web Consortium*).

React nudi različite načine korištenja CSS-a, kao što su *inline* stiliziranje, korištenje *CSS Stylesheet-a* i CSS modula. "MusicVideos.com" *web* aplikacija koristi CSS module koji omogućavaju dodavanje stilova u odvojenu datoteku od React komponente. CSS stilovi pojedinog modula pristupačni su samo onoj komponenti koja ih unosi (engl. *import*). Zbog toga se ne treba brinuti o konfliktima imena stilova.



Slika 2: *Primjer .module.css datoteke*

2.6. React

React je besplatna i *open-source frontend* JavaScript biblioteka koja omogućuje korisniku da brzo i efikasno stvara korisnička sučelja bazirana na komponentama za višekratnu uporabu.

React je stvorio *software* inženjer Jordan Walke 2011. godine za Facebookove potrebe (današnja Meta). Meta ga i dan danas održava. React je zbog svoje jednostavnosti i kvalitete u ovih nekoliko godina prerastao u najpopularniji *web* okvir/biblioteku s velikom internet zajednicom.

React se najčešće koristi za izradu jednostraničnih (engl. *single-page*) *web* stranica i *web* ekstenzija, a uz pomoć React Native okvira, izvedenog iz Reacta, moguće je razvijati mobilne aplikacije.

React se fokusira na upravljanje stanjem (engl. *state management*) i učitavanje stanja na DOM (engl. *Document Object Model*).

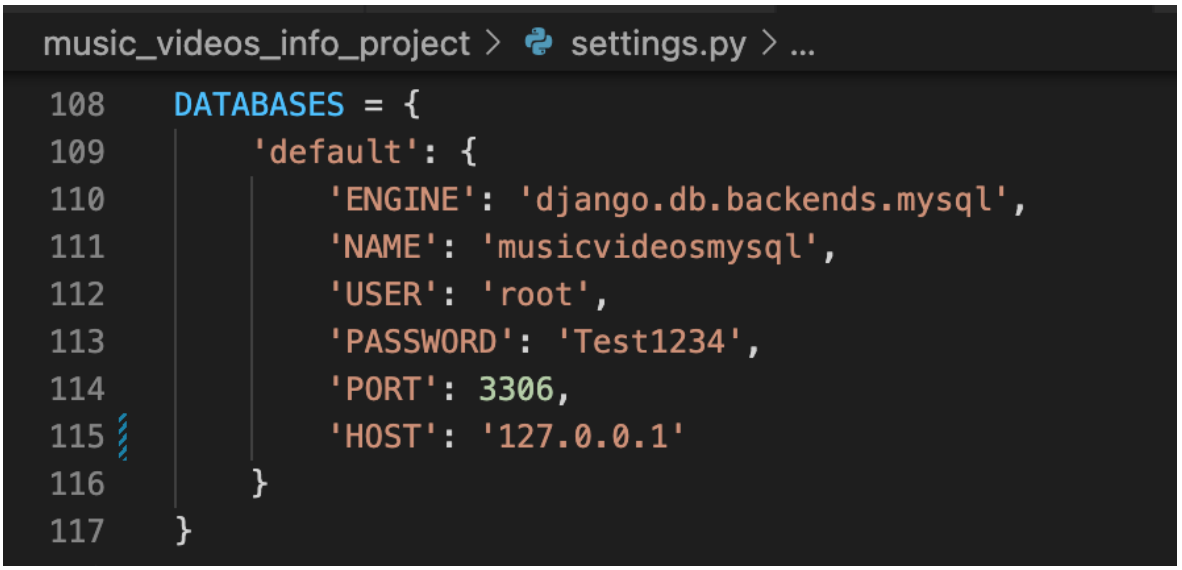
2.7. MySQL

MySQL je besplatan sustav za relacijskim upravljanjem bazom podataka (engl. *database management system*) otvorenog kôda (engl. *open source*). Relacijsko upravljanje bazom podataka omogućuje organiziranje podataka u tablice baze podataka u kojima podaci mogu biti u relaciji jedni s drugima.

Postoje tri tipa relacija između dviju tablica:

- jedan-na-jedan (engl. *one-to-one*)
- jedan prema više (engl. *one-to-many*)
- više-na-više (engl. *many-to-many*)

Povezivanje MySQL-a s Djangom postignuto je konfiguracijom "DATABASES" objekta u settings.py datoteci Django projekta (Slika 3).



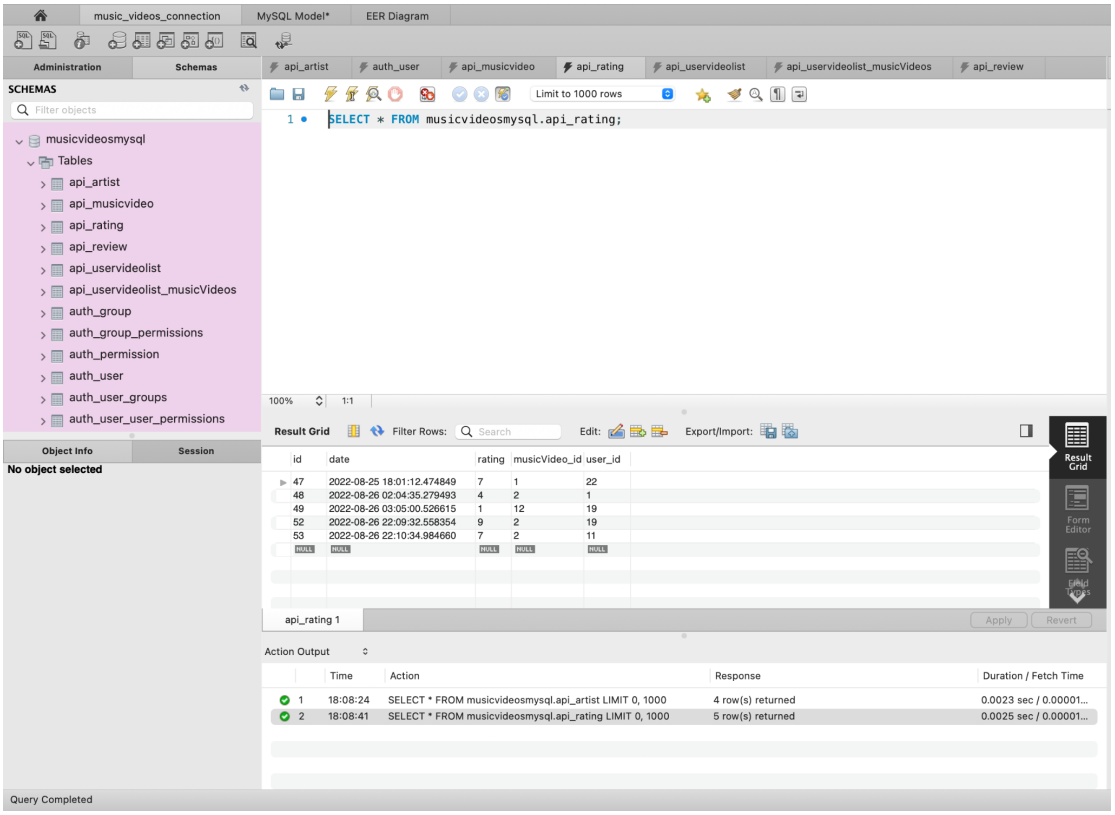
Slika 3: *Konfiguracija baze podataka u settings.pys*

2.8. Vizualni alati

Vizualni alat je bilo koji program, uslužnost (engl. *utility*), rutina ili funkcija koja omogućuje korisniku vizualnu opciju da riješi neki problem koji bi inače morao rješavati pisanjem kôda ili nekim drugim kompleksnijim radnjama. Vizualni alati danas su postali jako korisni u gotovo svakoj grafički baziranoj (Windows, Mac itd.) aplikaciji.

2.8.1. MySQL Workbench

MySQL Workbench (Slika 4) je vizualni alat za dizajn baze podataka koja ima integriran SQL razvoj, administraciju, stvaranje i kontrolu integriranog razvojnog okruženja MySQL sistema baze podataka. Jednostavan je za korištenje i pogotovo ubrzava izvršavanje jednostavnih upita (engl. *query*), kao što su brisanje, stvaranje i ažuriranje polja, vrijednosti tablica. MySQL Workbench omogućuje i pregled EER (engl. *Enhanced Entity-Relationship*) dijagrama koji omogućuju vizualnu prezentaciju odnosa između tablica određene baze podataka.

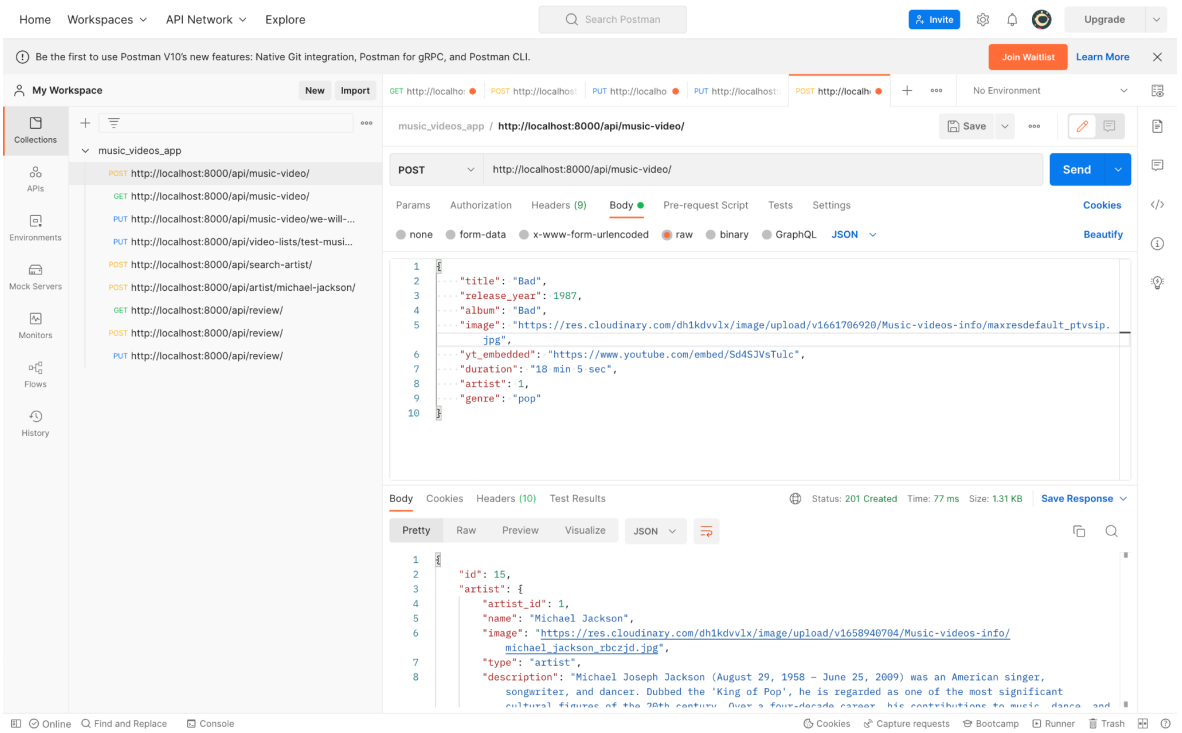


Slika 4: *Prikaz MySQL Workbench sučelja*

2.8.2. Postman

Postman je API platforma za izgradnju i korištenje API-ova. Postmanov glavni zadatak je da korisniku pojednostavni stvaranje, dijeljenje, testiranje i dokumentiranje API-ova [4].

Postman je 2012 započeo kao projekt programerskog inženjera Abhinava Asthana, koji je želio pojednostaviti API testiranje. Budući da je na tržištu bila velika potreba za takvim alatom, do danas je skupio više od 20 milijuna korisnika. Takvom brzom rastu doprinijela je i jednostavnost korištenja alata te intuitivno korisničko sučelje (Slika 5). Postman od korisnika traži samo da doda vrstu zahtjeva (GET, POST, PUT) te *endpoint* putanju i po potrebi ključ-vrijednost (engl. *key-value*) ili potrebne podatke. Uz to ima još dodatne opcije prikaza podataka. U konzoli na dnu prikazane su informacije o uspješnosti API zahtjeva. Postman omogućuje i spremanje zahtjeva API podataka koji se kasnije mogu ponovo koristiti.



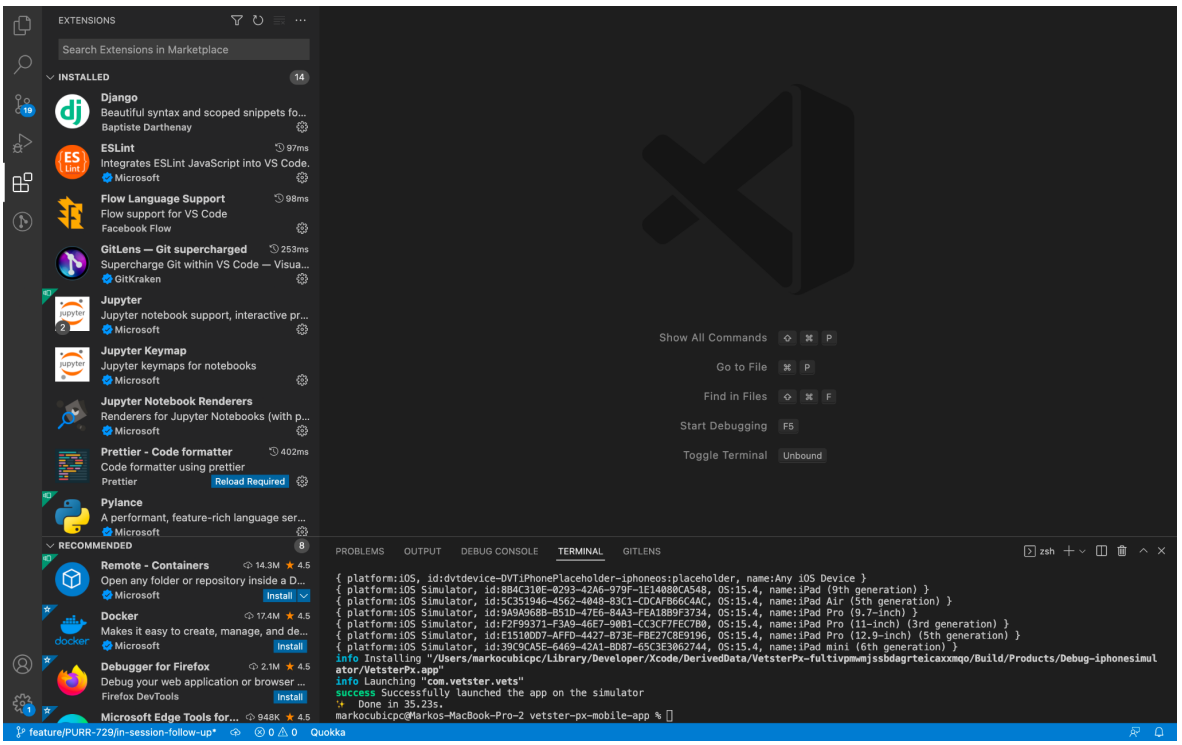
Slika 5: *Prikaz Postman sučelja*

2.9. Visual Studio Code

Visual Studio Code je uređivač kôda lake kategorije (engl. lightweight). Dolazi s ugrađenom podrškom JavaScripta, TypeScripta i Node.jsa. Uz to ima i bogat ekosistem ekstenzija koje mogu korisniku optimizirati i olakšati proces programiranja. Instalirane ekstenzije mogu se pronaći na izborniku na lijevoj strani sučelja (Slika 6). Podržava debagiranje, *syntax highlighting*, *intelligent code completion*, refaktoriranje kôda, *snippete* i ugrađen Git. VSC (engl. Visual Studio Code) podržava i uređivanje i prilagođavanje izgleda sučelja i raznih kratica.

Razvio ga je Microsoft 2015. godine u svrhu korištenja na Windowsu, Linuxu i macOSu. Danas je jedan od najkorištenijih uređivača kôda na svijetu. Po anketi Stack Overflowa iz 2021., VSC je rangiran kao najpopularnije programsko razvojno okruženje, gdje je 70% od 82,000 ispitanika potvrdilo da ga koriste[5].

Za izgradnju ovog projekta korištene su sljedeće VSC ekstenzije: Django, Python, Simple React Snippets, Prettier, GitLens.



Slika 6: *Prikaz Visual Studio Code sučelja*

3. Baza podataka

3.1. Planiranje

Budući da *web* aplikacija "MusicVideos.com" mora imati implementiran niz značajki, koje trebaju funkcionirati uz pomoć API poziva, potrebno je pomno i pažljivo isplanirati bazu podataka.

Nakon što se razvije jasan plan i svrha *web* aplikacije odlučeno je koji su sve API *endpointovi* potrebni da bi se postigli određeni zahtjevi plana. Nakon toga se izrađuje plan modela koji predstavljaju tablice i njihova polja. Uz to je potrebno odrediti primarne i strane ključeve i odnose između tih tablica.

Potrebni *endpointovi*:

- Prijava i odjava korisnika
- Stvaranje korisničkog računa
- Dohvaćanje, stvaranje i ažuriranje glazbenih videa.
- Dohvaćanje umjetnika/benda
- Pretraživanje benda ili umjetnika
- Dohvaćanje, stvaranje, ažuriranje i brisanje listi glazbenih videa
- Dohvaćanje, stvaranje, ažuriranje i brisanje ocjena glazbenih videa
- Računanje i brojanje ocjena glazbenog videa
- Dohvaćanje korisnikovih ocijenjenih glazbenih videa
- Dohvaćanje, stvaranje i ažuriranje korisničkih osvrta

Nakon toga u odabranom uređivaču kôda, u ovom slučaju Visual Studio Code, mogu se početi stvarati potrebni API *endpointovi* tako što se uspostavi komunikacija između klasa datoteka "models.py". "serializers.py" i "views.py". Uz to je još potrebno napisati rute *endpointova* u "urls.py" datoteci.

Testiranje tablica i endpointova odrađuje se u MySQL Workbenchu i Postmanu.

3.2. Modeli

3.2.1. Korisnik

Korisnička tablica generirana je automatski pri prvoj migraciji na bazu podataka pod imenom “auth_user” (Tablica 1). Budući da zadani User model sadrži sva potrebna polja za ovu *web* aplikaciju, nije bilo potrebno preklopiti (engl. *override*) postojeći User model s prilagođenim (engl. *custom*) User modelom.

Za prijavu i odjavu korisnika potrebni su JWT (engl. *JSON Web Token*) tokeni koji se generiraju pomoću Simple JWT dodatka (engl. *plugin*) za Django REST okvir. Potrebno je kreirati vlastiti serijalizator koji će naslijediti “TokenObtainPairSerializer” iz Simple JWT dodatka. Novonastali serijalizator poziva se u *view* za prijavu. Kad god se korisnik uspješno prijavi imat će generiran token za pristup (engl. *access token*) i token za osvježavanje pristupnog tokena (engl. *refresh token*). Tokenu za pristup dodani su *claimovi* koji se mogu dekodirati i tako koristiti na frontend dijelu *web* aplikacije.

Tablica 1: *Entitet “auth_user”*

auth_user				
Naziv	Tip	Veličina	Primarni ključ	Kardinalnost
id	Broj	32 bits	Da	(1,1)
password	Niz znakova	128		(1,1)
is_superuser	Mali broj	1 byte		(1,1)
username	Niz znakova	150		(1,1)
first_name	Niz znakova	150		(0,1)
last_name	Niz znakova	150		(0,1)
email	Niz znakova	254		(1,1)
is_staff	Mali broj	1 byte		(1,1)
is_active	Mali broj	1 byte		(1,1)

Za registraciju korisnika koriste se drugi serijalizator i view. U serijalizatoru se definira koja su polja neophodna (engl. *required*) i validatore na tim poljima. Uz pomoć "set_password" metode obična, tekstualna zaporka, pretvara se u hashed zaporku što je važno za njenu sigurnost.

3.2.2. Umjetnik

Umjetnika predstavlja entitet "api_artist" (Tablica 2), a umjetnik može biti osoba ili bend. Budući da se kod navigacije na frontendu koristi *slug*, kako bi program prepoznao o kojoj osobi/bendu se radi, stvaranjem nove osobe ili benda generira se jedinstveni *slug*. To je postignuto "unique_slugify" metodom paketa "django_unique_slugify".

Tablica 2: *Entitet "api_artist"*

api_artist				
Naziv	Tip	Veličina	Primarni ključ	Kardinalnost
artist_id	Broj	32 bits	Da	(1,1)
name	Niz znakova	100		(1,1)
image	Dugi niz znakova	1200		(1,1)
type	Niz znakova	10		(1,1)
description	Dugi tekst	1200		(1,1)
birth	Niz znakova	255		(1,1)
slug	Niz znakova	50		(1,1)

3.2.3. Glazbeni video

Tablica za glazbena videa "api_musicvideo" (Tablica 3) koristi "MusicVideo" model. Entitet "api_musicvideo" ima polje "artist_id" koji je strani ključ i tako povezuje "api_musicvideo" s "api_artist" tablicom, odnosno "api_artist" je u jedan prema više vezi s "api_musicvideo". Budući da su "api_artist" i "api_musicvideo" povezani, na stranici

umjetnika moguće je prikazati cijelu njegovu videografiju. Automatsko generiranje *sluga* je isto tako omogućeno.

Tablica 3: *Entitet "api_musicvideo"*

api_musicvideo				
Naziv	Tip	Veličina	Primarni ključ	Kardinalnost
id	Veliki broj	64 bits	Da	(1,1)
artist_id	Broj	32 bits		(1,1)
title	Niz znakova	255		(1,1)
slug	Niz znakova	50		(1,1)
release_year	Niz znakova	255		(1,1)
album	Niz znakova	255		(1,1)
image	Dugi niz znakova	1200		(1,1)
yt_embedded	Niz znakova	255		(1,1)
rate_score	Decimalni broj	2,1		(1,1)
votes_number	Pozitivan broj	4 bytes		(1,1)
duration	Niz znakova	255		(1,1)
genre	Niz znakova	50		(0,1)
song_description	Dugi niz znakova	1200		(1,1)

3.2.4. Korisničke liste glazbenih videa

Registrirani korisnici mogu kreirati više lista videa koje će moći ažurirati i brisati. Da bi to bilo postignuto potreban je entitet koji predstavlja listu pa je stvorena tablica "api_uservideolist" (Tablica 4). Ona je s "auth_user" tablicom povezana stranim ključem

“user_id”. Nadalje, stvorena je tablica “api_uservideolist_musicVideos” (Tablica 5) koja omogućava više-na-više vezu između api_uservideolist i api_musicvideo tablica.

Postignuto je da svaki registrirani korisnik može imati više vlastitih lista glazbenih videa za koje se zna da su njegove. Svaka od tih listi može se prikazati na “MusicVideos.com” zaslonu za liste, a razlikuju se po *slugovima* koji su automatski generirani po svom “title” polju.

Tablica 4: *Entitet “api_uservideolist”*

api_uservideolist				
Naziv	Tip	Veličina	Primarni ključ	Kardinalnost
id	Veliki broj	64 bits	Da	(1,1)
user_id	Broj	32 bits		(1,1)
title	Niz znakova	100		(1,1)
date_created	Datum/vrijeme	6		(1,1)
date_updated	Datum/vrijeme	6		(1,1)
slug	Niz znakova	50		(1,1)

Tablica 5: *Entitet “api_uservideolist”*

api_uservideolist				
Naziv	Tip	Veličina	Primarni ključ	Kardinalnost
id	Veliki broj	64 bits	Da	(1,1)
uservideolist_id	Veliki broj	64 bits		(1,1)
musicvideo_id	Veliki broj	64 bits		(1,1)

3.2.5. Ocjene

Svaki registrirani korisnik ima mogućnost ocjenjivanja glazbenih videa. Tu ocjenu može mijenjati i brisati. Korisnik isto tako može pristupiti listi ocijenjenih videa. To znači da će svaka ocjena imati svoj video i svog korisnika.

Tablica “api_rating” (Tablica 6) predstavlja ocjenu. Da bi se povezala s korisnikom i videom, tablice “api_musicvideo” i auth_user potrebno je povezati s tablicom “api_rating” s jedan prema više vezom.

Tablica 6: *Entitet “api_rating”*

api_rating				
Naziv	Tip	Veličina	Primarni ključ	Kardinalnost
id	Veliki broj	64 bits	Da	(1,1)
music_video_id	Broj	32 bits		(1,1)
user_id	Broj	32 bits		(1,1)
rating	Broj	32 bits		(1,1)
date	Datum/vrijeme	6		(1,1)

3.2.6. Osvrti

Svaki registrirani korisnik ima mogućnost ostavljanja osvrta na stranici glazbenog videa. Korisnik isto tako ima mogućnost uređivanja svog osvrta i pregledavanja tuđih.

Kao kod slučaja “api_rating” entiteta, “api_musicvideo” i “auth_user” su u jedan prema više vezi s “api_review” koja je ostvarena kroz atribute “musicVideo_id” i “user_id” koji su primarni ključevi “api_review” tablice (Tablica 7).

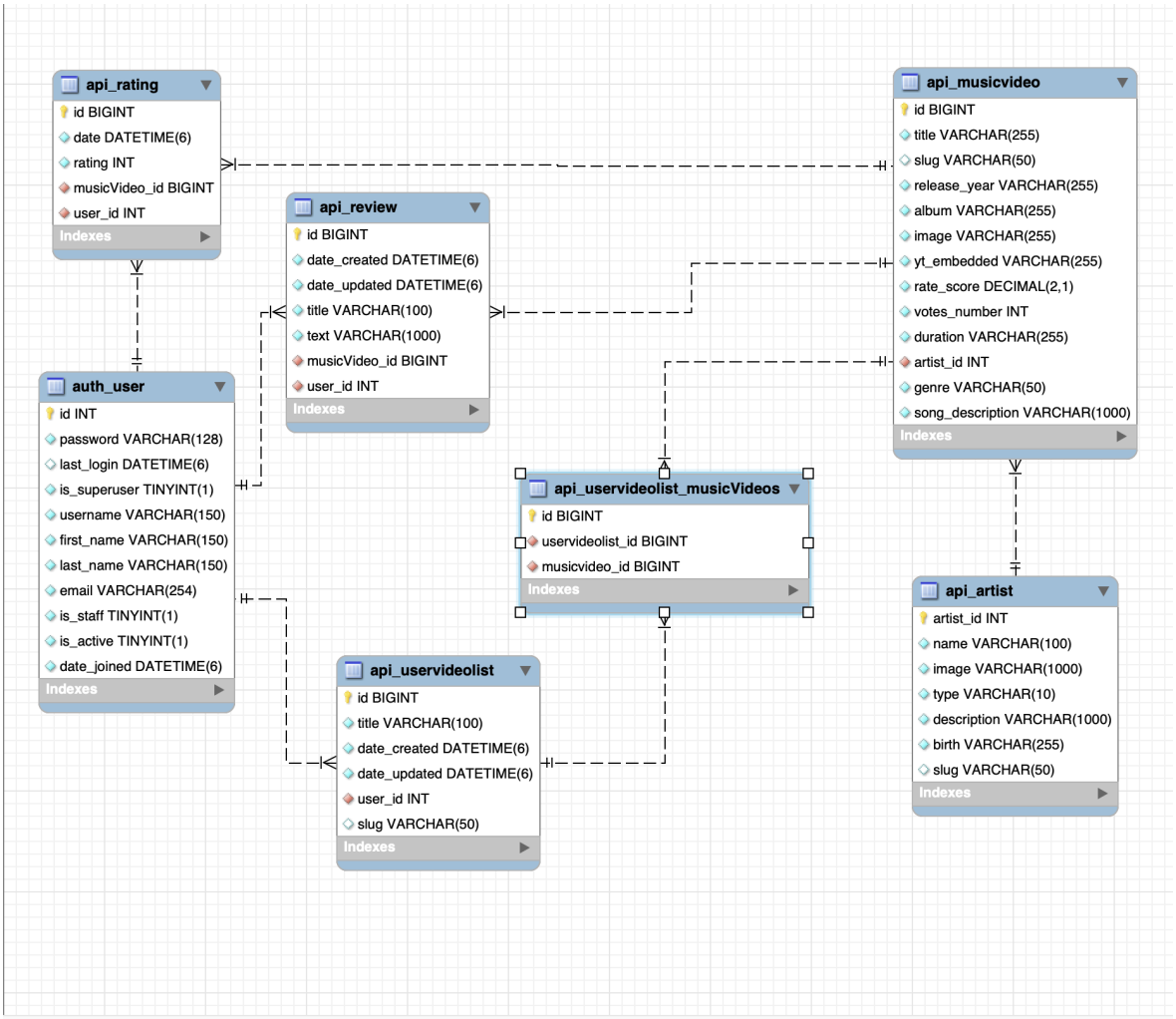
Tablica 7: *Entitet “api_review”*

api_review				
Naziv	Tip	Veličina	Primarni ključ	Kardinalnost
id	Veliki broj	64 bits	Da	(1,1)
musicVideo_id	Broj	32 bits		(1,1)
user_id	Broj	32 bits		(1,1)
title	Niz znakova	100		(1,1)
text	Dugi niz znakova	1200		(1,1)
date_created	Datum/vrijeme	6		(1,1)
date_updated	Datum/vrijeme	6		(1,1)

3.3. Dijagram ER

Dijagram ER (engl. *Entity Relationship Diagram*) je vizualizacija odnosa između entiteta u bazi podataka. Atributi entiteta isto tako mogu biti prikazani (Slika 7).

Dijagram ER može biti koristan inženjerima zato što može ilustrirati logičku strukturu baze podataka. Ako inženjer iz nekog razloga želi dokumentirati ili skicirati bazu podataka ili dizajnirati novu bazu podataka, onda može koristiti dijagram ER za to.



Slika 7: *Dijagram ER MusicVideos.com web aplikacije*

Dijagram ER s primjera (Slika 6) prikazuje 7 glavnih tablica "MusicVideos.com" *web* aplikacije. Ovaj dijagram generiran je uz pomoć MySQL Workbench *Reverse*

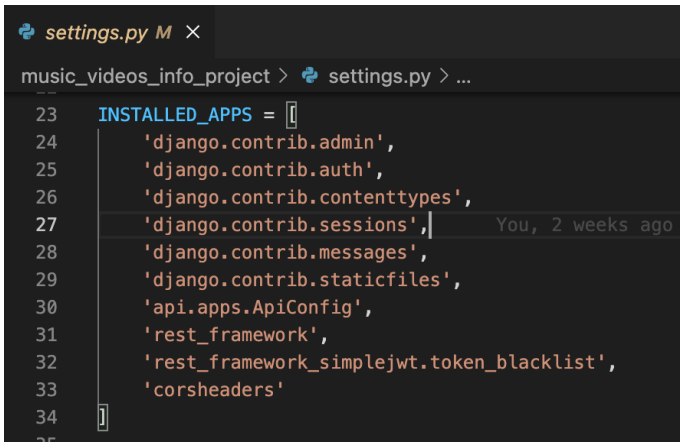
Engineer opcije. Može se primijetiti da su neke od tablica povezane isprekidanim linijama. Te linije označavaju vezu te ovisno o simbolu kojim linija dotiče tablicu može se zaključiti o kakvoj vezi se radi. Naprimjer, tablica “api_review” povezana je s tablicom “auth_user” tako da je “auth_user” u jedan prema više vezi s “api_review”. Isto tako je “api_musicvideo” u jedan prema više vezi s “api_review”. Strani ključevi u tablici “api_review” označeni su crvenim dijamantima lijevo od atributa.

Gledajući odnose između ove tri tablice moguće je povući zaključak da svaki video može imati više osvrta i da svaki korisnik može imati više osvrta, ali da korisnik može imati samo jedan osvrt za određeni video i obrnuto.

4. Funkcionalnost i implementacija

U ovom poglavlju prolazit će se kroz sve glavne funkcionalnosti i implementacije *web* aplikacije "MusicVideos.com".

Backend i frontend dijelovi projekta odvojeni su na Django projekt i React projekt. *Backend* ne ovisi o *frontendu* dok *frontend* ovisi o *enpointovima* i podacima s *backenda*. *Backend* je strukturiran tako da je stvoren projekt "api" u kojem se nalaze modeli, *viewovi*, serijalizatori, *urlovi* (engl. *Uniform Resource Locator*) itd. "Api" je dodan u instalacije glavnog Django projekta "music_videos_info_project" (Slika 8).



Slika 8: *Prikaz instalacija u "settings.py" datoteci glavnog Django projekta*

Frontend je strukturiran tako što su datoteke organizirane u više različitih direktorija ovisno o kategoriji kojoj pripadaju. Npr. sve komponente za izgradnju projekta odvojene su u direktorij "components", dok su roditeljske komponente koje prezentiraju cijelu određenu stranicu *web* aplikacije odvojene u direktorij "pages". Budući da React nema standardiziranu strukturu projekta korištene su smjernice grupiranja i izbjegavanja previše ugniježđenih direktorija s reactjs *web* stranice[6].

Korisnici ove aplikacije mogu imati tri različite uloge:

- Obični korisnik (nije registriran)
- Registrirani korisnik (registriran)
- Administrator korisnik (registriran s ulogom osoblja (engl. *staff*))

Obični korisnik može pristupiti samo početnom zaslonu, modalima za registraciju i prijavu te stranicama videa, umjetnika ili benda.

Registrirani korisnik može pristupati svemu što i obični korisnik. Uz to može pristupiti stranici za prikaz lista videa. Na toj stranici može stvarati, ažurirati i brisati vlastite liste videa. Isto tako može ući u neku od svojih lista i pregledavati je na stranici za listu. Registrirani korisnik može ocjenjivati videa, a sva videa koje je ocijenio može vidjeti na stranici za listu ocijenjenih videa.

Administrator korisnik, uz sve mogućnosti koje imaju registrirani i obični korisnik, ima mogućnost stvaranja novih videa i ažuriranja postojećih.

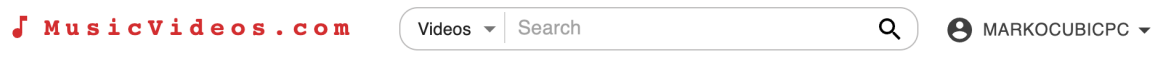
Svaka od navedenih funkcionalnosti opisana je u sljedećim potpoglavljima.

4.1. Početni zaslon

Početni zaslon nema nikakvih specifičnih funkcionalnosti koje bi ga dijelile od drugih zaslona aplikacije.

Svi zasloni aplikacije dijele isto zaglavlje (engl. *header*) koje sadrži logo, tražilicu i određene dugmadi koji se prikazuju ovisno o tome je li korisnik prijavljen (Slika 10) ili ne (Slika 9).

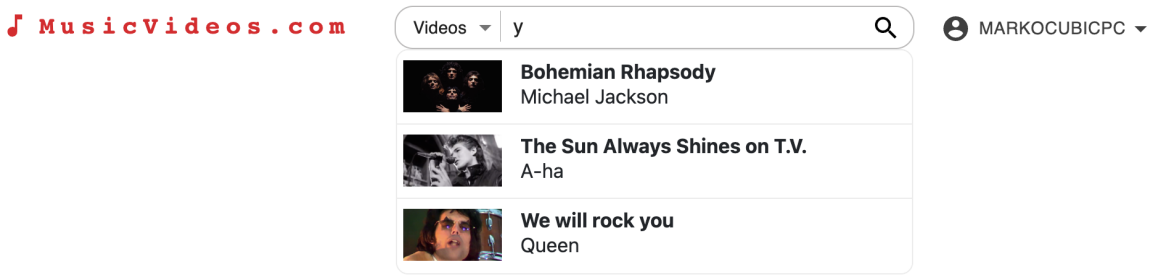
Slika 9: *Zaglavlje bez prijavljenog korisnika*



Slika 10: *Zaglavlje s prijavljenim korisnikom*

Logo, osim vizualne funkcije ima i funkciju vraćanja na početni zaslon prilikom klika na njega.

Tražilica omogućava korisniku pretraživanje umjetnika/bendova ili glazbenih videa, ovisno o filteru kojeg je korisnik označio. Upisivanjem slova u tražilicu može se pojaviti padajuća lista stavki (Slika 11) na koje se može kliknuti. Ako korisnik klikne na stavku bit će usmjeren na stranicu te stavke. To može biti stranica glazbenog videa ili stranica umjetnika ili benda (oboje dijele isti tip stranice).



Slika 11: *Zaglavlje s prijavljenim korisnikom*

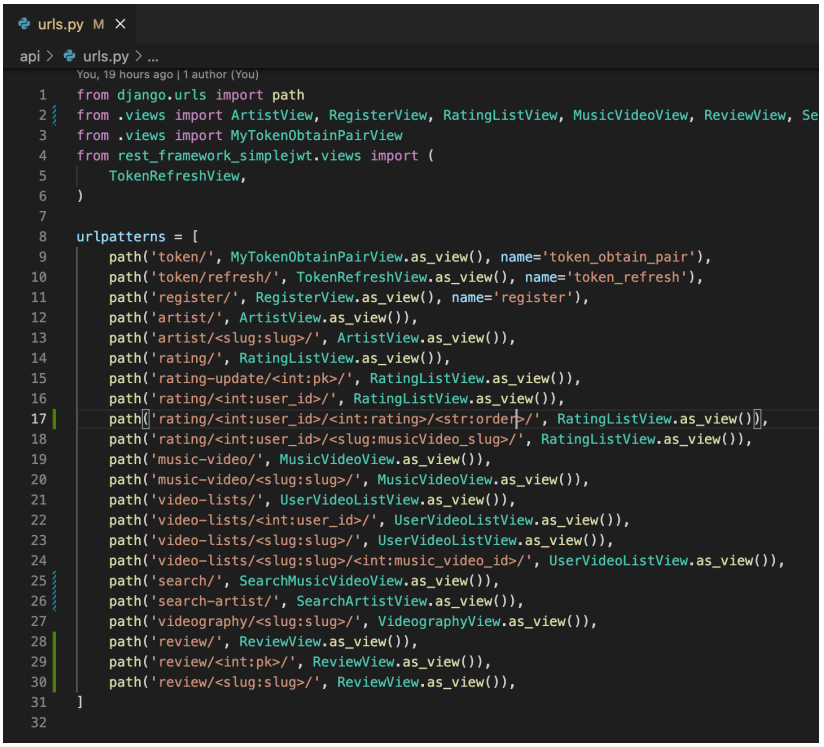
Na *backendu*, tražilica je implementirana u views.py datoteci u definiciji POST zahtjeva. U *viewu* za tražilicu filtriraju se svi umjetnici korištenjem "filter" metode u koju se dodaje upit "name__icontains" (Slika 12). "Name" predstavlja polje imena u "Artist" modelu, a "icontains" je upit koji traži postoji li ime koje ima u sebi niz slova koji se nalazi u dodanom parametru. Filtriranje nije osjetljivo na velika slova (engl. *case-insensitive*).

```
class SearchArtistView(APIView):
    def post(self, request):
        artist = Artist.objects.filter(
            name__icontains=request.data['parameter'])
        serializer = ArtistSerializer(artist, many=True)

        return Response(serializer.data)
```

Slika 12: Prikaz viewa tražilice

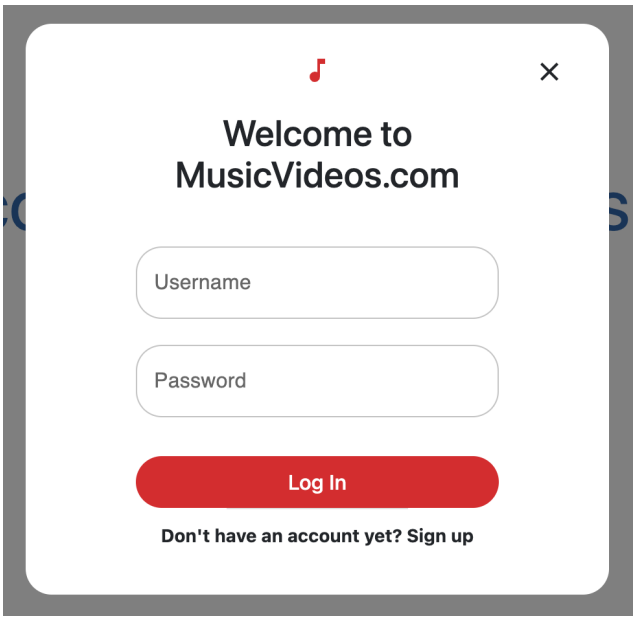
Svi viewovi na projektu koriste "APIView" klasu iz REST okvira koja omogućuje korištenje zahtjeva REST okvira umjesto Djangovog HTTP zahtjeva.

Zahtjevi definirani u *viewovima* pozivaju se uz pomoć *urlova* definiranih u "urls.py" datoteci. (Slika 13)

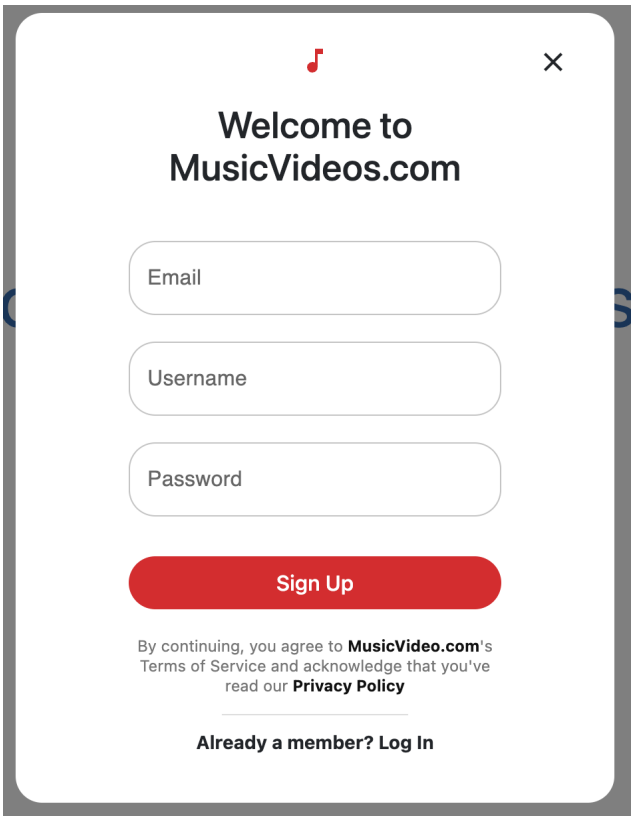


Slika 13: Prikaz *urlova* iz "urls.py" datoteke

Ako korisnik nije prijavljen onda će mu s desne strane zaglavlja biti prikazana dva dugmeta. Jedan za prijavu (engl. *sign in*) i jedan za registraciju (engl. *sign up*). Klikom na bilo koje od dugmeta otvara prozor (engl. *modal*), ali sadržaj prozora ovisi o tome koji je kliknut. Slika 14 prikazuje sadržaj prozora koji se otvori kad je kliknuto dugme za prijavu, a slika 15 sadržaj prozora kad je kliknuto dugme za registraciju. Uz to svaki sadržaj nudi opciju koja mijenja jedan od prikaza, opcija se mijenja ovisno o tome koji je sadržaj prikazan.

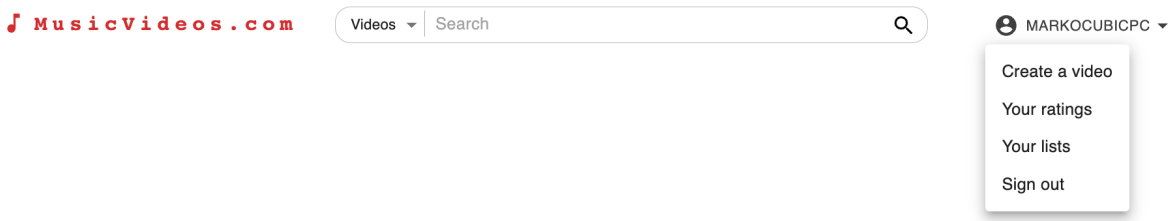


Slika 14: *Prozor za prijavu*



Slika 15: *Prozor za registraciju*

Ako je korisnik prijavljen onda će mu biti prikazano dugme s njegovim korisničkim imenom. Klikom na to dugme otvara se padajući izbornik (Slika 16) koji nudi nekoliko opcija koje korisnika usmjeravaju na različite rute.



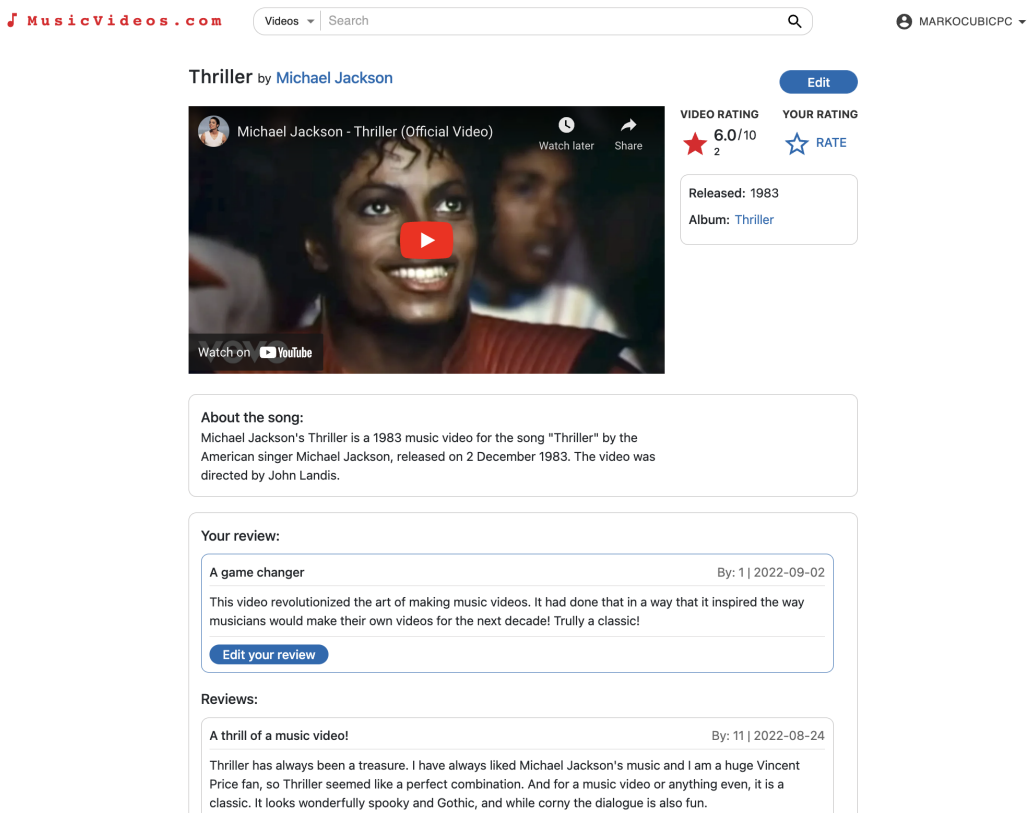
Slika 16: *Padajući izbornik*

4.2. Prikaz glazbenog videa

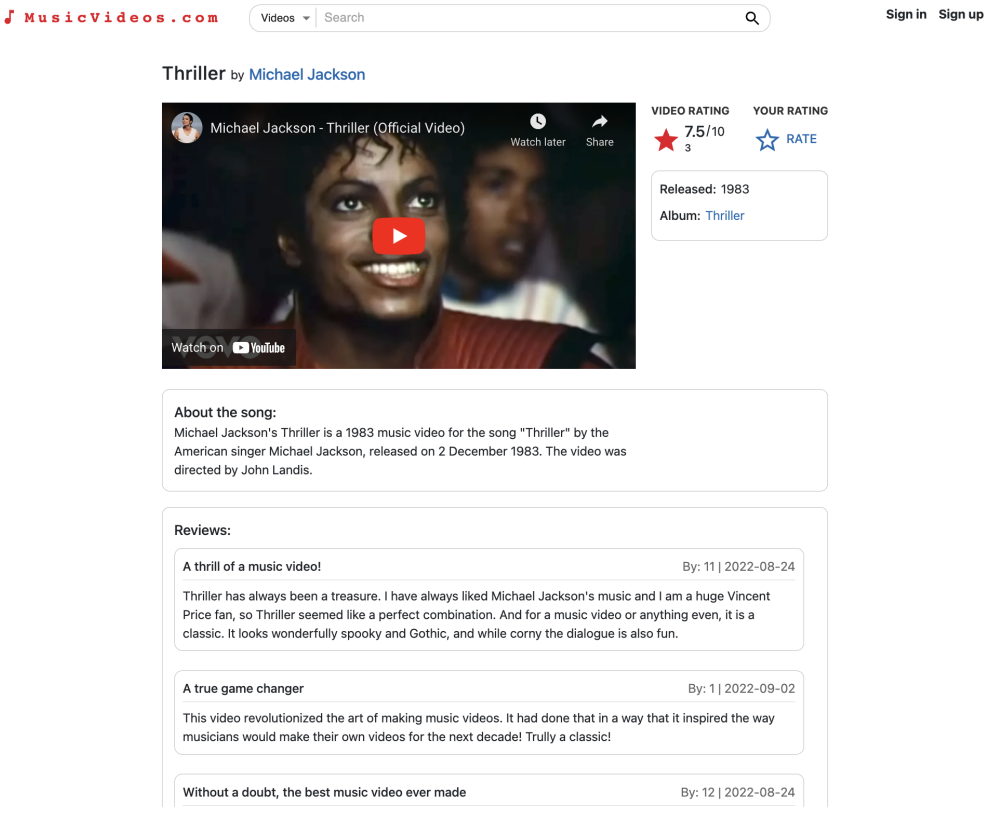
Zaslon za prikaz glazbenog videa korisnik može koristiti kako bi pregledavao informacije o glazbenom videu, ocjenjivao ga, pisao osvrt na njega te ga uređivao u slučaju da ima ovlasti za to. Svaki video ima svoju stranicu i svoju putanju (*engl. path*).

Od generalnih informacija na zaslonu su: ime videa, ime izvođača, *embedded* video s Youtubea koji isto tako služi kao slika, ukupna ocjena videa, korisnička ocjena, godina objave, ime albuma, opis videa i korisnički osvrti među kojima je i korisnikov vlastiti ako ga je napisao. Ako korisnik nije prijavljen (Slika 18), onda neće moći vidjeti svoju ocjenu, ni svoj osvrt. Ako je korisnik administrator onda će moći uređivati neke od informacija prikazanih na zaslonu kao što su naslov videa i opis videa. To mu je omogućeno klikom na dugme "*Edit*" koji se nalazi na vrhu zaslona (Slika 17). Kada korisnik završi s uređivanjem onda može kliknuti na "*Cancel*" dugme kako bi odbacio promjene ili "*Done editing*" dugme kojim sprema promjene. Ta dugmad prikaže se nakon što se klikne na "*Edit*" dugme, a kad se klikne jedan od njih "*Edit*" se vrati.

Youtube video nudi opciju *embedded* videa koja omogućava gledanje videa na drugim *web* stranicama osim samog Youtubea. Ako korisnik želi otići na Youtube da pogleda video, onda može samo kliknuti na ime videa koje će ga usmjeriti na Youtube stranicu.



Slika 17: *Prikaz glazbenog videa kad je korisnik prijavljen s ulogom administratora*



Slika 18: *Prikaz glazbenog videa kad korisnik nije prijavljen*

Prikaz glazbenog videa funkcionira uz pomoć zahtjeva koji su većinski implementirani u "MusicVideoView" *viewu* (Slika 19). U njemu se koristi "MusicVideoSerializer" serijalizator koji je definiran u "serializers.py" (Slika 20). Taj serijalizator definira neophodna polja koja su važna za stvaranje i ažuriranje novih glazbenih videa. Osim toga u metodi za stvaranje "create" polju "artist" dodaje se objekt umjetnika čiji je identifikacijski broj prethodno bio na tom polju. Tako će GET zahtjev vraćati objekt čiji će parametar "artist" sadržavati informacije tog umjetnika.

GET zahtjev vraćat će pojedini glazbeni video ako je preko *urla* poslan *slug*. Detekcija sluga omogućena je tako što je u "urls.py" dodana posebna "music-video/<slug:slug>/" url putanja koja prima slug. Ako se *urlu* ne nalazi slug onda će biti vraćeni svi videi.

POST zahtjevom se uz pomoć "MusicVideoSerializer" stvara glazbeni video. Da bi "artist" polje sadržavalo objekt "Artist" sa svim njegovim informacijama u klasu serijalizatora je potrebno proslijediti dodati kontekst (engl. *context*) atribut. U objektu konteksta uz pomoć request argumenta dodjeljuje se cijeli objekt umjetnika polju "artist".

PUT zahtjev, za razliku od GET zahtjeva mora se pozivati s *urlom* koji ima *slug* u sebi. *Slugom* se identificira koji točno objekt je potrebno ažurirati.

```
You, 23 minutes ago | 1 author (You)
class MusicVideoView(generics.ListAPIView):
    queryset = MusicVideo.objects.all()
    serializer_class = MusicVideoSerializer

    def get(self, request, slug=None):
        if slug:
            try:
                music_video = MusicVideo.objects.get(slug=slug)
                serializer = self.serializer_class(music_video)

            except MusicVideo.DoesNotExist:
                return Response({'status': 'error', 'message': 'Music Video does not exist'}, status=status.HTTP_400_

            return Response(serializer.data, status=status.HTTP_200_OK)

        music_videos = MusicVideo.objects.all()
        serializer = self.serializer_class(music_videos, many=True)

        return Response(serializer.data, status=status.HTTP_200_OK)

    def post(self, request):
        serializer = self.serializer_class(data=request.data, context={
                                           'artist': request.data['artist']})
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status=status.HTTP_201_CREATED)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

    def put(self, request, slug):
        video = MusicVideo.objects.get(slug=slug)
        serializer = self.serializer_class(
            video, data=request.data, partial=True)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

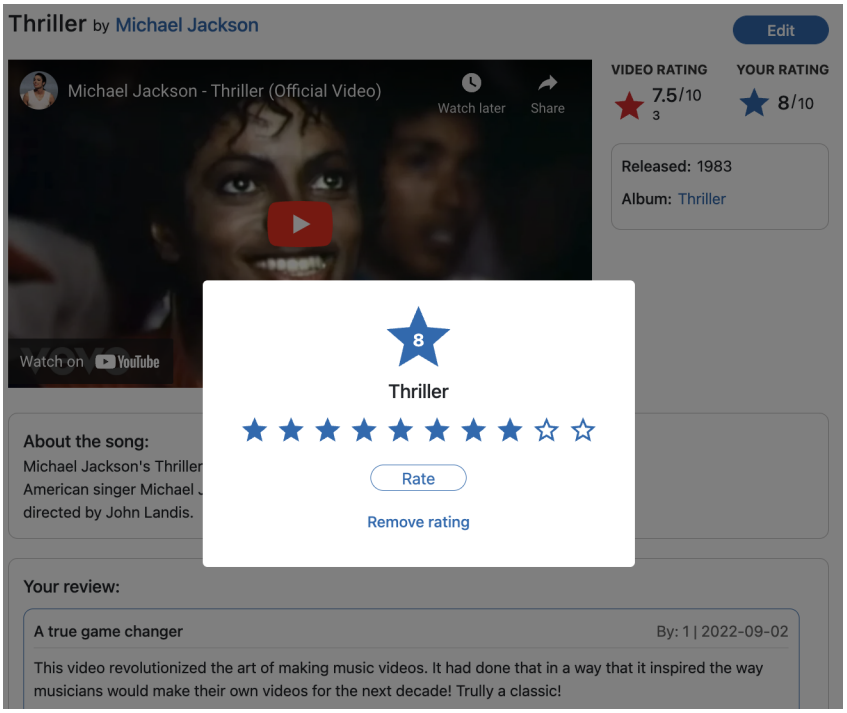
Slika 19: *Prikaz viewa za glazbena videa*

```
You, 29 minutes ago | 1 author (You)
class MusicVideoSerializer(serializers.ModelSerializer):
    artist = ArtistSerializer(read_only=True)

    You, 29 minutes ago | 1 author (You)
    class Meta:
        model = MusicVideo
        fields = ('id', 'artist', 'title', 'slug', 'release_year', 'album', 'image',
                  'yt_embedded', 'rate_score', 'votes_number', 'duration', 'song_description')

    def create(self, validated_data):
        validated_data['artist'] = Artist.objects.get(
            artist_id=self.context['artist'])
        return super().create(validated_data)
```

Slika 20: *Serijalizator "MusicVideo" modela*

Prijavljeni korisnik, klikom na neku od zvjezdica prikazanih u gornjem desnom kutu zaslona, može ocijeniti video ocjenama od 1 do 10. Klikom na jednu od tih zvjezdica otvara se prozorčić u kojem može označiti željenu ocjenu i potvrditi je. Ta ocjena će odmah biti uračunata u ukupni broj ocjena videa i bit će uvažena u sami prosjek svih ocjena koji čini ukupnu ocjenu videa. Kad korisnik ocijeni glazbeni video onda će je moći

vidjeti na zaslonu tog videa. Ona će biti prikazana plavom zvjezdicom, dok je ukupna ocjena videa prikazana crvenom. Ako korisnik želi promijeniti ocjenu onda to isto tako može učiniti klikom na jednu od zvjezdica, a ako je želi izbrisati ona to može učiniti klikom na botun "Remove rating" na dnu prozora (Slika 21).



Slika 21: *Prozorčić za ocjenjivanje*

Računanja ukupnog broja ocjena i ukupnog prosjeka svih ocjena videa odrađuje se u POST, PUT i DELETE definicijama zahtjeva "RatingListView" *viewa* (Slika 22).

Ako korisnik prvi put ocijeni video odradit će se inkrementacija na "votes_number" polju u objektu tog videa. Nakon toga će se izračunati ukupni prosjek tako što se na polje videa "rate_score" dodaje rezultat postignut uzimanjem svih ocjena tog videa i izvlačenjem njihovog prosjeka "aggregate" metodom i "Avg" klasom unesenom iz django modela. Ako korisnik ažurira svoju ocjenu onda se samo ponovo računa prosjek. Ako korisnik briše ocjenu onda se dekrementira "votes_number" polje videa i ponovo se računa prosjek (Slika 1).

```
def post(self, request):
    serializer = RatingSerializer(data=request.data, context={
                                  'musicVideo_id': request.data['musicVideo']})
    if serializer.is_valid():
        serializer.save()

        ratings = Rating.objects.filter(
            musicVideo_id=request.data['musicVideo']).aggregate(Avg('rating'))
        music_video = MusicVideo.objects.get(id=request.data['musicVideo'])
        music_video.votes_number = music_video.votes_number + 1
        music_video.rate_score = ratings['rating__avg']
        music_video.save()

        return Response(serializer.data, status=status.HTTP_201_CREATED)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

def put(self, request, pk):
    rating = self.get_object(pk)
    serializer = RatingSerializer(rating, data=request.data)
    if serializer.is_valid():
        serializer.save()

        ratings = Rating.objects.filter(
            musicVideo_id=rating.musicVideo.id).aggregate(Avg('rating'))
        music_video = MusicVideo.objects.get(id=rating.musicVideo.id)
        music_video.rate_score = ratings['rating__avg']
        music_video.save()

        return Response(serializer.data)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

def delete(self, request, pk):
    snippet = self.get_object(pk)
    snippet.delete()

    ratings = Rating.objects.filter(
        musicVideo_id=snippet.musicVideo.id).aggregate(Avg('rating'))
    music_video = MusicVideo.objects.get(id=snippet.musicVideo.id)
    music_video.votes_number = music_video.votes_number - 1
    music_video.rate_score = ratings['rating__avg']
    music_video.save()

    return Response(status=status.HTTP_204_NO_CONTENT)
```

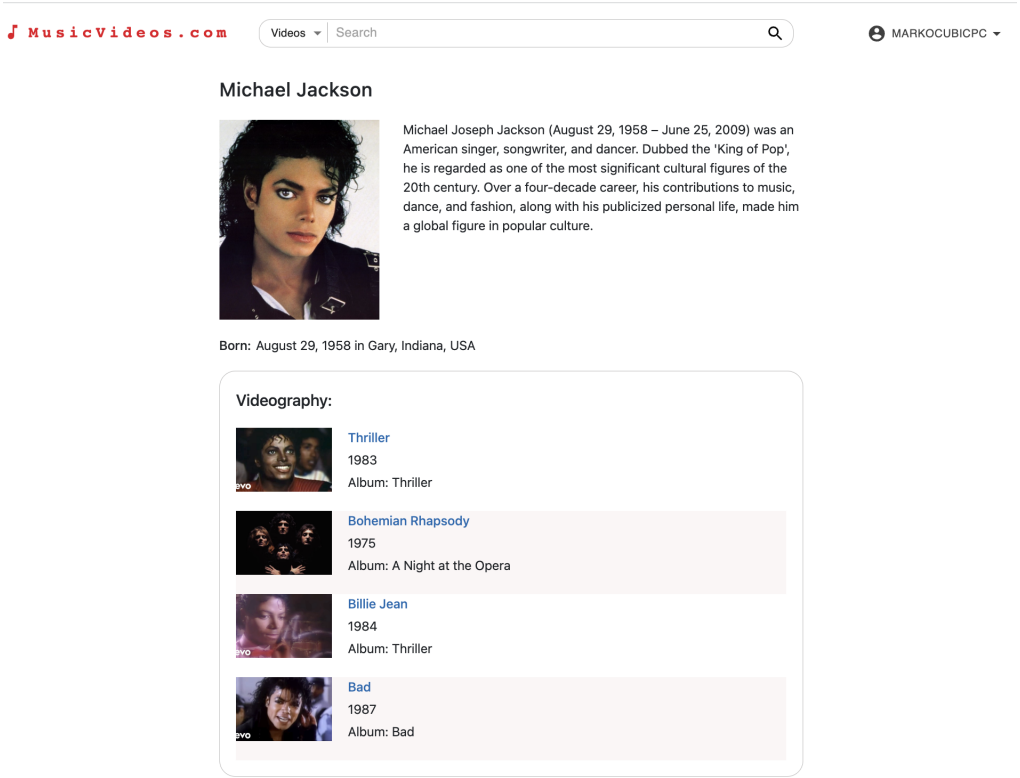
Slika 22: *Prikaz zahtjeva "RatingListView" viewa*

Ako prijavljeni korisnik nije prethodno napisao osvrt na određeni glazbeni video onda će na zaslonu tog videa biti prazna *forma* koje će mu to omogućiti. Nakon što potvrdi unos, taj osvrt će biti vidljivo odvojen na vrhu liste osvrta. Nakon toga korisnik može uređivati svoj osvrt po svojoj volji klikom na dugme "*Edit your review*", a završiti uređivanje klikom na jedno od dugmadi ispod *forme* za unos. Ispod korisnikovog osvrta izlistani su svi ostali osvrti drugih korisnika na taj određeni video.

4.3. Prikaz umjetnika

Umjetnik i bend dijele isti zaslon (Slika 23) i razlikuju se po zadnjem dijelu putanje isto kao i glazbena videa. Svi korisnici vide iste informacije na zaslonu bez obzira na ulogu koju imaju ili nemaju.

Na zaslonu su prikazane sljedeće informacije: ime umjetnika ili benda, opis, dan rođenja ili osnutka, videografija koja prikazuje listu videa s kojim je bend ili umjetnik povezan i koja su prisutna na "MusicVideos.com" *web* aplikaciji. Svaki od izlistanih video sadrži neke od svojih osnovnih informacija i poveznicu (engl. *link*) koja vodi na njegov zaslon.

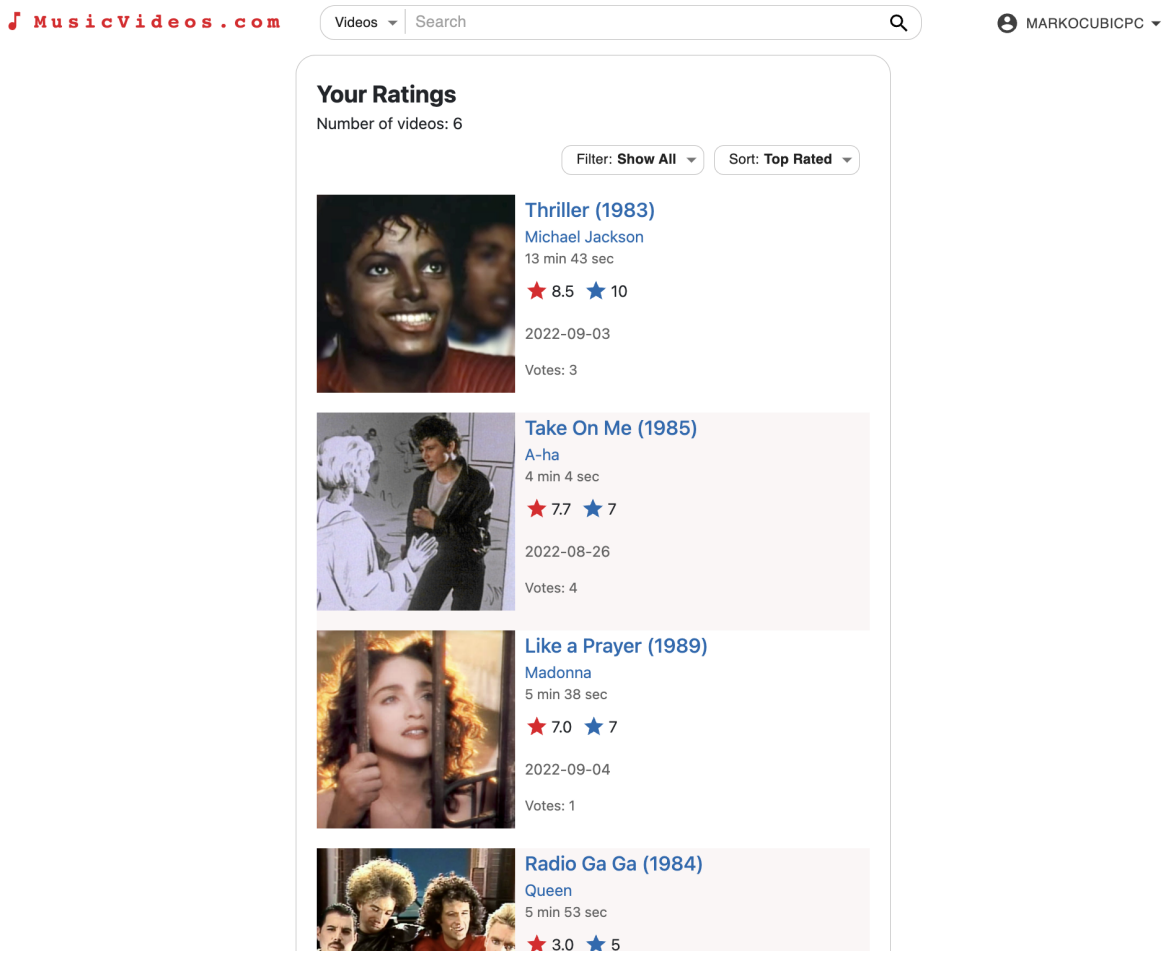


Slika 23: *Prikaz zaslona umjetnika*

4.4. Prikaz liste ocjena

Prijavljeni korisnik može pristupiti listi svih svojih ocijenjenih videa kroz padajući izbornik aplikacije koji se otvara klikom na korisničko ime u gornjem desnom kutu. Ta lista se sastoji od stavki od kojih svaka sadržava: ime videa i ime umjetnika s poveznicama

koje vode na njihove zaslone, vrijeme trajanja videa, ukupnu ocjenu i ocjenu korisnika, datum ocjenjivanja i broj ocjena (Slika 24). Lista se može filtrirati po veličini ocjene koju je korisnik dao videu. Lista se isto tako može sortirati prema najbolje ocijenjenim videima ili prema zadnjim ocijenjenim videima. Filtriranje i sortiranje omogućavaju "Filter" i "Sort" botuni s padajućim izbornicima. Klikom na neku od stavki u padajućim izbornicima tih botuna lista videa se osvježava.



Slika 24: *Prikaz zaslona korisničke liste ocijenjenih glazbenih videa*

Filtriranje je izvedeno uz pomoć "filter" metode koja je zapravo upit koji izvlači sve objekte iz liste pod određenim kriterijem. U ovom filteru kriterij je da objekti moraju imati identifikacijski broj prijavljenog korisnika. Sortiranje je izvedeno "order_by" metodom u koju su dodana polja prema kojima će se sortirati. Lijevo od polja dodan je znak minus koji zadaje silazno sortiranje (Slika 25).

```
elif (user_id):
    ratings = Rating.objects.all().filter(user_id=user_id)
    if (rating):
        ratings = ratings.filter(rating=rating)
    if (order):
        if order == 'date':
            ratings = ratings.order_by('-date')

        elif order == 'top':
            ratings = ratings.order_by('-rating')

        serializer = RatingSerializer(ratings, many=True)
```

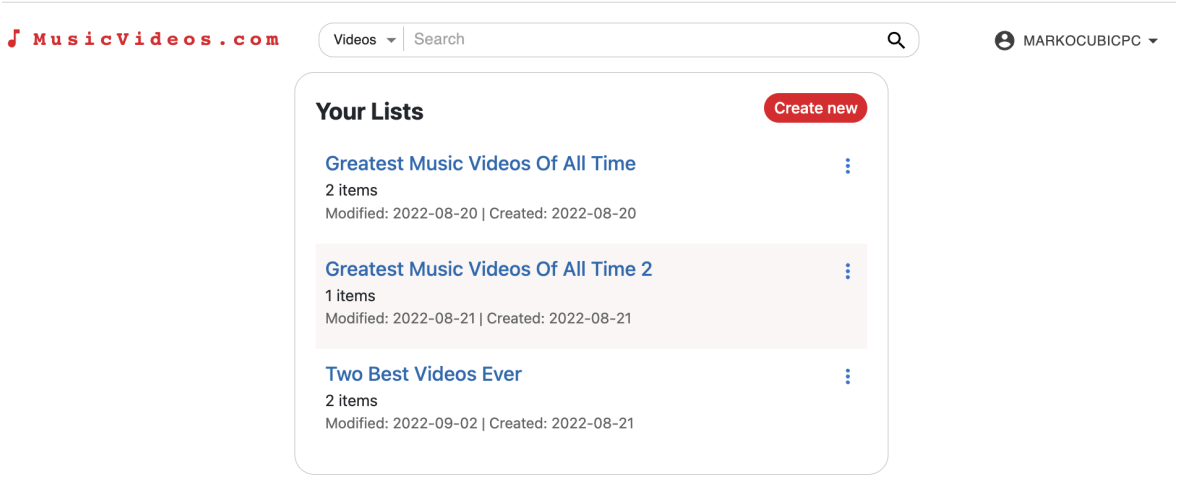
Slika 25: *Prikaz zaslona umjetnika*

4.5. Prikaz korisničkih lista glazbenih videa

Određenoj listi pristup ima samo prijavljeni korisnik koji ju je stvorio. Isto vrijedi i za listu korisničkih lista.

Korisničke liste imaju dva zaslona kojima korisnik može pristupiti. Prvi zaslon prikazuje sve liste koje je korisnik stvorio (Slika 26), a drugi pojedinu listu sa svim njenim informacijama (Slika 27).

Zaslonu koji prikazuje sve korisnikove liste pristupa se kroz padajući izbornik aplikacije koji se otvara klikom na korisničko ime u gornjem desnom kutu. Na tom zaslonu prikazane su stavke od kojih svaka sadrži sliku, ime liste koje je ujedno i poveznica koja vodi na zaslon te liste, broj stavki liste, datum uređivanja i datum stvaranja te s desne strane dugme koje otvara padajući izbornik s dvije opcije. Prva opcija je "*Edit*" koja korisnika vodi na zaslon te liste s već unaprijed otvorenim načinom rada koji omogućava uređivanja liste. Druga opcija "*Delete*" služi za brisanje liste.

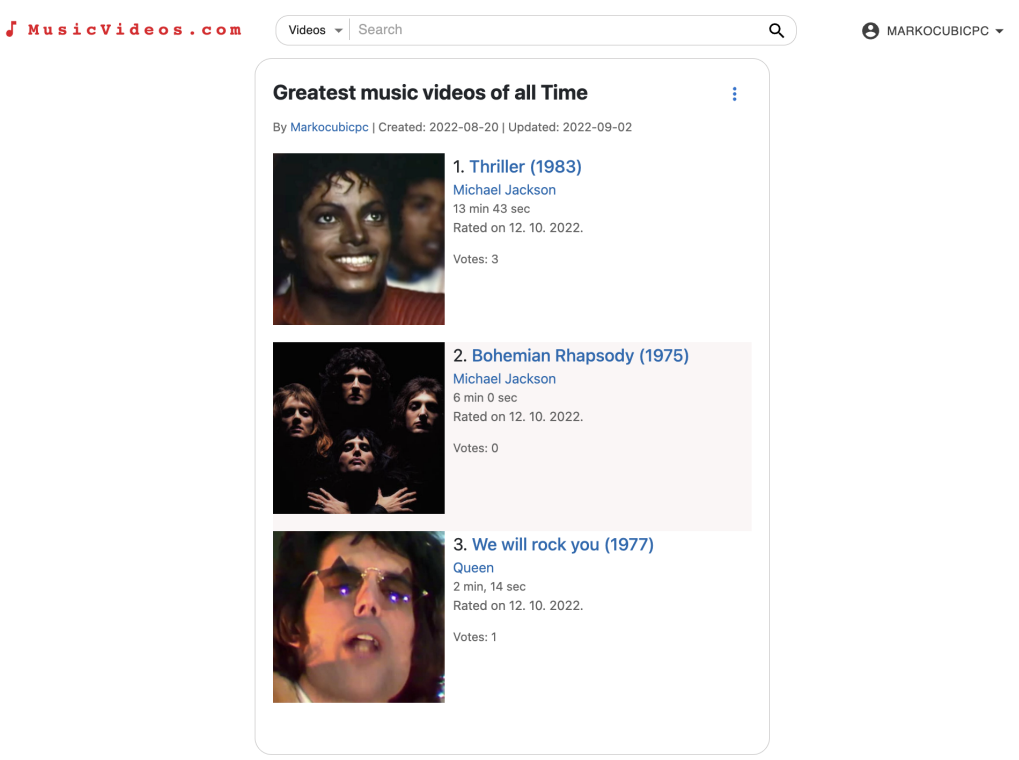


Slika 26: *Prikaz svih korisnikovih lista*

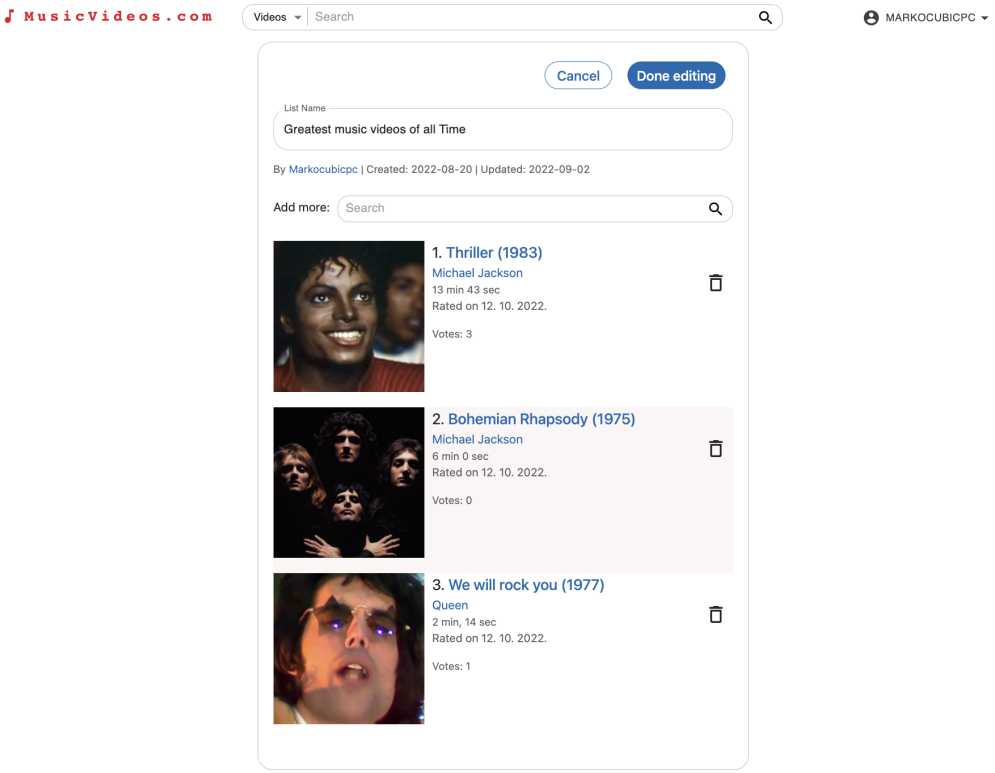
Zaslon pojedine liste prikazuje listu glazbenih videa koje je korisnik dodao. Stavke u listi prikazane su istom komponentom kao kod liste ocijenjenih videa tako da sadrže iste tipove informacija, jedina razlika je što u slučaju ove liste stavka ima redni broj prije imena videa. Na vrhu liste nalazi se ime liste, a ispod toga ime korisnika koji ju je stvorio, datum stvaranja te datum posljednjeg uređivanja.

U gornjem desnom kutu nalazi se isto dugme kao na zaslonu korisnikovih lista i ima iste opcije u svom padajućem izborniku koje rade iste funkcije.

Klikom na dugme “Edit” korisnik otvara način rada koji nudi uređivanje liste (Slika 28). U tom načinu rada na vrhu zaslona dodaju se dva dugmeta: “Cancel” kojim korisnik poništava sve promjene i “Done editing” kojim potvrđuje i sprema sve promjene. Ime liste pojavljuje se u polju za unos gdje ga korisnik može mijenjati. Poviše liste videa pojavljuje se polje tražilice uz pomoć korisnik može dodavati nova videa u listu tako što klikne na neki od ponuđenih rezultata pretraživanja. Budući da je moguće izrađivati samo liste glazbenih videa, tražilica pretražuje samo njih dok umjetnike i bendove ignorira. Svaka od stavaka liste dobiva svoje dugme s ikonom kante za otpad kojim korisnik može ukloniti video s liste.



Slika 27: *Prikaz korisničke liste*



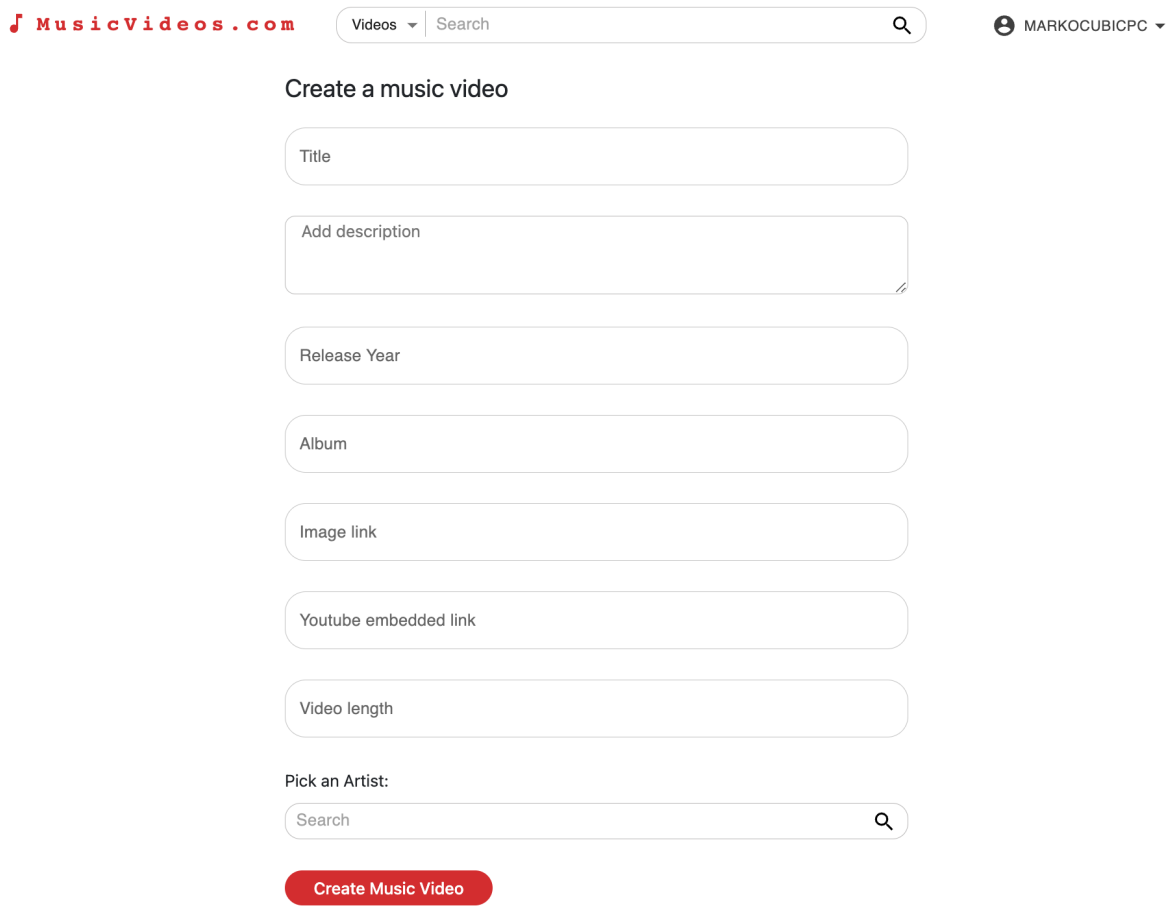
Slika 28: *Prikaz korisničke liste u načinu rada za uređivanje*

4.6. Forma za stvaranje glazbenog videa

Prijavljeni korisnik s ulogom administratora ima mogućnost stvaranja glazbenog videa. Stvoreni video se dodaje u bazu podataka i moguće mu je pristupiti na "MusicVideos.com" *web* aplikaciji.

Zaslon za stvaranje videa sadrži formu (Slika 29) koja se sastoji od nekoliko polja za unos: ime videa, opis, godina objave videa, album, poveznica za sliku, Youtube *embedded* poveznica, trajanje videa, umjetnik. Sva od navedenih polja su obična polja za unos osim onoga za umjetnika. To polje je tražilica uz pomoć koje korisnik odabire jednog od umjetnika ili bendova s "MusicVideos.com" *web* aplikacije.

Pritiskom na dugme "*Create Music Video*" korisnik stvara video te ga se preusmjerava na zaslon tog novog glazbenog videa.



Slika 29: *Prikaz zaslona za stvaranje glazbenih videa*

5. Zaključak

Ovim završnim radom prikazan je proces i rezultat stvaranja jednostavne *web* aplikacije koristeći najmodernije i najzastupljenije tehnologije za izradu *web* aplikacija. Isto tako korištene su moderne i aktualne konvencije pisanja programskog kôda i organizacije strukture *backend* i *frontend* dijelova projekta. Proces planiranja projekta i načina izrade detaljno je opisan. Uloga svake od tablica baze podataka objašnjena je i prikazana u obliku tablica entiteta, kao i njihovi međusobni odnosi koji su prikazani dijagram ER-om. Glavni zasloni *web* aplikacije prikazani su slikama, a tekstom je objašnjena njihova svrha i način korištenja. Predana je pažnja i svim ključnim tehnologijama koje su bile potrebne da se ovaj završni rad uspješno odradi. Svaka od tih tehnologija opisana je tako da je jasno zašto su korištene.

Unatoč tome što je postignut cilj stvaranja temelja i osnovnih funkcionalnosti ove *web* aplikacije, ostaje još dosta prostora za poboljšanja, od samog refaktoriranja kôda pa do dodavanja novih zanimljivih značajki. Neke od njih su: dodavanje korisničkog profila, omogućavanje stvaranja i uređivanja umjetnika ili benda, povezivanje na CMS (engl. *Content Management System*) kako bi se olakšao unos podataka, dodavanje novih vrsta informacija na zaslone videa i umjetnika/bendova, dodavanje još nekih od elemenata društvenih mreža itd.

6. Literatura

[1] https://docs.python.org/3/ (Posjećeno 27.08.2022.)

[2] https://www.django-rest-framework.org/ (Posjećeno 27.08.2022.)

[3] https://www.w3.org/Style/CSS20/history.html (Posjećeno 28.08.2022.)

[4] https://www.postman.com/ (Posjećeno 05.09.2022.)

[5] https://insights.stackoverflow.com/survey/2021 (Posjećeno 28.08.2022.)

[6] https://reactjs.org/docs/faq-structure.html (Posjećeno 05.09..2022.)