

UNIVERZITET U NIŠU

ELEKTRONSKI FAKULTET

**MODEL ZA PREPOZNAVANJE IMENOVANIH ENTITETA**

Tehnički izveštaj

Studijski program: Računarstvo i informatika

Modul: Softversko inženjerstvo

Studenti:

Uroš Milić, br. ind. 1615  
Jovan Mladenović br. ind. 1277Profesor:

Dr Aleksandar Milosavljević

Niš, jun 2024. godina

Sadržaj

[1. Uvod 3](#_Toc169035243)

[2. Obrada prirodnog jezika 4](#_Toc169035244)

[2.1. Prepoznavanje imenovanih entiteta 4](#_Toc169035245)

[3. Arhitektura sistema 5](#_Toc169035246)

[3.1. React frontend 5](#_Toc169035247)

[3.2. Flask backend 6](#_Toc169035248)

[3.3. Model za prepoznavanje imenovanih entiteta 8](#_Toc169035249)

[4. Literatura 11](#_Toc169035250)

# Uvod

U ovom projektu biće predstavljenvirtualni asistent dizajniran da pušta muziku na osnovu zahteva korisnika nazvan **MusicHub**. Napravljen je korišćenjem modela mašinskog učenja, i Youtube API-ja koji omogućava pretragu na Youtube-u. MusicHub je sposoban da izdvoji relevantne entitete koji predstavljaju šta to korisnik želi da sluša, kao što su imena izvođača, naslovi pesama, albuma, žanrova i sl. Izvorni kod ovog sistema je dostupan na javnom GitHub repozitorijumu, na sledećoj adresi: <https://github.com/umilic16/dl-projekat>

Projekat je izrađen korišćenjem **Flask** backend-a koji komunicira sa modelom za prepoznavanje imenovanih entiteta (engl. *Named Entity Recognition,* skraćeno *NER*). Frontend je napravljen pomoću **React**-a, pružajući korisnicima jednostavan ali intuitivan korisnički interfejs.

U narednom poglavlju će biti ukratko definisana tehnika **prepoznavanja imenovanih entiteta** (engl. *Named Entity Recognition*, skraćeno *NER*), koja spada u tehnike **obrade prirodnog jezika** (engl. *Natural Language Processing*, skraćeno *NLP*). Zatim je u trećem poglavlju dat pregled komponenti MusicHub-a, uključujući njegovu arhitekturu, model za prepoznavanje imenovanih entiteta i API koji se koristi, pored toga biće opisano i kako je obučen model, kako su prikupljeni podaci za obuku, kao i kako je API integrisan u sistem.

# Obrada prirodnog jezika

Obrada prirodnog jezika (engl. *Natural Language Processing*, skraćeno *NLP*) je interdisciplinarna podoblast lingvistike, računarstva i veštačke inteligencije koja se fokusira na omogućavanje računarima da razumeju, tumače i generišu ljudski jezik. Uključuje primenu različitih računarskih tehnika i algoritama za analizu i obradu teksta i govornih podataka na prirodnom jeziku. Cilj je razumevanje sadržaja dokumenata, uključujući kontekstualne nijanse jezika u njima. Tehnologija tada može precizno da izdvoji informacije sadržane u dokumentima, kao i da kategorizuje i organizuje same dokumente [5].

Neke od tehnika obrade prirodnog jezika su:

* Tokenizacija
* Stemovanje i lematizacija
* Prepoznavanje vrste reči (engl. *Part-of-speech tagging*, skraćeno *POS tagging*)
* Prepoznavanje imenovanih entiteta
* Analiza sentimenta
* …

## Prepoznavanje imenovanih entiteta

Prepoznavanje imenovanih entiteta (engl. *Named Entity Recognition*, skraćeno *NER*) uključuje prepoznavanje i klasifikovanje imenovanih entiteta u tekstu, kao što su ljudi, organizacije, lokacije i datumi, što pomaže u razumevanju konteksta korisničkog unosa. Nakon što se završe prethodni koraci u obradi vrši se prepoznavanje imenovanih entiteta korišćenjem **gramatičkih pravila** i/ili algoritama **mašinskog učenja** [4][5].

**Pristupi zasnovani na gramatičkim pravilima** koriste ručno napisana pravila za identifikaciju entiteta na osnovu obrazaca u tekstu. Ovaj pristup postiže veću preciznost ali ima i veću cenu jer je za pisanje pravila neophodan rad računarskih lingvista [4][5].

**Pristupi zasnovani na mašinskom učenju** koriste velike količine označenih podataka za obuku modela koji može automatski da identifikuje entitete u tekstu [4][5].

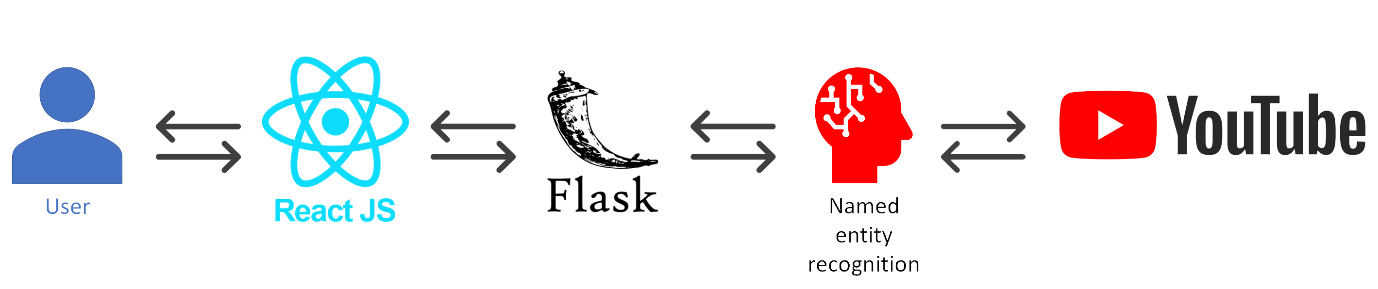
**Hibridni pristupi** kombinuju metode zasnovane na pravilima i mašinskom učenju kako bi se poboljšala tačnost prepoznavanja, i smanjio napor potreban za obeležavanje [4][5].

Primer [5]:

* Korisnički unos: „Džim je 2006. kupio 300 akcija kompanije Acme Corp“.
* NER algoritam bi prepoznao sledeće entitete: „Džim“ bi bio prepoznat kao osoba, „Acme Corp.“ kao organizacija a „2006“" kao datum.
* Rezultat: „[Džim]osoba je kupio 300 akcija [Acme Corp]organizacija [2006]datum“.

# Arhitektura sistema

MusicHub je izrađen korišćenjem klijent-server arhitekture, sa frontend-om i backend-om koji vrše interakciju preko API-ja.

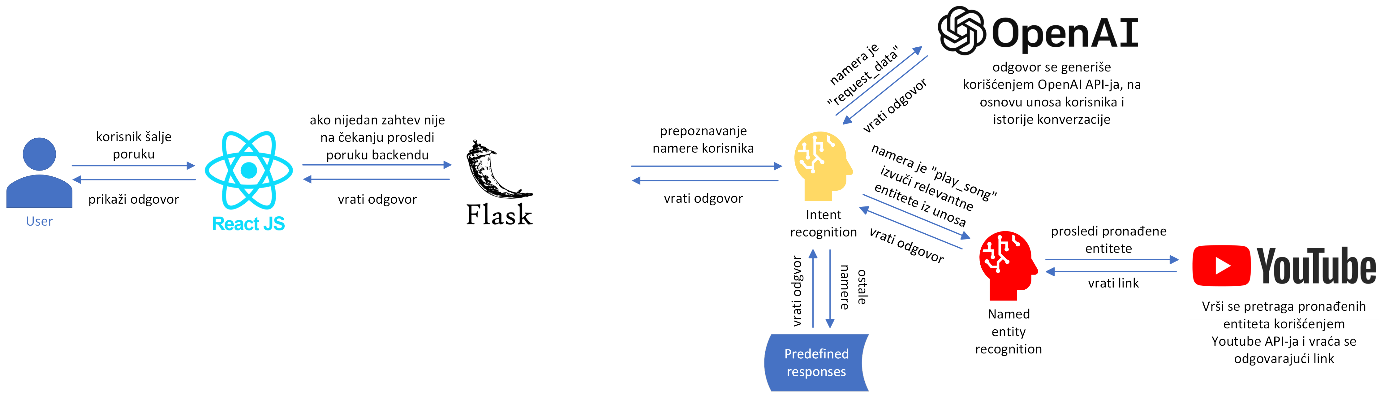


Slika 1. Arhitektura sistema

## React frontend

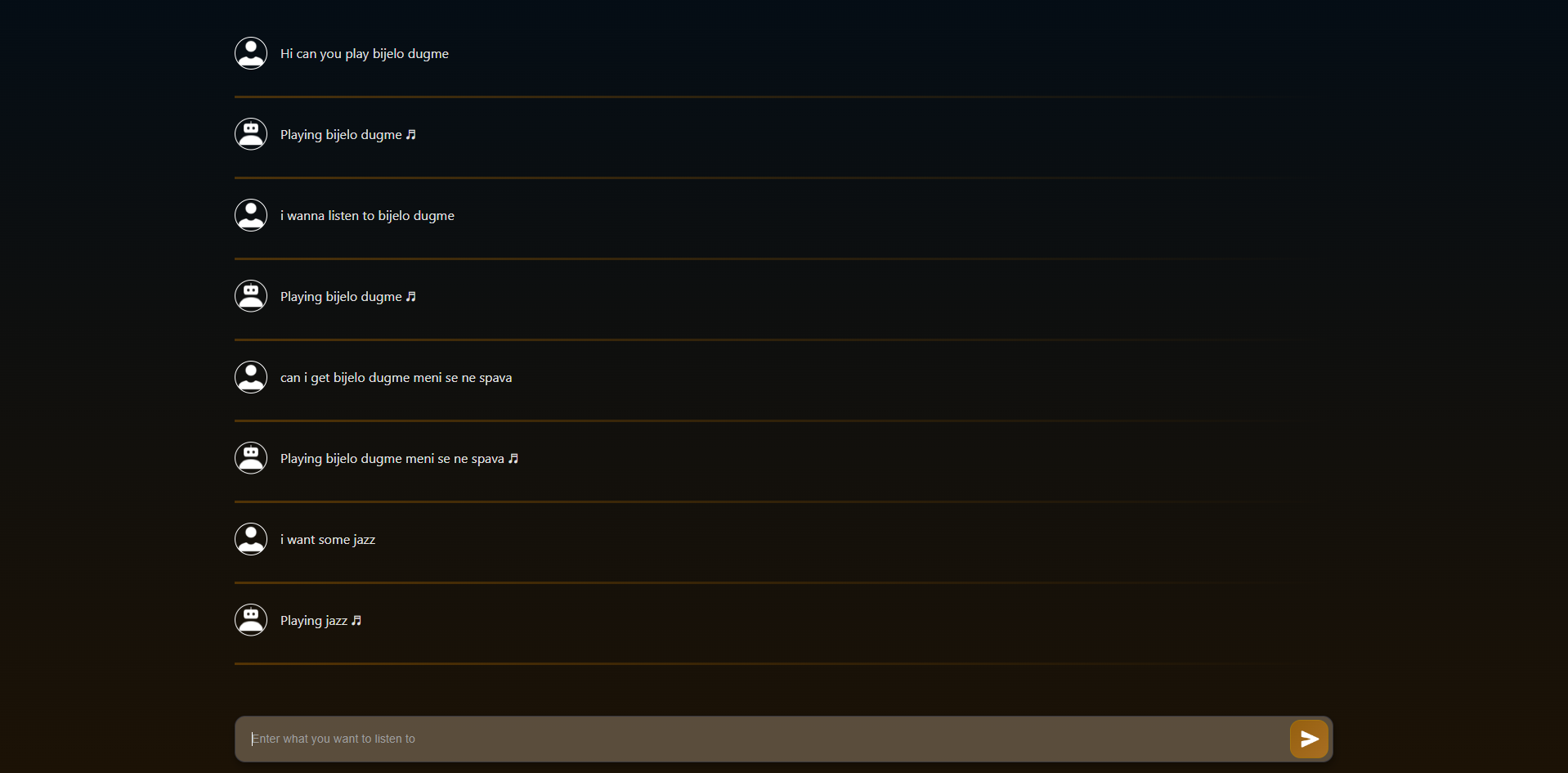
Frontend je napravljen pomoću React-a, popularnog JavaScript framework-a za kreiranje korisničkih interfejsa. Frontend komunicira sa backend-om koristeći HTTP (*HyperText Transfer Protocol*) zahteve, tako što šalje korisnički unos i prima odgovore u JSON (*JavaScript Object Notation*) formatu.

React frontend je odgovoran za prikazivanje čet interfejsa, koji omogućava korisnicima da interaguju sa virtualnim asistentom. Kada korisnik pošalje poruku, frontend šalje zahtev Flask backend-u sa tekstom poruke putem POST zahteva. Odgovor primljen od backend-a se zatim prikazuje u interfejsu za ćaskanje.



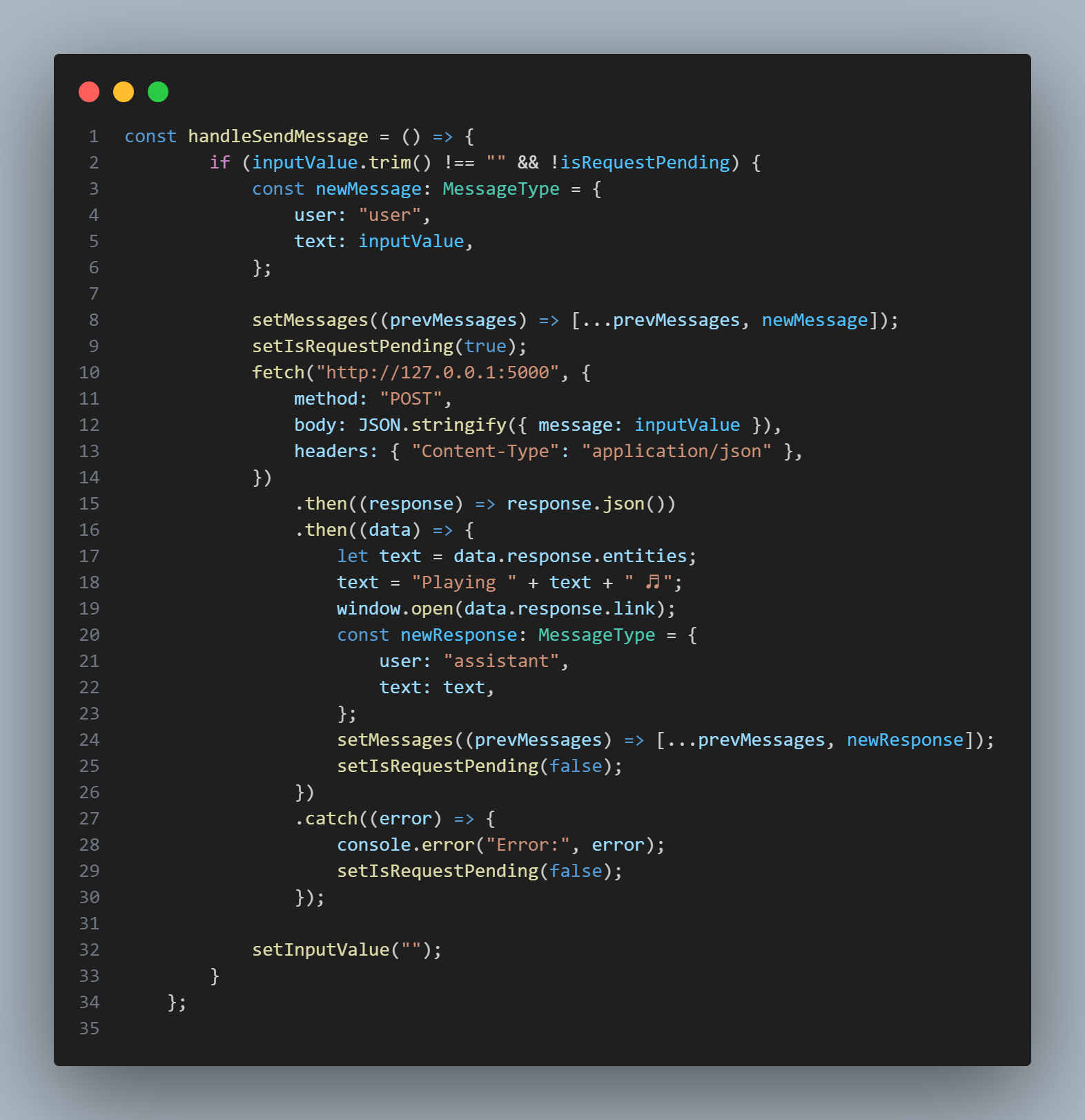
Slika 2. Interakcija frontend-a i backend-a

Frontend se sastoji od čet komponente koja održava stanje tj. istoriju konverzacije tako što prikazuje poruke koje je korisnik poslao, i odgovore koje je dobio. Kada korisnik pošalje poruku, poziva se funkcija **handleSendMessage**. Ova funkcija proverava da li polje za unos korisnika nije prazno i da li nema zahteva na čekanju. Ako su oba uslova tačna ažurira se stanje konverzacije tako da uključuje novu poruku, postavlja se stanje **isRequestPending** na *tačno* (kako bi sistem znao da je jedan zahtev već u toku) i šalje se POST zahtev Flask backend-u sa tekstom poruke.



Slika 3. Izgled frontend-a (primer konverzacije)

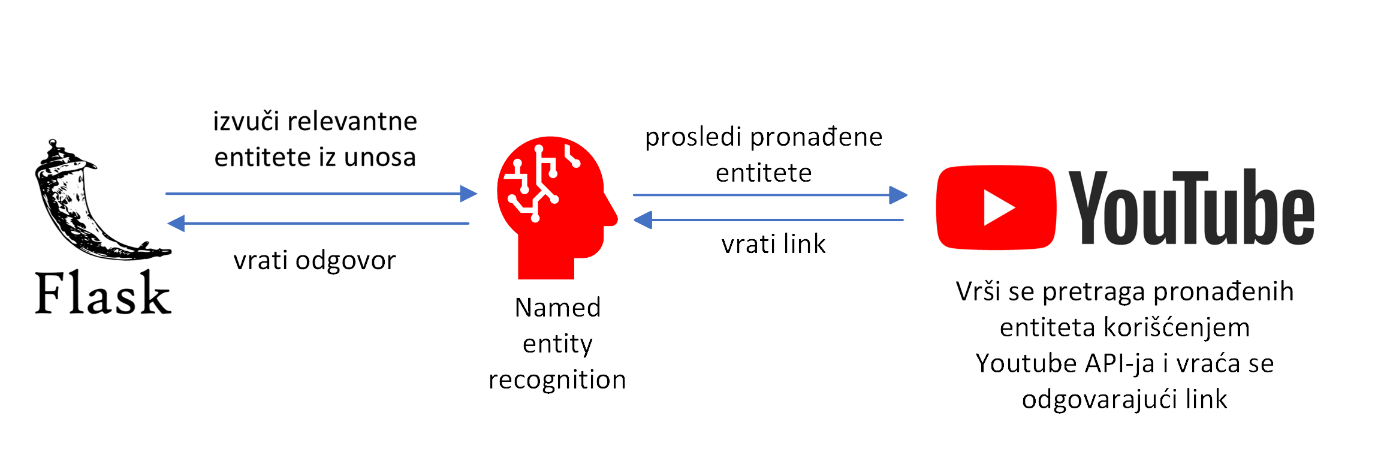
Ako je zahtev uspešan, odgovor dobijen od backend-a je link koji se otvara u novom prozoru. Nakon prikazivanja odgovora **isRequestPending** se postavlja na *netačno* i sistem je spreman da prihvati naredne zahteve korisnika.



Slika 4. Funkcija handleSendMessage

## Flask backend

Backend je napravljen pomoću Flask-a, web framework-a za Python. Flask aplikacija služi kao API za frontend, obrađuje dolazne zahteve i vraća odgovarajuće odgovore. Backend deo se sastoji od nekoliko komponenti, uključujući model za prepoznavanje imenovanih entiteta, kao i integraciju **Youtube API**-ja za reprodukciju muzike.



Slika 5. Tok obrade korisnikovog unosa

Kada korisnik pošalje poruku MusicHub-u frontend prosleđuje poruku Flask backend-u, u JSON obliku, preko POST zahteva. Funkcija **handle\_request** rukuje ovim zahtevima, zatim nakon primljenog zahteva poziva funkciju **search\_youtube**.



Slika 6. Funkcija search\_youtube

Ova funkcija koristi **Youtube API** i model za **prepoznavanje imenovanih entiteta**. Nakon pozivanja ova funkcija koristi model za prepoznavanje imenovanih entiteta koji je istreniran da prepozna šta to korisnik želi da sluša, kog izvođača, pesmu, album, žanr i sl. Ako se pronađu relevantni entiteti u unosu korisnika funkcija vrši pretragu pomoću Youtube API-ja koji vraća odgovarajući URL.



Slika 7. Primer prepoznavanja entiteta u korisnikovom unosu

## Model za prepoznavanje imenovanih entiteta

Model za prepoznavanje imenovanih entiteta je istreniran da prepozna entitete u korisnikovom unosu koji se odnose na izvođače, pesme, albume, žanrove i sl. Korišćenjem ovog modela se iz korisnikovog unosa izvlače informacije koje predstavljaju šta to korisnik želi da sluša, ove informacije se zatim mogu proslediti servisima za pretragu/puštanje muzike kako bi se ispunio zahtev korisnika. Model je istreniran korišćenjem **spaCy** biblioteke. SpaCy je „open-source“ Python biblioteka koja se koristi za obradu prirodnog jezika. Pruža razne funkcije poput tokenizacije, prepoznavanja vrste reči, prepoznavanje imenovanih entiteta itd. Ima razne prethodno obučene modele, za različite svrhe i jezike, pored toga korisnici mogu kreirati i izvršiti obuku svojih modela korišćenjem funkcionalnosti ove biblioteke.

Prvi korak u procesu kreiranja ovog modela je prikupljanje i obrada podataka koji će biti korišćeni za obuku. Podaci za obuku su prikupljeni iz mnogih skupova podataka skinutih sa sajta **keggle** [1], kao i podataka sa **vikipedije**. Skupovi podataka skinutih sa keggle sajta su u CSV formatu, i sadrže razne informacije relevantne za obuku ovog modela ali i mnoge irelevantne informacije, pa su kreirane funkcije koje vrše obradu ovih fajlova kako bi se kreirali podaci za obuku. Ova obrada je izvršena tako što su prvo sve kolone koje predstavljaju imena izvođača, albuma, pesama, žanrova, instrumenata i sl. izvučene i sačuvane svaka u zaseban fajl (kolone sa albumima u fajlu za albume itd.).

Iako je količina prikupljenih podataka verovatno dovoljna, napisane su funkcije koje prikupljaju podatke sa vikipedije kako bi skup podataka za obuku bio još veći i potpuniji, najviše zbog manjka entiteta (u većini dostupnih skupova podataka) koji predstavljaju muzičke instrumente i neke manje popularne žanrove i izvođače. Podaci sa vikipedije su prikupljeni korišćenjem **BeautifulSoup** biblioteke, tako što se iz HTML strukture stranice sa koje se pribavljaju podaci izvlače delovi koji sadrže potrebne informacije. Npr. stranica koja predstavlja listu svih muzičara [2] sadrži linkove koji vode do liste muzičara za određeni žanr, prvo se pronalaze ovi linkovi u strukturi stranice, zatim se pristupa svakom od linkova i izvlače imena izvođača koja se dodaju u fajl koji sadrži izvođače.

Slika 8. Deo liste muzičara po žanrovima sa vikipedije [2]



Slika 9. Funkcija koja prikuplja izvođače sa vikipedije

Na sličan način su prikupljena i imena raznih žanrova i instrumenata, postupak se razlikuje u zavisnosti od strukture stranice sa koje se izvlače Nakon toga, pošto postoje vrednosti u skupovima podataka koje sadrže više entiteta kao npr. u podacima o žanrovima vrednost „Funk/Soul“ ili „Rap/Hip-hop“, kao i u podacima za izvođače „Eminem ft. 50cent“ i sl., potrebno je izvršiti obradu ovakvih vrednosti. Kreirane su funkcije koje vrše ovu obradu i tako dele ovakve vrednosti u zasebne entitete. Zatim je izvršeno uklanjanje duplikata, nakon toga i dodatna obrada i provera kreiranih fajlova korišćenjem regularnih izraza (engl. *regex*) u programu Notepad++. Rezultat ovog prikupljanja i obrade su fajlovi u JSON formatu koji sadrže veliki broj imena izvođača, albuma, žanrova, pesama i instrumenata i koji će biti korišćeni za kreiranje podataka za obuku modela.

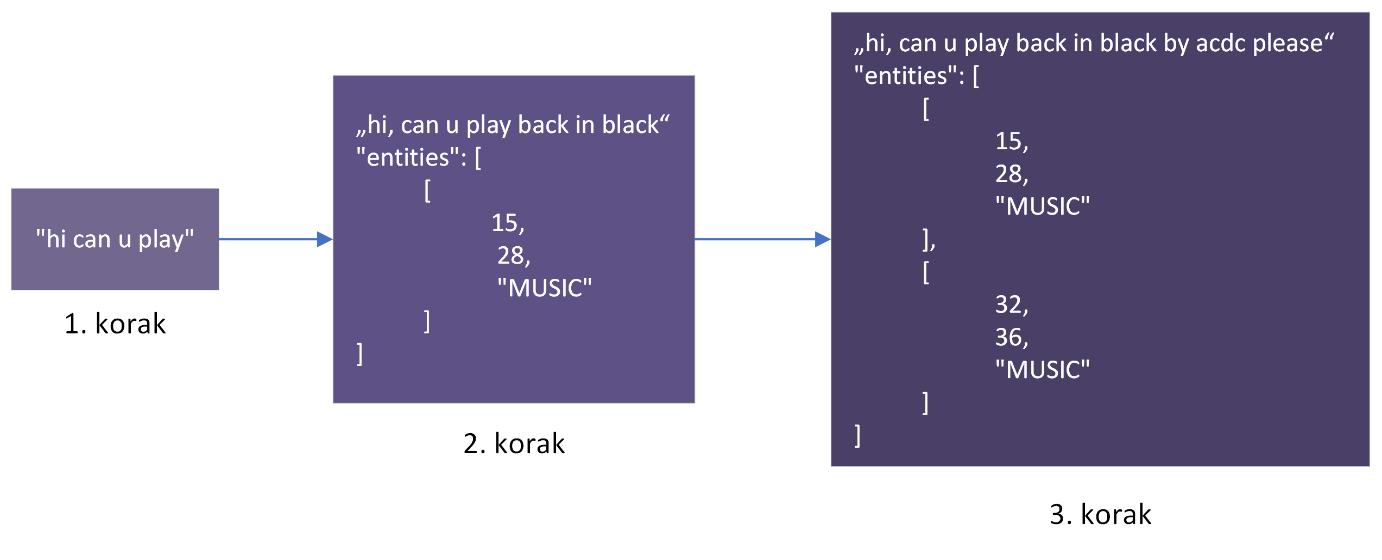


Slika 10. Funkcija koja vrši razdvajanje izvođača



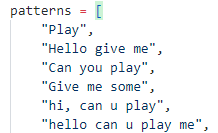
Slika 11. Funkcija koja vrši uklanjanje duplikata

Kako bi model bio obučen da prepozna sve ove entitete u unosu korisnika, potrebno je kreirati konkretne primere tj. rečenice koje predstavljaju različite oblike unosa. U ovu svrhu kreirane su funkcije koje uz pomoću prikupljenih podataka o izvođačima, pesmama i sl. kreiraju podatke koje spaCy može da koristi za obuku modela. Ovi podaci sadrže kreiranu rečenicu i obeležene entitete u rečenici. Za svaki entitet navedena je pozicija prvog i poslednjeg karaktera entiteta, kao i tip entiteta (u ovom slučaju svi entiteti su tipa „music“). Prvo se učitavaju prikupljeni podaci i dele na podatke za obuku i podatke za validaciju. Nakon toga se za svaki od fajlova (izvođači, albumi, pesme itd.), i svaki element iz fajla kreira rečenica uz pomoć predefinisanih šablona. Rečenice se generišu tako što se prvo uzima nasumičan početak rečenice iz liste predefinisanih primera. Zatim se na tekst dodaje entitet za koji se kreira primer i vrši se obeležavanje entiteta tako što se njegov početak, kraj i tip dodaju u listu entiteta za primer koji se generiše. Nakon toga se na osnovu nasumičnih vrednosti dodaje ostatak teksta, a način na koji se ovo obavlja zavisi i od entiteta za koji se generiše primer. Recimo, za naziv pesme na osnovu nasumične vrednosti moguće je dodati naziv izvođača čija je pesma, ili album sa kog pesma potiče. Na ovaj način, u podacima za obuku se neće nalaziti samo primeri kao što su „play song\_name“ već i „play song\_name by artist\_name“ kao i „play song\_name from album\_name“. Ovi entiteti koji se dodaju se takođe označavaju i dodaju u listu entiteta za generisani primer. Reči koje se dodaju između entiteta za ovakve primere, kao što su „by“, „from“, „feat“ i mnoge druge, se takođe uzimaju nasumice iz predefinisane liste u zavisnosti od toga da li se generiše primer za izvođača, album ili pesmu. Proces generisanja rečenice za primer albuma možete videti na sledećoj slici (slika 6).



Slika 12. Proces generisanja rečenice za album

U prvom koraku se uzima nasumičan početak rečenice iz predefinisane liste i dodaje se u tekst koji će biti kreiran.



Slika 13. Deo liste iz koje se uzima nasumičan početak rečenice

U drugom koraku se dodaje ime albuma za koji se generiše primer, označava se njegov početak, kraj i tip entiteta i dodaje se u listu entiteta.

U trećem koraku se na osnovu nasumične vrednosti određuje da li će primer koji se kreira da sadrži samo naziv ovog albuma, ili će biti dodato ime jednog ili više izvođača (isto na osnovu nasumične vrednosti) ili ime neke pesme. Ako se kreira rečenica sa imenom izvođača ili pesme, između tih entiteta se dodaje i nasumična vrednost iz odgovarajuće predefinisane liste, kao što su „by“, „from“, „-“ i sl., ovi entiteti (izvođači, pesme i sl.) se takođe označavaju pozicijom u rečenici i dodaju u listu entiteta. Takođe se na kraju generisane rečenice dodaje i nasumičan sufiks kao npr. „please“. Na ovaj način se kreira veća raznovrsnost rečenica koje će biti korišćene za obuku, pa će model biti obučen da prepozna različite šablone u unosu korisnika i tako prepozna relevantne entitete.

Na sličan način, u zavisnosti od predefinisanih šablona, se kreiraju i primeri za izvođače, žanrove, instrumente i pesme. Na kraju, vrši se mešanje ovih podataka i pretvaranje u spaCy format, i nakon toga podaci su spremni za obuku modela.

Za obuku modela potrebno je kreirati spaCy konfiguracioni fajl, koji se koristi za definisanje raznih parametara za obuku modela. Sadrži razne informacije kao što su lokacija podataka za obuku i validaciju, koje komponente se koriste, arhitektura neuronske mreže koja se kreira i još mnoge druge. Sledeći korak je pokretanje obuke modela korišćenjem spaCy interfejsa komandne linije (engl. command-line interface, skraćeno CLI). Više o kreiranju i podešavanju konfiguracionog fajla, komandama dostupnim u spaCy CLI-u i obuci različitih modela možete pronaći u spaCy dokumentaciji [3].



Slika 14. Format spaCy komande za obuku modela [3]

Parametar „config\_path“ predstavlja putanju do konfiguracionog fajla a „output\_path“ predstavlja putanju do direktorijuma u kojem će model biti sačuvan nakon obuke. Obuka modela će se nastaviti sve dok se ne dostigne maksimalni broj epoha ili koraka, ili dok se ne dostignu određene performanse modela. Kada se obuka završi, model će biti sačuvan u izlaznom direktorijumu koji je naveden u komandi. Flask backend će pri inicijalizaciji učitati ovaj model korišćenjem **spacy.load** funkcije i koristiti za prepoznavanje naziva izvođača, pesama i sl. u korisnikovom unosu.

### Performanse modela

Model za prepoznavanje imenovanih entiteta istreniran korišćenjem kreiranog seta podataka dostiže preciznost od 84%. Na slici 15. su prikazani rezultati treniranja modela, koje generiše spaCy biblioteka.

* ents\_f = f1 score
* ents\_p = precision
* ents\_r = recall



Slika 15. Performanse modela, output koji generiše spaCy biblioteka

# Literatura

[1] “Find open datasets and Machine Learning Projects,” Kaggle, https://www.kaggle.com/datasets (accessed Jun. 9, 2024).

[2] “Lists of musicians,” Wikipedia, https://en.wikipedia.org/wiki/Lists\_of\_musicians (accessed Jun. 9, 2024).

[3] “Training Pipelines & models · Spacy Usage Documentation,” Training Pipelines & Models, https://spacy.io/usage/training (accessed Jun. 9, 2024).

[4] “Natural language processing,” Wikipedia, https://en.wikipedia.org/wiki/Natural\_language\_processing (accessed Jun. 11, 2024).

[5] “Named-entity recognition,” Wikipedia, https://en.wikipedia.org/wiki/Named-entity\_recognition (accessed Jun. 11, 2024).