

Zahvaljujem obitelji, posebice roditeljima, Ani Deškin i prijateljima koji su mi pomogli prebroditi teškoće studiranja te mentoru izv. prof. dr. sc. Ivici Botičkom

# Sadržaj

Uvod .....	1
1. Teorijska pozadina .....	2
1.1. Umjetna inteligencija .....	2
1.2. <i>Chatbot</i> .....	2
1.3. Obrada prirodnog jezika .....	3
1.4. Organizacija web sustava .....	3
1.4.1. Aplikacijska programska sučelja .....	4
2. Korištene tehnologije .....	5
2.1. JavaScript .....	5
2.1.1. Iznimke .....	5
2.1.2. Asinkronost .....	6
2.1.3. Ključne riječi <i>async</i> i <i>await</i> .....	7
2.2. React .....	9
2.3. Node.js .....	10
2.4. Ostale web tehnologije .....	11
2.4.1. Bootstrap .....	11
2.4.2. Axios proxy .....	12
2.5. PostgreSQL .....	12
3. Arhitektura sustava .....	14
3.1. Baza podataka .....	14
3.1.1. ER model baze podataka .....	14
3.1.2. Relacijska shema baze podataka .....	15
3.1.3. Opis tablica baze podataka .....	16
3.2. Dijagram komponenti .....	19

3.3.	Aplikacijska programska sučelja vanjskih usluga .....	20
3.3.1.	OpenAI API .....	20
3.4.	Play.ht API .....	21
4.	Upute za korištenje .....	24
4.1.	Prijava korisnika .....	24
4.2.	Funkcionalnosti administratora sustava .....	26
4.2.1.	Dodavanje novih korisnika .....	26
4.3.	Funkcionalnosti učitelja .....	28
4.3.1.	Kreacija nove osobe .....	28
4.3.2.	Modifikacija postojeće osobe .....	31
4.3.3.	Uređivanje osobe .....	33
4.4.	Funkcionalnosti učenika .....	34
4.4.1.	Tekstualni način .....	34
4.4.2.	Interaktivan način .....	37
4.4.3.	Način konverzacije dvije osobe .....	39
5.	Izazovi tijekom izrade sustava .....	43
5.1.	Izazovi oblikovanja upita .....	43
5.2.	Izazovi oblikovanja sustava .....	44
	Zaključak .....	46
	Literatura .....	47
	Sažetak .....	48
	Abstract .....	49

# Uvod

U današnjem smo vremenu svjedoci ubrzanog razvoja umjetne inteligencije, jezičnih modela i, posljedično, raznih generativnih alata kao što su generatori slika, zvuka, govora ili teksta. Kao daleko najpopularniji alat ističe se ChatGPT, jezični model tvrtke OpenAI, koji je dobio veliku medijsku pozornost i popularnost među širom javnosti zbog svojih mogućnosti i stila odgovora koji izrazito nalikuje ljudskom.

Kao i sa svakom novom tehnologijom, tako i s ChatGPT-em, ljudi nastoje pronaći primjenu koja će im olakšati svakodnevne poslove ili ih potpuno automatizirati. OpenAI je, baš za tu svrhu, ChatGPT, odnosno, preciznije, GPT-3.5 model koji se nalazi iza ChatGPT-a, stavio na raspolaganje programerima preko posebnog aplikacijskog programskog sučelja (engl. *application programming interface*, API) [1] [2].

No, osim olakšavanja nekih poslova, ova nam tehnologija omogućuje puno više, pogotovo u području računarstva gdje umjetna inteligencija, a tako i jezični modeli OpenAI-a, imaju širok raspon primjena i načina na koje se mogu koristiti i ugraditi u sustave. Jedno od važnijih područja primjene te tehnologije je obrazovanje. Tehnologija se u obrazovanju može iskoristiti na razne načine, npr. zadatak pronalaska pojedinih informacija, no potencijalno najbolja primjena bi bila u izradi *chatbota* koji bi učenicima omogućavao interaktivno učenje kroz razgovor s osobama relevantnim za pojedino područje.

Cilj je ovog rada bio izrada aplikacije koja bi omogućila upravo to: čavljanje (engl. *chat*) s pojedinim osobama. Aplikacija podržava i interaktivnu komunikaciju u smislu sinteze govora u tekst (engl. *text to speech*) i teksta u govor (engl. *speech to text*) i funkcionalnost čavljanja dvije osobe čije poruke generira isključivo GPT-3.5 model.

# 1. Teorijska pozadina

## 1.1. Umjetna inteligencija

Umjetna se inteligencija definira kao dio računarske znanosti koja se bavi razvojem i izučavanjem sustava koji pokazuju neki oblik inteligencije te se ti sustavi koriste za efikasnije obavljanje poslova u domeni računarstva i izvan nje [3]. Dok definicije i kriteriji same računalne inteligencije među ljudima i područjima nisu ujednačeni, generalno je u javnosti prihvaćena činjenica da se računalni sustav smatra inteligentnim kad pokazuje neka svojstva koja inače pokazuju ljudi, npr. prepoznavanje govora, fotografija ili sinteza teksta [3]. Postoji puno različitih sustava umjetne inteligencije koji se razlikuju po svojim funkcijama, a neki od njih su:

- ekspertni sustavi
- sustavi za obradu prirodnog jezika
- sustavi za prepoznavanje govora
- sustavi za računalni vid
- *chatboti*.

## 1.2. Chatbot

*Chatbot* je računalni sustav ili program koji koristi umjetnu inteligenciju i obradu prirodnog jezika kako bi razumio upite korisnika i kreirao odgovor koji u najvećoj mogućoj mjeri odgovara na upit i sliči ljudskom prirodnom jeziku [4]. Osim same mogućnosti obrade prirodnog jezika, *chatboti* moraju imati i bazu znanja na temelju koje će uzimati elemente jezika (riječi, rečenice) iz kojih će kreirati odgovore. Najpoznatiji primjeri chatbota su:

- ChatGPT

- Siri
- Cortana
- LaMDA.

Rapidnim razvojem umjetne inteligencije *chatboti* postaju sve bolji i sintetiziraju tekst sve sličniji ljudskom govoru. Povećanjem računalnih resursa koji stoje iza tih sustava se u model sustava može staviti sve više i više podataka pa tako dobiti i tekst koji, osim izuzetne sličnosti ljudskom govoru, posjeduje i iznimnu točnost podataka koje koristi u generiranom tekstu. Nadalje, kako bi konverzacija između sustava i korisnika bila što realnija, *chatboti* imaju sposobnost i nastavljavanja razgovora ili postavljanja pitanja korisniku te predlaganja potencijalnih pitanja koja korisnik može postaviti sustavu. Sve te značajke čine ovaj tip sustava umjetne inteligencije izuzetno moćnim alatom, a pogotovo za programere koji ih korištenjem programskih sučelja mogu implementirati u svoje sustave.

### 1.3. Obrada prirodnog jezika

Obrada prirodnog jezika (engl. *natural language processing*, NLP) je sposobnost pojedinog sustava ili programa da razumije ljudski jezik u pisanom ili izgovorenom obliku. Obrada prirodnog jezika se koristi u raznim područjima računarske znanosti i sustavima [5]. Obrada prirodnog jezika svoje korijene vuče iz područja lingvistike, a u samoj implementaciji u računalnim sustavima koristi se umjetna inteligencija. Neke od primjena sustava obrade prirodnog jezika su:

- Email filteri
- Tražilice (engl. *search engines*)
- Predviđanje teksta
- Analiza teksta.

### 1.4. Organizacija web sustava

Web aplikacija je interaktivan program kojem korisnik pristupa preko internetske veze. Aplikacija se nalazi na nekom jedinstvenom URL-u (*Uniform Resource Locator*). Web aplikacije imaju izuzetno široko područje primjena, sve od aplikacija

za čitanje i slanje e-mailova, internih stranica neke tvrtke ili ustanove pa do zabavih sadržaja ili igara. Web aplikacije se često sastoje od nekoliko različitih dijelova, a to su tipično:

- Klijentski dio (engl. *clientside, frontend*)
- Poslužiteljski dio (engl. *server, serverside, backend*)
- Baza podataka (engl. *database*)

Danas su web aplikacije široko raširene i vrlo popularne zbog činjenice da im se lako može pristupiti i da ima iznenađujuće velik broj programskih jezika i njihovih radnih okvira (eng. *workframe*) u pomoću kojih se aplikacija može izraditi. Zbog široke dostupnosti dokumentacije i vodiča za pojedine tehnologije, moguće je raditi s puno različitih tehnologija.

#### **1.4.1. Aplikacijska programska sučelja**

Aplikacijsko programsko sučelje je skup definiranih rutina koje omogućavaju korisnicima pristup funkcionalnostima koje određeno sučelje nudi. Ovakva sučelja omogućavaju nezavisnost i odvajanje dijelova sustava, npr. klijentskog i poslužiteljskog dijela web aplikacije. Ta nam nezavisnost omogućuje da dijelove sustava pišemo u različitim programskim jezicima ili koristeći druge tehnologije i radne okvire, a da oni svejedno mogu komunicirati i razmjenjivati podatke.

Kao u navedenom primjeru, aplikacijska programska sučelja mogu se koristiti za povezivanje različitih dijelova sustava, ali i za korištenje nekih vanjskih usluga, kao što je u ovom radu korišteno sučelje za pristup uslugama OpenAI-evih jezičnih modela i sučelje za sintezu govora iz teksta.

## 2. Korištene tehnologije

### 2.1. JavaScript

JavaScript<sup>1</sup> je skriptni programski jezik koji je najzastupljeniji u domeni web programiranja i izrade web aplikacija. On se parsira i odmah izvodi. To je proceduralan jezik koji podržava objektno orijentiranu paradigmu i najčešće se izvodi u samom pregledniku kod pokretanja web aplikacija. Danas je JavaScript jedan od najpopularnijih i najraširenijih programskih jezika koji se koristi i na klijentskoj i poslužiteljskoj strani razvoja web aplikacija. Zajednica programera koja koristi JavaScript je velika, stoga postoji i puno različitih radnih okvira za izradu aplikacija.

JavaScript kompleksan jezik s puno specifičnih aspekata koji su bitni programerima web aplikacija, ali njegove specifičnosti i karakteristike nisu fokus ovog rada, stoga će biti opisani detalji koji su bili bitni za realizaciju izrađene web aplikacije.

#### 2.1.1. Iznimke

Iznimke (engl. *Exception*) su događaji koji se pojavljuju kod izvođenja kôda koji ometaju izvođenje programa. Iznimke mogu nastati na mnogo načina, npr. greškom programera, neslaganjem tipova podataka pa čak i nedostatkom memorije.

U JavaScriptu se iznimkama upravlja, slično kao i u drugim programskim jezicima, posebnim blokovima koji imaju nazive *try*, *catch* i *finally*.

U *try* blok se stavlja kôd za koji postoji mogućnost da generira iznimku. U trenutku generiranja iznimke, ovaj blok prestaje s izvođenjem.

U *catch* blok se piše kôd koji se treba izvršiti ako dođe do iznimke. Ovaj blok prima i argument (kao funkcija) objekta za grešku tipa *Error* koji sadrži korisne

---

<sup>1</sup> <https://developer.mozilla.org/en-US/docs/Web/JavaScript>



informacije za obradu i analizu greške, npr. vrsta i poruka greške. Takvi se objekti mogu i stvarati u kôdu.

*Finally* blok sadrži kôd koji se izvršava u svakom slučaju, dogodila se iznimka ili ne.

```
async function fillData(){
  try {
    let response = await backend.get("/getvoice/voices");
    setVoices(response.data.map(voice => ({
      value : voice.name, language : voice.language, sample
      :voice.sample, gender : voice.gender
    })));
    let categories = await backend.get("/categories");
    setCategories(categories.data.map(c => ({value : c.id, name :
    c.name})))
  } catch (e) {
    alert(e);
  } finally {
    console.log("Data fetch done.")
  }
}
```

Kôd 2.1 Primjer dohvaćanja podataka s try, catch i finally blokovima

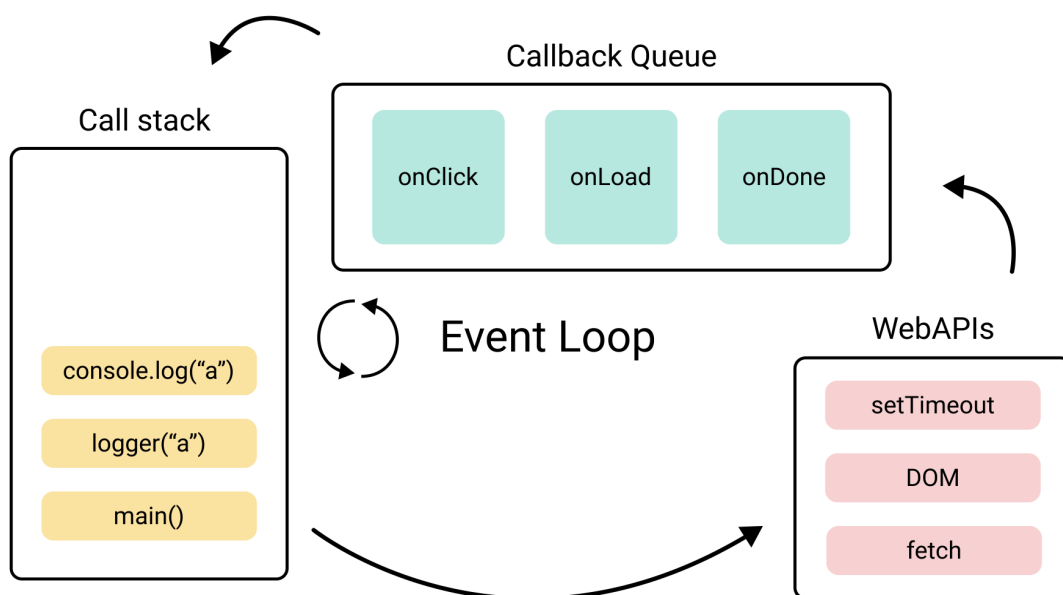
### 2.1.2. Asinkronost

JavaScript je izvorno osmišljen kao jednodretven jezik u smislu da ne postoji opcija da se kreira nova dretva u pojedinom procesu. Naredbe se izvršavaju slijedno, sinkrono [6]. Četo je na webu potrebno dohvaćati resurse, ažurirati DOM (*Document Object Model*) ili čekati neki događaj na DOM-u, npr. klik gumba, što bi blokiralo izvođenje danjih naredbi i tako učinilo kôd sporim.

Kao rješenje, JavaScript koristi petlju događaja (engl. *event loop*), red čekanja poruka (engl. *message queue*) i *web APIs*. Sve se naredbe spremne za izvršavanje stavljaju na sistemski stog. Ako se naredba treba izvršiti asinkrono, ona se preusmjerava na *web APIs*. On preuzima naredbu i izvršava ju. Nakon njenog izvršavanja se njena povratna funkcija (engl. *callback*) stavlja u red čekanja poruka. Povratna funkcija je funkcija koja sadrži kôd koji želimo da se izvrši nakon

uspješnog izvršavanja asinkrone naredbe. Petlja događaja konstantno provjerava postoji li u redu čekanja povratna funkcija koju treba izvršiti te ju stavlja na sistemski stog s kojeg se ta povratna funkcija izvodi sinkrono [6]. Dakle, iako ne postoji višedretvenost kao takva, dio posla koji treba odraditi asinkrono može se prebaciti na *web APIs*.

Asinkrono izvođenje može se ostvariti korištenjem obećanja (engl. *Promises*) ili ključnih riječi *async* i *await*.



Slika 2.1 Prikaz asinkronog izvođenja [7]

### 2.1.3. Ključne riječi *async* i *await*

*Async* i *await* su ključne riječi programskog jezika JavaScript koje koristimo kako bismo efektivno mogli izvoditi asinkroni kôd sinkrono, slijedno. U ovom su radu ove ključne riječi često korištene jer se dohvaća puno podataka i čeka na puno odgovora s raznih sučelja.

Označavanje funkcije ključnom riječi *async* znači da će se u funkciji izvoditi neki asinkroni kôd.

Ključna riječ *await* može se koristiti isključivo unutar funkcije označene s *async*. Njome se čeka izvršavanje pojedine asinkrone operacije, ta se operacija u kontekstu asinkrone funkcije izvršava sinkrono, dakle čeka se izvršavanje

asinkrone naredbe kako bi se moglo dalje nastaviti s izvršavanjem naredbi funkcije.

Ove su ključne riječi ravnopravne obećanjima, ali njihovim korištenjem dobiva se čitljiviji i jednostavniji kôd.

```
async function getAudio(){
    const request = {
        "voice": personaObj.voice,
        "content": [props.data],
    };
    let response
    try {
        response = await backend.post("/getvoice", request)
    }
    catch(e){
        alert("An error has occurred while fetching voices!")
        return
    }
    isSpeaking = true
    let a = new Audio(response.data)
    a.onended = function(e){
        clearInterval(interval);
        isSpeaking = false
        speakIdx = 0
        setMouth(defaultMouth)
    }
    await a.play()
    interval = setInterval(() => {
        handleAnimation();
    }, 60);
}
```

Kôd 2.2 Primjer async funkcije

„Kôd 2.2 Primjer async funkcije“ pokazuje kako se ključne riječi mogu koristiti za dohvat podataka i reprodukciju glasa.

## 2.2. React

React<sup>2</sup> je besplatna JavaScript knjižnica (engl. *library*) otvorenog kôda (engl. *open source*) koja se koristi za izradu klijentskog dijela web aplikacije. Danas je React jedna od najpopularnijih i najkorištenijih knjižnica za izradu web aplikacija. React se bazira na komponentama (engl. *components*) koje se mogu više puta koristiti kod izrade pa ubrzavaju vrijeme izrade aplikacije i čine kôd čitljivijim. Osim standardnih web aplikacija, React se često upotrebljava za izradu jednostraničnih web aplikacija (engl. *single-page web application*), ali i mobilnih aplikacija.

React koristi JSX (*JavaScript Syntax Extension*), ekstenziju JavaScript-a koja omogućava pisanje HTML-a (*HyperText Markup Language*) u Reactovim komponentama. Uz to, koristi i tzv. *Hooks* i stanja (engl. *states*) kako bi ostvario željene funkcionalnosti te mogao pratiti vrijednosti pojedinih varijabli i na temelju njih ažurirati osnovni HTML stranice.

React koristi virtualni HTML dokument koji se mijenja, a onda se samo promjene prikazuju korisniku. Ne mora ažurirati prikaz cijele stranice, već samo onaj dio stranice koji se promijenio.

```
function ProfileAvatar(props) {  
  return (  
    <img src={props.image} alt={props.image} className = "avatar"/>  
  );  
}  
export default ProfileAvatar;
```

Kôd 2.3 Jednostavan primjer React kôda

---

<sup>2</sup> <https://react.dev/>

## 2.3. Node.js

Node.js<sup>3</sup> je poslužiteljsko okruženje otvorenog kôda za JavaScript koje se može pokrenuti na raznim operacijskim sustavima. U ovom se okruženju često izrađuju poslužiteljske strane web aplikacija.

U Node.js uključen je i npm<sup>4</sup> (*Node Packet Manager*). To je upravitelj paketa za Java Script koji omogućava preuzimanje paketa. Paketi su različite kompleksnosti, od jednostavnih paketa koji provjeravaju je li broj neparan pa sve do samog Reacta. U ovom su radu korišteni razni paketi i na klijentskoj i poslužiteljskoj strani.

Jedan od korištenih paketa je i *express*<sup>5</sup> koji je korišten za izradu poslužiteljskog dijela web aplikacije. Fleksibilnost i robusnost te lakoća korištenja glavne su karakteristike ovog radnog okvira. U sklopu *servera* korišten je i *express-validator*<sup>6</sup> koji predstavlja posrednički softver (engl. *middleware*) koji služi za validaciju i sanitizaciju podataka koji se primaju na poslužitelj.

```
app.use('/edit', editRouter)
app.use('/log', logsRouter)
const PORT = 5300;
app.listen(PORT, () => {
  console.log(`Server listening on ${PORT}`);
});
```

Kôd 2.4 Isječak poslužiteljskog kôda u expressu

---

<sup>3</sup> <https://nodejs.org/en>

<sup>4</sup> <https://www.npmjs.com/>

<sup>5</sup> <https://expressjs.com/>

<sup>6</sup> <https://www.npmjs.com/package/express-validator>

## 2.4. Ostale web tehnologije

Osim navedenih web tehnologija koje čine glavninu sustava i koje su bitne za njegov rad, korištene su i druge tehnologije koje su pomogle u obavljanju poslova kao što su stilizacija stranice.

Standardno za izradu web stranice, korišteni su i HTML i CSS (*Cascading Style Sheets*), HTML u velikoj mjeri u sklopu JSX kôda, a CSS kod izrade nekih specifičnih stilova elemenata na sučelju.

### 2.4.1. Bootstrap

Bootstrap<sup>7</sup> je radni okvir za razvoj korisničkog sučelja sadržaja na webu koji sadrži predloške (engl. *templates*) za uređivanje kontejnera, gumba, obrazaca i ostalih HTML elemenata. Bootstrapovi su predlošci oblikovani kao klase te su responzivni i laki za čitanje, što znači da je lako iz samog naziva klase zaključiti kakve ona modifikacije primjenjuje na element koji je njom označen. Trenutno je u upotrebi verzija 5, koja je izašla 2021. godine.

```
<label className={ (props.name === radio ? "btn-light" : "btn-outline
dark") + " btn mx-3 radio-button"} htmlFor={props.name}>
  <div className='container-fluid d-flex flex-column align-
items-center justify-content-center flex-wrap'>
    <h6>{props.name}</h6>
    <div><img className='radio-img' src={props.image}
alt={props.name}></img></div>
  </div>
</label>
```

Kôd 2.5 Primjer JSX kôda s Bootstrap klasama

---

<sup>7</sup> <https://getbootstrap.com/>

### 2.4.2. Axios proxy

Axios<sup>8</sup> je knjižnica koja se ponaša kao posrednik kod slanja HTTP (*Hypertext Transfer Protocol*) zahtjeva. Preuzima se i instalira kao npm paket. Korištenjem Axiosa se mogu definirati specifični parametri zahtjeva i mogu se ponovo koristiti.

```
const {API_KEY} = require('../data/voice_api_key.js')
const {USER} = require('../data/voice_user.js')
let voiceRequest = axios.create({
  headers: {
    "Authorization": API_KEY,
    "X-User-ID": USER,
    'Content-Type': 'application/json'
  }
});
```

Kôd 2.6 Definiranje instance Axiosa

```
response = await voiceRequest.get("https://play.ht/api/v1/getVoices")
```

Kôd 2.7 Slanje zahtjeva preko definirane instance Axiosa

## 2.5. PostgreSQL

PostgreSQL<sup>9</sup> je sustav za upravljanje bazama podataka koji upravlja relacijskim bazama podataka i koristi jezik SQL. Sustav koristi transakcije sa svojstvima atomarnosti, konzistencije, izolacije i izdržljivosti.

---

<sup>8</sup> <https://axios-http.com/docs/intro>

<sup>9</sup> <https://www.postgresql.org/>

PersonaApp/projektadmin@ZavRad				
Query Editor   Query History				
1 <b>SELECT</b> * <b>FROM</b> Logs				
Data Output   Explain   Messages   Notifications				
	logid	time	type	data
	[PK] integer	timestamp without time zone	character varying (20)	character varying (30000)
1	1	2023-05-18 12:33:54.733273	login	"User has logged in."
2	2	2023-05-18 12:34:49.323024	user_created	{"user_created":{"username":"testuser","password":"testuser123","roleid":3}}
3	3	2023-05-18 12:34:53.971684	logout	"User has logged out using the logout button."
4	4	2023-05-18 12:36:45.316028	login	"User has logged in."
5	5	2023-05-18 12:36:48.477003	conversation	{"persona":{"id":3,"name":"Lionel Messi","gender":"male","imageid":99,"initial":
6	6	2023-05-18 12:36:48.512077	conversation	{"persona":{"id":3,"name":"Lionel Messi","gender":"male","imageid":99,"initial":
7	7	2023-05-18 12:37:15.334521	conversation	{"persona":{"id":3,"name":"Lionel Messi","gender":"male","imageid":99,"initial":
8	8	2023-05-18 12:37:30.228871	login	"User has logged in."
9	9	2023-05-18 12:45:19.347481	2_persona_talk	{"topic":"Cars","persona1":{"id":3,"name":"Lionel Messi","gender":"male","imageid":99,"initialprompt":"

Slika 2.2 Prikaz PostgreSQL u aplikaciji pgAdmin



## 3. Arhitektura sustava

### 3.1. Baza podataka

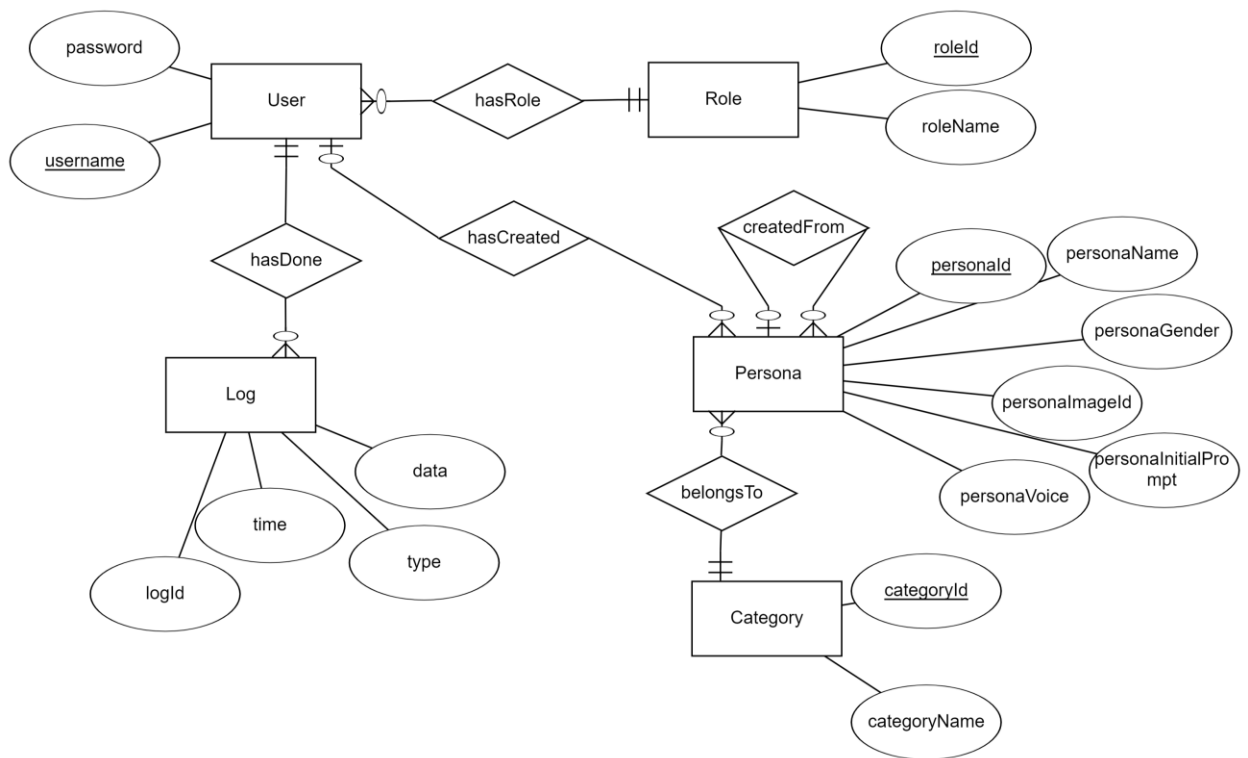
Baza podataka je za ovaj sustav relativno jednostavna. Njena temeljna svrha je čuvanje informacija i podataka o osobama (u kôdu i dijagramima označeno engl. *persona*) koje sustav treba imitirati. Korištena je relacijska baza gdje svaka relacija predstavlja neki entitet iz stvarnog života. Entiteti su međusobno povezani vezama različitih spojnosti.

#### 3.1.1. ER model baze podataka

Baza podataka sadrži sljedeće entitete:

- *User*
- *Role*
- *Log*
- *Persona*
- *Category*.

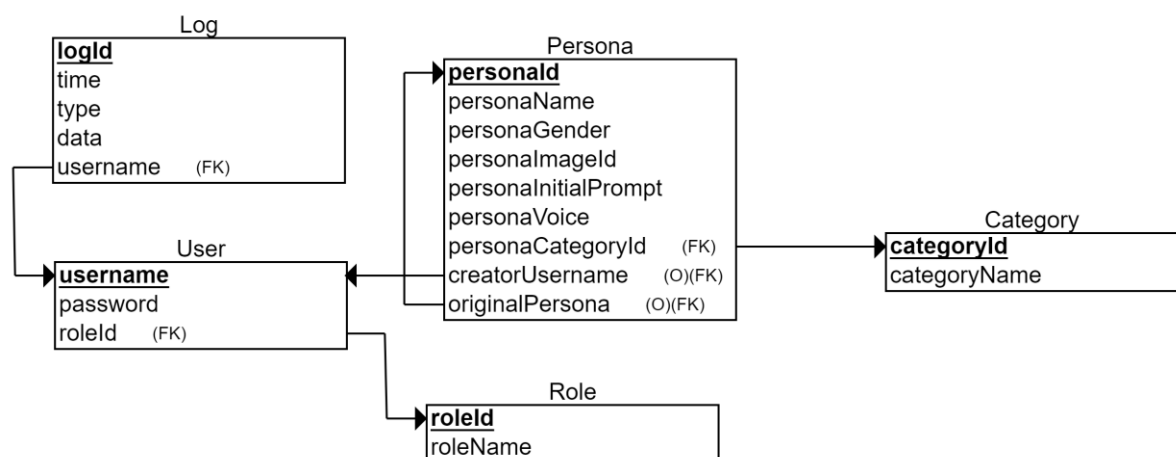
Entitet *User* predstavlja korisnika te je povezan s entitetima *Role* (koji označava jednu od uloga koje pojedini korisnik može imati), *Log* (koji modelira neku radnju koju je korisnik obavio) te *Persona* (veza se temelji na tome da korisnici mogu kreirati nove osobe te se u zapisu te osobe nalazi informacija tko ju je stvorio). *Persona* je središnji entitet baze podataka ovog sustava i modelira pojedinu osobu koju sustav može oponašati. Taj je entitet refleksivno vezan sam za sebe s obzirom na to da jedna osoba može biti napravljena na temelju druge. *Persona* pripada pojedinoj kategoriji koja je modelirana zasebnim entitetom *Category* i stoga je povezana s tim entitetom.



Slika 3.1 ER model baze podataka

### 3.1.2. Relacijska shema baze podataka

Relacijska shema baze podataka temelji na ER modelu. Entiteti se pretvaraju u relacije, a veze se modeliraju dodavanjem atributa u relacije. Logika i razlozi za povezanost pojedinih relacija ostali su isti kao i za veze između entiteta u ER modelu.



Slika 3.2 Relacijska shema baze podataka

### 3.1.3. Opis tablica baze podataka

Primarni će ključevi biti označeni zlatnom pozadinskom bojom imena atributa, a atributi koji se referenciraju na drugi entitet, odnosno strani ključevi svijetloplavom pozadinskom bojom naziva atributa.

#### 3.1.3.1 User

Entitet *User* sadrži sve potrebne informacije o korisnicima sustava. Sadrži attribute *username*, *password* i *roleId*. Ovaj je entitet u vezi *One-to-many* s entitetom *Role* preko *roleId*, u vezi *Many-to-one* s entitetom *Log* preko *username* te u vezi *Many-to-one* s entitetom *Persona* preko *creatorUsername*.

Tablica 3.1 Entitet *User*

User		
username	VARCHAR	jedinstveno korisničko ime korisnika
password	VARCHAR	lozinka korisnika
roleId	INT	razina prava korisnika, njegova uloga ( <i>Role.roleId</i> )

#### 3.1.3.2 Role

Entitet *Role* sadrži informacije o ulogama korisnika sustava. Sadrži attribute *roleId* i *roleName*. Ovaj je entitet u vezi *Many-to-one* s entitetom *User* preko *roleId*.

Tablica 3.2 Entitet *Role*

Role		
roleId	INT	jedinstven identifikator uloge
roleName	VARCHAR	Ime pojedine uloge

### 3.1.3.3 Log

Entitet *Log* sadrži informacije o aktivnostima u sustavu. Postavljanje upita osobi, prijava, odjava i slične aktivnosti bit će zapisane u ovoj tablici. Sadrži attribute *logId*, *time*, *type*, *data* i *username password*. Ovaj je entitet u vezi *One-to-many* s entitetom *User* preko *username*.

Tablica 3.3 Entitet *Log*

Log		
logId	INT	jedinstven identifikator <i>loga</i>
time	TIMESTAMP	trenutak u kojem je <i>log</i> napravljen ili zadnje izmijenjen
type	VARCHAR	tip <i>loga</i> (prijava, odjava, razgovor...)
data	VARCHAR	podatci o aktivnosti koja se izvršila (JSON razgovora, kreiran osoba,...)
username	VARCHAR	Korisnik koji je izvršio radnju zapisanu u <i>log</i> ( <i>User.username</i> )

### 3.1.3.4 Persona

Entitet *Persona* sadrži karakteristike osoba koje sustav oponaša. Sve su potrebne informacije za postavljanje karakteristika osobe te njene slike i glasa u ovoj tablici. Sadrži attribute *personald*, *personaName*, *personaGender*, *personalmageld*, *personaInitialPrompt*, *personaVoice*, *personaCategoryId*, *creatorUsername* i *originalPersona*. Ovaj je entitet u vezi *One-to-many* s entitetom *Persona* preko *creatorUsername*, u vezi *One-to-many* s entitetom *Category* preko atributa *personaCategoryId* te u *Many-to-one* vezi sa samim sobom preko atributa *originalPersona* koji referencira *personald* neke postojeće osobe. *creatorUsername* i *originalPersona* su opcionalni atributi.

Tablica 3.4 Entitet *Persona*

<b>Persona</b>		
personaId	INT	jedinstven identifikator osobe
personaName	VARCHAR	ime ili naziv osobe
personaGender	VARCHAR	spol osobe, može biti „ <i>male</i> “ ili „ <i>female</i> “
personalmageld	INT	identifikator avatara osobe
personalInitialPrompt	VARCHAR	inicijalan upit kojim se dobiva željeno ponašanje modela, postavljaju se karakteristike željene osobe
personaVoice	VARCHAR	naziv glasa koji se koristi u interaktivnom načinu rada
personaCategoryId	INT	identifikator kategorije kojoj osoba pripada
creatorUsername	VARCHAR	<i>username</i> korisnika koji je kreirao osobu ili prazno polje ako je osoba dodana u bazu od strane admina
username	VARCHAR	Korisnik koji je izvršio radnju zapisanu u <i>log (User.username)</i>

### 3.1.3.5 Category

Entitet *Category* sadrži informacije o kategorijama osoba. Sadrži attribute *categoryId* i *categoryName*. Ovaj je entitet u vezi *Many-to-one* s entitetom *Persona* preko *categoryId*.

Tablica 3.5 Entitet *Category*

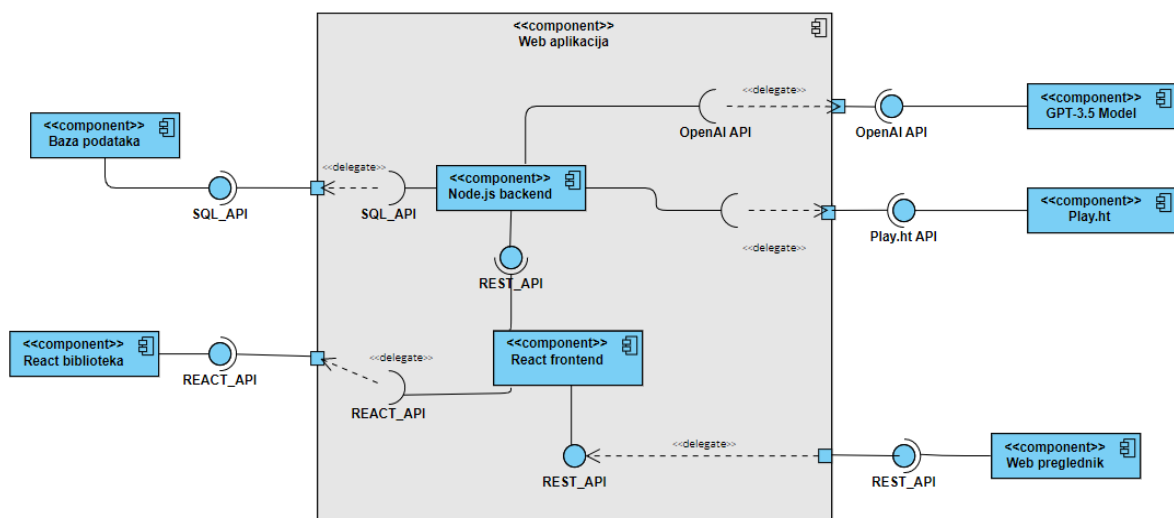
Category		
categoryId	INT	jedinstven identifikator kategorije
categoryName	VARCHAR	Ime pojedine kategorije

## 3.2. Dijagram komponenti

Web aplikacija se sastoji od dva dijela, klijentskog i poslužiteljskog. Klijentski dio šalje zahtjeve na poslužiteljski preko sučelja koje mu pruža te na taj način ta dva dijela aplikacije razmjenjuju podatke.

Klijentski dio koristi React biblioteku koja mu pruža sučelje, a pruža sučelje Web pregledniku kako bi on mogao korisniku prikazati elemente aplikacije.

Poslužiteljski dio koristi programsko sučelje prema bazi podataka u koju pohranjuje sve podatke (3.1) koji omogućavaju rad aplikacije. Ovaj dio aplikacije također koristi sučelja prema dva vanjska servisa koji su opisani u 3.3.



Slika 3.3 Dijagram komponenti

### 3.3. Aplikacijska programska sučelja vanjskih usluga

Aplikacijska programska sučelja vanjskih usluga korištena su pri izradi ovog sustava sa svrhom realnih odgovora pojedinih osoba i sa svrhom omogućavanja interaktivnosti kreacijom zvučnih zapisa za pojedine poruke.

Oba u nastavku opisana sučelja koriste tzv. *API token* za autentifikaciju korisnika.

#### 3.3.1. OpenAI API

OpenAI API [8] se u kontekstu ovog sustava koristi za generiranje odgovora pojedinih osoba u pojedinim kontekstima. Ima nekoliko korisnih funkcija kao što su generiranje teksta, generiranje slika i sinteza teksta iz govora. Ovo aplikacijsko programsko sučelje sadrži nekoliko modela kojima se može pristupiti, uključujući GPT-4, GPT-3.5, DALL-E i Whisper.[1]. Konkretno, ovdje se koristi model GPT 3.5, model iza popularnog alata ChatGPT. Također, postoji i *openai* npm paket koji se u kodu koristi za olakšani pristup funkcijama sučelja kreiranjem instance API-ja *OpenAIApi* i postavljanjem zadane konfiguracije.

Iako ovo aplikacijsko programsko sučelje sadrži mnoge URL-ove na koje se zahtjevi mogu slati, u implementaciji sustava se zahtjevi šalju samo na <https://api.openai.com/v1/chat/completions> [8]. Slanje zahtjeva na ovaj URL omogućava generiranje odgovora na poruke priložene u zahtjevu, što znači da generira odgovor na temelju konteksta trenutnog razgovora. Bez te funkcionalnosti ne bi bilo moguće definirati karakteristike pojedinih osoba i da one ostanu pohranjene, tj. da ih model uzme u obzir. Postoji i nedostatak: mora se svaki put slati cijeli razgovor kao kontekst, ali u praksi nije predstavljalo probleme s brzinom odgovora.

```
const { Configuration, OpenAIApi } = require("openai");
const model = "gpt-3.5-turbo"
const {OPENAI_API_KEY} = require('../data/openai_api_key')
const configuration = new Configuration({ apiKey: OPENAI_API_KEY, });
const openai = new OpenAIApi(configuration);
router.post("/", async (req, res) => {
```

```

chatData = req.body
console.log(chatData)
try {
  const response = await openai.createChatCompletion({
    model: model,
    messages: chatData,
    temperature : 0.8,
    presence_penalty : 1.3,
  })
  if(response.status == 429){
    console.log(response)
    res.status(429)
    res.send(response)
  }
  res.send(response.data.choices[0].message);
}
catch (e) {
  res.status(400)
  console.log(e)
  res.send()
}
});

```

Kôd 3.1 Primjer slanja zahtjeva na OpenAI API pomoću npm paketa

### 3.4. Play.ht API

Play.ht<sup>10</sup> je internetska usluga koja omogućava sintezu govora iz teksta na više različitih jezika što je bilo potrebno za implementaciju i omogućavanje interaktivne komunikacije s pojedinim osobama. Ovo aplikacijsko programsko sučelje je jednostavno za korištenje: šalje se zahtjev za generiranjem audio zapisa te se onda šalje zahtjev za dohvat tog audio zapisa. Nedostatak korištenja sučelja je vrijeme koje je potrebno da se zapis kreira za dulje odgovore osoba. Iz tog se razloga ne koriste vanjske usluge za drugi smjer, sintezu teksta iz govora, već

---

<sup>10</sup> <https://play.ht/>



npm paket react-speech-kit<sup>11</sup> koji koristi sustav za prepoznavanje govora ugrađen u pregledniku koji se koristi.

```
router.get("/voices", async (req, res) => {
  try {
    response = await voiceRequest.get("https://play.ht/api/v1/getVoices")
    let returnArray = response.data.voices.filter((v) =>
      v.voiceType === "Standard"
    )
    res.send(returnArray)
  }
  catch {
    res.status(400)
    res.send()
  }
})
```

Kôd 3.2 Primjer dohvata mogućih glasova tipa *Standard* s Play.ht API-ja

```
router.post("/", async (req, res) => {
  try {
    const sendData = {
      "voice": req.body.voice,
      "content": req.body.content,
    }
    console.log(sendData)
    let response
    try {
      response = await
voiceRequest.post("https://play.ht/api/v1/convert", sendData)
    } catch (error) {
      console.error(error);
    }
    let audio = {
      "converted" : false
    }
    console.log(response)
```

---

<sup>11</sup> <https://www.npmjs.com/package/react-speech-kit>

```

    console.log(response.data)
    let transcriptionId = response.data.transcriptionId
    do {
        response = await
voiceRequest.get("https://play.ht/api/v1/articleStatus?transcriptionId="
+ transcriptionId )
        audio = response.data
    } while (!audio.converted)
    res.send(response.data.audioUrl)
}
catch (e) {
    res.status(429)
    res.send()
}
});

```

Kôd 3.3 Primjer slanja zahtjeva za kreacijom audio zapisa i preuzimanja istog

U Kôd 3.3 generira se i šalje zahtjev za generiranjem audio zapisa za pojedini tekst te se u petlji ispituje u *do-while* petlji je li audio zapis spreman za preuzimanje te ako je, preuzima se i šalje na klijentsku stranu gdje se taj zapis i reproducira.

## **4. Upute za korištenje**

Korisnici sustava podijeljeni su u 3 uloge: administrator, učitelj i učenik. Funkcionalnosti sustava podijeljene su između te 3 uloge po principu razdvajanja zaduženja.

### **4.1. Prijava korisnika**

Prijava korisnika jedina je zajednička funkcionalnost korisnika. Prilikom inicijalnog pristupa sustavu, dok korisnik još nije autentificiran, otvara se obrazac za prijavu. Korisnik unosi svoje jedinstveno korisničko ime i lozinku. Ako je lozinka ispravna za tog korisnika, sustav ga preusmjerava na neku od stranica kojoj ima pristup korisnik s tom ulogom, a ako nije sustav ispisuje obavijest. Također postoji mogućnost prikaza upisane lozinke.

Nakon uspješne prijave u zaglavlju aplikacije se na sredini prikazuju gumbi koji vode do svih funkcionalnosti korisnika te na desnoj strani zaglavlja gumb za odjavu te informacija o tome koji je korisnik prijavljen i koja je njegova dodijeljena uloga u sustavu.

# Persona App

## Login

Username

Password

☐ Show password

Login

Slika 4.1 Obrazac za prijavu korisnika

# Persona App

## Login

Username or password incorrect!

Username

Password

☐ Show password

Login

Slika 4.2 Poruka o neispravnom korisničkom imenu ili lozinki

### Login

Username

User123

Password

somepassword123

☒ Show password

Login

Slika 4.3 Mogućnost prikaza upisane lozinke

## 4.2. Funkcionalnosti administratora sustava

### 4.2.1. Dodavanje novih korisnika

Administrator sustava ima mogućnost dodjele novih korisnika kroz poseban obrazac za to. To je jedina funkcionalnost administratora. Administrator kod kreacije korisnika upisuje njegovo korisničko ime, lozinku, ponavlja lozinku te bira kategoriju korisnika. Ukoliko je neko od polja prazno ili se lozinke ne podudaraju, aplikacija ispisuje prikladnu obavijest. Lozinke također moraju biti minimalno 8 znakova duge. Gumb *Submit* potvrđuje podatke i šalje ih na provjeru na poslužiteljski dio ako su prošli provjeru na klijentskom, a *Reset* briše sve upisane vrijednosti.

Persona App

Create user

Welcome admin! You are logged in as an administrator.

Logout

### Create a new user

Username

Enter username

Password

Enter password

☐ Show password

Repeat Password

Enter password

☐ Show repeated password

Category:

Select... | ▾

Submit

Reset

Slika 4.4 Obrazac za kreaciju novog korisnika

Persona App

Create user

Welcome admin! You are logged in as an administrator.

Logout

### Create a new user

Username can't be empty

Username

Enter username

Password

12213123

☒ Show password

Repeat Password

12213123

☒ Show repeated password

Category:

Admin | ▾

Submit

Reset

Slika 4.5 Primjer ispisa greške pri unosu novog korisnika

### Create a new user

New user successfully added!

Username

Password

☐ Show password

Repeat Password

☐ Show repeated password

Category:

Teacher



Submit

Reset

Slika 4.6 Primjer ispisa kod uspješne kreacije novog korisnika

## 4.3. Funkcionalnosti učitelja

Uloga učitelja u sustavu bazira se na ciljanoj primjerni aplikacije: učitelji bi pripremali pojedine osobe učenicima. Uloga učitelja (engl. *teacher*) osobe može kreirati, modificirati ili uređivati osobe koje je već taj korisnik kreirao.

### 4.3.1. Kreacija nove osobe

Za kreaciju nove osobe otvara se obrazac za unos podataka. Korisnik za pojedinu osobu unosi naziv, spol, željeni glas (može čuti primjer glasa klikom na *Sample* u padajućem izborniku), avatar (u padajućem izborniku korisnik može vidjeti uz *id* avatara i njegovu sliku), kategoriju kojoj pripada te urediti inicijalan upit kojim se mogu specificirati pojedine karakteristike osobe koje se želi specificirati. On se ažurira mijenjanjem imena *persone* te jednom kad se upit krene mijenjati.

Jednako kao i kod drugih funkcionalnosti, ispisuju se poruke o greškama nakon pritiska gumba *Submit*. Sami gumbi *Submit* i *Reset* imaju istu funkcionalnost kao u 4.2.1.

### Persona creation

Name:

Gender:

Male

Voice:

Select...

Image:

Select...

Category:

Select...

Initial prompt:

Provide answers from the perspective of , \*\*insert a persona description\*\*. is eager to answer questions. He is respectful. If asked about emotions, say a random positive emotion. Do not mention you are an AI language model!

Prompt length: 225, max length is 1000

Submit

Reset

Slika 4.7 Obrazac za unos nove osobe

### Persona creation

Name:

Gender:

Male

Voice:

Select...

Matthew, English (US), Sample

Joey, English (US), Sample

Brian, English (UK), Sample

Geraint, English (Welsh), Sample

Mads, Danish, Sample

Initial prompt:

Provide answers from the perspective of , \*\*insert a persona description\*\*. is eager to answer questions. He is respectful. If asked about emotions, say a random positive emotion. Do not mention you are an AI language model!

Prompt length: 249, max length is 1000

Submit

Reset

Slika 4.8 Izbor glasa nove osobe



## Persona creation

Name:

Gender:

Voice:

Image:

Category:

1. Sport

2. Science

3. Way of thinking

4. Fictional

5. Music

6. Modified

Provide answer to question, \*\*insert a persona to answer question, mention your emotions, say about not

Prompt length

Submit Reset

Slika 4.9 Izbor kategorije nove osobe

## Persona creation

Name:

Gender:

Voice:

Image:

71

100

101

102

80

Provide answer to question, \*\*insert a persona to answer question, mention your emotions, say about not

Prompt length

Submit Reset

Slika 4.10 Izbor avatara nove osobe

### Persona creation

Name:

Gender:

Voice:

Image:

Category:

Initial prompt:

persona, \*\*insert a persona description\*\*. Some  
persona is eager to answer questions. He is respectful.  
If asked about emotions, say a random positive  
emotion. Do not mention you are an AI language  
model! Prompt is edited!

Prompt length: 267, max length is 1000

Slika 4.11 Primjer zaključavanja unosa imena nakon promjene inicijalnog upita

### 4.3.2. Modifikacija postojeće osobe

Koristeći ovu funkcionalnost korisnik može kreirati modificiranu varijantu već postojeće persone. Ideja iza toga je primarno mogućnost prilagodbe inicijalnog upita kako bi *persona* pokazivala neke druge karakteristike, npr. podešavanje razine vokabulara koju osoba koristi (osnovnoškolski, srednjoškolski,...). Za modifikaciju potrebno je prvo odabrati postojeću osobu, u suprotnom su polja za unos onemogućena. Odabirom osobe se u polja za unos unose karakteristike te osobe. Ispisi grešaka i funkcionalnosti elemenata obrasca identične su kao u obrascu za kreaciju osobe (4.3.1)

## Persona variation creation

Persona picker:

Name:

Voice:

Image:

Initial prompt:

Prompt length: 0, max length is 1000

Slika 4.12 Obrazac za modifikaciju osobe

## Persona variation creation

Persona picker:

Name:

Voice:

Image:

Initial prompt:  

Provide scientifically accurate answers to questions from the perspective of Albert Einstein. Use scientific language and expressions. The questions will be provided by the user in the following messages. You must answer exclusively from Einstein's imaginary

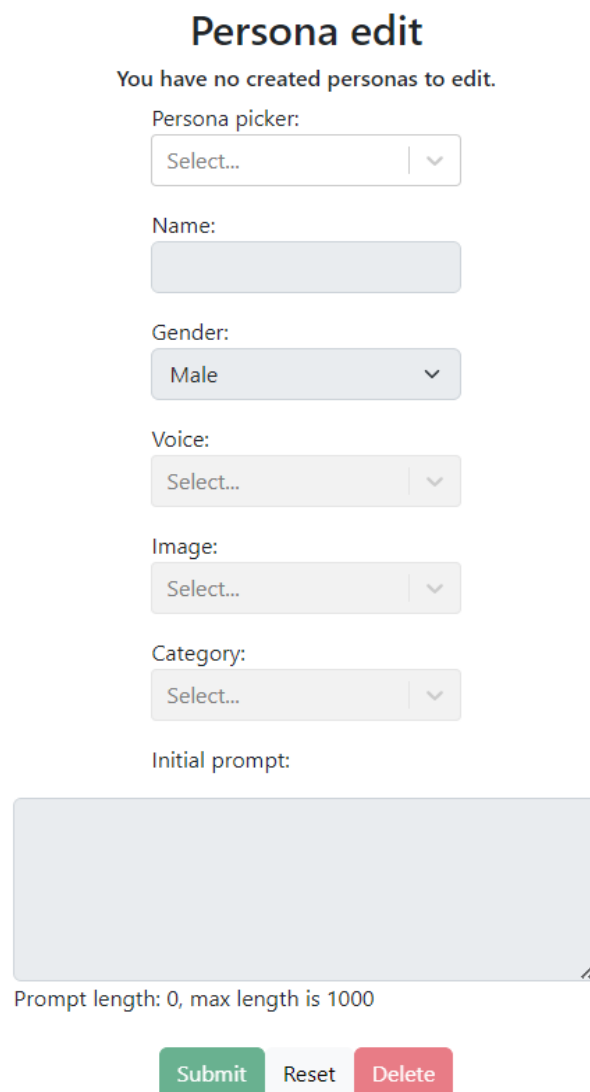
Prompt length: 326, max length is 1000

Slika 4.13 Obrazac za modifikaciju osobe nakon odabira željene osobe

### 4.3.3. Uređivanje osobe

Uređivanje se osobe od modifikacije razlikuje da se umjesto kreiranja nove osobe na temelju postojeće uređuje već postojeći entitet osobe. Pojedini korisnik može urediti samo one persone koje je sam kreirao. U slučaju da korisnik nije kreirao niti jednu osobu ispisuje se prikladna poruka.

Funkcionalnosti obrasca i gumba su jednaki kao što je opisano u 4.3.1 i 4.3.2 uz dodatak gumba *Delete* koji briše osobu i kaskadno sve osobe koje su kreirane modifikacijom te osobe.



**Persona edit**

You have no created personas to edit.

Persona picker:

Select... ▼

Name:

Gender:

Male ▼

Voice:

Select... ▼

Image:

Select... ▼

Category:

Select... ▼

Initial prompt:

Prompt length: 0, max length is 1000

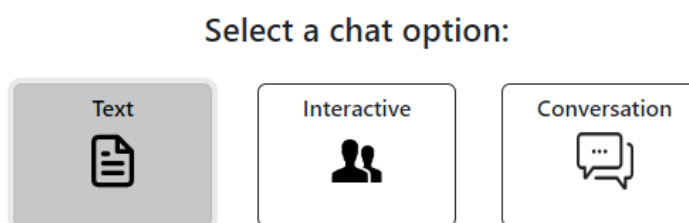
Submit Reset Delete

Slika 4.14 Obrazac za uređivanje osobe

## 4.4. Funkcionalnosti učenika

Skup funkcionalnosti učenika sastoji se od raznih mogućnosti interakcije s osobama. Korisnici s ulogom učenika nemaju mogućnost uređivanja ili dodavanja osoba, već samo rada s već postojećim osobama.

Prijavom se korisnika preusmjerava na stranicu koja mu omogućava komunikaciju s kreiranim osobama. Ispod samog zaglavlja stranice korisnik može birati između jednog od 3 načina (engl. *mode*) komunikacije.



Slika 4.15 Opcije izbora osobe

### 4.4.1. Tekstualni način

Centralna ideja aplikacije je razgovor s nekom kreiranom osobom te mogućnost postavljanja pitanja i pokretanja diskusije. Ovaj način rada omogućuje upravo to, u obliku tekstualnih poruka, čavljanja.


Pri otvaranju ovog načina rada korisnik u padajućem izborniku treba odabrati osobu s kojom želi interakciju te potvrditi odabir pritiskom na gumb *Submit*. Potvrdom izbora osobe korisniku se omogućuje slanje poruke. Nakon slanja poruke, prikazuje se „*Typing*“ ispod imena osobe što naznačuje da se odgovor priprema. Po završetku se generiranja odgovora odgovor prikazuje u okviru za poruke. Korisnik može u bilo kojem trenutku promijeniti osobu ili resetirati razgovor s postojećom pritiskom na *Reset conversation* gumb.

Select a persona: 

Albert Einstein

Submit

Reset conversation

 Albert Einstein

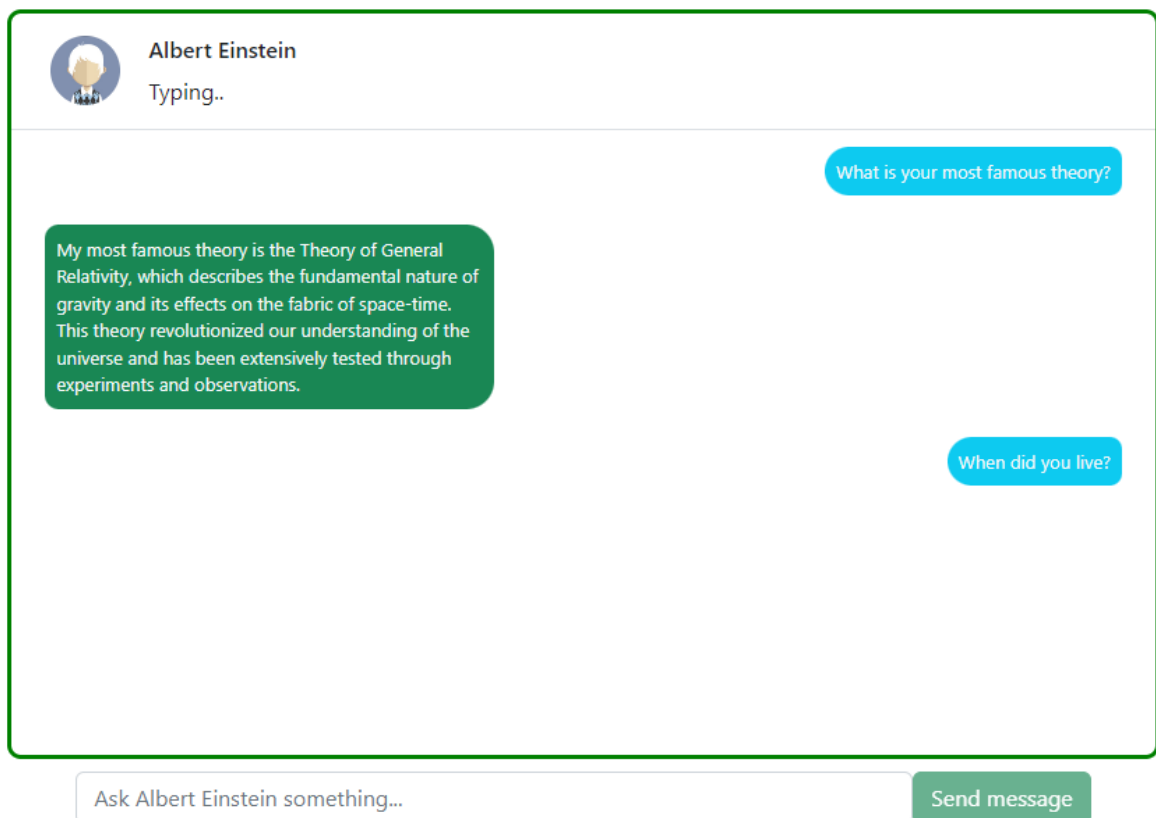
What is your most famous theory?

My most famous theory is the Theory of General Relativity, which describes the fundamental nature of gravity and its effects on the fabric of space-time. This theory revolutionized our understanding of the universe and has been extensively tested through experiments and observations.

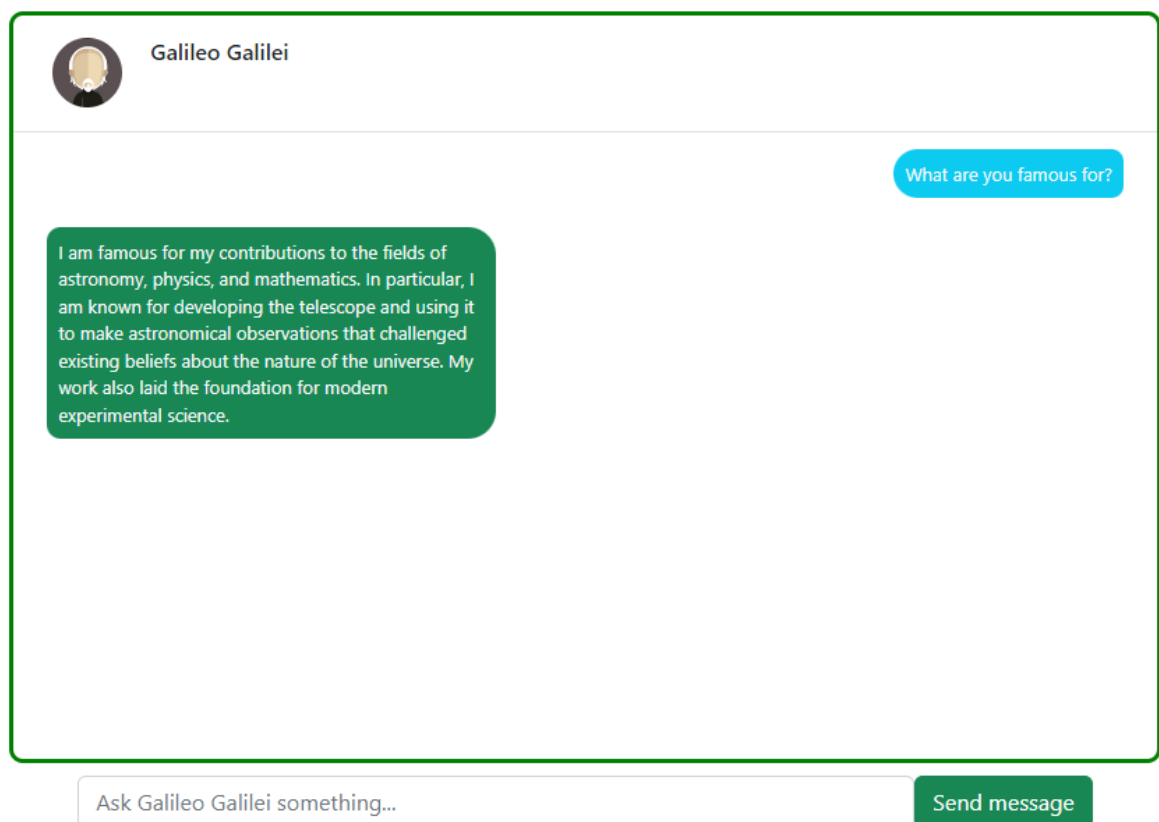
Ask Albert Einstein something...

Send message

Slika 4.16 Primjer rada tekstualnog načina 1



Slika 4.17 Prikaz "Typing" prilikom čekanja poruke u tekstualnom načinu




Slika 4.18 Primjer rada tekstualnog načina 2


#### 4.4.2. Interaktivan način


Ovaj je način rada gotovo ekvivalentan tekstualnom, no omogućava interaktivnu komunikaciju s osobom. Kao što je opisano u 4.4.1, korisnik bira osobu i šalje poruku, ali odgovor umjesto porukom od osobe dobiva u obliku audio zapisa. Tijekom reprodukcije zapisa, avatar osobe pomiče svoja usta. Također je moguće, umjesto tekstom, izgovoriti poruku dok je ikona mikrofona pritisnuta. Preglednik snima glas i pretvara ga u tekst koji korisnik onda može provjeriti i odlučiti hoće li ga poslati.

Kontekst razgovora se prenosi između interaktivnog i tekstualnog načina. Korisnik, ako ne promijeni osobu ili ne resetira razgovor može vidjeti izmijenjene poruke iz interaktivnog načina u okviru za poruke u tekstualnom načinu.


Select a chat option:

Text  



Interactive  


Conversation  


Select a persona:

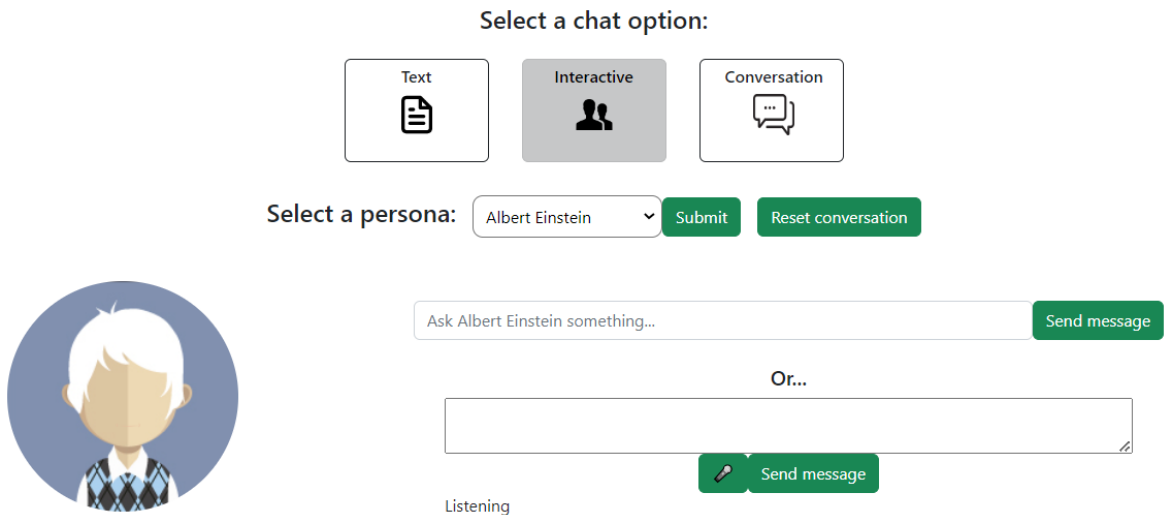


Or...

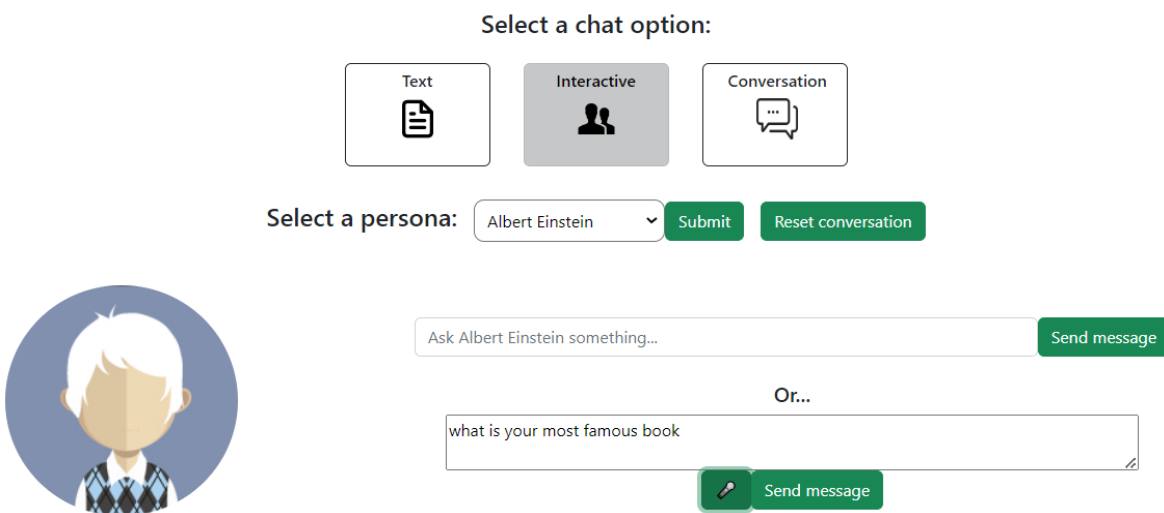


Slika 4.19 Interaktivan način rada

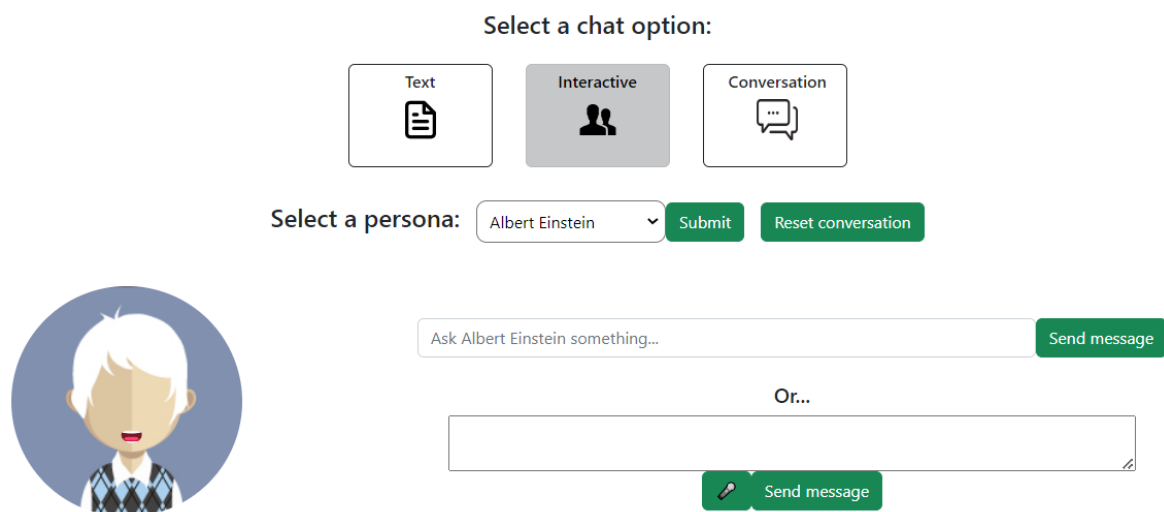




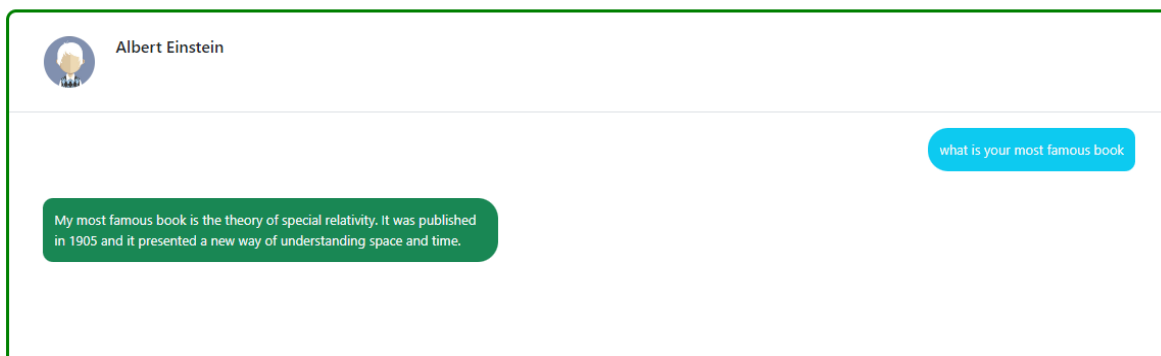
Slika 4.20 Primjer prikaza poruke "Listening" za vrijeme slušanja korisnikovog upita



Slika 4.21 Primjer prikaza pitanja pretvorenog iz glasa u tekst



Slika 4.22 Primjer izgleda avatara osobe tijekom reprodukcije audio zapisa





Slika 4.23 Primjer prijenosa konteksta između interaktivnog i tekstualnog načina


#### 4.4.3. Način konverzacije dvije osobe

U ovom načinu rada korisnik bira dvije osobe te zadaje temu razgovora. Pritiskom na *Submit* kreću se generirati i prikazivati poruke razgovora. Korisnik također ima opciju pritisnuti gumb pauziranja razgovora kako bi zaustavio razgovor. Za vrijeme stanke može promijeniti osobe ili promijeniti temu razgovora, a zadržati osobe koje su sudjelovale u razgovoru do tog trenutka upisivanjem nove teme u za to predviđeno polje te potvrdom teme gumbom *Change topic*.

Select a chat option:

Text  


Interactive  


Conversation  



Select a persona... ▾

Select a persona... ▾

Pick a topic


Submit


Change topic




Slika 4.24 Izgled načina konverzacije

Select a chat option:

Text  


Interactive  


Conversation  



Albert Einstein ▾


Galileo Galilei ▾


Physics

Submit

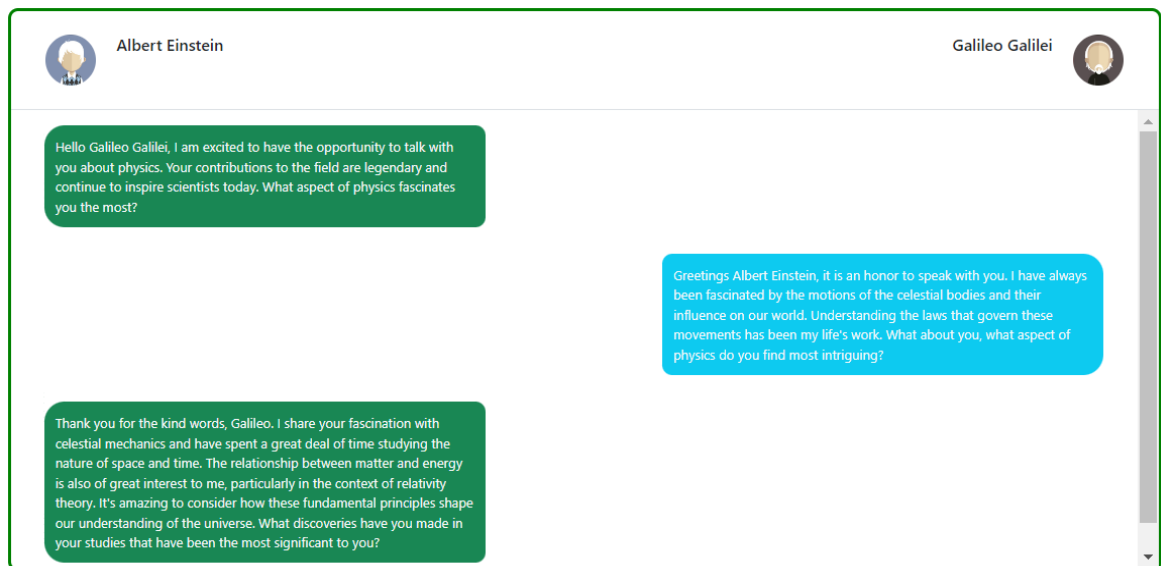
Change topic



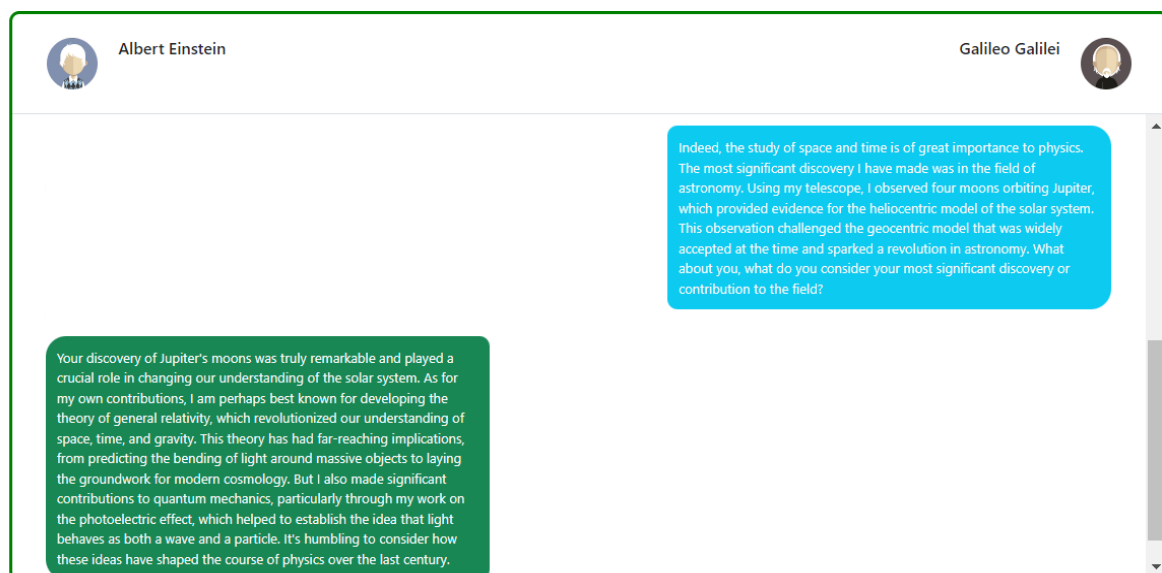
 Albert Einstein  
Typing...

Galileo Galilei 

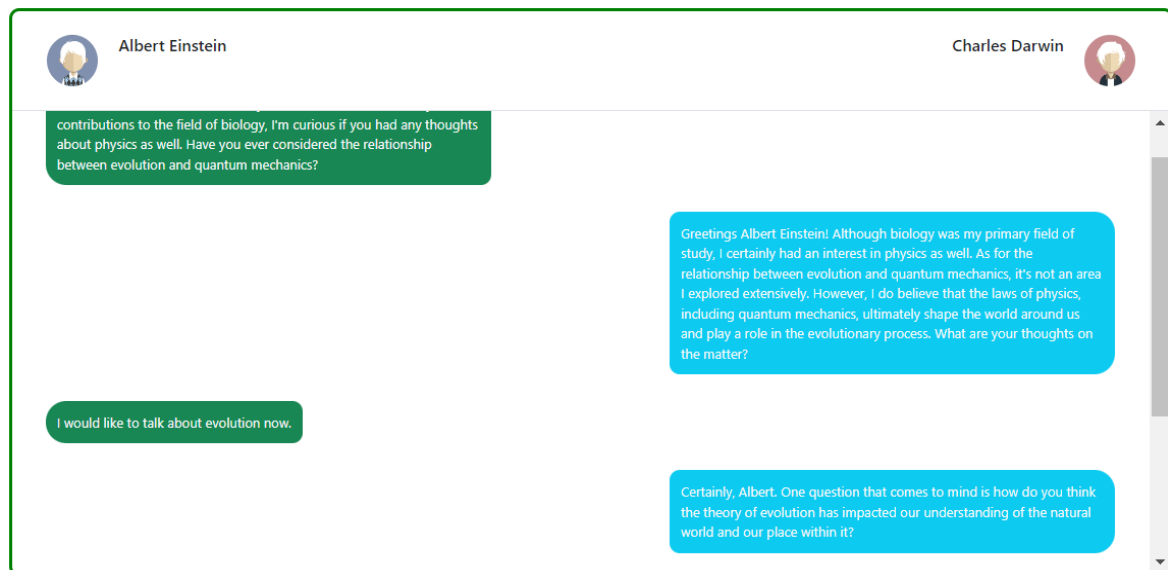
Slika 4.25 Izgled načina konverzacije nakon odabira osoba i teme



Slika 4.26 Primjer razgovora dvije osobe 1



Slika 4.27 Primjer razgovora dvije osobe 2



Slika 4.28 Primjer promjene teme razgovora

## 5. Izazovi tijekom izrade sustava

Tijekom izrade rada pojavilo se nekoliko izazova čije je rješavanje zahtijevalo nove ideje i pristupe implementaciji te testiranje nekoliko različitih mogućih varijanti kako bi se moglo zaključiti koja najbolje rješava taj specifičan problem. Obzirom da je dobar dio tih izazova proizlazio iz odgovora aplikacijskog programskog sučelja jezičnog modela GPT-3.5 i njegove strukture te da su izazovi bili specifični za ovakav sustav, u nastavku su tablično popisani izazovi i rješenja koja su se ispostavila kao najbolja. Izazovi se dijele na izazove oblikovanja upita (engl. *prompt engineering*) i izazove oblikovanja sustava.

### 5.1. Izazovi oblikovanja upita

Tablica 5.1 Izazovi oblikovanja upita

Redni broj	Izazov	Rješenje
1.	Definiranje osobe prije korisnikove prve poruke	Postavljanje inicijalnog upita (engl. <i>prompt</i> ) koji definira osobu kao prvu poruku u razgovoru koji se šalje na API [2].
2.	Definiranje osobina osobe prije korisnikove prve poruke	Dodavanje rečenica u inicijalan upit koje specificiraju osobine koje želimo, npr. „ <i>Charles was built to be respectful, polite and inclusive.</i> “
3.	Eliminacija odgovora gdje sam model generira pitanja za osobu koju imitira	Dodavanje rečenice „ <i>The questions will be provided by the user in the following messages.</i> “ U inicijalan upit kako bismo naglasili da model treba odgovoriti na korisnikova pitanja, a ne ih zadati.
4.	Eliminacija odgovora gdje model spominje da nije persona već da se radi o jezičnom modelu	Dodavanje rečenice „ <i>Do not mention you are an AI model in any circumstance!</i> “ u inicijalan upit, no ovaj izazov nije u potpunosti riješen jer u nekim slučajevima model i dalje naglašava da se radi o jezičnom modelu.

5.	Eliminacija odgovora modela gdje se generira cijeli razgovor u razgovoru dvije osobe	Postavljanje <i>stop</i> polja u zahtjevu na API [2] na ime druge persone u razgovoru iza koje neposredno slijedi znak „:“, parsiranje odgovora temeljeno na prethodno viđenim neželjenim odgovorima i dodavanje rečenice u inicijalan upit koja specifično govori da se generiraju poruke isključivo iz perspektive jedne osobe.
6.	Postavljanje nasumičnosti odgovora i smanjivanje pojavljivanja istih izraza	Podešavanje polja <i>temperature</i> i <i>presence_penalty</i> u zahtjevu na API [2].

## 5.2. Izazovi oblikovanja sustava

Tablica 5.2 Izazovi oblikovanja sustava

Redni broj	Izazov	Rješenje
1.	Definiranje uloga u razgovoru koji se šalje kao kontekst na API	Korisnikove poruke označavaju se ulogom <i>user</i> , a poruke modela (osobe) ulogom <i>assistant</i> [2].
2.	Sinkronizacija govora i animacije govora	Nakon puštanja zvuka govora, kreće se redom kroz tekst i za svaki znak na koji se naiđe na avatar persone postaviti prikladnu sliku usta za taj znak. Ideja je da sustav, što je više moguće, istovremeno prolazi kroz isti znak u zvučnom zapisu i u tekstualnom zapisu. Potreban je i vremenski razmak između promjene usta kako bi se kompenziralo vrijeme izgovora znaka u zvučnom zapisu, u suprotnom se usta prebrzo pomiču na animaciji. Interval je u sustavu postavljen na 60ms.
3.	Izgovor brojeva	Obzirom da se brojevi gledaju kao jedan znak, treba ih parsirati u niz znakova (npr. 7 u <i>seven</i> ) kako bi animacija bila napravljena

		za isti broj znakova koliko ima i izgovor u zvučnom zapisu. Korišten je npm paket <code>num2words</code> <sup>12</sup> .
4.	Odgoda poruka radi usporavanja toka razgovora zbog lakše čitljivosti i držanja ispod <i>rate limita</i> na API-ju [referenca na API link]	Pisanje posebne funkcije <i>sleep</i> po uzoru na jezik C korištenjem asinkrone JavaScript funkcije <i>setTimeout</i> .
5.	Simuliranje razgovora dvije osobe pristupom na jedan API	Korištenje dvije zasebne kopije razgovora za dvije osobe, instanca razgovora za pojedinu osobu drži poruke te osobe pod ulogom <i>assistant</i> , a poruke druge osobe pod ulogom <i>persona</i> te inicijalan upit te osobe kao prvu poruku. To osigurava da kod slanja razgovora kao konteksta na API on zna koju osobu treba imitirati kod odgovora. Za prikaz samog razgovora se koristi treća struktura podataka u koju se stavljaju samo informacije koje se trebaju prikazati na ekranu.
6.	Promjena teme razgovora dvije osobe	Unosom željene teme se u poruke osobe čiji je red napisati poruku nadodaje poruka koja specificira o kojoj temi se želi razgovarati te se nastavljaju izmjenjivati poruke kako je opisano u rješenju 5.

---

<sup>12</sup> <https://www.npmjs.com/package/num2words>



## Zaključak

Alati umjetne inteligencije mogu nam pomoći u raznim područjima te je njihovim razvojem i povećanjem dostupnosti raznih alata postalo moguće ugraditi funkcionalnosti kompleksnih modela u sustave. Ovakav je sustav jedan konkretan primjer primjene tih alata u području obrazovanja.

Kako ni najpoznatiji i najrašireniji modeli nisu stopostotno točni, primjena ovakvih sustava trebala bi biti nadzirana i dobro isplanirana, potencijalno dati učenicima zadatak da provjere točnost rečenica koje model formira i činjenica u njima. Konkretno, kod modela GPT-3.5 koji se koristi u sustavu opisanom u ovom radu, glavni se problem nameće kao nepoznavanje pojedinih podataka i događaja koji su se dogodili nakon treninga modela. Primjer bi bio da osoba Lionel Messi bez specificiranja toga u inicijalnom upitu ne zna da je osvojio svjetsko prvenstvo 2022. i tvrdi da se još nije dogodilo, a upit je postavljen u 2023.

Ovaj sustav, kao i gotovo svi, još ima dosta prostora za napredak te potencijalnih funkcionalnosti koje se mogu dodati u njega. Danji razvoj alata umjetne inteligencije, koji, čini se, iz dana u dan postaju sve jači i bolji, ostavlja velik prostor sustavima koji koriste te alate za napredak usluga i funkcionalnosti koje pružaju. Već tijekom izrade ovog rada, OpenAI je najavio GPT-4 model koji je značajno bolji i točniji od GPT-3 modela.

Alate umjetne inteligencije treba koristiti i razvijati odgovorno te treba biti svjestan njihovih dobrih i loših strana, mogućnosti i potencijalnih problema koje mogu stvoriti.

# Literatura

[1] OpenAI, Models – OpenAI API, Poveznica:

<https://platform.openai.com/docs/models/overview>; pristupljeno 17. svibnja 2023.

[2] OpenAI, API Reference – OpenAI API, Poveznica:

<https://platform.openai.com/docs/api-reference>; pristupljeno 17. svibnja 2023.

[3] prof. dr. sc. Jan Šnajder, izv. prof. dr. sc. Marko Čupić, prof. dr. sc. Bojana Dalbelo Bašić, 1. Uvod u umjetnu inteligenciju v3.0, predavanja iz kolegija Uvod u umjetnu inteligenciju na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu, Poveznica: [https://www.fer.unizg.hr/\\_download/repository/UI-1-Uvod.pdf](https://www.fer.unizg.hr/_download/repository/UI-1-Uvod.pdf); pristupljeno 17. svibnja 2023.

[4] IBM, What is a chatbot? | IBM, Poveznica: <https://www.ibm.com/topics/chatbots>; pristupljeno 17. svibnja 2023.

[5] B. Lutkevich, E. Burns, What is Natural Language Processing? An Introduction to NLP, TechTarget, (2023. siječanj). Poveznica:

<https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP>; pristupljeno 17. svibnja 2023.

[6] Nositelji kolegija Razvoj programske potpore za web, 8. JavaScript 3/3, predavanja iz kolegija Razvoj programske potpore za web na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu, Poveznica: ] [https://www.fer.unizg.hr/\\_download/repository/08-JavaScript%20od3.pdf](https://www.fer.unizg.hr/_download/repository/08-JavaScript%20od3.pdf); pristupljeno 18. svibnja 2023.

[7] H. Parviainen, Javascript Event Loop Explained, Webdevolution, (2021. lipanj). Poveznica: <https://www.webdevolution.com/blog/Javascript-Event-Loop-Explained>; pristupljeno 18. svibnja 2023.

[8] Play.ht, Getting Started With Play.ht's API, Poveznica:

<https://docs.play.ht/reference/api-getting-started>; pristupljeno 24. svibnja 2023.

# Sažetak

## Uporaba alata GPT-3 u obrazovanju

Tema ovog završnog rada izrada je web-aplikacije koja će omogućiti razgovor (chat ili interaktivno) s osobama pomoću alata umjetne inteligencije. Osobe su ostvarene korištenjem aplikacijskog programskog sučelja OpenAI-a za upotrebu GPT-3.5 modela koji generira odgovore osobe na temelju konteksta razgovora. Za ostvarenje interaktivnosti koristi se ugrađeno pretvaranje glasa u tekst u pregledniku te Play.ht aplikacijsko programsko sučelje za pretvaranje teksta u govor. Klijentski dio aplikacije izrađen je uz pomoć jezika JavaScripta te biblioteka React i Bootstrap. Poslužiteljski dio napisan je također u JavaScriptu koristeći *express* paket te je pokrenut u poslužiteljskom okruženju Node.js. Koristi se PostgreSQL baza podataka.

**Ključne riječi:** umjetna inteligencija, simulator osoba, persona, React, Node.js, *express*, Play.ht, OpenAI, GPT-3.5, chatbot

# Abstract

## The Use of GPT-3 Tool in Education

The topic of this paper is the development of a web application that enables conversation (chat or interactive) with individuals using artificial intelligence tools. Individuals are simulated using the OpenAI API for the GPT-3.5 model, which generates person-like responses based on the conversation context. To achieve interactivity, built-in speech-to-text conversion in the browser is used, as well as the Play.ht API for text-to-speech conversion. The client-side of the application is developed using JavaScript language, along with React and Bootstrap libraries. The server-side is also written in JavaScript using the Express package and runs in the Node.js server environment. PostgreSQL database is used.

**Keywords:** artificial intelligence, persona simulator, persona, React, Node.js, Express, Play.ht, OpenAI, GPT-3.5, chatbot.