SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 872

**Uporaba alata GPT-3 u obrazovanju**Ivan Terzić

Zagreb, lipanj 2023.

Zahvaljujem obitelji i prijateljima koji su mi puno pomogli i olakšali studij te mentoru izv. prof. dr. sc. Ivici Botičkom.

# 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: čavrljanje (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 čavrljanja dvije osobe čije poruke generira isključivo GPT-3.5 model.

# Teorijska pozadina

## 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.*

## 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 nastavljanja 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.

## 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.

## Web aplikacija

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.

## Aplikacijsko programsko sučelje

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.

# Korištene tehnologije

## JavaScript

JavaScript[[1]](#footnote-1) 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.

### Iznimke

Iznimke (engl. *Exception*) su događaji koji se pojavljuju kod izvođenja koda 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 kod za koji postoji mogućnost da generira iznimku. U trenutku generiranja iznimke, ovaj blok prestaje s izvođenjem.

U *catch* blok se piše kod koji želimo da se izvrši ako dođe do iznimke. Ovaj blok prima i argument (kao funkcija) objekta za grešku tipa *Error* koji sadrži korisne informacije za obradu i analizu greške, npr. vrsta i poruka greške. Takvi se objekti mogu i stvarati u kodu.

*Finally* blok sadrži kod 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.")

}

}

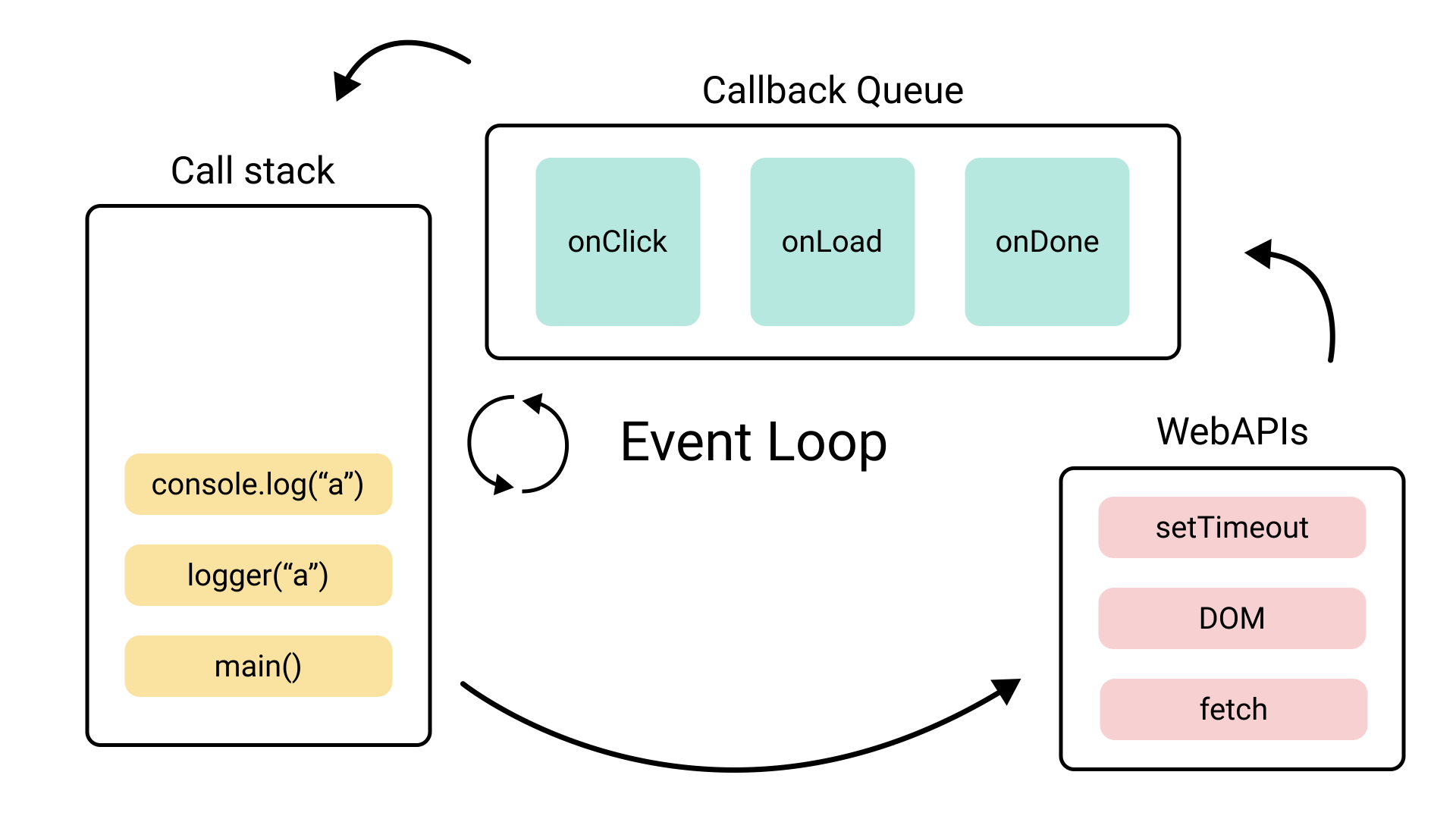
Kod 2.1 Primjer dohvaćanja podataka s try, catch i finally blokovima

### 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 kod 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 kod 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]

### Ključne riječi *async* i *await*

*Async* i *await* su ključne riječi programskog jezika JavaScript koje koristimo kako bismo efektivno mogli izvoditi asinkroni kod 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 kod.

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 kod.

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 occured 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);

}

Kod 2.2 Primjer *async* funkcije

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

## React

React[[2]](#footnote-2) je besplatna JavaScript knjižnica (engl. *library*) otvorenog koda (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 kod 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 varijablii 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;

Kod 2.3 Jednostavan primjer React koda

## Node.js

Node.js[[3]](#footnote-3) je poslužiteljsko okruženje otvorenog koda za JavaScript koje se može pokrenuti na raznim operacijskim sustavima. U ovom se okuženju često izrađuju poslužiteljske strane web aplikacija.

U Node.js uključen je i npm[[4]](#footnote-4) (*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*[[5]](#footnote-5)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* [[6]](#footnote-6) 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}`);

});

Kod 2.4 Isječak poslužiteljskog koda u *expressu*

## 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 koda, a CSS kod izrade nekih specifičnih stilova elemenata na sučelju.

### Bootstrap

Bootstrap[[7]](#footnote-7) 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>

Kod 2.5 Primjer JSX koda s Bootstrap klasama

### Axios proxy

Axios[[8]](#footnote-8) 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'

}

}

);

Kod 2.6 Definiranje instance Axiosa

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

Kod 2.7 Slanje zahtjeva preko definirane instance Axiosa

## PostgreSQL

## Aplikacijska programska sučelja

# Arhitektura sustava

Dijagram baze

Opis tablica

Dijagram komponenti

# Zaključak

Na kraju rada piše se kratak zaključak, duljine do najviše jedne stranice.

# Literatura

Popis literature dolazi na kraju rada, iza zaključka, a prije ostalih priloga.

Na naslov Literatura primijenite stil Heading 1, a zatim ručno maknite brojčanu oznaku (to je važno kako bi i naslov „Literatura“ ušao u sadržaj na početku rada, prije uvoda).

Pri kreiranju navoda u popisu literature koristite stil literatura.

Primjeri u nastavku ilustriraju navođenje raznih izvora u popisu literature: (1) knjige, (2) članka u časopisu, (3) članka u zborniku konferencije, (4) doktorskog, magistarskog ili diplomskog rada, (5) web-stranice.

1. Tanenbaum, A. S., Wetherall, D. J. Computer Networks. 5. izdanje. London: Pearson, 2013.
2. Brady, P.T. A Statistical Analysis of On-off Patterns in 16 Conversations, Bell System Technical Journal, 47,1 (1998), str. 55-62.
3. Brady, N. A Statistical Analysis of Use Case. Proceedings of the 7th International Conference on Telecommunications ConTEL, Zagreb, (2003), str. 45-52.
4. Ivić, M. Analiza ponašanja korisnika u digitalnim igrama namijenjenima učenju. Diplomski rad. Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva, 2016.
5. Epstein M., The best VR headset in 2019, PC Gamer, (2019, listopad). Poveznica: <https://www.pcgamer.com/best-vr-headset/>; pristupljeno 4. listopada 2019.

Uz svaki preuzeti sadržaj u svom radu – bilo da je riječ o tekstu (izravno citiranome ili „prepričanome“), slici ili grafičkom prikazu – treba navesti oznaku izvora (članak, knjiga, web-stranica ...) u popisu literature te se na nju „pozvati“, na primjer:

Međusobno povezivanje mreža zasniva se na primjeni komunikacijskih protokola (Tanenbaum i Wetheral, 2014).

Podaci o karakteristikama uređaja za virtualnu stvarnost preuzeti su s portala PC Gamer [5].

Početna verzija programa preuzeta je iz diplomskog rada [5].

U danim primjerima mogli ste uočiti dva načina referenciranja:

* (Tanenbaum i Wetheral, 2014),
* [1].

Kad izaberete jedan od njih svakako ga se držite konzistentno u cijelome radu.

DODATI SVE POD INSERT CITATIONS SOURCES

[1] <https://platform.openai.com/docs/models/overview>

[2] <https://platform.openai.com/docs/api-reference/introduction>

[3] <https://www.fer.unizg.hr/_download/repository/UI-1-Uvod.pdf>

[4] <https://www.ibm.com/topics/chatbots>

[5] <https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP>

[6] <https://www.fer.unizg.hr/_download/repository/08-JavaScript%203od3.pdf> pristupano 17.5.

[7] <https://www.webdevolution.com/blog/Javascript-Event-Loop-Explained>

# Sažetak

Naslov, sažetak, ključne riječi (na hrvatskom jeziku)

Sažetak opisuje sadržaj rada, prepričan u stotinjak riječi.

# Summary

Title, summary, keywords (na engleskom jeziku)

# Skraćenice

Ovo poglavlje nije obavezno, ali se može dodati radi preglednosti.

ATM Asynchronous Transfer Mode asinkroni način prijenosa

ISDN Integrated Services Digital Network digitalna mreža integriranih usluga

Napomena: na naslov Skraćenice primijenite stil Heading 1, a zatim ručno maknite brojčanu oznaku (to je važno kako bi i skraćenice ušle u sadržaj na početku rada, prije uvoda). Pri kreiranju popisa skraćenica koristite stil nabrajanje.

# Privitak

Privitak je također opcionalno poglavlje (u dogovoru s mentorom).

Sadržaj koji se stavlja u privitak je, općenito, nešto što je, kao cjelinu, prikladno izdvojiti iz sadržaja samog rada.

Mogući primjer je tehnička dokumentacija vezana uz završni rad - npr. električka i položajna shema sklopa, sastavnica, predložak tiskane veze, plan bušenja, ispis programa s detaljnim opisom.

Drugi primjer uključuju upute za korištenje rezultata rada (softvera ili hardvera), detaljni ispisi mjerenja čiji su rezultati sažeto ili grafički prikazani u radu. Ako se radi o softveru, uobičajeno je navesti podatke o platformi na kojoj se izvodi (npr., karakteristike uređaja i operacijskog sustava te pomoćnog softvera), kao i upute za instalaciju.

U privitku nemojte koristiti stilove razine Heading, već samo (nenumerirani) stil Podnaslov.

Na primjer:

Instalacija programske podrške

Upute za korištenje programske podrške

.

.

.

Ostali savjeti

U izborniku Tools - opcija Language postavite Croatian kao default jezik. Na kraju provedite strojnu provjeru teksta (spell checking, ako ga imate ugrađenoga), ali svakako pažljivo i pročitajte vlastiti tekst.

Ako nemate ugrađeni spelling checker za hrvatski jezik, možete se poslužiti Hascheckom (izgovara se Hašek, a ime dolazi od kratice za Hrvatski akademski spelling checker), dostupan putem poveznice <https://ispravi.me/>.

1. <https://developer.mozilla.org/en-US/docs/Web/JavaScript> [↑](#footnote-ref-1)
2. <https://react.dev/> [↑](#footnote-ref-2)
3. <https://nodejs.org/en> [↑](#footnote-ref-3)
4. <https://www.npmjs.com/> [↑](#footnote-ref-4)
5. <https://expressjs.com/> [↑](#footnote-ref-5)
6. <https://www.npmjs.com/package/express-validator> [↑](#footnote-ref-6)
7. <https://getbootstrap.com/> [↑](#footnote-ref-7)
8. <https://axios-http.com/docs/intro> [↑](#footnote-ref-8)