

SVEUČILIŠTE U ZAGREBU  
FAKULTET ORGANIZACIJE I INFORMATIKE  
VARAŽDIN

Aleksandar Babić

**INFORMATIVNA PLATFORMA S  
INTELIGENTNIM ASISTENTOM O  
ERASMUS+ ISKUSTVU U ŁÓDŽU**

DIPLOMSKI RAD

Varaždin, 2025.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Aleksandar Babić

Matični broj: 0016147642

Studij: Baze podataka i baze znanja

## INFORMATIVNA PLATFORMA S INTELIGENTNIM ASISTENTOM O ERASMUS+ ISKUSTVU U ŁÓDŽU

DIPLOMSKI RAD

Mentor:

doc. dr. sc. Bogdan Okreša Đurić

Varaždin, rujan 2025.

## **Zahvala**

Na samom početku želim izraziti iskrenu zahvalnost svom mentoru, doc. dr. sc. Bogdanu Okreši Đuriću, na izdvojenom vremenu, nesebičnoj podršci i strpljenju koje mi je pružio tijekom izrade ovog diplomskog rada.

Posebnu zahvalnost dugujem svojoj obitelji – ocu, majci i sestri – za njihovu podršku i ohrabrenje tijekom cijelog mog studija. Njihova prisutnost i vjera u mene davali su mi snagu u trenucima kada je to bilo najpotrebnije.

Na kraju, veliko hvala svim prijateljima koje sam upoznao tijekom studiranja. Zajedno smo prošli kroz brojne izazove, ali i podijelili mnogo smijeha i lijepih trenutaka koje ću zauvijek nositi sa sobom.

Aleksandar Babić

**Izjava o izvornosti**

Izjavljujem da je ovaj diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

*Autor potvrdio prihvaćanjem odredbi u sustavu FOI Radovi*

---

## **Sažetak**

Teoretski dio rada istražuje studiranje kao važan segment u životnom razdoblju pojedinca te se dotiče razmjene studenata u svrhu Erasmus+ studijskog boravka. Teoretski dio završava poglavljem posvećenim inteligentnim asistentima temeljenima na RAG (eng. Retrieval-Augmented Generation) tehnologiji. Praktični dio rada obuhvaća izradu web-stranice s tematikom Erasmusa+ studijskog boravka i izradu intelligentnog asistenta, pri čemu je iskorišten postojeći jezični model. To podrazumijeva treniranje i prilagođavanje specifičnom skupu podataka, kako bi se osiguralo da intelligentni asistent odgovara upitima korisnika i pruža relevantne informacije.

**Ključne riječi:** Intelligentni asistent, RAG, web stranica, studiranje, Erasmus+, umjetna inteligencija.

# Sadržaj

<b>1. Uvod</b>	1
<b>2. Studiranje kao dio života pojedinca</b>	2
2.1. Odluka i benefiti	2
2.2. Erasmus+ studenska mobilnost	4
2.2.1. Izazovi i strahovi	5
<b>3. Aktualne tehnologije i alati za uspješno snalaženje i informiranje</b>	7
3.1. Važnost inteligentnih pomoćnih sustava	7
3.1.1. Asistent	8
3.1.1.1. Siri	8
3.1.2. Copilot	9
3.1.2.1. Microsoft Copilot Studio	9
3.1.3. Agent	11
3.1.3.1. ChatGPT Operator	11
3.1.4. Chatbot	12
3.1.4.1. Zendesk	12
3.2. Inteligentni asistenti u obrazovnom sustavu	13
3.2.1. Primjena inteligentnih asistenata u obrazovanju	13
3.2.2. Izazovi inteligentnih asistenata u obrazovanju	14
<b>4. Retrieval Augmented Generation metoda</b>	15
4.1. Uvod u RAG	15
4.1.1. Povijest i razvoj	16
4.2. Arhitektura RAG modela	18
4.2.1. Faza dohvaćanja (Retriever)	19
4.2.1.1. BM25	20
4.2.1.2. Dense Passage Retrieval	20
4.2.1.3. REALM	20
4.2.2. Faza generiranja (Generator)	21
4.2.2.1. T5	21
4.2.2.2. BART	21
4.3. Primjena RAG modela	21
4.3.1. Ključne prednosti RAG sustava	23
4.4. Izazovi RAG pristupa	23
4.4.1. Halucinacije	25
4.5. Dostupni alati i biblioteke	26

4.5.1. Okviri (frameworkovi) za upravljanje RAG-om . . . . .	26
4.5.1.1. Haystack . . . . .	26
4.5.1.2. LangChain . . . . .	27
4.5.1.3. LlamalIndex . . . . .	27
4.5.2. Vektorske baze podataka . . . . .	28
4.5.2.1. FAISS . . . . .	29
4.5.2.2. Qdrant . . . . .	30
4.5.2.3. Pinecone . . . . .	30
4.5.3. Veliki jezični modeli . . . . .	30
4.5.3.1. OpenAI . . . . .	30
4.5.3.2. Hugging Face . . . . .	31
4.5.3.3. Anthropic . . . . .	31
<b>5. Korištene tehnologije i alati . . . . .</b>	<b>32</b>
5.1. React . . . . .	32
5.2. FastAPI . . . . .	33
5.3. SMTP . . . . .	33
<b>6. Praktični dio rada . . . . .</b>	<b>34</b>
6.1. Struktura web stranice . . . . .	34
6.2. Korisničko sučelje (Frontend) . . . . .	35
6.3. Poslužiteljski dio (Backend) . . . . .	36
6.3.1. Server . . . . .	36
6.3.2. Servisi . . . . .	37
6.3.3. Rute . . . . .	38
6.3.4. Inteligentni asistent . . . . .	40
6.3.4.1. Prikupljanje podataka . . . . .	40
6.3.4.2. Web scraping . . . . .	41
6.3.4.3. Čišćenje i obrada podataka . . . . .	45
6.4. Prikaz rada aplikacije . . . . .	45
<b>7. Zaključak . . . . .</b>	<b>49</b>
<b>Popis literature . . . . .</b>	<b>52</b>
<b>Popis slika . . . . .</b>	<b>54</b>
<b>Popis tablica . . . . .</b>	<b>55</b>
<b>Popis isječaka koda . . . . .</b>	<b>56</b>

# 1. Uvod

Na početku rada obrađuje se pojam studiranja kao i svi aspekti koji se usko vežu uz njega. Poseban naglasak je stavljen na važnost studiranja u kontekstu osobnog razvoja, izgradnje karaktera te stjecanja različitih znanja. U nastavku se detaljnije razmatra program Erasmus+, s ciljem približavanja njegovih mogućnosti i koristi. Glavni cilj je ukazati na izazove poput nedostatka interesa za prijavu na Erasmus+ studijski boravak, prisutnog straha, nesigurnosti te neinformiranosti među studentima.

Kao rješenje na navedene izazove izrađena je web stranica s inteligentnim asistentom koji je namijenjen da odgovori na specifične upite korisnika. Web stranica predstavlja sredstvo informiranja prvotno za sve studente koji dolaze u Łódź iz svojih država. Osim toga web stranica je naravno dostupna i namijenjena i za sve ostale zainteresirane posjetitelje. Web stranica se sastoji od početne stranice, stranice koja opisuje kratko Łódź, stranice koja prikazuje 15 najvećih grada Poljske i stranice na kojoj se nalazi forma. Forma je namijenjena za pitanja na koja nije mogao AI (eng. Artificial Intelligence) asistent odgovoriti.

Za izradu AI modela odabранo je korištenje RAG (eng. Retrieval Augmenting Generation) gdje se uzima postojeći AI model i on se dotrenirava na specifičnom skupu podataka. Za takve potrebe pronađeno je nekoliko lokalnih web stranica koje opisuju život u Łódžu. Za potrebe treniranja razvijen je scraper koji dohvata primarno tekstualni sadržaj s istih tih stranica i spremi ih u JSON obliku. Naknadno su te JSON datoteke pročišćene i iskorištene za daljnje treniranje. Tijekom svoje Erasmus+ studentske mobilnosti imao sam prilike razgovarati s profesoricom na kolegiju Umjetna inteligencija, kojoj sam ukratko predstavio ideju svog diplomskog rada. Naknadno sam je zamolio da mi preporuči nekoliko korisnih stranica koje su vezane za Łódź. Budući da se profesorka rodila u Łódžu i tamo živi te radi cijeli život, smatrao sam da je upravo ona prava osoba kojoj se mogu obratiti s pitanjem o korisnim izvorima informacija vezanima za život u Łódžu.

Za tehnologije su odabrani: React (Java Script) za frontend, Python (za razvoj scrapera), NodeJS (Za razvoj servera na backendu), Python (za razvoj modela unutar Google Colab platforme).

Prethodno ulomci uvode nas u temu s kojom se bavi ovaj rad, a to je „Informativna platforma s inteligentnim asistentom o Erasmus+ iskustvu u Łódžu“. Prvotna motivacija za proučavanje ove teme proizšla je iz Erasmus+ studijskog boravka u Łódžu (Poljska) u trajanju jednog semestra.

## **2. Studiranje kao dio života pojedinca**

Važnost učenja se proteže kroz sve uzraste i faze života. Već od malih nogu školska djeca se potiču da marljivo uče jer to na neki način može odrediti njihov put u budućnosti. Unutar ovog poglavlja više je vremena posvećeno studiranju kao važnom segmentu u životnom razdoblju pojedinca. U tom kontekstu visoko obrazovanje može napraviti razliku između uspješne osobe i neuspjeha. Ulaganje u znanje je beskrajno i nikada ne prestaje i može se sagledati kao investicija u budućnost. Život bez obrazovanja je kao tijelo bez duše, jer obrazovanje pojedincu donosi smisao, znanje i prilike koje oblikuju njegov osobni i profesionalni razvoj. Pomaže u razvoju kritičkog mišljenja, ima plemenit karakter i razvija vještine koje su potrebne za suočavanje sa svim životnim izazovima. Prema riječima stručnjaka iz škola u Bengaluru (Indija), studij je važan jer se koristi za ublažavanje kompleksnosti većine izazova s kojim se susretne pojedinac [1].

### **2.1. Odluka i benefiti**

Kod donošenja velikih odluka kao što je početak studiranja vrlo je bitno uzeti u obzir nekoliko čimbenika koji mogu oblikovati bolji pogled na razmišljanje o nečemu. Kada se razmišlja o tome da li krenuti studirati ili ne, postoje ljudi koji govore da upisivanje fakulteta više nije presudno ni važno za uspjeh ili da se može dobiti dobar posao bez odlaska na fakultet. Primarni cilj ove sekcije nije zastupanje određene strane, već pružanje uvida u razloge zbog kojih bi upis na fakultet mogao biti dobra odluka. Započetak fakultet priprema radnu snagu za budući posao i može dati značajnu prednost prilikom prijave za posao. Fakultetsko obrazovanje pokazuje poslodavcima da je osoba sposobna dovršiti dugoročni projekt, da je sposobna kritički razmišljati, rješavati probleme i da je sposobna učiti nove stvari. Većina poslova danas zahtijeva barem neko fakultetsko iskustvo i vjerojatno će osobe bez diplome biti u nepovoljnjoj situaciji u usporedbi s konkurencijom. Čak i ako se ne odluči upisati fakultet postoje razni programi, certifikati i tečajevi koji mogu pomoći u razvoju vještina i znanja koje su potrebne za radno mjesto [2].

Umrežavanje je možda i najveći benefit koji se može steknuti tijekom studiranja. Upoznavanje novih ljudi i gradnja važnih odnosa koji mogu potrajati dugo u budućnosti. Fakultet je mjesto koje nudi priliku upoznavanja drugih studenata s istim/različitim interesima a osim toga postoje i prilike umrežavanja s profesorima i drugim stručnjacima. S toliko studenata i članova fakulteta iz različitih geografskih sredina, sigurno će se pronaći neko tko dijeli iste stavove. Pronaći će se kolega za učenje, kavu ili osoba koja će postati najbolji prijatelj. Osim toga unutar studentskih kampusa moguće je upoznavanje velikog broja ljudi kroz razna sportska događanja. Kroz upoznavanja, osoba može puno naučiti o sebi i onome što ju zanima. Moguće je upoznavanje s osobama koje dolaze iz svih krajeva svijeta čime se mogu okusiti nove kulture i perspektive. Možda će se pojavitи prilika za život sa studentom/studenticom iz druge zemlje ili će se možda pojavitи profesor koji predaje s međunarodnim iskustvom. Takva iskustva mogu razviti bolje razumijevanje svijeta oko sebe. Pohađanje sajmova karijera na fakultetima ili pridruživanje profesionalnim organizacijama su također neke od prilika koje mogu osnažiti i

izgraditi mrežu koja može obogatiti buduću karijeru [2].

Mnogi ljudi vjeruju da je primarna svrha fakulteta pripremiti osobu za određenu karijeru. Fakulteti općenito nude nekoliko programa koji se mogu upisati, ali fakultetsko obrazovanje je više od "samog dobivanja posla". Jedna od važnijih stvari je razvoj kritičkog razmišljanja i rješavanja problema. U stvarnom svijetu vjerojatno će biti situacija kada će osoba ostati sama i u tom trenutku mora biti sposobna shvatiti što je problem i kako ga riješiti [2].

Na današnjem "konkurentnom tržištu rada", fakultetska diploma može dati prednost koja je potrebna kako bi se osoba izdvojila u moru ostalih kandidata. Iako je za dosta poslova dovoljna srednjoškolska diploma, poslodavci često preferiraju kandidate s fakultetskim obrazovanjem. U zadnje vrijeme mnogo fakulteta nudi i zahtjeva stažiranje ili obavljanje nekog oblika prakse čime se zapravo pruža prilika osobi dati uvid u buduću profesiju [2].

Svaka industrija više-manje zahtjeva da osoba bude u toku s trendovima. Ako je prisutna svjesnost o najnovijima dostignućima i osim toga ako postoji šansa da se primjene u radu time je zapravo pokazana želja za povećanjem produktivnosti i učinkovitosti. Praćenje trendova obično radi razliku i privlači tržište. Na fakultetima su profesori većinom dobro povezani i informirani o najnovijim trendovima u industriji [2].

Fakultet pruža pristup vrijednim resursima tijekom studiranja pa i nakon diplomiranja. Jedan od primjera su sveučilišne knjižnice koje često nude širok izbor knjiga i drugih materijala. Također nude pomoći u istraživanju od strane knjižničara koji su stručnjaci u pronalaženju potrebnih informacija. Centar karijera je još jedno vrijedno sredstvo koje može pomoći u domenama poput pisanja životopisa pa do vježbanja razgovora za posao [2].

Pojedini ljudi odlaze na fakultet iz razloga jer do tog trenutka još ne znaju što će biti njihov posao i kako će izgledati njihova karijera. U tom slučaju fakultet može biti korisna odskočna daska ako osoba nije sigurna što želi raditi sa svojim životom. Istraživanjem različitih područja studija može se više saznati o svojim interesima i samim time prepoznati koja područja su vrijedna daljnog istraživanja. I onda naknadno to može biti izvrsna prilika za istraživanje ili specijalizaciju područja koje je predmet interesa [2].

Osim toga što fakultet nudi određeni oblik akademskog učenja, on također nudi priliku razvoja osobnih i profesionalnih vještina. Na primjer, učinkovite komunikacijske vještine i vještine upravljanja vremenom ključne su za uspjeh u svakoj karijeri. Izvannastavne aktivnosti također mogu pomoći u razvoju liderskih vještina ili vještina timskog rada [2].

Fakultet predstavlja veliku promjenu jer je to uglavnom za većinu studenata prvi put da žive daleko od doma i sami donose odluke. To podrazumijeva veliku prilagodbu, ali ujedno i uzbudljivo vrijeme za stjecanje neovisnosti. Kako teče proces rasta tako se i osoba mijenja. U svemu tome postoje trenutci kada se pojavi osjećaj izgubljenosti ili nesigurnosti u vezi budućnosti (pokazatelj da osoba brine o sebi) [2].

I za kraj jedna od zanimljivijih prednosti je prilika za putovanje i upoznavanje drugih drugih kultura. Mnogi fakulteti nude programe studiranja u inozemstvu koji studentima omogućuju da provedu semestar (ili dulje) živeći u drugoj zemlji. To se može smatrati vrijednim iskustvom jer omogućuje izravan uvid u nove običaje i perspektive [2].

## 2.2. Erasmus+ studenska mobilnost

Erasmus+ je program Europske unije za potporu obrazovanju, osposobljavanju, mlađima i sportu u Evropi. Program za razdoblje 2021. do 2027. godine snažno se fokusira na socijalnu uključenost, zelenu i digitalnu tranziciju te promicanje sudjelovanja mladih u demokratskom životu. Erasmus+ program nudi mogućnosti i prilike u područjima kao što su: visoko obrazovanje, strukovno obrazovanje i osposobljavanje, školsko obrazovanje, obrazovanje odraslih, mlađi i sport. Na stranicama [erasmus-plus.ec.europa.eu](http://erasmus-plus.ec.europa.eu) postoje izvješća o državama s interaktivnim podacima o razmjennama i projektima suradnje za sljedeće države članice: Austrija, Belgija, Bugarska, Hrvatska, Cipar, Češka, Danska, Estonija, Finska, Francuska, Grčka, Mađarska, Island, Irska, Italija, Latvija, Lihtenštajn, Litva, Luksemburg, Malta, Sjeverna Makedonija, Nizozemska, Njemačka, Norveška, Poljska, Portugal, Rumunjska, Srbija, Slovačka, Slovenija, Španjolska, Švedska i Turska [3].

U nastavku su detaljnije razmotreni podaci o Erasmus+ programu za Hrvatsku u 2023. godini. Hrvatska je jedna od 27 zemalja EU-a, čime ima pristup svim Erasmus+ aktivnostima kao što su razmjene, osposobljavanja, financiranja i slično. U nastavku za 2023. godinu su doneseni sljedeći zaključci [4]:

- U 2023. godini najveći broj projekata mobilnosti bio je u području školskog obrazovanja a slijede ga strukovno učenje i obrazovanje,
- Najveći udio sredstava u projektima mobilnosti potrošen je u sektoru visokog obrazovanja,
- Broj sudionika koji putuju u Hrvatsku povećava se svake godine,
- Tijekom 2023. godine došlo je do smanjenja broja sudionika koji su odlazili na razmjene u inozemstvo,
- Digitalna transformacija istaknuta je kao glavni prioritet u ukupnim bespovratnim sredstvima u 2023. godini.

Boravak u inozemstvu predstavlja izazovnu stvar koju poslodavci cijene i uzimaju u obzir prilikom intervjuiranja. Svaka osoba se može prijaviti na Erasmus+ program tako da se započetek obrati međunarodnom ili Erasmus+ uredu matičnog sveučilišta ili visokoškolske ustanove. Visokoškolska ustanova pošiljateljica dužna je odabrati kandidate na pošten i transparentan način. Maksimalno ukupno trajanje boravka u inozemstvu je 12 mjeseci unutar jednog studijskog ciklusa i shodno tome moguće je obaviti više od jedne razmjene unutar tog ograničenja. U nastavku se spominju dvije vrste mobilnosti [5]:

- Dugoročna mobilnost - minimalno 2 mjeseca do maksimalno 12 mjeseci u inozemstvu,
- Kratkotrajna mobilnost - između 5 i 30 dana u inozemstvu.

## 2.2.1. Izazovi i strahovi

Erasmus+ predstavlja uzbudljivo iskustvo koje je istovremeno veliki skok u nepoznato. Takvo iskustvo zahtijeva da osoba ostavi sve prethodno iza sebe i da se suoči s potpuno novim stvarima s kojima se prethodno nije imala prilike susresti. Na „Blogu Erasmus generacije“ navodi se nekoliko strahova koje Erasmus+ studenti mogu imati [6]:

- **Odlazak iz doma** - ako je napravljen popis za i protiv prije nego što se je predala prijava za Erasmus+, onda je odlazak iz svog doma bio poprilično visoko na lošoj strani popisa. Taj loš osjećaj traje neko vrijeme ali nakon par provedenih dana kreće upuštanje u avanturu „ograničenu vremenom“. U nekim trenutcima zna se pojaviti osjećaj kada nedostaje sve ono od čega se je otišlo (to je u situacijama kada je osoba sama) ili kada se razgovara s bližnjima (dobiva se osjećaj da se propušta nešto dok je osoba na drugom mjestu). „Biti daleko od onih s kojima ste nekada dijelili sve je pravi izazov, ali kada se vratite kući i otkrijte tko je prošao test „udaljenost“, shvatit ćete tko su vam pravi prijatelji“.
- **Finacijska ne(odgovornost)** - pri početku Erasmusa+ dobiva se veći dio stipendije koji je namijenjen za život tijekom cijelog Erasmusa+. Bitno je da osoba ima vještinu upravljanja velikom količinom novca i da racionalno troši novce na potrebne stvari. Ovdje je posebno izazovno jer se radi o trošenju novca u stranoj zemlji pa se onda obično u početku potroše veće količine novca (npr. radi nepredvidivih troškova). Najbolja je stvar na početku donesti odluku što se želi raditi s novcem. Jesu li to putovanja, uživanje u hrani, tjedna kupovina, izlasci itd... Zanimljiva činjenica je da obično postoji želja za kombiniranjem/isprobavanjem svega.
- **Jezična barijera** - nedostatak povjerenja u znanje stranih jezika predstavlja jedan od glavnih razloga zašto studenti ne sudjeluju u programima kao što je Erasmus+. Pretjerana briga oko toga nema smisla jer Erasmus+ zahtijeva osnovno znanje stranog jezika i nije problem ako neko nije profesionalac u tome. Erasmus+ je upravo dobra prilika da se razbije ta barijera i nauči novi jezik.
- **Akademski život** - postoje studenti koji nikada prije nisu imali nastavu ili prezentiranje materijala na stranom jeziku. Bez obzira na to koliko osoba dobro zna jezik, učenje o svojoj budućoj profesiji na stranom jeziku može biti nezgodno. Upisuju se kolegiji o kojima se vrlo vjerojatno malo ili ništa ne zna, gdje predaju profesori koji nikada prije nisu viđeni. Bez obzira koliko god je to izazovno, iskušavanje drugačijeg načina podučavanja u kompletno drugačijem sustavnom obrazovanju je neprocjenjivo iskustvo.
- **Pakiranje** - je vrlo bitan faktor koji se općenito veže za svaku vrstu putovanja (trodnevni izlet, dvojtedni odmor) a u pravilu znači pakiranje cijelog života u jedan kofer. Pakiranjem se zapravo dobiva svijest o tome koliko stvari osoba posjeduje i koristi na dnevnoj bazi.

U nastavku su prikazani primjeri prijašnjih Erasmus+ studenta koji pričaju konkretno o prvom izazovu koji je naveden na prošloj stranici (Odlazak iz doma) [7]:

**Ixan Curtamet – Rumunjska**

Tijekom mog Erasmusa+ imao sam osjećaj da mi nedostaje dom. Osjećao sam da propuštam puno toga u svojoj državi, falili su mi moji prijatelji. Imao sam periode kada baš nisam uživao na Erasmusu+ jer sam se osjećao loše zbog toga što sam bio tužan.

**Katarina Bižanović – Hrvatska**

U početku mi nije nedostajao dom. Pogodilo me nakon 3 ili 4 mjeseca kada su mi prijatelji počeli nedostajati, a to je obično bilo za vrijeme praznika, jer sam tada bila sama u stranoj državi.

**Agnese Smiroldo – Italija**

Nedostajao mi je dom jer su mi nedostajale određene kulturne stvari poput hrane. Falio mi je i moj tata.

**Marc Dolcet Sadurni – Estonija**

U početku mi je falio dom, ali uglavnom se puno toga događalo u isto vrijeme i nisam imao vremena razmišljati o tome. Uživao sam puno i upoznao sam mnogo ljudi.

**Vitor Bizarro – Portugal**

Mislim da je nemoguće da ti fali dom kad si na Erasmusu+. Kuhao sam kao kod kuće, družio se kao kod kuće i stalno sam zvao prijatelje iz svoje države i pritom uživao.

**Minerva Martinez – Italija**

Nije mi nedostajao dom jer su Italija i Španjolska jako slične. Ipak, nakon tri mjeseca znala sam ponekad osjetiti neki loš osjećaj.

**Agnes Mfourmou – Belgija**

Nije mi nedostajao dom jer sam stalno bila u kontaktu s prijateljima, dopisivali smo se, zvala sam mamu, slala im slike i videe. Morate uživati koliko god možete, jer će vam kasnije biti žao što odlazite s Erasmusa+ i tada će vam nedostajati Erasmus+ dom. To je zapravo samo loš osjećaj jer se ne nalazite na nekom mjestu u nekom trenutku.

### **3. Aktualne tehnologije i alati za uspješno snalaženje i informiranje**

Unutar ovog poglavlja opisana je važnost inteligentnih pomoćnih sustava te je napravljena podijela na asistente, copilote, agente i chatbotove. Svaki od njih je kratko opisan i navedeno je nekoliko primjera. Za kraj ovog poglavlja opisano je korištenje inteligentnih asistenata u obrazovnom sustavu, njihova primjena i koji su izazovi takvog korištenja.

#### **3.1. Važnost inteligentnih pomoćnih sustava**

Davne 1961. godine IBM je predstavio prvi sustav koji je mogao prepoznati govor (izgovorene znamenke). Do 1990-ih kreirani su prvi komercijalni osobni asistenti koji su se aktivirali na postojanje glasa. Godine 2010. svijet je prvi put čuo za Siriјa, Appleovog osobnog asistenta koji je pokrenut na temelju umjetne inteligencije. U to vrijeme Siri je predstavljao revolucionarno rješenje. Petnaest godina kasnije, korisnici su zatrpani s ogromnom količinom inteligentnih asistenta. Korisničko iskustvo (eng. User experience) je sada više nego ikada važno. Održavanje zadovoljstva korisnika/kupca i upravljanje njihovim očekivanjima zahtijeva stalne inovacije i poboljšanja. Korisnici žele da stranice budu intuitivne i učinkovite te da na neki način zadovolje njihove potrebe. AI asistenti su postali popularni alati koji pomažu u poboljšanju korisničkog iskustva. Točnije rečeno dodaju novi način interakcije između korisnika i pružatelja usluge i samim time povećavaju angažman, zadovoljstvo ili profit.

U svijetu umjetne inteligencije često se pojavljuju pojmovi poput bota, asistenta, copilota i agenta. Iako na prvi pogled djeluju kao sinonimi, svaki od njih ima svoju specifičnu svrhu, primjenu i skup prednosti.

Tip	Ključne značajke	Primjena	Primjer
Asistent	Personalizacija i multitasking	Dnevna produktivnost	Siri
Copilot	Specijalizirana podrška	Razvoj, IT	GitHub Copilot
Agent	Autonomija i proaktivnost	Upravljanje prijevarama, korisničko iskustvo	OpenAI Codex
Chatbot	Jednostavni ponavljajući zadaci	Osnovna korisnička podrška	Zendesk

Tablica 1: Razlika između pojmova [8]

### 3.1.1. Asistent

AI asistenti donose inteligenciju automatizaciji razumijevanjem konteksta i time se personalizira interakcija kako bi se mogao obaviti niz zadatka. Ovi sustavi nadilaze skriptirane odgovore koristeći tehnologije poput obrade prirodnog jezika (eng. Natural processing language) ili strojnog učenja (eng. Machine learning). Kontekstualno su svjesni i razumiju korisnički unos što omogućava vođenje višestrukih razgovora. Integriraju se s alatima poput kalendarja, aplikacija za razmjenu poruka i poslovnim sustavima. Prilagođava se ponašanju korisniku kako bi se s vremenom poboljšao. Npr. mogu se koristiti za zakazivanje sastanaka, upravljanje e-poštom ili prilagođavanje rasvjete i temperature u pametnim domovima [8].

U nastavku su navedene prednosti inteligentnih asistenata [9]:

- **Brzina i jednostavnost** - AI asistenti omogućuju korisnicima da brzo i jednostavno dobiju ono što im je potrebno, smanjujući gubitak vremena. TikTok je dobar primjer kako usluga može neindirektno zadržati korisnika na njihovoј platformi jer su razvili mehanizam koji je naučio prepostavljati što korisnik želi gledati.
- **Pružanje podrške 24/7** - AI asistenti su dostupni stalno, čime je omogućena podrška korisnicima kad god je to potrebno.
- **Prihvaćanje od strane ljudi** - razgovor s AI asistentima se odvija kroz korisnička sučelja, gdje korisnici mogu komunicirati na ljudski razumljiv način s asistentima.

#### 3.1.1.1. Siri

Siri je Appleov virtualni asistent za IOS, macOS, tvOS, i watchOS uređaje koji se koristi preko glasa i pokreće ga umjetna inteligencija. Siri odgovara na izgovorena pitanja korisnika tako što im odgovara putem zvučnika uređaja i prikazuje paralelno relevantne informacije na početnom zaslonu iz određenih aplikacija kao što je Kalendar. Siri se počeo izradivati u neprofitnom istraživačkom institutu sa sjedištem u Kaliforniji nakon čega su pojedini članovi odlučili osnovati startup tvrtku pod nazivom Siri. Godine 2010. Apple je preuzeo Siri što je dovelo do brze integracije u IOS uređaje. Tijekom 2011. i 2012. godine Siri je bio na Iphone 4S i na iPAD-ovima treće generacije. U lipnju 2024. Apple je na konferenciji WWDC 2024 najavio značajnu reviziju Sirija gdje je predstavljen Apple Intelligence koji predstavlja novi osobni AI sustav koji poboljšava Sirijeve mogućnosti. U početku glas Sirija je bio od umjetnice Susan Bennet. Zanimljiv slučaja je bio da sve dok joj nije prijateljica javila, Bennet nije bila svjesna da je postala Sirin glas. Iako Apple nikada nije priznao da je Bennet originalni Siirin glas, audio stručnjaci na CNN-U su to potvrdili. Karen Jacobsen je žena koja je posudila svoj originalni ženski glas za GPS sustave, dok je Jon Biggs, posudio svoj glas za mušku verziju Sirija. U međuvremenu Apple je razvio novi ženski glas za IOS11 pomoću tehnologije dubokog učenja snimanjem i kombiniranjem govora stotine kandidatkinja. Siri može obavljati razne zadatke kao što su: navigacija uputama (npr. Kakav je promet na putu do posla?), zakazivanje događaja i podsjetnika (npr. Slanje poruke za rođendan), pretraživanje weba (npr. Pronađi sliku psa) ili mjenjanje postavki (npr. Povećanje svjetline zaslona) [10].

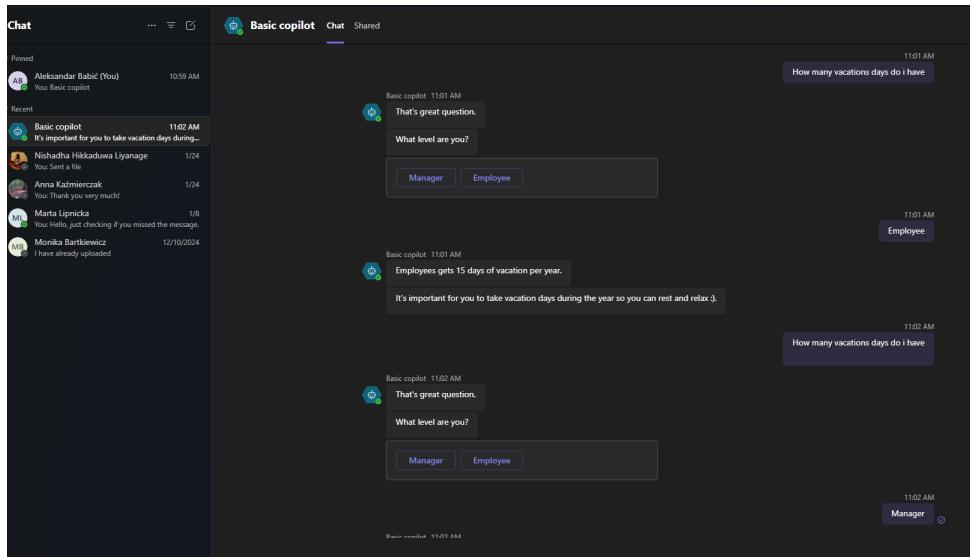
### **3.1.2. Copilot**

Copilot je AI alat koji je u ulozi suradnika i primarno je osmišljen za poboljšanje ljudskih sposobnosti u složenijim zadacima. Za razliku od asistenata opće namjene, copiloti se usredotočuju na specifične domene, nudeći stručnost i proaktivnu podršku. Duboko su ugrađeni u specifične platforme čime pružaju podršku. Predviđaju korisničke potrebe analiziranjem konteksta i nude relevantne prijedloge ili rješenja. Na primjer, Github Copilot se koristi u razvoju softvera gdje predlaže isječke koda, otkriva pogreške i ubrzava razvoj automatizacijom rutinskih zadataka [8].

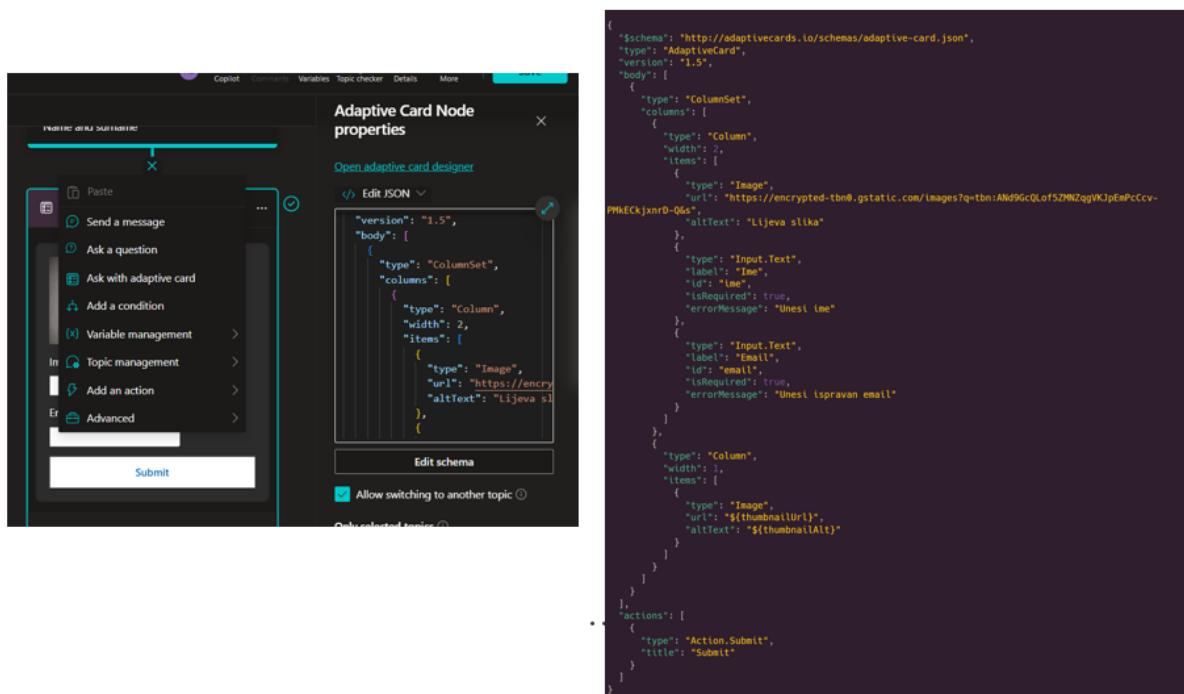
#### **3.1.2.1. Microsoft Copilot Studio**

Tijekom studentske prakse u jednoj varaždinskoj firmi bila je prilika raditi i istraživati alat Microsoft Copilot Studio. Rukovodstvo te tvrtke u tom trenutku je imalo mišljenje da korištenjem AI alata se znatno povećava produktivnost zaposlenika. Primarno bio je zadatak napraviti razna istraživanja u AI domeni jer su svi popularniji AI sustavi primjeri nekog oblika općenitog znanja. Ni jedan od njih ne zna ništa o poslovanju i domeni kojim se firma bavi. Unutar Copilot Studija mogu se razvijati copiloti u područjima kao što su: prodaja i podrška, informacije (podaci o otvaranju trgovina), zdravlje zaposlenika i godišnji odmori. Također, unutar sustava postoji nekolicina kreiranih predložaka s copilotima koji odgovaraju na karakteristična pitanja poput IT Helpdeska ili Team Navigatora. Primarno taj sustav je razvijen od strane Microsofta i koristi se za izradu kopilota koji mogu odgovarati na specifična pitanja koja su usko vezana za neku domenu. U alatu se je isprobalo korištenje varijabli, entiteta, petlji, tijeka rada (eng. Workflow) ali ono što je bilo najzanimljivije je baza znanja (eng. Knowledge base) gdje se može spremiti velika količina podataka koja je baza za odgovore. Osim toga kada se kreira agent on se može ugraditi u Microsoft Teams ili u web stranicu. Microsoft Copilot studio se bazira na kreiranju tema (eng. Topic) i dok god ima definirane teme za određene slučajevе od tamo će crpiti inspiraciju za odgovore. Postoje prilagođene (eng. Custom) i sistemske (eng. System). Prilagođene teme se koriste za komuniciranje s korisnikom (Dobar dan, doviđenja) dok se sistemske koriste za pozadinske operacije (inicijalna poruka pri početku konverzacije ili završetak razgovora). Kada kopilot nema definiranu temu za odgovor na neko pitanje onda odlazi u sistemske teme i tamo otvara conversational boosting temu gdje je to zapravo tijek rada koji traži informacije iz baze znanja [11]. Kao i svaki sustav, Copilot Studio ima svoje limitacije a to su: max 8000 zahtjeva po minuti u Dataverse okruženju, max datoteka veličine 512 MB, max 500 datoteka, max 200 fraza po temi i max 50 agenata po timu [12].

Osim toga isprobao je i Bing Custom Search koji je vanjski servis koji se može koristiti za navođenje više izvora koji se koriste za odgovaranje na upite. Cilj je da se ne koristi cijeli internet nego da se koristi taj skup podataka koji je naveden. Putem "custom configuration ID-a" može se povezati Custom Bing Search i Copilot Studio. Naknadno isprobana je funkcionalnost Adaptive Cards Designer koji omogućava kreiranje vlastitih UI (eng. User Interface) elemenata koji se mogu koristiti unutar razgovora s korisnikom. Na sljedećoj slici može se vidjeti da je to JSON koji je potrebno importati u Copilot Studio. Na stranici <https://adaptivecards.io/designer/> postoji veliki broj primjera koji su već kreirani i besplatni su za korištenje.



Slika 1: Primjer copilota kreiranog u Microsoft Copilot Studiu



Slika 2: Adaptive cards

Od drugih stvari koje još postoji u Copilot Studiju:

- Dvije vrste kreiranje entiteta (Copilot sadrži veliki broj entiteta koji se mogu koristiti ali se mogu kreirati i vlastiti (Closed List i Regular Expression)),
- Message variation koji omogućava postojanje nekoliko verzija incijalnih poruka koje se pojavljuju kada se copilot pokrene.

### **3.1.3. Agent**

AI agenti predstavljaju vrhunac automatizacije i autonomije jer su sposobni samostalno donositi odluke i poduzimati radnje bez stalnog ljudskog vodstva. Dizajnirani su za složena okruženja a također se dinamički prilagođavaju kako bi postigli definirane ciljeve. Djeluju samostalno kako bi analizirali podatke i odredili prioritetne zadatke. Prate sustave i okruženja, predviđaju i rješavaju potencijalne probleme prije nego što eskaliraju. Pri radu s složenijim problemima, identificiraju optimalnija rješenja. Bankarski agenti zamrzavaju kompromitirane račune kada se otkrije neovlašteni pristup s visokorizične lokacije. Na primjer, prate stanje servera, otkrivaju anomalije, identificiraju kašnjenje pošiljka i proaktivno obavještavaju kupca nudeći naknadu ili alternative i poduzimaju korektivne mjere bez ljudske intervencije [8].

#### **3.1.3.1. ChatGPT Operator**

ChatGPT Operator je agent koji može ići na web kako bi obavljao zadatke umjesto korisnika. Može pregledavati web stranicu i komunicirati s njom tipkanjem, klikanjem i pomicanjem. Pojavio se je u siječnju 2025. godine i trenutno je još u fazi razvoja gdje se nastoji poboljšat rješenje na temelju povratnih informacija korisnika. Od Operatora se može zatražiti da obavi širok raspon repetitivnih zadataka u pregledniku, kao što je ispunjavanje formi, naručivanje proizvoda, pa čak i stvaranje memeova. Operator je dostupan unutar PRO verzije na području SAD-a i pokreće se na temelju modela CUA (eng. Computer-Using Agent). CUA je obučen za interakciju s grafičkim korisničkim sučeljima (eng. Graphical User Interface) - gumbima, izbornicima i tekstualnim poljima koje ljudi vide na zaslonu. Ako se dogodi da se Operator susretne s izazovima ili ako napravi pogreške, u tom slučaju vraća kontrolu korisniku, osiguravajući glatko i korisničko iskustvo [13].

U nastavku se navodi jedan primjer koji ChatGPT Operator obavlja [14]:

1. Upravlja preglednikom i obavlja radnje u ime korisnika.
2. Analizira podatke na temelju YouTube veze.
3. Generira PNG sliku analize.
4. Ide na X.com, lokalno odabire sliku i prenosi je.
5. Lajka i komentira sliku.
6. Često traži potvrdu prije izvođenja radnji.
7. Može se koristiti za pretraživanje weba, kupnju ulaznica ili rezervaciju restorana.

### **3.1.4. Chatbot**

Chatbotovi su sustavi temeljeni na kreiranim pravilima koji su osmišljena za izvršavanje određenih zadataka putem unaprijed definirane interakcije. Iako posjeduju dosta ograničenja kako se dobro i brzo snalaze u dosljednom rješavanju jednostavnih repetativnih zadataka. Dje luju na temelju stabla odlučivanja ili prepoznavanja ključnih riječi ograničavajući svoju interakciju na unaprijed postavljene scenarije. Koriste se za automatizaciju zadataka gdje se obrađuje veliki broj ponavljujućih upita poput rezervacija ili narudžbi. Njihova mana je da se ne mogu prilagoditi izvan svog programiranja što ih čini neprikladnim za složenije zadatke [8].

#### **3.1.4.1. Zendesk**

Zendesk je sustav koji je namijenjen za slanje zahtjeva putem e-pošte i pomaže tvrtka u praćenju, određivanju prioriteta i rješavanju zahtjeva za korisničku podršku. Može pomoći u prikupljanju zahtjeva korisnika koji su poslani putem više kanala, uključujući e-poštu, chat ili društvene mreže i kroz nadzornu ploču pratiti status takvih zahtjeva [15].

## **3.2. Inteligentni asistenti u obrazovnom sustavu**

Današnje obrazovanje se sve više oslanja na tehnologiju, a inteligentni asistenti postaju alat koji može poboljšati iskustvo učenja i poučavanja. Na primjer, svaki učenik može imati personaliziranu pomoć, a učitelji imaju više vremena za fokusiranje na ono što je zaista važno. Takvi asistenti mogu prilagodavati materijale, odgovarati na pitanja, ocjenjivati domaće zadaće čime se rasterećuje profesore. Jedna od najvećih prednosti koje donose inteligentni asistenti je personalizirano učenje. Umjesto univerzalnog pristupa, asistenti se mogu individualno posvetiti svakom učeniku/studentu.

### **3.2.1. Primjena inteligentnih asistenata u obrazovanju**

**Automatizacija procesa ocjenjivanja** - ocjenjivanje pismenih zadataka, pružajući detaljne povratne informacije u roku od nekoliko minuta umjesto dana. Osim što skraćuje vrijeme ocjenjivanja profesorima, učenici dobivaju brže povratne informacije o svom radu. Jedan od primjera je Gradescope kojeg je razvio Turnitin. Kroz Gradescope moguće je ocjenjivati zadatke iz različitih predmeta, uključujući programiranje, fiziku, matematiku, kemiju, biologiju i ekonomiju. Također moguće je ocjenjivati programske projekte (u digitalnom obliku) i zadatke napisane na papiru. Neki od komentara profesora koji su koristili Gradescope su [16]:

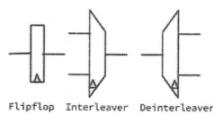
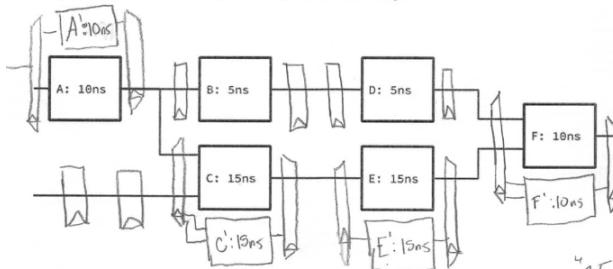
- "S Gradenkopom je čast ocjenjivati. Prije što sam radio u rasponu od 2 do 3 sata sada odradim u 15 minuta" - Romulo Chumacero - University of Chile
- "Statistika mi stvarno pomaže da shvatim što sljedeći put mogu drugačije objasniti kako bih pomogla svojim učenicima da bolje uče" - Katie Johnson - Florida Gulf Coast University
- "Gradescope omogućuje da svaki dan svojoj grupi od 60 učenika dam kratki kviz i da ih sve ocijenim tijekom 30-minutne vožnje vlakom kući." - Jesse Tov - Northwestern University

**Smanjenje administrativnog opterećenja** - na primjer, sustavi pokretani umjetnom inteligencijom mogu pratiti prisutnost, upravljati zadacima pa čak i predviđati kojim studentima je potrebna dodatna pomoć. Takav pristup pomaže u otkrivanju problema prije nego što postane još veći problem. Jedan od zanimljivijih citata od Rose Luckin, profesorce dizajna usmjerenog na učenike na University College London, je: "Prava moć inteligentnih agenata u obrazovanju ne leži u zamjeni učitelja, već u pojačavanju njihovog utjecaja. Obavljanjem rutinskih zadataka, AI asistenti oslobađaju edukatore da se usredotoče na ljudske elemente poučavanja, inspiraciju, mentorstvo i njegovanje vještina kritičkog razmišljanja" [17].

**Prilagođavanje nastavnih materijala** - analiziranje snaga, slabosti i načina učenja svakog učenika kako bi se stvorili prilagođeni materijali. Na primjer, ako postoji učenik koji se muči s razlomcima, asistent može pružiti dodatne resurse kako bi se poboljšalo njegovo razumijevanje [17].

**Question 2 (points total):**

Pipeline the circuit below. Optimize throughput (inputs processed per ns) and cost (\$). You may add any number of the blocks in the circuit (A-F), edge-triggered flipflops, edge-triggered interleavers, and edge-triggered de-interleavers (see figure below). No other modifications are permitted. The latency of flipflops, interleavers, and de-interleavers is 0ns. The cost of any item (whether already there or one you add), regardless of which item is \$1. You have a total budget of \$25 and the components already drawn below cost \$6. Draw your answer on top of the diagram below if at all possible.



$$\text{Cost} = \$6 + \$18 = \$24$$

$$\text{Latency} = 7.5 \times 8 = 60\text{ns}$$

$$\text{Throughput} = 1/60\text{ns}$$

If we add a latch at the output,

Total Points

**24.0 / 33.0 pts**

**1** +6.0

Correct interleaving concept

**2** +6.0

Correct interleaving of multiple inputs

**3** -3.0

Abused interleavers (too many outputs/inputs)

**4** +6.0

Well-formed ("balanced" latch count)

**5** +8.0

Correct extra stages for long-latency interleaved approach

**6** -4.0

Not including latch for each input / extra latches if assuming interleavers latch

Slika 3: Ocjenjivanje zadatka - Gradescope [16]

### 3.2.2. Izazovi inteligentnih asistenata u obrazovanju

Jedan od najvećih izazova je pitanje privatnosti podataka. Obično sustavi zahtijevaju ogromne količine podataka o studentima da bi učinkovito funkcionali, što ukazuje na opasnost od potencijalne zlouporabe tih podataka. Na primjer, studija Svučilišta u Michiganu iz 2020. godine otkrila je da 86 posto obrazovnih aplikacija umjetne inteligencije prikuplja osobne podatke od studenta, a samo 42 posto ima jasne politike privatnosti [17].

Asistenti su dobri onoliko koliko su dobri podaci na kojima su obučeni. U obrazovnom sustavu, osiguranje kvalitete i relevantnosti tih podataka je od najveće važnosti. Kako bi rješenje bilo zadovoljavajuće škole i programeri moraju surađivati kako bi stvorili sustave koji su ne samo učinkoviti, već i etični i sigurni [17].

Implementacija inteligentnih asistenata zahtijeva značajnu tehnološku infrastrukturu. Mnoge škole, posebno u ruralnim područjima, nemaju potreban hardver, brzi internet ili softver za podršku ovim sustavima. Izvješće za obrazovanje iz 2022. godine kaže da gotovo 17 milijuna učenika u SAD-u još uvijek nema pristup internetu kod kuće [17].

Unatoč tim izazovima, integracija inteligentnih asistenata u obrazovni sustav donosi brojne prednosti. Sustavi pokretani umjetnom inteligencijom mogu automatizirati dugotrajne administrativne zadatke, oslobađajući učitelje da se usredotoče na ono što najbolje rade: inspiriranje i poučavanje učenika. Osim toga, AI asistenti mogu pružiti personalizirano učenje, povećanu produktivnost i učinkovitost, te poboljšano korisničko iskustvo [17].

Kroz sljedeće poglavje je prikazano kako funkcioniraju inteligentni asistenti temeljeni na RAG (eng. Retrieval-Augmented generation) tehnologiji.

## 4. Retrieval Augmented Generation metoda

U ovom poglavlju obrađena je RAG (eng. Retrieval Augmented Generation) metoda, koja predstavlja suvremenii pristup u području obrade prirodnog jezika i kombinira sposobnosti generativnih modela s mehanizmima za dohvaćanje informacija iz vanjskih izvora. Poglavlje započinje pregledom povijesti i ključnih prednosti RAG pristupa, zatim slijedi teorijska osnova modela i njegova arhitektura kroz faze dohvaćanja i generiranja. Nadalje, razrađeni su tehnički aspekti implementacije, uključujući vrste retrievera, generativne modele i metode rangiranja rezultata, kao i integraciju s vektorskim bazama podataka. Poseban naglasak stavljen je na primjenu RAG sustava, postupke treniranja i fino podešavanje modela, kao i izazove poput halucinacije i ažurnosti podataka. Na kraju predstavljene su dostupne biblioteke i alati koji olakšavaju razvoj RAG rješenja u praksi.

### 4.1. Uvod u RAG

RAG je proces optimizacije izlaza velikog jezičnog modela pri čemu se referencira na autoritativnu bazu znanja izvan izvora podataka za obuku prije generiranje odgovora. Veliki jezični modeli (eng. Large Language Models - LLM) obučavaju se na ogromnim količinama podataka i koriste milijarde parametara za generiranje izlaza za zadatke poput odgovaranja pitanja, prevodenja jezika i dovršavanja rečenica. Ideja RAG-a je da proširi već moćne mogućnosti LLM-ova na određene specifične domene, a sve to bez potrebe za ponovnom obukom modela. Takav pristup poboljšava LLM-ove u smislu točnosti, korisnosti i relevantnosti [18].

Model velikog jezika može se zamisliti kao entuzijastičnog zaposlenika koji odbija biti informiran o aktualnim događajima, ali uvijek odgovara na pitanja s visokim samopouzdanjem. Nažalost, takav stav može negativno utjecati na povjerenje korisnika i nije nešto što bi trebalo postojati kod AI sustava. Iz takvih razloga RAG tehnologija preusmjerava LLM kako bi dohvatio relevantne informacije iz unaprijed definiranih izvora znanja [18].

Bez RAG-a, LLM uzima korisnički unos i stvara odgovor na temelju informacija na kojima je obučen. S RAG-om se uvodi komponenta za pronalaženje informacija koja koristi korisnički unos kako bi prvo privukla informacije iz novog izvora podataka. Nakon toga LLM kombinira novo znanje i svoje podatke za stvaranje boljih odgovora [18].

Primjerice, ako se postavi pitanje: "Koji planet u Sunčevom sustavu ima najviše satelita?", osoba može odgovoriti: "Odlično pitanje! Kao dijete sam volio astronomiju i pročitao sam članak o tome - mislim da je to Jupiter s 88 satelita." Međutim, takav odgovor ima nekoliko problema: osoba ne navodi konkretni izvor, iako nastupa s visokim samopouzdanjem ("ja to znam, pročitao sam") te informacija može biti zastarjela jer se temelji na sjećanju iz prošlosti [19].

Kada bi osoba rekla: "Idem prvo na NASA-inu stranicu i tamo ću potražiti tu informaciju", tada bi traženje odgovora bilo temeljeno na provjerenom i ažurnom izvoru. Nasuprot tome, LLM-ovi, poput GPT-a, često odgovaraju vrlo samouvjereni, primjerice: "Na temelju po-

dataka na kojima sam treniran, odgovor je Jupiter." Iako zvuči uvjerljivo, takav odgovor može biti netočan, a korisnik često nije svjestan ograničenja modela ni datuma do kojeg je model treniran. U ovom slučaju, točan odgovor zapravo može biti Saturn, koji trenutno ima 114 poznatih prirodnih satelita. Rješenje za ovaj problem je korištenje spremišta znanja (eng. content store) koje može sadržavati izvore poput internetskih stranica, PDF dokumenta ili baza podataka. LLM tada prvo pristupa tom vanjskom spremištu informacija, pretražuje ga, identificira relevantne sadržaje i tek tada generira odgovor na temelju tih aktualnih podataka. U RAG sustavu prompt se najčešće sastoji od tri dijela: instrukcija (što model treba učiniti), korisnikovog upita (pitanje ili zahtjeva) i informacija koje su dohvaćene iz spremišta znanja [19].

Još jedan primjer je situacija u kojoj Alice želi saznati koliko dana porodiljnog dopusta može dobiti. AI sustav koji ne koristi RAG odgovara veselo i netočno "Uzmi koliko želiš". Politike o porodiljnog dopustu su složene i razlikuju se ovisno o državi u kojoj se boravi. U ovom slučaju, veliki jezični model (LLM), umjesto da prizna ograničenje u znanju odgovarajući "Žao mi je, ne znam", generirao je frazu iz svog skupa za treniranje koristeći korisnički orijentiran, ali neutemeljen jezik [20].

#### 4.1.1. Povijest i razvoj

Korijeni RAG-a mogu se pratiti od 1950-ih i 1960-ih kada su se koristili sustavi za pronađenje informacija. Istraživači Hans Peter Luhn i Gerald Salton postavili su temelje za modele vektorskog prostora omogućujući računalima da pronađu dokumente koji su relevantni za korisničke upite. Obrada prirodnog jezika značajnije je napredovala u 1980-ima i 1990-ima putem statističkih jezičnih modela, poput n-gram modela [21].

Integracija pretraživanja i generiranja pojavila se u 2000-ima s razvojem sustava poput IBM Watsona, koji je koristio metodu pretraživanja za prikupljanje informacija i statističke modele za generiranje odgovora. Razvoj neuronskih mreža u 2010-ima dodatno je ubrzao ovu integraciju. Transformatori omogućuju značajan napredak u modeliranju jezika, dok tehnike gustog pretraživanja, poput pretraživanja temeljenog na ugrađivanju, dodatno poboljšavaju točnost pretraživanja [21].

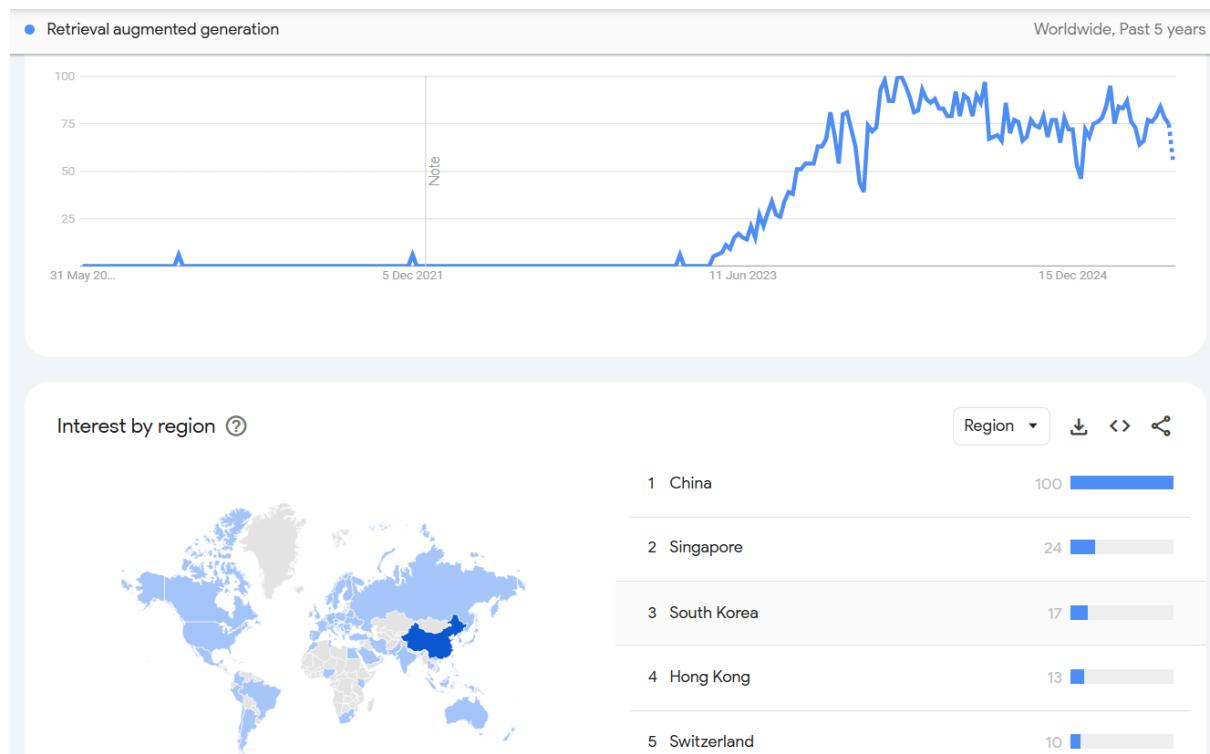
RAG se pojavljuje u radu "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks" 2020. godine, gdje su autori predložili sustav u kojem vanjski model dohvaća relevantne dokumente kao odgovor na upit, a generativni model uvjetovan tim dohvaćenim znanjem daje odgovor. U radu su autori koristili bi-enkoder za pretraživanje, temeljen na pretraživanju gustih prolaza (eng. Dense Passage Retrieval) i T5 transformator kao generativni model, pokazujući izvanredne performanse na testovima koji zahtijevaju puno znanja. Ovaj pristup rješava ključne izazove s kojima se suočavaju modeli velikih jezika, poput halucinacija i zastarijelog znanja [21].

Objavljivanje RAG okvira potaknulo je nagli rast istraživanja, gdje je napravljeno nekoliko napredaka koja su se nadovezala na izvorni koncept [21]:

- **Poboljšani modeli pretraživanja** - modeli neuronskog pretraživanja poput ColBERTa postaju standard u RAG sustavima,

- **Skalabilnost i učinkovitost** - kako bi se obradile ogromne količine podataka. Tehnike poput pretraživanja najbližeg susjeda (ANN) i destilacije modela pomogle su u optimizaciji performansi,
- **Primjene specifične za domenu** - do 2023. godine RAG sustavi su se prilagođavali specifičnim slučajevima upotrebe, kao što su pravne i medicinske domene, gdje su točnost i objašnjivost ključne stavke. Dodatno se je istraživalo fino podešavanje kako bi se dobio što precizniji odgovor,
- **Multimodalni RAG** - proširivanje RAG-a kako bi uključio tekst, slike, zvuk i video. Proširivanje na područja poput obrazovanja i zabave,
- **Prilagođeno učenje i petlje povratnih informacija** - prilagodba prema korisnicima je ostvarena kroz razumijevanje njihovih povratnih informacija u stvarnom vremenu.

Kroz povijest RAG-a pokazuje se kako su se temeljna istraživanja u odvojenim područjima, poput pretraživanja i generiranja podataka, spojila kako bi se riješili izazovi u zadacima koji zahtijevaju puno znanja. Također može se vidjeti važnost interdisciplinirane suradnje u području umjetne inteligencije, gdje se otkrića često javljaju kombiniranjem različitih metodologija [21].



Slika 4: Popularnost RAG-a; Izvor

Važnost RAG-a povećala se posljednjih godina zbog pojave velikog broja aplikacija temeljenih na umjetnoj inteligenciji. Na slici 4. se može vidjeti kako uzlazna putanja popularnosti RAG-a kreće od petog mjeseca 2023. godine.

## 4.2. Arhitektura RAG modela

RAG se sastoji od faze pronalaženja i faze generiranja sadržaja. U fazi pronalaženja algoritmi traže relevantne informacije za korisnikov upit dok se u fazi generiranja generira tekst kako bi se odgovorilo na korisnički upit. U okruženju otvorene domene te činjenice uglavnom dolaze iz indeksiranih dokumenata na internetu. U okruženju zatvorene domene obično se koristi uži skup izvora radi dodatne sigurnosti i pouzdanosti. Prije LLM-a, agenti su kroz razgovor pratili ručni tijek dijaloga. Pretpostavljali su namjeru klijenta, dohvaćali tražene informacije i dostavljali odgovor za određeni scenarij. Za jednostavne upite, ova ručna metoda stabla odlučivanja funkcionalala je sasvim dobro. Problem takvog pristupa je da se je gubilo vrijeme na pisanje i predviđanje svakog pitanja koje bi korisnik mogao postaviti. U takvim slučajevima, ako tražena informacija nije definirana u dostupnim podacima, chatbot neće biti u mogućnosti pružiti relevantan odgovor[20].

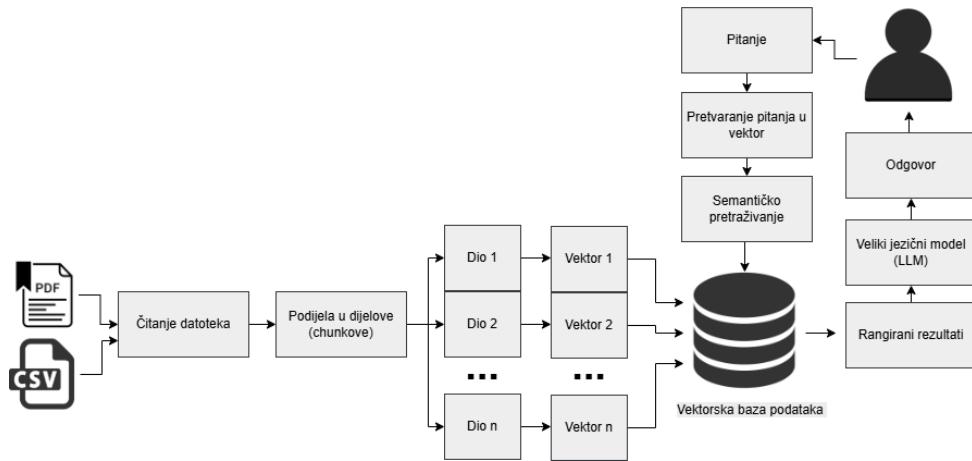
Generalno RAG arhitektura se može objasniti kroz sljedeće karakteristike [18]:

1. **Stvaranje vanjskih podataka** - novi podaci izvan orginalnog skupa podataka koji su se koristili za treniranje LLM-a se zovu vanjski podaci (eng. external data). Oni dolaze iz više izvora podataka kao što su: API-ji ili baze podataka. Podaci mogu postojati u različitim formatima. Jedna od tehnika (eng. embedding language models) pretvara podatke u numerički oblik i sprema ih u vektorsku bazu podataka (kako bi generativni modeli umjetne inteligencije to mogli razumjeti).
2. **Preuzimanje relevantnih informacija** - Korisnički upit pretvara se u vektorski prikaz i uspoređuje se s vektorskим bazama podataka. Na primjer, ako korisnik postavi pitanje chatbotu "Koliko dana godišnjeg odmora imam?", chatbot će morati preuzeti dokumente o politici godišnjeg odmora uz evidenciju prošlih odmora za pojedinog zaposlenika. Takvi dokumenti će biti vraćeni korisniku jer su relevantni za njegov upit.
3. **Proširivanje LLM upita** - Zatim, RAG model proširuje korisnički upit dodavanjem relevantnih dohvaćenih podataka u kontekstu. Prošireni upit omogućuje modelima velikih jezika generiranje točnih odgovora na korisničke upite.
4. **Ažuriranje vanjskih podataka** - Vrlo je bitno održati ažurne informacije i to se može ostvariti automatiziranjem procesa u stvarnom vremenu ili periodičnoj obradi.

Današni AI sustavi bazirani na LLM-ovima mogu dati korisnicima odgovore bez da ljudi pišu definirane skripte. Samim nadovezivanjem RAG-a na LLM-ove smanjuje se potreba za ponovnim obučavanjem modela na novim primjerima [20]. Upiti korisnika nisu uvek jednostavni, mogu biti dvosmisленo formulirani, složeni ili mogu zahtijevati znanje koje model nema. U takvima uvjetima LLM-ovi su skloni izmišljanju stvari. Rod Lastra opisuje velike jezične model kao "Zamislite model kao pretjerano nestrpljivog mlađeg zaposlenika koji izbrblja odgovor prije nego što provjeri činjenice" Lastra dodatno ističe važnost svjesnog upravljanja neizvjesnošću

u odgovaranjum, navodeći: "Iskustvo nas uči da zastanemo i kažemo kada nešto ne znamo. LLM-ovi moraju biti eksplisitno obučeni da prepoznaju pitanja na koja ne mogu odgovoriti" [20].

Sljedeća slika objašnjava kako RAG funkcioniра na konkretnom primjeru. Ulaz u RAG sustav mogu biti različite vrste dokumenta, poput PDF-ova ili CSV datoteka. Nakon toga čita se sadržaj dokumenta i dijeli se u manje dijelove (eng. chunks) koji se mogu lakše obraditi. Svaki od tih dijelova pretvara se u vektor koristeći model ugrađivanja (eng. embedding model) koji pretvara tekstualne podatke u numerički oblik. Ti vektori se zatim pohranjuju u vektorsku bazu podataka koja omoguće brzi pretraživanje informacija. Nakon toga dolazi do korisničkog upita koji se također pretvara u vektor. Taj vektor se uspoređuje s vektorima u bazi podataka kako bi se pronašle relevantne informacije. Nadalje se rangiraju rezultati i odabiru oni najrelevantniji. Na kraju, LLM koristi te informacije za generiranje odgovora na korisnički upit.



Slika 5: Konceptualni tok korištenja RAG-a s LLM-ovima [18]

Jedna od zanimljivih stvari ovdje su vektorske baze podataka koje se mogu učinkovito indeksirati, pretraživati te je u njih jednostavno pohraniti veliku količinu informacija o čemu će se reći nešto detaljnije u jednom od sljedećih poglavlja.

#### 4.2.1. Faza dohvaćanja (Retriever)

Retriever je komponenta koja je odgovorna za pronalaženje relevantnih dokumenata na temelju ulaznog upita. Retriever bilježi složenije semantičke odnose između upita i dokumenta, što dovodi do točnijih rezultata dohvaćanja. Retriever može ugraditi dokumente i upite u visokodimenzionalni vektorski prostor, gdje udaljenost između vektora odgovara relevantnosti dokumenta za upit. Dokumenti koji imaju veću "udaljenost" su manje relevantni za ulazni upit i time se mogu protumačiti kao nebitni. Retriever uzima ulazni upit, pretvara ga u vektor pomoću koder upita, a zatim pronalazi najsličnije vektore dokumenta. Dokumenti povezani s tim vektorima se naknadno prosljeđuju generatoru. Broj dokumenta dohvaćenih u bilo kojem trenutku može se prilagoditi na temelju specifičnih zahtjeva zadatka i obično za zadatke koji su kompleksniji može se pozvati opsežniji skup dokumenata. Bitan aspekt kod retrivera je da se obrati pažnja na točnost dohvaćenih dokumenta tijekom pronalaženja. Postavljanje praga za udaljenost nije isključivo tehnička odluka, već i poslovna odluka. Prag bi trebao biti usklađen

s općim ciljevima, bilo da se radi o pružanju sveobuhvatnog pregleda ili navođenju konkretnih detalja [22].

Korisnik postavlja pitanje ili šalje upit nakon čega retriever traži relevantne dokumente u unaprijed definiranom izvoru znanja (npr. Wikipedia, interni podaci tvrtke ili online članci). Nakon toga se rangiraju dokumenti na temelju njihove relevantnosti za upit i za kraj se odabrani dokumenti prosljeđuju sljedećoj komponenti - generatoru - kako bi se pomoglo mu u generiranju odgovora. Retriever je važan jer za razliku od tradicionalnih AI modela, koji se oslanjaju samo na unaprijed obučene podatke, može dohvatiti informacije u stvarnom vremenu. Također korištenje vanjskih izvora pomaže u smanjenju AI halucinacija i omogućuje modelu da se referira na pozadinsko znanje, što dovodi do smislenijih odgovora [23].

#### **4.2.1.1. BM25**

BM25 (eng. Best Matching 25) je popularni algoritam koji koristi pojam frenkvencijsko inverzne dokumentne frekvencije (eng. term frequency-inverse document frequency - TF-IDF) za rangiranje dokumenata na temelju relevantnosti. BM25 izračunava ocjenu relevantnosti dokumenta na temelju učestalosti pojavljivanja upita u dokumentu. Iako je učinkovit za podudaranja između ključnih riječi, BM25 može imati ograničenja u razumijevanju semantičkih odnosa. Unatoč tome BM25 se široko koristi zbog svoje jednostavnosti i učinkovitosti. Često se koristi za jednostavnije upite temeljene na ključnim riječima, gdje je važno brzo dohvaćanje relevantnih informacija [24].

#### **4.2.1.2. Dense Passage Retrieval**

DPR (eng. Dense Passage Retrieval) je predstavljen 2020.godine i predstavlja moderniji pristup pretraživanja informacija. Koristi gustih vektorski prostor u kojem su upit i dokumenti kodirani u višedimenzionalne vektore. DPR koristi arhitekturu bi-enkodera, gdje su upit i dokument kodirani odvojeno, što omogućuje učinkovito pretraživanje najbližeg susjeda. Za razliku od BM25, DPR se ističe u pretraživanju semantičkih odnosa između upita i dokumenta, što ga čini pogodnim za složenije upite koji zahtijevaju dublje razumijevanje konteksta. DPR može pronaći dokumente koji su kontekstualno povezani s upitom, čak i ako ne sadrže točne ključne riječi [24].

#### **4.2.1.3. REALM**

REALM (eng. Retrieval-Augmented Language Model) je model koji integira pronalaženje informacija i generiranje teksta u jednom sustavu. Ključna inovacija REALM-a je da uči pronalaziti dokumente koji poboljšavaju performanse modela na određenim zadacima kao što je npr. sažimanje teksta. Tijekom treninga REALM ažurira retriever i generativni model zajedno, osiguravajući da je proces pronalaženja optimizirana za zadatak generiranja. REALM je treniran kako bi prepoznao dokumente koji nisu relevantni za upit, ali su korisni za generiranje odgovora. U literaturi se često spominje da REALM nadmašuje BM25 i DPR u zadacima koji zahtijevaju kombinaciju pronalaženja i generiranja informacija [24].

## 4.2.2. Faza generiranja (Generator)

Generator je komponenta koja koristi dohvaćene dokumente i izvorni upit kako bi generirala odgovor. To je obično veliki model transformatora kao što je GPT3.5, GPT4, Llama2, Falcpm, PaLM i BERT. Dohvaćeni dokumenti i ulazni upit ulančavaju se i unose u generator, koji zatim koristi kombinirani ulaz za generiranje odgovora pri čemu dohvaćeni dokumenti pružaju dodatni kontekst i informacije koje pomažu generatoru da bude informiraniji i precizniji u svojim odgovorima [22].

Generator je važan jer pretvara neobrađene podatke u konherentan i strukturiran tekst i osigurava da odgovori zvuče prirodno i privlačno. Također, spaja unaprijed obučeno znanje o umjetnoj inteligenciji s dohvaćenim informacijama u stvarnom vremenu za najbolji mogući odgovor [23].

### 4.2.2.1. T5

T5 (eng. Text-to-Text Transfer Transformer) je jedan od najčešće korištenih modela za zadatke generiranja u RAG sustavima. T5 je svestran model koji pretvara sve zadatke u tekstu-alni format, što ga čini pogodnim za širok spektar aplikacija. Može se koristiti za odgovaranje na pitanja, sažimanje ili generiranje dijaloga. Kroz praksu se pokazalo da RAG modeli temeljeni na T5 nadmašuju tradicionalne generativne model poput GPT-3 i BART-a u raznim zadacima kaš što su odgovaranje na pitanja ili sažimanje teksta [24].

### 4.2.2.2. BART

BART (eng. Bidirectional and Auto-Regressive Transformers) je model koji je prikladan za zadatke koji uključuju generiranje teksta iz nečistih ili neuređenih podataka. Za enkoder BART koristi dvosmjerni pristup, što znači da može razumijeti kontekst cijelog ulaza, dok dekoder generira tekst an temelju tog konteksta. BART može rekonstruirati oštećene tekstove, što ga čini korisnim za zadatke poput izvlačenja informacija iz nepotpunih uli neurednih podataka [24].

## 4.3. Primjena RAG modela

Kroz ovo poglavlje istražena je primjena RAG-a. Na početku istaknut je jedan primjer koji je podijelio Siddharth Asthana u jednom od svojih članaka na LinkedInu. U članku se govori o tome kako je velika bolnička mreža integrirala RAG u svoj sustav podrške kako bi donijela bolje odluke. Sustav se povezao s elektroničkim zdravstvenim kartonima i višestrukim medicinskim bazama podataka, što je dovelo do [25]:

- Smanjenja pogrešnih dijagnoza za složene slučajeve za 30 posto,
- Smanjenje vremena koje su liječnici proveli pregledavajući literaturu za 25 posto,
- Povećanje ranog otkrivanja rijetkih bolesti za 40 posto.

U nastavku je navedno nekoliko domena u kojoj se RAG može primjenjivati citenishtha2025rag:

- **Sustavi za korisničku podršku** - kupac postavlja pitanje, chatbot dohvaća relevantne informacije iz izvora kao što su baze znanja, često postavljena pitanja ili zapisi kupaca i pritom koristi generativni model za izradu personaliziranog odgovora na temelju dohvaćenih podataka.
- **Sažimanje i pretraživanje dokumenta** - korištenje naprednih tehnika dohvaćanja informacija kako bi se poboljšale mogućnosti velikih LLM-ova. Na primjer, Google-ov Vertex AI Search koristi prvo algortime poput ANN kako bi brzo prikupio rezultate i zatim primjenjuje modele dubokog učenja za ponovo rangiranje kako bi se osiguralo da su najrelevantniji dokumenti prioritetni.
- **Medicinska dijagnostika i istraživanje** - korištenje ogromnih baza podataka medicinskog znanja, elektroničkih zdravstvenih kartona, kliničkih smjernica i medicinske literature kako bi se podržalo zdravstvene djelatnike u postavljanju točnih dijagnoza i dobro informiranih odluka o liječenju. Jedan od primjera je IBM Watson koji analizira podatke o pacijentima s opsežnom medicinskom literaturom i time pomaže liječnicima u diagnosticiranju složenijih slučajeva. IBM Watson Health se također koristi za dijagnostiku raka i shodno tome predlaže preporuke liječenja.
- **Personalizirani sustavi učenja i podučavanja** - jedan od primjera je RAMO (eng. Retrieval Augmented Generation for MOOCs) koji se bavi problemom hladnog starta u preporukama tečajeva korištenjem LLM-ova za generiranje personaliziranih prijedloga tečajeva. RAMO pomaže u razumijevanju preferencija i ciljeva osobe, nudeći relevantnije mogućnosti tečaja. Na primjer, sveučilišta su počela uvoditi sustave podučavanja vođene RAG-om kako bi pomogli studentima u učinkovitijem snalaženju u materijalima, potičući dublje razumijevanje i bolje akademske rezultate.
- **Otkrivanje prijevara i procjena rizika** - sposobnost RAG-a da pristupi podacima u stvarnom vremenu i uključi ih tijekom postupka donošenja odluka. Financijske tvrtke poput JPMorgan Chase koriste sustave za otkrivanje prijevara koje su vođene umjetnom inteligencijom. Takvi sustavi kontinuirano dohvaćaju i analiziraju podatke u stvarnom vremenu iz različitih izvora kako bi se pratile transakcije i time otkrile potencijalne prijevare.
- **Preporuke proizvoda za e-trgovinu** - za razliku od tradiocionalnih sustava preporuka, RAG se dinamički prilagođava specifičnim potrebama kupca u stvarnom vremenu čime preporuke postaju relevantnije i točnije što rezultira povećanom prodajom.
- **Upravljanje znanjem u poduzeću** - posebno korisno kod velikih tvrtki s opsežnom dokumentacijom i izvorima podataka.

#### 4.3.1. Ključne prednosti RAG sustava

U nastavku, kroz tablicu su prikazane ključne prednosti RAG sustava, koje objašnjavaju zašto je ova tehnologija sve češći odabir u izradi modernih AI rješenja. Navedene karakteristike posebno dolaze do izražaja u izradi inteligentnih asistentata koji trebaju pružati točne, relevantne i ažurirane informacije.

Tablica 2: Prednosti RAG-a [18]

Prednost	Opis
Isplativa implementacija	Razvoj AI sustava započinje korištenjem temeljnog modela (eng. Foundation models). Temeljni modeli su obično dostupni putem API ključeva i obučeni su na širokom spektru generaliziranih i neoznačenih podataka. Računalni i finansijski troškovi za treniranje takvih modela su jako visoki.
Trenutne informacije	Iako su izvorni podaci LLM-a relevantni za specifične potrebe, teško je održati relevantnost. RAG se može koristiti za izravno povezivanje LLM-a s feedovima društvenih medija, web stranicama s vijestima i samim time pružiti korisnicima najnovije informacije.
Povećano povjerenje korisnika	RAG omogućuje prikazivanje točnih informacija s navođenjem izvora ili citata. Korisnici također mogu pretraživati izvorne dokumente ako im je potrebno dodatno pojašnjenje.
Više kontrole za razvojne programere	Učinkovitije testiranje i poboljšavanje aplikacija. Kontroliranje i mijenjanje izvora informacija. Rješavanje i ispravljanje problema.

#### 4.4. Izazovi RAG pristupa

LLM-ovi su ključna tehnologija umjetne inteligencije koja pokreće chatbotove. Priroda LLM-ova je takva da ponekad unose dozu nepredvidljivosti u generiranje odgovora. Osim toga, podaci o LLM obuci su statični i imaju krajnji rok za znanje koje posjeduju. Negativne strane LLM-ova uključuju [18]:

- Iznošenje lažnih informacija kada na njih nema odgovora (nepoželjni efekti nazvani - halucinacijama),
- Prikazivanje zastarjelih informacija kada korisnik očekuje aktualan odgovor,
- Izrada odgovora iz neautoriziranih izvora,
- Stvaranje netočnih odgovora zbog terminološke zbrke, različiti izvori obuke koriste istu terminologiju za opisivanje različite stvari.

Veliki jezični modeli mogu biti nekonzistentni što bi značilo da ponekad točno odgovore na pitanja, a drugi put nasumično pogađaju. Ako povremeno zvuče kao da nemaju pojma što govore, to je zato što ustvari nemaju pojma. LLM-ovi znaju kako se riječi statistički povezivaju, ali ne znaju značenje riječi kao ni cjelinu njih, poznatu kao rečenica. RAG također smanjuje potrebu korisnika da kontinuirano treniraju model na novim podacima i ažuriraju njegove parametre. Na taj način, RAG može smanjiti računalne i finansijske troškove pokretanja chatbotova [20].

Sljedeći izvor navodi sljedeće izazove kod RAG pristupa [26]:

- **Sadržaj koji nedostaje** - jedan od glavnih izazova je kada odgovor na upit korisnika nije prisutan u indeksiranim dokumentima. U takvim slučajevima sustav je sklon generiranju obmanjujućih odgovora ili u nekim slučajevima ne prepoznaže da mu nedostaju potrebne informacije. Na primjer, idealno ponašanje u takvim slučajevima je najbolje priznati poput: "Žao mi je, ne znam", iako to većina sustava ne uspijeva, što dovodi do smanjenog povjerenja korisnika. Ovaj izazov se može umanjiti povremenim dodavanjem podataka, implementacijom povratnih informacija kroz pogreške kako bi se ukazalo na nedostatak informacija i uvođenje prestanka generiranja odgovora kada odgovor ne zadovoljava potrebe korisnika.
- **Neoptimalno dohvaćanje i rangiranje** - u slučaju da odgovor postoji unutar indeksiranog skupa podataka, moglo bi se desiti da neće biti dovoljno visoko rangiran za dohvaćanje npr. zbog neoptimalnih algoritama rangiranja. Algoritmi rangiranja koji se oslanjaju isključivo na rezultate sličnosti često zanemaruju kontekst i specifičnost. Ovaj izazov se može umanjiti poboljšavanjem mehanizama rangiranja uključivanjem metapodataka kao što su vrste dokumenata, autorstvo ili datum objavljenja. Također može se umanjiti s eksperimentiranjem s jačim modelima rangiranja kako bi se vidjelo hoće li se relevantniji dokumenti drugačije pozicionirati.
- **Proturiće informacije** - proturiće informacije mogu dovesti do toga da LLM generira zbumnuće informacije (halucinacije). Na primjer, RAG sustav korisničke podrške dohvaća informacije zastarijele politike uz trenutne, uzrokovajući da LLM generira netočne ili zbumnuće odgovore. To se može riješiti dodatnim filtriranjem nebitnih i kontradiktornih informacija prije nego što dođu do LLM-a. Dodatno trebalo bi još provesti testiranja i fino podesiti upite kako bi se usmjerila pozornost LLM-a na najrelevantnije / najnovije informacije.
- **Nepotpuni odgovori** - nepotpuni odgovori nastaju kada sustav ne uspije spojiti sve relevantne informacije. Na primjer, RAG sustav koji se bavi domenom prava i od njega se traži da sažme ključne točke tri različita slučaja, može se baviti samo jednim slučajem, izostavljajući kritične detalje iz ostala dva.
- **Izazovi s performansama i skalabilnošću u pronalaženju** - takvi izazovi nastaju kada se sustav bori s učinkovitim rukovanjem skupova podataka velikih razmjera. Kako veličina skupa raste, latencija dohvaćanja se povećava zbog računalnih troškova u pretraživanju, rangiranju i dohvaćanju relevantnih dokumenata.

#### 4.4.1. Halucinacije

AI halucinacije su izlazi iz modela koji značajno odstupaju od stvarnosti i često predstavljaju netočne ili izmišljene informacije kao istinite. Halucinacije nastaju kao rezultat inherentnih ograničenja podataka za LLM obuku ili kada model ne uspije povezati namjeru ili kontekst upita s podacima potrebnim za generiranje smislenog odgovora. Iako je RAG osmišljen kako bi pomogao u smanjenju AI halucinacija, RAG halucinacije se i dalje mogu pojavljivati. Na takve stvari treba pripaziti jer na primjer, pretplatnik mobilne mreže može dobiti netočan odgovor o svom prosječnom mjesečnom računu od operaterovog RAG chatbota - jer podaci tvrtke možda uključuju račune koji nisu bili njegovi [27].

AI halucinacije se mogu suzbiti kroz nekoliko pristupa [27]:

- **Fino podešavanje provjerom činjenica** - postoji sloj koji proverava činjenice prije nego što se generira odgovor. Nedosljednosti se označavaju, što potiče LLM da poboljša svoj rezultat na temelju činjeničnih temelja.
- **Navođenje izvora** - razumijevanje načina na koji generativna umjetna inteligencija dolazi do odgovora ključno je. Korisnicima se pokazuju izvori koji su korišteni za generiranje odgovora, čime je osigurana pouzdanost i transparentnost.
- **Povezivanje modela s javnim podacima više kvalitete** - daje se prednost visokokvalitetnim, raznolikim i činjeničnim informacijama. Tehnike poput čišćenja podataka i filtriranja pristranosti mogu pomoći u smanjenju halucinacija.
- **Korištenje RAG-a za proširenje LLM-ova s privatnim organizacijskim podacima** - nadopunjavanje znanja s privatnim podacima organizacije može smanjiti rizik od izmišljenih odgovora.

RAG ne može u potpunosti eliminirati halucinacije i ograničen je kvalitetom podataka, kontekstualnom svijesti (možda se neće svaki put najbolje razumijet što korisnik želi - propuštanje poante prompta), razmišljanjem (zdravi razum) i kreativnošću modela.

## 4.5. Dostupni alati i biblioteke

U ovoj sekciji su navedeni neki od najpopularnijih alata i biblioteka koje se koriste za implementaciju RAG sustava. Izvršen je pregled i analiza svakog alata nakon čega je donesen zaključak koji od njih je korišten u praktičnom dijelu rada. Na sljedećoj slici može se vidjeti kratki sažetak alata i biblioteka koje se mogu koristiti za implementaciju RAG sustava.



Slika 6: Sažetak dostupnih alata i biblioteka za razvoj RAG sustava; Izvor

### 4.5.1. Okviri (frameworkovi) za upravljanje RAG-om

#### 4.5.1.1. Haystack

Haystack je platforma za obradu prirodnog jezika otvorenog koda koja je specijalizirana za stvaranje RAG cjevovoda za sustave pretraživanja i odgovaranja na pitanja. Framework pruža komponente za dohvaćanje dokumenta, odgovaranje na pitanja i generiranje teksta, omogućujući korisnicima da brzo i jednostavno izrade RAG aplikacije. Haystack se također povezuje s modelima kao što su BERT i RoBERTa, što poboljšava njegovu sposobnost rukovanja s težim upitima. Također Haystack ima API koji je prilagođen korisniku i korisničkom sučelju temeljenom na webu, omogućujući korisnicima jednostavnu interakciju sa sustavom [28].

Značajke koje Haystack nudi su [28]:

- Podržavanje pozadinskih sustava za pohranu kao što su Elasticsearch, FAISS, SQL i InMemory,
- Pretraživanje na temelju ključnih riječi pomoću BM25,
- GenerativePipeline - kombiniranje retrivera i generatora (GPT - 3/4),
- TransformersReader - ekstrakcijska kontrola kvalitete koristeći Hugging Face modele,
- DensePassageRetriever - dohvaća gusto ugrađene vektore koristeći DPR (eng. Dense Passage Retrieval),
- EmbeddingRetriever - custom embeddings koristeći Hugging Face modele.

Vrlo lagano se može instalirati lokalno na Linux sustavu pomoću pip-a ili conde (pip install haystack-ai). Dokumentacija se nalazi na sljedećem linku - <https://docs.haystack.deepset.ai/docs/intro>.

#### 4.5.1.2. LangChain

Langchain je Python paket otvorenog koda koji pruža sveobuhvatnu osnovu za izgradnju aplikacija s velikim jezičnim modelima. Kombinira modularni i fleksibilni dizajn s sučeljem visoke razine, što ga čini idealnim za razvoj RAG sustava. Također, LangChain omogućuje jednostavnu integraciju različitih izvora podataka, uključujući dokumente, baze podataka i API-je što pomaže u procesu generiranja odgovora. Langchain nudi širok raspon funkcionalnosti i omogućuje korisnicima izmjenu i kombiniranje različitih komponenti kako bi odgovarale jedinstvenim zahtjevima aplikacije, što olakšava stvaranje dinamičkih i prilagodljivih aplikacija [28].

Značajke koje LangChain nudi su [28]:

- Integracija s vektorskim bazama podataka kao što su Pinecone, Chroma i FAISS,
- Učitavanje i dohvaćanje podataka iz baza podataka, API-ja i lokalnih datoteka,
- Retrieveri uključuju BM25, Chroma, FAISS i druge,
- Učitavanje PDF, teksta, web scrapinga i SQL/NoSQL baza podataka,
- Generiranje dinamičkih upita pomoću predloženih struktura.

Langchain se može instalirati lokalno na Linux sustavu koristeći pip (pip install -U langchain). Dokumentacija se nalazi na - <https://python.langchain.com/docs/introduction/>

#### 4.5.1.3. Llamaindex

Llamaindex je robusna biblioteka otvorenog koda koja se usredotočuje na učinkovito indeksiranje i dohvaćanje iz ogromnih skupova podataka. Llamaindex koristi napredne tehnike kao što je pretraživanje vektorske sličnosti i hijerarhijsko indeksiranje kako bi se omogućilo brzo i točno dohvaćanje relevantnih informacija, poboljšavajući mogućnosti generativnih jezičnih modela. Lako se povezuje s uobičajenim velikim jezičnim modelima, omogućujući umetanje primljenih podataka u proces stvaranja i čineći ga učinkovitim alatom za poboljšanje performansi RAG sustava [28].

- Optimizirano je dohvaćanje s malom latencijom,
- Moguće je učitavanje različitih vrsta podataka kao što su: TXT, PDF, DOC, CSV, API-je, baze podataka i web scraping,
- Kombiniranje modela (OpenAI ili Hugging Face) s vektorskim bazama podataka,
- Sadrži više indeksa kao što su:

- **Indeks pohrane vektora** - se koristi za pohranu podataka kao vektora, što omogućuje brzo pretraživanje sličnosti kao što su sustavi za dohvaćanje dokumenata i sustavi preporuka,
- **Indeks popisa** - je jednostavan, sekvensijalan indeks za manje skupove podataka koji omogućuje brzo linearno pretraživanje,
- **Indeks stabla** - koristi hijerarhijsku strukturu za izvođenje učinkovitih semantičkih pretraživanja čime je idealan za složene upite kod podataka gdje je jasna hijerarhija,
- **Indeks tablice ključnih riječi** - tablica mapiranja se koristi kako bi olakšala pretraživanje temeljeno na ključnim riječima.

LlamaIndex je moguće instalirati pomoću naredbi pip install llama-index ili pip install llama-index-core. Dokumentacija se nalazi na sljedećem linku <https://docs.llamaindex.ai/en/stable/>.

#### 4.5.2. Vektorske baze podataka

Tradicionalne baze podataka pohranjuju podatke u strukturiranom formatu (redci i stupci), dok vektorske baze podataka su optimizirane za rukovanje visokodimenzionalnim vektorskim podacima i podržavaju operacije poput pretraživanja sličnosti i pretraživanja najbližih susjeda (k-NN). Umetanje podataka se vrši kroz generiranje ugrađivanja (eng. embeddings) iz neobrađenih vrsta podataka (npr. teksta) i pohranjivanje tih ugrađivanja u vektorskiju bazu podataka. Vektorske baze podataka ključne su kako bi omogućile LLM-ovima da daju točnije i skalabilnije informacije. Osim toga, vektorske baze podataka omogućuju rad s bilo kojim drugim tipovima podataka, uključujući slike, videozapise i zvuk, čime se proširuju mogućnost RAG sustava. Većina podataka koja se generira svakodnevno su nestrukturirani podaci, bilo to tekst, slike, videozapisi ili slično. Nestrukturiranim podacima nedostaje strogi format, što otežava upravljanje konvencionalnim bazama podataka i iako se ponekad čine beznačajnim, oni sadrže vrijedne informacije koje imaju ogroman potencijal za umjetnu inteligenciju [29].

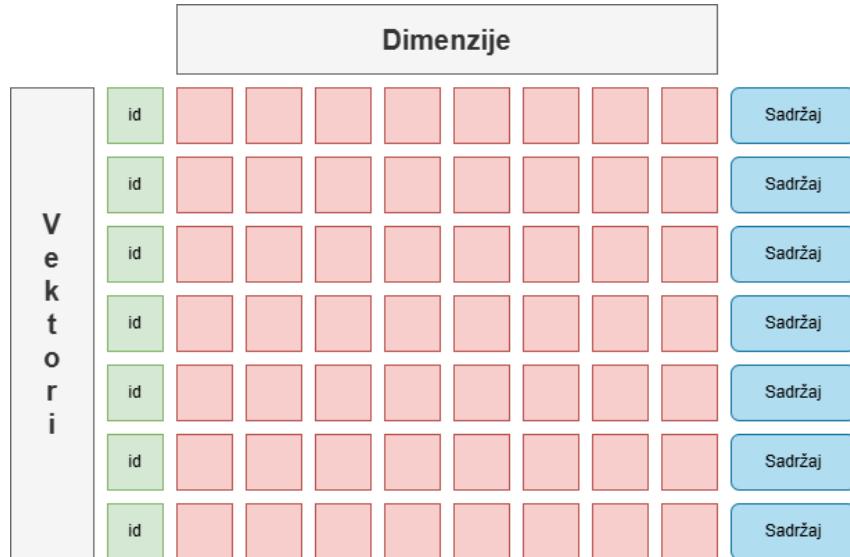
Tradicionalne baze podataka poput OLTP i OLAP odlično upravljaju s strukturiranim podacima, međutim kada se pojave podaci koji se ne mogu lako kategorizirati poput sadržaja unutar PDF datoteka stvari počinju biti komplikirane. Takva baza ne može razumijeti što se nalazi u dokumentu, ne može ga kategorizirati ili pretraživati. Vektorske baze podataka dobile su na značaju u posljednjih nekoliko godina zbog porasta strojnog učenja i dubokog učenja. Vektorska ugrađivanja pretvaraju složene podatke (npr. slike ili tekst) u visokodimenzionalne vektore tako da su slične stavke bliže jedna drugoj u vektorskome prostoru [29].

Tradicionalne baze podataka su neučinkovite pri pretraživanju sličnosti u visokodimenzionalnim prostorima. Takav problem vrlo dobro rješavaju vektorske baze podataka koje vrlo dobro indeksiraju i pretraživaju opsežne skupove podataka. Kako bi se proveo rad s vektorskimi bazama podataka, potrebno je koristiti vektor upita koji obuhvaća kriterije pretraživanja. Sljedeći korak je korištenje metrike sličnosti za određivanje blizine između vektora upita i vektora

u bazi podataka. To može uključivati metriku kao što su kosinusna sličnost, euklidska udaljenost ili Jaccardov indeks. U prošlosti su samo velike tehnološke tvrtke koje su imale resurse za njihovo stvaranje i održavanje koristile vektorske baze podataka. S obzirom na napredak tehnologije, pojavile su se mnoge vektorske baze podataka otvorenog koda koja su dostupne za korištenje i implementaciju u RAG sustavima. Sljedeći grafikon prikazuje kada se koriste vektorske baze podataka u odnosu na tradicionalne baze podataka [29].

Karakteristika	OLTP baza podataka	OLAP baza podataka	Vektorska baza podataka
<b>Struktura podataka</b>	Redci i stupci	Redci i stupci	Vektori
<b>Podaci</b>	Strukturirani	Strukturirani / nestrukturirani	Nestrukturirani
<b>Metoda upita</b>	SQL (Transakcije)	SQL (agregacije i analitički upiti)	Vektorsko pretraživanje na temelju sličnosti
<b>Fokus</b>	Optimizirano za ažuriranje	Optimizirano za čitanje	Kontekst i semantika
<b>Slučajevi upotrebe</b>	CRM, obrada narudžbi	Poslovna inteligencija, skladišta	RAG, otkrivanje anomalija

Tablica 3: Usporedba OLTP, OLAP i vektorskih baza podataka [29]



Slika 7: Vektorska baza - primjer; Izvor

#### 4.5.2.1. FAISS

FAISS (eng. Facebook AI Similarity Search) je robusna vektorska baza podataka koju je razvio Facevook-ov istraživačkim tim za umjetnu inteligenciju. Vrlo je učinkovita i brza, što je čini savršenim izborom za razvoj aplikacija u stvarnom vremenu specifičnih za industriju. Dostupno je nekoliko opcija indeksiranja, učinkovito grupira i izgrađena je na Pythonu i C++ jeziku. Jedne od poznatijih tvrtki koje koriste FAISS su Meta i Pinterest [30].

#### **4.5.2.2. Qdrant**

Qdrant je vektorska baza podataka koja je optimizirana za pretraživanje po sličnosti i omogućuje brzo i učinkovito pretraživanje velikih skupova podataka. Od dodatnih mogućnosti Qdrant nudi: mogućnost preciznog pretraživanja, pametno pohranjivanje vektora, skalabilni dizajn prilagođen oblaku. Klijenti Qdranta su: Bosch, Johnson i Johnson i Discord [30].

#### **4.5.2.3. Pinecone**

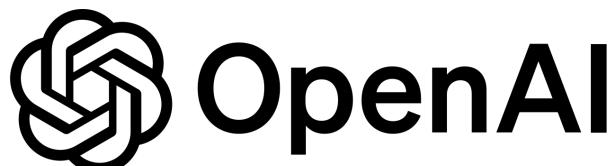
Pinecone je vektorska baza podataka u oblaku (eng. cloud-native) zatvorenog koda koja nudi upravlјivost, skalabilnost i visoku učinkovitost. Jednostavnost, podrška za hibridne upite i uska integracija s strojim učenjem čine Pinecone popularnim izborom za RAG sustave. Značajke koje Pinecone nudi su: ažuriranje u stvarnom vremenu bez zastoja, visokodimenzijsko vektorsko pretraživanje u velikoj mjeri i integracija s Pythonom, Langchainom i drugima. Neki od klijenta Pinecone su: Microsoft, Notion, Hubspot i Shopify. Pinecone je vektorska baza podataka u oblaku koja je potpuno upravljana i ne zahtijeva nikako odražavanje infrastrukture [30].

### **4.5.3. Veliki jezični modeli**

Pružatelji LLM modela su organizacije koje razvijaju i nude pristup velikim jezičnim modelima bez potrebe da ih sami izgrađujemo ili obučavamo. Obično nude API-jeve koji omogućuju korisnicima da integriraju LLM-ove u svoje aplikacije.

#### **4.5.3.1. OpenAI**

OpenAI je jedan od začetnika i najpoznatijih pružatelja usluga u području umjetne inteligencije. Poznat je po razvoju GPT modela, uključujući GPT-3 i GPT-4, kao i Codexa za generiranje koda. Od dodatnih mogućnosti generira tekst, razumije jezik, pomaže pri kodiranju i može se koristiti za razvoj AI sustava i drugih aplikacija. Dostupan je putem OpenAI API-ja i putem Microsoftove usluge Azure OpenAI [31].



Slika 8: OpenAI - ikona; Izvor

#### 4.5.3.2. Hugging Face

Hugging Face je platforma koja nudi širok spektar modela otvorenog koda a poznat je po svojoj biblioteci Transformers i Model Hub-u. Hugging Face nudi različite modela poput BERT-a, GPT-2 i T5-a koji se mogu koristiti za različite zadatke. Transformers biblioteka koja omogućuje jednostavno korištenje i integraciju najsuvremenijih modela dubokog učenja. ModelHub je platforma na kojoj programeri i istraživači mogu prenositi, dijeliti i suradivati na modelima. Također Hugging Face potiče okruženje u kojem se dijeli znanje i resursi čime se ubrzava razvoj i potiče se na inovacije u području umjetne inteligencije [31].



#### HUGGING FACE

Slika 9: Hugging Face - ikona; Izvor

#### 4.5.3.3. Anthropic

Anthropic je tvrtka koja se fokusira na razvoj sigurnosnih i etičkih AI modela koji se ponašaju u skladu s ljudskim vrijednostima. Poznata je po seriji Claude modela [31].



Slika 10: Anthropic - ikona; Izvor

Zaključak je da su svi navedeni alati i biblioteke korisni za implemetaciju RAG sustava, ali za potrebe ovog rada korištena je kombinacija Langchaina, ChatGPT (OpenAI) modela i FAISS vektorske baze podataka.

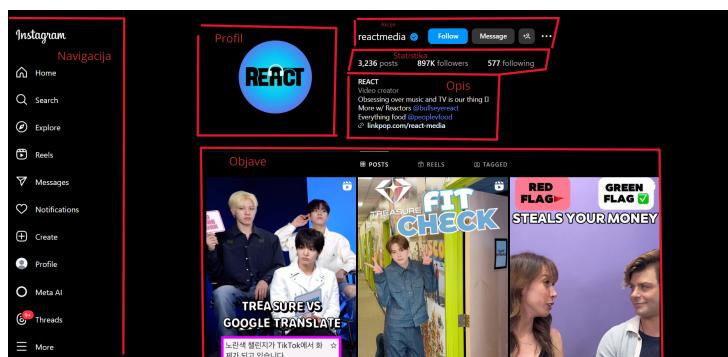
## 5. Korištene tehnologije i alati

Za realizaciju praktičnog dijela korišteno je nekoliko tehnologija koje su omogućile razvoj web stranice i inteligentnog asistenta. Odabir tehnologija temelji se na njihovoj popularnosti, pouzdanosti i sposobnosti da zadovolje zahtjeve ovog rada. U nastavku su detaljnije opisani korišteni alati i tehnologije.

### 5.1. React

Svijet danas ne funkcioniira bez mobilnih i web aplikacija. Sve se orijentira prema digitalizaciji, od rezerviranja hrane do naručivanja vožnje u taksiju. Vrlo je bitno napraviti vizualno primamljivu aplikaciju a React je upravo jedna od popularnijih frontend biblioteka koja to omogućava. React je temeljen na JavaScriptu i široko se koristi u web razvoju. U usporedbi s drugim tehnologijama , React je nova tehnologija, a osnovan je od strane Jordan Walkea 2011. godine (softverski inženjer u Facebooku). Reactova popularnost dolazi radi nekoliko razlog [32]:

- jednostavno stvaranje dinamičkih aplikacija,
- korištenje Virtual DOM čime se aplikacije brže izrađuju (ažurira komponente samo koje su promijenjene),
- podjela na komponente - jedna aplikacija se sastoji od više komponenti koje se mogu koristit kroz cijelu aplikaciju (što smanjuje vrijeme razvoja aplikacije),
- mala krivulja učenja - React je jednostavan za učenje jer kombinira HTML i JavaScript koncepte,
- koristi se za razvoj web i mobilnih aplikacija - React Native - za mobilne aplikacije,
- jednostavno otklanjanje grešaka - Facebook je izdao Chromeovo proširenje koje se može koristit za otklanjanje pogrešaka u React aplikacijama.



Slika 11: React komponenta

Jedna zanimljivost vezana uz React programere, promatrana na dvije potpuno različite lokacije, otkriva da su indijski developeri za isti posao plaćeni i do deset puta manje (podaci

za 2021. godinu). To jasno odražava razliku u životnom standardu i ekonomskim potrebama pojedinih zemalja [32].

- Prosječna plaća za početnog React developera u SAD-u iznosi oko 87000 USD godišnje,
- Prosječna plaća za početnog React developera u Indiji iznosi oko 8000 USD godišnje.

## 5.2. FastAPI

FastAPI je moderan, brz, visokoučinkovit web framework za izgradnju API-ja s Pythonom. Vrlo je brz (u rangu s NodeJS-om i Go-om) i omogućuje brzo kodiranje i pojavljivanje manjeg broja grešaka. Dosta je intuitivan i jednostavan za korištenje i učenje. FastAPI podržava CRUD (eng. Create, Read, Update, Delete) operacije koje je moguće koristiti kroz ključne riječi HTTP metoda (GET, POST, PUT, DELETE). Za pokretanje FastAPI koristi port 8000 i može se testirati putem preglednika ili alata poput Postmana. 8000/docs putanja omogućuje pristup Swagger UI sučelju koje omogućuje pregled i testiranje API-ja [33].



Slika 12: FastAPI - ikona; Izvor

## 5.3. SMTP

## 6. Praktični dio rada

Praktični dio rada se sastoji od izrade web stranice i izrade inteligentnog asistenta temeljenog na RAG tehnologiji. Frontend aplikacije je izrađen pomoću biblioteke React, pri čemu je korisničko sučelje strukturirano korištenjem JSX sintakse. React predstavlja jedno od najpopularnijih frameworka za izradu web aplikacija. Posebno se ističe zbog toga što omogućava razvoj aplikacija koje su brze, jednostavne i lako održive. Osim toga, najbitnija stvar je što React gradi aplikaciju po komponentama, što omogućava jasnije razumijevanje koda kao i njegovo olakšano preuređivanje.

Backend aplikacije razvijen je korištenjem Python web frameworka FastAPI, koji je poznat po svojoj jednostavnosti korištenja.

Za izradu inteligentnog asistenta prvotno je napravljen scraper podataka s kojim je izrađen skup podataka koji je poslužio za treniranje inteligentnog asistenta.

### 6.1. Struktura web stranice

Sljedeća tablica prikazuje pregled stranica web aplikacije te pripadajuće komponente koje su na njima implementirane. Svaka stranica ima svoju svrhu čime se osigurava intuitivna navigacija. Svaka stranica sadrži obavezne elemente kao što su podnožje (eng. footer) i zaglavlje (eng. header).

Tablica 4: Struktura stranica i njihovih komponenti

Stranica / Element	Komponente	Opis
Početna	Slider, Motivacija, Kratko o Łódżu, Drugi gradovi	Uvodna stranica s informacijama i poveznicama prema ostalim dijelovima stranice.
O Łódżu	Dodatno o gradu, Iskustva bivših studenata	Informacije o gradu i iskustvima Erasmus+ studenata.
Drugi gradovi	Pregled gradova	Pregled ostalih gradova Poljske koje je moguće posjetit.
Pitanja	Forma za postavljanje pitanja	Korisnik postavlja pitanje ako AI asistent ne ponudi tražene informacije.
Dokumenti	Pregled, dodavanje i brisanje dokumenata za RAG	Sekcija za prikaz i pregled dokumenata koji služe kako bi se odgovorilo na pitanja korisnika.
AI asistent (pop-up)	Sučelje za razgovor	Interaktivni chat s korisnikom; otvara se klikom na ikonu asistenta.

**[Napomena]** - Svaka stranica sadrži obavezne elemente kao što su podnožje (eng. footer) i zaglavlje (eng. header).

## 6.2. Korisničko sučelje (Frontend)

Na sljedećoj slici može se vidjeti struktura frontend dijela. Prikazane su samo određeni dijelovi koji predstavljaju najvažnije stvari u frontend dijelu.

```
Frontend/
└── node_modules/ - paketi i biblioteke potrebne za rad projekta
    └── src/ - izvorni kod aplikacije
        ├── api/ - komunikacija s backendom (npr. slanje korisnikovog pitanja)
        ├── assets/ - slike, video materijali i druge statičke datoteke
        ├── components/ - višekratne komponente koje se koriste unutar stranica (npr. navigacija)
        ├── contexts/ - globalna stanja dostupna kroz cijelu aplikaciju, npr. promjena jezika
        ├── locales/ - JSON datoteke s prijevodima ako aplikacija podržava više jezika
        ├── pages/ - glavne stranice aplikacije (npr. Početna)
        ├── App.jsx - definira rute pomoću BrowserRouter i omotava aplikaciju LanguageProviderom radi podrške za više jezika
        └── main.jsx - ulazna točka aplikacije, ovdje se React aplikacija renderira unutar HTML-a
```

Slika 13: Struktura frontend dijela

Funkcija askAssistant šalje pitanje i opcionalno PDF datoteke backendu putem HTTP POST zahtjeva. Koristi se za komunikaciju s FastAPI-jem ( ruta /ask) koji obrađuje pitanje uz kontekst PDF-ova.

```
1  export async function askAssistant(question, files = []) {
2      const formData = new FormData();
3      formData.append("question", question);
4
4      if (files && files.length > 0) {
5          files.forEach(file => formData.append("pdfs", file));
6      }
7
7      const response = await fetch("http://localhost:8000/ask", {
8          method: "POST",
9          body: formData,
10     });
11
11     if (!response.ok) {
12         throw new Error(`Request failed: ${response.statusText}`);
13     }
14
14     const data = await response.json();
15     return data;
16 }
```

Isječak koda 1: Chat.jsx

Funkcija sendContactForm šalje podatke iz kontakt forme na backend rutu /send\_email koristeći POST zahtjev s JSON tijelom. Koristi se za slanje emailova s korisničkim podacima (ime, email, poruka itd.) na server putem FastAPI endpointa.

```
1  export async function sendContactForm(formData) {
2      const response = await fetch("http://localhost:8000/send-email", {
3          method: "POST",
4          headers: {
5              "Content-Type": "application/json",
6          },
7          body: JSON.stringify(formData),
8      });
9 }
```

```

9   if (!response.ok) {
10     const data = await response.json();
11     throw new Error(data.error || "Greška prilikom slanja e-maila.");
12   }
13
14   return response.json();
}

```

Isječak koda 2: Contact.jsx

### 6.3. Poslužiteljski dio (Backend)

Na sljedećoj slici može se vidjeti struktura backend dijela. Prikazane su sve stvari koje su od značaja za backend dio aplikacije.

```

Backend/
  routes/
    ask.py
    documents.py
    email.py
    stats.py
  services/
    pdf_handler.py
    qa.py
    vectorstore.py
  vectorstore/
  main.py
  requirements.txt

```

- API rute koje frontend koristi za komunikaciju s backendom
- postavljanje pitanja na temelju PDF dokumenata
- dohvatanje i brisanje dokumenata iz baze (GET, DELETE)
- slanje poruke s kontakt forme putem e-maila
- vraćanje statistike o pohranjenim dokumentima i ulomcima
- pomoćne funkcije koje podržavaju rute (logika)
- obrada PDF-ova: čitanje, chunkanje i dodavanje metapodataka
- QA lanac koji koristi OpenAI LLM i FAISS za odgovaranje na upite
- upravljanje vektorskog bazom (učitavanje, spremanje, brisanje dokumenata)
- direktorij u kojem se lokalno pohranjuje FAISS vektorska baza dokumenata
- glavni ulaz u aplikaciju – pokretanje FastAPI servera i uključivanje svih ruta
- popis svih potrebnih Python paketa za pokretanje aplikacije

Slika 14: Struktura backend dijela

Env datoteka sadrži konfiguracijske varijable koje se koriste u aplikaciji. OpenAI API ključ se koristi za pristup OpenAI modelima, Gmail user je adresa s koje se šalju poruke s kontakt forme, Gmail password je aplikacijska lozinka za autentifikaciju Gmail računa i Gmail receiver je adresa primatelja poruka koja može biti kao Gmail user.

```

1  OPENAI_API_KEY=
2  GMAIL_USER=
3  GMAIL_PASSWORD=
4  GMAIL_RECEIVER=

```

Slika 15: Sadržaj env datoteke

#### 6.3.1. Server

Main predstavlja glavnu datoteku koja pokreće FastAPI aplikaciju. Učitava se konfiguracija iz .env datoteke, omogućuje se CORS pristup kako bi frontend i backend mogli komunicirati bez ograničenja, te se u aplikaciju uključuju sve definirane rute za pitanja, dokumente, statistiku i email.

---

```

1 from fastapi import FastAPI
2 from fastapi.middleware.cors import CORSMiddleware
3 from dotenv import load_dotenv

4 from routes import ask, documents, stats, email

5 load_dotenv()

6 app = FastAPI()

7 app.add_middleware(
8     CORSMiddleware,
9     allow_origins=["*"],
10    allow_credentials=True,
11    allow_methods=["*"],
12    allow_headers=["*"],
13 )

14 app.include_router(ask.router)
15 app.include_router(documents.router)
16 app.include_router(stats.router)
17 app.include_router(email.router)

```

---

Isječak koda 3: Main.py

### 6.3.2. Servisi

Sljedeći dio koda upravlja vektorskog bazom podataka koristeći FAISS i OpenAIEmbbeddings. Omogućuje učitavanje vektorske baze podataka, dodavanje novih dokumenata te brisanje dokumenata na temelju imena datoteke. Vektorska baza podataka se lokalno pohranjuje u direktorij vectorstore.

---

```

1 import os
2 from langchain_community.embeddings import OpenAIEmbbeddings
3 from langchain_community.vectorstores import FAISS

4 VECTORSTORE_DIR = "vectorstore"

5 def load_vectorstore():
6     embeddings = OpenAIEmbbeddings()
7     if not os.path.exists(VECTORSTORE_DIR):
8         raise FileNotFoundError("Vectorstore not found.")
9     return FAISS.load_local(VECTORSTORE_DIR, embeddings,
10                           allow_dangerous_deserialization=True)

11 def get_or_create_vectorstore(all_docs):
12     embeddings = OpenAIEmbbeddings()

13     if os.path.exists(VECTORSTORE_DIR):
14         db = FAISS.load_local(VECTORSTORE_DIR, embeddings,
15                               allow_dangerous_deserialization=True)
16         if all_docs:
17             db.add_documents(all_docs)
18             db.save_local(VECTORSTORE_DIR)
19         elif all_docs:
20             db = FAISS.from_documents(all_docs, embeddings)
21             db.save_local(VECTORSTORE_DIR)
22     else:

```

---

```

21     raise ValueError("No documents provided and vectorstore doesn't exist.")
22
23     return db
24
25 def delete_file_from_vectorstore(filename):
26     try:
27         db = load_vectorstore()
28     except FileNotFoundError:
29         return False
30
31     remaining_docs = [
32         doc for doc in db.docstore._dict.values()
33         if doc.metadata.get("source") != filename
34     ]
35
36     if len(remaining_docs) == len(db.docstore._dict):
37         return False
38
39     embeddings = OpenAIEmbeddings()
40     new_db = FAISS.from_documents(remaining_docs, embeddings)
41     new_db.save_local(VECTORSTORE_DIR)
42
43     return True

```

Isječak koda 4: Vectorstore.py

Nadalje se definira mehanizam za dohvaćanje odgovora na pitanja pomoću modela gpt-3.5-turbo-16k i prethodno izgrađene vektorske baze. Koristi se RetrievalQA lanac koji kombinira jezični model s retrieverom za dohvrat relevantnih dokumenata. Može se podešiti broj dokumenata koji se koriste za generiranje odgovora.

```

1 from langchain_openai import ChatOpenAI
2 from langchain.chains import RetrievalQA
3
4 def get_qa_chain(db):
5     retriever = db.as_retriever(search_kwargs={"k": 2})
6     llm = ChatOpenAI(model="gpt-3.5-turbo-16k")
7
8     return RetrievalQA.from_chain_type(llm=llm, retriever=retriever)

```

Isječak koda 5: qa.py

### 6.3.3. Rute

Ova API ruta omogućuje korisniku da pošalje jedan ili više PDF dokumenata i postavit pitanje vezano uz njihov sadržaj. PDF dokumenti se obrađuju i dijele na manje dijelove, a zatim se kreira ili nadopunjuje vektorska baza podataka. Jezični model koristi najrelevantnije dijelove za generiranje odgovora. Ruta vraća generirani odgovor zajedno s porukom o uspješnoj obradi i indeksiranju PDF dokumenata.

```

1 from fastapi import APIRouter, UploadFile, File, Form
2 from services.pdf_handler import process_pdfs
3 from services.vectorstore import get_or_create_vectorstore
4 from services.qa import get_qa_chain
5
6 router = APIRouter()
7
8 @router.post("/ask")

```

```

7  async def ask_pdf(
8      question: str = Form(...),
9      pdfs: list[UploadFile] = File(None)
10 ) :
11     all_docs = await process_pdfs(pdfs)
12     db = get_or_create_vectorstore(all_docs)
13     qa_chain = get_qa_chain(db)
14     response = qa_chain.run(question)
15
16     return { "answer": response, "message": "PDF document successfully added and
17         indexed." }

```

### Isječak koda 6: ask.py

Nadalje su definirane dvije rute: jedna za dohvat popisa svih PDF dokumenata koji su trenutno indeksirani u vektorskoj bazi podataka, a drugačije za brisanje dokumenata po nazivu dokumenta. Ruta za brisanje vraća grešku ako dokument s tim imenom ne postoji u vektorskoj bazi podataka.

```

1  from fastapi import APIRouter, HTTPException
2  from services.vectorstore import load_vectorstore, delete_file_from_vectorstore
3
4  router = APIRouter()
5
6  @router.get("/documents")
7  def list_documents():
8      db = load_vectorstore()
9      sources = set()
10     for doc in db.docstore._dict.values():
11         if "source" in doc.metadata:
12             sources.add(doc.metadata["source"])
13
14     return { "documents": list(sources) }
15
16 @router.delete("/delete-document")
17 def delete_document(filename: str):
18     success = delete_file_from_vectorstore(filename)
19     if not success:
20         raise HTTPException(status_code=404, detail=f"No document found with source:
21             {filename}")
22
23     return { "status": "success", "message": f"Deleted document: {filename}" }

```

### Isječak koda 7: documents.py

Sljedeća API ruta omogućuje slanje e-mail poruke s kontakt forme. Korisnik ispunjava ime, email, broj telefona, adresu i poruku, a backend zatim šalje e-mail na zadalu adresu koristeći Gmail SMTP. Podaci se šalju putem SSL-a, a konfiguracija dolazi iz .env datoteke.

```

1  from fastapi import APIRouter
2  from pydantic import BaseModel, EmailStr
3  import smtplib
4  from email.mime.text import MIMEText
5  from email.utils import formataddr
6  from dotenv import load_dotenv
7  import os
8
9  load_dotenv()
10
11 router = APIRouter()

```

```

10  class ContactForm(BaseModel):
11      fullName: str
12      phone: str
13      email: EmailStr
14      address: str
15      description: str

16  @router.post("/send-email")
17  def send_email(data: ContactForm):
18      sender_email = os.getenv("GMAIL_USER")
19      sender_password = os.getenv("GMAIL_PASSWORD")
20      receiver_email = os.getenv("GMAIL_RECEIVER") or sender_email

21      msg_content = f"""
22          Nova poruka s kontakt forme:

23          Ime i prezime: {data.fullName}
24          Telefon: {data.phone}
25          E-mail: {data.email}
26          Adresa: {data.address}
27          Poruka:
28          {data.description}
29          """

30      msg = MIMEText(msg_content, "plain", "utf-8")
31      msg["Subject"] = "Nova poruka s web stranice"
32      msg["From"] = formataddr(("Web kontakt forma", sender_email))
33      msg["To"] = receiver_email
34      msg["Reply-To"] = data.email

35  try:
36      with smtplib.SMTP_SSL("smtp.gmail.com", 465) as server:
37          server.login(sender_email, sender_password)
38          server.sendmail(sender_email, receiver_email, msg.as_string())
39      return {"message": "Email je uspješno poslan."}
40  except Exception as e:
41      print("Greška prilikom slanja emaila:", e)
42  return {"error": "Neuspješno slanje emaila. Provjerite postavke."}

```

Isječak koda 8: email.py

### 6.3.4. Inteligentni asistent

test

#### 6.3.4.1. Prikupljanje podataka

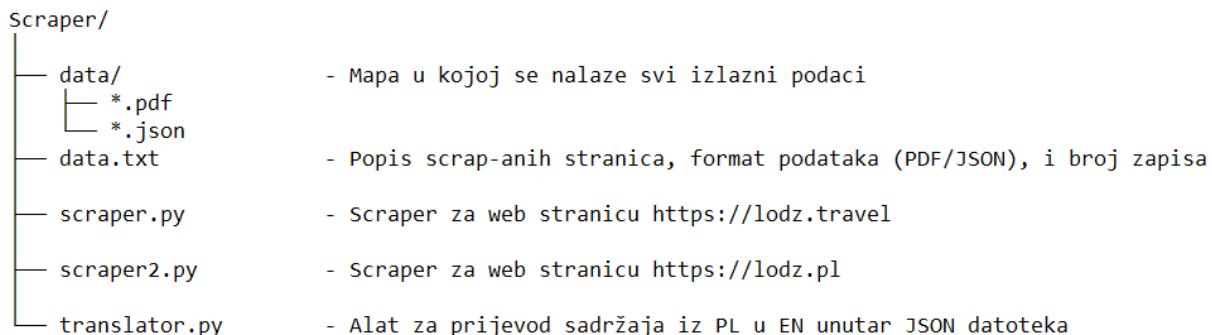
Prikupljanje podataka započelo je kroz razgovor s profesoricom umjetne inteligencije koju sam upoznao tijekom Erasmus+ razmjene. Profesorica je bila izuzetno korisna osoba za ovaj rad jer je rođena i živi u gradu Łódź. Ukratko sam joj predstavio temu rada, a ona mi je preporučila nekoliko relevantnih internetskih izvora.

Sljedeća tablica prikazuje pregled korištenih izvora, broj zapisa, naziv rezultirajuće datoteke i alate korištene za obradu podataka:

Web izvor	Veličina skupa podataka	Naziv datoteke	Korišteni alati
<a href="https://lodz.pl/">https://lodz.pl/</a>	1000 zapisa	lodz_articles	scraper2.py, translator.py
<a href="https://lodz.travel/turystyka/co-zobaczyc/">https://lodz.travel/turystyka/co-zobaczyc/</a>	185 zapisa	lodz_destinations	scraper.py, translator.py
<a href="https://polskapogodzinach.pl/lodz-atrakcje/">https://polskapogodzinach.pl/lodz-atrakcje/</a>	52 stranice	lodz_attractions	ručno konvertirano u PDF

Tablica 5: Korišteni izvori podataka i alati za obradu

#### 6.3.4.2. Web scraping



Slika 16: Struktura scraper foldera

Sljedeća metoda dohvata HTML sadržaj stranice. Prima URL i vraća BeautifulSoup objekt za lakšu obradu HTML-a. Postavlja User-Agent kako bi se izbjeglo blokiranje od strane servera. U slučaju do pogreške (npr. timeout) ispisuje poruku i vraća None.

```

1 import requests
2 from bs4 import BeautifulSoup
3 from urllib.parse import urljoin
4 import json
5 import time

6 BASE_URL = "https://lodz.travel"
7 visited_urls = set()

8 def get_soup(url):
9     print(f"Fetching: {url}")
10    try:
11        res = requests.get(url, headers={
12            "User-Agent": "Mozilla/5.0"
13        })
14        res.encoding = 'utf-8'
15        return BeautifulSoup(res.text, 'html.parser')

```

```
16     except Exception as e:  
17         print(f"Error fetching page: {e}")  
18         return None
```

Isječak koda 9: Scraper 1/6

Ova metoda rekurzivno prolazi kroz listu i sve li stavke u njoj. Za svaku pronađenu stranicu iz izbornika izdvaja se title i href te se stvara struktura koja sadrži title, url, content i subcategories.

```
1 def parse_menu(ul_tag, level=1):
2     data = []
3     for li in ul_tag.find_all("li", recursive=False):
4         a_tag = li.find("a", recursive=False)
5         if not a_tag:
6             continue
7         title = a_tag.text.strip()
8         href = urljoin(BASE_URL, a_tag["href"])
9         print(f"{' ' * (level-1)}({level}) {title} -> {href}")
10        node = {
11            "title": title,
12            "url": href,
13            "content": "",
14        }
15        sub_ul = li.find("ul")
16        if sub_ul:
17            node["subcategories"] = parse_menu(sub_ul, level + 1)
18            data.append(node)
19    return data
```

Isječak koda 10: Scraper 2/6

Sljedeća metoda dohvata sadržaj za svaku stranicu. Ulazi u svaku stranicu u hijerarhiji i izdvaja tekst iz diva s klasom ce-bodytext. Sprema pročišćeni tekst u item["content"] te traži dodatne poveznice unutar sadržaja i dodaje ih kao subcategories i rekursivno ih obaduje.

```
1 def fill_content_recursively(data):
2     for item in data:
3         if item["url"] in visited_urls:
4             continue
5         visited_urls.add(item["url"])
6
7         print(f"\nRetrieving content for: {item['title']}")
8         soup = get_soup(item["url"])
9         if not soup:
10            continue
11
12         content_div = soup.select_one("div.ce-bodytext")
13         if content_div:
14             item["content"] = content_div.get_text(separator="\n", strip=True)
15             print(f"Content retrieved successfully ({len(item['content'])} 
16             ↴ characters)")
17
18             sub_links = []
19             for a in content_div.find_all("a", href=True):
20                 href = urljoin(BASE_URL, a["href"])
21                 if BASE_URL in href and href not in visited_urls:
22                     title = a.get_text(strip=True)
23                     if title:
```

```

20         sub_links.append({
21             "title": title,
22             "url": href,
23             "content": ""
24         })
25
25     if sub_links:
26         print(f"Found {len(sub_links)} internal subpages inside:
27             → {item['title']}")"
27         item["subcategories"] = sub_links
28         fill_content_recursively(sub_links)
29
29     if "subcategories" in item:
30         fill_content_recursively(item["subcategories"])

```

### Isječak koda 11: Scraper 3/6

Ova metoda broji sve stavke (stranice i podstranice) u hijerarhiji podataka. Koristi se za statistiku na kraju izvršavanja.

```

1 def count_total_items(data):
2     count = 0
3     for item in data:
4         count += 1
5         if "subcategories" in item:
6             count += count_total_items(item["subcategories"])
7     return count

```

### Isječak koda 12: Scraper 4/6

Ova metoda pokreće cijeli proces. Kreće od glavne stranice navigacije gdje se dohvata struktura linkova pomoću parse menu funkcije. Zatim se rekursivno dohvata sadržaj sa svih stranica pomoću fill content recursively funkcije. Na kraju svi podaci se spremaju u JSON datoteku i ispisuje se koliko je ukupno stranica pronađeno i spremljeno.

```

1 def main():
2     start_url = "https://lodz.travel/turystyka/"
3     soup = get_soup(start_url)
4     aside_nav = soup.select_one("ul.aside-nav")
5     if not aside_nav:
6         print("Navigation not found! Check the URL or HTML structure.")
7         return
8
8     print("Parsing main navigation...")
9     structured_data = parse_menu(aside_nav)
10
10    print("\nFetching content from all pages...")
11    fill_content_recursively(structured_data)
12
12    with open("lodz_content.json", "w", encoding="utf-8") as f:
13        json.dump(structured_data, f, indent=2, ensure_ascii=False)
14
14    total = count_total_items(structured_data)
15    print(f"\nTotal records extracted: {total}")
16    print("Done! Data saved to: lodz_content.json")
17
17    if __name__ == "__main__":
18        main()

```

### Isječak koda 13: Scraper 5/6

Ova python skripta učitava JSON s člancima na poljskom jeziku i prevodi ih na engleski koristeći biblioteku deep-translator. Funkcija rekurzivno pronalazi kroz sve stavke i podkategorije u JSON strukturi. Napredak prevođenja ispisuje se u konzoli, a prevedeni podaci spremaju se u novu .json datoteku.

```
1 import json
2 from deep_translator import GoogleTranslator
3
4 INPUT_FILE = "lodz_articles_pl.json"
5 OUTPUT_FILE = "lodz_articles_en.json"
6
7 def translate_text(text, src="pl", target="en"):
8     try:
9         return GoogleTranslator(source=src, target=target).translate(text)
10    except Exception as e:
11        return f"[Translation Error] {text}"
12
13 def translate_node(node, counter, total):
14     if "title" in node:
15         node["title"] = translate_text(node["title"])
16     if "content" in node and node["content"]:
17         node["content"] = translate_text(node["content"])
18
19         counter[0] += 1
20         print(f"\n{counter[0]}/{total} translated")
21
22         if "subcategories" in node:
23             for sub in node["subcategories"]:
24                 translate_node(sub, counter, total)
25
26 def count_nodes(data):
27     count = 0
28     for item in data:
29         count += 1
30         if "subcategories" in item:
31             count += count_nodes(item["subcategories"])
32     return count
33
34 def main():
35     print("Loading JSON...")
36     with open(INPUT_FILE, "r", encoding="utf-8") as f:
37         data = json.load(f)
38
39         total = count_nodes(data)
40         print(f"\nTranslating {total} entries...")
41
42         counter = [0]
43         for entry in data:
44             translate_node(entry, counter, total)
45
46         print("Saving translated JSON...")
47         with open(OUTPUT_FILE, "w", encoding="utf-8") as f:
48             json.dump(data, f, ensure_ascii=False, indent=2)
49
50         print(f"\nDone! Translated file saved as: {OUTPUT_FILE}")
51
52 if __name__ == "__main__":
53     main()
```

## Isječak koda 14: Scraper 6/6

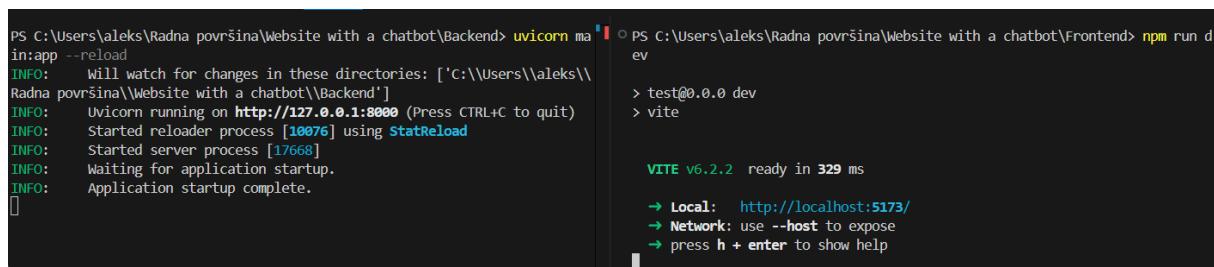
### 6.3.4.3. Čišćenje i obrada podataka

test

## 6.4. Prikaz rada aplikacije

Što se tiče frontenda, projekt je razvijan korištenjem Vite okoline, a pokretanje aplikacije omogućeno je korištenjem naredbe `npm run dev`.

Backend se pokreće lokalno pomoću Uvicorn ASGI servera, korištenjem naredve `uvicorn main:app --reload` koja omogućuje automatsko ponovno učitavanje prilikom promjena u kodu.

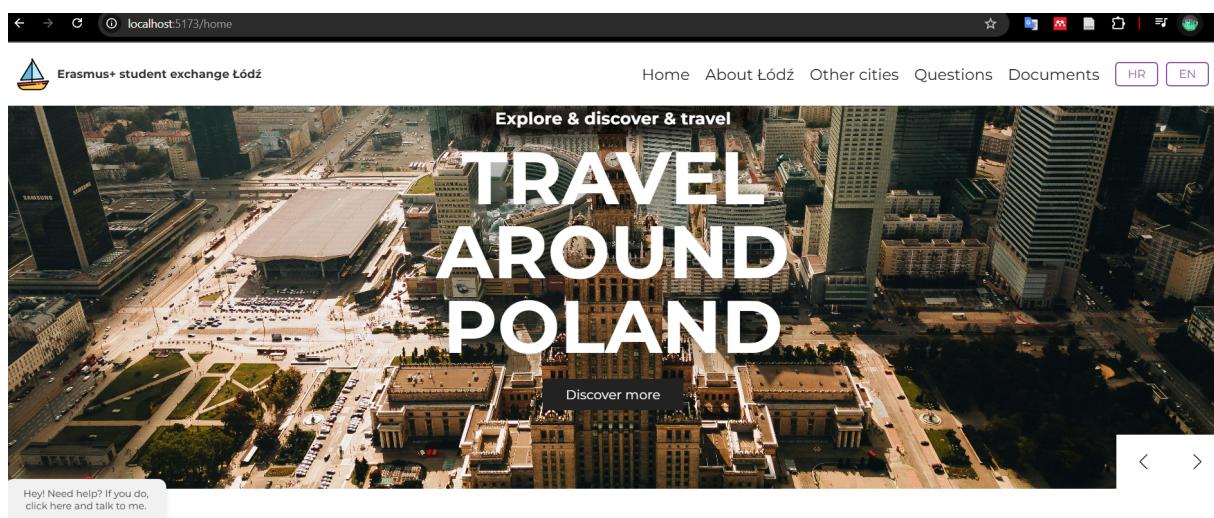


```
PS C:\Users\aleks\Radna površina\Website with a chatbot\Backend> uvicorn main:app --reload
INFO: Will watch for changes in these directories: ['C:\\Users\\aleks\\Radna površina\\Website with a chatbot\\Backend']
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Started reloader process [10076] using StatReload
INFO: Started server process [17668]
INFO: Waiting for application startup.
INFO: Application startup complete.

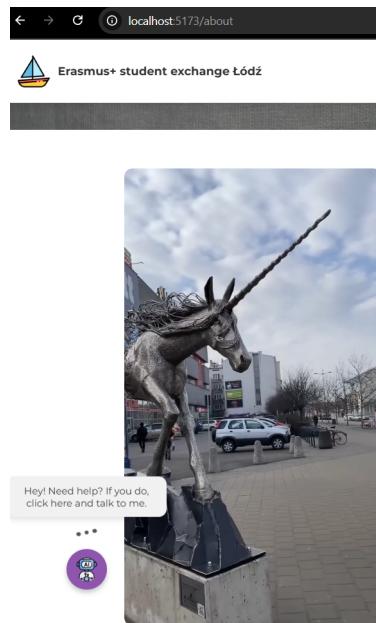
PS C:\Users\aleks\Radna površina\Website with a chatbot\Frontend> npm run dev
> test@0.0.0 dev
> vite

VITE v6.2.2 ready in 329 ms
→ Local: http://localhost:5173/
→ Network: use --host to expose
→ press h + enter to show help
```

Slika 17: test



Slika 18: test



**Why choose Łódź?**

Łódź is a city that perfectly blends a rich industrial heritage with a modern spirit and creative energy. Located in the heart of Poland, it is an ideal base for exploring the entire country. Łódź offers a unique mix of cultural diversity, vibrant arts, and innovative projects, creating an inspiring environment for students, young professionals, and travelers alike.

The city is renowned for its murals, historic factories transformed into modern spaces, and excellent transport connections. With affordable living costs and a welcoming atmosphere, Łódź is a perfect choice for those seeking a dynamic yet comfortable experience.



Safe

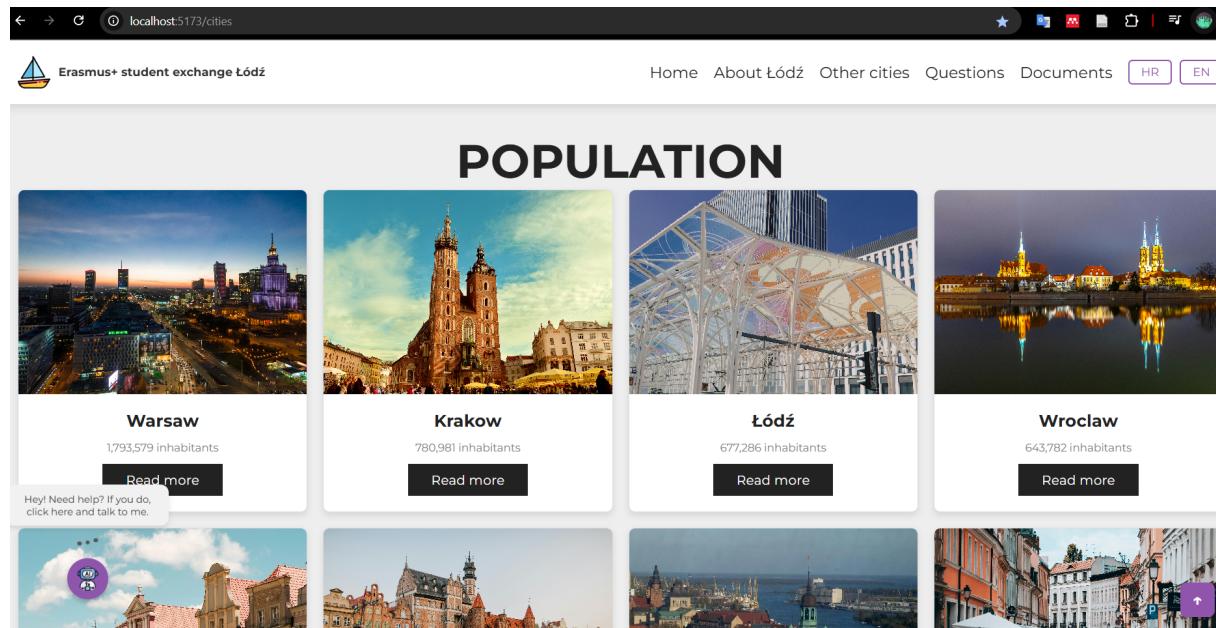


Clean



Modern

Slika 19: test



## POPULATION



**Warsaw**  
1,793,579 inhabitants

[Read more](#)

Hey! Need help? If you do, click here and talk to me.



**Krakow**  
780,981 inhabitants

[Read more](#)



**Łódź**  
677,286 inhabitants

[Read more](#)



**Wrocław**  
643,782 inhabitants

[Read more](#)









Slika 20: test

If you have any questions or need assistance, feel free to reach out – I'm here to help.

**Full Name:**  
John Doe

**Phone:**  
(123) 456-7890

**Email:**  
john.doe@example.com

**Address:**  
123 Main St, Cityville

**Description:**  
Briefly describe your inquiry or issue

Hey! Need help? If you do, click here and talk to me.

Slika 21: test

**Statistics**

- Total documents: 2
- : 129
- Last added: babic\_aleksandar\_foi\_2023\_predr\_sveuc.pdf

**Document Management**

File	Action
Chapter_Springer.pdf	Delete
babic_aleksandar_foi_2023_predr_sveuc.pdf	Delete

**Add PDF Documents**

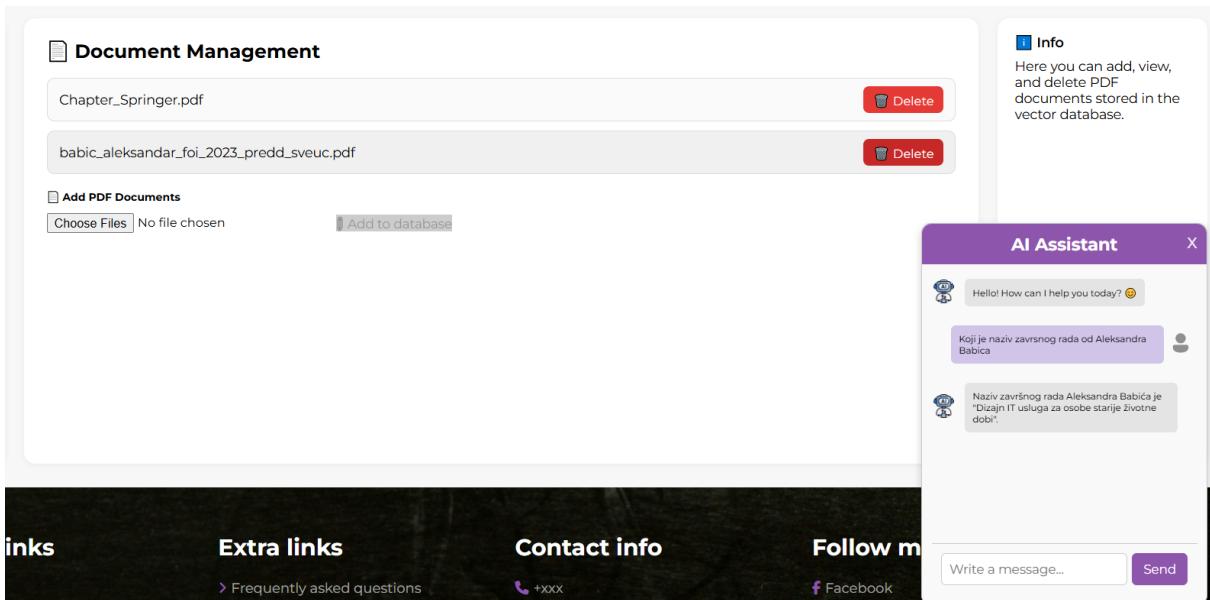
Choose Files | No file chosen | Add to database

**Info**

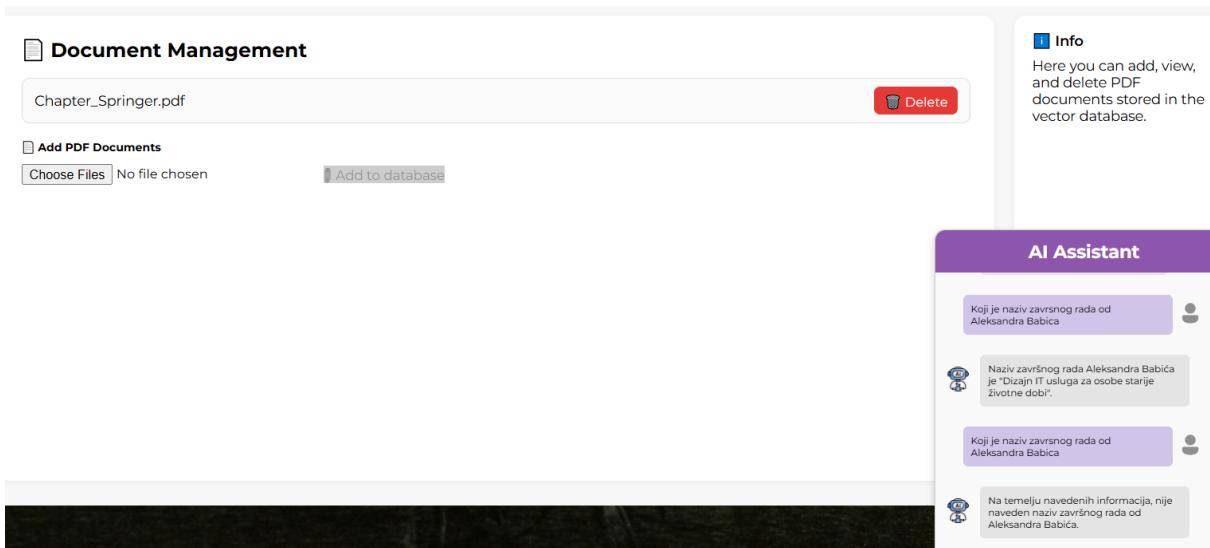
Here you can add, view, and delete PDF documents stored in the vector database.

Hey! Need help? If you do, click here and talk to me.

Slika 22: test



Slika 23: test



Slika 24: test

```

response = qa_chain.run(question)
INFO:    127.0.0.1:16127 - "POST /ask HTTP/1.1" 200 OK
INFO:    127.0.0.1:16134 - "OPTIONS /delete-document?filename=babic_aleksandar_foi_2023_predd_sveuc.pdf HTTP/1.1" 200 OK
INFO:    127.0.0.1:16134 - "DELETE /delete-document?filename=babic_aleksandar_foi_2023_predd_sveuc.pdf HTTP/1.1" 200 OK
INFO:    127.0.0.1:16134 - "GET /documents HTTP/1.1" 200 OK
INFO:    127.0.0.1:16135 - "GET /stats HTTP/1.1" 200 OK
INFO:    127.0.0.1:16135 - "POST /ask HTTP/1.1" 200 OK

```

Slika 25: test

## **7. Zaključak**

Za kraj, Erasma se ne treba bojati jer je to prekasno iskustvo koje se pamti do kraja života. Svaki problem s kojim se suoči osoba, čini osobu jačom i neovisnijom. Boravak na jednom mjestu dulje vrijeme postat će čudan način života. Jednom Erasmus+, uvijek Erasmus+!

Poveznica na Github: **Poveznica**

# Popis literature

- [1] Digital 360. „Are studies important in life? If yes, then why?” Pristupljeno: 10. svibanj 2025. adresa: <https://digital360india.medium.com/are-studies-important-in-life-if-yes-then-why-4583029b95e3>.
- [2] K. Bisio. „15 Reasons Why College is Important for Your Career.” Pristupljeno: 11. svibanj 2025. adresa: <https://www.msudenver.edu/15-reasons-why-college-is-important-for-your-career/>.
- [3] „What is Erasmus+? - Erasmus+.” Pristupljeno: 27. svibanj 2025. adresa: <https://erasmus-plus.ec.europa.eu/about-erasmus/what-is-erasmus>.
- [4] „Data on Erasmus+ in Croatia in 2023 - Erasmus+.” Pristupljeno: 10. svibanj 2025. adresa: <https://erasmus-plus.ec.europa.eu/factsheets/2023/croatia>.
- [5] „Studying abroad - Erasmus+.” Pristupljeno: 10. svibanj 2025. adresa: <https://erasmus-plus.ec.europa.eu/opportunities/opportunities-for-individuals/students/studying-abroad>.
- [6] Erasmus Generation Blog. „Six common Erasmus fears and how to deal with them.” Pristupljeno: 22. svibanj 2025. adresa: <https://blog.erasmusgeneration.org/six-common-erasmus-fears-and-how-deal-them>.
- [7] Erasmus Student Network AISBL. „Erasmus Careers - Feeling homesick.” Video. Objavljeno: 15. listopad 2024. Pristupljeno: 22. svibanj 2025. adresa: [https://www.youtube.com/watch?v=TT\\_\\_3yOkAdk](https://www.youtube.com/watch?v=TT__3yOkAdk).
- [8] c2061348. „The difference between ChatBot, AI Assistant, Copilot, and AI Agent.” Pristupljeno: 10. svibanj 2025. adresa: <https://exomindset.co/the-difference-between-chatbot-ai-assistant-copilot-and-ai-agent/>.
- [9] J. Buchan. „Why you should add an AI assistant to your app or website.” Pristupljeno: 06. svibanj 2025., Zudu. adresa: <https://zudu.co.uk/blog/why-you-should-add-an-ai-assistant-to-your-app-or-website/>.
- [10] E. Mixon. „What is Siri? | Definition from TechTarget.” Pristupljeno: 20. lipanj 2025. adresa: <https://www.techtarget.com/searchmobilecomputing/definition/Siri>.
- [11] „Create Virtual Agents with Copilot | Microsoft Copilot Studio.” Pristupljeno: 27. svibanj 2025. adresa: <https://www.microsoft.com/en-us/copilot/microsoft-copilot-studio>.

- [12] iaanw. „Quotas and limits - Microsoft Copilot Studio.” Pristupljeno: 27. svibanj 2025. adresa: <https://learn.microsoft.com/en-us/microsoft-copilot-studio/requirements-quotas>.
- [13] „Introducing Operator.” Pristupljeno: 10. svibanj 2025. adresa: <https://openai.com/index/introducing-operator/>.
- [14] Riley Brown. „TikTok video objavljen 24. siječnja 2025.” [Online video]. Pristupljeno: 4. lipanj 2025., TikTok. adresa: <https://www.tiktok.com/@rileybrown.ai/video/7463264903327976734>.
- [15] „AI for customer service.” Pristupljeno: 20. lipanj 2025. adresa: <https://www.zendesk.com/service/ai/>.
- [16] „Gradescope | Save time grading.” Pristupljeno: 26. svibanj 2025. adresa: <https://www.gradescope.com/>.
- [17] P. Weber. „SmythOS - Intelligent Agents in Education.” Pristupljeno: 26. svibanj 2025. adresa: <https://smythos.com/ai-industry-solutions/education/intelligent-agents-in-education/>.
- [18] Amazon Web Services, Inc. „What is RAG? - Retrieval-Augmented Generation AI Explained - AWS.” Pristupljeno: 06. svibanj 2025. adresa: <https://aws.amazon.com/what-is/retrieval-augmented-generation/>.
- [19] Marina Danilevsky, *What is Retrieval-Augmented Generation (RAG)?* [Online video]. Pristupljeno: 6. svibanj 2025., IBM Technology, 8.2023.
- [20] IBM Research. „What is retrieval-augmented generation (RAG)?” Pristupljeno: 06. svibanj 2025. adresa: <https://research.ibm.com/blog/retrieval-augmented-generation-RAG>.
- [21] R. W. G. Studio) MD (Custom AI. „This history of Retrieval-Augmented Generation in 3 minutes...!” Medium, pogledano 29. svibnja 2025. adresa: [https://medium.com/@custom\\_aistudio/this-history-of-retrieval-augmented-generation-in-3-minutes-f7f07073599a](https://medium.com/@custom_aistudio/this-history-of-retrieval-augmented-generation-in-3-minutes-f7f07073599a).
- [22] Redis. „Introduction to Retrieval Augmented Generation (RAG).” Redis, pogledano 29. svibnja 2025. adresa: <https://redis.io/glossary/retrieval-augmented-generation/>.
- [23] Shaheryar. „Understanding the Key Components of RAG: Retriever and Generator.” Pristupljeno: 29. svibanj 2025. adresa: <https://dev.to/shaheryaryousaf/understanding-the-key-components-of-rag-retriever-and-generator-1a1j>.
- [24] S. Gupta, R. Ranjan i S. N. Singh, „A Comprehensive Survey of Retrieval-Augmented Generation (RAG): Evolution, Current Landscape and Future Directions,” *arXiv preprint arXiv:2410.12837*, 10.2024., Preuzeto: 3. listopada 2024. DOI: 10.48550/arXiv.2410.12837.
- [25] A. Siddharth. „A practical guide to RAG implementation: Part I | LinkedIn.” Pristupljeno: 04. lipanj 2025. adresa: <https://www.linkedin.com/pulse/practical-guide-rag-implementation-part-i-siddharth-asthana-bppbe/>.

- [26] Aim. Labs i P. Anand. „Top Problems with RAG systems and ways to mitigate them - AIMon Labs.” Pristupljeno: 2025-06-04. adresa: [https://www.aimon.ai/posts/top%5C\\_problems%5C\\_with%5C\\_rag%5C\\_systems%5C\\_and%5C\\_ways%5C\\_to%5C\\_mitigate%5C\\_them](https://www.aimon.ai/posts/top%5C_problems%5C_with%5C_rag%5C_systems%5C_and%5C_ways%5C_to%5C_mitigate%5C_them).
- [27] K2View. „RAG Hallucination: What is It and How to Avoid It,” pogledano 29. svibnja 2025. adresa: <https://www.k2view.com/blog/rag-hallucination/>.
- [28] I. Novogroder. „Top 9 RAG Tools to Boost Your LLM Workflows.” Pristupljeno: 2025-06-02. adresa: <https://lakefs.io/blog/rag-tools/>.
- [29] S. Aquino. „An Introduction to Vector Databases - Qdrant.” Pristupljeno: 2. srpnja 2025. adresa: <https://qdrant.tech/articles/what-is-a-vector-database/>.
- [30] V. Baghel. „13 Best Vector Databases for AI-Powered Solutions.” Openxcell. Pristupljeno: 04. lipanj 2025. adresa: <https://www.openxcell.com/blog/best-vector-databases/>.
- [31] M. Ozkaya. „LLM Providers: OpenAI, Meta AI, Anthropic, Hugging Face, Microsoft, Google, and Mistral AI.” Medium. Pristupljeno: 04. lipanj 2025. adresa: <https://mehmetozkaya.medium.com/llm-providers-openai-meta-ai-anthropic-hugging-face-microsoft-google-and-mistral-ai-46ad8c027f6b>.
- [32] „What Is React? [Easily Explained].” Pristupljeno: 11. svibanj 2025. adresa: <https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs>.
- [33] Obafemi. „Building a Website with Python, FastAPI, and Streamlit.” Pristupljeno: 7. srpnja 2025., pogledano 7. srpnja 2025. adresa: <https://medium.com/@obaff/building-a-website-with-python-fastapi-and-streamlit-418f48c41af2>.

# Popis slika

1.	Primjer copilota kreiranog u Microsoft Copilot Studiu . . . . .	10
2.	Adaptive cards . . . . .	10
3.	Ocjenvivanje zadatka - Gradescope [16] . . . . .	14
4.	Popularnost RAG-a; Izvor . . . . .	17
5.	Konceptualni tok korištenja RAG-a s LLM-ovima [18] . . . . .	19
6.	Sažetak dostupnih alata i biblioteka za razvoj RAG sustava; Izvor . . . . .	26
7.	Vektorska baza - primjer; Izvor . . . . .	29
8.	OpenAI - ikona; Izvor . . . . .	30
9.	Hugging Face - ikona; Izvor . . . . .	31
10.	Anthropic - ikona; Izvor . . . . .	31
11.	React komponenta . . . . .	32
12.	FastAPI - ikona; Izvor . . . . .	33
13.	Struktura frontend dijela . . . . .	35
14.	Struktura backend dijela . . . . .	36
15.	Sadržaj env datoteke . . . . .	36
16.	Struktura scraper foldera . . . . .	41
17.	test . . . . .	45
18.	test . . . . .	45
19.	test . . . . .	46
20.	test . . . . .	46
21.	test . . . . .	47
22.	test . . . . .	47
23.	test . . . . .	48

24. test . . . . .	48
25. test . . . . .	48

# **Popis tablica**

1.	Razlika između pojmove [8] . . . . .	7
2.	Prednosti RAG-a [18] . . . . .	23
3.	Usporedba OLTP, OLAP i vektorskih baza podataka [29] . . . . .	29
4.	Struktura stranica i njihovih komponenti . . . . .	34
5.	Korišteni izvori podataka i alati za obradu . . . . .	41

# Popis isječaka koda

1.	Chat.jsx . . . . .	35
2.	Contact.jsx . . . . .	36
3.	Main.py . . . . .	37
4.	Vectorstore.py . . . . .	38
5.	qa.py . . . . .	38
6.	ask.py . . . . .	39
7.	documents.py . . . . .	39
8.	email.py . . . . .	40
9.	Scraper 1/6 . . . . .	42
10.	Scraper 2/6 . . . . .	42
11.	Scraper 3/6 . . . . .	43
12.	Scraper 4/6 . . . . .	43
13.	Scraper 5/6 . . . . .	44
14.	Scraper 6/6 . . . . .	44