Sustav korisničke podrške temeljen na bazi znanja i korištenju alata umjetne inteligencije za brzo odgovaranje na ponavljajuća pitanja korisnika

Balen, Ian

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1297

SUSTAV KORISNIČKE PODRŠKE TEMELJEN NA BAZI ZNANJA I KORIŠTENJU ALATA UMJETNE INTELIGENCIJE ZA BRZO ODGOVARANJE NA PONAVLJAJUĆA PITANJA KORISNIKA

Ian Balen

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1297

SUSTAV KORISNIČKE PODRŠKE TEMELJEN NA BAZI ZNANJA I KORIŠTENJU ALATA UMJETNE INTELIGENCIJE ZA BRZO ODGOVARANJE NA PONAVLJAJUĆA PITANJA KORISNIKA

Ian Balen

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1297

Pristupnik:	**Ian Balen (0036538107)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentorica:	prof. dr. sc. Ljiljana Brkić

Zadatak: **Sustav korisničke podrške temeljen na bazi znanja i korištenju alata umjetne inteligencije za brzo odgovaranje na ponavljajuća pitanja korisnika**

Opis zadatka:

Uloga korisničke podrške je ključna za održavanje zadovoljnih klijenata i izgradnju dugoročnih odnosa s njima. Brzo odgovaranje na pitanja korisniku pruža osjećaj podrške i važnosti, čime se poboljšava njihovo iskustvo i lojalnost. Jedan od načina da se ubrza proces odgovaranja na pitanja je korištenje baze znanja s prethodnim pitanjima i odgovorima. To omogućuje agentima za korisničku podršku da brzo pronađu relevantne informacije i da ih dijele s klijentima. Osim toga, baza znanja može se koristiti za edukaciju klijenata o često postavljenim pitanjima i za poticanje samostalnog rješavanja problema. Potrebno je proučiti i usporediti postojeće sustave za korisničku podršku, a potom osmisliti i implementirati vlastiti sustav. Sustav treba omogućiti praćenje, upravljanje i rješavanje problema i upita korisnika i imati funkcionalnosti tipičnog ticketing sustava. Dodatno, korištenjem prikupljenih korisničkih pitanja i odgovora treba izgraditi komponentu koja će primjenom prikladnih metoda umjetne inteligencije analizirati dokumentaciju, identificirati obrasce i veze među podacima te automatski predložiti efikasna rješenja/odgovore za ponavljajuće probleme/pitanja. Donijeti ocjenu ostvarenog pristupa, navesti glavne izazove u radu te smjernice za budući razvoj.

Rok za predaju rada: 14. lipnja 2024.

Sadržaj

Uvod

U današnjem digitalnom dobu, tvrtke neprestano traže načine kako poboljšati interakciju s klijentima i unaprijediti efikasnost svojih usluga podrške. U svijetu gdje su korisnička očekivanja na visokoj razini, sposobnost brzog i preciznog rješavanja upita i problema postaje ključna za održavanje konkurentnosti i zadovoljstva klijenata. U ovom kontekstu, tehnologije umjetne inteligencije, posebno veliki jezični modeli (*Large Language Models*, *LLM*), nude obećavajući pristup za premošćivanje jaza između kompleksnih korisničkih zahtjeva i potrebe za brzom, relevantnom podrškom.

Ovaj završni rad fokusira se na integraciju naprednih tehnologija umjetne inteligencije unutar internih procesa podrške kompanije, s posebnim naglaskom na brzo pronalaženje adekvatnih informacija u bogatim internim bazama znanja. S obzirom na to da tvrtke često raspolažu obimnim repozitorijima dokumentacije, uključujući upute, hodograme i rješenja za česte probleme, ključni izazov leži u efikasnom pretraživanju i korištenju ovih resursa za rješavanje specifičnih upita. Tradicionalni pristupi mogu biti vremenski zahtjevni i ne uvijek efikasni, stvarajući potrebu za inovativnim rješenjima koja mogu dinamički dohvatiti i primijeniti relevantne informacije.

Izrađen je „*chatbot*“ temeljen na principima *Retrieval Augmented Generation* (RAG) i *Large Language Model* (*LLM*), koji bi omogućio brz i precizan pristup potrebnim informacijama koristeći prirodne jezične upite. Implementacijom ovakvog sistema, kompanije bi mogle značajno unaprijediti učinkovitost svojih usluga podrške, smanjiti vrijeme potrebno za obuku novih agenata te poboljšati ukupno korisničko iskustvo.

U ovom radu su istraženi postojeći izazovi s kojima se tvrtke suočavaju prilikom pristupa i korištenja internih baza znanja, analizirane su potencijalne prednosti integracije tehnologija umjetne inteligencije u te procese, te je predloženo praktično rješenje koje koristi najnovija dostignuća u području *LLM*-ova i *RAG*-a. Kroz teorijski okvir i praktičnu implementaciju, rad pokazuje kako tehnološki napredak može transformirati internu podršku i upravljanje znanjem unutar IT tvrtke, čime se otvara moguće novo poglavlje u interakciji s klijentima i unutarnjoj efikasnosti.

1. Terminologija i teorija

1.1. Uvod u *Retrieval-Augmented Generation* (RAG)

Retrieval-Augmented Generation (*RAG*) je metoda koja kombinira generativne jezične modele s mehanizmima za dohvaćanje informacija iz vanjskih izvora kako bi se stvorili kontekstualno bogati i točni odgovori. Ova metoda koristi dohvaćanje podataka iz baza podataka ili drugih vanjskih izvora kako bi obogatila kontekst koji jezični model koristi za generiranje odgovora. Korištenjem *RAG*-a, modeli mogu generirati odgovore koji su relevantniji i precizniji u usporedbi s tradicionalnim jezičnim modelima koji se oslanjaju samo na podatke ugrađene tijekom treniranja.

1.2. Pozadina i razvoj RAG-a

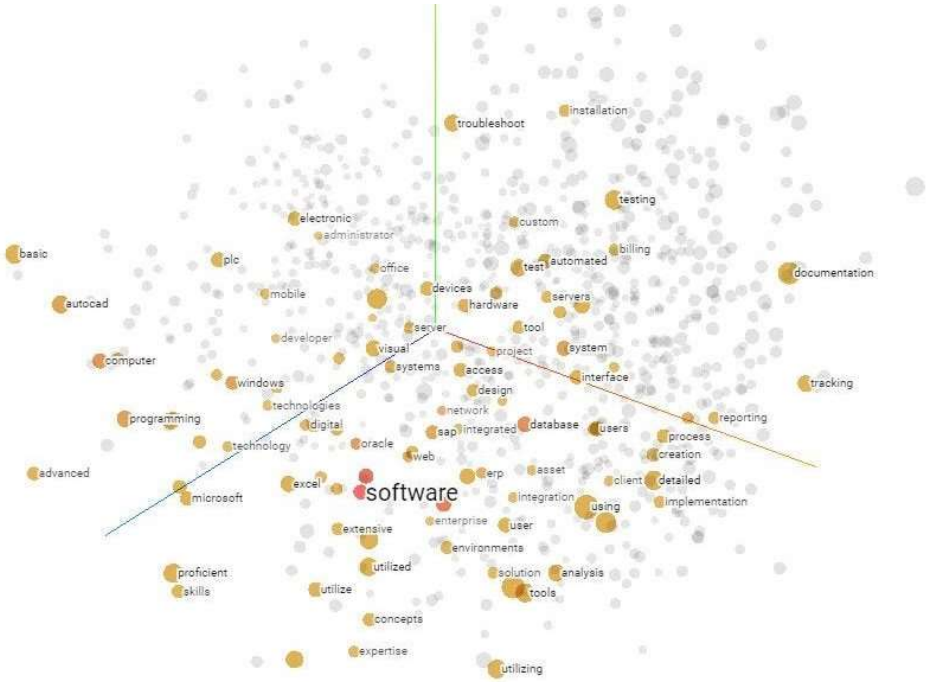
Tradicionalni jezični modeli, iako sposobni za generiranje koherentnog teksta, često pate od nedostatka specifičnog konteksta i mogu proizvesti odgovore koji su netočni ili generički. RAG tehnologija razvijena je kao odgovor na ove izazove kombiniranjem sposobnosti jezičnih modela s moćima sustava za dohvaćanje informacija. U radu "*Investigating the performance of Retrieval-Augmented Generation and fine-tuning for the development of AI-driven knowledge-based systems*" ističu da je "*RAG posebno učinkovit u smanjenju fenomena halucinacije u usporedbi s modelima finog podešavanja*" te da je "*RAG sustav u prosjeku nadmašio FN modele za 16% u pogledu ROGUE rezultata, 15% u pogledu BLEU rezultata i 53% u pogledu cosine sličnosti*" (Lakatos i sur., 2024.).

ROGUE i BLEU su metrike za procjenu kvalitete generiranog teksta. ROGUE (*Recall-Oriented Understudy for Gisting Evaluation*) ocjenjuje preklapanje između generiranog teksta i referentnih tekstova, dok BLEU (*Bilingual Evaluation Understudy*) mjeri preciznost preklapanja. Cosine sličnost, koja će biti objašnjena u sljedećem potpoglavlju, koristi se za mjerenje sličnosti između tekstova pohranjenih u obliku vektora riječi (tokena).

1.3. Cosine sličnost

Cosine sličnost je metrika koja se koristi za mjerenje sličnosti između dva vektora, često primijenjena u obradi prirodnog jezika za mjerenje sličnosti između tekstualnih dokumenata. Izračunava kosinusni kut između dva vektora, što daje vrijednost između -1 i 1. Vrijednost blizu 1 ukazuje na visoku sličnost, dok vrijednost blizu -1 ukazuje na visoku različitost. U *RAG* sustavima, cosine sličnost se koristi za usporedbu generiranih odgovora s referentnim tekstovima kako bi se procijenila njihova točnost i sličnost.

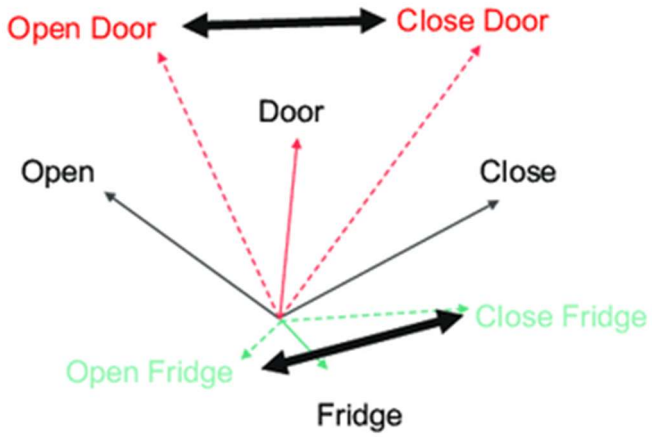
Na slici 1 prikazana je vizualizacija ugradnje teksta („*text embedding*“), koja pokazuje kako se riječi mogu predstaviti u vektorskom prostoru. Ovakva vizualizacija pomaže razumjeti kako se tekstualni podaci mogu strukturirati i analizirati pomoću vektorskih modela.



Slika 1: Vizualizacija ugradnje teksta (Chotrani, 2024).

Na slici 2 prikazan je vektorski prikaz teksta i njihovo moguće zbrajanje. Ova slika ilustrira kako se značenja različitih tekstualnih izraza mogu kombinirati pomoću vektorske aritmetike. Primjerice, vektor za izraz "Open Door" dobiva se zbrajanjem vektora za "Open"

i "Door". Slično tome, vektor za "Close Door" dobiva se zbrajanjem vektora za "Close" i "Door". Ovo zbrajanje vektora omogućava modelu da razumije složenije pojmove na temelju njihovih sastavnih dijelova. Ovakav prikaz omogućava da se pomoću kosinusne sličnosti između vektora mjeri sličnost značenja između različitih tekstualnih izraza. Kosinus kuta između dva vektora pruža mjeru sličnosti koja je neovisna o njihovoj veličini, što je posebno korisno za analizu semantičkih odnosa između izraza.



Slika 2: Vektorski prikaz teksta i njihovo moguće zbrajanje (Matuski, M., Lago, P. & Sozo, I., 2019)

1.4. Primjena RAG-a u aplikacijama

RAG tehnologija ima široku primjenu u različitim domenama:

1. **Pametna poljoprivreda**:

- U radu "*Web Application for Retrieval-Augmented Generation: Implementation and Testing*" (Radeva i sur., 2024). razvili su aplikaciju PaSSER za testiranje modela u domenu pametne poljoprivrede, koja koristi *RAG* sustav za generiranje specifičnih i informiranih odgovora vezanih uz poljoprivredne prakse, bolesti biljaka i vremenske uvjete .

2. **Zdravstvena skrb**:

- Pružanje točnih medicinskih informacija pacijentima i liječnicima, integriranjem medicinskih baza podataka i najnovijih istraživanja s jezičnim modelima. Ističe se da "*RAG*

tehnologija može značajno poboljšati točnost i relevantnost medicinskih informacija" (Lakatos i sur., 2024).

3. **Financijske usluge**:

- Generiranje preciznih financijskih izvješća i analiza korištenjem relevantnih financijskih podataka. "*Integracija vanjskih financijskih podataka s jezičnim modelima omogućava stvaranje točnijih i informiranijih financijskih analiza*" (Lakatos i sur., 2024).

4. **Obrazovanje**:

- Stvaranje personaliziranih obrazovnih materijala i odgovora na pitanja učenika na temelju velikih baza podataka obrazovnih sadržaja. Huang i sur. navode da "*RAG sustavi mogu transformirati način na koji se obrazovni sadržaji prilagođavaju potrebama učenika*". (Huang i sur., 2023)

1.5. Prednosti RAG-a nad finim podešavanjem (FN)

RAG ima nekoliko ključnih prednosti u odnosu na *fine-tuning* (FN):

- **Smanjenje halucinacija:** U kontekstu umjetne inteligencije, "halucinacija" se odnosi na situaciju kada model generira odgovore koji su netočni ili izmišljeni, a nemaju osnovu u stvarnim podacima ili informacijama. *RAG* smanjuje rizik od halucinacija jer dohvaća relevantne informacije iz vanjskih izvora, dok *FN* modeli često pate od generiranja netočnih informacija. Na taj način, *RAG* omogućuje generiranje točnijih i pouzdanijih odgovora.

- **Veća efikasnost**: *RAG* sustav je dokazano učinkovitiji u usporedbi s modelima finog podešavanja (*FN*). Kroz razne evaluacije, uključujući *ROGUE*, *BLEU* i *cosine* sličnost, pokazalo se da *RAG* modeli postižu bolje rezultate u generiranju preciznih i relevantnih odgovora. Ova efikasnost proizlazi iz sposobnosti *RAG*-a da koristi dodatne vanjske informacije za poboljšanje točnosti generiranih odgovora.

- **Širi opseg znanja**: *RAG* može koristiti dodatne informacije koje nisu uključene u inicijalni trening modela, čime se omogućuje kontinuirano ažuriranje i obogaćivanje baze znanja.

2. Funkcionalni zahtjevi

U okviru završnog rada razvijen je sustav čiji je primarni cilj unaprijediti internu komunikaciju i pristup informacijama unutar timova korisničke podrške. Sustav koristi napredne tehnologije umjetne inteligencije, uključujući velike jezične modele i metode temeljene na principu *Retrieval Augmented Generation* (*RAG*), kako bi omogućio efikasno pretraživanje internih baza znanja te pružio podršku zaposlenicima u realnom vremenu. Kroz sljedeće funkcionalne zahtjeve detaljno je opisano kako korisnici mogu koristiti sustav te kako *AI* komponente pridonose njegovoj funkcionalnosti.

2.1. Korisnik

Pod terminom 'korisnik' smatra se zaposlenik odjela korisničke podrške

2.1.1. Slanje poruke

- Korisnik može kreirati i poslati tekstualnu poruku pomoću korisničkog sučelja. Ova poruka predstavlja upit na koji se očekuje odgovor korištenjem integriranog *AI* sustava. Upiti mogu uključivati pitanja ili probleme koje korisnik želi riješiti uz pomoć sustava
- Sustav omogućava unos i slanje poruka u realnom vremenu

2.1.2. Prikaz poruka

- Korisnik može vidjeti sve poslane poruke/upite
- Poruke se prikazuju u kronološkom redoslijedu, s najnovijim porukama pri dnu

2.1.3. Pregled pojedinačnog razgovora

- Korisnik može odabrati bilo koji razgovor kako bi vidio sve poruke koje su dio tog razgovora
- Sustav prikazuje sve poruke unutar odabranog razgovora, omogućujući korisniku detaljan uvid u komunikaciju
- Sustav u svojem sučelju pokazuje i sve hiperveze vezane uz razgovor

2.2. Sustav

Pod ‘sustav’ se smatra poslužiteljska i klijentska aplikacija s bazama podataka

2.2.1. Interaktivno korisničko sučelje

- Sustav pruža sučelje koje omogućava laku navigaciju i uporabu

2.2.2. Sigurnost i privatnost

- Pristup sustavu je ograničen na ovlaštene zaposlenike korisničke podrške

2.2.3. Umjetna inteligencija

- Korisnik unosi upit, sustav koristi *LLM* i *RAG* (s vektorskom bazom) za analizu upita i pronalazak najrelevantnijih informacija unutar baze znanja
- Sustav dinamički prilagođava odgovore na temelju konteksta razgovora, koristeći povratne informacije od korisnika za unapređenje preciznosti
- Sustav omogućava korisnicima postavljanje dodatnih pitanja ili traženje pojašnjenja u vezi s dobivenim odgovorima, koristeći *AI* za generiranje prirodnih i kontekstualno relevantnih odgovara

2.3. Ostali zahtjevi

- Podrška za paralelni rad više korisnika u stvarnom vremenu
- Automatsko ažuriranje korisničkog sučelja
- Otpornost na greške
- Jednostavno održavanje sustava
- Optimizacija interakcije s umjetnom inteligencijom

3. Tehnologije, alati i arhitektura

Pri razvoju inovativnog sustava za unapređenje korisničke podrške kroz umjetnu inteligenciju, ključno je odabrati prave tehnologije i alate koji će omogućiti efikasnu implementaciju i skalabilnost projekta. Ovaj segment detaljnije opisuje tehnološki stog koji je korišten za razvoj klijentske i poslužiteljske strane aplikacije, kao i tehnologije baze podataka prilagođene specifičnim potrebama projekta.

3.1. Tehnologije klijentske strane

Za razvoj korisničkog sučelja, korištena je *React*, moderna *JavaScript* biblioteka dizajnirana za izgradnju dinamičnih i reaktivnih web aplikacija. Zahvaljujući svojoj komponentnoj arhitekturi, *React* omogućava brzo razvijanje interaktivnih korisničkih sučelja koji mogu efikasno ažurirati pravilne komponente u trenutku kada se podaci promijene. Ova karakteristika čini *React* idealnim za razvoj aplikacija poput *chatbot*-a, gdje brzina i reaktivnost predstavljaju ključne faktore za osiguravanje pozitivnog korisničkog iskustva.

3.2. Tehnologije poslužiteljske strane

Na poslužiteljskoj strani, aplikacija je izgrađena koristeći *Java* programski jezik, uz korištenje *Spring Boot* radnog okvira (*framework*). Spring Boot nudi širok spektar funkcionalnosti koji olakšavaju razvoj, od jednostavne konfiguracije do integracije sa sigurnosnim protokolima i upravljanjem bazama podataka. Za potrebe ovog projekta, posebno je bila važna integracija s *LangChain4j*, Java bibliotekom koja omogućava laku integraciju s tehnologijama umjetne inteligencije i velikim jezičnim modelima. *LangChain4j* služi kao most između Java aplikacija i naprednih *AI* modela, omogućavajući razvojnim timovima da implementiraju kompleksne *AI* funkcionalnosti poput generiranja teksta, razumijevanja prirodnog jezika i dohvaćanja informacija na efikasan i učinkovit način. Također, bilo je potrebno i povezivanje sa drugim sustavima poput *Azure* i *Confluence*.

3.3. Tehnologije baze podataka

Za upravljanje i pretraživanje vektorskih reprezentacija dokumenata koristila se *Qdrant*, specijalizirana vektorska baza podataka dizajnirana za brzo i efikasno pretraživanje kroz

velike količine vektorskih podataka. *Qdrant* je optimizirana za rad s vektorima, što je čini idealnim izborom za aplikacije koje se oslanjaju na semantičko pretraživanje i analizu velikih skupova nestrukturiranih tekstualnih podataka. Osim mogućnosti brzog dohvaćanja relevantnih dokumenata na temelju semantičke sličnosti, *Qdrant* nudi i skalabilnost potrebnu za upravljanje rastućim volumenima podataka kako se baza znanja proširuje.

Kombinacija *React*-a, *Spring Boot*-a uz *LangChain4j*, i *Qdrant* tehnologija predstavlja temelj za razvoj sofisticiranog sustava koji integrira umjetnu inteligenciju u procese korisničke podrške. Ovaj tehnološki stog omogućit će izgradnju robusne, efikasne i skalabilne aplikacije sposobne za obradu kompleksnih upita i pružanje brzih, relevantnih odgovora korisnicima.

3.4. Arhitektura

Arhitektura sustava temelji se na tri sloja (*Three Tier Architecture*), koja osigurava jasnu podjelu odgovornosti, skalabilnost i održivost aplikacije. Ova arhitektura dijeli sustav na tri glavna sloja: prezentacijski sloj, aplikacijski sloj i sloj baze podataka.

3.4.1. Prezentacijski sloj

Prezentacijski sloj (*Presentation Tier*) predstavlja korisničko sučelje i služi za interakciju korisnika s aplikacijom. Ovaj sloj je implementiran korištenjem *React* biblioteke, koja omogućuje izgradnju dinamičnih i interaktivnih korisničkih sučelja. *React* pruža brzu i efikasnu manipulaciju *DOM*-om, čime se osigurava visoka responzivnost aplikacije.

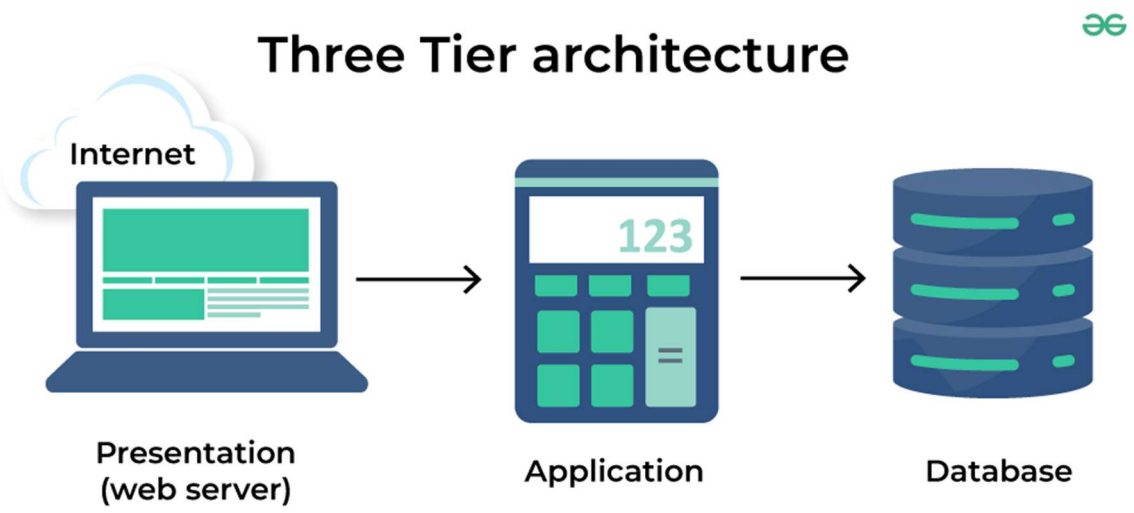
3.4.2. Aplikacijski sloj

Aplikacijski sloj (*Application Tier*) sadrži poslovnu logiku aplikacije. Ovaj sloj je implementiran u Javi koristeći *Spring Boot* radni okvir (*framework*). *Spring Boot* omogućuje jednostavnu konfiguraciju i integraciju s različitim servisima, uključujući *LangChain4j* biblioteku za integraciju s umjetnom inteligencijom i velikim jezičnim modelima. Aplikacijski sloj također osigurava povezivanje s vanjskim sustavima kao što su Azure i *Confluence*, omogućujući dohvaćanje i obradu podataka u realnom vremenu.

3.4.3. Sloj baze podataka

Sloj baze podataka (Database Tier) zadužen je za pohranu i upravljanje podacima. Za upravljanje vektorskim reprezentacijama dokumenata koristi se *Qdrant*, specijalizirana vektorska baza podataka koja omogućuje brzo i efikasno pretraživanje velikih količina vektorskih podataka. *Qdrant* je optimizirana za semantičko pretraživanje i analizu podataka, čime se osigurava brza i točna obrada upita korisnika. Također, koristi se *PostgreSQL* za tradicionalno upravljanje relacijskim podacima.

Na slici 3 prikazana je troslojna arhitektura koja jasno pokazuje kako su različiti dijelovi sustava odvojeni i međusobno povezani. Prezentacijski sloj komunicira s aplikacijskim slojem, koji zatim pristupa sloju baze podataka za pohranu i dohvaćanje informacija. Ova podjela omogućuje bolju organizaciju koda, lakše održavanje i skalabilnost sustava.



Slika 3: Three Tier Architecture (GeeksforGeeks, 2024)

4. Poslužiteljski dio aplikacije

Poslužiteljski dio aplikacije sastoji se od nekoliko ključnih komponenti koje zajedno omogućuju efikasno funkcioniranje korisničke podrške putem umjetne inteligencije. Ove komponente uključuju dockerizirane baze podataka (*PostgreSQL* i *Qdrant*), inicijalno ugrađivanje podataka (*data embedding*), te integraciju s *API*-jem *Confluence* i *OpenAI* modelom. Cilj je pružiti brze i točne odgovore na korisničke upite putem *chatbot*-a.

4.1. Tehnološki stog poslužiteljske strane

Za razvoj poslužiteljske strane koristi se kombinacija modernih tehnologija kako bi se postigla skalabilnost, sigurnost i učinkovitost:

- **Docker**: Za kreiranje izoliranih okruženja za *PostgreSQL* i *Qdrant* baze podataka.
- **PostgreSQL**: Relacijska baza podataka za pohranjivanje poruka i razgovora.
- **Qdrant**: Vektorska baza podataka optimizirana za brzo pretraživanje vektorskih reprezentacija dokumenata.
- **Confluence API**: Koristi se za dohvaćanje sadržaja s „*Call Center*“ prostora (*space*), uključujući upute, hodograme, rješenja za česte probleme i druge relevantne dokumente. Ovi sadržaji se koriste za obogaćivanje baze znanja koju sustav koristi za generiranje odgovora na korisničke upite.
- **OpenAI "text-embedding-3-large" model**: Za ugradnju tekstualnih podataka u vektorske reprezentacije i generiranje odgovora na korisničke upite. Vektori su predefenirane veličine reda 3072.
- **OpenAI GPT-4o**: *LLM* model za generiranje odgovora na korisničke upite.
- **Azure Storage**: Za pohranjivanje i dohvaćanje datoteka povezanih s korisničkim upitima.

4.2. Opis glavnih funkcionalnosti poslužiteljske strane

Poslužiteljski sustav omogućava nekoliko ključnih funkcionalnosti koje su temeljne za operacije *chatbot*-a:

- **Inicijalizacija baza podataka**: Pokretanje „*docker compose*“ akcije za inicijalizaciju *PostgreSQL* baze podataka te programsko pokretanje *Qdrant* docker kontejnera.
- **Ugradnja podataka**: Automatska ugradnja stranica s *Call Center* prostora pomoću *Confluence API*-a i *OpenAI* modela.
- **Ubacivanje podataka u Qdrant**: Pohranjivanje vektorskih polja koja su dobivena transformacijom teksta u vektore i metapodataka u *Qdrant* bazu podataka.
- **Generiranje odgovora na korisničke upite**: Korisnička poruka se ugrađuje, uspoređuje se s vektorima u *Qdrant* bazi pomoću *Cosine* udaljenost, te se koristi *LLM* model *GPT-4o* za procjenu i generiranje odgovora.
- **Dohvaćanje i prikaz datoteka**: Koristeći *Azure Storage*, dohvaćaju se hiperveze za datoteke koje se prikazuju na korisničkom sučelju.

5. Detaljni opis poslužiteljskih dijelova

5.1. Implementacija baze podataka

Prije pokretanja aplikacije potrebno je inicijalizirati kontejnerizirane *PostgreSQL* i *Qdrant* baze podataka. „*docker dompose*“ skripta automatski postavlja potrebne servise i osigurava njihovu dostupnost za aplikaciju. *PostgreSQL* se koristi za pohranu poruka i razgovora, dok se u *Qdrant* bazi pohranjuju vektorske reprezentacije dokumenata.

5.2. Integracija s LangChain4j

LangChain4j biblioteka omogućava jednostavnu integraciju s naprednim *AI* modelima unutar *Java* aplikacija. Korištenjem ove biblioteke, aplikacija može efikasno koristiti *LLM* modele za generiranje tekstualnih odgovora na korisničke upite. Integracija uključuje:

- **Tokenizacija** i **ugradnja podataka**: Koristi se model za tokenizaciju i ugradnju podataka.
- **Sučelje za LLM modele:** Olakšava konfigurabilnu integraciju s *OpenAI* modelima putem jednostavne biblioteke.
- **Definicija promptova**: Kreiranje specifičnih došaptavanja (promptova) za različite zadatke kako bi se osiguralo precizno generiranje odgovora.

5.3. Operativni tijek aplikacije

Aplikacija funkcionira na sljedeći način:

1. **Inicijalizacija sustava**:

Pokretanje „*docker compose*“ akcije koja inicijalizira kontejner *PostgreSQL* baze podataka. Pokretanje aplikacije koja također inicijalizira *Qdrant* vektorsku bazu podataka.

2. **Inicijalna ugradnja podataka**:

Sve Confluence stranice s „*Call Center*“ prostora (*Confluence space*) povlače se pomoću Confluence API-a. Stranice sadrže tekst, slike i njih se dobiva u HTML obliku. Datoteke se dobivaju posebnom krajnjom točkom (*endpoint*). Stranice se ugrađuju u vektorska polja koristeći OpenAI model "*text-embedding-3-large*". Ugrađivanje znači pretvaranje tekstualnih podataka u numeričke vektore koji omogućuju efikasno pretraživanje i analizu sadržaja. Koristi se rekurzivni „*document splitter*“ koji reže podatke na konfigurabilan broj tokena (jedan token je približno ekvivalentan 4 znaka teksta za uobičajeni engleski tekst, što je otprilike ¾ riječi, tako da je 100 tokena ≈ 75 riječi). Vektorska polja se unose u *Qdrant* bazu zajedno s metapodacima (naslov stranice, naslov roditeljske stranice, identifikator stranice, identifikator prostora, URL stranice).

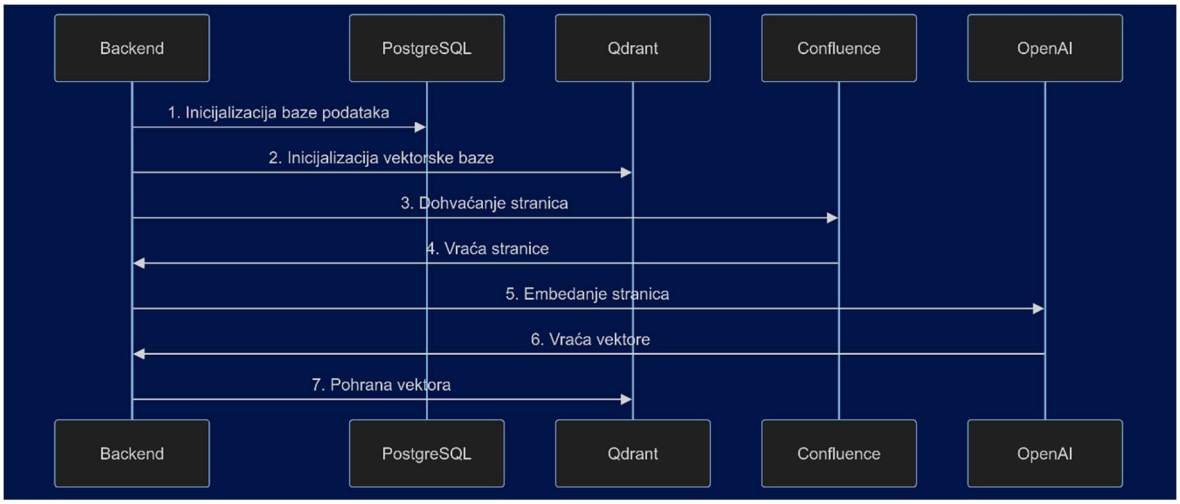
3. **Generiranje odgovora na korisničke upite**:

Korisnik postavlja pitanje putem korisničkog sučelja. Korisnikova poruka se ugrađuje u vektor pomoću *OpenAI* modela. Taj vektor se uspoređuje s vektorima u *Qdrant* bazi pomoću Cosine udaljenost kako bi se pronašli najbliži vektori te se u sustav vraćaju top 4 rezultata. *LLM* model *GPT-4o* procjenjuje koji od tih 4 sadržaja najviše odgovara korisnikovom pitanju i vraća identifikator stranice. *Confluence API* se ponovno poziva za dohvaćanje tražene stranice. *GPT-4o* model generira odgovor na korisnikovo pitanje, uključujući slike i link na (*Confluence*) stranicu za brzi pristup. Ukoliko odgovor nije relevantan za znan skup podataka kao na primjer pitanje „Koliko je sati u Londonu?“, odgovor *chatbot*-a će biti da nije treniran na takvim podacima te da se pita druga pitanje.

4. **Dohvaćanje datoteka**:

Nakon generiranja odgovora, aplikacija dohvaća hiperveze za sve datoteke povezane sa *Confluence* stranicom iz *Azure Storage*-a. Hiperveze za datoteke prikazuju se na korisničkom sučelju. Postoji krajnja točka (endpoint) za povlačenje datoteka s *Confluence* stranice i spremanje na *Azure Storage*, no ta akcija se ne radi pri pokretanju aplikacije zbog njezine vremenske kompleksnosti (6 – 8 minuta).

Na slici 3 prikazan je sekvencijski dijagram koji opisuje proces inicijalizacije sustava i inicijalnog ugrađivanja podataka. Proces počinje inicijalizacijom *PostgreSQL* i *Qdrant* baza podataka, nakon čega slijedi dohvaćanje stranica iz *Confluence*, ugrađivanje tih stranica pomoću *OpenAI* modela, te pohrana vektora u *Qdrant* bazu.

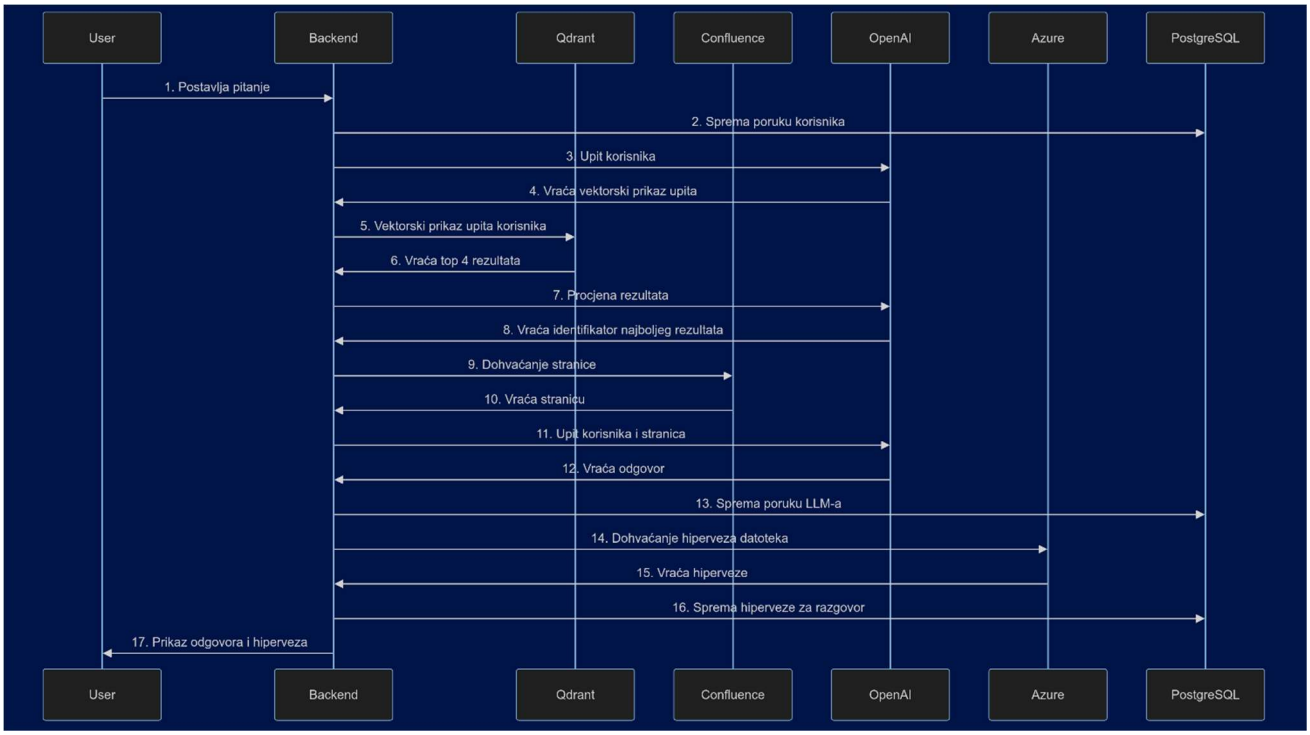


Slika 4: Sekvencijski dijagram za inicijalizaciju sustava i ugradnju podataka.

Na slici 4 prikazan je sekvencijski dijagram koji ilustrira tijek događaja kada korisnik postavi pitanje. Proces započinje ugrađivanjem korisnikovog pitanja pomoću *OpenAI* modela, nakon čega se upit (prikazan u vektorskom obliku) uspoređuje sa drugim vektorima u bazi podataka pomoću *Cosine* sličnosti . Najbolji rezultat se zatim dohvaća iz *Confluence* sustava i vraća korisniku zajedno s hipervezama za povezane datoteke.

Dijagram također prikazuje ponovno pozivanje *Confluence* sustava, unatoč tome što su sve stranice inicijalno spremljene tijekom procesa ugradnje (*embedding*). Razlog za ovo ponovno pozivanje je mogućnost promjena u sadržaju stranica. Iako ove promjene ne utječu na bazu podataka za pretragu, one su ključne za generiranje točnih odgovora. Na primjer, prilikom traženja brojeva određenih klijenata, pretraga će identificirati relevantan odlomak, ali konkretne informacije poput brojeva klijenata možda neće biti ažurirane u vektorskoj bazi. Stoga je dohvaćanje najnovije verzije stranice iz *Confluence* sustava važno za točnost odgovora. Ovaj postupak nije vremenski zahtjevan, što ga čini praktičnim izborom. S druge strane, proces ponovnog ugrađivanja cijelih stranica je vremenski intenzivniji, ali može se pokrenuti po potrebi putem dostupne krajnje točke (endpoint) za ponovno ugrađivanje.

Ovaj pristup osigurava detaljan i jasan opis slike koji objašnjava ne samo tijek događaja, već i razloge za određene dizajnerske odluke unutar sustava.



Slika 5: Sekvencijski dijagram za generiranje odgovora na korisničke upite.

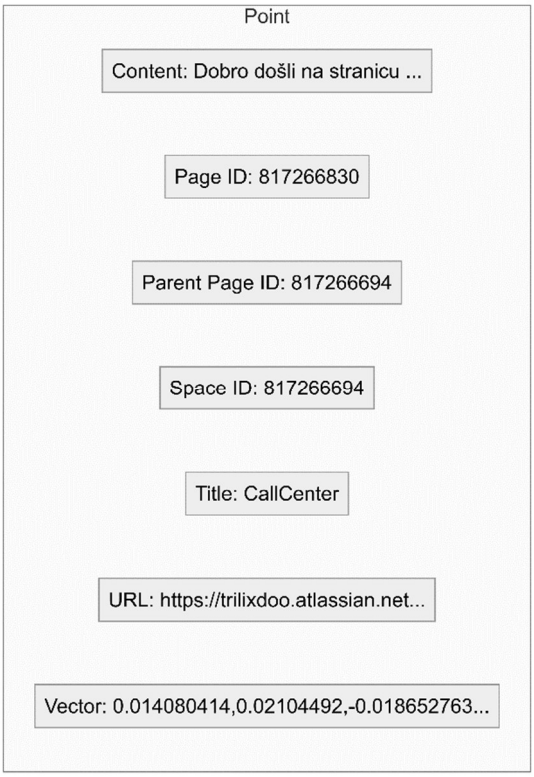
6. Baze podataka i njihova struktura

Poslužiteljski dio aplikacije koristi dvije glavne baze podataka: *Qdrant* za vektorske reprezentacije dokumenata i *PostgreSQL* za relacijske podatke. U nastavku su detaljno opisane sheme i funkcionalnosti obje baze podataka.

6.1. Qdrant baza podataka

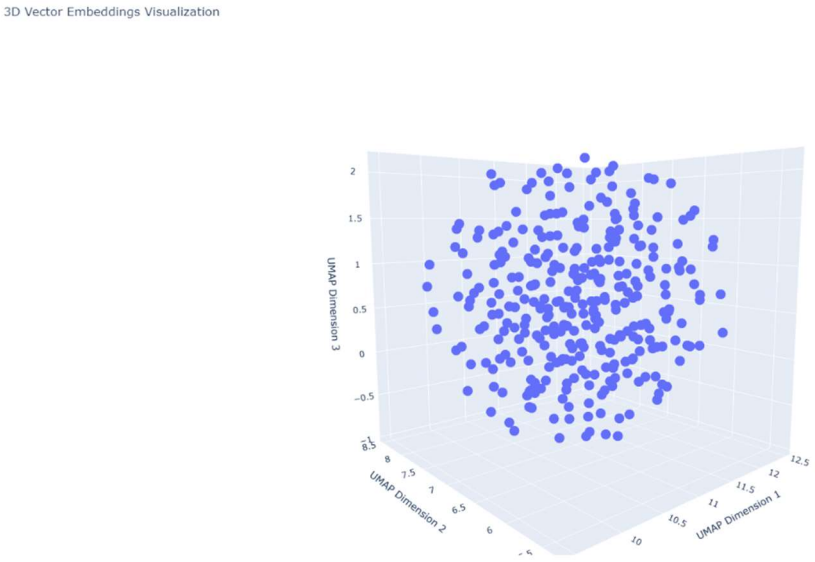
Qdrant je specijalizirana vektorska baza podataka dizajnirana za brzo i efikasno pretraživanje kroz velike količine vektorskih podataka. Ova baza podataka koristi se za pohranu ugrađenih tekstualnih podataka koji omogućuju semantičko pretraživanje i analizu. Na slici 6 prikazan je jedan zapis (*Point*) u *Qdrant* bazi podataka. Zapis je vektorski prikaz teksta sa prije spomenutim metapodacima, on sadrži sljedeća polja:

- `*content*`: Sadržaj stranice.
- `*pageId*`: ID stranice.
- `*parentPage*`: ID roditeljske stranice.
- `*spaceId*`: ID prostora.
- `*title*`: Naslov stranice.
- `*url*`: URL stranice.
- `*vector*`: Vektor koji je dobiven ugradnjom sadržaja teksta u dokumentu.



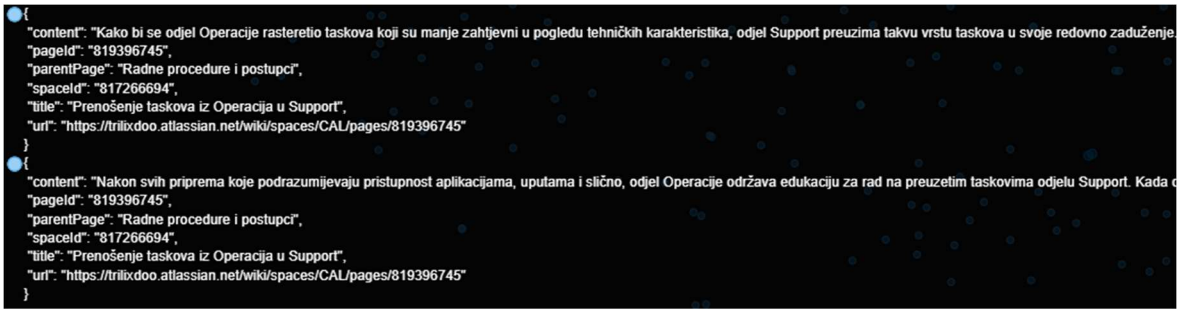
Slika 6: Struktura jednog zapisa (Point) u Qdrant bazi podataka.

Na slici 7 uz pomoć *Qdrant* baze podataka i napisane Python skripte vizualizirani su vektorski podaci u trodimenzionalnom prostoru. Svaka točka predstavlja jedan zapis, a njihova udaljenost ukazuje na semantičku sličnost.



Slika 7: Vizualizacija vektorskih podataka u Qdrant bazi podataka

Na slici 8 prikazane su dvije susjedne točke, odnosno vektori, što omogućuje detaljniji pregled njihovog sadržaja i semantičke sličnosti.

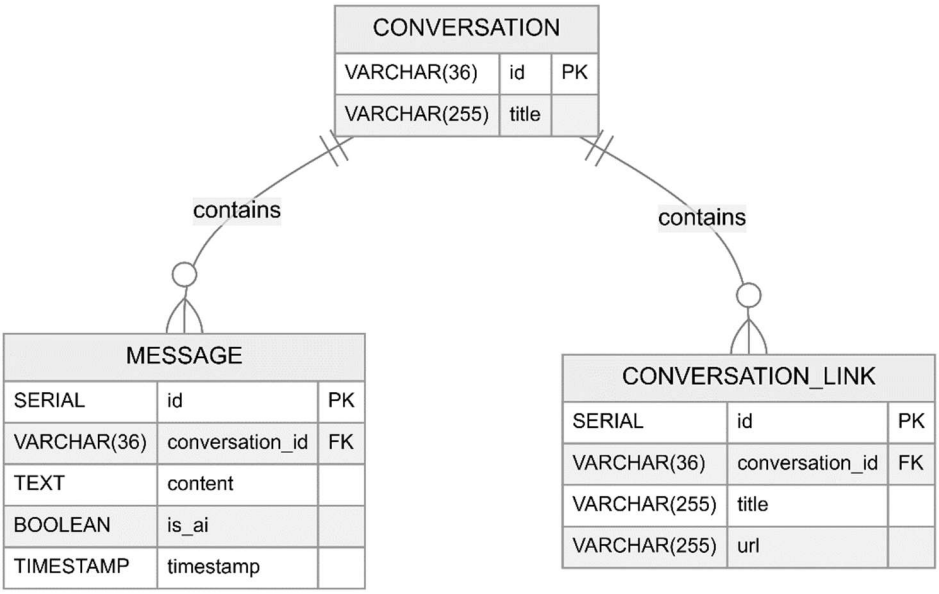


Slika 8: Detaljan prikaz dviju susjednih točaka u Qdrant bazi podataka.

6.2. PostgreSQL baza podataka

PostgreSQL je relacijska baza podataka koja se koristi za pohranu poruka i razgovora. Struktura baze podataka prikazana je na *ER* dijagramu (slika 9). Baza se sastoji od tri glavne tablice:

- `*conversation*`: Pohranjuje informacije o razgovorima.
- `*message*`: Pohranjuje poruke unutar razgovora.
- `*conversation_link*`: Pohranjuje poveznice na relevantne dokumente ili stranice povezane s razgovorima.



Slika 9: ER dijagram PostgreSQL baze podataka.

Svaka tablica sadrži sljedeća polja:

CONVERSATION:

- `*id*`: Primarni ključ, UUID (VARCHAR(36)).
- `*title*`: Naslov razgovora (VARCHAR(255)).

MESSAGE:

- `*id*`: Primarni ključ (SERIAL).
- `*conversation_id*`: Strani ključ povezan s `*conversation*` tablicom (VARCHAR(36)).
- `*content*`: Sadržaj poruke (TEXT).
- `*is_ai*`: Indikator je li poruka generirana od strane umjetne inteligencije ili je to poruka/upit korisnika (BOOLEAN).
- `*timestamp*`: Vremenska oznaka kada je poruka stvorena (TIMESTAMP).

CONVERSATION_LINK:

- `*id*`: Primarni ključ (SERIAL).
- `*conversation_id*`: Strani ključ povezan s `*conversation*` tablicom (VARCHAR(36)).
- `*title*`: Naslov poveznice (VARCHAR(255)).
- `*url*`: URL poveznice (VARCHAR(255)).

Ova struktura omogućuje učinkovito upravljanje porukama, razgovorima i povezanim resursima unutar sustava korisničke podrške.

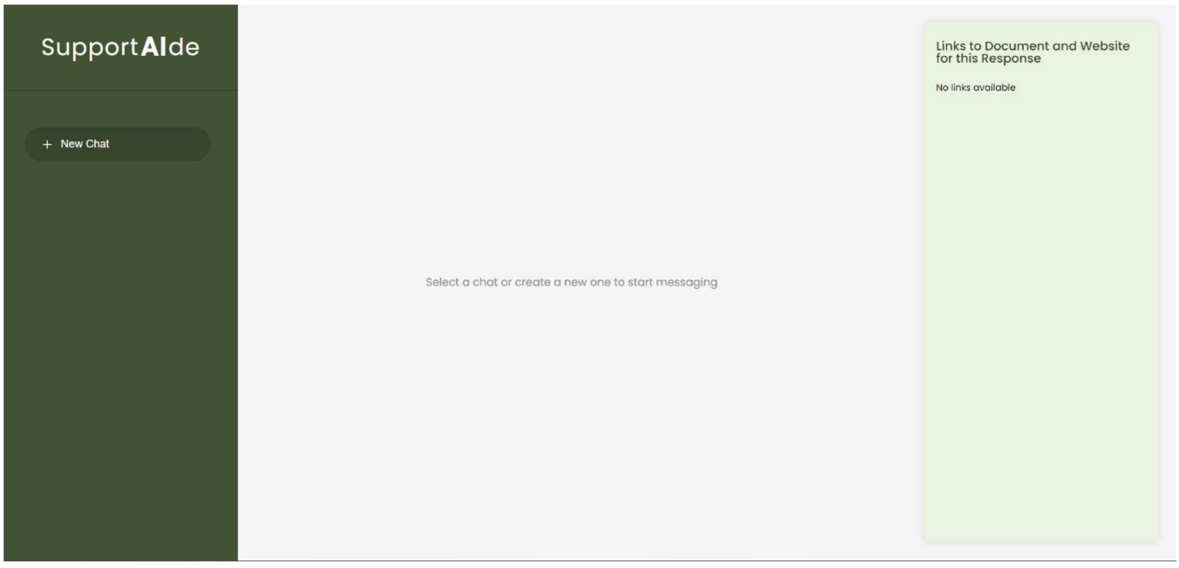
Integracija *Qdrant* i *PostgreSQL* baza podataka omogućuje učinkovito upravljanje i pretraživanje vektorskih i relacijskih podataka. *Qdrant* omogućuje brzo i efikasno semantičko pretraživanje kroz vektorske podatke, dok *PostgreSQL* pruža robusno rješenje za upravljanje relacijskim podacima.

7. Klijentski dio aplikacije

Klijentski dio aplikacije dizajniran je za omogućavanje korisnicima da postavljaju pitanja i dobivaju odgovore od umjetne inteligencije, uz integraciju relevantnih dokumenata i poveznica. U ovom dijelu su detaljno opisani primjeri interakcija korisnika s aplikacijom, prikazujući kako aplikacija obrađuje upite i vraća korisne odgovore.

7.1. Početno sučelje

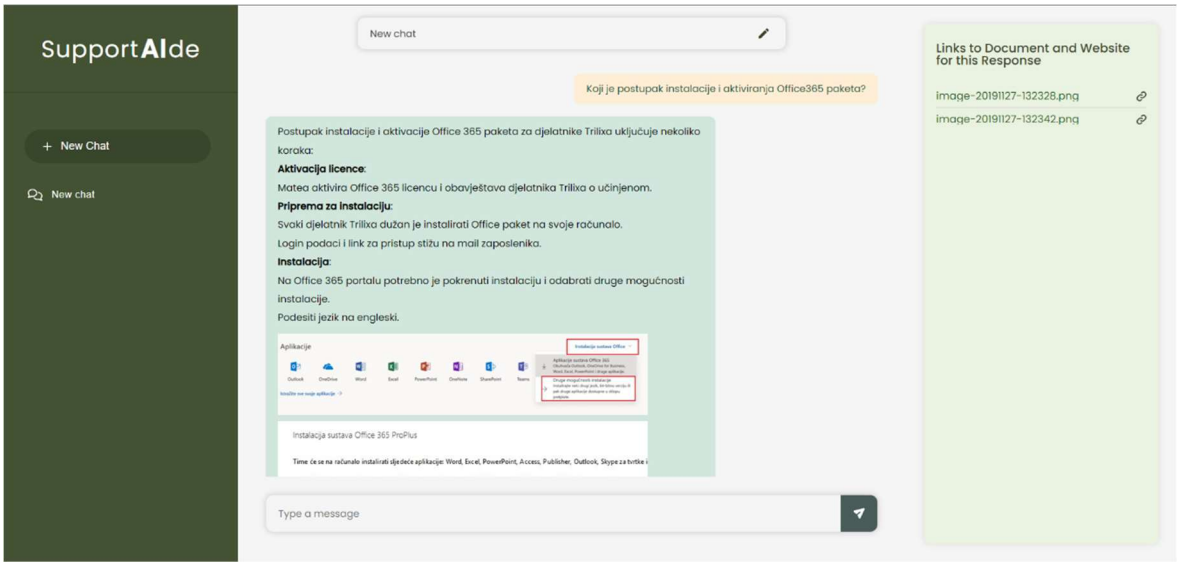
Na slici 10 prikazano je početno sučelje aplikacije. Korisnici imaju mogućnost započeti novi chat ili odabrati postojeći chat kako bi postavili pitanje. Desna strana sučelja rezervirana je za prikaz poveznica na relevantne dokumente za razgovor. Za svaki odgovor se dodatno vraća i *URL Confluence* stranice na kojoj se može pogledati od kuda su uzete dane informacije.



Slika 10: Početno sučelje aplikacije.

7.2. Primjer interakcije: Instalacija i aktivacija Office 365 paketa

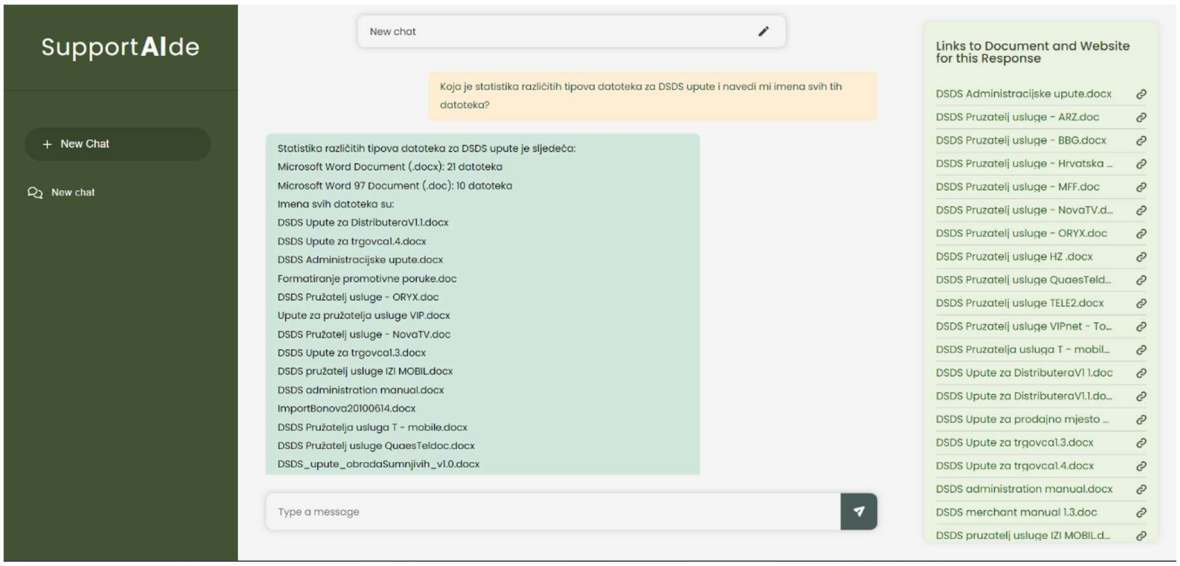
Na slici 11 prikazana je interakcija gdje korisnik postavlja pitanje: "Koji je postupak instalacije i aktiviranja *Office365* paketa?". Aplikacija vraća odgovor s detaljnim koracima za instalaciju i aktivaciju *Office365* paketa. Osim tekstualnog odgovora, aplikacija prikazuje slike relevantne za instalaciju i aktivaciju, te s desne strane pruža poveznice za preuzimanje datoteka, u ovom slučaju slika.



Slika 11: Interakcija s aplikacijom za instalaciju i aktivaciju Office 365 paketa.

7.3. Primjer interakcije: Statistika datoteka za DSDS upute

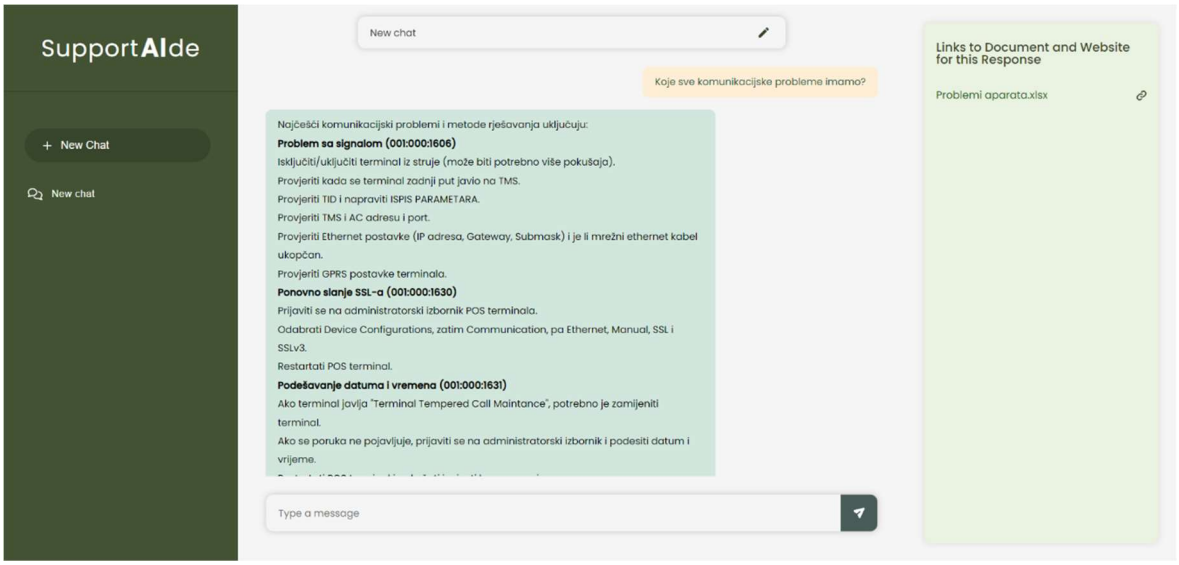
Na slici 12 korisnik pita: "Koja je statistika različitih tipova datoteka za DSDS upute i navedi mi imena svih tih datoteka?". Aplikacija odgovara s detaljnom statistikom tipova datoteka, uključujući broj datoteka za svaki tip i popis imena svih datoteka. Također za sve datoteke je dostupna poveznica za preuzimanje sa desne strane.



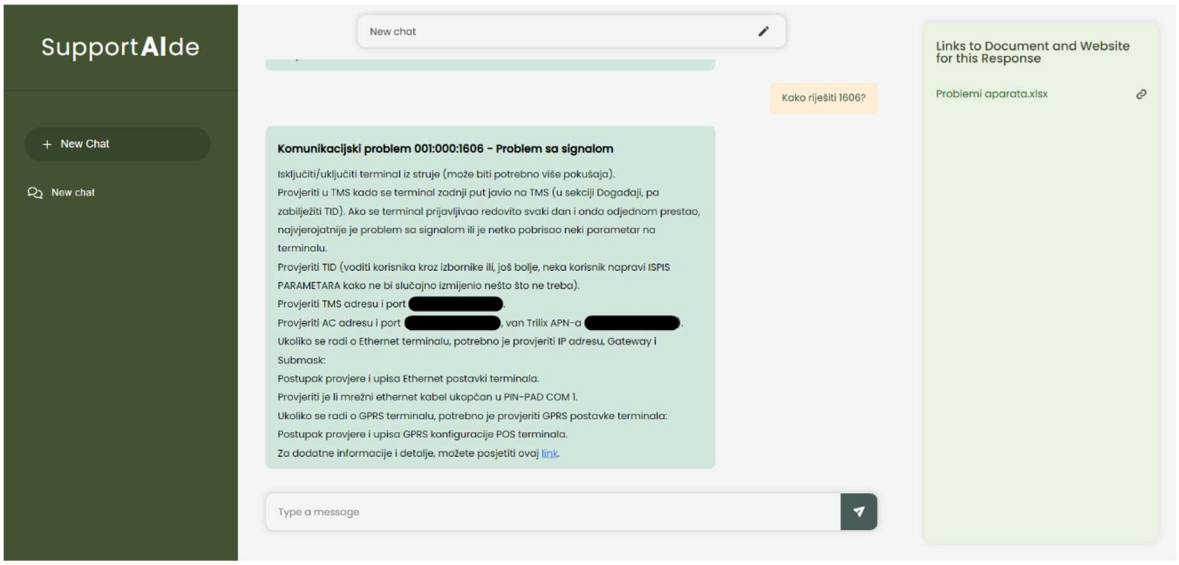
Slika 12: Interakcija s aplikacijom za statistiku datoteka za DSDS upute.

7.4. Primjer interakcije: Komunikacijski problemi

Na slici 13 i slici 14 prikazana je nadovezana interakcija gdje korisnik prvo pita: "Koje sve komunikacijske probleme imamo?" a zatim: "Kako riješiti 1606?". Aplikacija vraća komunikacijske probleme zatim odgovor s koracima za rješavanje problema sa signalom (kod 1606), pružajući poveznicu na relevantni povezani dokument „*Problemi aparata.xlsx*“. Tu se može vidjeti i održavanja konteksta u samom razgovoru. Drugo pitanje ne sadrži „komunikacijski problem“ u sebi, no zbog konteksta razgovora *LLM* model zaključi pozadinu pitanja. Također pri kraju odgovora u slici 14 se može vidjeti podcrtani „link“ za *Confluence* stranicu.



Slika 13: Interakcija s aplikacijom za komunikacijske probleme.



Slika 14: Rješavanje problema sa signalom (kod 1606).

Klijentski dio aplikacije pruža intuitivno i responzivno korisničko sučelje koje omogućava korisnicima da jednostavno postavljaju pitanja i dobivaju precizne odgovore. Integracija slika, poveznica i dodatnih dokumenata unutar odgovora dodatno obogaćuje korisničko iskustvo, čineći aplikaciju korisnom alatkom za podršku korisnicima.

8. Zaključak

Razvoj naprednog sustava korisničke podrške temeljenog na tehnologijama umjetne inteligencije predstavlja značajan korak naprijed u optimizaciji internih procesa i poboljšanju korisničkog iskustva. Korištenjem velikih jezičnih modela i metode *Retrieval-Augmented Generation* (*RAG*), sustav omogućuje efikasno pretraživanje i korištenje interne baze znanja, što dovodi do bržeg i preciznijeg rješavanja korisničkih upita.

Implementacija tehnologija kao što su *React* na klijentskoj strani i *Spring Boot* na poslužiteljskoj strani, uz uporabu *PostgreSQL* i *Qdrant* baza podataka, osigurava skalabilnost i robusnost sustava. Sustav je dizajniran tako da podržava dinamičko dohvaćanje informacija i njihovu primjenu u realnom vremenu, čime se smanjuje vrijeme potrebno za obuku novih agenata i povećava ukupna učinkovitost korisničke podrške.

Tijekom razvoja sustava identificirani su ključni izazovi i prednosti integracije umjetne inteligencije u procese podrške. Evaluacija sustava pokazala je da primjena *RAG* metode značajno smanjuje rizik od generiranja netočnih informacija (halucinacija) i povećava točnost odgovora. Osim toga, sustav omogućuje kontinuirano ažuriranje i obogaćivanje baze znanja, što dodatno doprinosi njegovoj relevantnosti i korisnosti.

Zaključno, implementacija ovog sustava predstavlja značajan napredak u području korisničke podrške. Njegova primjena u različitim domenama može značajno unaprijediti način na koji tvrtke komuniciraju s korisnicima i upravljaju internim znanjem, čime se doprinosi njihovoj konkurentnosti i uspjehu na tržištu.

Literatura

[1]. **Radeva, I., Popchev, I., Doukovska, L., & Dimitrova, M**. *Web Application for Retrieval-Augmented Generation: Implementation and Testing*. Electronics. Vol. 13, No. 1 (2024). Pristupljeno 10.5.2024. Dostupno na: https://ojs.aaai.org/index.php/AAAI/article/view/29728/31250

[2]. **Huang, W., Lapata, M., & Vougiouklis, P.** *Retrieval Augmented Generation with Rich Answer Encoding*. Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics. Kyoto, Japan (2023). Pristupljeno 11.5.2024. Dostupno na: https://aclanthology.org/2023.ijcnlp-main.65.pdf

[3]. **Lakatos, R., Pollner, P., Hajdu, A., & Joo, T.** *Investigating the performance of Retrieval-Augmented Generation and fine-tuning for the development of AI-driven knowledge-based systems*. arXiv preprint (2024). Pristupljeno 13.5.2024. Dostupno na: https://arxiv.org/pdf/2403.09727

[4]. **Matuski, M., Lago, P. & Sozo, I.** *Characterizing Word Embeddings for Zero-Shot Sensor-Based Human Activity Recognition.* ResearchGate (2019). Pristupljeno 9.4.2024. iz: https://www.researchgate.net/publication/337372499_Characterizing_Word_Embeddings_for_Zero-Shot_Sensor-Based_Human_Activity_Recognition

[5]. **Chotrani, A.** *Visualizing your own word embeddings using TensorFlow*. Medium. Pristupljeno 9. 4. 2024. iz https://medium.com/@aakashchotrani/visualizing-your-own-word-embeddings-using-tensorflow-688b3a7750ee

[6]. **GeeksforGeeks.** (2024). *Three Tier Client-Server Architecture in Distributed System.* Pristupljeno 9. 6. 2024. iz https://www.geeksforgeeks.org/three-tier-client-server-architecture-in-distributed-system/

Sažetak

Razvoj Internog Sustava za Podršku Korisničkoj Službi Koristeći Tehnologije Umjetne Inteligencije

U ovom radu istražuje se razvoj naprednog internog sustava za podršku korisničkoj službi, temeljenog na tehnologijama umjetne inteligencije. Sustav koristi velike jezične modele i princip Retrieval Augmented Generation (*RAG*) za poboljšanje pristupa i upravljanja internom bazom znanja, što omogućava brže i efikasnije rješavanje upita korisnika. Integracijom *React*-a na klijentskoj strani i Spring Boot-a na poslužiteljskoj strani, uz upotrebu *PostgreSQL* i *Qdrant* baza podataka, razvijen je robustan i skalabilan sustav. Rad detaljno opisuje tehničku implementaciju sustava, uključujući arhitekturu, korištene tehnologije i alate, te način integracije umjetne inteligencije u procese korisničke podrške. Cilj rada je unaprijediti interne procese podrške, povećati produktivnost agenata i poboljšati kvalitetu usluge krajnjim korisnicima.

Ključne riječi

- Umjetna inteligencija
- Veliki jezični modeli
- Retrieval Augmented Generation (RAG)
- Interna baza znanja
- Korisnička podrška
- Vektorska baza

Summary

Development of an Internal Support System for Customer Service Using Artificial Intelligence Technologies

This paper explores the development of an advanced internal support system for customer service, based on artificial intelligence technologies. Utilizing large language models and the Retrieval Augmented Generation (RAG) principle, the system enhances access to and management of an internal knowledge base, enabling faster and more efficient resolution of user queries. Integrating React on the client side and Spring Boot on the server side, along with the use of PostgreSQL and Qdrant databases, a robust and scalable system has been developed. The paper provides a detailed description of the system's technical implementation, including its architecture, utilized technologies and tools, and the integration of artificial intelligence into customer support processes. The aim of the study is to improve internal support processes, increase the productivity of agents, and enhance the quality of service for end users.

Keywords

- Artificial Intelligence
- Large Language Models
- Retrieval Augmented Generation (RAG)
- Internal Knowledge Base
- Customer Support
- Vector database