Primjena tehnologija razgovornih robota u modernim programskim proizvodima

Viljevac, David

Master's thesis / Diplomski rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:355956

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

David Viljevac

PRIMJENA TEHNOLOGIJA RAZGOVORNIH ROBOTA U MODERNIM PROGRAMSKIM PROIZVODIMA

DIPLOMSKI RAD

Varaždin, 2024.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

David Viljevac

JMBAG: 0016143211

Studij: Informacijsko i programsko inženjerstvo

PRIMJENA TEHNOLOGIJA RAZGOVORNIH ROBOTA U MODERNIM PROGRAMSKIM PROIZVODIMA

DIPLOMSKI RAD

Mentor:

Izv. prof. dr. sc. Zlatko Stapić

Varaždin, srpanj 2024.

David Viljevac

Izjava o izvornosti

Izjavljujem da je moj 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

U suvremenom svijetu razvoja programskih proizvoda, razgovorni roboti (engl. Chatbot) koriste se kao interaktivni alati koji omogućuju korisnicima savjetovanje i prikupljanje informacija o nekom sustavu. Cilj ovog diplomskog rada je pružiti sveobuhvatan pregled tehnologija za razvoj razgovornih robota, predstaviti prednosti i nedostatke te razviti prototip programskog proizvoda koji koristi funkcionalnosti razgovornog robota. U uvodu je definirana motivacija analize tehnoloških opcija u području razgovornih robota. Definiraju se ključni pojmovi te se uvodi u rad kroz kratak pregled alata koji se koriste za razvoj razgovornih robota. Nadalje, opisani su kriteriji za usporedbu tehnologija, kao što su jednostavnost korištenja, performanse, funkcionalnosti, podrška za svaku tehnologiju i troškovi. Ti kriteriji su potkrijepljeni i dijelovima programskog koda koji se odnose na njihovu primjenu. U praktičnom dijelu rada razvio se prototip programskog proizvoda koji implementira funkcionalnost programskog proizvoda. Dva razvijena programska proizvoda su mobilna aplikacija i internet aplikacija sa informacijama stranice tvrtke Solvis. Na kraju rada je sadržana dokumentacija implementacije razgovornog robota u praktičnom primjeru, uključujući opis funkcionalnosti, upute za korištenje alata, primjere koda itd. Informacije ili novo stečeno znanje koje se može dobiti kroz ovaj rad temelji se na smjernicama za odabir najprikladnije tehnologije te praktične primjere implementacije razgovornih robota.

Ključne riječi: razgovorni roboti; programski proizvod; usporedba; analiza; implementacija; savjetovanje; prototip; kriteriji; tehnologije;

Sadržaj

1. Uvod

Za današnje moderne programske proizvode bitno je imati velik broj dobro razvijenih kompetitivnih značajki kako bi se izdvojili na tržištu, privukli veći broj korisnika i obezbjedili dugoročnu održivost i uspjeh proizvoda. Jedna od takvih značajki je i razgovorni robot koji bi pomagao i savjetovao korisnike o programskom proizvodu ili sadržaju programskog proizvoda.

Ovaj diplomski rad ima cilj istražiti razvoj razgovornih robota za moderne proizvode kroz dva ključna aspekta: teorijski i praktični. U teorijskom dijelu rada analiziraju se osnovni koncepti NLP-a i NLU-a, kao i različite kategorije razgovornih robota. Pored toga, rad pruža komparativnu analizu nekoliko popularnih razvojnih okvira ili tehnologija za razvoj razgovornih robota, ističući njihove prednosti i mane.

Praktični dio rada fokusira se na razvoj prototipa razgovornog robota korištenjem Dialogflow ES tehnologije, u kombinaciji s Node.js za servisni dio (engl. backend) i Vue.js za vizualni dio (engl. frontend) aplikacije razgovornog robota te Android Studia inačice „Koala“ kao dva odvojena primjera modernog programskog proizvoda. Ovaj dio rada obuhvaća opis implementacije, uključujući razvoj i testiranje razgovornog robota.

Cilj ovog istraživanja je da pruži sveobuhvatan pregled trenutnog stanja tehnologije razgovornih robota i da demonstrira praktičnu primjenu kroz izradu funkcionalnog prototipa. Očekuje se da rezultati rada doprinesu boljem razumijevanju i unapređenju razvoja razgovornih robota u kontekstu modernih proizvoda.

2. Metode i tehnike rada

Metode i tehnike rada pomoću kojih će se ovaj rad napisat će biti usko povezane s tehnologijama koje su izabrane. Razlikujemo metode unutar teorijskog i praktičnog dijela rada. U sljedeća dva paragrafa će biti kratko opisano svaka tehnika koja je korištena.

Za početak provest će se opsežno istraživanje dostupne literature o procesiranju, razumijevanju i generiranju izvornog govora. Cilj ovog koraka je dobiti sveobuhvatan pregled razlika između procesiranja, razumijevanja i generiranja izvornog koraka kako bi se moglo bolje razumijevati što znače ti pojmovi. Nadalje klasifikacijom će se raspodijeliti razgovornog robote u nekoliko kategorija kako bi dobili uvid zašto su nam gore navedeni pojmovi bitni i kako ih pojedina kategorija definira. Sljedeći korak u pisanju ovoga rada bit će selekcija tehnologija na temelju istraživanja. Svaku od odabranih tehnologija će se posebno objasniti temeljem dostupne dokumentacije o samoj tehnologiji. Bližeći se kraju teorijskog dijela rada, usporedit će se i autor će objektivno ocijeniti svaku od tehnologija temeljem odabranih kriterija. Kriteriji koji će biti odabrani su najbitniji kriteriji koje sagledava pojedinac ili tvrtka prilikom korištenja neke od tehnologija. Ocjenjivanjem će se također dati usporedba i evaluirat će se konačna ocjena svake od tehnologija.

U praktičnom dijelu rada će se prototipiranjem pokušati napraviti početna verzija razgovornog robota u odabranoj tehnologiji. Nakon izrađenog prototipa razvojem programa izradit će se sučelja kroz koja možemo razgovarati s robotom. Sučelja će biti izrađena u obliku internet i mobilnih aplikacija. Internet aplikacija će se izraditi u Javascript programskom jeziku kao i njezin servisni dio dok će mobilna aplikacija koristiti Kotlin programski jezik te isti servisni dio kao i internet aplikacija. Izrađena sučelja i robot će se istestirati te će se evaluirati rezultat spajanja robota i sučelja u jedno. Kod programa će biti verzioniran pomoću Github[1] tehnologije dok će se sam program razvijati unutar programa Visual Studio Code[2] i Android Studio[3]-a.

Od generativnih alata umjetne inteligencije korišten je ChatGPT te njegov proizvod Write For Me. Korišten je uglavnom za parafraziranje i bolje strukturiranje rečenica kao i za inicijalni format pisanja fusnota. U kodu je korišten samo kao inicijalni pomagač prilikom

[1] Github je platforma za razvoj softvera i upravljanje verzijama koda sastavljena od repozitorija koda pomoću kojih više različitih korisnika mogu kolaborirati. Dostupno na: https://github.com/
[2] Visual Studio Code je besplatan, open-source uređivač koda razvijen od strane Microsofta. Podržava različite programske jezike te nudi značajke pronalaženja pogreške (engl. debug), kontrolu verzija i ugrađeni terminal. Dostupno na https://code.visualstudio.com/
[3] Android Studio je integrirano razvojno okruženje (IDE) koje se koristi za razvoj Android aplikacija. Razvija ga Google, a bazira se na IntelliJ IDEA. Dostupno na: https://developer.android.com/studio

generiranja klasa stilova i datoteka koje oblikuju sam prikaz. Nakon generiranja inicijalnog izgleda dalje je autor sam razvio i oblikovao proizvod prema svojim željama. Sveukupno gledano ChatGPT je korišten samo kao pomagač pri početku razvijanja proizvoda za razjašnjavanje nejasnih elemenata te oblikovatelj rečenica dok je sav sadržaj pronađen na referenciranim izvorima i/ili od prije poznat autoru rada.

3. Pregled tehnologija razgovornih robota

Razgovorni roboti su postali svakodnevica čovjeka pri korištenju internetskih usluga. Za njihov razvoj u ovom trenutku više nije potrebno veliko znanje o programiranju, arhitekturi informacijskih sustava, strukturama ili algoritmima. U ovom poglavlju objasnit će se algoritmi, procesi i tehnike koje se u modernom svijetu koriste za razvoj razgovornih robota. Za algoritme procesiranja i razumijevanja izvornog govora će biti riječ u sljedećim podpoglavljima. U svakom od tih poglavlja opisati će se što točno i na koji način pojedini algoritam procesira potrebne ulaze te kako generira izlaz.

3.1. Procesiranje izvornog govora

Procesiranje izvornog govora (engl. Natural Language Processing, u daljnjem tekstu NLP) i razumijevanje izvornog/prirodnog govora (engl. Natural Language Understanding, u daljnjem tekstu NLU) su jako bliski no u radu će biti opisani odvojeno kako bi prikazali nekoliko različitosti između njih.

NLP je sposobnost računalnog programa da procesira ulazne podatke ljudskog govora i pisma tj. u referenci prirodnog jezika. NLP postoji već više od 50 godina, a korijeni i temelji su postavljeni u lingvistici. Može se koristiti na temelju postavljenih pravila (engl. rule-based) ili strojnog učenja (engl. machine learning) kako bi razumio smisao teksta koji je ulazni parametar pripremljen od osobe. Osim što će se kroz rad vidjeti da nosi veliku ulogu u razvoju razgovornih robota, može imati ulogu i u glasovnim asistentima, programima za skeniranje teksta, aplikacijama za prijevod i svim ostalim sličnim programima koji su vezani uz obradu teksta [1].

Postoje dvije glavne faze u procesiranju prirodnog govora. To su pretprocesiranje podataka i razvoj algoritma. Pretprocesiranje podataka uključuje pripremu i čišćenje tekstualnih podataka za daljnju analizu unutar strojeva. Pretprocesiranje ustvari pretvara ulazne podatke osoba u prikladne ulazne podatke koje algoritam može koristiti. Također definira specifičnosti unutar teksta kako bi algoritam mogao stvoriti smisao. Ovaj korak se može provesti na nekoliko načina [1]:

- Tokenizacija – zamjenjuje osjetljive informacije s neosjetljivim informacijama ili tokenom. Često se koristi u financijskim transakcijama kako bi se zaštitile informacije kartice [1].

- Uklanjanje stop riječi – u tekstu ostaju jedinstvene riječi dok se one česte uklanjaju [1]
- Lematizacija i osnove – lematizacija grupira različite oblike jedne riječi npr. riječ „hodanje“ svedena bi bila na osnovni oblik „hodati“ [1]
- Označavanje dijelova govora – riječi su označene kao imenice, glagoli ili pridjevi [1]
- Sentimentalna analiza – analizira i potražuje riječi koje ukazuju na npr. nešto pozitivno, neutralno ili negativno te ih svrstava u grupe [2]. Time se može postići analiza komentara proizvoda te dinamički kreirati pokazatelje zadovoljstva ljudi s proizvodom.
- Prepoznavanje imenovanih entiteta (engl. Named Entity Recognition, u daljnjem tekstu NER) – unutar ulaznog teksta označuje „imenovane entitete“ te ih sprema za kasniju analizu. Za primjer uzet ćemo rečenicu: „Direktor Burai Software d.o.o organizacije je John Slavic, rođen u Kanadi“. U tom primjeru NER bi odvojio Burai Software d.o.o kao organizaciju, John Slavic kao osobu te Kanadi kao lokaciji [2].

Nakon pretprocesiranja podataka slijedi korak razvoja algoritma koji će iskoristiti te podatke i proizvesti nove izlazne podatke razumljive osobi. Kroz te korake istječe se važnost sposobnosti i prednosti NLP-a u analizi podataka unesenih unutar baze podataka velikih organizacija. Ljudima je teško procesirati velik skup podataka neke organizacije jer je za to potrebno vrijeme, a i što su veći podatci to je manje isplativo jer je potrebno više ljudi da obradi te podatke. U tom trenutku je NLP izvrstan izbor koji taj problem rješava puno efikasnije i točnije [1].

NLP se može smatrati kao komponenta generativne umjetne inteligencije (engl. Artificial Inteligence, u daljnjem tekstu AI) ili razgovornih robota koji se temelje na AI-u. Uloga koju ima kao komponenta je ta da se olakša komunikacija između robota i osobe. To se postiže time da NLP analizira ulaz podataka ili poruku osobe te ju procesira u čitljivi format za AI koji dalje obrađuje i ustvari stvara odgovor [1].

NLP može imati mnogo različitih funkcija unutar programa ili zasebno. Neke od tih funkcija su [1]:

- Klasifikacija i ekstrakcija teksta
- Prijevod teksta u razumljiv tekst za stroj
- Generiranje prirodnog jezika (engl. Natural Language Generation, u daljnjem tekstu NLG) – najbolji primjer za to je ChatGPT koji kao ulaz dobiva prirodnih govor, procesira ga, te izlaz pretvara u jezik koji je razumljiv osobi koje je zadala ulaz.
- Analiza povratnih informacija korisnika

- Automatizacije korisničke podrške
- Automatski prijevod
- Analiza i kategorizacija zapisa
- Otkrivanje plagijata

Prema tome može se zaključiti kako NLP ima nekoliko važnih slojeva:

- Aplikacijski
- Skladište podataka – mogućnost spremanja velikog broja statičnih podataka
- NLP procesnu jedinicu [3]

Uz sve mogućnosti koje daje NLP svakako ima i neke potencijalne probleme. Postoje problemi poput preciznosti. Prilikom pretprocesiranja ulaza, računalo zahtjeva da ulaz bude precizan, nedvosmislen i strukturiran, no jezik koji neka osoba govori to ne mora biti. Jezik neke osobe može sadržavati različita narječja, izreke, skraćenice čemu je teško prepoznati smisao s računalne strane. Također način izgovora nekih rečenica predstavlja problem. Neke rečenice se mogu izgovorit sarkastično ili stvarno misleći. Takve rečenice će se pisati isto al će zvučati drugačije [1].

3.2. Razumijevanje izvornog govora

Razumijevanje izvornog govora (engl. Natural Language Understanding, u daljnjem tekstu NLU) je područje računalne znanosti koja pokušava procesira ulazne podatke osoba i njihovih jezika tako da ih u potpunosti razumije, a ne da jednostavno razumije pojedine riječi. NLU pokušava razumjeti implikacije prirodnog jezika koji se koristi između komunikacije osobe i na primjer razgovornog robota koji se temelji na NLU. U te implikacije spadaju emocije, trud, namjere, ciljevi, sarkastičnost itd. To sve se prepoznaje koristeći velike knjižnice informacija i podataka o nekom jeziku [4].

Na početku NLU procesira ulazne podatke osobe slično kao i NLP. Na primjeru „karte New York do Miami 25. travanj 20:00h“, NLU analiza daje rezultate poput „karte(namjera kupovanja karti) New York(lokacija) do Miami(lokacija) 25. travanj(datum) 20:00h(vrijeme)“. Dva ključna koncepta koje koristi NLU za analizu su prepoznavanje namjera (engl. intent recognition) i prepoznavanje entiteta [4].

Prepoznavanje namjera identificira što osoba koja piše namjerava učiniti. Identifikacija namjera ili cilja tih namjera pomaže programu da razumije i klasificira cilj interakcije [5]. U

primjeru gore program će razumjeti da osoba želi kupiti karte za avion. Nakon analize i daljnjeg procesiranja koristeći NLU pretraživač bi vratio web stranice na kojem osoba može kupiti karte.

Prepoznavanje entiteta je isto kao prije imenovani NER. Prepoznaje entitete poput lokacija u gornjem primjeru su to New York i Miami te datume, količine, postotke, imena osoba, imena organizacija itd.

Zašto je NLU važan? Djelomičan odgovor na to pitanje je dano kroz objašnjenje da prirodni jezik nije ono što je korektan ulaz podataka za neki algoritam. Svakako, ako bi se i implementirala ideja transformacije svakog jezika i narječja to bi bilo jako zahtjevno gotovo i nemoguće zbog razvijanja ljudskog govora. Zato postoji NLU koji može opskrbiti različite tipove tehnologija sa sličnom razinom razumijevanja ljudi i njihovog govora. Prema tome NLU se može koristiti za:

- Odgovore na interne i eksterne e-pošte
- Komentare na društvenim mrežama
- Razgovore s razgovornim robotima
- Upite za pretraživanje na internetu
- Interaktivne odgovore glasovnih sustava za pozive
- Glasovne asistente

Time NLU ima sposobnost za klasifikaciju velikog volumena nestrukturiranog teksta u klase bez da prolazi kroz taj tekst jedan po jedan ulaz. To omogućuje izvršavanje zadataka poput analize sadržaja, tematsko modeliranje, strojno prevođenje i odgovaranja na pitanja u količinama koje bi bile nemoguće postići ljudskim naporom [4].

Kako bi NLU došao do rezultata mora biti istreniran nad velikom količinom podataka. To se može postići kroz [6]:

- Sakupljanje i označavanje obučnih podataka – prikupljanje i anotiranje raznolikog skupa podataka. Trebao bi sadržavati širok spektar tipova upita.
- Analiza pogrešaka – analiziranje pogrešaka sustava kako bi se identificirali obrasci i poboljšala točnost za obradu prirodnog jezika
- Povratne petlje – implementiranje mehanizma povratnih informacija kako bi se identificirali pozitivni i negativni rezultati.
- Kontinuirano poboljšanje – redovito ažuriranje i ponovno treniranje modela s novim podacima.

Nakon treniranja NLU bi trebao biti u mogućnosti podržavati sljedeće [4]:

- Razumijevanje mnogobrojnih jezika
- Dubinsku analizu
- Brzo tumačenje i odgovor
- Jednostavnost upotrebe i mogućnosti integracije
- Automatizirane akcije

NLU predstavlja napredak u načinu na koji strojevi komuniciraju s ljudskim jezikom, prelazeći granice osnovnog prepoznavanja i ulazeći u razumijevanju konteksta i namjera. Kako tehnologije i jezik dalje napreduju, integracija NLU-a obećava povećanje učinkovitosti i djelotvornosti automatiziranih sustava. Uz kontinuirana poboljšanja i treniranja, NLU sustavi će postajati sve sposobniji u suočavanju s kompleksnostima ljudskog jezika.

4. Klasifikacija razgovornih robota

Klasifikacija razgovornih robota predstavlja ključan korak u razumijevanju raznolikosti i funkcionalnosti istih. Razgovorni roboti su klasificirani prema njihovoj interakciji, sakupljanju informacija i ciljevima upotrebe. Prema toj raspodjeli može se govoriti o razgovornim robotima temeljeni na tekstu i glasu za interakciju, orijentirani i neorijentirani zadatkom za ciljeve upotrebe, roboti otvorene ili zatvorene domene za specifičnost tema razgovora itd. [3]. Neki od njih će se objasnit kroz sljedeća potpoglavlja.

4.1. Roboti orijentirani na zadatke

Roboti orijentirani na zadatke (engl. task oriented bots) imaju za cilj pomoći korisniku ispuniti zadatak. Takvi roboti temelji su na jednoj domeni te samo za nju mogu pomoći korisniku u njegovom zadatku. Primjer takvih domena su hoteli (rezervacije, lokacije...), rezerviranje karata, kreiranje rasporeda ili traženje specifične informacije u nekoj domeni [3]. Ova kategorija razgovornih robota temelji se na ručno napravljenim pravilima kako bi se usmjerilo korisnika u pravom smjeru. Također služe kao alat za smanjenje vremena obavljanja ponavljajućih zadataka.

Svakako razgovorni roboti orijentirani na zadatke imaju nedostatke. Najveći od njih je jaz u komunikaciji korisnika i neprikladno razvijenog robota. Istraživanje s National Yang Ming Chiao Tung fakulteta i fakulteta u Torontu dokazalo je da je teoretski moguće integrirati preporuke za dizajn u proces razvoja i stvaranja razgovornog robota. Time bi se postiglo smanjenje jaza u komunikaciji te bi korisnici bili zadovoljniji uslugom razgovornog robota kojeg koriste [7].

4.2. Roboti neorijentirani na zadatke

Razgovorni roboti neorijentirani na zadatke mogu podržati razgovor širokih tema s korisnikom. Takav način razgovora pridaje korisniku osjećaj razgovora s drugom osobom, a ne s umjetno izrađenim razgovornim robotom. Odgovori robota mogu biti razdvojeni u dvije kategorije. Kategoriju generativnih modela (generiraju ispravnije odgovore tijekom razgovora) i modela temeljenih na pretraživanju (uče odabrati informativnije odgovore iz repozitorija razgovora). Ovakvi modeli ne mogu dati odgovor ako nije predefiniran obrazac za to pitanje jer ih oni ne stvaraju već koriste već definirane odgovore [3].

Ovi roboti igraju važnu ulogu u ljudskom društvu jer su važan alat za kreiranje i jačanje veze između korisnika i umjetno izrađenog robota. Očekivanja su da roboti kao takvi imaju sve veći učinak u ljudskom životu na dnevnoj bazi. Prema tome jako je važno da se istraži način kako se može pridonijeti razvoju robota temeljenih na slobodnom govoru kao što su ne zadatkom orijentirani roboti.

4.3. Domenom specifični roboti

Ovdje razlikujemo dvije vrste razgovornih robota koji mogu biti roboti otvorene i zatvorene domene. Otvorena domena nema specifičnu temu/domenu razgovora dok zatvorena ima. Prema tome neke od zatvorenih domena mogu biti edukacija, zdravlje, automobili itd. Time se postiže efikasnost generiranja točnijeg odgovora od strane robota [3].

Roboti otvorene domene konstantno napreduju, ali tehnologija je i dalje nedovoljno razvijena da bi se dobio jasan odgovor sa 100% točnosti. Takvi roboti se koriste u tvrtkama s velikom vanjskom korisničkom podrškom, jer je potreba za podacima velika, a te podatke među prvima mogu pružiti te tvrtke. Roboti zatvorene domene su mnogo lakše ostvarivi s pogleda tehnologije, ali točnost prepoznavanja govora još uvijek nije na vrhuncu, korisničko iskustvo može biti loše [8].

4.4. Roboti temeljeni na tekstu i glasu

Razgovorni roboti temeljeni na tekstu i glasu se razlikuju po načinu interakcije s korisnicima. Pretvorba govora u tekst (engl. speech-to-text) je jedna od najvažnijih funkcija koju mora imati razgovornih robot ako želi biti kompetitivan na tržištu. Pretvorba govora u tekst uključuje kriterije poput veličine rječnika (broj riječi jezika – milijuni riječi) i neovisnost o korisniku koji govori (sposobnost razgovornog robota da prepozna takvog korisnika, njegov jezik, narječje...). Svakako, svaki robot mora biti u sposobnost obrade kontinuiranog toka riječi što zahtjeva segmentaciju i tokenizaciju ulaza. Kako je i poznato dok korisnik govori može postojati šum i smetnje prilikom govora (od okoline, od lošeg mikrofona...) te je posljedično tome važno da razgovorni robot može filtrirati smetnje te čim preciznije razumjeti govor korisnika [3].

5. Razvojni okviri za razgovorne robote

Razvoj razgovornih robota sve više napreduje kako i napreduje razvoj informacijskih sustava i ostalih tehnologija. Vrlo jednostavne razgovorne robote koji odgovaraju na pojedinačne komande ili pitanja sa striktno definiranim odgovorom je lako razviti, no kada se ta striktna pitanja i odgovori pretvore u nešto apstraktnije(slobodan razgovor s robotom) tada nastaje problem razvoja takvih robota. Taj problem, iako ne u potpunosti, rješavaju različiti razvojni okviri za razgovorne robote.

Prema enciklopediji razvojni okvir je programsko okruženje koje definira osnovnu funkcionalnost i arhitekturu proizvoda. Okvir pruža standardnu strukturu u kojoj se aplikacije kreiraju i prilagođavaju; često je izvršna platforma [9].

Kroz ovo poglavlje spomenut će se i opisati nekoliko razvojnih okvira gdje se efikasno mogu razviti razgovorni roboti. Neki od njih su Google Dialogflow, Amazon Lex, Wit.ai, Open Dialog, ChatGPT – restricted API, DeepPavlol, Microsoft Bot Framework, Botkit, Pandorabots, IBM Watson, LUIS, Rasa, Botonic, Tock, Claudia Bot Builder i mnogi drugi. Primjerom će se potkrijepiti njihove značajke, prednosti i nedostaci.

5.1. Google Dialogflow

Google Dialogflow[4] okvir je proizvod tvrtke Google. Prema dokumentaciji Dialogflow je NLU platforma koja olakšava dizajn i integraciju konverzacijskog korisničkog sučelja u mobilnu aplikaciju, web aplikaciju, uređaj, robota, interaktivni sustav za glasovne odgovore itd. Također Dialogflow može analizirati više vrsta ulaznih podataka od korisnika, uključujući tekst ili audio ulaze te isto tako može odgovoriti putem teksta ili sintetičkim govorom. Važno je napomenuti kako je Dialogflow dio ponude AI-a za razgovor unutar Google Clouda [10].

Google Dialogflow okvir za razvoj razgovornih robota se dijeli na dvije inačice. To su Dialogflow CX i Dialogflow ES. Oba dvije inačice pružaju servis virtualnog agenta za razgovorne robote i kontaktne centre [10]. U daljnjem tekstu će se kratko opisati obje inačice te će se dati primjer razgovornog robota razvijenog na jednoj od inačica.

[4] Google „Dialogflow“, Google Cloud, pristupljeno 10.lipnja 2024. https://cloud.google.com/dialogflow?hl=hr

5.1.1. Dialogflow CX

Dialogflow CX je namijenjen za razvoj naprednih agenata s inovacijskim sposobnostima za velike i komplekse upotrebe. Glavne značajke CX inačice su [11]:

- Višejezična podrška
  - 30+ jezika i narječja
- Analize: performanse agenta i angažman kupaca
  - Napredne nadzorne ploče performansi
  - Izvoz podataka na nadzorne ploče
  - Vizualizacija temeljene na stanju
- Integracija više kanala
  - Integracija na svim digitalnim kanalima, web-u, mobitelima i drugima
- Unaprijed izgrađeni agenti
  - Ubrzavaju vrijeme proizvodnje s bibliotekom agenata s uobičajene slučajeve upotrebe
  - 9 agenata spremnih za proizvodnju za slučajeve upotrebe unutar industrija poput: telekomunikacija, maloprodaja, financijskih usluga, putovanja i više
- Napredni AI
  - Najsuvremeniji modeli NLU temeljeni na BERT-u
  - Vrhunski modeli prepoznavanja i sinteze govora
- Grafički graditelj toka
  - Smanjuje vrijeme razvoja za 30% s intuitivnim grafičkim graditeljem za strojeve vizualnog stanja
- Modeli temeljni na stanju (prebacivanje između tema i upravljanje složenim tokovima)
  - Ponova upotreba namjera i intuitivno definiranje prijelaza i uvjeta podataka
- Dodatna pitanja (odstupanja u razgovoru)
  - Modeli sposobni za jednostavno definiranje i otkrivanje manjih skretanja s teme razgovora
- Interaktivni glasovni odgovori (engl. Interactive voice response) postavke (optimizacija za implementacije AI kontakt centra)
  - Podržava postavke poput DTMF (engl. Dual tone multi-frequency), prijenosa poziva agenta uživo, prekidanja, vremenskih ograničenja govora

- Moduli temeljeni na protoku (jednostavno upravljanje agentima i istovremen rad na neovisnim tokovima)
    - Podržava do 20 neovisnih tokova razgovora s 40 000 namjera (engl. intents)
    - Zajedničke namjere i fraze obuke kroz tokove
- Testiranje (procjena kvalitete agenata za otkrivanje pogrešaka)
    - Testni slučajevi za kontinuiranu evaluaciju
- Upravljanje od početka do kraja
    - Verzije na razini protoka i podrška za okruženja za testiranje i implementaciju
    - Izvorno podržani eksperimenti i podjela prometa
- Identifikacija glasovnog korisnika
    - Brzo identificiranje korisnika kroz njihov glas
    - Povećana sigurnost s dodatnim slojem verifikacije
    - Smanjenje frustracije eliminiranjem lozinka i pinova

Kako se može vidjeti iz priloženog Dialogflow CX je namijenjen za velika poduzeća koja moraju pratiti sve svoje podatke te ih analizirati i iz priloženih rezultata Dialogflow CX-a može čak i mijenjati način korištenja razgovornih robota. Iako CX inačica ima mnogo više značajki nego ES inačica, u radu će se detaljno objasniti ES inačica i pojedini pojmovi. Dialogflow ES inačica pruža mogućnost razvoja kvalitetnog razgovornog robota što će se moći i vidjeti u sljedećem poglavlju.

5.1.2. Dialogflow ES (Essentials)

Dialogflow ES (engl. Essentials) je namijenjen za razvoj standardnih agenata za male do srednje i jednostavne do umjerene složene primjene. Najbitnije značajke Dialogflow ES su [11]:

- Višejezična podrška
    - 30+ jezika i narječja
- Analize: performanse agenta i angažman kupaca
    - Nadzorne ploče performansi
    - Izvoz podataka na nadzorne ploče
- Integracija više kanala
    - Integracija u kanale poput Google Assistant, Slack, Twitter i druge
- Unaprijed izgrađeni agenti

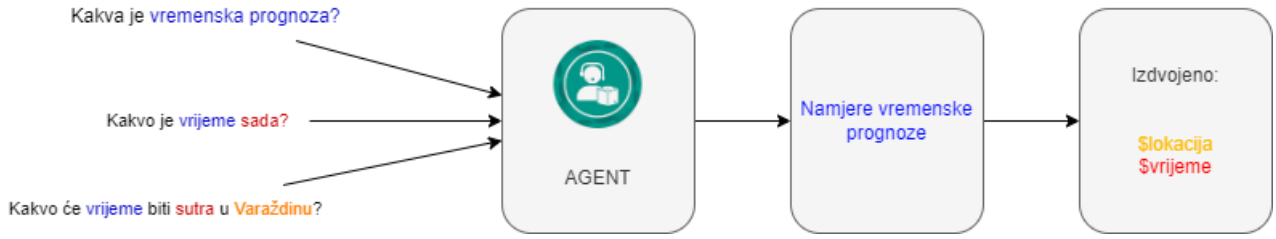
    - Ubrzavaju vrijeme proizvodnje s bibliotekom agenata s uobičajene slučajeve upotrebe
    - 40+ agentnih predloška za izgradnju razgovora za određivanje destinacija večere, rezervacije hotela, navigacije, IoT (engl. Internet of Things) i druge
- Napredni AI
    - Standardni visoko kvalitetni NLU modeli
    - Vrhunski modeli prepoznavanja i sinteze govora

Iz priloženog se može vidjeti kako ES inačica ima manje značajki od CX, no ne može se reći da je mnogo slabija. U daljnjem tekstu dati će se primjer razgovornog robota razvijenog na ES inačici te kako implementirati i integrirati razvijenog razgovornog robota u aplikaciju.

Prije nego što se da uvid u kreiranje razgovornog robota pomoću Dialogflow ES inačice objasnit će se nekoliko pojmova koji su osnova razvoja. Definirat će se što su agenti, namjere, entiteti, kontekst, slijedne namjere te opisat će se konzola kroz koju se razvija robot.

Dialogflow agent (engl. agent) je virtualni agent koji procesira istovremene razgovore s krajnjim korisnicima robota. To je NLU modul koji razumije nijanse ljudskog jezika. Obavlja funkciju prijevoda teksta ili audio izvora krajnjeg korisnika tokom razgovora u strukturirane podatke za aplikacije i usluge. Razvojni inženjeri dizajniraju i izgrađuju agenta kako bi obradili vrste razgovora potrebne za njihov sustav. Dialogflow agent je sličan ljudskom agentu u pozivnom centru. Oba su obučavana da obrade očekivane scenarije razgovora [12].

Namjere (engl. intents) kategoriziraju namjeru krajnjeg korisnika za jedan razgovor. Za svakog agenta razvojni inženjeri definiraju mnoge namjere, gdje tada te namjere upravljaju potpunim razgovorom. Dialogflow povezuje izraze korisnika s najboljom namjerom unutar agenta. Podudaranje s namjerom također je poznato kao klasifikacija namjera. Kao primjer, može se kreirati agent za vremensku prognozu koji prepoznaje i odgovara na pitanja o vremenu. Tada bi se definirale namjere za pitanja o vremenu i vremenskog prognozi. Ako korisnik pita „Kakva je prognoza?“ Dialogflow bi povezao to pitanje s nekom od namjera za prognozu. Također postoji mogućnost izdvajanja korisnih informacija iz izraza krajnjeg korisnika [12]. Na slici 1. se može vidjet vizualni prikaz izdvajanja takvih informacija.

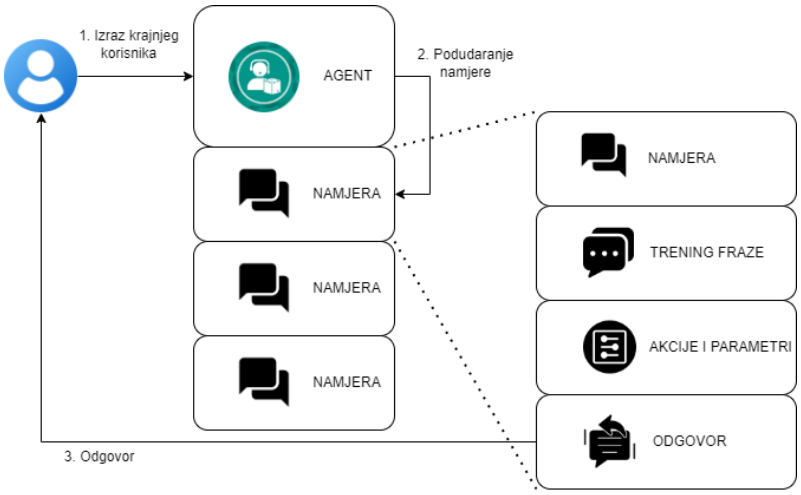


Slika 1: Izdvajanje informacija iz pitanja (izrada autora prema [12])

Svaka namjera sadrži sljedeće [12]:

- Trening fraze – su primjeri fraza koji bi krajnji korisnici mogli reći. Kada izraz krajnjeg korisnika nalikuje jednoj od ovih fraza, Dialogflow prepoznaje određenu namjeru. Također strojno učenje proširuje već unesene fraze tako da razvojni inženjer ne mora pokriti sva pitanja korisnika.
- Akcija – Razvojni inženjer definira akciju na temelju namjere. Kada se namjera podudara, Dialogflow pokreće definiranu akciju.
- Parametri – Kada se namjera podudara, Dialogflow pruža izdvojene vrijednosti iz izraza krajnjeg korisnika kao parametre. Svaki parametar ima entitetni tip. Parametri su ustvari strukturirani podaci koji se mogu koristiti za izvršavanje neke logike unutar aplikacije gdje je sadržan sam razgovorni robot.
- Odgovorni – Definirani tekstualni, govorni ili vizualni odgovori koji će se vratiti krajnjem korisniku. Ti odgovori mogu korisniku pružiti odgovore, postaviti mu dodatno pitanje ili jednostavno prekinuti razgovor.

Na slici 2. može se vidjeti tijek podudaranja namjera i kreiranja odgovora.



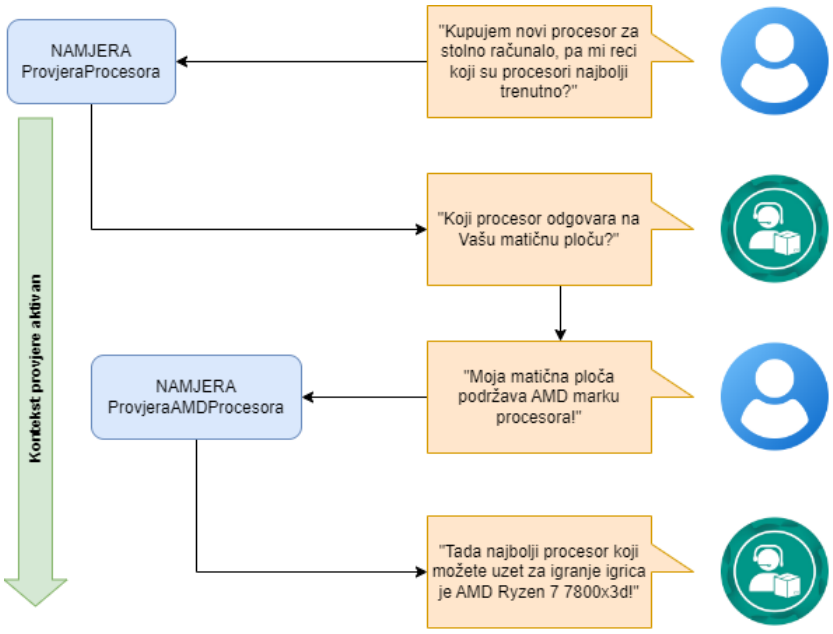
Slika 2: Tijek podudaranja namjera i kreiranja odgovora (izrada autora prema [12])

Svaka namjera ima tip, nazvan entitetni tip koji određuje kako se podaci iz izraza krajnjeg korisnika izdvajaju. Dialogflow pruža unaprijed definiran sustav entiteta koji se mogu koristiti za izdvajanje uobičajenih izraza. Neki od tih entiteta za podudaranje su za:

- Datume
- Vremena
- Boje
- E-mail adrese

Također razvojni inženjer je u mogućnosti proizvesti svoje entitete koji su mu potrebni u razvoju razgovornog robota na kojem trenutno radi. Ti entiteti tada mogu biti vezani za bilo koju temu poput npr. solarne ploče, komponente stolnog računala, namještaj itd. [12].

Dialogflow sadrži kontekste koji su slični kao u prirodnom govoru. Ako krajnji korisnik tj. sugovornik kaže „oni su iz Njemačke“ razgovorni robot ili druga osoba treba kontekst kako bi razumio tko su „oni“. Za Dialogflow razgovornog robota kontekst mora biti pružan uz izraz krajnjeg korisnika kako bi se podudarao s namjerom. Koristeći kontekste, može se kontrolirati tijek razgovora. Za svaku namjeru može se konfigurirati više konteksta, postavljajući ulazne i izlazne kontekste, definirane imenima. Kada se nađe namjera koja se podudara, njezin izlazni kontekst postane aktivan. Prema aktivnim kontekstima, Dialogflow s većom sigurnošću može pronaći podudaranje namjere koji imaju ulazne kontekste, a istovremeno odgovaraju trenutno aktivnim kontekstima. Na slici 3. možemo vidjeti vizualni prikaz toka razgovora koristeći kontekste.



Slika 3: Tijek razgovora s kontekstom (izrada autora prema [12])

Razvojni inženjeri također mogu koristit slijedne namjere (engl. follow-up intents) da automatski postave kontekste za parove namjera. Slijedna namjera je podnamjera svoje roditeljske namjere. Nakon kreiranje slijedne namjere, izlazni kontekst se automatski dodjelu roditeljskoj namjeri, a ulazni kontekst slijednoj namjeri. Slijedna namjera se podudara ako i samo ako se podudara roditeljska namjera. Postoji mogućnost kreiranja ugniježđenih slijednih namjera. Kako i za namjere Dialogflow pruža unaprijed definirane slijedne namjere uobičajenih odgovora poput „da“, „ne“, „otkaži“ itd. Svakako, i dalje ostaje mogućnost kreiranje potrebnih slijednih namjera prema temi razgovornog robota [12].

Postoje dvije vrste interakcija korisnika odnosno razvoja razgovornog robota na temelju interakcija s korisnikom. U sklopu Dialogflow-a može se govoriti o *interakciji korisnika s integracijama* te *interakciji korisnika s programskim sučeljem aplikacije* (engl. Application Programming Interface, u daljnjem tekstu API). Dialogflow se može integrirati u mnogo razgovornih platformi poput Google Assistant, Slack, Facebook Messenger itd. Prilikom korištenja integracijskog načina može se svakoj namjeri omogućiti ispunjenja (engl. fulfillment). Omogućavanjem te postavke namjere, kada se namjera podudara, prelazi se u proces dohvaćanja dinamičkih podataka postupkom koje je razvojni inženjer postavio. Dialogflow ustvari šalje zahtjev na servis s informacijama o namjeri te dobiva dinamički generiran odgovor. Također, ako ispunjenje ne prođe zbog npr. nedostupnosti servisa, tada se generira statički odgovor koji je zadan. Koraci u razgovoru s integracijom su sljedeći [12]:

1. Krajnji korisnik napiše ili kaže izraz ili rečenicu.
2. Dialogflow podudara izraz s namjerom i izdvaja parametre
3. Dialogflow šalje zahtjev na servis razvojnog inženjera. Zahtjev sadrži informacije o podudarnoj namjeri, akciju, atribute, i odgovor definiran za namjeru.
4. Servis tada odrađuje potrebne akcije, poput upita na bazu podataka ili vanjski API poziv.
5. Servis vraća odgovor prema Dialogflow-u. Taj odgovor sadrži odgovor koji bi se morao poslati krajnjem korisniku.
6. Dialogflow šalje odgovor krajnjem korisniku.
7. Krajnji korisnik vidi ili čuje odgovor.

Svakako, kada ne postoji direktna integracija Dialogflow-a s platformom tada se razvojni inženjeri oslanjaju na interakciju korisnika s API-em Dialogflow-a. Koraci u ovom postupku su sljedeći [12]:

1. Krajnji korisnik napiše ili kaže izraz ili rečenicu.
2. Servis razvojnog inženjera šalje taj izraz prema Dialogflow-u na proces podudaranja.
3. Dialogflow šalje odgovor podudarane namjere servisu. Zahtjev sadrži informacije o podudarnoj namjeri, akciju, atribute, i odgovor definiran za namjeru.
4. Servis vraća odgovor prema Dialogflow-u. Taj odgovor sadrži odgovor koji bi se morao poslati krajnjem korisniku.
5. Servis šalje odgovor krajnjem korisniku.
6. Krajnji korisnik vidi ili čuje odgovor.

Razvoj razgovornih robota kroz platformu kao što je Dialogflow omogućava fleksibilnost i jednostavnost u integraciji s različitim korisničkim interakcijama. Bez obzira na to da li se koristi direktna integracija s platformama poput Google Assistant, Slack ili Facebook Messenger, ili se oslanja na API za interakciju, Dialogflow pruža robustne alate za podudaranje namjera, obradu parametara i generiranje dinamičkih odgovora.

5.2. Amazon Lex

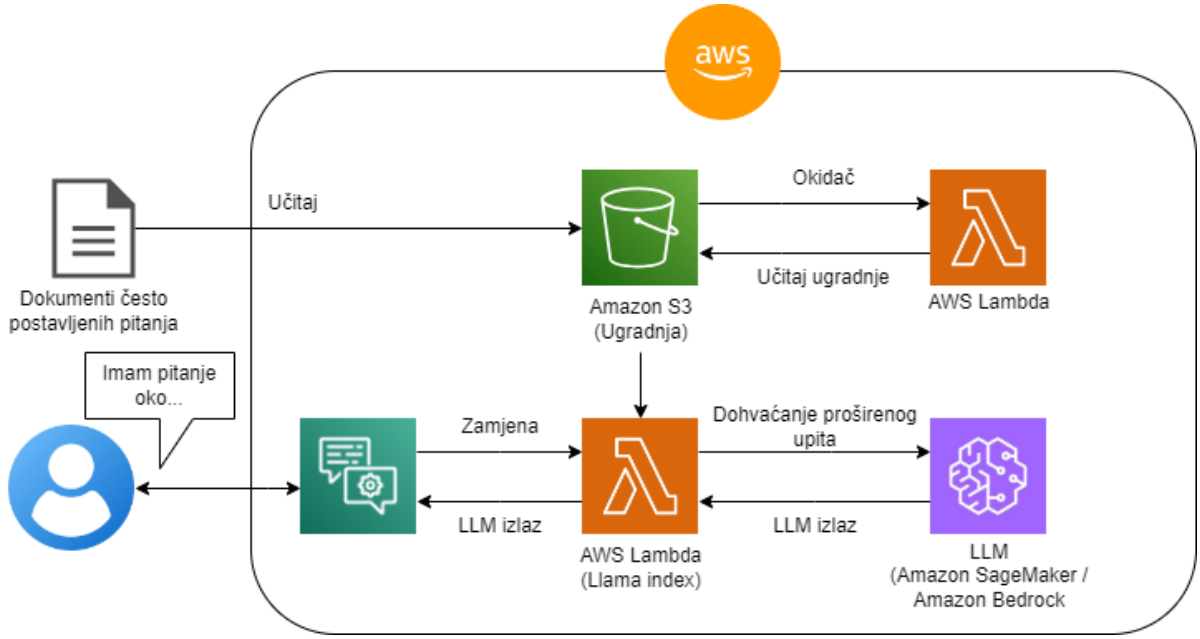
Amazon Lex[5] je potpuno upravljana AI usluga koja koristi napredne modele prirodnog jezika za dizajniranje, izgradnju, testiranje i implementaciju komunikativnih sučelja za glas i tekst. Kroz Amazon Lex servis može se [13]:

- Dodati AI koji razumije namjere, održava kontekst razgovora i automatizira zadatke na više jezika
- Dizajnirati i implementirati višekanalni (engl. omnichannel) konverzacijski AI bez da se brine o fizičkom sklopovlju ili infrastrukturi
- Besprijekorno povezati s ostalim Amazon Web servisima (AWS) kako bi omogućio upite na bazu podataka, izvršavanje poslovne logike, praćenje performansi itd.

Amazon Lex pruža funkcionalnost i fleksibilnost NLU-a i automatskog prepoznavanja govora (engl. automatic speech recognition, u daljnjem tekstu ASR) kako bi pruža mogućnost

[5] Amazon Web Services. „Amazon Lex“ Amazon, pristupljeno 10.lipnja 2024. https://aws.amazon.com/lex/

razvoja razgovornog robota s visokom razinom korisničkog iskustva s prirodnim, konverzacijskim interakcijama. Također koristi velike jezične modele (engl. large language models, u daljnjem tekstu LLM) kako bi poboljšao iskustvo izgradnje i omogućio prirodniji razgovor s krajnjim korisnicima. LLM-ovi unapređuju NLU i generiranje relevantnijeg odgovora. To je rješenje koje integrira sustav dohvaćanja proširenog generiranja (engl. Retrieval Augmented Generation, u daljnjem tekstu RAG) s razgovornim robotima koji može koristiti baze znanja za pružanje boljeg iskustva samoposluživanja. RAG sustav uzima korisnički upit, pronalazi relevantne dokumente i generira odgovor uzimajući u obzir dokumente i upit. RAG također pruža angažirano komunikacijsko iskustvo te koristi LLM-ove i podatke poduzeća za poboljšanje odgovora razgovornih robota. Kao potpuno upravljanja usluga, Amazon Lex se automatski skalira, tako da se ne mora brinuti u upravljanju infrastrukturom [13]. Na slici 4. može se vidjeti kako RAG može biti implementiran s Amazon Lex-om.



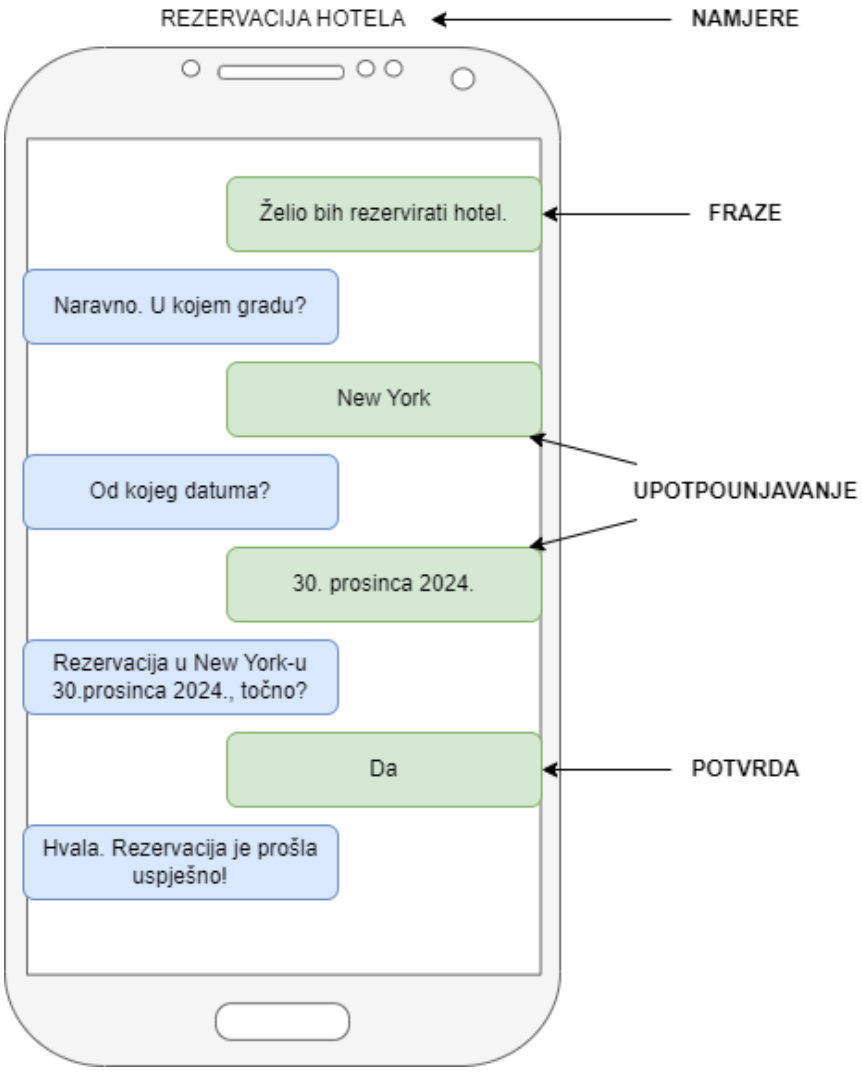
Slika 4: Implementacija RAG-a unutar Amazon Lex-a (izrada autora prema [13])

Najbitnije prednosti kod Amazon Lex servisa su da nudi poboljšanje korisničkog iskustva kroz sofisticirani NLU tako što smanjuje vrijeme čekanja korisnika te ima neprekidnu podršku na svim kanalima. Također korisnik se može angažirati preko njemu preferiranog kanala poput telefona, poruka, web stranica i drugih. Ima sposobnost rukovanja velikim brojem razgovora. Omogućuje razvojnim inženjerima besprijekornu implementaciju što smanjuje vrijeme razvoja. Amazon Lex V2 razgovorni roboti mogu integrirati s Facebook-om, Slack-om i Twilio razgovornim platformama [13].

Slučaji upotrebe (engl. use-cases) Amazon Lex-a mogu biti sljedeći [13]:

- Razgovorni robot za pitanja i odgovore ili često postavljena pitanja
- Samo usluživanje – izvršavanje transakcija, ponovno postavljanje lozinki, ažuriranje adresa.
- Pametno preusmjeravanje – efikasno usmjeravanje kontakta prema pravom agentu
- Povećanje produktivnosti zaposlenika

Amazon Lex kao i svaki drugi razgovorni robot sadrži koncepte bazičnog razgovornog robota. U te koncepte spadaju namjere, izrazi ili fraze koje se podudaraju unutar namjere iz ulaza krajnjeg korisnika (govor, tekst), upotpunjavanje konteksta namjere (primjer je dodavanje odredišta prilikom rezerviranja leta) te ispunjenje ili potvrda korisnika [13]. Koncepti su vizualno prikazani na slici 5.



Slika 5: Prikaz koncepata (izrada autora prema [13])

Amazon Lex sadrži i neke korisne značajke koje razvojni inženjeri mogu iskoristiti. To su značajke poput korištenja već postojećih prijepisa/transkripata razgovora, vizualni dizajner za izgradnju razgovornog tijeka te povećanje točnosti prepoznavanja govora s prilagođenim rječnikom, savjetima tijekom izvršavanja i pravopisnim stilovima [13].

5.3. Wit.ai

Wit.ai[6] je platforma za razvoj razgovornih robota koja je temeljena na suradnji s Facebook-om. Točnije rečeno Wit.ai je dio Meta platformi namijenjena za pomoć razvojnim inženjerima pri kreiranju aplikacije koja razumije ljudski jezik. Kako kažu, pružaju razvojnim inženjerima jednostavan način razvoja aplikacije koja razumije tekstualne i glasovne komande te uči iz svake interakcije [14].

Wit.ai pruža upute za izradu Wit aplikacije u službenoj dokumentaciji. Tamo se nalaze koraci koji su potrebni razvojnim inženjerima kako bi razvili Wit aplikacija odnosno razgovornog robota. Početak samo razvoja je kreiranje novi aplikacije unutar Wit.ai stranice. Nakon što se kreirala aplikacija je potrebno istrenirati samu aplikaciju. Wit aplikacija sadrži sekciju razumijevanja u čemu su uključene fraze i namjere. To su primjeri upita poput „Kakvo će vrijeme biti danas?“. Druga sekcija su upotpunjavanja. Wit ovdje upotpunjuje nedostale dijelove za potpuno podudaranje s nekom namjerom. Ovo se može opisati sljedećim primjerom: „Rezerviraj avionsku kartu?“. Wit će ovdje postaviti dodatna pitanja poput „Gdje želite ići?“ i „Kada želite ići?“. Također Wit je u mogućnosti pretpostaviti i sam upotpuniti nedostale atribute neke namjere. Svakako, to će se toliko dobro izvršiti koliko je razvojni inženjer dobro istrenirao samu aplikaciju te koliko je točno postavio namjere. Wit.ai pruža mogućnost, što je i sljedeći korak u razvoju, da razvojni inženjeri mogu unaprijediti razinu točnosti prepoznavanja tako da ručno unesu pitanja koja će se procesirati. Nakon procesiranja dobit će se rezultat samouvjerenosti aplikacije. Samouvjerenost tada razvojni inženjer može unaprijediti tako da sam validira odgovore Wit aplikacije. Nakon ovih koraka razvojni inženjer je već u mogućnosti preispitati Wit aplikaciju. Ako ste uputi pitanje ili komanda koja još ne postoji u sklopu neke namjere Wit aplikacija će proizvesti krivi odgovor, ali će isto tako pospremiti tu komandu u sekciju razumijevanja. Sve fraze koje još prije nisu viđene ili procesirane će se vidjeti u toj sekciji. Za svaku tu frazu možemo kreirati neku novu namjeru ili

[6] Facebook, „Wit.ai“ Wit.ai, pristupljeno 10.lipnja 2024. https://wit.ai

jednostavno svrstati te fraze unutar neke druge već kreirane namjere. Također na toj frazi Wit.ai nudi mogućnost kreiranje imenica, glagola ili pridjeva. To se može napraviti tako da se označi neka riječ ili skup riječi te se označi kao entitet. Na primjer, fraza je „Hoće li danas temperatura dostići 30 stupnjeva?“. Na toj frazi može se označiti „30 stupnjeva“ i kreirati novi entitet wit/temperatura. Nakon kreiranja entiteta razvojni inženjer bi trebao ponovo istrenirati Wit aplikaciju kako bi dobio bolje rezultate. Rezultati Wit aplikacije mogu se vidjeti unutar JavaScript objektne notacije (engl. JavaScript Object Notation, u daljnjem tekstu JSON) [15]. Jedan takav primjer je prikazan nakon paragrafa. Primjer je direktno preuzet s izvora [15].

```
{
        "text": "set the temperature to 70 degrees",
        "intents": [{
                "id": "228869141559004",
                "name": "temperature_set",
                "confidence": 0.9954
        }],
        "entities": {
                "wit$temperature:temperature": [{
                        "id": "957505051332853",
                        "name": "wit$temperature",
                        "role": "temperature",
                        "start": 23,
                        "end": 33,
                        "body": "70 degrees",
                        "confidence": 0.9785,
                        "entities": [],
                        "unit": "degree",
                        "type": "value",
                        "value": 70
                }]
        },
  "traits": {}
}
```

Programski kod 1: JSON prikaz rezultata Wit.ai (preuzeto s [15])

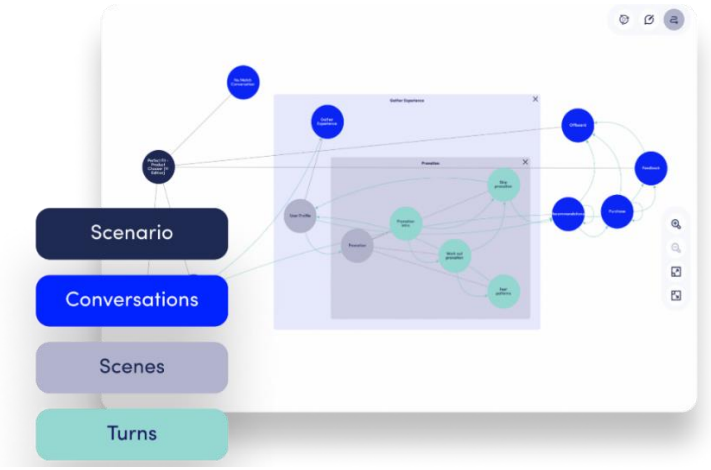
Gore opisan proces je proces kreiranja Wit aplikacije koja može biti integrirana u razgovorne robote. Razgovorni robot tada može dohvaćati ili ažurirati i dodavati nove podatke unutar Wit aplikacije koja pruža odgovore razgovornom robotu. Te odgovore razgovorni robot dalje prikazuje korisniku.

5.4. OpenDialog

OpenDialog[7] platforma je još jedna u nizu platformi za razvoj razgovornih robota. Prati smjernice izrade razgovornog robota koji se temelji na namjerama, podudaranjima namjera i izraza, ekstrakcija imenica, pridjeva i glagola iz izraza itd. To je platforma za kreiranje razgovornih robota temeljenih na generativnoj umjetnoj inteligenciji (engl. Generative AI). Pomaže razvojnim inženjerima da zadrže potpunu kontrolu nad korisničkim iskustvom te uključuje najbolje značajke LLM-ova za automatizirane razgovore [16].

Početna točka razvoja razgovornog robota na platformi je nadzorna ploča radnih prostora (engl. workspaces). Svaki radni prostor sadrži jedan ili više scenarija koji mogu biti nacrti (na početku stvaranja) i aktivni (nakon omogućavanja). Preko nadzorne ploče radnih prostora razvojni inženjer može kreirati, upravljati, duplicirati, brisati, izvoziti i uvoziti scenarije te upravljati radnim prostorima. Također, postoji mogućnost uvoz scenarija putem JSON datoteke [16].

Potrebno je razumjeti OpenDialog model kako bi se moglo razvijati razgovornog robota. OpenDialog model omogućava veliku fleksibilnost. Model se sastoji od razina i komponenti koji su potrebni kako bi se kreirao razgovorni robot. Iako je ovo platforma za razvoj razgovornog robota kao i prijašnje opisane u radu, ova platforma kroz OpenDialog model omogućuje da se dizajn sustava zamjeni s dizajnom razgovora, prelazeći sa scenarija visoke razine na pojedinačne zaokrete unutar razgovora. Različite razine modela možemo vidjeti na slici 6.



Slika 6. Modeli unutar OpenDialog platforme [16]

[7] „OpenDialog“ pristupljeno 10.lipnja 2024. https://opendialog.ai/

Prema slici 6. model se sastoji od 4 komponente, a to su scenariji, razgovori, scene i zaokreti. Kroz sljedeće paragrafe objasnit će se svaka komponenta detaljnije te će se na kraju sumirati.

Scenariji mogu biti aktivni ili u nacrtu (engl. draft). Aktivni scenariji su dostupni za korištenje u sučeljima razgovora dok scenariji u nacrtu nisu. Kroz platformu OpenDialog-a mogu se modificirati atributi scenarija poput imena, opisa, tumača i postaviti različiti uvjeti. Postavljanje uvjeta na scenarij je poput više agentne strukture, gdje uvjeti za scenarije identificiraju scenarij za razmatranje. Scenariji se mogu brisati, duplicirati i izvoziti [16].

Modelima razgovora se isto kao i scenarijima mogu mijenjati atributi poput imena, početnog ponašanja i tumača. Također razgovori imaju svoje uvjete koji omogućuju provjeru atributa unutar konteksta. Primjer, u scenariju e-trgovine, registrirani korisnici mogu vidjeti posebne ponude koje neregistrirani ne vide. Različiti razgovori mogu se postaviti za registrirane i neregistrirane korisnike [16].

Scenama se također mogu mijenjati postojeći atributi te isto tako dodavati uvjeti kao i na razgovore. Na primjer, u procesu s više koraka, određeni koraci mogu biti preskočeni ili uključeni na temelju atributa kao što je korisnikova dob. To se provjerava pomoću uvjeta [16].

Sličnost OpenDialoga s ostalim platformama je upravo u modelu zaokreta i namjera. Njima se mogu konfigurirati naziv, početno ponašanje i uvjeti. Prava sličnost je u tome što svaki zaokret sadrži namjere odnosno odgovore i zahtjeve namjera. Namjeri unutar zaokreta rade na način da se gleda koja vrsta namjere slijedi iza one prve. Kada postoji namjera zahtjeva, logika odnosno kako navodi OpenDialog „motor“ će pogledati postoji li namjera odgovora u istom zaokretu te će ju izvršiti. Moguće je imati samo jednu namjeru unutar zaokreta. Ako u zaokretu postoji samo jedna namjera (korisnička ili aplikacijska) dva su moguća scenarija:

1. Definirati prijelaz za namjeru u zaokretu. Motor ili logika će prema prijelazu probati pronaći odgovarajuću namjeru s kraja definiranog prijelaza. Ako je u zaokretu A aplikacijska namjera, tražit će se korisnička namjera u zaokretu B i obrnuto.
2. Ne definirati prijelaz, u tom slučaju se logika vraća na razinu scene i traži otvorene zaokrete s namjerom suprotnog sudionika.

Također, prihvatljivo je imati više namjera u listi. Namjere se izvršavaju redom od vrha prema dnu, pri čemu se najopćenitiji uvjet stavlja na kraj [16].

Kao što se opisalo kod platforme Wit, postoji i još nekoliko bitnih atributa i značajki koje se mogu mijenjati po potrebi. Na OpenDialog platformi to su tumač, minimalna razina samouvjerenosti za svaku namjeru, ponašanja namjera (u trenutku pisanja postoji samo *COMPLETING* ponašanje koje označuje da se izraz i namjera podudaraju, prijelazi koji su prouzročeni namjerama itd. Iz priloženog može se zaključiti kako namjere imaju neke dodatne postavke koje se mogu mijenjati. U dodatne postavke ubraja se [16]:

- Očekivani atribute – mogu se dodati posebni atributi na namjeru. Dodajući ih sustavu se kaže da pospremi specificirani atribut u dan kontekst.
- Uvjeti – Dodavajući uvjete na namjere, mogu se određene namjere skriti/pokazati prema određenim korisnicima.
- Akcije – U trenutku pisanja nema dovoljno specificiranih informacija o postavci,

Također, OpenDialog koristi predefinirana imena za pojedine namjere kako bi znao njihove točne funkcionalnosti. Neke od tih su [16]:

- Intent.core.welcome – ovo je namjera koja je povezana na kraj učitavanja neke stranice koja koristi razgovornog robota. Na završetak učitavanja se automatski otvara prozor razgovornog robota i postavljena je namjera zahtjeva.
- Intent.core.restart – ovo je namjera koja je povezana na gumb ponovnog učitavanja
- Intent.core.endChat – namjera povezana direktno na završetak razgovora s robotom
- Intent.core.TurnNoMatch – kada ne postoji podudaranje s trenutnom scenom. Logika ili motor će pokušati pronaći odgovarajuću scenu

OpenDialog sadrži i značajku dizajnera razgovora. Ta značajka pruža mogućnost kreiranja i povezivanja prije opisanih sastavnih dijelova nekog razgovora. Može povezivati namjere, zaokrete, scenarije, scene itd. Pojedini način odnosno rezultat povezivanja ne mora odgovarati drugom odnosno želi se reći da i mala promjena utječe na rezultat razgovora. Dizajniranjem razgovora daje se motoru razgovora njegova logika, drugim riječima rečeno, definira se logika razgovora. Postoji nekoliko pravila unutar samog motora koje on slijedi, a to su da ako je u trenutak prve interakcije motor prati jedino početne/startne komponente, no već prilikom druge interakcije motor ignorira te komponente i prelazi na aktivne komponente. Tijekom interakcija postoji još nekoliko prioritizacijskih pravila [16]:

- Ako namjera podudaranja definira tranziciju, motor će izvršiti tu tranziciju

- Ako ne postoji tranzicija, motor traži podudaranje s namjerom odgovora u istom zaokretu
- Ako ne postoji tranzicija niti namjera odgovora u postojećem zaokretu, motor će tražiti sljedeću namjeru unutar drugog zaokreta, ali iste scene
- Ako namjera sadrži ponašanje kraja, motor odlazi na razinu scenariji.
- Ako se ne može primijeniti ni jedno pravilo, tada je okinuta namjera „nema podudaranja“ odnosno intent.core.NoMatch ili lokalno ili na globalnoj razini (razina scenarija)

Bitna značajka OpenDialog platforme je to što se aplikacija odnosno funkcionalnost razgovornog robota može pregledati prije slanje u produkciju. U to su uključeni i predefinirani i ručno izrađeni scenariji. Također mogu se dobiti dodatne informacije o tome kada je korisnik pročitao poruku te se može pratiti rad cijelog razgovornog robota, što su funkcionalnosti za usporedbu koje će se obraditi u daljnjem dijelu rada.

6. Usporedba tehnologija razgovornih robota

Za usporedbu tehnologija razgovornih robota uzet će se u obzir tehnologije koje su opisane u radu. Uspoređivat će se prema sljedećim kriterijima, a to su jednostavnost korištenja, performanse, funkcionalnost, podrška te troškovi koji se vežu na samu tehnologiju.

6.1. Definiranje kriterija za usporedbu tehnologija

U ovom dijelu rada bit će prikazani kriteriji za usporedbu navedenih tehnologija. Za svaku od kriterija opisat će se što će se točno pratiti kod svake tehnologije, a u sljedećem poglavlju vidjet će se rezultati usporedbe. Na svakoj od tehnologija pokušat će se napraviti podjednaki razgovorni robot s istim ili sličnim mogućnostima razgovora. Najbitniji kriteriji kod platformi, aplikacija, sučelja ili tehnologija su jednostavnost korištenja jer se mnogo razvojnih inženjera može po prvi put susretati s takvom tehnologijom te ako nije jednostavno za korištenja velika je vjerojatnost zamjene te tehnologije. Sljedeće su performanse koje su bitne za razvoj aplikacija koje koriste tu tehnologiju. Ako su performanse agenta razgovornog robota loše tada će korisničko iskustvo biti loše što nikako nije dobro za razvoj aplikacije. Funkcionalnosti su uvijek bitne. Veliku ulogu igra broj funkcionalnosti i kvaliteta istih za odabir tehnologija za, u ovom slučaju razvoj agenta razgovornog robota. Ako tehnologija ima velik broj funkcionalnosti onda je i bitna podrška istih što podrazumijeva dobro razvijenu dokumentaciju, video uputstva, ljudsku podršku itd. Svakako, ne smiju se zaboraviti troškovi. Oni pak igraju veliku ulogu u svim poduzećima. Većina se poduzeća, bila mala ili velika, odlučuju za potpuni pristup tehnologiji kako bi dosegli čim veću iskoristivost i efikasnost. Svi kriteriji će se kroz sljedeća istoimena poglavlja još detaljnije objasnit.

6.1.1. Jednostavnost korištenja

Jednostavnost korištenja tehnologije često se procjenjuje kroz njezin dizajn korisničkog iskustva (engl. user interface / user experience, u daljnjem tekstu UI/UX). Dat će se objektivna ocjena prema tome koliko je dobro osmišljeno korisničko sučelje te koliko dobro to sučelje omogućuje korisnicima intuitivno snalaženje i učinkovito korištenje svih funkcionalnosti platforme.

6.1.2. Performanse

Performanse se mogu ocijeniti s nekoliko gledišta. Ako je moguće sagledat će se čimbenici kao što su izvršavanje zadataka, preciznost, pouzdanost te vrijeme odziva. Također, uzet će se u obzir vizualne metrike ako postoji pristup do njih od strane same tehnologije.

6.1.3. Funkcionalnost

Funkcionalnost tehnologije obuhvaća broj i kvalitetu funkcija koje nudi. Dat će se objektivan opis i ocjena prema samostalno provedenoj analizi. Analiza će se temeljiti na onome što tehnologija nudi kroz svoju dokumentaciju uključujući upute korištenja ako postoje.

6.1.4. Podrška

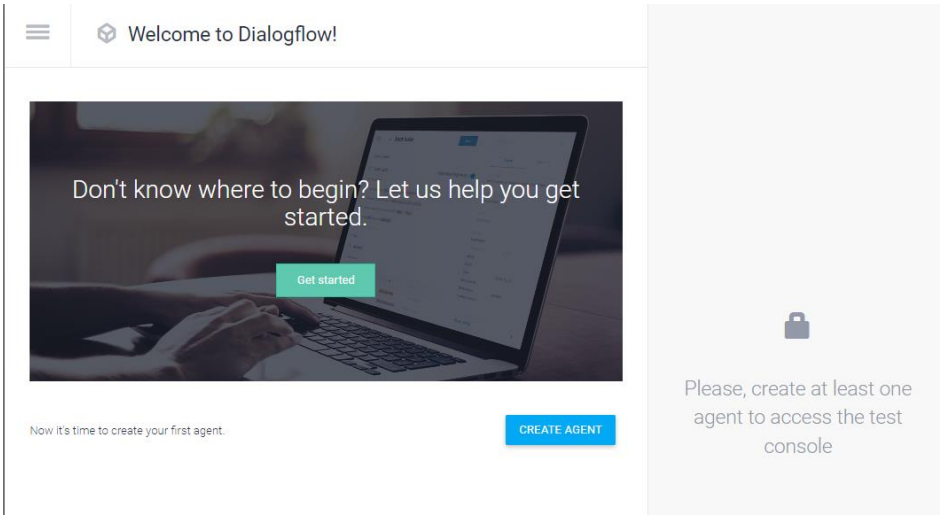
Podrška tehnologiji uključuje dostupnost dokumentacije, online resursa, foruma, te tehničke podrške od strane proizvođača ili agenata. Kvaliteta podrške može značajno utjecati na uspješno implementiranje i korištenje tehnologije u praksi te je zbog toga vrlo bitan kriterij pri usporedbi.

6.1.5. Troškovi

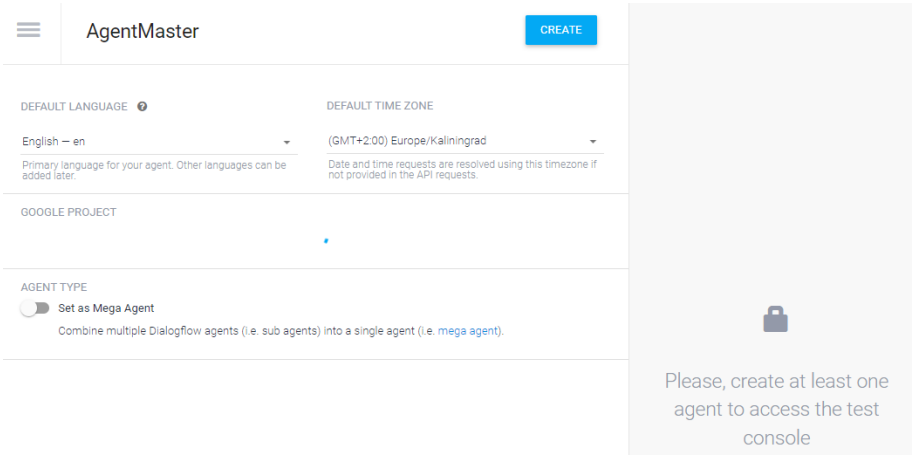
Troškovi se u ovoj temi mogu podijeliti na dvije kategorije. Cjenovne modele i dodatne troškove. Cjenovni modeli tehnologije variraju i uključuju različite opcije poput besplatnih planova, pretplata, te cijene po korisniku i interakciji. Pokušat će se dat objektivna ocjena na temelju tih modela i funkcionalnosti koje pruža svaki model. Također, u obzir će se uzeti i dodatni troškovi koji mogu biti korištenje funkcionalnosti koja je odvojena od cjenovnog modela te se plaća posebno ako postoji.

6.2. Dialogflow ES

Kreiranje početne namjere i upogonjavanje agenta razgovornog robota je bilo vrlo jednostavno. Autor je kreirao jednu testnu namjeru i isprobao kako agent radi u minimalnom vremenu. Proces korištenja je bio da se kreira agent gdje se zadalo samo ime agenta, vremenska zona i jezik, slika 7 i slika 8.

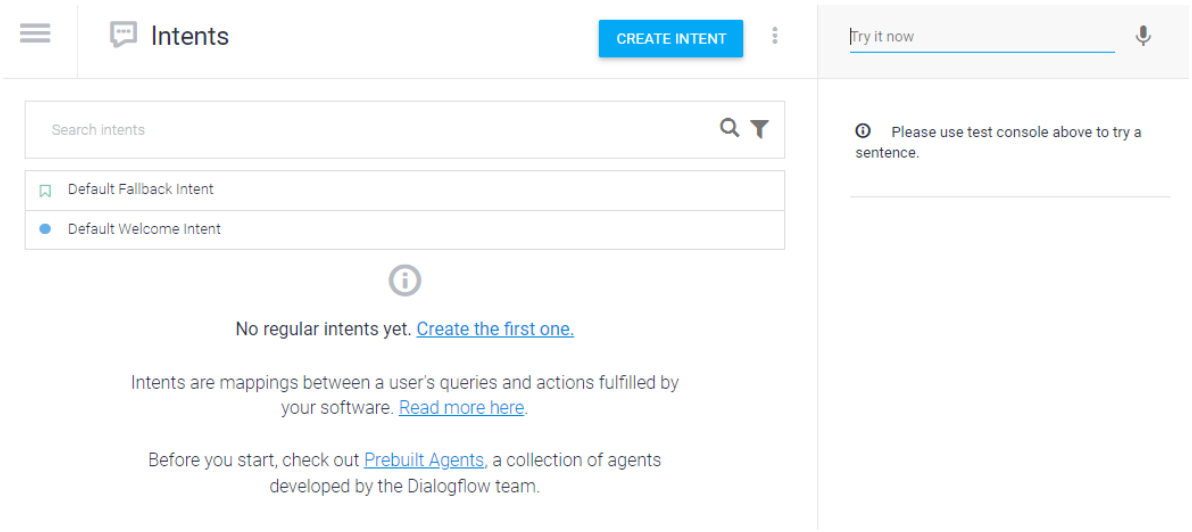


Slika 7: Kreiranje Dialogflow ES agenta (snimka zaslona)

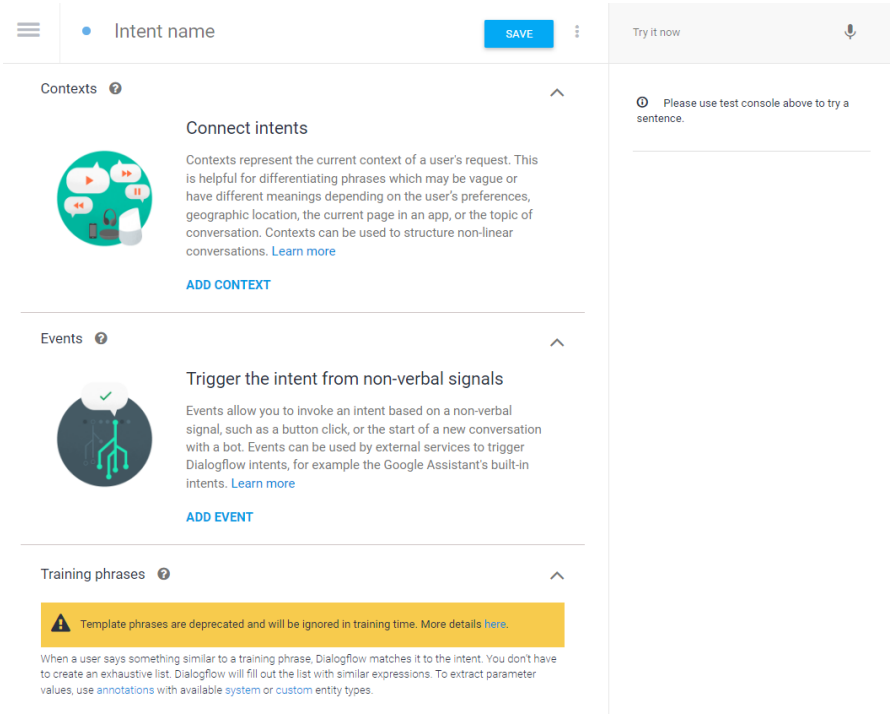


Slika 8: Pridruživanje imena, vremenske zone i jezika agentu Dialogflow ES (snimka zaslona)

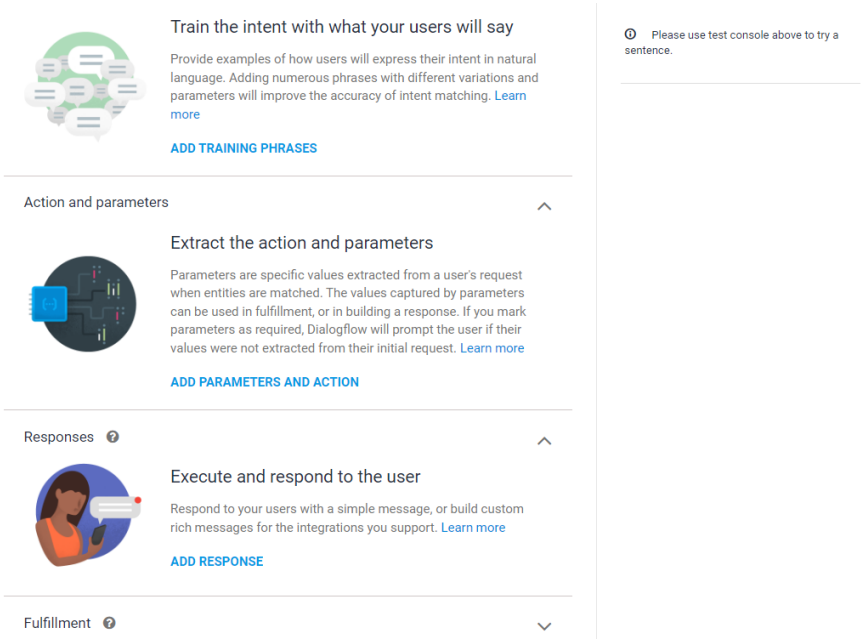
Nakon kreiranja agenta, dobila se lista namjera. Tu se klikom na jedan gumb može kreirati namjera. Kreiranje namjere je trajalo jako kratko jer je testna namjera gdje su se zadale samo trening fraze poput „Hi“ i „Hello“ te odgovor „Hello, Mr. Viljevac“. Kreiranje namjere se može vidjeti na slikama 9,10,11.



Slika 9: Kreiranje namjere Dialogflow ES (snimka zaslona)

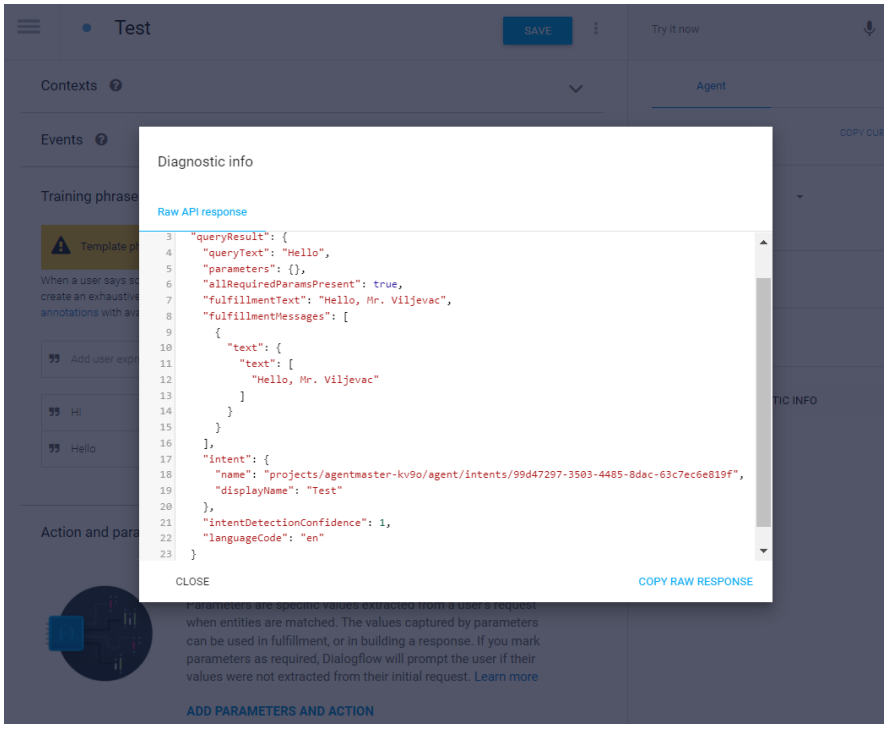


Slika 10: Dodavanje imena namjere Dialogflow ES (snimka zaslona)

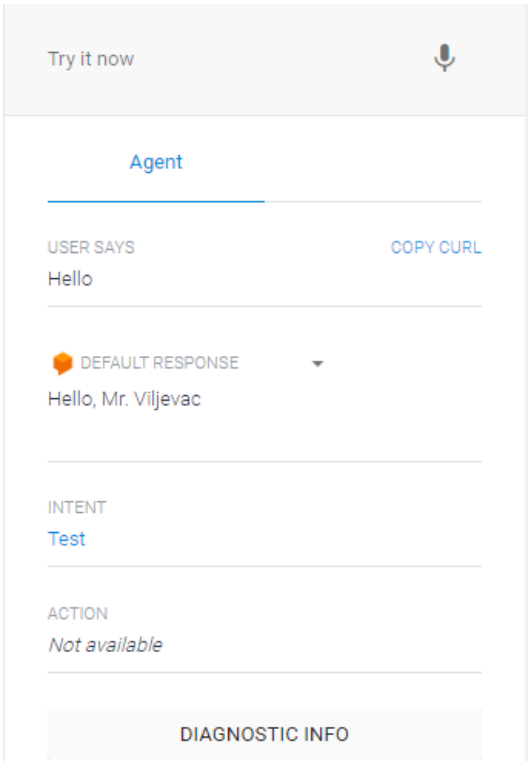


Slika 11: Dodavanje ostalih atributa Dialogflow ES (snimka zaslona)

Nakon toga kroz konzolu se odmah može testirati ta namjera gdje se dobio odgovor u jednostavnom obliku, ali može se i vidjeti odgovor koji je u JSON formatu. Odgovori se mogu vidjeti na slikama 12 i 13.

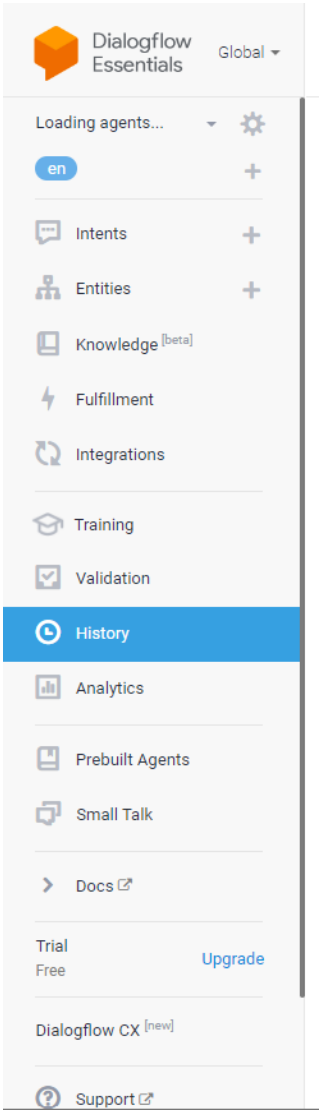


Slika 12: JSON format odgovora Dialogflow ES (snimka zaslona)



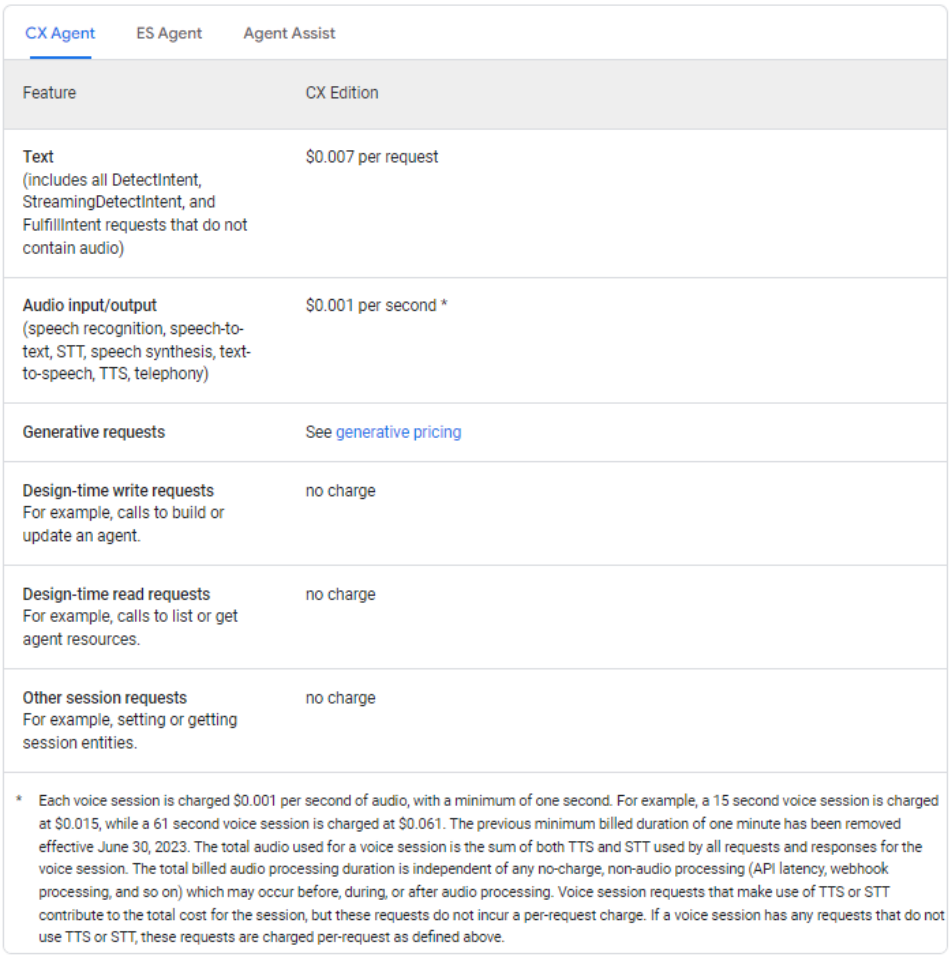
Slika 13: Odgovor agenta Dialogflow ES(snimka zaslona)

Dialogflow ES je s gledišta autora vrlo jednostavan za korištenje. Sučelje tehnologije je vrlo jednostavno za shvatiti te je posljedično tome za kreiranje agenta potrebno vrlo malo vremena (ovisno o opsegu). Performanse se konkretno nigdje ne mogu direktno vidjeti makar je odgovor od strane agenta brz i točan. Nigdje ne piše s kojom pouzdanošću je agent generirao odgovor. Posljedično tome, za performanse bi se trebao razviti opširan agent s velikim brojem kompliciranijih namjera te izračunati točnost ručno. Kao i performanse, vizualne metrike se ne mogu generirati iz poruka poslanih sa sučelja već bi se trebao integrirati agent u neku aplikaciju koji bi se tada pozivao preko API poziva. Kao što se vidi na prijašnjim slikama agent i njegove namjera imaju većinu ako ne i sve funkcionalnosti. Isto tako na slici 14 se mogu vidjeti funkcionalnost koje nisu direktno vezane na namjeru nego su više vezani uz agenta.



Slika 14: Funkcionalnosti Dialogflow ES agenta (snimka zaslona)

Vidimo kako Dialogflow ES agent ima puno funkcionalnosti koje definiraju vrlo dobro implementiranu tehnologiju za razvoj razgovornih robota. Prema mišljenju autora, podrška za Dialogflow ES je odlična. Ima jako puno strukturirane dokumentacije te je vrlo lako prema toj dokumentaciji razviti agenta za razgovornog robota. U toj dokumentaciji je također uključena i dokumentacija o API pozivima i kako implementirati i povezati agenta unutar aplikacije. Postoje tri različite vrste troškova. Troškovi CX agenta, ES agenta i agenta pomoći. CX agent naplaćuje tekstualne poruke(pojedinačno po zahtjevu u koji se ubrajaju detektiranje namjere, slanje odgovora namjere itd.) te glasovne poruke po sekundi. Svi troškovi CX agenta se mogu vidjeti na slici 15.



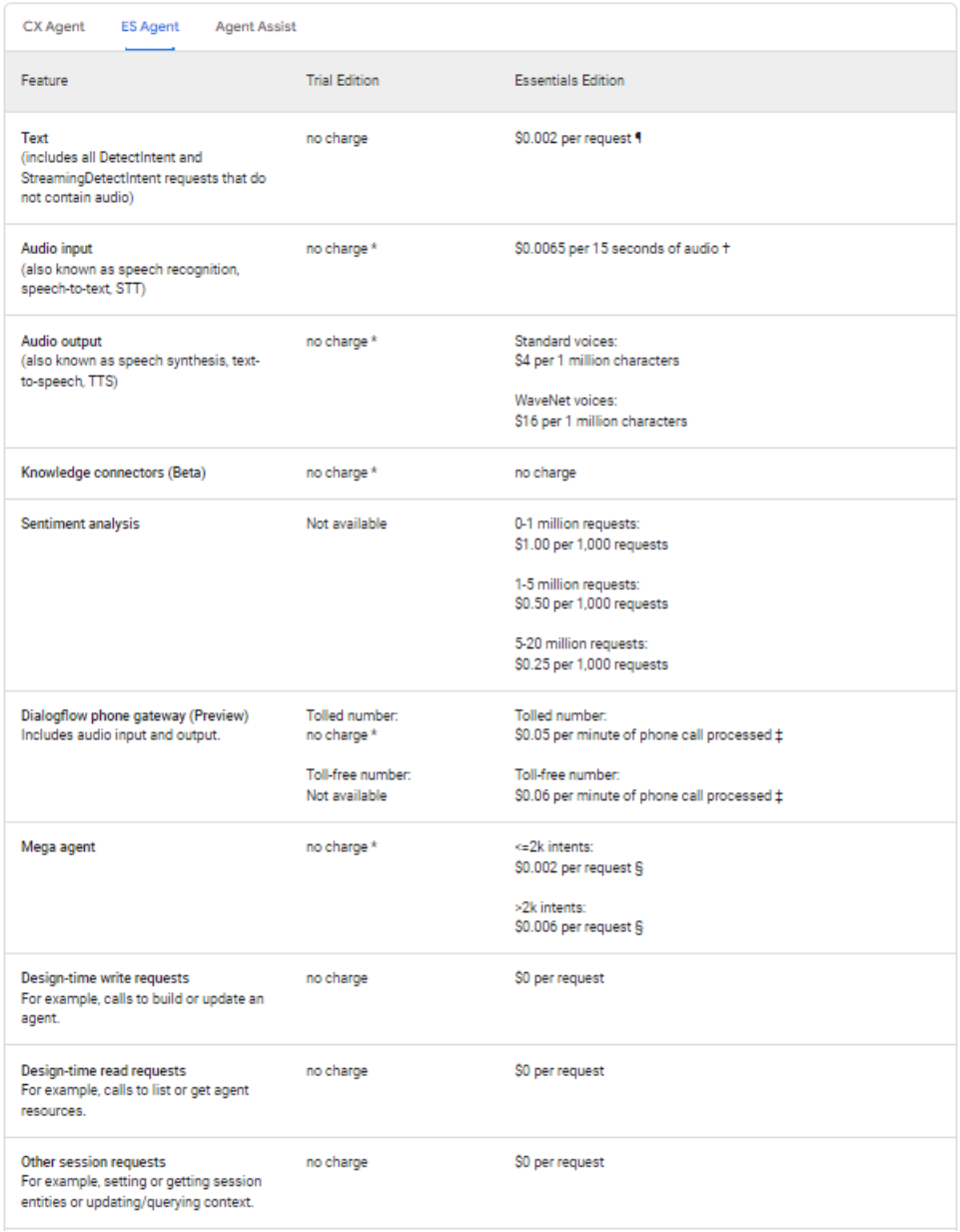
CX Agent | ES Agent | Agent Assist

Feature	CX Edition
Text (includes all DetectIntent, StreamingDetectIntent, and FulfillIntent requests that do not contain audio)	$0.007 per request
Audio input/output (speech recognition, speech-to-text, STT, speech synthesis, text-to-speech, TTS, telephony)	$0.001 per second *
Generative requests	See generative pricing
Design-time write requests For example, calls to build or update an agent.	no charge
Design-time read requests For example, calls to list or get agent resources.	no charge
Other session requests For example, setting or getting session entities.	no charge

* Each voice session is charged $0.001 per second of audio, with a minimum of one second. For example, a 15 second voice session is charged at $0.015, while a 61 second voice session is charged at $0.061. The previous minimum billed duration of one minute has been removed effective June 30, 2023. The total audio used for a voice session is the sum of both TTS and STT used by all requests and responses for the voice session. The total billed audio processing duration is independent of any no-charge, non-audio processing (API latency, webhook processing, and so on) which may occur before, during, or after audio processing. Voice session requests that make use of TTS or STT contribute to the total cost for the session, but these requests do not incur a per-request charge. If a voice session has any requests that do not use TTS or STT, these requests are charged per-request as defined above.

Slika 15: Troškovi CX agenta (snimka zaslona s [17])

ES agent ima dvije edicije troškova, a to su trial i plaćena edicija ES agenta. Trial edicija ne naplaćuje ništa ali ima ograničen broj zahtjeva. Plaćena edicija naplaćuje tekstualne poruke, odvojeno naplaćuje ulazne i izlazne glasovne poruke, analizu sentimenta i ostale stvari koje se mogu vidjeti na slici 16.



CX Agent | ES Agent | Agent Assist

Feature	Trial Edition	Essentials Edition
Text (includes all DetectIntent and StreamingDetectIntent requests that do not contain audio)	no charge	$0.002 per request ¶
Audio input (also known as speech recognition, speech-to-text, STT)	no charge *	$0.0065 per 15 seconds of audio †
Audio output (also known as speech synthesis, text-to-speech, TTS)	no charge *	Standard voices: $4 per 1 million characters WaveNet voices: $16 per 1 million characters
Knowledge connectors (Beta)	no charge *	no charge
Sentiment analysis	Not available	0-1 million requests: $1.00 per 1,000 requests 1-5 million requests: $0.50 per 1,000 requests 5-20 million requests: $0.25 per 1,000 requests
Dialogflow phone gateway (Preview) Includes audio input and output.	Tolled number: no charge * Toll-free number: Not available	Tolled number: $0.05 per minute of phone call processed ‡ Toll-free number: $0.06 per minute of phone call processed ‡
Mega agent	no charge *	<=2k intents: $0.002 per request § >2k intents: $0.006 per request §
Design-time write requests For example, calls to build or update an agent.	no charge	$0 per request
Design-time read requests For example, calls to list or get agent resources.	no charge	$0 per request
Other session requests For example, setting or getting session entities or updating/querying context.	no charge	$0 per request

Slika 16: Troškovi ES agenta (snimka zaslona s [17])

Dialogflow ES tehnologija pokazala se izuzetno jednostavnom i efikasnom za kreiranje i upogonjenje razgovornog robota. Proces kreiranja početne namjere bio je vrlo jednostavan i brz, što je potvrđeno kroz praktičan primjer kreiranja testne namjere. Autor je u minimalnom vremenu kreirao agenta i testirao njegovu funkcionalnost, potvrđujući jednostavnost i pristupačnost tehnologije. Kroz konzolu su testirani odgovori agenta, a rezultati su bili prikazani u jednostavnom i JSON formatu.

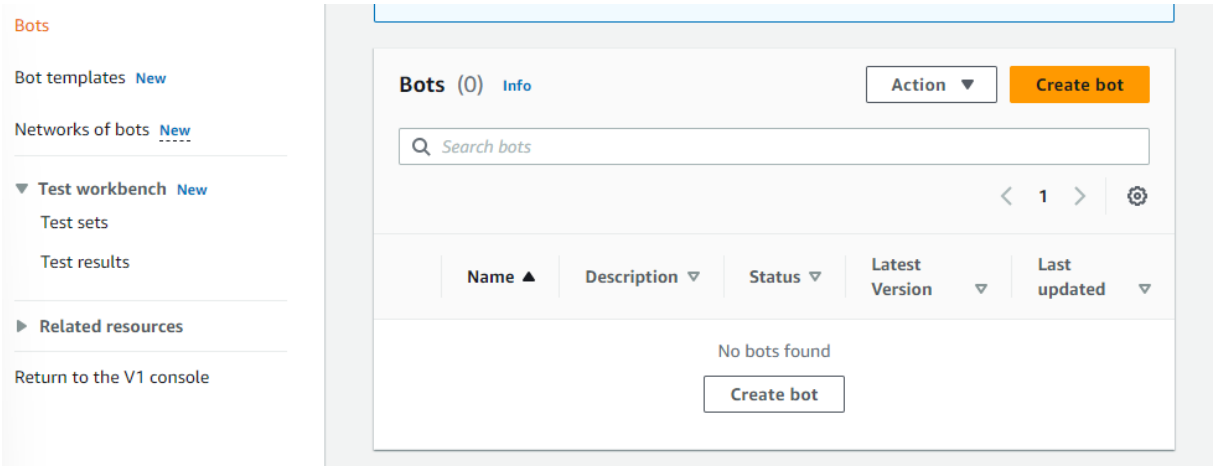
Tablica 1: Tablica ocjena Dialogflow ES tehnologije

Kriterij	Opis	Ocjena (1-10)
Jednostavnost korištenja	Dialogflow ES je jednostavan za korištenje, intuitivno sučelje omogućava brzo kreiranje i testiranje namjera.	10
Performanse	Iako performanse nisu direktno vidljive, agent brzo i točno odgovara, ali za detaljniju analizu potrebna je složenija implementacija.	10
Funkcionalnost	Različite funkcionalnosti su dostupne prema edicijama što omogućuje fleksibilnost prema potrebama projekta	10
Podrška	Dokumentacija je opsežna i strukturirana, pruža sve potrebne informacije za razvoj agenta i API integraciju.	10
Troškovi	Troškovi su jasno definirani za različite edicije, omogućavajući fleksibilnost prema potrebama projekta.	10

Prema tablici 1. možemo zaključiti kako je Dialogflow ES tehnologija vrlo pristupačna, jednostavna, u potpunosti dokumentirana tehnologija koja pruža mnoge funkcionalnosti. Plaćanje pojedinih funkcionalnosti daje fleksibilnost projektima da koriste točno one funkcionalnosti koje su im potrebne te samim time odredit i izračunat troškove za te funkcionalnosti.

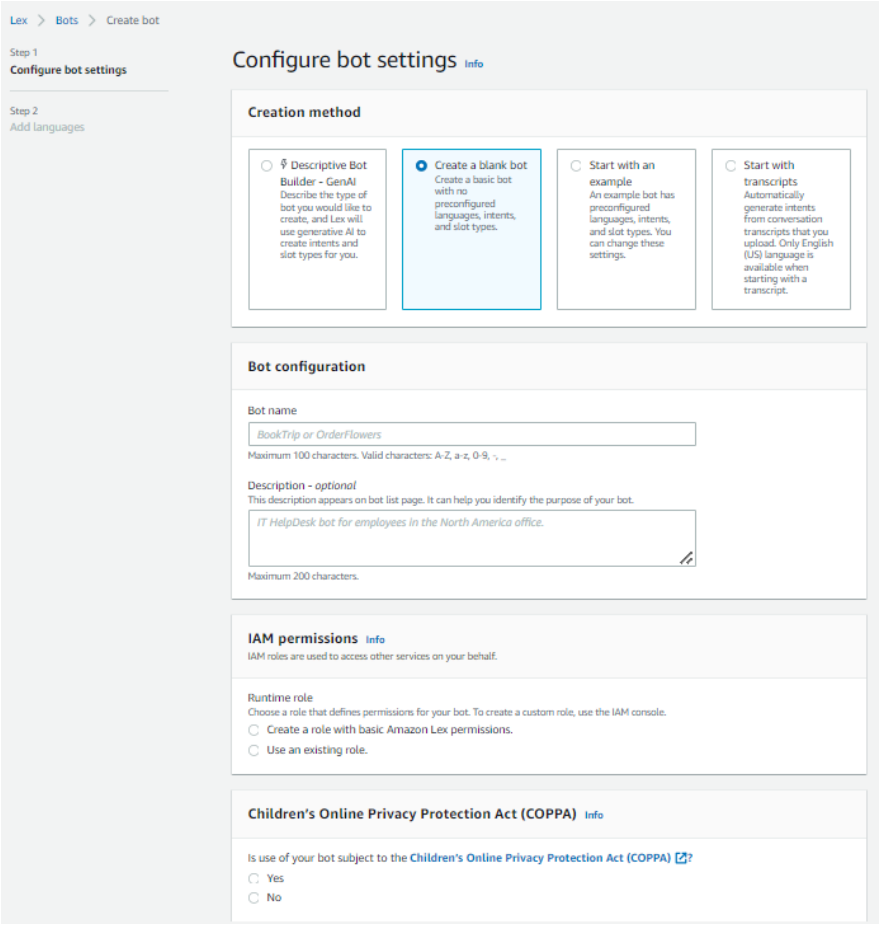
6.3. Amazon Lex

Za kreiranje besplatnog Amazon Lex razgovornog robota, autor se mora registrirati, dati podatke osobne kreditne kartice i ostale osobne podatke. Tek nakon toga se može kreirati Amazon Lex razgovorni robot. Kreiranje samog robota se započinje klikom na gumb Kreiraj robota što se može vidjeti na slici 17.

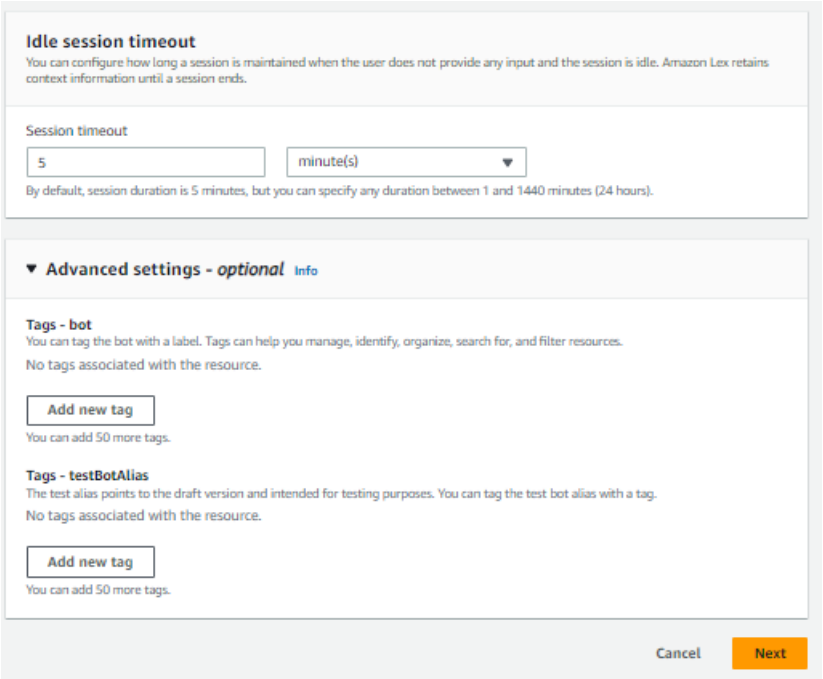


Slika 17: Kreiraj robota Amazon Lex (snimka zaslona)

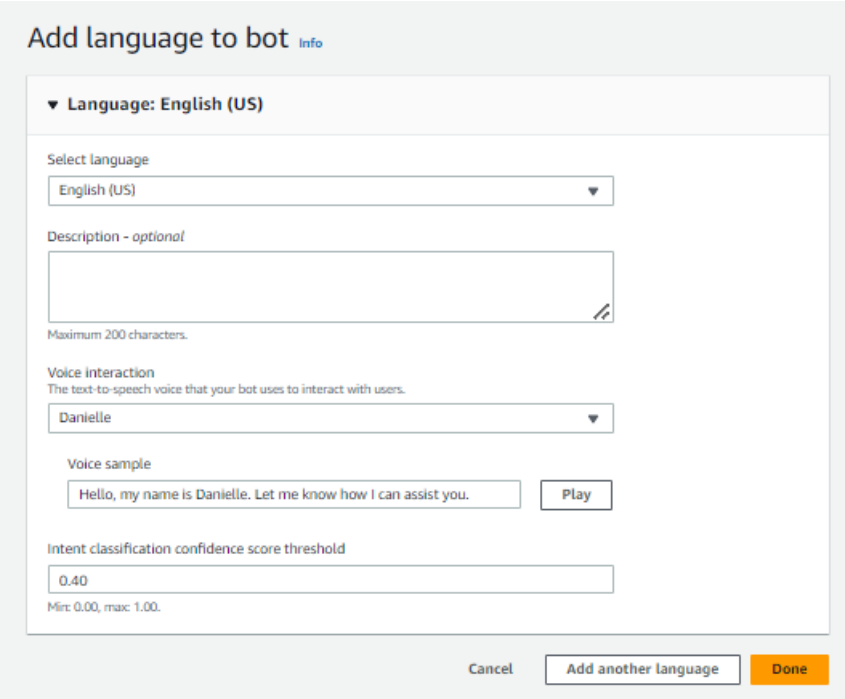
Sljedeći korak u kreiranju robota je određivanje načina izrade robota, određivanje imena robota, IAM dozvola (dozvole vezane uz Amazon Web servise), nakon koliko vremena odsutnosti se sesija prekida itd. Sve to se može vidjeti na slikama 18, 19, 20.



Slika 18: Postavke robota Amazon Lex - prvi dio (snimka zaslona)



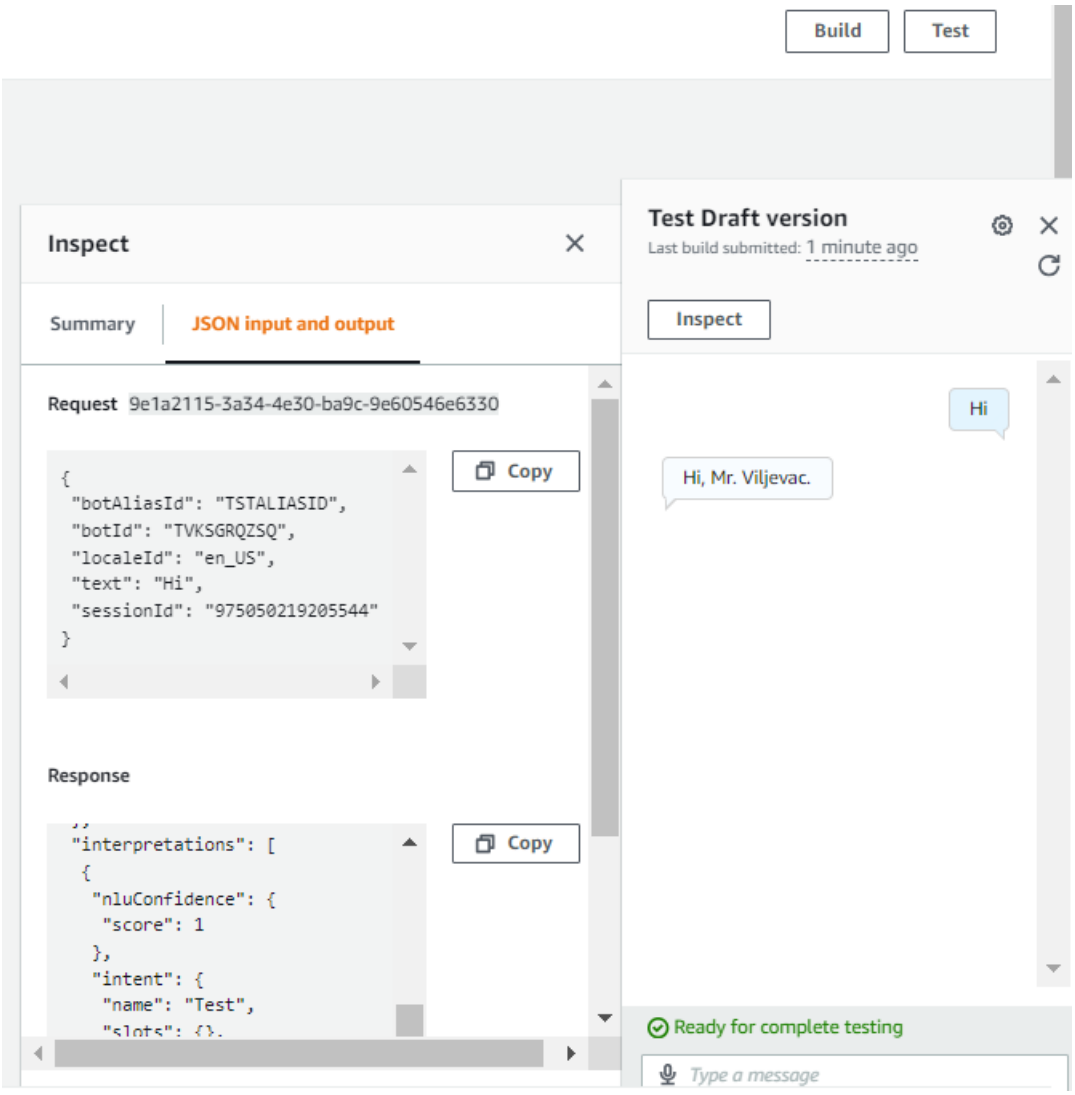
Slika 19: Postavke robota Amazon Lex - drugi dio (snimka zaslona)



Slika 20: Postavke robota Amazon Lex - treći dio (snimka zaslona)

Nakon uspješnog kreiranja robota autor dolazi na stranicu kreiranja namjera. Taj proces se može odviti kroz dva sučelja. Kroz vizualno sučelje tj. vizualni kreator ili kroz formu mijenjanja postavka (engl. editor). Kreiranje namjere ima više atributa nego Dialogflow ES, a

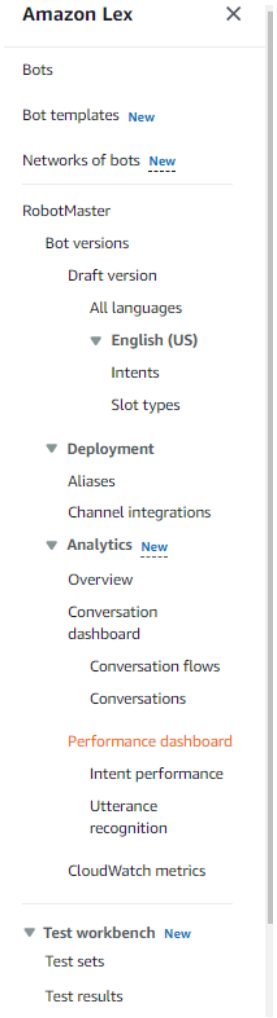
to su tok razgovora, detalji namjere (ime, opis), kontekst, uzorci fraza, inicijalni odgovor, mjesta (engl. slots), potvrda, ispunjenje, zatvarajući odgovor i web poveznici na razvijeni kod (engl. Code hooks/webhooks). Nekoliko od tih atributa imaju i dodatne postavke. Sve to vodi prema tome da se namjera može točnije definirati i pronaći prilikom razgovora s razgovornim robotom. Autor je kreirao namjeru istu kao i u Dialogflow-u gdje su fraze koje piše korisnik „Hi“ ili „Hello“, a odgovor je „Hello, Mrs. Viljevac“ Kako bi se testiralo robota, mora ga se prvo izgraditi kroz sučelje klikom na gumb „Build“ čija funkcija se izvršava na strani Amazon servisa. Nakon izgradnje, robot se može testirati klikom na gumb „Test“ gdje se dobiva prozor u obliku razgovora. Na slici 21. može se vidjeti testna poruka i odgovor.



Slika 21: Razgovor s testnim robotom Amazon Lex (snimka zaslona)

Za kreiranje Amazon Lex robota je trebalo nešto duže, no to je posljedica većeg broja atributa koji se moraju postaviti za kreiranje robota i namjera. Iako postoji veći broj atributa i

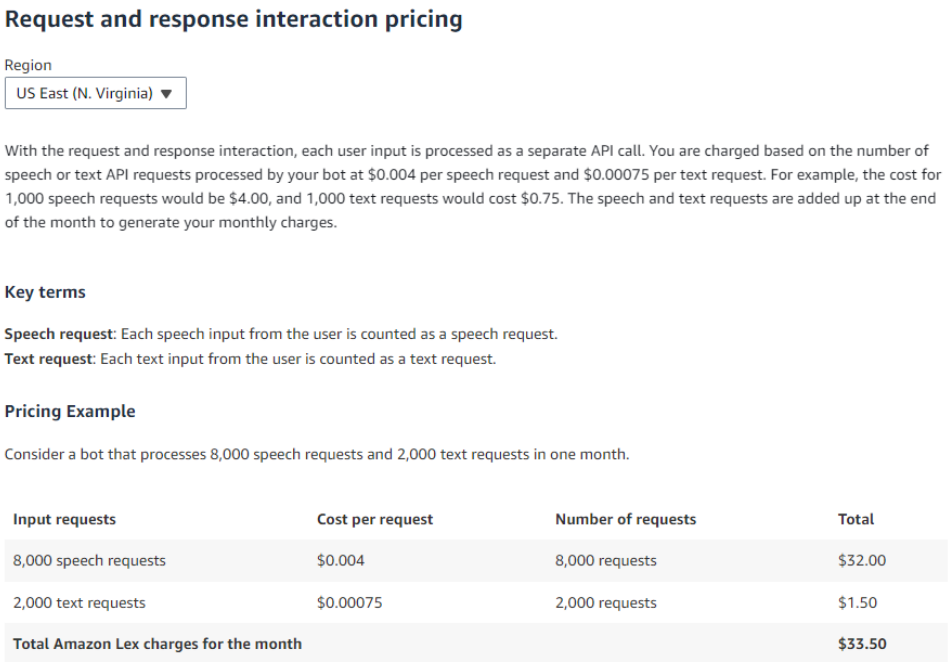
Amazon Lex kreiranje robota i namjera je malo složenije, Amazon Lex je vrlo jednostavan za korištenje. Performanse testnog robota se ne mogu vidjeti, već se mora kreirati aplikacija koja koristi robota, proslijediti atribut „sessionID“ te bi se tada dobile informacije o performansama tipa koliko je zahtjeva uspjelo ili ne, koliko se namjera zamijenilo drugom namjerom itd. Funkcionalnosti ovog robota su slične kao i kod Dialogflow ES robota no malo su drugačije strukturirane što možemo vidjeti na slici 22. Prema autorovom mišljenju struktura funkcionalnosti utječe na težinu korištenja, te je to posljedica da se na Amazon Lex robotu malo teže snalaziti unutar svih funkcionalnosti.



Slika 22: Funkcionalnosti Amazon Lex-a (snimka zaslona)

Amazon Lex podrška se temelji na dokumentaciji kao i Dialogflow ES podrška. Podrška Amazon Lex-a je detaljna i pruža sve informacije za kreiranje razgovornog robota te integraciju robota unutar aplikacije i korištenje API poziva. Svakako, dok je dokumentacija detaljna, u i isto vrijeme nije toliko lako pronaći sve potrebne detalje jer je drugačije strukturirana. Prema mišljenju autora, korištenje dokumentacija po prvi puta je dosta kompliciranо, no ako ste

razvojni inženjer s iskustvom onda će te se snaći unutar takve dokumentacije. Troškovi plaćene edicije Amazon Lex-a se mogu vidjeti na sljedećim slikama. Amazon Lex nudi i besplatnu edicije s kojom se može procesirati 10000 tekstualni poruka i 5000 glasovnih intervala u mjesecu u prvih godinu dana.



Request and response interaction pricing

Region

US East (N. Virginia)

With the request and response interaction, each user input is processed as a separate API call. You are charged based on the number of speech or text API requests processed by your bot at $0.004 per speech request and $0.00075 per text request. For example, the cost for 1,000 speech requests would be $4.00, and 1,000 text requests would cost $0.75. The speech and text requests are added up at the end of the month to generate your monthly charges.

Key terms

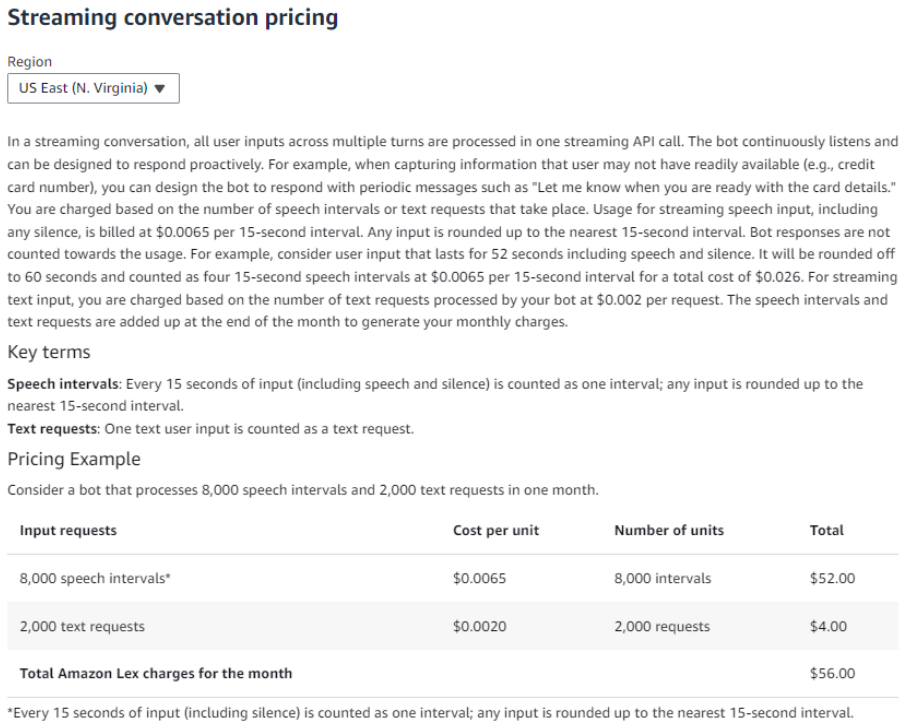
Speech request: Each speech input from the user is counted as a speech request.
Text request: Each text input from the user is counted as a text request.

Pricing Example

Consider a bot that processes 8,000 speech requests and 2,000 text requests in one month.

Input requests	Cost per request	Number of requests	Total
8,000 speech requests	$0.004	8,000 requests	$32.00
2,000 text requests	$0.00075	2,000 requests	$1.50
Total Amazon Lex charges for the month			**$33.50**

Slika 23: Troškovi zahtjeva i odgovora Amazon Lex-a (snimka zaslona s [18])



Streaming conversation pricing

Region

US East (N. Virginia)

In a streaming conversation, all user inputs across multiple turns are processed in one streaming API call. The bot continuously listens and can be designed to respond proactively. For example, when capturing information that user may not have readily available (e.g., credit card number), you can design the bot to respond with periodic messages such as "Let me know when you are ready with the card details." You are charged based on the number of speech intervals or text requests that take place. Usage for streaming speech input, including any silence, is billed at $0.0065 per 15-second interval. Any input is rounded up to the nearest 15-second interval. Bot responses are not counted towards the usage. For example, consider user input that lasts for 52 seconds including speech and silence. It will be rounded off to 60 seconds and counted as four 15-second speech intervals at $0.0065 per 15-second interval for a total cost of $0.026. For streaming text input, you are charged based on the number of text requests processed by your bot at $0.002 per request. The speech intervals and text requests are added up at the end of the month to generate your monthly charges.

Key terms

Speech intervals: Every 15 seconds of input (including speech and silence) is counted as one interval; any input is rounded up to the nearest 15-second interval.
Text requests: One text user input is counted as a text request.

Pricing Example

Consider a bot that processes 8,000 speech intervals and 2,000 text requests in one month.

Input requests	Cost per unit	Number of units	Total
8,000 speech intervals*	$0.0065	8,000 intervals	$52.00
2,000 text requests	$0.0020	2,000 requests	$4.00
Total Amazon Lex charges for the month			$56.00

*Every 15 seconds of input (including silence) is counted as one interval; any input is rounded up to the nearest 15-second interval.

Slika 24: Troškovi strujanja razgovora (snimka zaslona s [18])

Amazon Lex tehnologija pokazala se kao moćan alat za kreiranje i upogonjenje razgovornog robota, iako je proces nešto složeniji u usporedbi s Dialogflow ES. Registracija i kreiranje robota zahtijevaju osobne podatke i postavljanje načina plaćanja, ali nakon toga je postupak relativno jednostavan. Kreiranje namjera u Amazon Lex-u omogućava preciznije definiranje namjera zahvaljujući većem broju atributa, što može dodatno poboljšati točnost prepoznavanja korisničkih unosa.

Tablica 2: Tablica ocjena Amazon Lex tehnologije

Kriterij	Opis	Ocjena (1-10)
Jednostavnost korištenja	Amazon Lex je jednostavan za korištenje, iako zahtijeva više vremena zbog većeg broja atributa i postavki.	9
Performanse	Performanse nisu direktno vidljive, ali agent brzo i točno odgovara; za detaljniju analizu potrebna je složenija implementacija.	10
Funkcionalnost	Bogat skup funkcionalnosti omogućuje fleksibilnost i preciznost u definiranju namjera i tokova razgovora.	10
Podrška	Dokumentacija je detaljna i opsežna, no struktura može otežati snalaženje za nove korisnike.	10
Troškovi	Troškovi su jasno definirani s opcijom besplatne edicije.	9

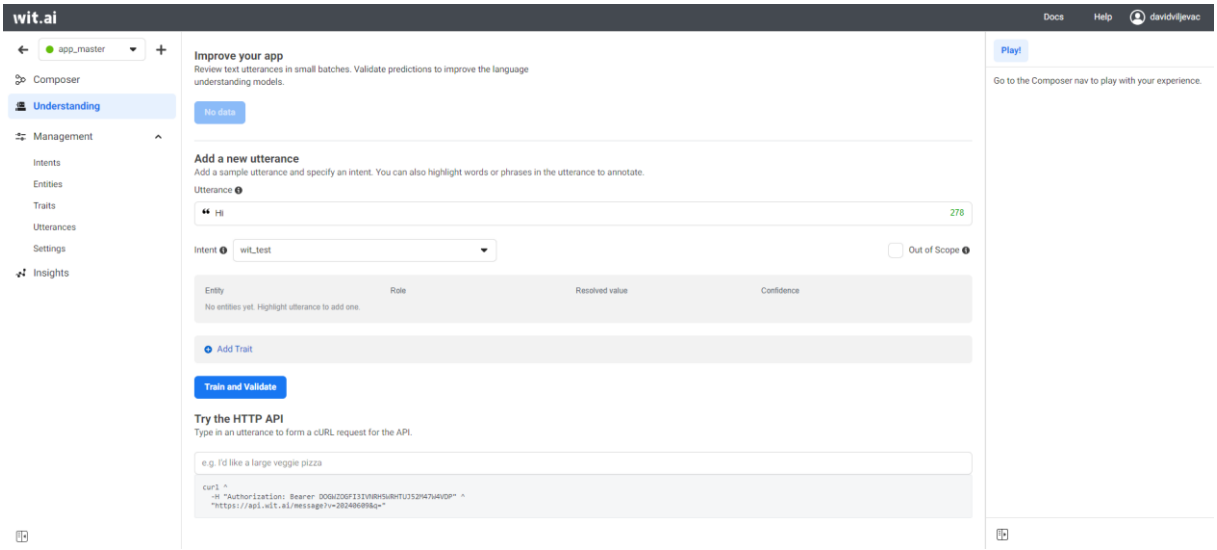
Amazon Lex je moćna tehnologija za razvoj razgovornih robota s bogatim skupom funkcionalnosti i fleksibilnošću koja omogućava precizno definiranje i upravljanje razgovorima. Iako je proces kreiranja robota i namjera nešto složeniji, rezultirajuća funkcionalnost i preciznost opravdavaju uloženo vrijeme. Dokumentacija je detaljna, ali njezina struktura može otežati snalaženje za nove korisnike. Troškovi su jasno definirani, a besplatna edicija omogućuje početno testiranje bez dodatnih troškova. Sveukupno, Amazon Lex je robustan alat koji može zadovoljiti potrebe različitih projekata, od jednostavnih testnih namjera do složenih razgovornih sustava.

6.4. Wit.ai

Za kreiranje razgovornog robota na platformi Wit.ai potrebno je napraviti novi korisnički račun koji je povezan s Meta platformom. Nakon kreiranja korisničkog računa odmah možemo početi s kreiranjem razgovornog robota. Na slici 25. se vidi kako izgleda sučelje koje se pojavi nakon registracije.

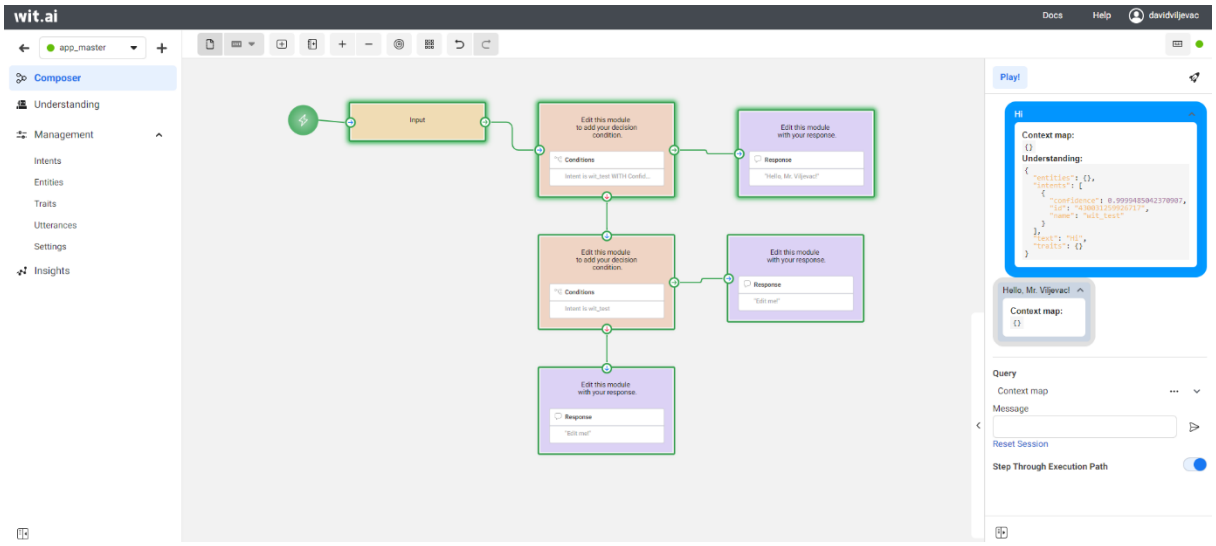
Slika 25: Wit.ai sučelje za kreiranje aplikacije (snimka zaslona)

Klikom na gumb „New App“ dobivamo prozor na kojem možemo urediti ime aplikacije, njezin jezik te vidljivost koja može biti javna ili privatna. Nakon kreiranja aplikacije dobiva se novi prozor gdje se može krenuti s kreiranjem namjera, fraza i svega ostalog potrebnog za razgovornog robota. Na slici 26. može se vidjeti sučelje tog prozora.



Slika 26: Wit.ai sučelje za kreiranje robota (snimka zaslona)

Odmah se može vidjeti kako je puno manje mogućnosti i funkcionalnosti u usporedbi s Dialogflow ES i Amazon Lex-om. U aplikaciji se kreirala testna namjera, koja ima puno manje atributa za promijenit/dodat u odnosu na namjere u druge dvije tehnologije. Nakon kreiranja testne namjere kojoj je autor pridodao samo naziv i fraze pri kojoj se ta namjera okida, autor je morao otići i vizualnog kreatora u kojem je trebao postaviti tok razgovora. Na slici 27. može se vidjeti testni tok razgovora s odgovorima i testnom namjerom.



Slika 27: Wit.ai kreirani tok razgovora i test s desne strane (snimka zaslona)

Prema autoru je trebalo puno više vremena da se prilagodi na sučelje makar je vizualno jednostavno. Kreiranje toka razgovora isto nije trivijalno pri prvom korištenju. Namjere se kreiraju kroz jednu vrstu sučelja dok se tok razgovora, odnosno povezivanje odgovora s namjerom radi kroz vizualno sučelje tj. „Composer“ u Wit.ai aplikaciji. Također, vrijeme čekanja na odgovor je puno duže u odnosu na prije spomenute tehnologije. Performanse i ostala statistika se može vidjeti klikom na gumb „Insights“ gdje je vođena statistika o svim namjerama i njihovim atributima, no i dalje nije toliko detaljno. Sva podrška koju Wit.ai pruža je lako dostupna na njihovoj stranici, iako kratka prema autorovom mišljenju je dovoljna za kreiranje robota. Za sve što pruža Wit.ai on ne obvezuje korisnike na bilo kakva plaćanja tj. u potpunosti je besplatan i za javne i za privatne edicije robota.

Wit.ai tehnologija omogućava jednostavno kreiranje razgovornih robota putem povezivanja s Meta platformom. Nakon registracije i kreiranja korisničkog računa, korisnici mogu odmah započeti s kreiranjem razgovornih robota. Iako Wit.ai nudi manje funkcionalnosti u usporedbi s Dialogflow ES i Amazon Lex-om, njegova jednostavnost i besplatna dostupnost čine ga pristupačnim alatom za osnovne potrebe.Tablica 3. daje tablični prikaz ocjena Wit.ai tehnologije u kojoj se mogu vidjeti kriteriji te razlog ocjene kriterija.

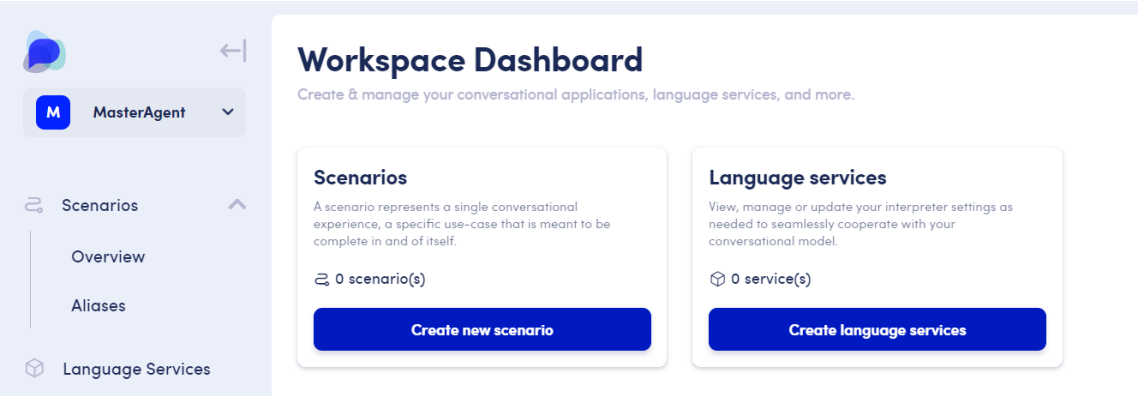
Tablica 3: Tablica ocjena Wit.ai tehnologije

Kriterij	Opis	Ocjena (1-10)
Jednostavnost korištenja	Sučelje je jednostavno, ali prilagodba može potrajati zbog odvojenih sučelja za namjere i tokove razgovora.	7
Performanse	Vrijeme čekanja na odgovor je dulje u usporedbi s drugim tehnologijama, no odgovori su točni.	6
Funkcionalnost	Manje funkcionalnosti u odnosu na Dialogflow ES i Amazon Lex, ali dovoljno za osnovne potrebe.	6
Podrška	Dokumentacija je kratka, ali dovoljna za osnovno kreiranje robota; dostupna na web stranici.	8
Troškovi	Wit.ai je potpuno besplatan za korištenje, bez obveza plaćanja, što je velika prednost.	BESPLATNO

Wit.ai je jednostavna tehnologija za kreiranje osnovnih razgovornih robota, s jednostavnim sučeljem i besplatnim pristupom svim funkcionalnostima. Iako nudi manje mogućnosti i dulje vrijeme čekanja na odgovore u usporedbi s Dialogflow ES i Amazon Lex-om, Wit.ai je dovoljno robustan za manje složene projekte. Dokumentacija je kratka, ali dovoljna za osnovno kreiranje robota, dok besplatna dostupnost bez ikakvih troškova čini Wit.ai vrlo privlačnom opcijom za početne i ograničene projekte. Sveukupno, Wit.ai je jednostavan i besplatan alat koji može zadovoljiti osnovne potrebe za kreiranje razgovornih robota, dok bi složeniji projekti mogli zahtijevati naprednije tehnologije.

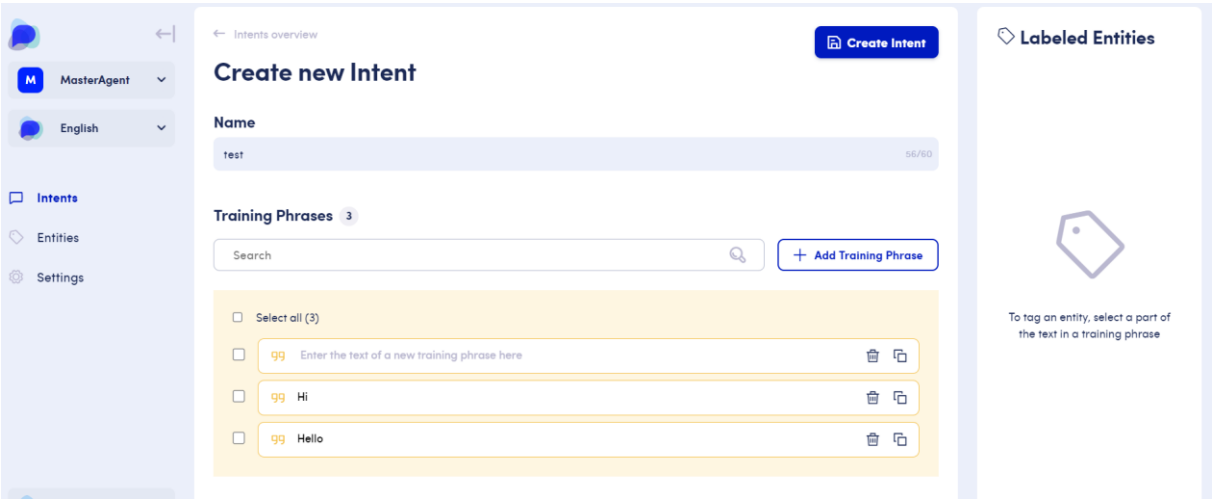
6.5. OpenDialog

Za OpenDialog platformu autor se morao registrirati i tokom registracije kreirati svoju prvu radni prostor (engl. workspace). Nakon kreiranja radnog prostora dobiva se sučelje koje je vrlo jednostavno gdje se mogu kreirati servisi jezika i servisi scenarija. Servisi jezika se koriste unutar servisa scenarija, a cijelo sučelje se vidi na slici 28.



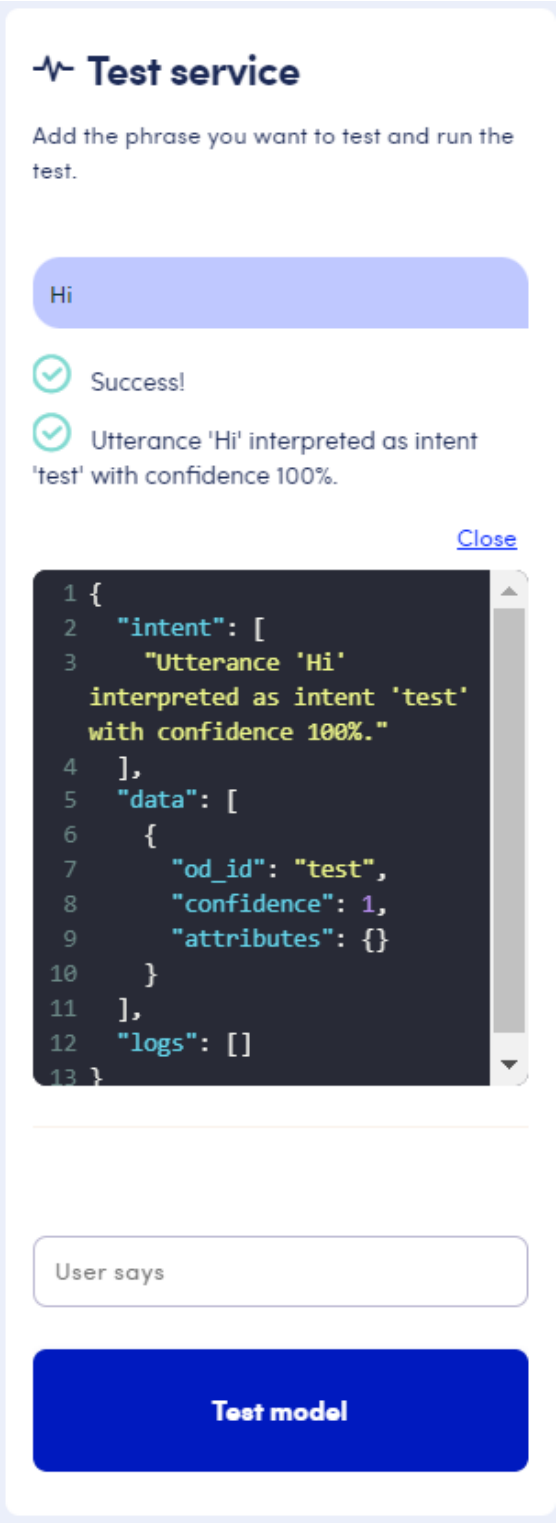
Slika 28: OpenDialog sučelje za kreiranje servisa (snimka zaslona)

U servisima jezika mogu se nadalje kreirati namjere. Kreiranje namjera je vrlo jednostavno i sastoji se od korak gdje se samo mora dodati fraza treninga i ime namjere što se može vidjeti na slici 29.



Slika 29: OpenDialog sučelje kreiranja namjera (snimka zaslona)

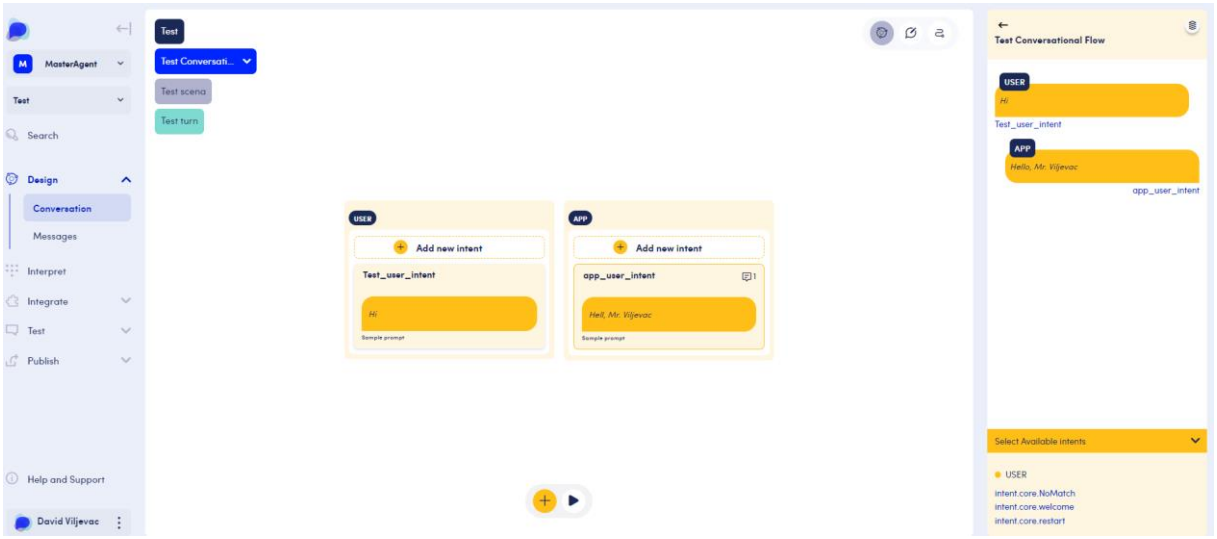
Nakon kreirane namjere, namjera se može istestirati te se povratno dobi informacija o namjeri koja se podudara i s kolikom je uvjerenošću to podudaranje. Rezultati testa se mogu vidjeti na slici 30.



Slika 30: OpenDialog testiranje namjere unutar servisa jezika (snimka zaslona)

Namjere se također mogu kreirati unutar servisa scenarija no te namjere su odvojene od namjera servisa jezika. Prvi korak je kreiranje projekta scenarija nakon čega se dobiva vizualno sučelje kroz koje se mogu kreirati razgovori povezani na taj projekt, nakon toga se

kreiraju scene unutar razgovora i preusmjerenja unutar scena odnosno pitanja i odgovori. Prema autorovom mišljenju sučelje je zbunjujuće na početku no i dalje je jednostavnije od kreiranja toka razgovora unutar Wit.ai aplikacije. Testiranje namjera se može provesti kroz testni prozor te se to sve može vidjeti na slici 31.



Slika 31: OpenDialog test namjera scenarija (snimka zaslona)

Također isto to se može testirati kroz prozor razgovora koji tada daje iste rezultate te povezuje sve razgovore kreirane unutar scena kako bi se sve mogle testirati od jedanput. Odgovori robota su prema autoru zadovoljavajuće brzi. OpenDialog ima samo jedan graf koji govori o tome koliko je korisnika nekog projekta i koliko dolazi zahtjeva. Uz to pruža informacije tj. zapise o interakcijama korisnika. Funkcionalnosti su bazične te ne pruža mnogo mogućnosti. Pruža mogućnost integracije, testiranja, dizajna razgovora te mogućnost promjene interpretera. OpenDialog je u potpunosti besplatan. Dokumentacija OpenDialoga je opširna i dovoljna za izradu razgovornog bota. Ako se dokumentacija slijedi korak po korak onda nije potrebno nikakvo pomagalo treće strane poput ChatGPT-a.

Tablica 4: Tablica ocjena OpenDialog tehnologije

Kriterij	Opis	Ocjena (1-10)
Jednostavnost korištenja	Sučelje je relativno jednostavno, ali može biti zbunjujuće na početku; proces kreiranja namjera i scenarija je intuitivan nakon prilagodbe.	7

Performanse	Vrijeme odgovora je zadovoljavajuće brzo; performanse su generalno dobre za osnovne potrebe.	5
Funkcionalnost	Bazične funkcionalnosti koje omogućuju integraciju, testiranje i dizajn razgovora mogućnosti u usporedbi s naprednijim platformama.	4
Podrška	Dokumentacija je opširna i dovoljna za izradu razgovornog bota; detaljne upute omogućuju jednostavno praćenje koraka.	7
Troškovi	OpenDialog je potpuno besplatan za korištenje, što je velika prednost.	BESPLATNO

OpenDialog je pristupačna platforma za kreiranje razgovornih robota s jednostavnim sučeljem i besplatnim pristupom svim funkcionalnostima. Iako sučelje može biti zbunjujuće na početku, proces kreiranja namjera i scenarija postaje intuitivan nakon prilagodbe. Performanse su zadovoljavajuće za osnovne potrebe, dok dokumentacija pruža sve potrebne informacije za razvoj robota. Funkcionalnosti su bazične, ali dovoljne za osnovne projekte, dok besplatna dostupnost čini OpenDialog vrlo privlačnom opcijom za projekte s ograničenim proračunom. Sveukupno, OpenDialog je jednostavna i besplatna platforma koja može zadovoljiti osnovne potrebe za kreiranje razgovornih robota, dok bi složeniji projekti mogli zahtijevati naprednije platforme.

6.6. Zaključak usporedbe

Zaključak usporedbe će se provesti kroz tablicu ocjena određenih kriterija. Prema autorovom mišljenju nije potrebno raditi anketu jer u trenutku pisanja autor ima radnog iskustva te je dovoljno upoznat sa svim kriterijima.

Sljedeća tablica prikazuje ocjene svih karakteristika pojedinog alata kako bi se moglo vizualno prikazati usporedba svih alata. Ocjene su objektivne od strane autora. Ocjene mogu biti od jedan (najlošije) do deset (najbolje). Ocjene troškova su u istom rangu no odnose se na omjer cijene i količine zahtjeva, interakcija, sekunda glasovne obrade itd.

Tablica 5: Usporedba tehnologija prema određenim kriterijima

	Jednostavnost korištenja	Performanse	Funkcionalnost	Podrška	Troškovi
Dialogflow ES	10	10	10	10	10
Amazon Lex	9	10	10	10	9
Wit.ai	7	6	6	8	BESPLATNO
OpenDialog	7	5	4	7	BESPLATNO

Iz tablice 1. može se vidjeti kako Dialogflow ES i Amazon Lex postavljaju visoke standarde. Besplatni alati su općenito slabije razrađeni te isti taj primjer možemo vidjeti i ovdje. Prema mišljenju autora, odabir tehnologija bi se trebao svesti na Dialogflow ES ili Dialogflow CX inačicu koja je spomenuta kroz rad te Amazon Lex servis jer oba dvije tehnologije/servisi imaju integraciju s drugim tehnologijama te pružaju veliku mogućnost povezivanja tih tehnologija.

Sumiranjem ocjena dobivamo sljedeće rezultate:

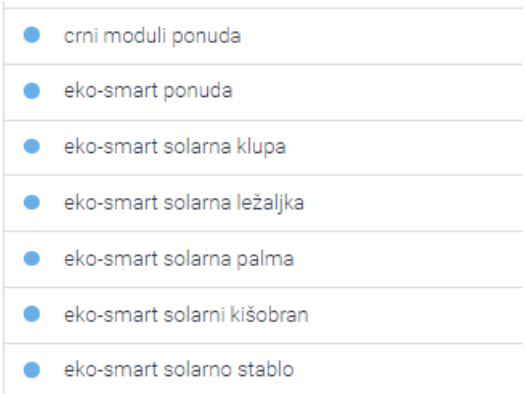
1. Dialogflow ES – 10
2. Amazon Lex – 9.6
3. Wit.ai – 6.75 (ne uračunavajući troškove)
4. OpenDialog – 5 (ne uračunavajući troškove)

Prema sumiranim rezultatima, jasno je da Dialogflow ES i Amazon Lex značajno nadmašuju besplatne alate u pogledu funkcionalnosti i integracijskih mogućnosti. Ove dvije tehnologije pružaju alate za razvoj naprednih razgovornih robota, omogućujući visoku razinu povezivanja s drugim sustavima. Stoga, preporuka autora je fokusirati se na ove dvije tehnologije pri izboru tehnologije za razvoj razgovornih robota, jer nude optimalnu kombinaciju performansi, fleksibilnosti i mogućnosti integracije.

7. Razvoj prototipa razgovornog robota za moderni programski proizvod

U ovom poglavlju detaljno će se objasniti proces razvoja prototipa razgovornog robota za stranicu Solvis, koji pruža savjetovanje tj. informacije o njihovoj tvrtki i proizvodima koji se mogu pronaći na njihovoj internet stranici. Prototip je razvijen korištenjem Dialogflow ES tehnologije za kreiranje agenta, Node.js [8]za servisni dio aplikacije (engl. backend) i Vue.js [9]za vizualni dio aplikacije (engl. frontend).

Prvi korak razvoja prototipa je razvoj agenta kroz Dialogflow ES tehnologiju. Prvo se kreirao agent te su se odabrale osnovne postavke. Nakon kreiranja samog agenta krenulo se na kreiranje namjera. Namjere su definirane temeljem informacija na web stranici Solvis. Definirane su na način da je jedna namjera odgovorna za jedan proizvod ili kategoriju na toj stranici. Također unutar tih namjera su uključene i namjere kontakta, misije i vizije tvrtke Solvis itd. Jedan dio definiranih namjera se može vidjeti na slici 32.

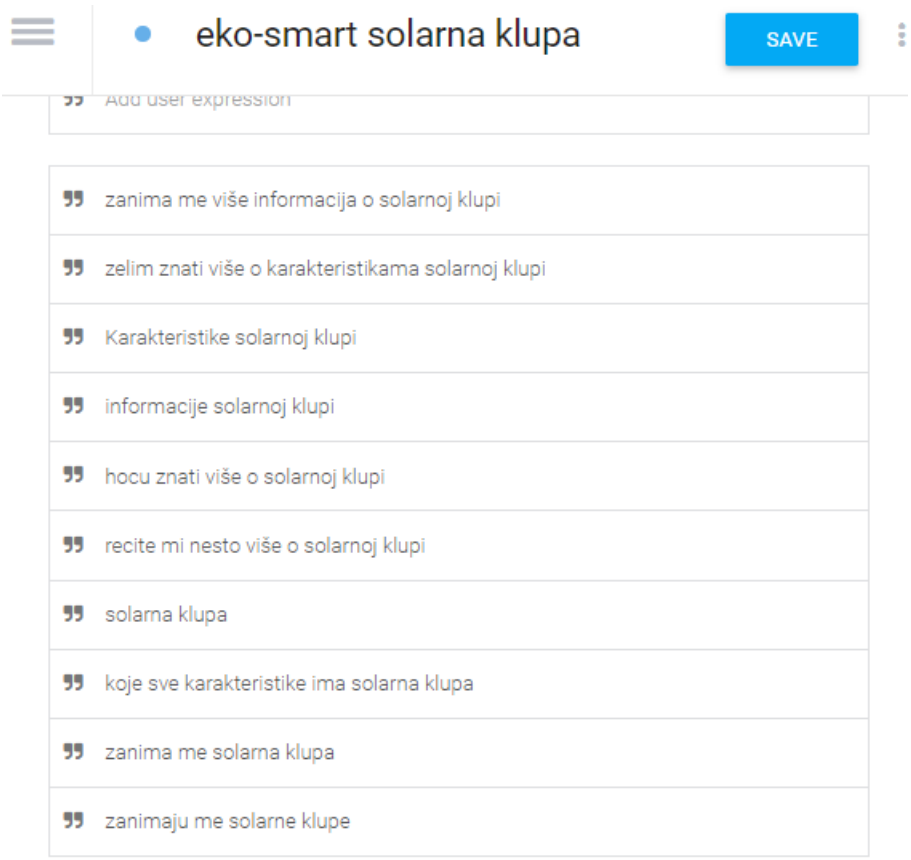


Slika 32: Primjeri definiranih namjera (snimka zaslona)

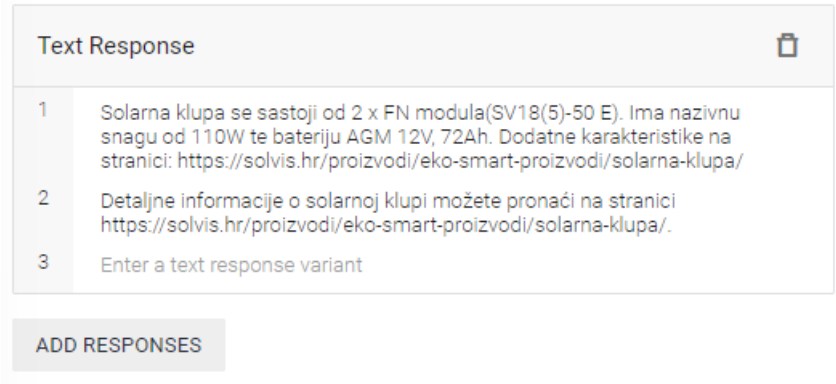
Kako je napisano u teorijskom dijelu, za izradu namjere su potrebne barem nekoliko trening tj. korisnički izrazi te tekstualni odgovor. Prema tom principu su se definirale i namjere prototipa agenta za tvrtku Solvis. Definiciju namjere *eko-smart solarna klupa* možemo vidjeti na slici 33. i 34.

[8] Node.js je otvorena platforma za izvršavanje JavaScript koda na strani poslužitelja. Razvio ga je Ryan Dahl 2009. godine, a trenutno ga održava Node.js Foundation. Više informacija dostupno je na https://nodejs.org/en

[9] Vue.js je progresivni JavaScript framework za izgradnju korisničkih sučelja. Razvio ga je Evan You 2014. godine. Vue.js je poznat po svojoj jednostavnosti i lakoći integracije s drugim projektima i bibliotekama. Pruža robustan ekosistem za izgradnju složenih jedno-straničnih aplikacija. Više informacija na https://vuejs.org/



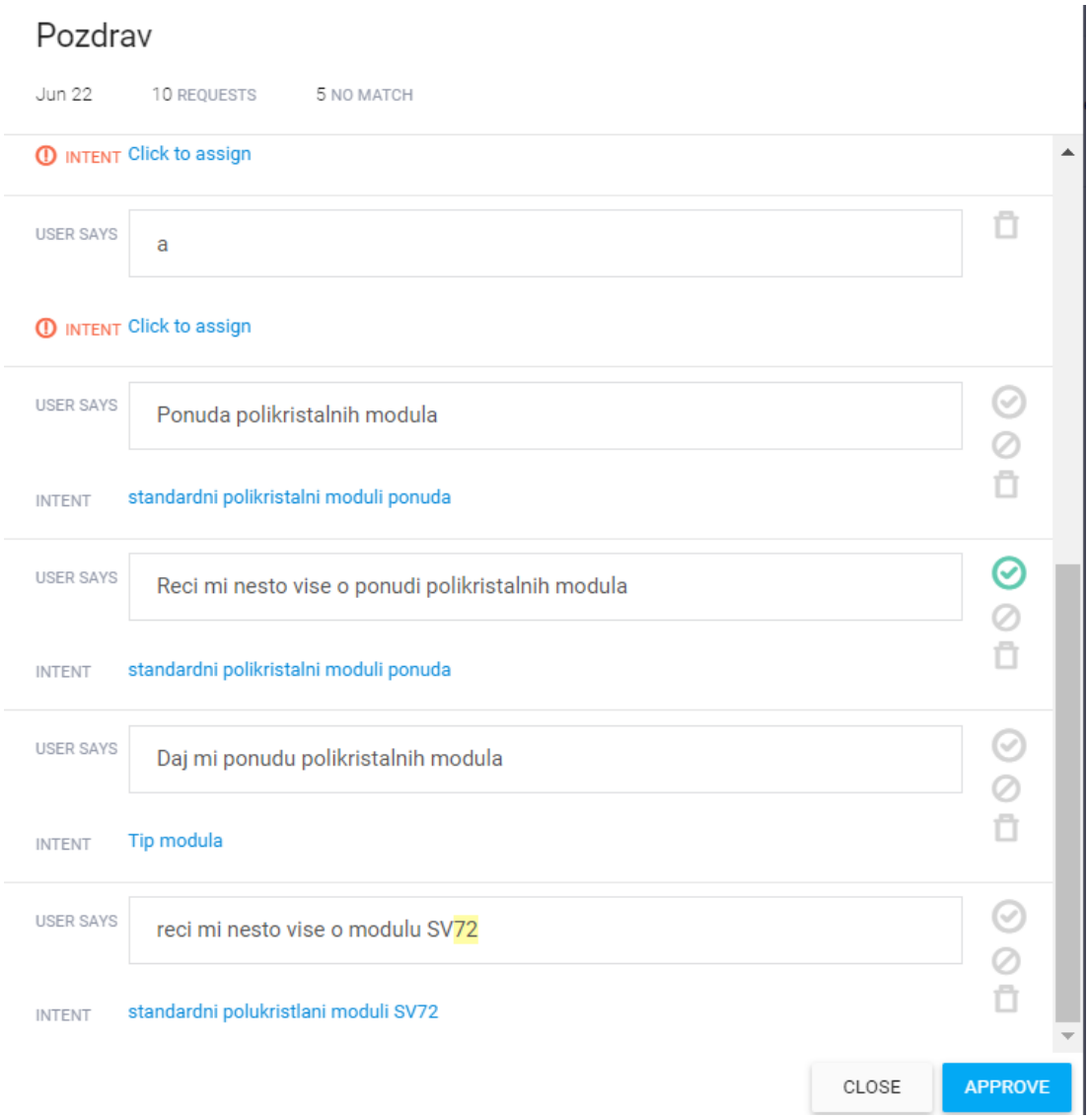
Slika 33: Definicija korisničkih izraza za eko-smart solarnu klupu



Slika 34: Definicija odgovora za eko-smart solarnu klupu

Svaka namjera je posebno testirana direktno unutar Dialogflow ES tehnologije kroz konzolu. Nakon testiranja odnosno upisa upita u konzolu, upit i odgovor se zapisuju unutar *Training* odjeljka Dialogflow ES tehnologije. Tu se mogu vidjeti sesije razgovora te se ručno može istrenirati robot dodatnim validacijama odgovora. Ako razvojni inženjer nije zadovoljan s

odgovorom odnosno podudarnom namjerom, on ju može promijeniti i validirati tu promjenu. Tim procesom se zapisuju dodatni korisnički izrazi unutar novo odabrane namjere te se smanjuje količina potencijalnih grešaka pri podudaranju. Na slici 35. se može vidjeti jedna takva sesija pri kojoj je autor promijenio namjeru te je tada taj korisnički unos dobio zelenu kvačicu što označuje da je validiran.



Slika 35: Sesija unosa i odgovora s validacijom

Nakon kreiranja agenta prelazi se na kreiranje servisa koji će koristiti tog agenta za generiranje odgovora na temelju korisničkog unosa te vizualnog dijela razgovornog robota. Za

verzioniranje takvog projekt korištena je platforma Github[10]. Na Github-u se kreirao projekt (repozitorij) koji je trenutno dostupan na: https://github.com/David-Viljevac/chat-bot-solar-panels. Unutar Visual Studio Code-a[11](u daljnjem tekstu VS Code) klonirao se repozitorij te se krenulo s radom na servisnom dijelu aplikacije. Za kreiranje servisnog dijela aplikacije koristili su se različite biblioteke koje su se instalirale kroz *terminal* s komandom *npm install *package**. Sve potrebne biblioteke za rad servisa se mogu vidjeti unutar projekta tj. datoteke *package.json.* Korištene biblioteke:

- dotenv – za korištenje *.env* datoteke koja se može mjenjati ovisno o okolini. Kada se sličan proizvod proizvodi unutar neke tvrtke, najčešće postoje tri okoline. *Develop*, *Stage, Production* okolina. Svaka okolina koristi drugačije *.env* datoteke kako bi se moglo ograničiti i prilagoditi određeni atributi u odnosu na implementirani servis.
- express – biblioteka za razvoj servisa tj. da servis sadrži pristupne točke kojima se može pristupi s neke druge internet lokacije, a pritom da se dobije odgovor sa servisa.
- nodemon – za lakše korištenje i ponovo pokretanje servisa nakon spremljenih promjena unutar VS Code-a
- dialogflow – za implementaciju razvijenog agenta unutar servisne arhitekture.

Unutar servisnog dijela aplikacije kreirale su se dvije pristupne točke. Jedna pristupna točka je za generiranje odgovora temeljem korisničkog unosa, a druga je za testiranje stanja servisa. Za ovakav prototip su dovoljne samo te dvije pristupne točke. Te pristupne točke se mogu vidjeti u programskog kodu 2. i 3.

```
server.post('/api/chat', async (req, res) => {
    let { queryText, sessionId } = req.body;
    let result = await detectIntent(queryText, sessionId);
    res.status(200).send(result);
});
```

Programski kod 2: Pristupna točka za generiranje odgovora

[10] GitHub je platforma za verzioniranje i suradnju na kodu. Osnovali su je Tom Preston-Werner, Chris Wanstrath, P. J. Hyett i Scott Chacon 2008. godine. GitHub omogućava programerima da zajedno rade na projektima, pregledaju i integriraju kod te prate promjene u verzijama. Internet stranica: https://github.com/

[11] Visual Studio Code (VS Code) je besplatan, open-source kodni editor razvijen od strane Microsofta. Internet stranica: https://code.visualstudio.com/

```
server.get('/health-check', (req, res) => {
    res.status(200).send('Successfully passed the health check for chat
bot');
});
```

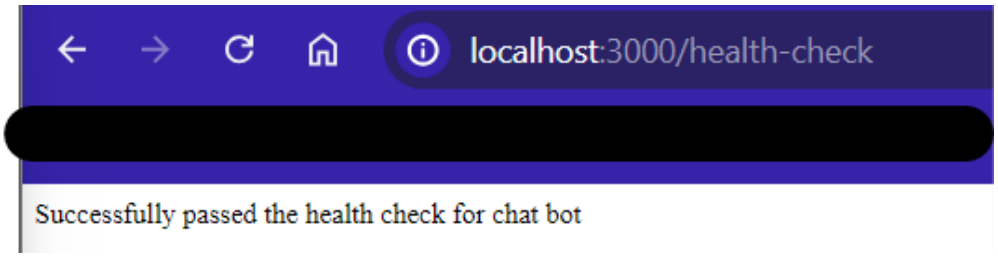
Programski kod 3: Pristupna točka za testiranje stanja servisa

Za povezivanje Dialogflow agenta potrebno je bilo kreirati poslužiteljski profil (engl. service account) kojem su se dala potrebna dopuštenja (engl. permissions) za pristup na cjelokupni projekt agenta koji se u ovom projektu naziva *agentmaster-kv9o.* Nakon toga generirao se JSON objekt koji se može vidjeti na slici 36. Tu se nalaze sve potrebne informacije koje su potrebno za povezivanje, autentifikaciju i dobivanje odgovora direktno s agenta za razgovornog robota. Ti podaci su se stavili unutar *.env* datoteke jer ne smiju biti javno dostupni. Svi potrebni koraci za generiranje tih informacija nalaze se unutar njihove dokumentacije koja se nalazi na internet stranici: https://console.cloud.google.com/apis/dashboard.

Slika 36: JSON objekt informacija za pristup Dialogflow agentu

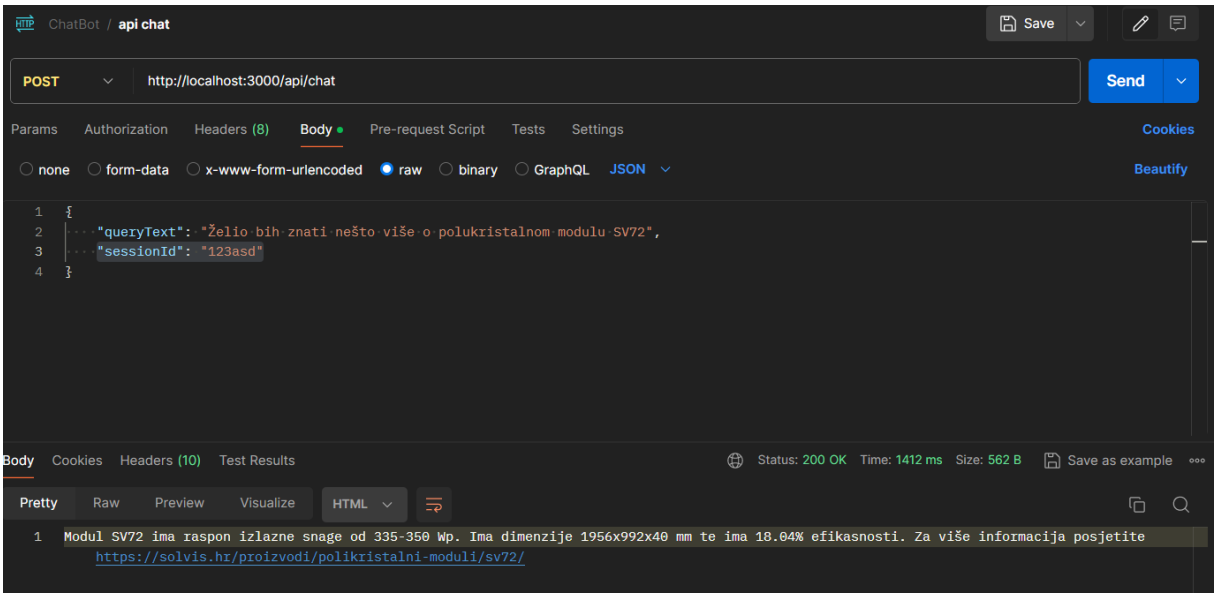
Ovaj servis se može pokrenuti kroz *terminal* s narednom *npm run start* te se servis može testirati na pristupnoj točci */health-check* što se može vidjeti na slici 37. Na slici 37. se testiranje provelo kroz internet preglednik, no to je u redu samo za testiranje jednostavnih pristupnih točki poput */health-check*. Autor predlaže korištenje Postman-a[12] kojeg je i on sam koristio za prvo testiranje servisa bez razrađenih vizualnih komponenti.

[12] Postman je alat za razvoj API-ja koji omogućava programerima da testiraju, dokumentiraju i razmjenjuju HTTP zahtjeve. Razvio ga je Abhinav Asthana 2012. godine, a danas je Postman Inc. službeni distributer. Više informacija na: https://www.postman.com/



Slika 37: Testiranje servisa razgovornog robota kroz internet preglednik

Kako bi testirao pristupnu točku */api/chat* autor je koristio Postman kroz koji je proslijedio i JSON objekt kroz tijelo zahtjeva. Prikaz Postman definirane pristupne točke skupa sa generiranim odgovorom agenta na prosljeđene parametre može se vidjeti na slici 38.



Slika 38: Testiranje */api/chat* pristupne točke kroz Postman

Testiranjem se zaključuje da servis radi i da vraća odgovor potreban za realizaciju vizualnih sučelja koje će se razviti u stilu internet i mobilne aplikacije. Internet aplikacija će se temeljit na Javascript programskom jeziku, a mobilna aplikacija na Kotlinu.

7.1. Prototip internet aplikacije

Prototip internet aplikacije se temelji na Javascript programskom jeziku unutar Visual Studio Code-a pomoću Vue.js tehnologije. Vue.js tehnologija omogućava da se vrlo i jednostavno kreira projekt koji se sastoji od različitih komponenti te je zato prikladan za razgovornog robota. Okvir razgovornog robota će se razviti kao komponenta koja se može implementirati u neki drugi već razvijeni projekt.

Nakon istestiranih i razrađenih pristupnih točki servisa kreirao se Vue projekt unutar istog repozitorija. Kako bi se kreirao Vue projekt prvo je potrebna instalacija Vue.js tehnologije. Nakon instalacije, postavi se unutar mape u kojoj želimo kreirati projekt i izvrši se komanda *vue create *ime-projekta** te se time dobiva projekt koji se automatski može pokrenuti. Za prototip razgovornog robota kreirala se dodatna komponenta *chat-container* koja predstavlja okvir razgovora korisnika s razgovornim robotom. Ta komponenta je koristila *axios* biblioteku za slanje zahtjeva na servis što se može vidjeti u programskog kodu 4. Metoda *sendMessage* kreira novu poruku koju šalje korisnik koju dalje koristi za dobivanje odgovora i pospremanje svih odgovora u jednu listu kako bi se dobila funkcionalnost i izgled razgovornog okvira s agentom.

```
async sendMessage() {
      if (this.userMessage.trim() === '') return;

      const newMessage = {
        id: this.messages.length + 1,
        text: this.userMessage,
        sender: 'user',
        time: new Date().toLocaleTimeString(),
      };

      this.messages.push(newMessage);

      try {
        const response = await axios.post('http://localhost:3000/api/chat',
{ queryText: this.userMessage, sessionId: '20230132182302' });
        this.messages.push({
          id: this.messages.length + 1,
          text: response.data,
          sender: 'bot',
          time: new Date().toLocaleTimeString(),
        });
      } catch (error) {
        console.error('Error sending message:', error);
        this.messages.push({
          id: this.messages.length + 1,
          text: 'Error getting response from server.',
          sender: 'bot',
          time: new Date().toLocaleTimeString(),
        });
      }

      this.userMessage = '';
      this.scrollToBottom();
    }
```

Programski kod 4: Slanje zahtjeva i dobivanje odgovora za vizualni prikaz

Unutar komponente *chat-container* se još nalazi i njezin izgled (engl. Cascading Style Sheets, u daljnjem tekstu css) te su se koristile biblioteke koje daju mogućnost korištenja ikona

sa stranice Font Awesome (poveznica: https://fontawesome.com/icons). Ta komponenta se uvezla u glavnu komponentu *App.vue* što prikazuje programski kod 5.

```
<template>
  <ChatContainer/>
</template>

<script>
import ChatContainer from './components/chat-container.vue';

export default {
  name: 'App',
  components: {
    ChatContainer
  },
}
</script>

<style>
#app {
  font-family: Avenir, Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
  margin-top: 60px;
  display: flex;
  justify-content: center; /* Center horizontally */
  align-items: center;     /* Center vertically */
  height: 100vh;           /* Full viewport height */
  width: 100vw;            /* Full viewport width (if needed) */
}
</style>
```

Programski kod 5: Uvoz chat-container komponente u glavnu App.vue

Nakon uvoza komponenti, glavna komponenta se pokreće kroz programski kod *main.js* datoteke odnosno izvršavanjem naredbe *npm run serve*. Sadržaj datoteke *main.js* se vidi u programskog kodu 6.

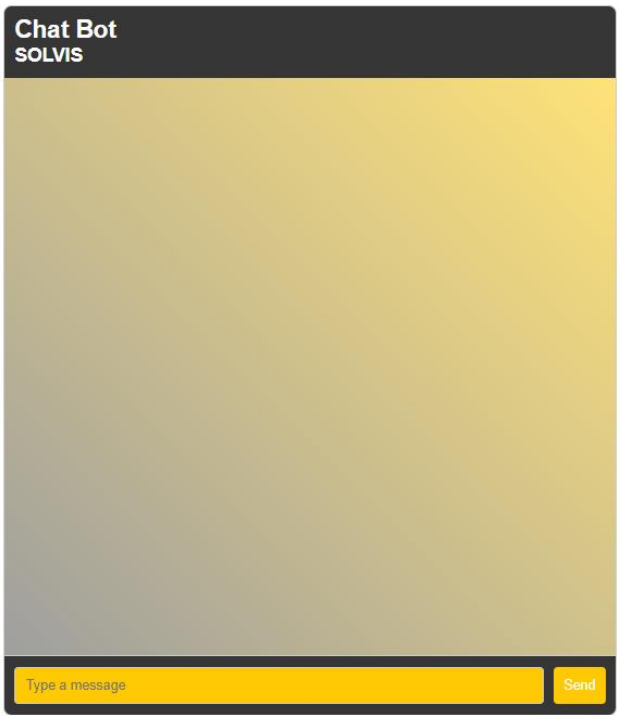
```
import { createApp } from 'vue'
import App from './App.vue'

createApp(App).mount('#app')
```

Programski kod 6: Main.js

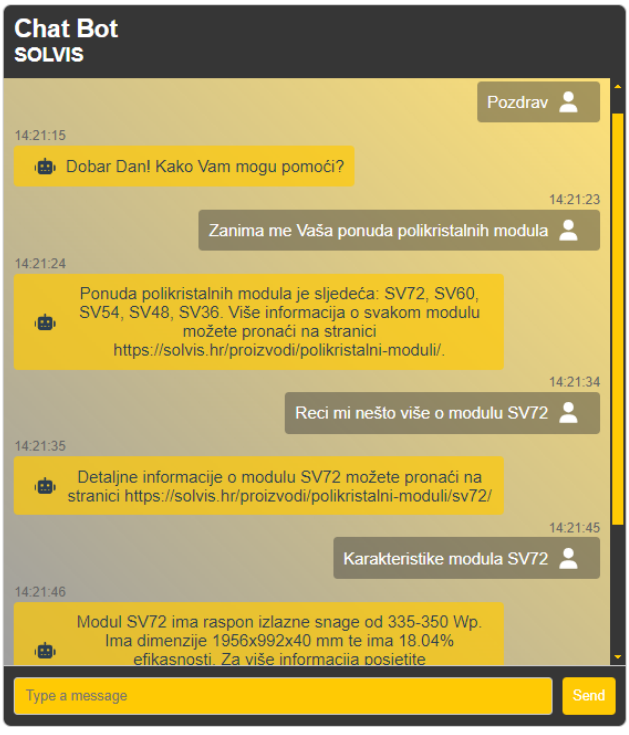
Nakon pokretanje vizualnog dijela projekta može se otići na http://localhos

t:8080/ što daje sučelje kao na slici 39. Kroz to sučelje se sada može testirati cijeli projekt koji je sastavljen od Dialogflow ES agenta, servisa s dohvaćanjem odgovora na izraze te vizualnog dijela aplikacije.



Slika 39: Sučelje razgovornog robota

Kako bi autor prikazao da prototip razgovornog robota radi kreirao je razgovor koji je povezan direktno sa Solvis tvrtkom i njihovim proizvodima. Taj razgovor se može vidjeti na slici 40.



Slika 40: Razgovor korisnika i robota kroz sučelje

Slikom 40. je dokazano da je razvijen funkcionalan okvir za razgovor između korisnika i robota, implementiran kao komponenta koja se lako može integrirati u druge projekte. Testiranjem i evaluacijom sučelja i robota potvrđena je funkcionalnost i pouzdanost sustava. Kreirani prototip služi kao solidna osnova za daljnji razvoj i integraciju razgovornog robota u različite internet programske proizvode, što dodatno potvrđuje njegovu praktičnu vrijednost i primjenjivost u stvarnom okruženju.

7.2. Prototip mobilne aplikacije

Prototip mobilne aplikacije autor je kreirao unutar programa Android Studio inačice „Koala“. Kreirana je mobilna aplikacija koja je povezana na postojeći servis. Čim se aplikacija otvori dobije se sučelje za razgovor s razgovornim robotom. Sučelje je kreirano kroz datoteke formata *.xml* (engl. EXtensible Markup Language, u daljnjem tekstu XML). Kreirano je slično sučelje kao i unutar internet programskog proizvoda. Sastoji se od naslova razgovornog robota, razgovornog dijela s ponavljajućim pogledima (engl. recycler view) i dijela upisa poruke korisnika. XML tog sučelja se vidi u programskih kodovima 7. i 8. dok se vizualno sučelje može vidjeti na slici 41.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!-- Header Layout -->
    <LinearLayout
        android:id="@+id/headerLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:background="@color/black"
        android:padding="16dp">

        <TextView
            android:id="@+id/headerTitle"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Chat Bot"
            android:textColor="@color/white"
            android:textSize="24sp"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/headerSubtitle"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="SOLVIS"
            android:textColor="@color/white"
            android:textSize="16sp" />
    </LinearLayout>
```

```
    <!-- RecyclerView with gradient background -->
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_below="@id/headerLayout"
        android:layout_above="@id/inputLayout"
        android:padding="10dp"
        android:clipToPadding="false"
        android:scrollbars="vertical"
        android:background="@drawable/gradient_background" />

    <!-- Input Layout -->
    <LinearLayout
        android:id="@+id/inputLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_alignParentBottom="true"
        android:background="@color/black"
        android:padding="8dp"
        android:gravity="center">

        <EditText
            android:id="@+id/inputMessage"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:hint="Type your message..."
            android:background="@drawable/rounded_yellow"
            android:textColor="@color/black"
            android:inputType="text"
            android:padding="12dp"
            android:layout_marginEnd="8dp" />

        <android.widget.Button
            android:id="@+id/sendButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Send"
            android:background="@drawable/rounded_yellow"
            android:textColor="@color/black"
            android:padding="12dp" />
    </LinearLayout>

</RelativeLayout>
```

Programski kod 7: XML datoteka sučelja

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="8dp">

    <TextView
        android:id="@+id/messageText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:padding="8dp"
        android:layout_margin="8dp"
        android:textColor="#000000"
        android:textSize="16sp" />

</RelativeLayout>
```

Programski kod 8: Ponavljajući objekt unutar sučelja

Slika 41: Sučelje mobilne aplikacije

Mobilna aplikacija koristi *Retrofit* biblioteku kako bi slala zahtjeve na pokrenuti servis od kojeg dobiva odgovor isto kao i internet aplikacija. Taj odgovor se prikazuje unutar razgovornog dijela mobilne aplikacije. Proces slanja zahtjeva i dobivanja odgovora sa servisa se sastoji od nekoliko koraka. Prvi korak je kreiranje retrofit instance s mogućnosti konvertiranja JSON-a, a što se može vidjeti u programskog kodu 9 i 10.

```
val retrofit = Retrofit.Builder()
            .baseUrl("http://10.0.2.2:3000/")
            .addConverterFactory(GsonConverterFactory.create())
            .build()

        chatbotService = retrofit.create(ChatbotService::class.java)
```

Programski kod 9: Kreiranje retrofit instace i servisa

```
public interface ChatbotService {
    @POST("api/chat")
    Call<ResponseBody> sendMessage(@Body Message message);
}
```

Programski kod 10: Sučelje ChatbotSerivce za slanje POST zahtjeva

Nakon kreiranja retrofit instance odnosno objekta servisa potrebno nam je nekoliko funkcija koje će poslati poruku, prikazati tu poruku u razgovornom dijelu te dobiti i prikazati odgovor. Sljedeći programski kodovi prikazuju kako se to implementiralo unutar android studia.

```
data class Message(
    @SerializedName("queryText") val content: String,
    @SerializedName("sessionId") val sessionId: String,
)
```

Programski kod 11: Objekt Message

Klasa koja se koristi prilikom kreiranja poruka za slanje na servis te objekt koji se šalje sučelju. Isti objekt se koristi za zadavanje sesije prema servisu te identificiranje poruke ako je ona od korisnika ili razgovornog robota na samom sučelju.

```
val sendButton = findViewById<Button>(R.id.sendButton)
sendButton.setBackgroundResource(R.drawable.rounded_yellow)
val inputMessage = findViewById<EditText>(R.id.inputMessage)
sendButton.setOnClickListener {
    val userInput = inputMessage.text.toString().trim()
    if (userInput.isNotEmpty()) {
        val userMessage = Message(userInput, "sess123")
        displayMessage(userMessage)
        sendMessageToChatbot(userMessage)
        inputMessage.text.clear()
    }
}
```

Programski kod 12: Čitanje i slanje korisnikove poruke

```
private fun sendMessageToChatbot(message: Message) {
    val call = chatbotService.sendMessage(message)
    call.enqueue(object : Callback<ResponseBody> {
        override fun onResponse(call: Call<ResponseBody>, response:
Response<ResponseBody>) {
            val chatbotMessage = response.body()?.string()
            chatbotMessage?.let  { displayMessage (Message(it, "bot") )}
        }

        override fun onFailure(call: Call<ResponseBody>, t: Throwable) {
            t.printStackTrace()
        }
    })
}
```

Programski kod 13: Slanje poruke prema servisu te slanje odgovora servisa na mobilnu aplikaciju

```
private fun displayMessage(message: Message) {
    Log.d("ChatBot", "Displaying message: ${message.content}")
    val recyclerView = findViewById<RecyclerView>(R.id.recyclerView)
    messageAdapter.messages.add(message)
    messageAdapter.notifyItemInserted(messageAdapter.messages.size - 1)
    recyclerView.scrollToPosition(messageAdapter.messages.size - 1)
}
```

Programski kod 14: Funkcija prikaza poruka na sučelju

Uz pomoć programskih kodova 12, 13 i 14 se dobivaju poruke se šalju poruke na servis i dobivaju poruke s Dialogflow agenta kroz isti servis. Također s funkcijom *displayMessage* se poruke prikazuju unutar sučelja.

```
class MessageAdapter(public val messages: ArrayList<Message>) :
RecyclerView.Adapter<MessageAdapter.MessageViewHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
MessageViewHolder {
        val view =
LayoutInflater.from(parent.context).inflate(R.layout.item_message, parent,
false)
        return MessageViewHolder(view)
    }

    override fun onBindViewHolder(holder: MessageViewHolder, position: Int)
{
        val message = messages[position]
        holder.bind(message)
    }

    override fun getItemCount(): Int {
        return messages.size
    }

    inner class MessageViewHolder(itemView: View) :
RecyclerView.ViewHolder(itemView) {
        private val messageText: TextView =
itemView.findViewById(R.id.messageText)

        fun bind(message: Message) {
            messageText.text = message.content

            val params = messageText.layoutParams as
RelativeLayout.LayoutParams
            if (message.sessionId != "bot") {
                params.addRule(RelativeLayout.ALIGN_PARENT_END)
                params.removeRule(RelativeLayout.ALIGN_PARENT_START)

messageText.setBackgroundResource(R.drawable.bot_message_background)

messageText.setTextColor(itemView.context.getColor(R.color.white))

            } else {
```

```
                params.addRule(RelativeLayout.ALIGN_PARENT_START)
                params.removeRule(RelativeLayout.ALIGN_PARENT_END)

messageText.setBackgroundResource(R.drawable.user_message_background)
            }
            messageText.layoutParams = params
        }
    }
}
```

Programski kod 15: Klasa koja definira prikaz poruka unutar ponavljajućeg pogleda

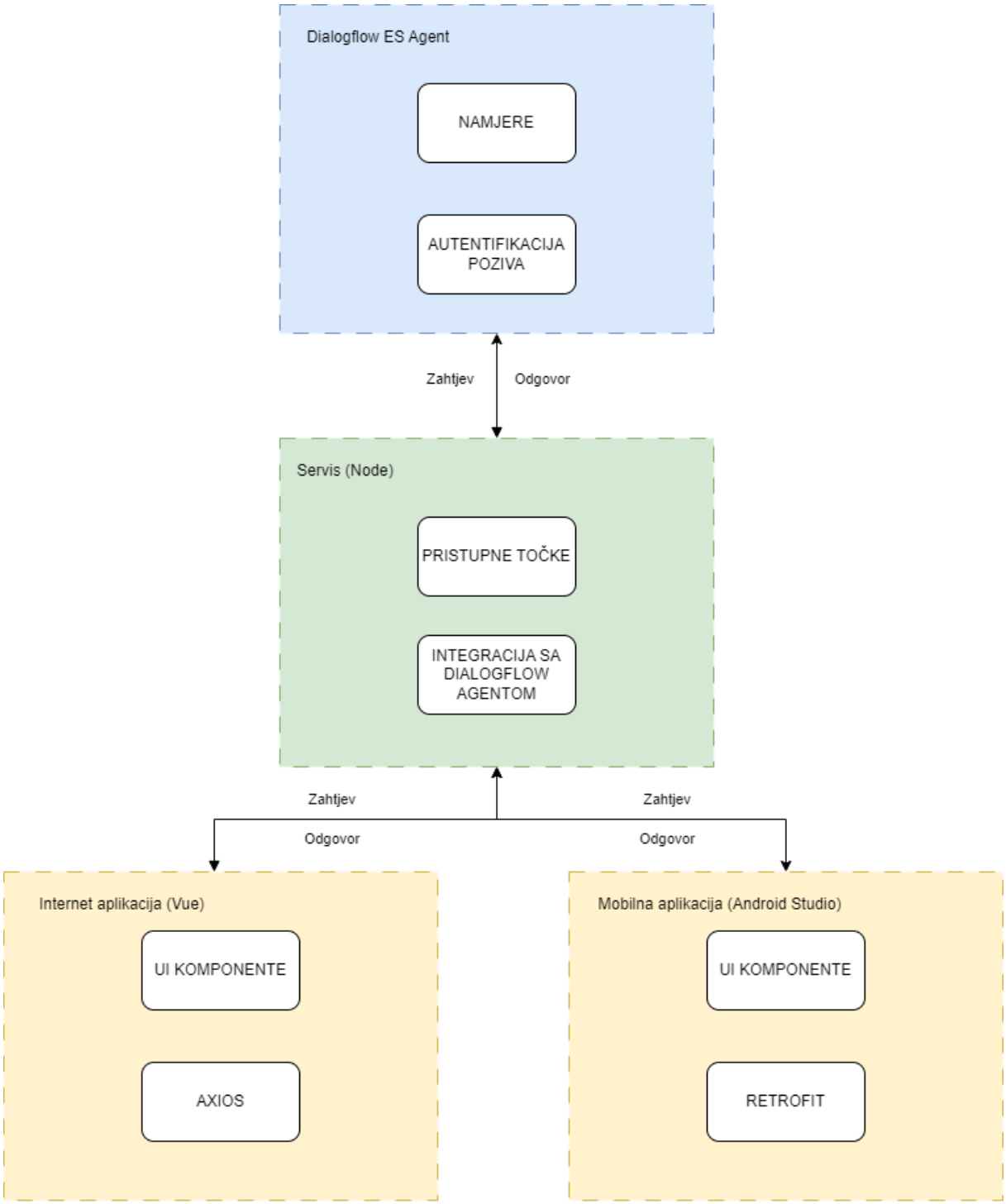
S *MessageAdapter* – om se kreira i održava format prikaza objekta identifikacijske oznake messageText unutar ponavljajućeg objekta. On održava broj poruka koje će se vidjeti na razgovornom dijelu, definira razliku između poruka razgovornog robota ili korisnika te za svaki tip definira određene argumente kako bi se poruke razlikovale na sučelju.

Uz pomoć ovih programskih kodova i definiranih stilova moguće je napraviti standardnu mobilnu aplikaciju razgovora između robota i korisnika kroz nekoliko sati. Ova mobilna aplikacija se konkretno povezuje na emulator tj. virtualni prikaz mobitela *Pixel 8 Pro API 24*. Zbog tog povezivanja bitno je postaviti bazični URL na http://10.0.2.2:3000/ kako bi se ustvari pristupilo do lokalnog poslužitelja na kojem kontinuirano radi servis. Ovaj razgovorni robot je razvijen unutar pogleda koji se može postaviti na bilo koju akciju, konkretno u prototipu je postavljen na učitavanje same aplikacije, no npr. može se postaviti da se učitava na klik gumba za prikaz razgovornog robota.

Zaključno s mobilnom aplikacijom vidi se kako je izrađeni servis i Dialogflow agent moguće iskoristiti u bilo kojem modernom programskog proizvodu te da je implementacija vrlo jednostavna i brza.

7.3. Dijagram arhitekture servisa, prototipa i agenta

Dijagram arhitekture prototipa modernog programskog proizvoda s funkcionalnosti razgovornog robota se može vidjeti na slici 42. Na slici se može vidjeti kako je za prototip modernog programskog proizvoda potreban agent (u ovom primjer Dialogflow ES), servis i sučelja. Servis dobiva zahtjev od sučelja te šalje agentu te od agenta dobiva odgovor kojeg prosljeđuje sučeljima. Na dobiveni odgovore, sučelja unutar pojedinih komponenti upisuju dobiveni odgovor koji je dobiven na poslani zahtjev pomoću biblioteka axios ili retrofit.



Slika 42: Dijagram arhitekture servisa, agenta i sučelja

Prema slici 42. može se vidjeti kako je funkcionalni prototip modernog programskog proizvoda vrlo jednostavno za izraditi te se svaki od prototipa još može nadograditi. U prototipe se vrlo jednostavno može dodati još komponenata dok se na servis može dodati još pristupnih točki koja bi koristila za druge funkcionalnosti povezane s dodatnim agentima/integracijama.

8. Zaključak

Kako bi opstali moderni programski proizvod poput internet stranica, mobilnih aplikacija, aplikacija za računalo i ostalih moraju sadržavati mnogo značajki. Po autorovom mišljenju integrirani razgovorni robot je jedna od njih. Razgovorni roboti pridonose poboljšanju korisničkog iskustva te samim time i npr. prometu na nekoj internet stranici.

Razvoj razgovornih robota je vrlo jednostavan. Naime, iako je razvoj jednostavan, vrlo je bitno savladati koncepte NLP-a, NLU-a te bitne pojmove razgovornih robota. Također, ovisno o tome gdje i kako će se koristiti razgovorni robot, odabir tehnologije za razvoj agenta razgovornog robota ima veliku ulogu u razvoju. Na to utječe opseg samog agenta, opseg korištenja, veličina tvrtke koja će ga koristiti tj. njihovi osobni i zakonski zahtjevi (licence) itd.

U radu su se usporedile 4 tehnologije za razvoj razgovornih robota kroz 5 karakteristika. Dobili su se rezultati koji su djelomično bili i očekivani. Prema tim rezultatima najbolje razvijena tehnologija je Google Dialogflow koji pruža veliki broj mogućnosti i dvije vrste Dialogflow-a (Dialogflow ES i CX) koje se razlikuju po pružanim mogućnostima. Dokumentacija za API pozive pruža sve potrebno kako bi se povezao agent s razvijenim servisom.

U praktičnom dijelu rada razvio se prototip sučelja razgovornog robota. Razvijeni prototip (repozitorij na stranici: https://github.com/David-Viljevac/chat-bot-solar-panels) se može nadograditi ili se vrlo jednostavno mogu iskoristiti pristupne točke i komponente iz rada za integraciju razgovornog robota unutar neke druge već razvijene stranice. Iako je sav programski kod dostupan na repozitoriju, pristupni podaci razgovornog agenta nisu kako bi se zaštitilo autora od potencijalnih napada.

Popis literature

[1] A. S. Gillis, „What is natural language processing? | Definition from TechTarget“, Enterprise AI. Pristupljeno: 04. svibanj 2024. [Na internetu]. Dostupno na: https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP

[2] R. Wolf, „Natural Language Processing (NLP): 7 Key Techniques“, MonkeyLearn Blog. Pristupljeno: 04. svibanj 2024. [Na internetu]. Dostupno na: https://monkeylearn.com/blog/natural-language-processing-techniques/

[3] J. Skrebeca, P. Kalniete, J. Goldbergs, L. Pitkevica, D. Tihomirova, i A. Romanovs, *Modern Development Trends of Chatbots Using Artificial Intelligence (AI)*. 2021, str. 6. doi: 10.1109/ITMS52826.2021.9615258.

[4] „What Is Natural Language Understanding (NLU)?“, Qualtrics. Pristupljeno: 05. svibanj 2024. [Na internetu]. Dostupno na: https://www.qualtrics.com/experience-management/customer/natural-language-understanding/

[5] Let The Data Confess, „Best Chatbot Frameworks You Must Know About“, Medium. Pristupljeno: 04. svibanj 2024. [Na internetu]. Dostupno na: https://medium.com/@letthedataconfess/best-chatbot-frameworks-you-must-know-about-6926d9e4d80f

[6] C. Owen, „The Complete Guide to NLU“, Inform Comms. Pristupljeno: 05. svibanj 2024. [Na internetu]. Dostupno na: https://www.inform-comms.com/the-complete-guide-to-nlu/

[7] S.-F. Yeh *i ostali*, „How to Guide Task-oriented Chatbot Users, and When: A Mixed-methods Study of Combinations of Chatbot Guidance Types and Timings“, u *CHI Conference on Human Factors in Computing Systems*, New Orleans LA USA: ACM, tra. 2022, str. 1–16. doi: 10.1145/3491102.3501941.

[8] „Enterprise Chatbots: What We Know, and How We'll Act – Kaleo Software“. Pristupljeno: 11. svibanj 2024. [Na internetu]. Dostupno na: https://www.kaleosoftware.com/enterprise-chatbot-data/

[9] „Framework“. Pristupljeno: 12. svibanj 2024. [Na internetu]. Dostupno na: https://encyclopedia.kaspersky.com/glossary/framework/

[10] „Dialogflow Documentation“, Google Cloud. Pristupljeno: 12. svibanj 2024. [Na internetu]. Dostupno na: https://cloud.google.com/dialogflow/docs

[11] „Dialogflow“, Google Cloud. Pristupljeno: 12. svibanj 2024. [Na internetu]. Dostupno na: https://cloud.google.com/dialogflow

[12] „Dialogflow ES basics | Google Cloud“. Pristupljeno: 18. svibanj 2024. [Na internetu]. Dostupno na: https://cloud.google.com/dialogflow/es/docs/basics

[13] „AWS Skill Builder“. Pristupljeno: 30. svibanj 2024. [Na internetu]. Dostupno na: https://explore.skillbuilder.aws/learn/course/17999/play/96297/amazon-lex-getting-started

[14] „Wit.ai“. Pristupljeno: 30. svibanj 2024. [Na internetu]. Dostupno na: https://wit.ai/jobs

[15] „Wit.ai“. Pristupljeno: 30. svibanj 2024. [Na internetu]. Dostupno na: https://wit.ai/docs

[16] „Getting Started | OpenDialog Documentation | OpenDialog Docs“. Pristupljeno: 01. lipanj 2024. [Na internetu]. Dostupno na: https://docs.opendialog.ai

[17] „Pricing | Dialogflow“, Google Cloud. Pristupljeno: 08. lipanj 2024. [Na internetu]. Dostupno na: https://cloud.google.com/dialogflow/pricing

[18] „Amazon Lex Pricing - Amazon Web Services“, Amazon Web Services, Inc. Pristupljeno: 08. lipanj 2024. [Na internetu]. Dostupno na: https://aws.amazon.com/lex/pricing/

Popis slika

Popis tablica

Popis programskih kodova