Web aplikacija s agentom za razgovor u prirodnom jeziku

Sajčić, Stella

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Stella Sajčić

WEB APLIKACIJA S AGENTOM ZA RAZGOVOR U PRIRODNOM JEZIKU

DIPLOMSKI RAD

Varaždin, 2019.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Stella Sajčić

Matični broj: 46402/17–R

Studij: Informacijsko i programsko inženjerstvo

WEB APLIKACIJA S AGENTOM ZA RAZGOVOR U PRIRODNOM JEZIKU

DIPLOMSKI RAD

Mentor/Mentorica:

Izv. prof. dr. sc. Markus Schatten

Varaždin, lipanj 2019.

Stella Sajčić

Izjava o izvornosti

Izjavljujem da je moj završni/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.

Autorica potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sažetak

Ovim radom čitatelj će moći dobiti uvid što je to konverzacijski agent, što ga čini manje ili više kompleksnim te kako dokumentirati zahtjeve istog prije njegovog razvoja. U praktičnom dijelu razviti će se konverzacijski agent (eng. *Chatbot*) kao komponenta web sučelja s kojim je korisnik u interakciji. Tijekom razrade praktičnog dijela opisati će se koraci koji su potrebni za razvoj konverzacijskog agenta bez primjene postojeće baze znanja te koji mogući problemi nastaju tijekom razvoja istog.

Web sučelje biti će razvijeno *React.js* bibliotekom koja će komunicirati sa *Node.js* serverom. Kako bi se razumjeli korisnički upiti koristi će se *Wit.ai* servis za obradu prirodnog jezika odnosno aplikacijsko programsko sučelje (eng. *Application programming interface*) razvijen od strane *Facebook*-a.

Ključne riječi: chatbot; konverzacijski agent; obrada prirodnog jezika; generiranje odgovora; web aplikacija; korisničko iskustvo; dokumentiranje zahtjeva agenta; razvoj konverzacijskog agenta;

Sadržaj

1. Uvod

Pojavom konverzacijskih agenata razvijeni od strane multinacionalnih kompanija, kao što su *Facebook*, *Google*, *Amazon* i drugi, doživio je svoj procvat i često se interpretira kao "nova" tehnologija, iako njegova povijest doseže još od 1950. godine pojavom Turingovog testa čija je korisnička zadaća prepoznati da li osoba s kojom komunicira čovjek ili stroj. Danas, nakon 69 godina, konverzacijski agent se proširio na mnoga tržišta/potrebe, najčešće u obliku korisničkih podrški. Smatramo ga korisničkim sučeljem pomoću kojeg omogućavamo interakciju korisnika (čovjeka) sa servisom i brendom proizvoda/usluge. Korisnik može na brži i jednostavniji način pronaći informacije nego što bi kroz web ili mobilno sučelje gdje su često informacije sakrivene ili teško dostupne. Naravno, takav željeni cilj uvelike ovisi o razumjevanju korisničkog upita, odgovaranje na njega i primjena korisničkog iskustva koje ne predstavlja samo animacije i pristupačnost već i brend proizvoda/usluge. [2, 25]

Primjene konverzacijskog agenta mogu biti mnoge, kao sto su: poboljšanje korisničkog iskustva, edukacija, financije, marketing, zapošljavanje, korisnička podrška i drugo, a njegova "kvaliteta" može varirati o različitim faktorima [3]. Umjetna inteligencija (eng. *Artificial intelligence*), obrada prirodnog jezika (eng. *Natural language process*), strojno učenje (eng. *Machine learning*), dubinsko učenje (eng. *Deep* learning), generiranje odgovora (eng. *Response generation*), baza znanja (eng. *Knowledge Base*), faktori su koji utječu na funkcionalnost konverzacijskog agenta [2]. Kakav značaj imaju, navedene funkcionalnosti, u agentu i što agenta čini složenijim opisivati ću u trećem poglavlju *Mogućnosti razumjevanja i rješavanja problema.* Nakon što uvidimo važnost obrade prirodnog jezika, u četvrtom poglavlju *Namjene i entiteti u Wit.ai servisu* opisivati ću kako servis Wit.ai rješava problematiku razumjevanja korisničkih upita.

Danas nije dovoljno samo zadovoljiti zahtjeve razumjevanja problema i njegovog rješavanja već moramo zadovoljiti i korisničko iskustvo koje korisnik stječe korištenjem konverzacijskog agenta. Kako korisničko iskustvo ima utjecaj na korisnika i kako dokumentiranje zahtjeva konverzacijskom agenta ima utjecaj na isto

govoriti ću u petom poglavlju *Korisničko iskustvo.* Peto poglavlje je ujedno i uvod o ciljevima projektnog zadatka razrađenim u šestom poglavlju.

Projektni zadatak opisivati će tehnologije, alate i servise koji su korišteni te definirati ciljeve, funkcionalnosti, arhitekturu i strukturu baze podataka za odabranu domenu web aplikacije. Sedmo poglavlje opisuje kritički prikaz nad projektnim zadatkom i konverzacijskim agentima općenito i na kraju zaključiti.

2. Metode i tehnike rada

U ovom radu obrađivati će se teorijski i praktični dio kako bi se prikazali potrebni koraci razvoja konverzacijskog agenta od njegove dokumentacije do razvoja. U teorijskom dijelu diplomskog rada opisivati će se ključni pojmovi koji čine konverzacijski agent manje ili više kompleksnim, koristeći literaturu iz različitih izvora.

Za prikaz potrebnih koraka razvoja konverzacijskog agenta razviti će se web aplikacija koja implementira konverzacijskog agenta kao samostalnu komponentu. Web aplikacija, biti će kreirana u *Javascript* programskom jeziku, točnije *React.js* [4] biblioteci, koja će se pokretati na korisničkom pregledniku. Za prikazivanje upita korisnika i odgovor na iste, web aplikacija će komunicirati sa serverom preko soketa kako bi uspostavila dvosmjernu komunikaciju [26]. Aplikacija pokrenuta na serveru, također će biti napisana u *Javascript* programskom jeziku odnosno *Node.js* [6] izvršnom okružju (eng. *Runtime environment*), koja će pristupati *PostgreSQL* [27] bazi podataka i *Wit.ai* [5] servisu za obradu prirodnog jezika za razumjevanje korisničkih upita. Odgovori *Wit.ia* servisa biti će preduvjet algoritmu za generiranje odgovora, koji će se izvršavati na *Node.js* serveru. Prije razvoja web aplikacije i konverzacijskog agenta definirati će se ciljevi, funkcionalnosti, arhitektura aplikacije, prototip dizajna koristeći *Marvel* [36] alat te dokumentirati korisničko iskustvo koje se želi postići kroz interakciju agenta i korisnika zasnovani na funkcionalnostima aplikacije.

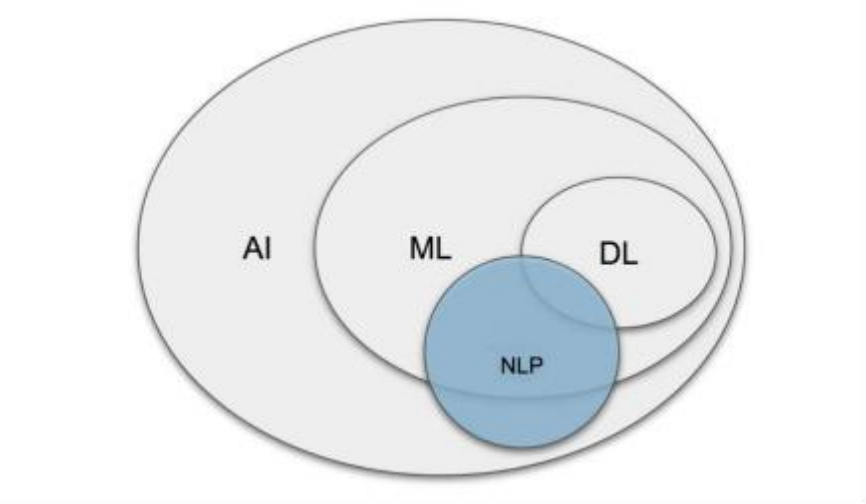
3. Mogućnosti razumjevanja i rješavanje problema

Bilo koji bot, da bi razumio problem koji korisnik postavlja, mora pretvoriti nestrukturirane u strukturirane podatke (eng. *Structured data*) odnosno u strukturu podataka koje računalo razumije. Taj zadatak nam olakšava tehnologija obrade prirodnog jezika (eng. *Natural procesing laguage - NLP*) zasnovana na tehnikama kao što su sintaksna analiza i semantička analiza. Ukoliko konverzacijski agent uči na svojim prethodnim iskustvima i ima mogućnost predviđanja, tada govorimo o strojnom učenju (eng. *Machine learning*) i dubokom učenju (eng. *Deep learning*). Iako strojno i dubinsko učenje u suštini vrlo slični i često poistovjećeni, dubinsko učenje je dio strojnog učenja razlikujući se po načinu učenja iz podataka odnosno algoritmima koje koriste. Algoritam strojnog učenja radi na način da raščlanjuje podatke i uči iz njih, osiguravajući mu određeni set znanja (podataka) na temelju kojeg će naučiti identificirati objekte određene domene. Dubinsko učenje se temelji na neuronskoj mreži koja zahtjeva veliku količinu podataka i procesorske snage (u oba slučaja veću nego što zahtjeva strojno učenje) pokušavajući samostalno identificirati objekte određene domene. Na primjer, ako imamo set fotografija koje prikazuju urede i zgrade organizacija, algoritmu strojnog učenja bi za svaki objekt, *ured* i *zgrada,* morali osigurati zasebne primjere na temelju kojih će naučiti kako identificirati ured, a kako zgradu. Za dubinsko učenje to ne bi bio slučaj, već će algoritam, bez eksplicitnog navođenja kako identificirati objekte, prepoznati razlike između njih. [16, 17, 18]

Strojno učenje i dubinsko učenje možemo sagledati kao instanca koja čini konverzacijski agent pametnijim i samostalnijim u poboljšavanju razumjevanja korisnika bez intervencije programera, ali to ujedno ne znači i samostalnijim u odlučivanju rješavanja problema. Kako bi bio u mogućnosti samostalnog obavljanja/rješavanja zadatka sadržavajući neki oblik inteligencije tada pričamo o umjetnoj inteligenciji (eng. *Artificial Intelligence*). [14, 15, 16]

Na sljedećoj slici možemo vidjeti vezu između umjetne inteligencije, strojnog učenja, dubokog učenja i obrade prirodnog jezika, gdje iz već opisanog možemo vidjeti njihovu međusobnu vezu. Umjetna inteligencija, strojno učenje i duboko učenje ne mogu postojati bez obrade prirodnog jezika - NLP jer uz pomoć istog

pretvara nestrukturirane podatke (korisnički upit), u strukturirane podatke razumljive računalu odnosno algoritmima. Strojno učenje može postojati bez dubokog učenja jer je duboko učenje grana strojnog učenja, kao što i umjetna inteligencija može postojati bez strojnog učenja.



Slika 1. Međusobna veza između umjetne inteligencije - AI, strojnog učenja - ML, dubokog učenja - DL i obrade prirodnog jezika - NLP [8]

Osim što konverzacijski agent mora razumjeti problem koji mu postavlja korisnik, isto tako mora znati odgovoriti korisniku. Postoje tri modela generiranja odgovora: model temeljen na pravilima (eng. *Rule-based model*), model pretraživanja (eng. *Retrieval-based model*) i generativni model (engl. *Generative model*). Model temeljen na pravilima se temelji na predlošcima i uzorcima gdje na temelju identificiranja uzorka u korisničkom upitu, agent vraća odgovor koji je definiran u predlošku napisan od strane čovjeka (programera). Prednost takvog modela je što je jednostavan za napisati, ali ujedno dugotrajan proces kreiranja predložaka pokrivajući sve moguće scenarije korisničkog upita za danu domenu konverzacijskog agenta. Bazu znanja (eng. *Base knowledge*) koriste druga dva modela - generativni model i model pretraživanja, koji na temelju prethodnih iskustava (upita korisnika i odgovaranja na istih) generira odgovor. Model pretraživanja ne koristi algoritam generiranja odgovora kao takvo, već pretražuje bazu znanja, temeljenu na prethodnim iskustvima, i vraća odgovor iz nj. Generativni model se smatra najtežim za implementirati jer se temelji na dinamičkom generiranju odgovora - riječ po riječ. Ovisno o odabiru modela generiranja odgovora

konverzacijski agent može postati složeniji, koji je opisan u sljedećem potpoglavlju *Kompleksnost konverzacijskog agenta*. [2, 7, 19]

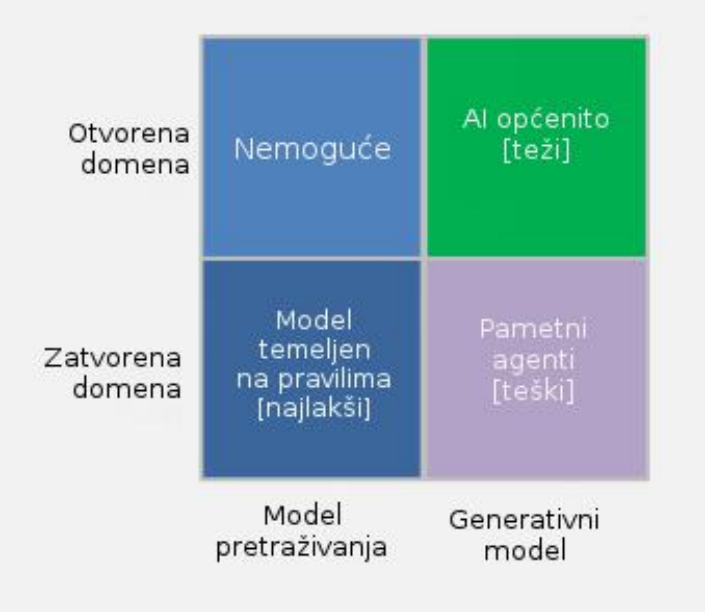
Za potrebe diplomskog rada aplikacija koristi generiranje odgovora temeljen na pravilima za koje postoje nekoliko rješenja. Najpoznatiji i najstariji pristup je *AIML* (eng. *Artificial Intelligence Markup Language*) *[32]*, temeljen na *XML* formatu odnosno proširivom jeziku za označavanje (eng. *Extensible Markup Language*) *[32]*, u kojem se definiraju uzorci i predlošci odgovora istog. Uzorci definiraju set rijeci koje korisnik može postaviti kao upit konverzacijskom agentu, slijedeći slijed riječi definiranim u istom. Nedostatak toga je ako korisnik promijeni slijed riječi pritom zadržavajući isti kontekst rečenice, uzorak neće biti okinut, tj. prepoznat. Njegov nasljednik je *RiveScript* skriptni jezik [30], koji za razliku *AIML*-a ne koristi *XML* format već sadrži set znakova koji definiraju da li se radi o okidaču (eng. *Trigger*), odgovor na isto ili nešto treće (na primjer, kreiranje varijable). Okidač se u tom slučaju smatra uzorakom kao i u *AIML* jeziku za označavanje (eng. *Markup Language*). Nadalje, postoji drugi skriptni jezik *ChatScript* [31] u kojem, također, možemo definirati uzorke korisničkih upita na koje će se generirati odgovor, ali za razliku *AIML* i *RiveScript*-a, uzorci su temeljeni na ključnim riječima koji se mogu prepoznati u korisničkom upitu. Zašto isti pristupi nisu dio rješenja projektnog zadatka opisano je u poglavlju *Projektni zadatak*.

3.1. Kompleksnost konverzacijskih agenata

Kompleksnost razvoja konverzacijskog agenta ovisi o dva faktora: domeni i modelu generiranja odgovora. Postoje dvije domene: otvorena i zatvorena. Zatvorena domena je domena u kojem konverzacijski agent “vezan” za određeni problem, domenu. Na primjer, ako je svrha konverzacijski agent pronaći korisniku najbližu praksu u gradu, na korisnički upit može odgovoriti samo za navedenu domenu. Ukoliko bi agent mogao odgovarati na sve korisničke zahtjeve neovisno o sektoru ili industriji, tada bi se radilo o otvorenoj domeni. Na primjer, ako bi korisnik osim upita o praksi mogao naručiti hranu [1].

Na sljedećoj slici možemo vidjeti klasifikaciju kompleksnosti razvoja konverzacijskog agenta. Ukoliko se radi o otvorenoj domeni i generativnom modelu, tada se razvoj smatra najtežim s obzirom na današnje znanja o istom. Naravno, o

samoj kompleksnosti agenta također ovisi da li konverzacijski agent implementira duboko i strojno učenje!



Slika 2. Klasifikacija težine razvoja konverzacijskog agenta na temelju domene i modela generiranja odgovora [7]

4. Namjene i entiteti u Wit.ai servisu

Kako obrada prirodnog jezika pretvara nestrukturirani tekst u strukturirani? Kako servis može prepoznati kada korisnik želi praksu sa zanimanjem programer? Kako servis prepoznaje da za sljedeće rečenice: “Ja sam programer. Želim pronaći praksu”, “Tražim praksu sa zanimanjem programer”, “Želim pronaći praksu. Programer.” identificira da korisnika zanima praksa?

Dva pojma se spominju kada pričamo o konverzacijskim agentima: namjera (eng. *Intent*) i entitet (engl. *Entity*). Namjera, kao što i sam naziv govori, opisuje korisnički upit odnosno postavljeni problem koji agent mora riješiti. Entiteti mogu definirati vrijeme, lokaciju, akciju, i slično, koji detaljnije opisuju namjenu odnosno postavljeni problem. Na primjer, ako korisnik ima sljedeći upit: “Želim danas ići sa sestrom u kino”, tada bi namjera bila ići u kino, a entiteti koje prepoznajemo u rečenici su vrijeme odlaska u kino - *danas*, i mjesto odlaska - *kino*. Kako bi servis za obradu prirodnog jezika što točnije prepoznao namjene i entitete unutar korisničkog upita, moramo mu priskrbiti što raznolikije primjere korisničkih upita. Za svaki primjer treniramo servis sa selektiranim riječima/rečenicama, za koje definiramo da li se radi o entitetu ili namjeni, ovisno o domeni interesa. Kako ćemo selektirati odnosno trenirati servis sa određenim skupom riječi na temelju kojeg će učiti, *Facebook* je predstavio poglednu strategiju [5].

Pogledna strategija (eng. *Lookup strategy*), definira tri vrste entiteta koje servis prepoznaje kod identifikacije istog, a to su: trenirani entitet (eng. *Train*), slobodni entitet (eng. *Free text*) i entitet ključnih riječi (eng. *Keywords*). Svaka od njih se koristi u različite svrhe. Entiteti se identificiraju kao ključne riječi kada pripadaju predefiniranoj listi riječi (na primjer popis zanimanja: *programer*, *dizajner*, *voditelj*) u kojoj onemogućavamo samostalno dodavanje novih riječi u listu od strane servisa. Ukoliko želimo omogućiti isto, tada ključne riječi dolaze u kombinaciji sa slobodnim riječima. Trenirani entiteti su oni koji se ne mogu svrstati u ključne niti u slobodne entitete, već cijela rečenica daje značenje entitetu što zapravo predstavlja namjenu (eng. *Intent*)! *Facebook* zapravo namjenu svrstava kao jedan od tipova treniranih podataka, što zapravo znači da svaka identifikacija riječi ili rečenice je entitet! Kako bi spriječili krivu interpretaciju između entiteta, sam *Facebook* u dokumentaciji namjenu

ne spominje kao entitet već kao zasebnu definiciju koja se razlikuje od entiteta, iako u osnovi to nije. [5]

Danas, često NLP servisi (kao *Rasa*, *DialogFlow*, *LUIS*) koriste predefinirane entitete [10, 11, 12, 13]. U *Wit.ai-u,* predefinirani entiteti se nalaze u samoj jezgri biblioteke *Duckling* zasnivani na pravilima i "treniranim" podacima. Na primjer, predefinirani jezgreni entitet *wit/amount_of_money* će za rečenicu "želim kupiti majicu za 10 eura" prepoznati entitet *novac*. Kako biblioteka zaista identificira entitet *novac*? Biblioteka za svaki entitet sadrži dvije datoteke - jedna za pravila, a druga za trenirane podatke. U tom slučaju će pravilo definirati valutu HRK, koji je ništa drugo nego regex (eng. *Regular expresssion*), sačinjen od svih mogućih oblika imenice u padežima, koji izgleda sljedeće "kn|(hrvatsk(a|ih|e))?kun(a|e)". Navedeni regex označava da će se valuta HRK prepoznati u rečenici kao "kn", "hrvatska kuna", "hrvatskih kuna", "hrvatske kune", pa čak i "hrvatskih kune" ili "hrvatska kune" koji su gramatički neispravni! Primjer nekih treniranih podataka bi bili sljedeći: "4 kune i 3 lipe", "četiri kune i 3 lipe", "četiri kune i tri lipe", "sto kuna". [9]

Važno je poznavanje algoritma za identificiranje entiteta u konverzacijskom agentu i identificiranje nedostataka istog kako bi poboljšali korisničko iskustvo korisnika. Što zapravo predstavlja korisničko iskustvo i kako ga poboljšati, opisano je u sljedećem poglavlju *Korisničko iskustvo*.

5. Korisničko iskustvo

Kada govorimo o dizajnu tada razlikujemo korisničko iskustvo (eng. *User experiece)* i korisničko sučelje (eng. *User interface*). Korisničko sučelje predstavlja grafičke elemente vidljive korisniku, a korisničko iskustvo predstavlja način kako korisnik doživljava grafičke elemente odnosno kakav utjecaj ima na njegove emocije, stavove, ponašanje, jednostavnost korištenja i dostupnost. [20]

Postoje nekoliko dobrih praksi kod implementacije korisničkog iskustva, kao što je personalizacija, učenje iz prethodnih razgovora, toleriranje pogrešaka, dvostruka provjera, evaluacija na temelju povratne informacije, usmjereni razgovori i drugi. Svaka od praksi se može primjeniti uzimajući u obzir, prije svega, o domeni konverzacijskog agenta. Kako bi konverzacijski agent učinili personaliziranijim moramo se obraćati korisniku individualno na temelju njegovih zahtjeva i situacije vodeći računa da potičemo korisnika na konverzaciju. Neki konverzacijski agenti znaju postavljati usmjerena pitanja sa mogućim odgovorima kako bi zadržali korisnika kako bi odali više informacija o svojem problemu. U tom slučaju, poželjno je da korisnik ne mora ponavljati svoje zahtjeve iz prethodnog razgovora već je agent svjestan scenarija s kojim se korisnik suočava. Ukoliko to nije u stanju, onda je poželjno pristupi podacima bez direktnog kontakta s korisnikom, kao što je dohvaćanje geografske lokacije, vremenske zone i slično, koje će pomoći agentu u usmjeravanju razgovora s korisnikom. Poželjno je da nakon što agent ustvrdi rješenje korisničkog problema upita korisnika za dodatnu provjeru. Dvostruka provjera se često koristi u agentima koji služe za kupnju proizvoda ili usluga, na primjer kupnja avionskih karata. [21, 22, 23]

Prije nego što želimo implementirati korisničko iskustvo u konverzacijski agent, moramo sagledati slabosti našeg agenta, tj. koju metodu generiranja odgovora koristimo te pokušati ublažiti robusnost agenta implementirajući neku od spomenutih dobrih praksi. Osim toga, moramo dokumentirati zahtjeve agenta kako bi mogli definirati moguće scenarije s kojim se korisnik može suočiti. Znamo da je skoro nemoguće pokriti sve moguće scenarije stoga je bitno poboljšavati agent na temelju korisničkih povratnih informacija. [22]

5.1. Kako dokumentirati zahtjeve konverzacijskog agenta

Najbolji način da postignemo što bolje korisničko iskustvo je dokumentiranje zahtjeva koje konverzacijski agent mora ispuniti za navedenu domenu. Postoje nekoliko koraka koje možemo slijediti za dokumentiranje istog.

Prvo i najvažnije što moramo ustvrditi je domena i ciljana skupina našeg konverzacijskog agenta. U ovom slučaju domena je pronalaženje prakse u Republici Hrvatskoj gdje su ciljana skupina školarci i studenti. Drugi korak je postaviti se u poziciju korisnika te postaviti scenarije zahtjeva ili motivacije korisnika. Definiranje takvog zahtjeva najčešće se prikazuje u sljedećem formatu: “Ja kao <tip korisnika> želim <cilj> za postizanje <rezultata>”. Kada postavljamo takav scenarij moramo razumjeti korisničke navike, ponašanje, preferencije, očekivanja i koji su ciljevi različitih tipova korisnika odnosno grupe korisnika. Student može imati drugačija očekivanja nego školarac, na primjer student želi praksu koja ima mogućnost zaposlenja, dok školarac ima u interesu pronaći praksu koja će biti u skladu s vremenom školske nastave. Treći korak, koji se postavlja kao alternativa drugom koraku, je definiranje slijeda konverzacije s korisnikom kako bi se odradio određeni zadatak. Dokumentiranje obavljanje zadatka je u formatu: “Kada se dogodi <situacija> za danu <motivaciju> dobiti <rezultat>”. Time definiramo kakva su očekivanja i motivacije korisnika za danu situaciju te možemo definirati manji set poslova koji mogu riješiti problem. Na primjer, ako korisnik traži praksu sa zanimanjem koje konverzacijski agent nije prepoznao ili nije pronašao moguću praksu koja bi zadovoljavala potrebama korisnika, tada bi moguća motivacija korisnika bila da agent predloži praksu koja se djelomice podudara sa njegovim zahtjevima. Četvrti korak je skiciranje slijeda identificiranja problema korisnika na način da se uzima u obzir korisničko iskustvo, koji su opisani u prethodnom poglavlju *Korisničko iskustvo*, kao što je toleriranje pogrešaka, usmjereni razgovori i drugo. Skiciranje slijeda identificiranja kakvu praksu korisnik traži može biti da prvo korisnika upitamo u kojem gradu traži praksu, nakon toga koje profesionalno zanimanje ga interesira te koje vještine posjeduje. Predzadnji korak je dokumentiranje liste namjera i entiteta koje koristi servis za obradu prirodnog jezika. Time možemo definirati koja će se akcija dogoditi kada imamo kombinaciju određenih namjena i entiteta. Najčešće se dokumentiranje prikazuje u obliku tablice koju možemo vidjeti na

sljedećoj tablici. Zadnji korak je kreiranje prototipa konverzacijskog agenta na razini dizajna kako bi se lakše mogli postaviti u ulozi korisnika. [24]

Namjena	Upit korisnika	Entitet	Akcija agenta
Praksa	Pronađi mi praksu mrežnog administratora	Administrator, Mrežnog	Pronađi prakse koje sadrže naziv administrator, mrežni administrator ili sinonimi mrežnog administratora (kao što je mrežaš)

Tablica 1. Dokumentiranje akcija na temelju namjene i entiteta

6. Projektni zadatak

U ovom poglavlju opisivat će se korištene tehnologije i servisi kako bi se kreirao konverzacijski agent, te koraci koji su potrebni da bi isto bilo razvijeno. Koraci su sljedeći:

- Definiranje domene i ciljne skupine korisnika
- Definiranje cilja, funkcionalnosti aplikacije i dokumentiranje korisničkog iskustva koji se želi postići interakcijom korisnika sa web aplikacijom
- Definiranje arhitekture aplikacije
- Definiranje slijeda poruka između svih elemenata koji su definirani u arhitekturi aplikacije
- Definiranje i implementacija strukture baze podataka na temelju definiranih funkcionalnosti aplikacije
- Definiranje entiteta odnosno treniranih podataka za servis za obradu prirodnog jezika te preslika istih u bazu podataka
- Razvoj web sučelja s kojim je korisnik u interakciji
- Razvoj aplikacije na serverskoj strani koja komunicirati sa web sučeljem, servisom obrade prirodnog jezika i baze podataka
- Razvoj algoritma za generiranje odgovora i primjena korisničkog iskustva

Svaki od navedenih koraka biti će opisani u sljedećim potpoglavljima gdje će čitatelj moći vidjeti detaljni proces razrade istog.

6.1. Dokumentiranje cilja, funkcionalnosti i korisničkog iskustva

Cilj projekta je omogućiti korisniku pronalazak prakse u Republici Hrvatskoj nakon završetka školovanja. Što je ujedno i domena aplikacije – pronalazak prakse. Kako bi korisnik pronašao praksu komunicirati će sa konverzacijskim agentom predlažući korisniku tri najrelevantnija odgovora odnosno prakse na temelju definiranih korisničkih zahtjeva/interesa. Korisnik mora odgovoriti na dva najvažnija pitanja: gdje želi pronaći praksu, tj. lokacija i koje zanimanje prakse ga zanima. Ti

podaci su preduvjet za dohvaćanja podataka iz baze podataka odnosno identificiranje korisničkog interesa. Ukoliko isti nisu poznati, agent nije u stanju predložiti prakse korisniku, što je ujedno i nedostatak algoritma agenta! Funkcionalnosti projekta/aplikacije su sljedeći:

- prikaz dostupnih praksi na temelju korisničkih interesa kroz konverzacijski agent
- generiranje odgovora na temelju poznatih informacija i temelju korisničkih upita
- dohvaćanje podataka iz baze podataka na temelju korisničkih interesa
- obrada korisničkih upita odnosno obrada prirodnog jezika kako bi se identificirali interesi korisnika
- identificiranje potvrde korisnika na temelju postavljenih upita
- mogućnost pregleda praksi bez korisničke interakcije s konverzacijskim agentom

Osnovne funkcionalnosti agenta odnosno održavanje komunikacije između korisnika i agenta su:

- mogućnost pronalaska prakse na temelju jednog ili više parametara: grad, županija, zanimanje, specijalizacija, vještine, alati koje poznaje
- postavljanja potpitanja korisniku za preciznije definiranje njegovog problema ukoliko servis nije siguran što korisnik postavlja kao problematiku

U potpoglavlju *Kako dokumentirati korisničko iskustvo* opisana su dva predloška dokumentiranja ispunjenja korisničkog iskustva:

- “Ja kao <tip korisnika> želim <cilj> za postizanje <rezultata>” kako bi opisali moguće scenarije konverzacijskog agenta
- “Kada se dogodi <situacija> za danu <motivaciju> dobiti <rezultat>” kako bi opisali način obavljanja zadatka odnosno rješavanja korisničkog upita

Na temelju tih predložaka, domene aplikacije, funkcionalnosti i definiranih funkcionalnosti održavanja interakcije korisnika sa agentom, scenariji su sljedeći:

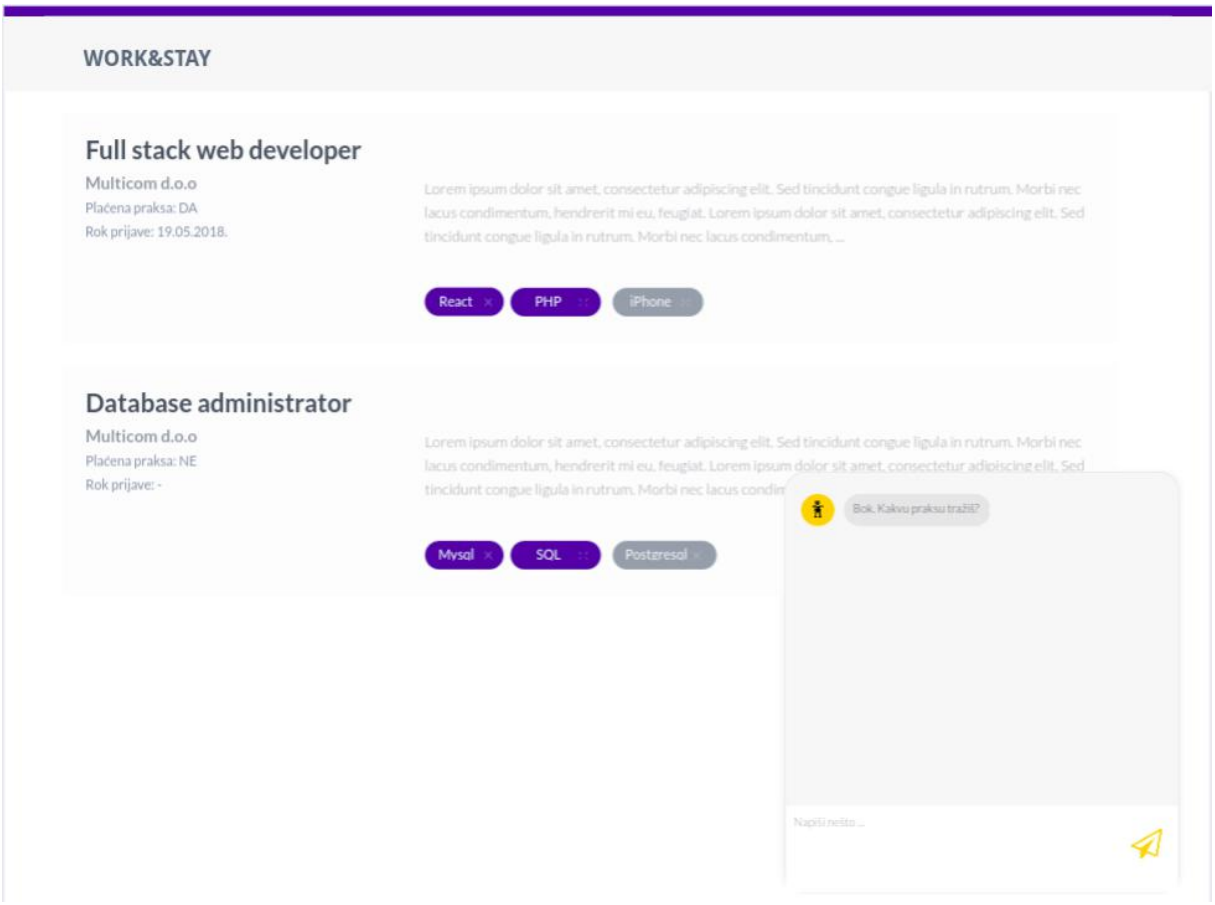
- “Ja kao učenik/školarac želim pronaći praksu u svojem gradu”

- “Ja kao učenik/školarac želim pronaći praksu u svojoj županiji”
- “Ja kao učenik/školarac želim pronaći praksu sa zanimanjem koji ću se u budućnosti baviti poslije školovanja”
- “Ja kao student želim pronaći praksu u gradu gdje idem na fakultet ili gdje živim”
- “Ja kao student želim pronaći praksu u županiji gdje idem na fakultet ili gdje živim”
- “Ja kao student želim pronaći praksu koja je plaćena”
- “Ja kao student želim pronaći praksu gdje se mogu specijalizirati”

Jedna od najbitnijih stavaka kod korisničkog sučelja je održavanje komunikacije s njim, kako bi to postigli definirani su sljedeći scenariji:

- “Kada ne mogu pronaći praksu u svojem gradu želim vidjeti prakse u svojoj županiji”
- “Kada nisam definirao specijalizaciju prakse želim da se pretraži na temelju mojih vještina”
- “Ako postoje prakse koje su plaćene, želim da ih agent predloži”

Kako bi se osigurao korisniku pregled dostupnih praksi bez interakcije konverzacijskog agenta definirano je sljedeće korisničko sučelje web aplikacije. Kroz cijelo web područje prikazuje se lista dostupnih praksi, a konverzacijski agent, kao opcija korisniku, smješten je na donjem desnom kutu korisničkog sučelja web aplikacije. Lista praksi mora sadržavati naziv radnog mjesta, naziv poduzeća, opis prakse, vještine koje korisnik mora znati prije nego se prijavljuje na praksu (označene ljubičastom bojom) te vještine koje su poželjne (označene sivom bojom), ali ne i preduvjet. Druge informacije koje objava može prikazivati je rok prijave na praksu te da li je plaćena. Konverzacijski agent može biti minimaliziran od strane korisnika kako bi lakše pregledavao listu praksi. Agent mora imati područje gdje će korisnik pisati svoj upit te područje gdje može vidjeti prethodni razgovor s agentom. Web aplikacija trenutno ne omogućava filtriranje listi praksi već se isto može dobiti interakcijom s agentom. Više o komponenti konverzacijskog agenta biti će opisano u potpoglavlju *React.js*.



Slika 3. Korisničko sučelje web stranice sa konverzacijskim agentom

6.2. Arhitektura projekta

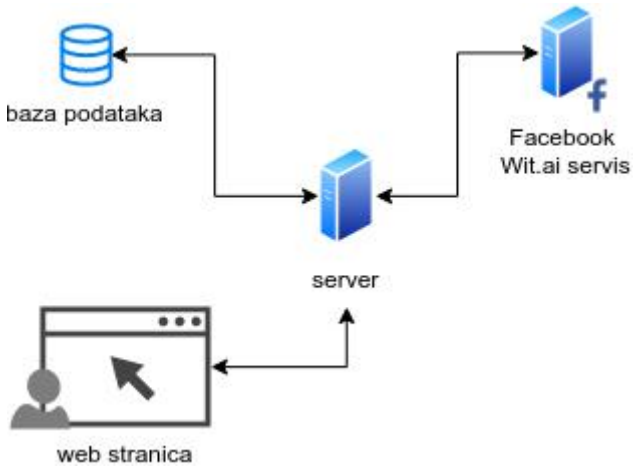
Na temelju zahtjeva opisanih u prethodnom potpoglavlju *Dokumentiranje cilja, funkcionalnosti i korisničkog iskustva*, aplikacija mora imati web sučelje kojim će korisnik biti u interakciji, server koji će primati korisničke upite, algoritam za generiranje odgovora, algoritam za obradu prirodnog jezika te bazu podataka. Na sljedećoj slici prikazana je arhitektura koja se sastoji od navedene baze podataka, servera, servisa koji provodi obradu prirodnog jezika i korisničkog preglednika.

Server, odnosno aplikacija na serveru, ima sljedeće funkcionalnosti:

- proslijediti korisnički upit servisu za obradu prirodnog jezika kako bi identificirao korisničke interese i korisničke potvrde na prethodno postavljena pitanja korisniku,
- dohvatiti podatke iz baze podataka na temelju korisnih interesa

- spremiti korisničke interese
- generirati odgovor
- poslati odgovor korisniku

Baza podataka sadrži objave dostupnih praksi u Republici Hrvatskoj te presliku entiteta iz *Wit.ai* servisa kako bi se lakše postaviti upiti za dohvaćanje željenih praksi. Na kraju web aplikacija koja se pokreće na korisničkom pregledniku mora komunicirati sa serverom tako što šalje i zaprima poruke istog. U sljedećem potpoglavlju *Slijed poruka* opisan je slijed poruka između web aplikacije i servera, servera i baze podataka te servera i servisa za obradu prirodnog jezika.



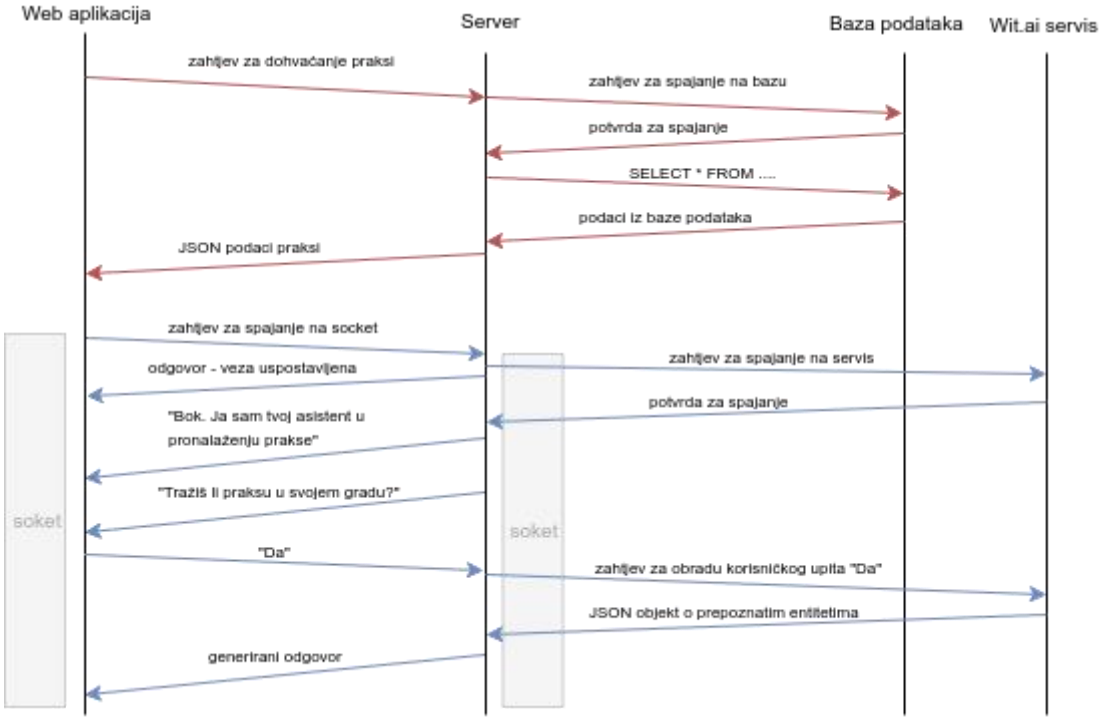
Slika 4. Arhitektura web aplikacije

6.3. Slijed poruka

U prethodnom potpoglavlju *Arhitektura projekta* spomenuta su četiri elementa koji čine arhitekturu projekta omogućavajući konverzacijski agent funkcionalnim. Kako oni zaista komuniciraju međusobno? Na sljedećoj slici prikazan je slijed poruka između njih. Kada korisnik otvori web aplikaciju, poslati će se dva zahtjeva prema serveru: zahtjev za dohvaćanje dostupne prakse (označeno crvenom bojom) i zahtjev za uspostavljanje dvosmjerne veze, tj. komunikacije u realnom vremenu (eng. *Real-time*) pomoću soketa. Razlog tome je što želimo osigurati da server u bilo kojem vremenu može poslati poruku web sučelju, tj. konverzacijskom agentu, bez da korisnik pošalje zahtjev za poruku/podacima [28]. Nakon što web aplikacija uspostavi vezu sa serverom preko soketa, server uspostavlja vezu sa servisom za obradu

prirodnog jezika - *Wit.ai*. Kada je veza uspostavljena server šalje generirane poruke konverzacijskom agentu kako bi uspostavio interakciju s korisnikom. Često konverzacijski agenti očekuju da korisnik prvi uspostavi komunikaciju s agentom, što ovdje nije slučaj! Glavni nedostatak ovog konverzacijskog agenta je što nema bazu znanja odnosno dovoljno treniranih podataka na temelju kojih može "pokriti" sve moguće scenarije koje korisnik postavlja, stoga želimo da agent definira slijed komunikacije s korisnikom, a ne obrnuto. Detaljnije o navedenom problemu govoriti će se u potpoglavlju *Generiranje odgovora*. Oba zahtjeva koja se šalju prema serveru, te zahtjev koji se šalje prema *Wit.ai* servisu i bazi podataka su asinkrona. To znači ako server šalje zahtjev bazi podataka, a potom *Wit.ai* servisu, ne čeka na odgovor baze podataka kako bi nastavo sa slijedom izvršavanja ostalih zahtjeva/koda, ako postoje.

Podaci iz baze podataka i *Wit.ai* servisa šalju se u JSON (eng. *JavaScript Object Notation*) formatu. JSON je format sačinjen od para ključa i vrijednosti koji se jednostavno može pretvoriti u *Javascript* objekte i obrnuto, što ga čini odličnim kandidatom za razmjenu poruka između servera i web aplikacije jer su oba pisana u *Javascript* programskoj jeziku [29].



Slika 5. Slijed poruka između web aplikacije, servera, baze podataka i servisa za obradu prirodnog jezika

6.4. Baza podataka

Baza podataka služi kako bi *Node.js* server dohvaćao podatke, o slobodnim praksama na koje se korisnik može prijaviti, temeljeno na korisničkom upitu prema konverzacijskom agentu. Postoje sljedeće relacije:

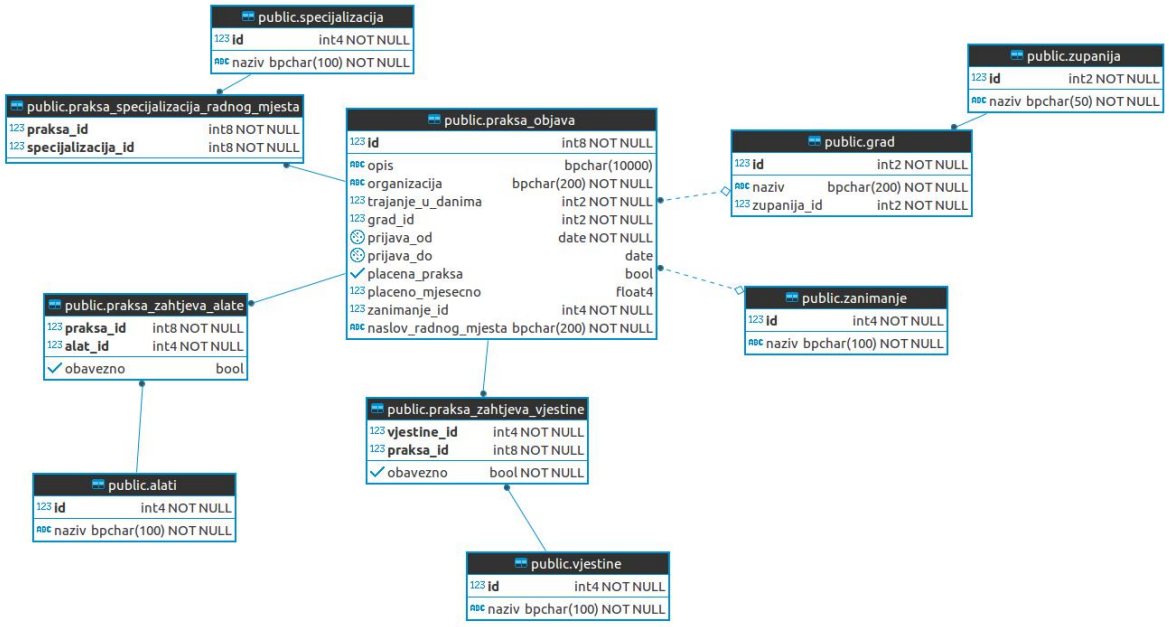
- zanimanje - skup profesionalnih zanimanja
- specijalizacija - skup specijalizirani zanimanja
- županija - definira županije RH
- grad - definira u kojem gradu je praksa dostupna
- vještine - skup vještina koje praksa može sadržavati
- praksa_zahtjeva_vještine - definira koje vještine praksa zahtjeva od korisnika
- alati - skup alata koje praksa može zahtijevati
- praksa_zahtjeva_alate - definira koje alate praksa zahtjeva od korisnika
- praksa_objava - sadrži skup objavi praksi
- praksa_specijalizacija_radnog_mjesta - jedno radno mjesto može imati definirano više specijalizacija. na primjer full stack web (developer), full stack software (developer)

Željene veze odnosno odnosi među relacijama su sljedeći:

- grad može pripadati jednoj županiji, a jedna županija može imati više gradova
- jedan alat može biti obavezna u više praksi, a jedna praksa može definirati više alata koje korisnik mora znati koristiti
- jedna vještina može biti obavezna u više praksi, a jedna praksa može definirati više vještina koje korisnik mora imati znanja
- jedna praksa može imati definirano jedno zanimanje, a jedno zanimanje može imati više praksi
- radno mjesto (prakse) može biti definirano više specijalizacija, i jedna specijalizacija može biti pridružena u više radnih mjestima

Na sljedećoj slici prikazan je *ERA* model (eng. *Entity-relationship model*) baze podataka sa pripadajućom shemom. Baza podataka mora sadržavati relacije popunjeni podacima koje sadrži servis *Wit.ai*, tj. mora sadržavati presliku entiteta. Time je proces pretraživanja praksi, na temelju identificiranih entiteta korisničkog upita, olakšan. Problematika koja se javlja je što mnoga radna mjesta imaju definirane višestruke specijalizacije. Jedna od primjera je *full stack web* programer gdje specijalizacija *full stack* obuhvaća *backend* i *frontend,* a *web* označava specijalizaciju za izradu web stranica. Jedini način rješavanja tog problema je kreiranje relacije *praksa_specijalizacija_radnog_mjesta* koja će sadržavati sve entitete specijalizacije za danu praksu.

Važno je napomenuti, kako bi agent pronašao praksu korisniku mora znati dvije informacije: lokacija i zanimanje. Ukoliko jedna od tih informacija nedostaje agent nije sposoban naći prakse korisniku!



Slika 6. ERA model baze podataka

6.5. *Wit.ai* servis za obradu prirodnog jezika

U poglavlju *4. Namjene i entiteti u Wit.ai servisu* opisana je pogledna strategija na temelju koje biti kreirani različiti tipovi entiteta uzimajući u obzir domenu web aplikacije. Tipovi entiteta prate sljedeću legendu:

- **S** - slobodni entitet
- **K** - entitet ključnih riječi
- **N** - namjena odnosno trenirani entitet

Naziv entiteta	Vrsta entiteta	Primjeri treniranih podataka	Opis
Zanimanje	**S**,**K**	Tester, podrška, dizajner, voditelj, inženjer, programer	Posao s kojim se korisnik želi baviti odnosno za koji traži praksu
Specijalizacija	**S**, **K**	Baze, hardware, web, mobile, frontend, backend	Detaljniji opis zanimanja odnosno specifičnog područja koji se odnosi na dano zanimanje
Namjena	**N**	Praksa	Ovaj entitet u pravilu nije potreban jer se domena aplikacije odnosi samo na prakse.
Vještine_alati	**K**	Mysql, postgresql	Alati koje praksa zahtjeva da korisnik istog zna prije prijave na isto
Vještine_prog_jezik	**K**	Javascript, java, php, c++, c#	Programski jezici koje praksa zahtjeva da korisnik istog zna prije prijave na isto
Vještine_biblioteke	**K**	React, angular, flask, express	Biblioteke koje praksa zahtjeva da korisnik istog zna prije prijave na isto

Potvrde_korisnika	**S, K**	Ok, Da, Jesam, Želim, Naravno	Identificiranje korisničkih odgovora kao potvrde. Ako agent upita korisnika da li želi praksu u svojem gradu, tada moramo i identificirati da li je korisnik potvrdio isto

Tablica 2. Entiteti definirani u Wit.ai servisu

U tablici možemo vidjeti kako su entiteti *zanimanje* i *specijalizacija* definirani kao slobodni entitet i entitet ključnih riječi. Razlog tome je mogućih identificiranja deklinacijskih oblika riječi (ovisno o padežu imenice). Ako korisnik pošalje upit "Tražim poziciju web testera" tada će *testera* prepoznati kao *tester* odnosno kao entitet *zanimanje.* Ukoliko entitet *zanimanje* postavimo kao entitet ključnih riječi, u tom slučaju prepoznavanje istog se neće dogoditi! Entitet *namjena* označava da li korisnički upit predstavlja namjenu *praksa.* Da li je to uopće potrebno? Domena aplikacije je pronalazak prakse, zašto onda imamo takav entitet? Razlog tome je jednostavan - mogućnost lakšeg nadograđivanja sustava sa novim namjenama. Na primjer, ako dođe do potrebe dodavanja nove namjene - *stručno osposobljavanje* tada bi se morali analizirati svi postojeći podaci u servisu odnosno korisnički upiti. Analiziranje istog zahtjeva mnogo vremena. Jedini način spriječavanja istog je treniranje podataka od trenutka kada su oni prvi put zaprimljeni na servisu.

6.6. *React.js* biblioteka za kreiranje korisničkog sučelja

React.js je *Javascript* biblioteka koja omogućava kreiranje modularnih komponenata web sučelja. To znači da jedna dizajnirana komponenta može biti korištena u različitim prilikama. Na primjer, komponenta za prikaz korisničkih poruka može također biti korištena za prikaz vještina koje praksa zahtjeva od korisnika. Jedino po čemu se razlikuju je boja teksta i boja labele. Osim toga, svaka komponenta može sadržavati stanja na temelju kojih će mijenjati korisničko sučelje ili korisničko iskustvo. Na temelju prethodnog primjera to bi značilo da ista komponenta unutar konverzacijskog agenta morala bi imati dva stanja: stanje kada se prikazuje korisnički upit i stanje kada se prikazuje agentov odgovor/upit korisniku. Na temelju tog stanja komponenta je u stanju promijeniti boju ili neku drugu stilsku vrijednost odnosno CSS (eng. *Cascading Style Sheets*). [4]

U potpoglavlju *Dokumentiranje cilja, funkcionalnosti i korisničkog iskustva* prikazana je slika web sučelja koje je programirano *React.js* bibliotekom. Web sučelje se sastoji od nekoliko komponenti. Glavne komponente web sučelja su *Chatbot* (u prijevodu konverzacijski agent) i *Homepage* (u prijevodu naslovna stranica). *Homepage* komponenta se sastoji od list praksi gdje svaka objava predstavlja zasebnu komponentu *Post* (hrv. *Objava*). Na sljedećoj slici prikazan je programski kod u kojem možemo vidjeti spomenute komponente. *Homepage* komponenta prima objave (eng. *Posts*) koje će prikazati na web sučelju, dok *Chatbot* komponenta prima poruke (eng. *Messages*) koje će prikazati u konverzacijskom agentu. *Chatbot* komponenta ima definirani slušatelj događaja (eng. *Event listener*) koji će se okinuti svaki put kada korisnik pritisne gumb za slanje poruke ili kada korisnik pritisne tipku *Enter*.

```
render() {
  const {messages, posts} = this.state;

  return (
    <div className="App">

      <Homepage posts={posts} />
      <Chatbot
        messages={messages}
        onSend={this.onMessageSend} />
    </div>
  );
}
```

Slika 7. Programski kod koji prikazuje glavne komponetne web sučelja

U potpoglavlju *Slijed poruka* opisana je komunikacija izmedu web sučelja i servera gdje možemo vidjeti kako web sučelje nema direktnu komunikaciju sa servisom za obradu prirodnog jezika i bazom podataka, vec server dostupa navedenim kako bi osigurao potrebne podatke za rad web sučelja odnosno konverzacijskog agenta. U sljedećem kodu prikazano je slanje zahtjeva za konekciju sa serverom te čekanje na odgovor istog. Za kreiranje soketa korištena je *Socket.io [26]* biblioteka kako bi se uspostavila dvosmjerna veza sa serverom te zaprima i slala zahtjeve od odnosno prema serveru. Kada web sučelje zaprimi odgovor od servera (linija koda 67 na slici 8.), kreira novu instancu *Message* koja za parametre prima poruku koju je server generirao i zastavicu koja definira da je poruka kreirana od strane agenta, tj. algoritma za generiranje odgovora.

Zahtjeve koje komponenta šalje ili prima su poruke JSON formata koje će se prikazati na komponenti konverzacijskog agenta (slika 3) i dobiveni podaci prakse biti će prikazani u listi web sučelja (komponenti *Homepage*). Drugi zahtjev koji se šalje prema serveru je dohvaćanje svih dostupnih praksi s ciljem da je komunikacija s konverzacijskim agentom opcionalna, korištena je uz pomoć *Axios* [33] biblioteke koja omogućava asinkrone zahtjeve prema serveru (linija koda 78 na slici 8.). Tim pristupom, gdje korisnika ne obavezujemo da komunicira s agentom, smanjujemo nezadovoljstvo korisnika ukoliko agent u određenim scenarijima nije sposoban obraditi odnosno identificirati korisnički problem ili generirati odgovor korisniku. Svaki primljeni odgovor od strane servera će ažurirati stanje *Homepage* i *Chatbot* pozivajući metodu *setState*.

```
46      componentDidMount() {
47        this.socket = io(SERVER_BASE_DEV, {});
50
51        if (this.socket) {
52          this.socket.on('connect', () => {});
55
56          this.socket.on('connect_err', (err) => {})
59
60          this.socket.on('disconnect', () => {});
65
66          /* listen for msgs send by bot and update state */
67          this.socket.on(SOCKET_ACTIONS.MSG_RESP, (data) => {
68            if (typeof data == "string") {
69              const response = new Message(data, MSG_SENDER.BOT);
70              const {messages} = this.state;
71              this.setState({
72                messages: [...messages, response]
73              })
74            }
75          });
76        }
77
78        axios.get(SERVER_BASE_DEV+'posts-all')
79             .then(res => {
80               this.setState({
81                 posts: res.data
82               })
83             })
84             .catch(err => {
85               console.error(err)
86             })
87      }
```

Slika 8. Primjer programskog koda u kojem web sučelje pokušava uspostaviti komunikaciju s serverom preko soketa i slanje asinkronog upita za dohvaćanje svih praksi

Komponenta konverzacijskog agenta sastoji se od dvije podkomponente PostSpace i SpaceBottom, koji se mogu vidjeti na sljedećoj slici. *PostSpace* komponenta prikazuje razmijenjene poruke između korisnika i agenta, dok *SpaceBottom* sadrži prostor za pisanje poruke te gumba za slanje poruke. Kada korisnik pritisne na gumb okinuti će se slušatelj događaja koji će poslati nadležnoj

komponenti (svojem roditelju) događaj, u tom slučaju *Chatbot* odnosno metodi *onMessageSend* prikazanoj na slici 7.

Slika 9. Komponente od kojih je komponenta konverzacijskog agenta sačinjena

Kada govorimo o korisničkom iskustvu, ono se ne mora smatrati samo iskustvu koji korisnik doživljava kada je u interakciji sa web sučeljem. Na temelju dokumentiranih scenarija poboljšanja korisničkog iskustva, navedeni u potpoglavlju *Dokumentiranje cilja, funkcionalnosti i korisničkog iskustva* i primjenom korisničkog iskustvo opisanim u petom poglavlju, korisničko iskustvo se može postići održavanje komunikacije korisnika i agenta. U tom slučaju, algoritam generiranja odgovora vodi brigu o održavanju napomenuti komunikacije, koji je opisan u potpoglavlju *Generiranje odgovora*.

6.7. *Node.js* server

Server kao glavni entitet u arhitekturi komunicira sa svim entitetima u predstavljenoj arhitekturi potpoglavlja *Arhitektura projekta*. U potpoglavlju *Slijed* poruka možemo vidjeti kako server ima pet zadaća:

- zaprimanje korisničkih upita poslani od strane web sučelja
- slanje korisnički upit *Wit.ai* servisu za obradu prirodnog jezika

- slanje odgovora korisničkom sučelju
- generiranje odgovora na korisnički upit (koristeći algoritam opisan u sljedećem potpoglavlju *Algoritam generiranja odgovora)*
- dohvaćanje podataka iz baze podataka

Na slici 5. potpoglavlja *Slijed poruka* možemo vidjeti kako server sa korisničkim sustavom komunicira pomoću soketa kako bi slanje poruka konverzacijskom agentu bila u realnom vremenu (eng. *Real-time*). Na sljedećoj slici prikazan je programski kod u kojem je kreiran soket koji će na uspostavu veze sa web sučeljem generirati dvije poruke metodom *generateGreeting* (linija koda 31 na slici 10). Ta metoda će generirati dva odgovora koji će konverzacijski agent prikazati korisniku. Više o generiranju odgovora opisano je u sljedećem potpoglavlju *Algoritam generiranja odgovora* koji opisuje na koji način algoritam radi. Nakon toga, server postavlja slušač koji će se okidati svaki put kada web sučelje pošalje događaj *NEW_MSG* odnosno nova poruka od strane korisnika. Svaka dobivena poruka pozvati će instancu generator (algoritam za generiranje odgovora) koji će obraditi korisnički upit na način da će poslati zahtjev servisu za obradu prirodnog jezika te generirati odgovor na temelju identificiranih entiteta, te na kraju poslati odgovor nazad web sučelju (linija kod 46 slika 10).

```
28  const generator = new responseGenerator();
29
30  io.on("connect", function(socket) {
31    const greetings = generator.generateGreeting();
32    socket.emit("MSG_RESP", greetings);
33    const firstTopic = generator.generateResponse();
34    socket.emit("MSG_RESP", firstTopic);
35
36    socket.on("disconnect", function() {…})
40
41    socket.on("NEW_MSG", function(data) {
42      new Promise(function(resolve, reject) {
43        generator.processClientResponse(resolve, reject, data, client)
44      }).then(() => {
45        const newResponse = generator.generateResponse();
46        socket.emit("MSG_RESP", newResponse);
47      }).catch(() => {
48        const newResponse = "Nažalost ne razumijem tvoj zahtjev.";
49        socket.emit("MSG_RESP", newResponse);
50      })
51    });
52  });
```

Slika 10. Programski kod koji prikazuje kako soket “sluša” zahtjeve web sučelja te šalje odgovore na njih

Nadalje, drugi zahtjev koji server može zaprimiti od web sučelja je dohvaćanje dostupnih praksi iz baze podataka. Server u tom slučaju definira rutu (eng. *Path*) */posts-all* kojem web sučelje može poslati zahtjev, odnosno URL (eng. *Uniform Resource Locator*) [34] *http://127.0.0.1/posts-all* jer je server pokrenut na lokalnom računalu (nije dostupan van lokalne mreže). Ukoliko je aplikacija pokrenuta na serveru koja je dostupna javnoj mreži odnosno Internetu, tada će URL za isto biti sličan gdje umjesto IP adrese (eng. *Internet Protocol address*) [35] biti će navedena domena servera. Prikazani kod na sljedećoj slici šalje upit bazi podataka prosljeđujući upit odnosno varijablu *queries.all_posts*. Upit koji se šalje serveru prikazan je na slici 12.

```
43    app.get("/posts-all", (req, res) => {
44      const db = new database();
45
46      new Promise(function(resolve, reject) {
47        db.doQuery(resolve, reject, queries.all_posts)
48      }).then(data => {
49        res.json(data);
50      }).catch(err => {
51        res.sendStatus(500);
52      });
53    })
```

Slika 11. Kreirana ruta na kojoj web sučelje može dostupiti podacima iz baze podataka za dohvaćanje dostupnih praksi

Upit koji se postavlja bazi podataka dohvaća podatke iz relacije *praksa_objava,* prikazana u *ERA* modelu potpoglavlja *Baza podataka*, koji su potrebni web sučelju kako bi prikazao listu dostupnih praksi, prikazana na slici 3. potpoglavlja *Dokumentiranje cilja, funkcionalnosti i korisničkog sučelja*. Na liniji koda 7-14 dohvaćaju se vještine, određene prakse, koje su u relaciji *praksa_zahtjeva_vjestine* označene kao obavezne. Sve ostale vještine, koje nisu označene kao obavezne, definirane su kao ostale vještine koje praksa zahtjeva. Više o poslovnoj logici opisano je u potpoglavlju *Dokumentiranje cilja, funkcionalnosti i korisničkog sučelja.*

```
const all_posts = "
  SELECT
    praksa_objava.id, opis, organizacija,
    trajanje_u_danima, prijava_od, prijava_do,
    placena_praksa, naslov_radnog_mjesta,
    zanimanje.naziv as zanimanje, grad.naziv as grad,
    array_to_string(array(
      SELECT ltrim(naziv, ' ' )
      FROM praksa_zahtjeva_vjestine
      LEFT JOIN vjestine
        ON praksa_zahtjeva_vjestine.vjestine_id = vjestine.id
      WHERE obavezno = TRUE
        AND praksa_objava.id = praksa_zahtjeva_vjestine.praksa_id
    ), ';') as obavezne_vjestine,
    array_to_string(array(
      SELECT ltrim(naziv, ' ')
      FROM praksa_zahtjeva_vjestine
      LEFT JOIN vjestine
        ON praksa_zahtjeva_vjestine.vjestine_id = vjestine.id
      WHERE (obavezno IS NULL OR obavezno = FALSE)
        AND praksa_objava.id = praksa_zahtjeva_vjestine.praksa_id
    ), ';') as ostale_vjestine
    FROM praksa_objava
    LEFT JOIN zanimanje
      ON zanimanje.id = praksa_objava.zanimanje_id
    LEFT JOIN grad
      ON grad.id = praksa_objava.grad_id";
```

Slika 12. Primjer upita za dohvaćanje svih dostupnih praksi

Na slici 10. možemo vidjeti metodu *responseGenerator* inicijalizira klasu koja ima zadaću da na temelju poznatih korisničkih interesa generira odgovor bilo ono upit na novu temu ili odgovaranje na postavljeni upit korisnika. Kako je navedeni algoritam "svjestan" već poznatih interesa korisnika opisano je u sljedećem potpoglavlju.

6.7.1. Algoritam generiranja odgovora

Na temelju dobre prakse korisničkog iskustva, opisanim u petom poglavlju, algoritam generiranja odgovora mora znati riješiti dva problema:

- Održavanje komunikacije sa korisnikom
- Generirati odgovor na temelju njegovih interesa odnosno postavljati pitanja na koje je korisnik već odgovorio.

Prvi problem može se riješiti na način da konverzacijski agent usmjerava korisnika na određenu temu, a ne obrnuto. Time se može spriječiti, ali ne isključiti, scenarij u kojem servis za obradu prirodnog jezika nije u stanju prepoznati korisničku

namjenu ili entitete odnosno algoritam generiranja odgovora nije u stanju odgovoriti na postavljeni upit. Rješenje drugog problema može biti sljedeće: svaki put kada korisnik pošalje upit agentu, bez obzira da li odgovara na postavljeno pitanje agenta prema korisniku, proslijediti će se servisu za obradu prirodnog jezika. Ukoliko servis za obradu prirodnog jezika identificira entitet isti će se spremiti na serveru na temelju kojeg će algoritam za generiranje odgovora biti "svjestan" dostupnih informacija te time generirati sljedeće pitanje (dodatne informacije ukoliko je potrebno) ili odgovor korisniku (vratiti listu praksi).

U poglavlju *Mogućnost razumjevanja i rješavanja problema* predstavljena su tri modela generiranja odgovora. Za potrebe diplomskog rada koristi će se model temeljen na pravilima. Model temeljen na pravilima ima dva moguća pristupa generiranja odgovora: korištenjem skriptnog jezika i korištenje proširivog jezika za označavanje (eng. *Extensible Markup Language*). Ti pristupi se baziraju na cijelim rečenica, definirajući njihov slijed (*AIML* [32] i *RiveScript* [30]) ili mogućih ključnih riječi (*ChatScript* [31]) koji se mogu pojaviti u korisničkom upitu, ne podržavajući scenarij u kojem se slijede definirana pitanja agenta prema korisniku. To znači da navedeni pristupi funkcioniraju samo u slučaju kada korisnik inicira komunikaciju s agentom (postavljajući pitanje istom), kojem će pronaći odgovarajući uzorak i odgovoriti na njega. Drugo, pristupi imaju ograničenja kod definiranja uzoraka koja mogu biti prepreka kod kompleksnih generiranja odgovora ili detektiranja istog. Treće, pristupi nemaju mogućnost okidanja događaja u kojem će servis poslati podatke iz baze podataka (dostupne prakse).

Kako bi se navedeni problemi zaobišli kreiran je algoritam koji definira slijed postavljanja pitanja koji se dinamički odabire. Pitanja koja će se dinamički postavljati korisniku definirani su zasebnoj datoteci koja sadrži set tema (eng. *Topic*) odnosno set pitanja koje agent može postaviti korisniku. Na sljedećoj slici prikazana je tema lokacije u JSON formatu. Svako pitanje ima zastavice koje označavaju da li će se krenuti na sljedeću temu nakon korisničke potvrde ili će ići u dubinsku raspravu s korisnikom. Na slici možemo vidjeti kako za pitanje lokacije za grad postoje dvije dubine. Prva dubina je osnovno pitanje koje će se postaviti ukoliko lokacija nije poznata, a dubina dva se postavlja korisniku ukoliko je korisnik na prethodno pitanje potvrdno odgovorio. U tom slučaju scenarij će biti sljedeći:

Agent: "Želiš li pronaći praksu u svojem gradu?" [Dubina 1]

Korisnik: “Da”

Agent: “U kojem gradu tražiš prakse?” [Dubina 2]

Korisnik: “Bjelovar”

Ukoliko korisnik na prvo pitanje odgovori negativno tada se preskače na sljedeću temu, na primjer na zanimanje radnog mjesta (prakse). Pitanja su postavljena na način da korisnik odgovara na njih potvrdno ili usmjereno na način da odaje samo informacije koje su potrebne za pronalaženje odgovarajućih praksi.

```
38  const LocationCity = [
39    {
40      question: "Želiš li pronaći praksu u svojem gradu?",
41      expect: TYPE_OF_QUESTION.CONFIRMATION,
42      deep: [1],
43      onConfirm: TYPE_OF_ACTIONS.GO_DEEP
44    },
45    {
46      question: "Tražiš li praksu u gradu u kojem živiš?",
47      expect: TYPE_OF_QUESTION.CONFIRMATION,
48      deep: [1],
49      onConfirm: TYPE_OF_ACTIONS.GO_DEEP
50    },
51    {
52      question: "U kojem gradu tražiš prasku?",
53      expect: TYPE_OF_QUESTION.INPUT,
54      deep: [1, 2],
55      onConfirm: TYPE_OF_ACTIONS.GO_TO_NEXT_QUESTION
56    }
57  ];
```

Slika 13. Definirana tema lokacije na temelju kojeg se generiraju upiti/odgovori korisniku

7. Kritički prikaz

Nakon 69 godina, od kako je predstavljen prvi konverzacijski agent, nisu riješene problematike obrade prirodnog jezika i generiranje odgovora korisniku, što čini razvoj istih težim i često nemogućim. Kroz projektni zadatak diplomskog rada može se vidjeti razvoj jednostavnog konverzacijskog agenta koji ne koristi postojeću bazu znanja na temelju kojeg će se moći trenirati podaci strojnim i/ili dubinskim učenjem i/ili umjetnom inteligencijom, što zahtjeva posebnu pažnju rješavanja spomenutih problematika konverzacijskih agenata. Osim toga konverzacijski agent ne čini samo razmjena smislenih poruka s korisnikom već u tu priču ulazi korisničko iskustvo odnosno kakve stavove, emocije, razmišljanja korisnik proživljava tijekom korištenja istog te da li je jednostavna i pristupačna za korisnika koja će zadržati korisničku pažnju odnosno interakciju s agentom.

Kako bi se pristupilo rješavanju tih problema najveću pažnju trebalo bi posvetiti korisničkom iskustvu kako bi korisnika, bez obzira na nepotpuno rješavanje njegovog problema, zadržali i pokušali navesti korisnika na usmjerene razgovore za koje je agent treniran. Takav pristup moguće je ostvariti na način da agent usmjeruje korisnika na pitanja na koje korisnik potvrđuje ili negira ili odgovara kratkim rečenicama kojima se može identificirati entiteti odnosno ključne riječi koje će agent moći razumjeti. Naravno, takav pristup moguće je primjeniti samo na zatvorene domene.

8. Zaključak

Tema ovog rada bila je razviti konverzacijski agent na odabranu domenu kako bi se opisali koraci razvoja agenta od njegovog dizajna, korisničkog iskustva, definiranja arhitekture te njegove realizacije. U prvom dijelu diplomskog rada opisivao je što je konverzacijski agent, koje funkcionalnosti konverzacijskog agenta čini kompleksnijim te kako primjeniti i dokumentirati korisničko iskustvo u razvoju agenta kako bi se održala komunikacija s korisnikom. Drugi dio diplomskog rada opisuje domenu koju pokriva razvijeni konverzacijski agent, koji su koraci bili potrebni za njegovu realizaciju te koji problemi su se pojavili kod razvoja istog i na koji način su se isti riješili.

Konverzacijski agenti, ne implementirajući umjetnu inteligenciju, duboko učenje i strojno učenje, mogu biti složeni iako je njegova zatvorena domena usko definirana. Problematika svakog agenta je baza znanja koja mora sadržavati kvalitetno trenirane podatke. Ukoliko agent isto ne sadrži, pitanje je da li agent može kvalitetno obavljati svoje zadaće koje se odnose na rješavanje korisničkih problema. Čak ako agent ima kvalitetno trenirane podatke upitno je da li algoritam za obradu prirodnog jezika je u stanju identificirati korisnički problem, što dan danas predstavlja problematiku u ovom području.

Popis literature

[1.] M. Sanjeevi, “Chapter 11: ChatBots to Question & Answer systems”, 2018. [Na internetu]. Dostupno: https://medium.com/deep-math-machine-learning-ai/chapter-11-chatbots-to-question-answer-systems-e06c648ac22a [pristupano 28.06.2019.]

[2.] Cahn, J., Chatbot: Architecture, Design, & Development [Završni rad]. University of Pennsylvania, SAD, 2017, Dostupno: https://s3.amazonaws.com/academia.edu.documents/57035006/CHATBOT_thesis_final.pdf [pristupano 28.06.2019.]

[3.] J. Baron, “2017 Messenger Bot Landscape, a Public Spreadsheet Gathering 1000+ Messenger Bots”, 2017. [Na internetu]. Dostupno: https://cai.tools.sap/blog/2017-messenger-bot-landscape/ [pristupano 28.6.2019.]

[4.] Facebook. React.js (verzija 16.8) (2019) [Na internetu]. Dostupno: https://reactjs.org/ [pristupano 28.6.2019.]

[5.] Facebook. Wit.ai (verzija 4.1.0). [Na internetu]. Dostupno: https://wit.ai/ [pristupano 28.6.2019.]

[6.] R. Dahl. Node.js (verzija 10.16.0). [Na internetu]. Dostupno: https://nodejs.org/ [pristupano 28.6.2019.]

[7.] S. Kojouharov, “Ultimate Guide to Leveraging NLP & Machine Learning for your Chatbot”, 2016. [Na internetu, Slika]. Dostupno: https://chatbotslife.com/ultimate-guide-to-leveraging-nlp-machine-learning-for-you-chatbot-531ff2dd870c [pristupano 28.06.2019.]

[8.] Sonix (bez dat.) *What's the difference between artificial intelligence (AI), machine learning (ML) and natural language processing (NLP)?* [Na internetu]. Dostupno: https://sonix.ai/articles/difference-between-artificial-intelligence-machine-learning-and-natural-language-processing [pristupano 29.6.2019.]

[9.] Facebook. Duckling (verzija 0.1.6.1). [Na internetu]. Dostupno: https://duckling.wit.ai/#date-and-time-implementation [pristupano 16.8.2019.]

[10.] Microsoft (2018.) *LUIS quick start with list entities*. [Na internetu]. Dostupno: https://blog.botframework.com/2018/01/16/luis-quick-start-list-entities/ [pristupano 16.8.2019]

[11.] Rasa. Rasa (verzija 1.2.3). [Na internetu]. Dostupno: https://rasa.com/docs/ [pristupano 16.8.2019]

[12.] Botpress. Botpress (verzija 12.1.0). [Na internetu]. Dostupno: https://botpress.io/docs/main/nlu/#using-entities [pristupano 16.8.2019]

[13.] Google. Dialogflow (verzija 1.14.0). [Na internetu]. Dostupno: https://cloud.google.com/dialogflow/docs/concepts [pristupano 16.8.2019]

[14.] "Umjetna inteligencija" (bez dat.). u Enciklopedija, *Leksikografski zavod Miroslava Krleža*. Dostupno: http://www.enciklopedija.hr/natuknica.aspx?ID=63150# [pristupano 21.8.2019]

[15.] Expert System (bez dat.) *What is Machine Learning? A definition* [Na internetu]. Dostupno: https://www.expertsystem.com/machine-learning-definition/ [pristupano 21.8.2019]

[16.] PCChip (2018.) Razlike izmedu strojnog ucenja, AI i "dubokog" ucenja. [Na internetu]. Dostupno: https://pcchip.hr/ostalo/tech/razlike-izmedu-strojnog-ucenja-ai-i-dubokog-ucenja/ [pristupano 30.8.2019]

[17.] Ghosh, P. "Machine learning vs. Deep learning", 2018. [Na internetu] Dostupno: https://www.dataversity.net/machine-learning-vs-deep-learning/ [pristupano 30.8.2019]

[18.] Robarts, K. "Differences between machine learning and deep learning", 2018. [Na internetu] Dostupno: http://www.differencebetween.net/technology/differences-between-machine-learning-and-deep-learning/ [pristupano 30.8.2019.]

[19.] Surlyadeepan, R. "Chatbots with Seq2Seq: Learn to build a chatbot using TensorFlow", 2016. [Na internetu] Dostupno:http://complx.me/2016-06-28-easy-seq2seq/ [pristupano 31.8.2019]

[20.] Lanoue, S. “UI vs. UX: What’s the difference between user interface and user experience? : What’s the difference between UI and UX”, 2016. [Na internetu] Dostupno: https://www.usertesting.com/blog/ui-vs-ux/ [pristupano 31.8.2019]

[21.] Philips, C. “The Ultimate chatbot UX design strategy for 2019: What a chatbot needs to be successful in 2019”, 2019. [Na internetu] Dostupno: https://chatbotsmagazine.com/the-ultimate-chatbot-ux-design-strategy-for-2019-b77b85e36f5a [pristupano 31.8.2019]

[22.] “13 essential UX principles to build a great chatbot” (12.12.2018). Medium [Na internetu]. Dostupno: https://chatbotslife.com/13-essential-ux-principles-to-build-a-great-chatbot-c8c2006961b [pristupano 31.8.2019]

[23.] Pandey, A. “Chatbot UX - Design Tips and Considerations”, 2018. [Na internetu]. Dostupno: https://medium.com/aitareyagency/chatbot-ux-design-tips-and-considerations-84b19afb88a7 [pristupano 31.8.2019]

[24.] Lazarevich, K. “How to document chatbot requirements”, 2018. [Na internetu]. Dostupno: https://chatbotsmagazine.com/how-to-document-chatbot-requirements-7df81275cc66?gi=78e8c0e66805 [pristupano 31.8.2019]

[25.] Shevat, A., *Designing Bots: Creating conversational experience*. O’Reilly Media, CA, USA. 2017

[26.] Socket.io. Socket (verzija 2.0). [Na internetu]. Dostupno: https://socket.io/ [pristupano 15.9.2019.]

[27.] PgAdmin. PgAdmin (verzija 4.12). [Na internetu]. Dostupno: https://www.pgadmin.org/ [pristupano 15.9.2019]

[28.] J. Tomić, WebSocket protokol [Seminar]. Fakultet elektrotehnike i računarstva, Zagreb, Sveučilište u Zagrebu, 2018, Dostupno: http://nevena.lss.hr/recordings/fer/predmeti/racfor/2018/seminari/jtomic/seminar.pdf [pristupano 15.9.2019]

[29.] IETF. The JavaScript Object Notation (JSON) Data Interchange Format. (2014) [Na internetu]. Dostupno: https://tools.ietf.org/html/rfc7159 (pristupano 15.9.2019.)

[30.] RiveScript. RiveScript-JS (verzija 2.0.0) (2019) [Na internetu]. Dostupno: https://www.rivescript.com [pristupano 15.9.2019]

[31.] ChatScript. ChatScript.js (verzija 9.70) (2019) [Na internetu]. Dostupno: https://github.com/ChatScript/ChatScript/releases [pristupano 15.9.2019]

[32.] Aiml fundation. Arificial Intelligence Markup Language (verzija 2.1) (2018) [Na internetu]. Dostupno: http://www.aiml.foundation/ [pristupano 15.9.2019]

[33.] Axios. Axios. (verzija 0.18.1) (2019) [Na internetu]. Dostupno: https://github.com/axios/axios [pristupano 15.9.2019]

[34.] IETF. Uniform Resource Identifier (URI): Generic Syntax. (2005) [Na internetu]. Dostupno: https://tools.ietf.org/html/rfc3986 (pristupano 16.9.2019.)

[35.] IETF. Internet Official Protocol Standards. (2008) [Na internetu]. Dostupno: https://tools.ietf.org/html/std1 (pristupano 16.9.2019)

[36.] Marvel HQ. Marvel (verzija 2.0) (2017) [Web aplikacija]. Dostupno: https://marvelapp.com/ (pristupano 16.9.2019)

Popis slika

Popis tablica