Izrada chatbota Microsoft Bot Frameworkom

Orešić, Vjekoslav

VELEUČILIŠTE SUVREMENIH INFORMACIJSKIH TEHNOLOGIJA

STRUČNI DIPLOMSKI STUDIJ INFORMACIJSKIH TEHNOLOGIJA

Vjekoslav Orešić

DIPLOMSKI RAD

IZRADA CHATBOTA MICROSOFT BOT FRAMEWORKOM

Zagreb, listopada 2024.

Veleučilište suvremenih informacijskih tehnologija
10000 Zagreb, Ulica Vjekoslava Klaića 7

Studij:	Stručni diplomski studij informacijskih tehnologija smjer programsko inženjerstvo i informacijski sustavi
Student:	**Vjekoslav Orešić**
Matični broj:	2014063

Zadatak diplomskog rada

Predmet:	Principi objektno orijentiranog dizajna
Naslov:	**Izrada chatbota Microsoft Bot Frameworkom**
Zadatak:	Pomoću Microsoft Bot Frameworka izraditi ogledni chatbot. Pomoću Microsoft Bot Frameworka izraditi ogledni chatbot. U okviru rada opisati povijest i funkcionalnosti chatbota te mogućnosti i korištenje Microsoft Bot Frameworka te moguće pristupe pri izradi chatbota.
Mentor:	mr. sc. Julijan Šribar, v. pred.
Zadatak uručen kandidatu:	2.10.2023.
Rok za predaju rada:	17.10.2024.
Rad predan:	_____

Povjerenstvo:

Jurica Đurić, v. pred.	član predsjednik	_____
mr. sc. Julijan Šribar, v. pred.	mentor	_____
Mariza Maini, pred.	član	_____

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS KODOVA

1. UVOD

Umjetna inteligencija (engl. *artificial intelligence*, AI) sve više se integrira u svakodnevne živote stvaranjem inteligentnog softvera. Prvi chatbotovi su bili predprogramirani za određeni skup pitanja i odgovora dok je u današnje chatbotove uključena umjetna inteligencija. Današnji chatbot je tipičan primjer sustava s umjetnom inteligencijom i jedan je od najelementarnijih i najraširenijih primjera inteligentne interakcije čovjeka i računala. Chatbot je računalni programi koji reagira poput pametnog entiteta kada se s njim komunicira putem teksta ili glasa te razumije jedan ili više ljudskih jezika pomoću obrade prirodnog jezika (engl. *natural language processing*, NLP). Chatbotovi su još poznati kao interaktivni agenti ili virtualni asistenti.

Chatbotovi mogu oponašati ljudski razgovor i zabavljati korisnike, ali nisu stvoreni samo za to. Koriste se u aplikacijama za obrazovanje, zdravstvo, politiku, poslovanje, e-trgovinu, službu za korisnike itd. Chatbotovi su postali popularni jer pružaju mnoge prednosti i za obične korisnike i programere.

Dolaskom ere pametnih mobilnih uređaja, chatbotovi su postali poznati zahvaljujući virtualnim asistentima kao što su Appleov Siri, Amazonova Alexa ili Googleov Assistant. Gotovo svaki čovjek ima chatbot u svojem džepu. Predstavljanjem popularnog OpenAI ChatGPT još više se popularizirало korištenje chatbota u svakodnevnom životu.

Većina implementacija chatbotova neovisna je o platformi i trenutno je dostupna bez potrebnih instalacija. Kontakt s chatbotom širi se kroz društvenu mrežu korisnika, a da se ne napuštaju aplikacije za razmjenu poruka u kojoj chatbot djeluje. Chatbotovi su integrirani u grupne razgovore u aplikacijama za razmjenu poruka ili se dijele kao i bilo koji drugi kontakt u istim aplikacijama, te se može paralelno odvijati više razgovora s chatbotom. Znanje korištenja jednog chatbota lako se prenosi na korištenje drugih chatbotova.

Pouzdanost komunikacije, brze i jednostavne razvojne iteracije, mala fragmentacija verzija te mali napori u dizajnu sučelja su neke su od prednosti i za programere. Jednostavni alati za implementaciju chatbotova popularizirali su ih te omogućuju implementaciju chatbota bez znanja programiranja.

U prvom poglavlju rada pojasnit će se što je chatbot i bit će opisan njegov povijesni razvoj. Ostatak diplomskog rada će predstaviti neke od tehnologija za razvoj i implementaciju chatbotova te će se na praktičnom radu prikazati kako se izrađuju chatbotovi.

Praktičnim radom pokazati će se primjer jednostavne implementacije chatbota bez znanja programiranja. Pokazati će se kako izraditi i testirati chatbot na jednostavan način koristeći Bot Framework Composer, alat intuitivna sučelja i unaprijed konfiguriranih opcija. Primjerom chatbota koji se temelji na umjetnoj inteligenciji, koristeći mogućnosti ChatGPT modela, pružit će se uvid u potencijal integracije naprednih jezičnih modela u chatbotove. Primjerom tradicionalnog chatbota temeljenog na unaprijed definiranim pravilima, gdje su odgovori unaprijed određen na temelju specifičnih pravila ili obrazaca, pokazati će se kako tradicionalni chatbotovi slijede postavljene smjernice za interakciju s korisnicima. Ovim praktičnim radom istaknuti će se prednosti chatbota s umjetnom inteligencijom te nedostatci tradicionalnih chatbotova.

Cijeli praktični rad se može pronaći na javnim GitHub repozitorijima, DiplomskiChatBot (chatbot s integriranim ChatGPT modelom izrađen uz pomoć Microsoft Bot Framework SDK -om) i DiplomskiChatBot_Composer (chatbot s integriranim ChatGPT modelom izrađen uz pomoć grafičkog alata Microsoft Bot Framework Composer), na poveznicama https://github.com/Vjeko-Oresic/DiplomskiChatBot, odnosno https://github.com/Vjeko-Oresic/DiplomskiChatBot_Composer.

2. CHATBOT

Chatbot je računalni program ili web sučelje koje ima za cilj oponašati ljudski razgovor putem teksta ili glasovnih interakcija. Chatbotovi se obično koriste za interakciju s korisnicima u prirodnom jeziku, pružajući informacije, odgovarajući na pitanja, pružajući pomoć ili obavljajući određene zadatke na temelju unaprijed definiranih pravila, uzoraka ili algoritama strojnog učenja. Chatbotovi se mogu integrirati u različite platforme, poput web stranica, aplikacija za razmjenu poruka, društvenih mreža i mobilnih aplikacija. Chatbotovi mogu voditi razgovore, razumjeti korisničke unose, analizirati ih i generirati odgovarajuće odgovore. Razine složenosti i mogućnosti chatbotova mogu varirati ovisno o njegovom programiranju, dizajnu i osnovnoj tehnologiji.

Moderni chatbotovi obično su online i koriste sustave umjetne inteligencije koji su sposobni održavati razgovor s korisnikom na prirodnom jeziku i simulirati način na koji bi se čovjek ponašao kao partner u razgovoru. Moderni chatbotovi često koriste aspekte dubokog učenja i obrade prirodnih jezika, ali jednostavniji chatbotovi prisutni su desetljećima.

Tijekom 2023. godine chatbotovi su privukli veliku pozornost objavom OpenAI ChatGPT-a (GPT-3 i GPT-4) [8] i Google Geminija, nekada poznatog kao Bard.

2.1. Povijest chatbotova

1950. godine je Alan Turing objavio članak [2] u kojem je postavio pitanje može li stroj razmišljati i predložio test „igra imitacije“ kao kriterij inteligencije računalnog programa. Tim testom se provjerava može li osoba ustanoviti je li na drugoj stani razgovora stroj ili druga osoba. Ako osoba nije to u stanju ustanoviti, onda se smatra da je stroj prošao test. Danas je taj test poznat pod nazivom Turingov test.

Turingov test je potakao Josepha Weizenbauma da 1966. godine objavi program ELIZA, prvi chatbot za kojeg se smatralo kako može prevariti korisnika i uvjeriti ga da razgovara s pravim čovjekom. ELIZA je funkcionirala na metodi prepoznavanja uzoraka i zamjeni. ELIZA je simulirala razgovor, prepoznajući ključne riječi ili fraze, te je generirala odgovore prema unaprijed programiranim pravilima. Nakon ELIZA-e se 1972. godine pojavio drugi značajan chatbot, PERRY. Chatbot PERRY je razvio Kenneth Colby sa Stanforda te je tijekom razgovora simulirao ponašanje osobe sa shizofrenijom.

Tijekom 1980-ih i 1990-ih chatbotovi su postali strukturiraniji i utemeljeni na pravilima, ali je tek 1995. godine Richard Wallace napravio chatbot ALICE (engl. *Artificial Linguistic Internet*

Computer Entity), primjer chatbota s mogućnostima obrade prirodnih jezika. ALICE je koristila AILM (engl. *Artificial Intelligence Markup Language*) jezik za kreiranje XML dokumenta koji je sadržavao set pravila ili predložaka s definicijama kako chatbot treba odgovoriti na određeni upit. AILM jezik se i danas koristi kao rješenje za chatbotove.

Dolazak ere pametnih mobilnih uređaja i virtualnih asistenata, kao što su Apple Siri, Google Assistant ili Amazon Alexa, doveo je do značajnog pomaka u razvoju chatbotova i do porasta njihove popularnosti. Chatbotovi su se sve više integrirali u platforme društvenih mreža, a tvrtke su ih počele koristiti za automatizaciju korisničke podrške i interakciju s korisnicima.

Napredak u umjetnoj inteligenciji, strojnom učenju i obradi prirodnog jezika (engl. *Natural language processing*, NLP) potaknuo je razvoj sofisticiranijih i kontekstualno svjesnih chatbotova. Chatbotovi pokretani umjetnom inteligencijom, kao što su OpenAI-jev GPT-4.5 i Google Gemini, značajno su poboljšali sposobnosti razgovora. Chatbotovi su postali sveprisutni u raznim industrijama, pomažući u korisničkoj podršci, zdravstvu, financijama, e-trgovini itd. Nastavljaju se razvijati, uključujući duboko učenje (engl. *deep learning*), učenje s pojačanjem (engl. *reinforcement learning*) i druge tehnike umjetne inteligencije za pružanje interakcija koje su sve više slične ljudskim.

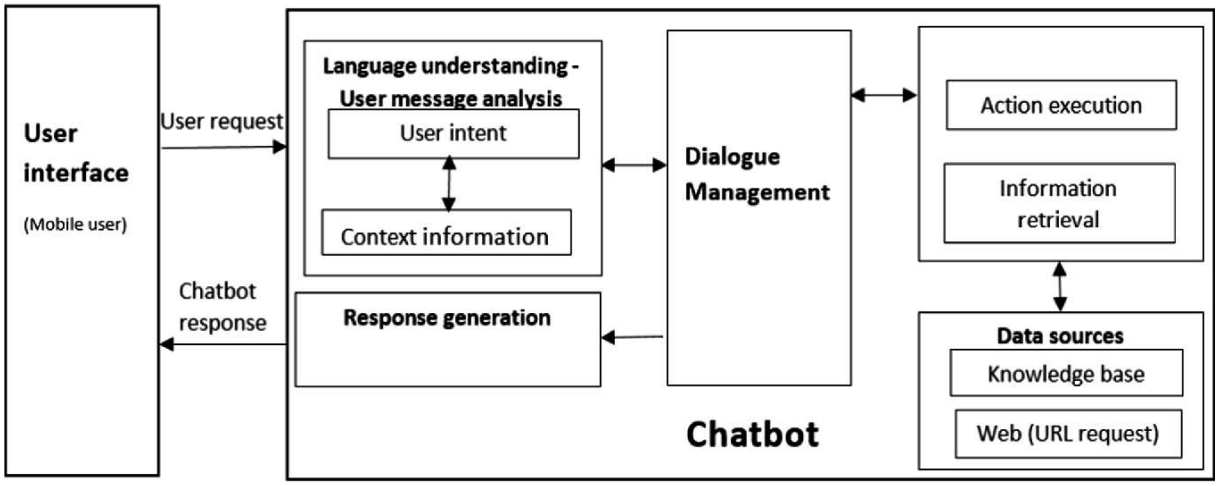
2.2. Vrste chatbotova

Postoje različite vrste chatbotova [3]:

- Chatbotovi temeljeni na pravilima (engl. *Rule-based chatbots*) slijede unaprijed definirana pravila i stabla odluka kako bi odgovorili na upite korisnika. Relativno su jednostavni i pružaju specifične odgovore na temelju ključnih riječi ili uzoraka za koje su programirani da ih prepoznaju.
- Chatbotovi temeljeni na strojnom učenju (engl. M*achine learning-based chatbots*) koriste algoritme strojnog učenja kako bi poboljšali svoje odgovore tijekom vremena. Uče iz podataka i interakcija korisnika kako bi unaprijedili svoje razumijevanje i generirali točnije i kontekstualno relevantne odgovore.
- Chatbotovi za obradu prirodnog jezika (engl. *Chatbots with natural language processing*) koriste napredne tehnike obrade prirodnog jezika kako bi razumjeli i interpretirali jezik korisnika. Mogu shvatiti kontekst, sentiment i namjeru kako bi pružili odgovore koji su slični ljudskim i kontekstualno prikladni.

2.3. Arhitektura chatbota

Neovisno o vrsti, svaki chatbot prati osnovnu arhitekturu (Slika 1) koja je sastavljena od temeljnih elemenata potrebnih za potpuni i ispravan rad chatbota.



Slika 1. Osnovna arhitektura chatbota (Adamopoulou, Eleni, Moussiades, Lefteris, 2020)

Svaki proces započinje korisnikovim zahtjevom. Nakon što je zaprimljen korisnikov zahtjev, komponenta za razumijevanje jezika obrađuje zahtjev kako bi zaključila korisnikovu namjeru i povezane informacije.

Nakon što chatbot postigne najbolju moguću interpretaciju, mora odlučiti kako nastaviti. Može izravno djelovati na nove informacije, sjetiti se svega što je razumio i čekati da vidi što će se sljedeće dogoditi, zahtijevati više informacija o kontekstu ili tražiti pojašnjenje.

Kada je zahtjev shvaćen, chatbot izvodi tražene radnje ili dohvaća podatke iz svojih izvora podataka, koji mogu biti baza podataka, poznata kao baza znanja chatbota, ili vanjski resursi kojima se pristupa putem API poziva.

Nakon dohvaćanja podataka, na temelju informacija o namjeri i kontekstu koje vraća komponenta analize korisničke poruke, komponenta za generiranje odgovora koristi generiranje prirodnog jezika (NLG) za pripremu odgovora korisniku na prirodnom jeziku nalik ljudskom.

Komponenta za upravljanje dijalogom čuva i ažurira kontekst razgovora koji se trenutna odvija te identificira entitete koji su potrebni za ispunjavanje zahtjeva korisnika. Traži informacije koje nedostaju, obrađuje pojašnjenja korisnika i postavlja dodatna pitanja.

2.4. Ograničenja chatbota

Stvaranje i implementacija chatbota još uvijek je područje u razvoju, uvelike povezano s umjetnom inteligencijom i strojnim učenjem, tako da ponuđena rješenja, unatoč očitim prednostima, imaju neka važna ograničenja u smislu funkcionalnosti i slučajeva upotrebe. Najčešća ograničenja chatbotova su:

- Budući da je ulazno/izlazna baza podataka fiksna i ograničena, chatbotovi mogu zakazati dok se bave nespremljenim upitom.
- Učinkovitost chatbota uvelike ovisi o obradi jezika i ograničena je zbog nepravilnosti, poput naglasaka i pogrešaka.
- Chatbotovi ne mogu rješavati više pitanja istovremeno pa su mogućnosti razgovora ograničene.
- Chatbotovi zahtijevaju veliku količinu konverzacijskih podataka za treniranje. Generativni modeli koji se temelje na algoritmima dubokog učenja, za generiranje novih odgovora riječ po riječ, obično se treniraju na velikom skupu podataka prirodnog jezika.
- Chatbotovi imaju poteškoća u upravljanju nelinearnim razgovorima koji s korisnikom moraju ići po raznim temama naprijed-natrag.

3. IZRADA CHATBOTA

Izrada chatbota uključuje korištenje različitih tehnologija, ovisno o složenosti, funkcionalnosti i mogućnostima koje se žele ostvariti. Korištenje frameworka ili platforme za razvoj chatbota može pojednostaviti proces razvoja. Platforme za razvoj chatbotova često pružaju gotove komponente, integracije i alate za projektiranje, izgradnju i implementaciju chatbota. Neke popularne platforme uključuju Google Dialogflow, Microsoft Bot Framework, Amazon Lex, IBM Watson Assistant i Rasa.

U nastavku će biti opisane tehnologije korištene za izradu chatbota u okviru ovog diplomskog rada.

3.1. Microsoft Bot Framework

Microsoft Bot Framework je platforma koja programerima omogućuje izgradnju, povezivanje, implementaciju i upravljanje chatbotovima na više kanala i platformi. Bot Framework uključuje modularni i proširivi SDK za izradu chatbotova i AI usluga za jednostavniji razvoj i implementaciju rješenja. Pomoću ovog frameworka programeri mogu stvoriti chatbotove koji koriste govor, razumiju prirodni jezik, odgovaraju na pitanja i još mnogo toga.

3.1.1. Bot Framework SDK

Bot Framework SDK je moćan set alata koji pružaju okruženje za razvoj chatbotova i konverzacijskih aplikacija. U središtu SDK-a je niz komponenata koje značajno pojednostavljuju složeni proces razvoja chatbotova:

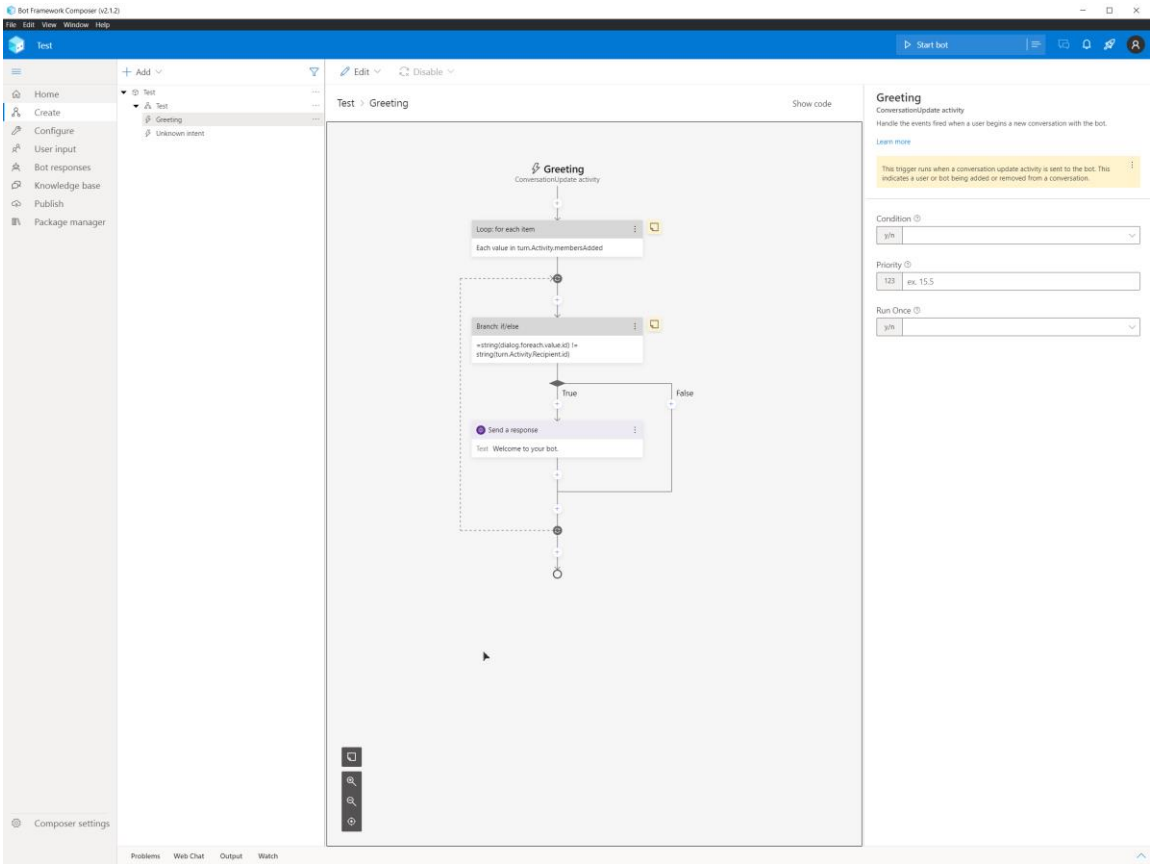
- Jedna od ključnih komponenata je Bot Builder, koji obuhvaća skup razvojnih alata za olakšani razvoj chatbotova korištenjem popularnih programskih jezika, poput C#, JavaScript i Python.
- Konektori Connector Service i Bot Connector su elementi koji djeluju kao most između chatbota i različitih kanala komunikacije, poput Microsoft Teamsa, Slacka i Facebook Messengera. Oni pružaju standardizirane API točke, omogućavajući chatbotovima komunikaciju s mnoštvom kanala, ujedinjujući napore u komunikaciji i pojednostavljujući razvoj.
- State Service omogućuje chatbotovima pohranu i upravljanje stanjem razgovora i korisnika. Ova značajka je ključna za očuvanje konteksta, personalizaciju odgovora i pružanje koherentnog korisničkog iskustva tijekom razgovora.

- Integracija s Microsoftovim Language Understanding (LUIS) omogućuje chatbotovima razumijevanje i interpretaciju namjera korisnika te izdvajanje entiteta iz njihovih unosa. Ova integracija poboljšava sposobnost chatbota da razumije prirodne jezike, omogućavajući smislenije i kontekstualno relevantne odgovore.
- Activity Model je definirani format poruke. Activity Model strukturira poruke i aktivnosti koje se razmjenjuju između chatbota i korisnika. Ova standardizacija osigurava dosljednost i jasnoću u komunikaciji, obuhvaćajući tekst, privitke i različita druga svojstva.
- Za učinkovito upravljanje dijalozima, SDK uključuje Dialog System. Ovaj sustav pojednostavljuje kreiranje konverzacijskih dijaloga, omogućujući programerima definiranje niza dijaloga, upita i akcija, čime se poboljšava tijek i struktura interakcija chatbota.
- SDK podržava Adaptive Cards, značajku koja olakšava dizajn vizualno privlačnih i interaktivnih poruka. Ove prilagodljive kartice povećavaju angažman i interakciju korisnika, nudeći vizualno bogato i dinamično korisničko sučelje kroz različite komunikacijske kanale.
- Uz sve navedeno, SDK sadrži i komponente Middleware koje omogućuju programerima proširenje sposobnosti chatbota. Mogu presretati poruke i ubaciti prilagođenu logiku, mehanizme autentifikacije ili obradu podataka u chatbot, prilagodivši tako ponašanje chatbota prema potrebama i zahtjevima korisnika.

Ovaj set alata i značajki čini Bot Framework SDK moćnim alatom za dizajniranje, razvoj, testiranje, implementaciju i upravljanje chatbotovima i konverzacijskim aplikacijama, pružajući fleksibilnost, skalabilnost i mogućnost prilagodbe različitim potrebama i platformama.

3.1.2. Bot Framework Composer

Bot Framework Composer je alat za vizualni razvoj, otvorenog koda i jednostavan za korištenje, koji olakšava stvaranje i upravljanje chatbotovima. Omogućuje dizajniranje tokova razgovora pomoću intuitivnog grafičkog sučelja (Slika 2). U središtu Composera je vizualni pristup izgradnji chatbota. Korisnici mogu dizajnirati kompleksne tijekove razgovora jednostavnim povlačenjem i ispuštanjem elemenata na grafičko sučelje. To čini definiranje strukture dijaloga i interakcija izuzetno jednostavnim i vizualnim, smanjujući potrebu za kompleksnim pisanjem koda.



Slika 2. Sučelje Bot Framework Composera

Korisnici mogu jednostavno integrirati generiranje i razumijevanje jezika (LUIS) kako bi izradili odgovore na prirodnom jeziku i razumjeli namjere korisnika. Alat podržava prilagodljive dijaloške okvire, okidače događaja, izvoz koda, testiranje, uklanjanje pogrešaka, kontrolu verzija i značajke suradnje, pojednostavljujući proces razvoja chatbotova.

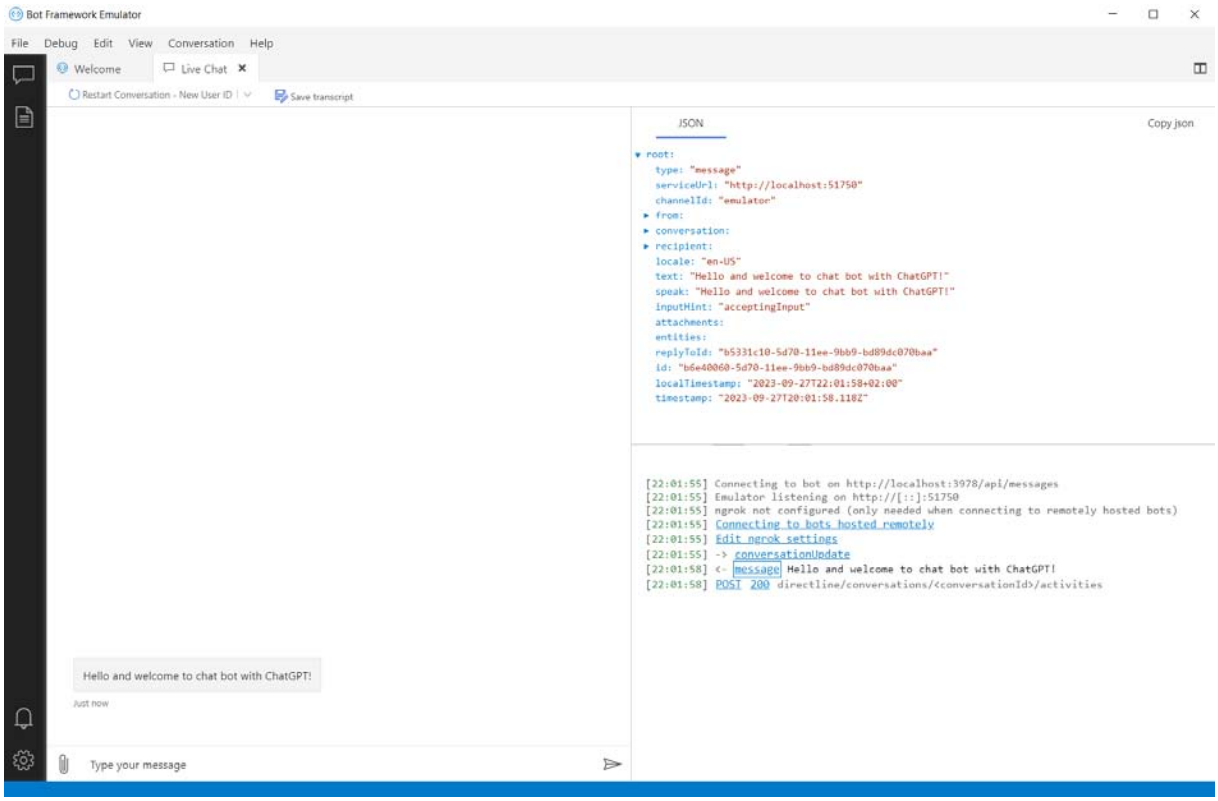
3.1.3. Bot Framework Emulator

Bot Framework Emulator (Slika 3) je važan alat za programere koji stvaraju i usavršavaju chatbotove. Lokalna mogućnost testiranja omogućuje programerima eksperimentiranje s različitim funkcionalnostima te prepoznavanje i ispravak problema na kontroliran i učinkovit način.

Jedna od ključnih značajki Bot Framework Emulatora je njegova sposobnost repliciranja sučelja za razgovor. Ovo emuliranje omogućuje programerima da komuniciraju s chatbotovima na isti način kao krajnji korisnici, stvarajući simulirano okruženje slično stvarnom razgovoru na platformi za razmjenu poruka. Programeri mogu slati poruke chatbotu i promatrati kako chatbot reagira, što omogućuje jasno razumijevanje korisničkog iskustva i omogućava da se ponašanje chatbota uskladi s očekivanjima.

Emulator također nudi uvid u poruke. Programeri mogu pregledati osnovni JSON paket poruka koje se razmjenjuje između chatbota i emulatora. Ova mogućnost inspekcije neprocjenjiva je u svrhu otklanjanja pogrešaka i razumijevanja tijeka podataka tijekom interakcije. Upravljanje stanjem chatbota još je jedna bitna značajka koju pruža emulator. Programeri mogu pregledati i mijenjati stanje razgovora, što je ključno za testiranje različitih scenarija i potvrdu da chatbot učinkovito obrađuje i zadržava kontekst razgovora.

Osim toga, emulator bilježi sve aktivnosti, uključujući poruke, događaje i pogreške. Taj zapis aktivnosti odličan je resurs za otklanjanje problema i dijagnosticiranje poteškoća.



Slika 3. Sučelje Bot Framework Emulatora

3.2. Open AI ChatGPT

ChatGPT je model za generiranje jezika koji je razvio OpenAI. ChatGPT je izrađen na GPT-3.5 i GPT-4 serijama modela te je jedan od najnaprednijih modela za generiranje jezika. ChatGPT se temelji na arhitekturi dubokog učenja i obučen je na ogromnoj količini tekstualnih podataka kako bi generirao tekst sličan ljudskom te razumio kontekst.

ChatGPT je dizajniran da sudjeluje u konverzacijskim interakcijama s korisnicima. Obrađuje ulazni tekst i generira koherentne i kontekstualno relevantne odgovore. Može simulirati

razgovore slične ljudskima i pružiti korisne, informativne ili kreativne odgovore na temelju dobivenog unosa. Općenito, ChatGPT ima širok spektar primjena, uključujući sastavljanje e-mailova, generiranje koda, odgovaranje na pitanja, podučavanje u različitim predmetima, pružanje pomoći u kreativnom pisanju, što ga čini svestranim alatom za različite namjene.

Važno je napomenuti da ChatGPT generira odgovore na temelju obrazaca koje je naučio iz podataka na kojima je obučen i ne posjeduje razumijevanje ili svijest. Također, posljednje ažuriranje znanja je imao u rujnu 2023. godine i ChatGPT ponekad može proizvesti netočne ili besmislene odgovore.

ChatGPT sadrži i chatbot koji je u siječnju 2023. godine postao najbrže rastuća aplikacija, stekavši više od 100 milijuna korisnika.

3.2.1. OpenAI API

OpenAI API omogućuje programerima integraciju ChatGPT-a u aplikacije, proizvode i usluge. API pruža programski pristup ChatGPT modelu, omogućujući generiranje odgovora te dinamičke i interaktivne razgovore s modelom.

OpenAI API omogućuje pristup raznolikom skupu modela s različitim mogućnostima, a programeri sami mogu prilagoditi modele za svoj specifičan slučaj. Neki od modela su:

- GPT-3.5 model, koji može razumjeti i generirati prirodni jezik ili kôd
- GPT-3.5 Turbo model, poboljšanje performansi GPT-3.5 modela
- GPT-4 model, poboljšanje GPT-3.5 modela
- GPT-4 Turbo model, poboljšanje performansi GPT-4 modela
- GPT-4o model, poboljšanje GPT-4 Turbo modela
- DALL-E model koji može generirati i uređivati slike uz upit prirodnog jezika
- Whisper model koji može pretvoriti govor u tekst
- TTS model koji može pretvoriti tekst u govor.

3.3. Azure Blob Storage

Azure Blob Storage je Microsoftova platforma za pohranu velikih količina raznolikih podataka u oblaku (engl. *cloud*). Ona omogućuje skalabilnu i trajnu pohranu objekata poput teksta i binarnih podataka. Podaci se organiziraju u objekte i mogu se klasificirati u različite slojeve ovisno o potrebama za pristupom. Osim toga, platforma pruža sigurnosne značajke, integraciju s drugim Azure uslugama, RESTful API za programski pristup i široki spektar primjena, uključujući pohranu medija, sigurnosne kopije, web hosting.

4. PRAKTIČNI RAD – CHATBOT

Izradi praktičnog rada pristupilo se na tri načina:

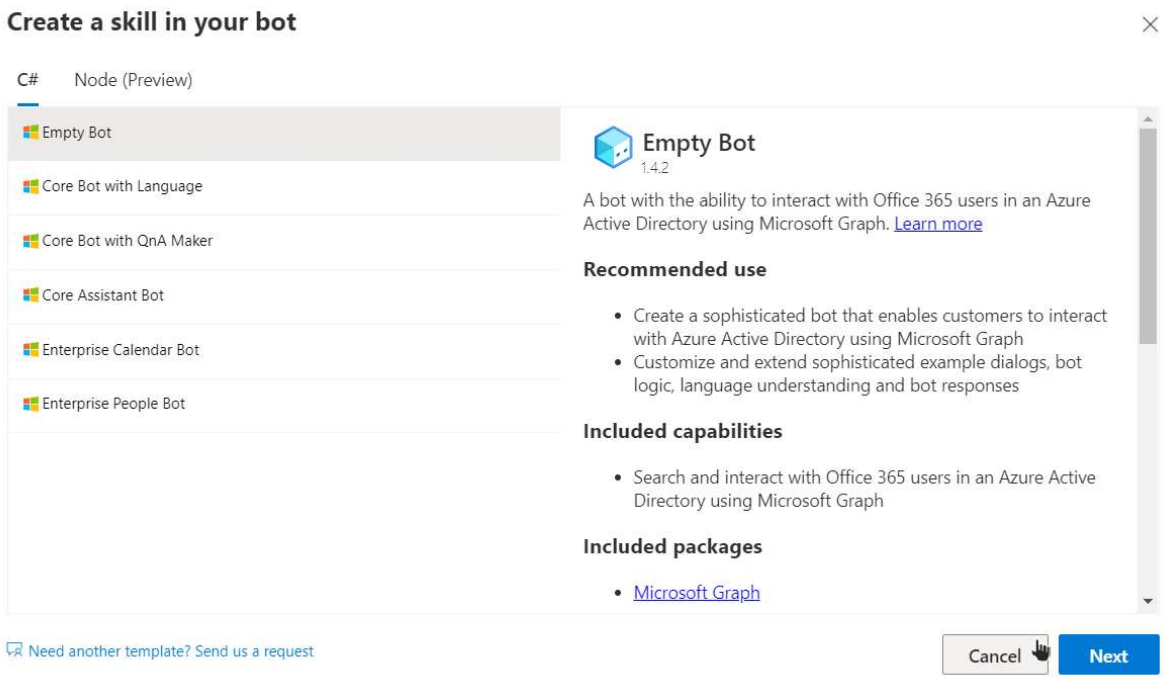
1. Izrada chatbota s integriranim ChatGPT modelom uz pomoć grafičkog alata Microsoft Bot Framework Composer
2. Izrada chatbota s integriranim ChatGPT modelom uz pomoć Microsoft Bot Framework SDK
3. Izrada chatbota temeljenog na pravilima uz pomoć Microsoft Bot Framework SDK

4.1. Izrada chatbota s integriranim ChatGPT modelom uz pomoć grafičkog alata Microsoft Bot Framework Composer

Bot Framework Composer je alat za vizualni razvoj chatbota, jednostavan za korištenje te će se uz pomoć ovog alata pokazati kako jednostavno izraditi chatbot bez znanja programiranja, ali uz minimalno znanje korištenja JSON formata i HTTP zahtjeva.

4.1.1. Kreiranje novog chatbota

Izrada chatbota u Bot Framework Composeru započinje kreiranjem novog chatbota iz predloška (Slika 4).



Slika 4. Kreiranje novog chatbota iz predloška

Predlošci daju početnu točku novog chatbota. Za potrebe izrade praktičnog rada odabran je predložak *Empty Bot* u programskom jeziku C#. Ovaj predložak stvara chatbot s korijenskim dijalogom, početnim pozdravnim dijalogom (*Greeting*) i upravljanje nepoznatom namjerom (*Unknown intent*) (Slika 5).

Slika 5. Popis kreiranih dijaloga

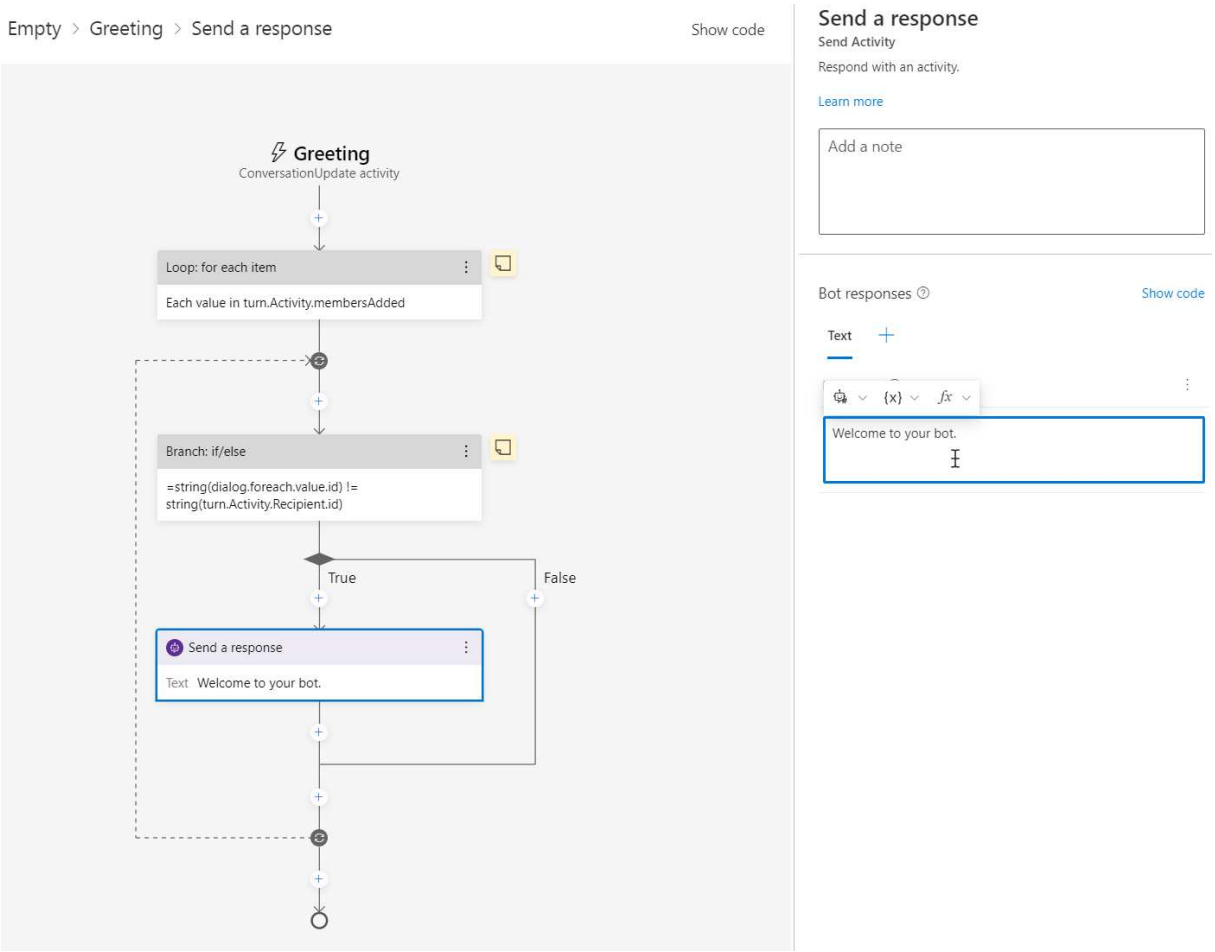
Chatbot je organiziran u dijaloge i okidače. Dijalozi predstavljaju dio funkcionalnosti chatbota i sadrže upute kako će chatbot reagirati na unos. Dijalozi mogu uključivati prilagođenu poslovnu logiku, pozive API-ima u oblaku, podatke o obuci sustava za obradu jezika i, što je još važnije, stvarni sadržaj koji se koristi u razgovorima s krajnjim korisnicima. Jednostavni chatbotovi imat će samo nekoliko dijaloga, dok sofisticirani mogu imati desetke ili stotine zasebnih dijaloga.

Svaki dijalog uključuje jedan ili više rukovatelja događaja koji se nazivaju okidači. Svaki okidač sadrži jednu ili više radnji. Akcije su upute što će chatbot izvršiti kada dijalog primi bilo koji događaj za koji ima definiran okidač. Nakon što okidač obradi određeni događaj, ne poduzimaju se daljnje radnje na tom događaju. Neki okidači imaju naveden uvjet koji mora biti ispunjen prije nego što počnu rukovati događajem, a ako taj uvjet nije ispunjen, događaj se prosljeđuje sljedećem rukovatelju događajima. Ako se događaj ne obrađuje u podređenom dijalogu, prosljeđuje se nadređenom dijalogu na obradu i to se nastavlja sve dok se ne obradi ili dok ne dosegne glavni dijalog chatbota. Ako se ne pronađe rukovatelj događajima, zanemarit će se događaj i neće se poduzeti nikakve radnje.

4.1.2. Uređivanje pozdravnog dijaloga

Pozdravni dijalog (Slika 6) je prvi dijalog koji se prikazuje korisniku. U pozdravnom dijalogu se provjerava identifikator korisnika koji je pokrenuo chatbot. Ako je identifikator korisnika različit od identifikatora chatbota, korisniku se ispisuje pozdravna poruka. Ovo je bitno provjeriti, jer u svakom razgovoru postoje dva člana: chatbot i korisnik, koje je neophodno razlikovati identifikatorom.

U praktičnom radu je pozdravni dijalog ostavljen gotovo nepromijenjen te je generiran na osnovu predloška. Samo je promijenjena pozdravna poruka te je dodana provjera vrijednosti parametra `user.isNew`, čije značenje će biti objašnjeno u kasnijem potpoglavlju.



Slika 6. Primjer pozdravnog dijaloga

4.1.3. Konfiguracija ChatGPT modela

Kako bi se chatbotu omogućio pristup ChatGPT modelu, potrebno je kreirati OpenAI API ključ te ga dodati u konfiguraciju chatbota (Kôd 1). Kako bi konfiguracija bila potpuna, potrebno je dodati link na koji se šalju upiti u API te ChatGPT model koji će se upotrebljavati u razgovoru. U praktičnom radu korišten je model GPT-3.5 Turbo.

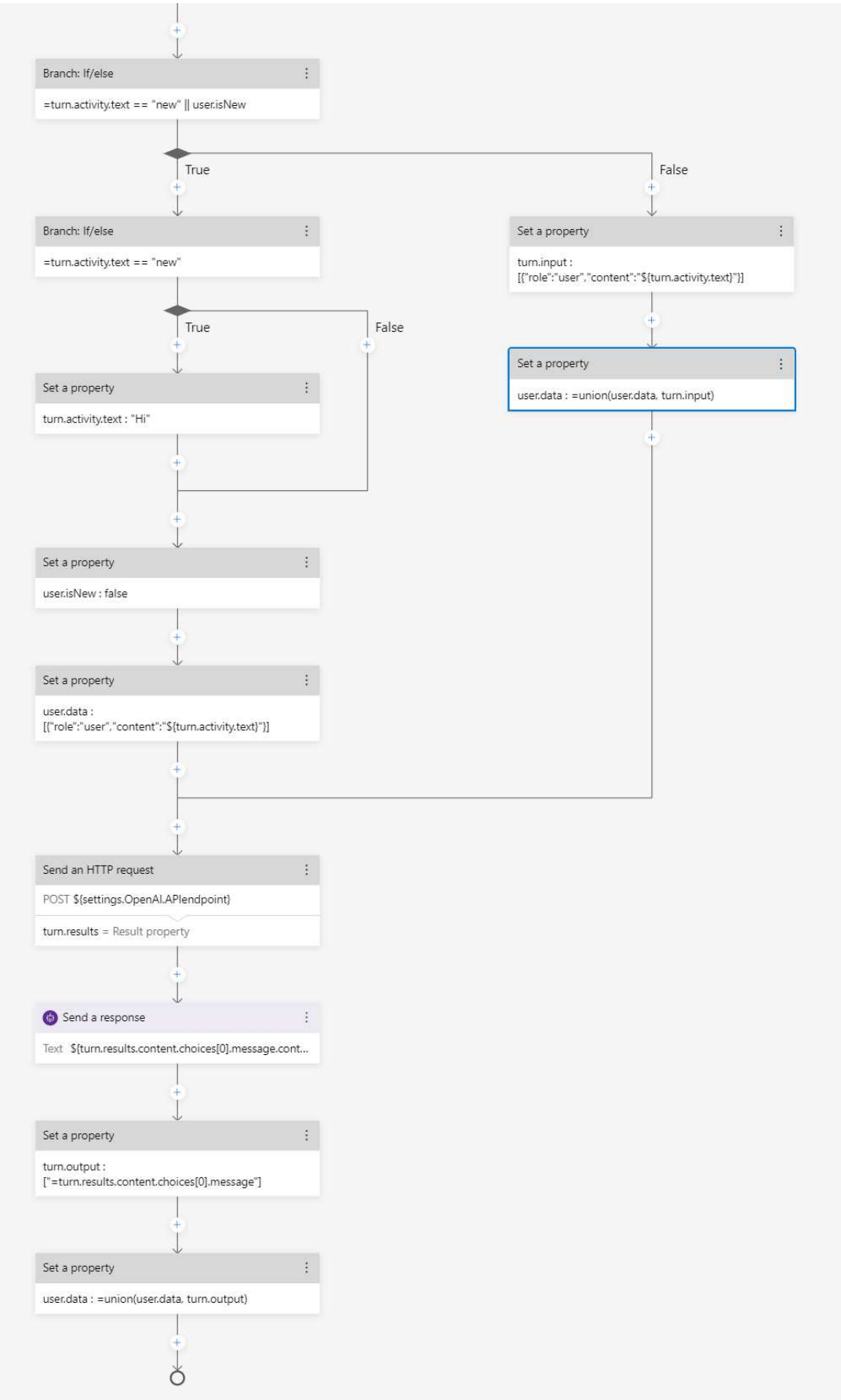
```
"OpenAI": {
  "APIkey": "…",
  "APIendpoint": "https://api.openai.com/v1/chat/completions",
  "APImodel": "gpt-3.5-turbo"
}
```

Kôd 1. Primjer podataka za konfiguraciju OpenAI API

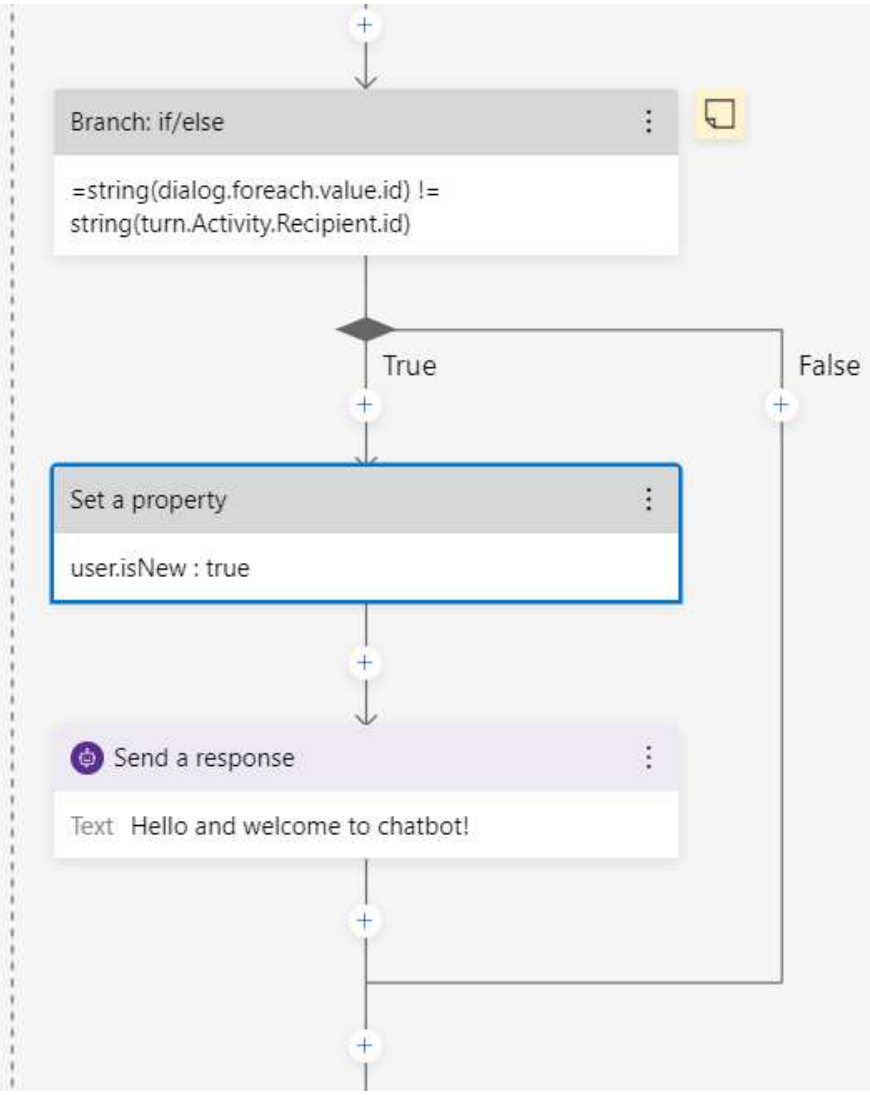
4.1.4. Izrada okidača s implementiranim ChatGPT modelom

Nakon što je dodana konfiguracija za pristup ChatGPT modelu, moguće je započeti izradu rukovatelja događaja tj. okidača (Slika 7). Prije toga potrebno je znati kako je ChatGPT model „zaboravan“ te ga je potrebno „podsjećati“ na cijelu prošlost trenutnog razgovora pružanjem pristupa cijelom razgovoru. Osim toga, potrebno je zapamtiti kako nakon svakog korisnikovog unosa okidač počinje od prve akcije.

Imajući sve to na umu, potrebno je uvesti parametar `user.isNew` koji će provjeravati je li unos potekao od novog korisnika ili od trenutnog korisnika u razgovoru. Početna vrijednost parametra `user.isNew` se postavlja u pozdravnom dijalogu na `true` (Slika 8).



Slika 7. Primjer okidača za implementaciju ChatGPT modela



Slika 8. Akcija postavljanja vrijednosti parametru `user.isNew`

Također, korisniku je potrebno omogućiti započinjanje novog razgovora s ChatGPT-em zato jer ChatGPT generira odgovore na temelju obrazaca koje je naučio iz podataka na kojima je obučen i ne pamti nove podatke. Za ovu svrhu će se koristiti unos ključne riječi `new` kojom se definira započinje li se novi razgovor ili se nastavlja trenutni razgovor s ChatGPT-ijem.

Sljedeća bitna akcija je priprema zahtjeva za API u JSON formatu (Kôd 2). Parametar `model` se popunjava iz konfiguracije te je tako lakše promijeniti korišteni ChatGPT model. Bitno je pripaziti da `role` bude `user` kako bi ChatGPT model znao da je to korisnikov upit. Korisnikov unos je sadržan u parametru `turn.activity.text` te se taj parametar u zahtjev unosi u `content` dio zahtjeva za API koji sadrži korisnikov unos (poruku) odnosno odgovor od ChatGPT-a.

```
{
  "model": "${settings.OpenAI.APImodel}",
  "messages":
    [
      {
        "role": "user",
        "content": "${turn.activity.text}"
      }
    ]
}
```

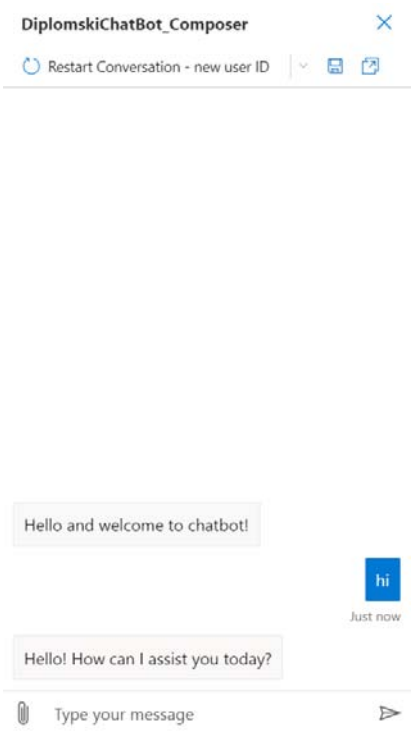
Kôd 2. Zahtjev za API u JSON formatu

Zahtjev se u API šalje preko HTTP zahtjeva, kojem se parametri popunjavaju iz konfiguracije a njegov rezultat se zapisuje u parametar `turn.results`, koji se zatim ispiše korisniku u aplikaciji. Kako bi se ChatGPT model moglo „podsjetiti“ da trenutni razgovor i dalje traje, potrebno je zapamtiti dobiveni rezultat u parametru `user.data`.

Pri sljedećem korisnikovom unosu, parametar `user.isNew` bit će `false` pa je u zahtjev za API, uz novi korisnikov unos, u `content` potrebno dodati i vrijednost parametra `user.data`. Zahtjeve za API je potrebno kreirati na ovaj način sve dok traje aktualni razgovor.

4.1.5. Testiranje chatbota

U Bot Framework Composeru je moguće jednostavno testirati kreirani chatbot s ugrađenim web chatom ili korištenjem Bot Framework Emulatora. Treba samo pokrenuti chatbot i odabrati Open Web Chat (Slika 9). Chatbot će pokrenuti pozdravni dijalog i ispisati pozdravnu poruku. Nakon prvog korisnikovog unosa aktivirat će se okidač koji će prosljeđivati sve korisnikove unose u ChatGPT model putem OpenAI API-a.



Slika 9. Integrirani web chat za testiranje chatbota

Najbolji način za testiranje chatbota s integriranim ChatGPT modelom je pitati ga za ime korisnika. U prvom odgovoru neće znati ime, ali ako mu korisnik unese svoje ime, kod sljedećeg upita će ga znati. Na ovaj način se može potvrditi radi li ispravno veza s API-jem te šalje li se ispravno povijest razgovara u ChatGPT model.

4.2. Izrada chatbota s integriranim ChatGPT modelom s Microsoft Bot Framework SDK -om

Za izradu chatbota s integriranim ChatGPT modelom s Microsoft Bot Framework SDK-om moguće je iskoristiti primjer chatbota kreiranog u Bot Framework Composeru te istu logiku primijeniti i ovom primjeru u praktičnog radu. Za razliku od prethodnog primjera chatbota s integriranim ChatGPT-om, u ovom će se koristiti Azure Blob Storage za pamćenja povijesti razgovora te Bot Framework Emulator za testiranje chatbota.

4.2.1. Kreiranje novog chatbota

Kao i u prethodnom primjeru izrade chatbota, i za Bot Framework SDK postoje predlošci koji se mogu preuzeti na stranici Visual Studio Marketplacea. Za potrebe izrade praktičnog rada korišten je predložak „Empty Bot“.

Tablica 1. Predlošci za Microsoft Bot Framework SDK

Predložak	Opis
Echo Bot	Predložak za početnike ako žele nešto više od „Hello World!“. Ovaj predložak obrađuje osnove slanja poruka chatbotu, a chatbot obrađuje poruke ponavljajući ih nazad korisniku.
Core Bot	Najnapredniji predložak koji pruža 6 osnovnih značajki koje će vjerojatno imati svaki chatbot. Ovaj predložak pokriva temeljne značajke AI chatbota koji koristi LUIS.
Core Bot with Tests	Predložak sadrži sve značajke Core Bot predloška uz kompletan skup testova.
Empty Bot	Predložak za one koji su upoznati s Bot Framework, a žele jednostavan osnovni kostur projekta.

4.2.2. Implementacija chatbota

U klasi GPTBot se nalazi primjer izrađenog chatbota s integriranim ChatGPT modelom i prvo što je u njoj bilo potrebno kreirati je metoda `OnMembersAddedAsync` (Kôd 3) koja se prva poziva nakon pokretanja chatbota te ispisuje pozdravnu poruku korisniku ako je identifikator korisnika različit od identifikatora chatbota.

```
var welcomeText = "Hello and welcome to chatbot with ChatGPT!";
foreach (var member in membersAdded)
{
  if (member.Id != turnContext.Activity.Recipient.Id)
  {
    await turnContext.SendActivityAsync(MessageFactory.Text
                (welcomeText), cancellationToken);
  }
}
```

Kôd 3. Sadržaj metode `OnMembersAddedAsync`

U SDK-u se `turnContext.SendActivityAsync` koristi za prikazivanje poruka chatbota, a `turnContext.Activity.Text` za čitanje korisnikovog unosa.

`OnMessageActivityAsync metoda je okidač` te je u njoj implementirana ista logika kao i prethodnom primjeru chatbota s integriranim ChatGPT modelom. Ova metoda na isti način provjerava je li korisnik unio ključnu riječ `new` za započinjanje novog razgovora (Kôd 4). Na sličan način se brine da se ChatGPT modelu šalje povijest trenutnog razgovora koja se ovaj put čuva u tablici u Azure Blob Storage te se tako korisniku omogućava korištenje istog razgovara kadgod se vrati u razgovor s chatbotom, uz uvjet da se nije promijenio identifikator korisnika ili identifikator razgovora s chatbotom.

```
if (text.ToLower().Equals("new"))
{
  request = new CompletionRequest
  {
    Model = _configuration["OpenAI:APImodel"],
    Messages = new List<Messages>()
    {
      new Messages()
      {
         Role = "user",
         Content = "hi"
      }
    }
  };
}
```

Kôd 4. Primjer pripreme zahtjeva ako je korisnik unio ključnu riječ `new`

Slanje HTTP zahtjeva je napravljeno u metodi `GetGPTResponse` koju poziva metoda `OnMessageActivityAsync`. Za slanje i čitanje rezultata koristi se `HttpClient`. Svi potrebni podatci za slanje zahtjeva preko API-a čitaju se iz konfiguracije koja je napravljena na isti način kao prethodnom primjeru (Kôd 1).

Za rad s Azure Blob Storageom se koristi klasa `StorageHelper` koja sadrži metode `GetEntityAsync` i `UpsertEntityAsync` (**Error! Reference source not found.**). One se pozivaju u metodi `OnMessageActivityAsync`.

```
var gptContext = await _storageHelper.GetEntityAsync<GPTResponse>
                      (_configuration["AzureStorage:GPTContext"],
                       turnContext.Activity.From.Id,
                       turnContext.Activity.Conversation.Id);

await _storageHelper.UpsertEntityAsync
  (_configuration["AzureStorage:GPTContext"], gptResponseObject);
```

Kôd 5 Pozivi metoda `GetEntityAsync` i `UpsertEntityAsync`

Metodom `GetEntityAsync` se provjerava ima li trenutni korisnik povijest razgovora s chatbotom. Ako ima povijest razgovora, onda se ona dohvati iz tablice `GPTContext` i dodaje u zahtjev u JSON formatu, koji izgleda slično kao u prethodnom primjeru (Kôd 2). Metoda `UpsertEntityAsync` sprema dobiveni odgovor s API-a za pamćenje povijesti razgovora. Metoda `GetTableServiceClient` se koristi za spajanje na Azure Blob Storage i pristup tablici `GPTContext` (Slika 10) u kojoj se nalazi povijest svih razgovora s chatbotom. Tablica `GPTContext` je definirana u klasi `GPTResponse` koja nasljeđuje klasu `CommonEntity` i definira osnovni dio svih tablica u Azure Blob Storageu.

Showing all 6 items

PartitionKey	RowKey	Timestamp	GPTContext
01f6c217-c3e3-4ab3-94...	ead63090-5cd1-11ee-9...	2023-09-27T01:07:34.88...	[{"role":"user","content":"hi"},{"role":"assistant","content":"...
1f3b50d1-f054-4ed4-96...	ef322650-4749-11ee-89...	2023-08-30T15:29:37.57...	[{"role":"user","content":"hi"},{"role":"assistant","content":"...
353e96cc-2bc9-439c-81...	0e65ab00-474a-11ee-a...	2023-08-30T15:29:56.88...	[{"role":"user","content":"what my name"},{"role":"assistant...
5618debe-ada0-4813-b...	9636e290-5d45-11ee-b...	2023-09-27T14:53:41.10...	[{"role":"user","content":"hi"},{"role":"assistant","content":"...
6f92ac5f-7320-49f8-bfc...	67a64a10-4724-11ee-b...	2023-08-30T11:01:06.65...	[{"role":"user","content":"hi"},{"role":"assistant","content":"...
9e48d4c1-a9a9-4b78-a...	ec132050-477b-11ee-9...	2023-08-30T21:27:53.88...	[{"role":"user","content":"can you write in croatian?"},{"role...

Slika 10. Tablica `GPTContext` na Azure Blob Storage

Ako se slučajno ne dobije odgovor od ChatGPT modela, korisniku se ispisuje poruka isprike, a za sve druge pogreške se SDK sam brine uz pomoć klase `AdapterWithErrorHandler` u kojoj je moguće samostalno definirati što će se događati u slučaju pogrešaka.

JSON zahtjevi i odgovori od API-a se pretvaraju u objekt i obratno, a za te akcije se koristi metoda `JsonConvert` iz Newtonsoft JSON frameworka. U klasama `CompletionRequest` i `CompletionResponse` je definiran objekt za zahtjev odnosno odgovor od API-ja.

Konfiguracija Azure Blob Storagea (Kôd 6) i OpenAI API-a se nalaze u datoteci appsettings.Development.json.

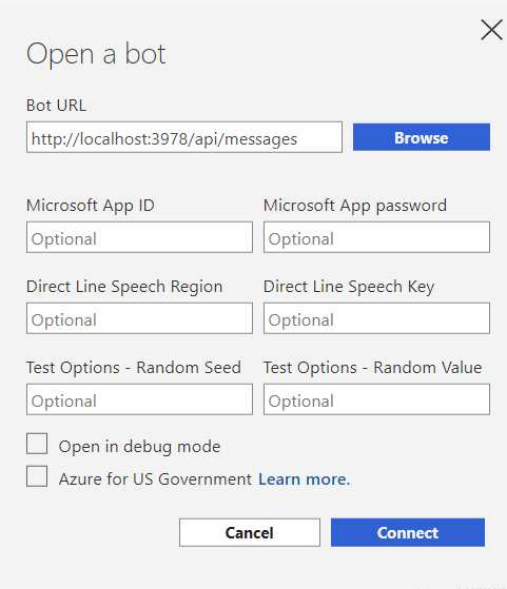
```
"AzureStorage": {
    "GPTContext": "GPTContext"
  },
  "ConnectionStrings": {
    "AzureStorage":
      "DefaultEndpointsProtocol=https;
      AccountName=…;AccountKey=…;EndpointSuffix=core.windows.net"
  }
```

Kôd 6. Primjer podataka za konfiguraciju Azure Blob Storage

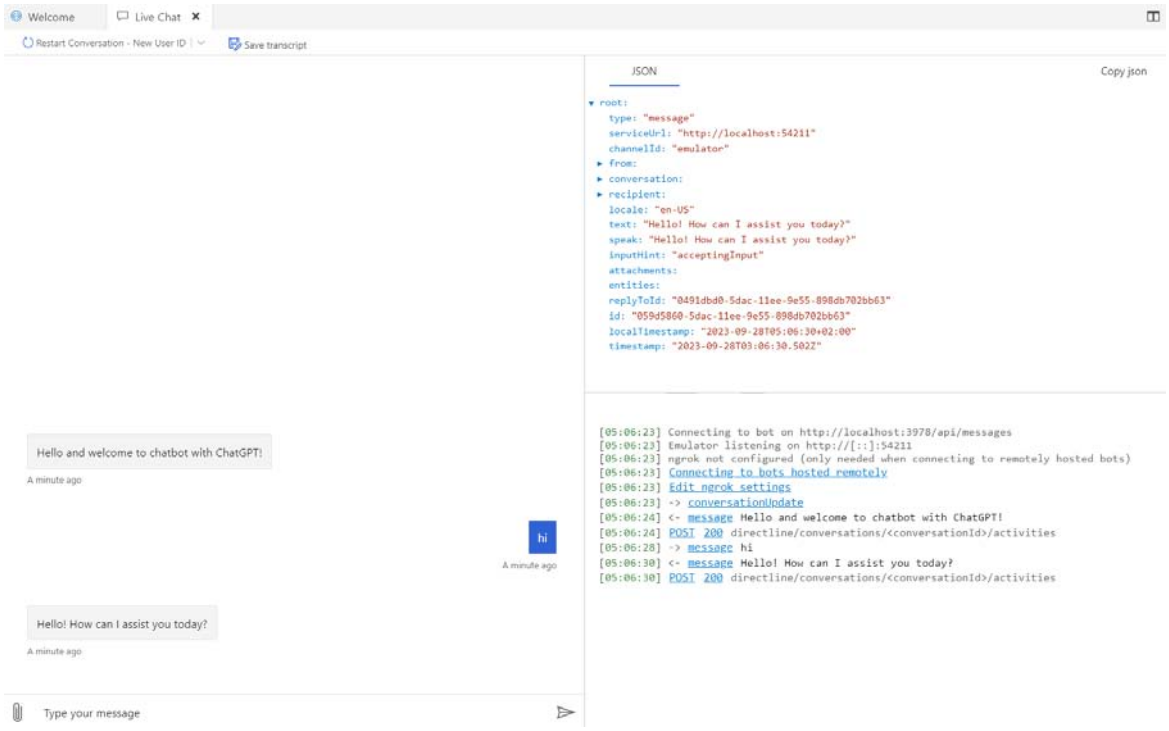
4.2.3. Testiranje chatbota

Za testiranje chatbota kreiranog s Bot Framework SDK-om koristi se Bot Framework Emulator. Nakon pokretanja chatbota u Visual Studiju, potrebno je u Emulatoru unijeti link do pokrenutog chatbota te po potrebi popuniti i ostale podatke (Slika 11). Nakon pokretanja, moguće je ponoviti isti test s imenom korisnika, opisan u odlomku testiranja chatbota kreiranog uz pomoć Bot Framework Composera (4.1.5).

U Bot Framework Emulatoru, osim slanja poruka chatbotu, moguće je odabirati pojedine poruke i provjeravati u desnom dijelu prozora (Slika 12) neobrađene informacije o poruci u JSON formatu. Informacije o poruci sadrže ključne meta podatke, uključujući identifikator kanala, vrstu aktivnosti, identifikator razgovora, tekstualnu poruku, link do krajnje točke. Mogu se provjeriti aktivnosti koje šalje korisnik i aktivnosti na koje chatbot odgovara.



Slika 11. Postavke za spajanje Bot Framework Emulatora na chatbot

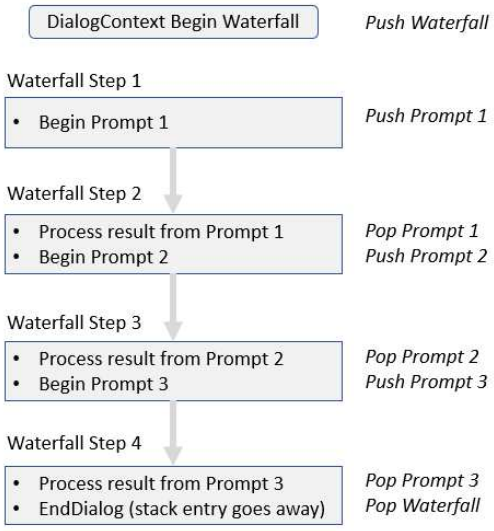


Slika 12. Primjer testiranja chatbota u Bot Framework Emulatoru

4.3. Izrada chatbota temeljenog na pravilima uz pomoć Microsoft Bot Framework SDK-a

Za primjer izrade chatbota temeljenog na pravilima napravljen je chatbot za kreiranje profila korisnika. Za izradu ovog chatbota korišten je vodopadni dijalog (engl. *waterfall dialog*) (Slika

13) koji definira niz koraka koji omogućuju chatbotu da vodi korisnika kroz unaprijed definirani linearni proces. U svakom koraku chatbot traži od korisnika unos, čeka odgovor, a zatim prosljeđuje odgovor sljedećem koraku.



Slika 13. Prikaz rada vodopadnog dijaloga (Microsoft, 2022)

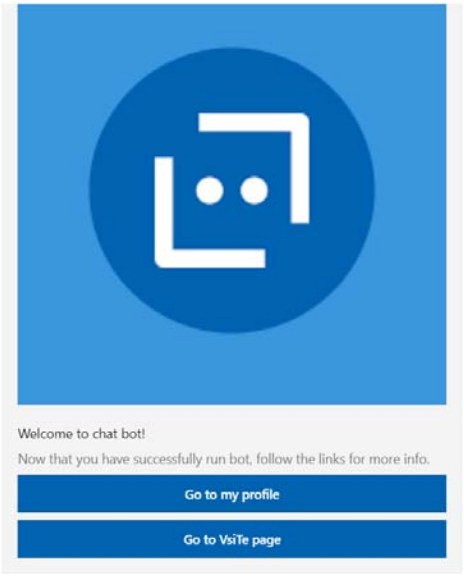
U ovom chatbotu koristit će se sve vrste upita na koje će korisnik odgovarati. Upiti traže unos odgovora te vraćaju vrijednost ili se ponovno pokreće s ponovnim upitom. Postoje različite vrste osnovnih upita od kojih se svaki koristi za prikupljanje odgovora različitih tipova podataka.

Tablica 2. Vrste upita u vodopadnom dijalogu

Upit	**Opis**	**Rezultat**
Attachment prompt	Traži jedan ili više privitaka poput dokumenta ili slike	Kolekcija objekata privitaka
Choice prompt	Traži izbor iz niza opcija	Objekt odabrane opcije
Confirm prompt	Traži potvrdu	Vrijednost tipa Boolean
Date-time prompt	Traži datum i vrijeme	Kolekcija objekata datum-vrijeme
Number prompt	Traži broj	Numerička vrijednost
Text prompt	Traži unos teksta	Tekst

4.3.1. Implementacija chatbota

Implementacija chatbota temeljenog na pravilima započinje izradom klase `MainBot` koja sadrži metodu `OnMembersAddedAsync` za ispisivanje pozdravne poruke, ali ovaj put u obliku adaptivne kartice (Slika 14). Adaptivna kartica je komponenta korisničkog sučelja neovisna o platformi, koja JSON datoteku `welcomeCard.json` transformira u izvorno korisničko sučelje platforme te se automatski prilagođava okolini.



Slika 14. Primjer adaptivne kartice

Nakon toga se pokreće vodopadni dijalog, definiran u klasi `MainDialog`, koji sadrži sve korake potrebne za kreiranje korisnikovog profila. U konstruktoru klase potrebno je definirati sve potrebne vrste upita te korake vodopadnog dijaloga (Kôd 7**Error! Reference source not found.**). Svi koraci koriste metodu `stepContext.PromptAsync` za postavljanje pitanja i čekanje odgovora te u slučaju neispravnog odgovora ponavljaju pitanje.

```
AddDialog(new TextPrompt(nameof(TextPrompt)));
AddDialog(new              NumberPrompt<int>(AgePromptDlgId,
ValidateAgeAsync));
AddDialog(new DateResolverDialog());
AddDialog(new ChoicePrompt(nameof(ChoicePrompt)));
AddDialog(new AttachmentPrompt(nameof(AttachmentPrompt)));
AddDialog(new ConfirmPrompt(nameof(ConfirmPrompt)));

var waterfallSteps = new WaterfallStep[]
{
    IntroStepAsync,
    NameStepAsync,
    AgeStepAsync,
    DateStepAsync,
    LanguageStepAsync,
    PhotoStepAsync,
    ConfirmStepAsync,
    FinalStepAsync,
};

AddDialog(new WaterfallDialog(nameof(WaterfallDialog),
          waterfallSteps));
```

Kôd 7. Definirani upiti i koraci vodopadnog dijaloga

Korak `NameStepAsync` postavlja pitanje koristeći upit `TextPrompt` i kao odgovor čeka unos imena i prezimena. Taj odgovor se pročita i spremi u koraku `AgeStepAsync` (Kôd 8**Error! Reference source not found.**) koji traži unos korisnikove dobi, koristeći upit `NumberPrompt`. Dobiveni odgovor se provjerava u metodi `ValidateAgeAsync`, koja provjerava je li uneseni broj veći od 18. Ako je broj manji od 18, korisniku će se ispisati poruka da je premlad za kreiranje profila.

```
stepContext.Values["name"] = (string)stepContext.Result;
var promptMessageText = $"How old are your?";
var promptMessage = MessageFactory.Text(promptMessageText);
var retryMessageText = $"Please enter a valid age.";
var retryMessage = MessageFactory.Text(retryMessageText);
return await stepContext.PromptAsync(AgePromptDlgId,
    new PromptOptions
    {
      Prompt = promptMessage,
      RetryPrompt = retryMessage
    }, cancellationToken);
```

Kôd 8. Primjer čitanja i spremanja korisnikova odgovora te postavljanje novog pitanja

U koraku `DateStepAsync` se poziva dijalog `DateResolverDialog` koji ima definirane svoje korake `InitialStepAsync` i `FinalStepAsync` za postavljanje pitanja korištenjem upita `DateTimePrompt` te ti koraci sadrže provjeru ispravnosti unesenog datuma. Ovaj dijalog ponavlja isto pitanje sve dok korisnik ne unese datum u ispravnom formatu. Nakon što korisnik unese ispravan odgovor, dijalog završava te se nastavlja `MainDialog` koji u svom sljedećem koraku `LanguageStepAsync` sprema odgovor iz prethodnog koraka `DateStepAsync`.

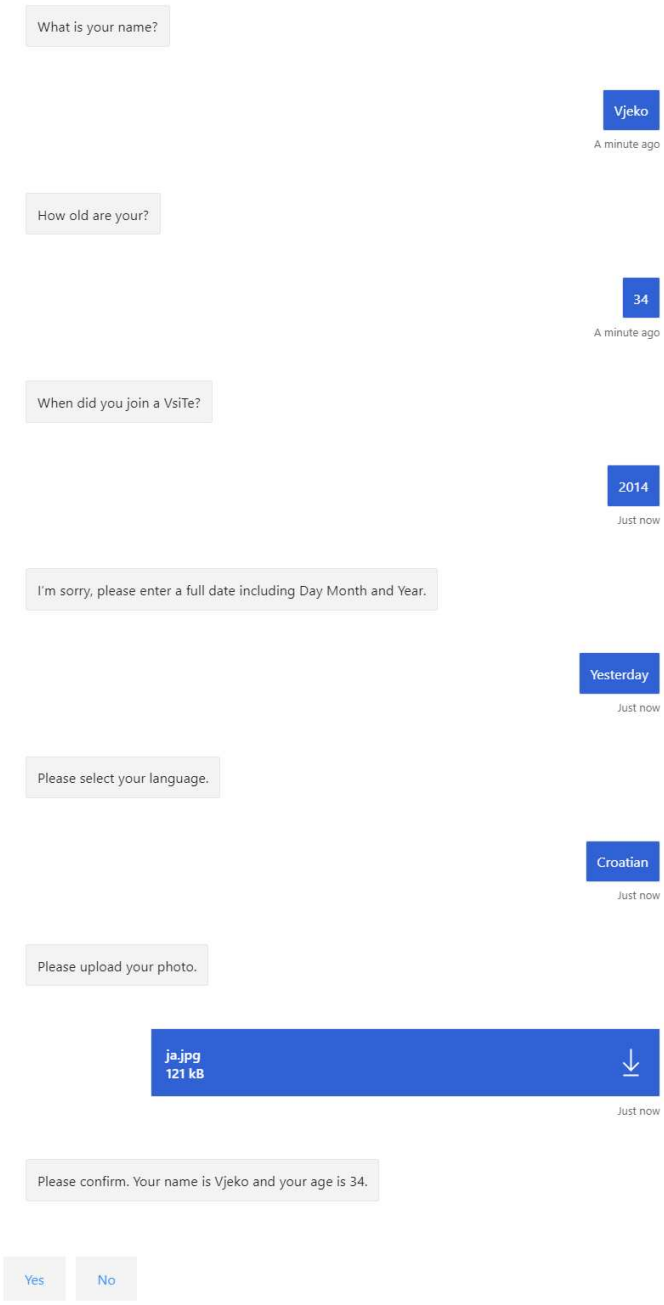
Korak `LanguageStepAsync` postavlja pitanje koji jezik koristi, ali korisniku ispisuje i ponuđene opcije jezika, jer ovaj korak koristi upit `ChoicePrompt`. Ako korisnik unese odgovor na bilo koji način koji nije odabir jedne od opcija, ovaj korak će ponavljati isto pitanje.

U sljedećem koraku `PhotoStepAsync`, nakon spremanja rezultata iz prethodnog koraka i postavljanja pitanja, čeka se odgovor u obliku priloga, jer ovaj korak koristi upit `AttachmentPrompt`. Korisnik može priložiti jednu ili više datoteka kao priloge. Rezultat ovog upita će se u sljedećem koraku `ConfirmStepAsync` spremiti na disk.

Korak `ConfirmStepAsync` koristi upit `ConfirmPrompt` i od korisnika očekuje odabir jedne od dvije ponuđene opcije: „Da“ ili „Ne“. Ako je odgovor „Da“, u koraku `FinalStepAsync`, svi će se odgovori spremiti u objekt `UserDetails`, koji se zatim može proslijediti sljedećem dijalogu ili nekoj drugoj povezanoj aplikaciji ili servisu te ovim korakom završava dijalog `MainDialog`. U slučaju odgovora „Ne“, zaboravljaju se svi zaprimljeni odgovori te se dijalog pokreće od prvog koraka `IntroStepAsync`.

4.3.2. Testiranje chatbota

Testiranje se odvija u Bot Framework Emulatoru na sličan način opisan u odlomku testiranja chatbota kreiranog uz pomoć Bot Framework SDK (4.2.3), s razlikom da ovaj put treba pratiti stroga pravila chatbota temeljenog na unaprijed definiranim pravilima i upitima (Slika 15).



Slika 15. Chatbot temeljen na pravilima

5. ZAKLJUČAK

U ovom diplomskom radu su istražene i navedene informacije o osnovnim principima chatbotova, na osnovu kojih se može preciznije razumjeti kako ih koristiti i stvarati.

Chatbotovi su svestrani alati koji poboljšavaju korisnička iskustva, pružajući trenutnu pomoć dvadeset četiri sata dnevno, sedam dana u tjednu. Oni učinkovito automatiziraju zadatke, smanjuju troškove i poboljšavaju produktivnost. Chatbotovi nude personalizirana iskustva, integriraju se s različitim platformama i prikupljaju vrijedne podatke za uvide. Uz prilagodljivost, višejezičnu podršku i aplikacije u različitim industrijama, chatbotovi igraju ključnu ulogu u modernom poslovanju.

Chatbotovi mogu doprijeti do široke publike u aplikacijama za razmjenu poruka i biti učinkovitiji od ljudi. U isto vrijeme, mogu se razviti u sposoban alat za prikupljanje informacija. Omogućuju značajne uštede u radu službi za korisnike. S daljnjim razvojem umjetne inteligencije i strojnog učenja, korisnici u budućnosti neće moći razaznati razgovaraju li s chatbotom ili agentom iz stvarnog života.

ChatGPT model je svestran alat za generiranje prirodnog jezika. Pomaže u stvaranju ljudskog teksta za razne svrhe. Međutim, korisnici bi trebali biti oprezni u pogledu točnosti i odgovorno koristiti sadržaj koji generira ChatGPT i slični modeli umjetne inteligencije.

Microsoft Bot Framework pruža moćan i učinkovit način za razvoj chatbota. Bot Framework Composer posebno usmjerava razvoj chatbotova, nudeći vizualno sučelje i pristup vođen dijaloškim okvirom. To programerima, ali i onima koji nemaju programersko iskustvo olakšava stvaranje složenih agenata za razgovor i upravljanje njima. Microsoft Bot Framework SDK je svestran i robustan alat za stvaranje sučelja za chatbot. Jednostavno se integrira s različitim platformama za razmjenu poruka, poboljšava razumijevanje jezika i nudi proširivost kroz pristup otvorenog koda.

Praktičnim radom se došlo do zaključka da svaka vrsta chatbota ima svoju primjenu. Chatbot temeljen na unaprijed definiranim pravilima, iako je često jednostavan i manje fleksibilan, odličan je za prikupljanje obaveznih podataka. Chatbotovi s integriranim ChatGPT modelom ili nekom drugom umjetnom inteligencijom razumiju i generiraju odgovore sličnih ljudskim. Mogu naučiti i poboljšati svoje odgovore tijekom vremena, pružajući interaktivnije i dinamičnije iskustvo razgovora, ali mogu korisnicima dati i krive odgovore jer umjetnoj inteligenciji još treba vremena da dođe do svog punog potencijala.

Trenutno najbolja vrsta su hibridni chatbotovi. Oni kombiniraju pristupe temeljene na pravilima s mogućnostima umjetne inteligencije. Koriste unaprijed definirana pravila za određene zadatke i AI za složenije interakcije, postižući ravnotežu između strukture i prilagodljivosti.

LITERATURA

1. Wikipedia, (2024), Chatbot, https://en.wikipedia.org/wiki/Chatbot (pristupljeno 16. 9. 2024.)
2. Alan M. Turing, (1950), Computing machinery and intelligence
3. Adamopoulou, Eleni, Moussiades, Lefteris, (2020), An Overview of Chatbot Technology, *IFIP Advances in Information and Communication Technology*, 584
4. A.I. For Anyone, AILM, https://www.aiforanyone.org/glossary/aiml (pristupljeno 26. 9. 2023.)
5. Microsoft, (2022), Bot Framework SDK documentation, https://learn.microsoft.com/en-gb/azure/bot-service/index-bf-sdk?view=azure-bot-service-4.0 (pristupljeno 26.9.2023.)
6. Microsoft, (2022), Bot Framework Composer documentation, https://learn.microsoft.com/en-us/composer/ (pristupljeno 26.9.2023.)
7. Microsoft, (2023), Test and debug with the Emulator, https://learn.microsoft.com/en-gb/azure/bot-service/bot-service-debug-emulator?view=azure-bot-service-4.0&tabs=csharp (pristupljenjo 26.9.2023.)
8. Open AI, Documentation ChatGPT API, https://platform.openai.com/docs/introduction/overview (pristupljeno 16.9.2024.)
9. Wikipedia, (2024), ChatGPT, https://en.wikipedia.org/wiki/ChatGPT (pristupljeno 16.9.2024.)
10. Microsoft, (2022), Azure Blob Storage documentation, https://learn.microsoft.com/en-gb/azure/storage/blobs/ (pristupljeno 27.9.2023.)

SAŽETAK

Ovaj diplomski rad opisuje izradu tri vrste chatbotova: chatbot s integriranim ChatGPT modelom u grafičkom alatu Bot Framework Composer, chatbot s integriranim ChatGPT modelom uz pomoć Bot Framework SDK-a te tradicionalni chatbot temeljen na unaprijed definiranim pravilima uz pomoć Bot Framework SDK-a. U okviru rada su opisani alati korišteni za izradu chatbotova. Izradom više vrsta chatbotova došlo se do zaključaka o prednostima i manama pojedinih vrsta chatbotova.

Ključne riječi: chatbot, OpenAI, ChatGPT, Microsoft Bot Framework, Microsoft Bot Framework Composer, Microsoft Bot Framework Emulator

SUMMARY

This thesis is based on three chatbots created: a chatbot with an integrated ChatGPT model in the Bot Framework Composer graphic tool, a chatbot with an integrated ChatGPT model with the Bot Framework SDK and a traditional chatbot based on predefined rules with the Bot Framework SDK. By creating several types of chatbots, a conclusion has been reached about the advantages and disadvantages of each type of chatbot and chatbot creation tool.

Keywords: chatbot, OpenAI, ChatGPT, Microsoft Bot Framework, Microsoft Bot Framework Composer, Microsoft Bot Framework Emulator