Fakultet elektrotehnike i računarstva

Digitalno obrazovanje

Projekt

Korištenje dijaloške tehnike u obrazovanju

Dorian Dinčir

Filip Mutnjaković

Jasmin Redžepović

Tomislav Maslač

Sadržaj

1. Opis zadataka

1.1 Automatski Administrativni Asistent u nastavi

Studenti tijekom semestra postavljaju niz organizacijsko administrativnih pitanja (ne odnose se na znanje u predmetu). Većina pitanja se ponavlja i unutar jedne generacije i između generacija. Neki predmeti grade bazu pitanja i odgovora (FAQ – FrequentlyAskedQuestions), ali studenti ne provjere u toj bazi odgovor na svoje pitanje već ga šalju nastavnicima.

1.2 Automatski odgovori na FAQ

Studenti tijekom učenja predmeta imaju pitanja na koja ne mogu naći odgovor. Najradije bi pitali nastavnika ili znalca. Neki studenti se stide ili boje pitati nastavnike, a neki nastavnici na stignu promptno odgovoriti na pitanja.

1.3 Automatsko ocjenjivanje odgovora na otvorena pitanja

Računalnu provjeru (i samoprovjeru) znanja najlakše je napraviti pitanjima s višestrukim odgovorima. Međutim u tom slučaju se provjerava sposobnost studenta da se prepozna odgovor. Viša razina znanja bi tražila da se sam prisjeti odgovora, ali to znači da odgovor može poprimati različite oblike. Na primjer: „Kad sviće dan?“, „U jutro“, „Nakon noći“, „Jutrom“, „Između 4 i 7“ itd. Dosadašnji računalni programi/alati ne mogu lagano prepoznati točan odgovor, odnosno trebalo bi unijeti sve varijacije odgovora.

1.4 Konverter stabla odluke u Dialogflow

Za potrebe dijagnostike potrebno je s korisnikom proći niz pitanja, od općenitijih do specifičnih. Jedan način organiziranja ekspertnog znanja za dijagnostiku su stabla odlučivanja. Postoji mnogo alata kojima se jednostavno grade i održavaju stabla odlučivanja.
S druge strane, korisnicima je ponekad lakše eksplicirati svoju potrebu ili problem dijalogom.

2. Automatski Administrativni Asistent u nastavi i Automatski odgovori na FAQ

Zbog sličnosti prva dva zadatka kod implementacije oni su promatrani zajedno. Cilj oba zadatka bio je napraviti automat koji će odgovarati na često postavljana pitanja, bila ona vezana za znanje o predmetu ili organizaciji predmeta. Na poznata (već odgovorena) pitanja program bi trebao moći dati zadovoljavajući odgovor, u suprotnom pitanje bi trebalo poslati profesoru koji će znati odgovor. Pri implementaciji korišten je programski jezik C# (.NET), Google Dialogflow te React za korisničko sučelje.

2.1 Baza podataka

Za bazu podataka koristili smo MSSQL. Baza se sastoji od 4 tablice: Users, Questions, Answers i Subjects.

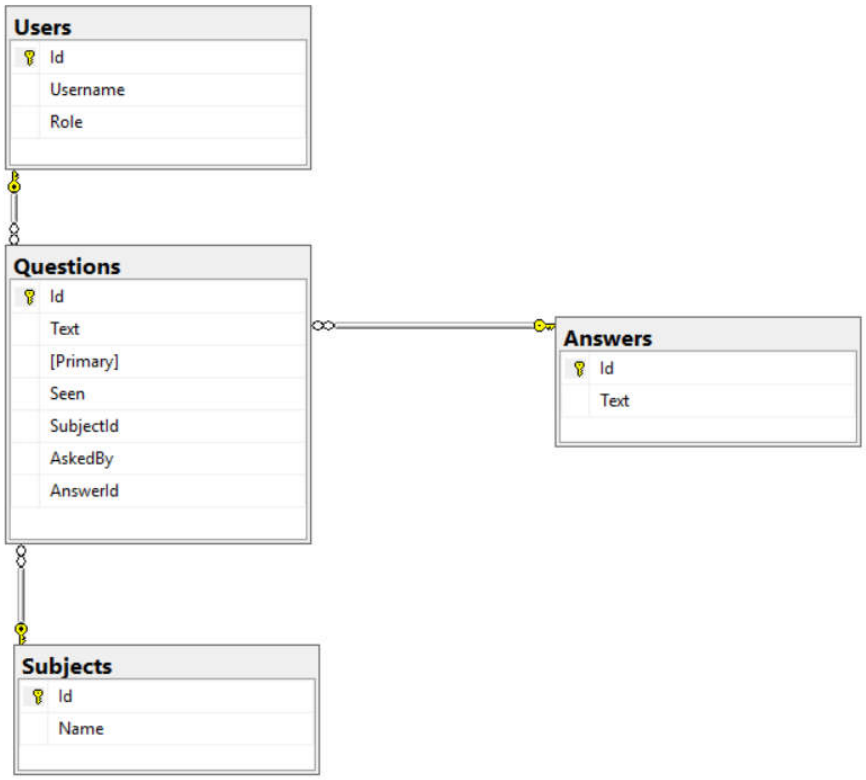
U tablicu Users spremamo podatke o korisnicima aplikacije. Za svakog korisnika pamtimo korisničko ime (Username) i ulogu koju ima (Role). Uloga korisnika može bili ili student ili profesor.

Svi podaci vezani uz pitanja spremaju se u tablicu Questions.

- Text – tekst pitanja
- Primary – TRUE za ona pitanja koja se prikazuju korisniku u aplikaciji
- Seen – je li korisnik pogledao novo odgovoreno pitanje
- SubjectId – Id predmeta za koje je pitanje vezano
- AskedBy – Id studenta koji je postavio pitanje
- AnswerId – Id odgovora na pitanje

Answers tablica služi za pohranu odgovora. Sprema se samo tekst (Text) odgovora.

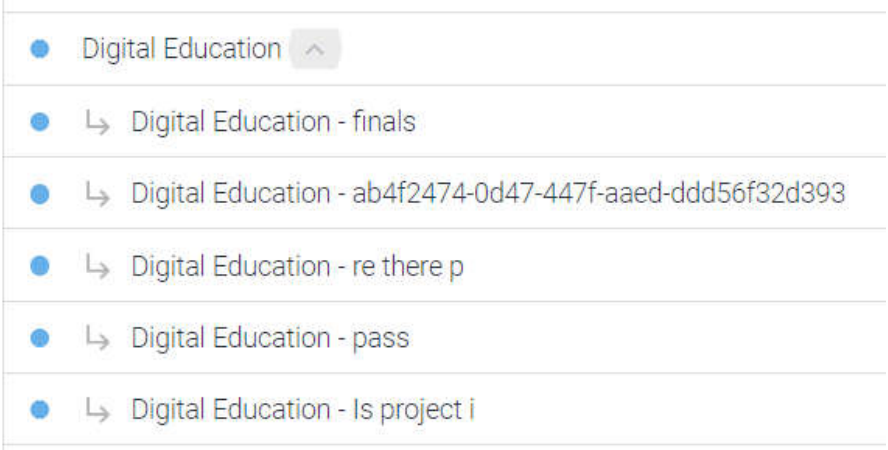
U Subjects tablici nalazi se popis predmeta. Za svaki predmet sprema se samo naziv (Name).



Slika 2.1 Dijagram baze podataka

2.2 Dialogflow struktura podataka

Struktura podataka za ove zadatke organizirana je tako da je korijenski intent ime kolegija, a specifična pitanja su tzv. follow-up intenti. To znači da je najprije potrebno postaviti pitanje kao ime predmeta, a zatim pitanje na koje tražimo odgovor. Svako pitanje može biti postavljeno na jedan ili više načina, ovisno o tome koliko je puta odgovor označen kao točan. Što je više različitih pitanja koja odgovaraju s istim odgovorom (tzv. Training phrases) to će Dialogflow bolje istrenirati mrežu i lakše odgovarati na pitanja.



Slika 2.2 Dialoflow intenti

Prema primjeru sa slike ako je postavljeno pitanje „Digital Education“ tada je iduće postavljeno pitanje vezano za taj predmet. Napomena: korisnik ne mora eksplicitno postavljati pitanje kako bi odredio predmet već ga odabire iz padajućeg izbornika (više u poglavlju Korisničke upute).

Svaki follow-up intent unutar predmeta sadrži training phrases koje su korištene kao pitanja te za treniranje mreže za to pitanja i odgovora.

2.3 API

Komunikaciju s bazom podataka i Dialogflowom omogućili smo preko API-ja. Za API smo koristili Visual Studio i C#. Za spajanje s bazom koristili smo Entity Framework, a Dialogflow nudi svoj API koji smo iskoristili za komunikaciju s njim (https://dialogflow.com/docs/reference/agent). U tablici ispod popisane su i opisane funkcije koje naš API nudi.

Tablica 1 API pozivi

	Url	Opis
USERS	api/users	
GET	/login?username={username}	Vraća id korisnika ili not found, ako ne postoji
POST	/register	Registracija korisnika
QUESTIONS	api/questions	
GET	/unanswered	Pitanja na koja profesor još nije odgovorio

GET	/primary	FAQ lista
GET	/answered?username={username}	Odgovorena pitanja koja je postavio student
PUT	/edit/{id}	Uređivanje pitanja
POST	/new	Dodavanje novog pitanja
POST	/ask	Student postavlja pitanje
DELETE	/remove/{id}	Brisanje pitanja
ANSWERS	api/answers	
POST	/reply/{id}	Dodavanje novog pitanja
POST	/new	Odgovaranje na studentovo pitanje koje nema odgovor. id = id pitanja.
DELETE	/remove/{id}	Brisanje odgovora
PUT	/edit/{id}	Uređivanje odgovora
SUBJECTS	api/subjects	
GET	/all	Popis svih predmeta
PUT	/edit/{id}	Uređivanje predmeta
POST	/new	Dodavanje predmeta
DELETE	/remove/{id}	Brisanje predmeta

2.4 Komunikacija backenda i Dialogflowa

U pozadini aplikacije je .NET web api koji komunicira s frontendom i Dialogflow servisom. U nastavku je opisan Dialogflow i komunikacija servisa s aplikacijom.

Kontroler (razred) zadužen za komunikaciju s Dialogflowom naziva se DialogflowController. Njegova zadaća je komunikacija s Dialogflow servisom i internom bazom podataka. DialogflowController sadrži nekoliko metoda za uspostavljanje komunikacije sa web apijem Dialogflowa: Get, Post, ConfirmAnswer te Ask. Get i Post metode koriste Ask metodu. Get metoda prima zahtjev s korisnika koji šalje id predmeta i odgovarajuće pitanja. Navedena metoda tada koristi metodu Ask kako bi provjerila postoji li odgovor na postavljeno pitanje na Dialogflowu. Ova metoda koristi službeni Googleov SDK čija je primjena opisana u dokumentaciji[1] i primjerima na službenoj Github stranici[2]. U slučaju da odgovor postoji metoda Ask vraća odgovor, a inače vraća

[1] https://dialogflow.com/docs

[2] https://github.com/googleapis/google-cloud-dotnet/tree/master/apis/Google.Cloud.Dialogflow.V2

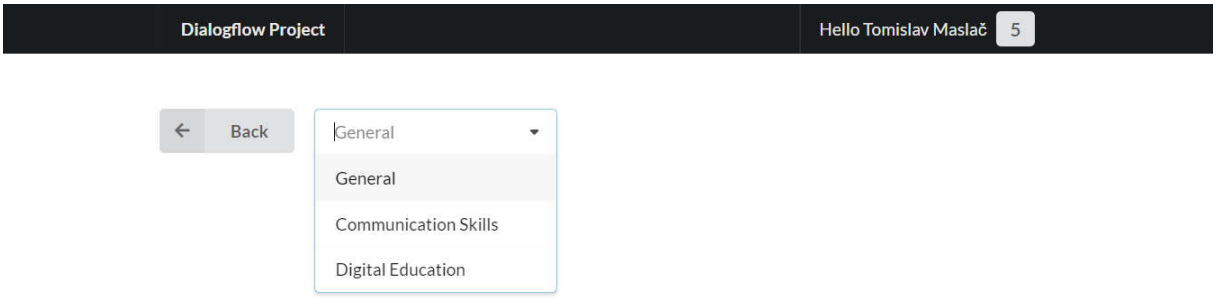
tzv. Fallback intent koji kaže da odgovor na pitanje ne postoji.

2.5 Korisničke upute

Aplikacija je podijeljena na dva dijela: profesorski i studentski. Profesorski dio služi odgovaranju na postavljena pitanja, a studentski za postavljanje pitanja.

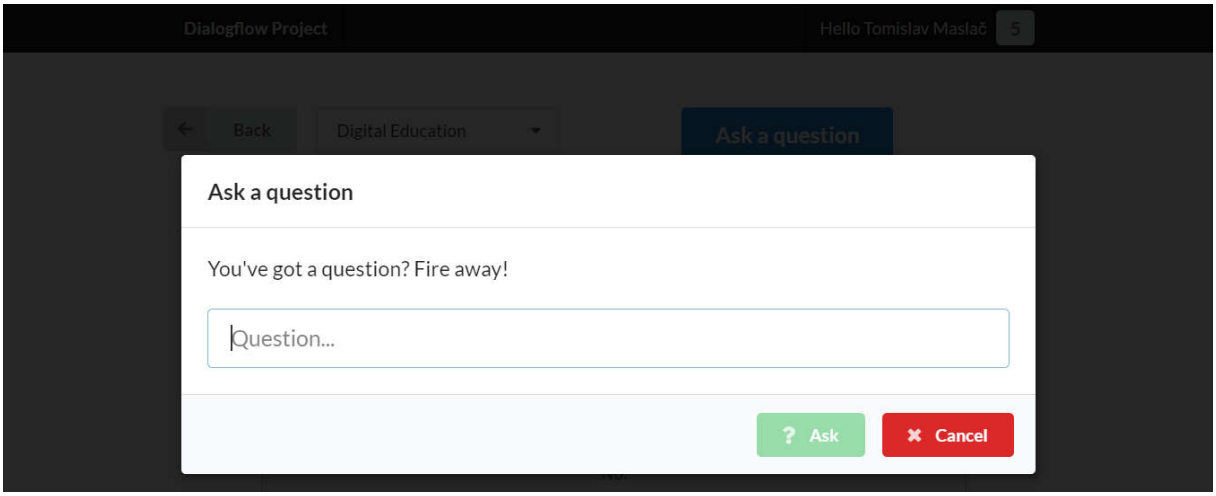
2.5.1 Studentski panel

Prilikom ulaska u aplikaciji student mora odabrati predmet za koji želi postaviti pitanje. Predmeti su već postavljeni u internoj bazi podataka, a lako bi bilo dodati akciju za dodavanje novih ukoliko bi bilo potrebno ili ih dodati izravno u bazu podataka.



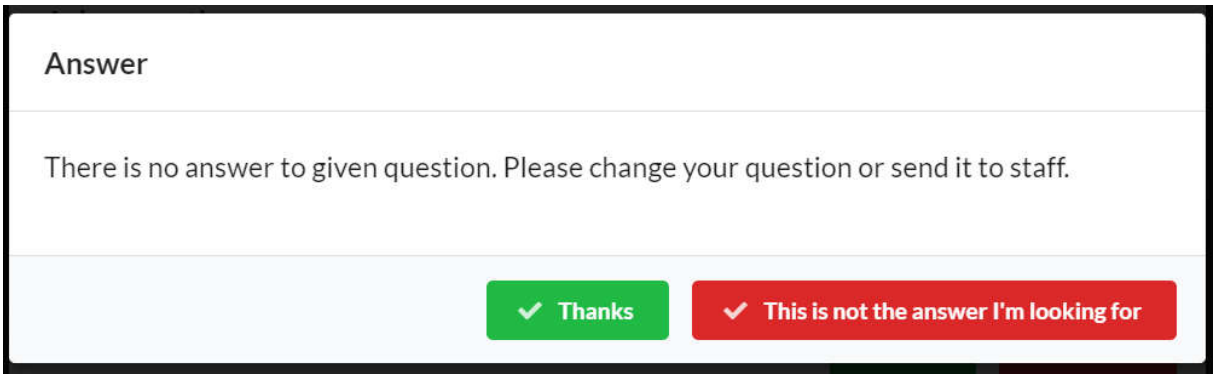
Slika 2.3 odabir predmeta

Po odabiru predmeta dostupna su već postavljena pitanja koja student može čitati ili može postaviti svoje pitanje klikom na gumb „Ask a question“. Klikom na gumb otvara se dijaloški okvir u kojem se može postaviti pitanje (slika 2.3).



Slika 2.4 postavljanje pitanja

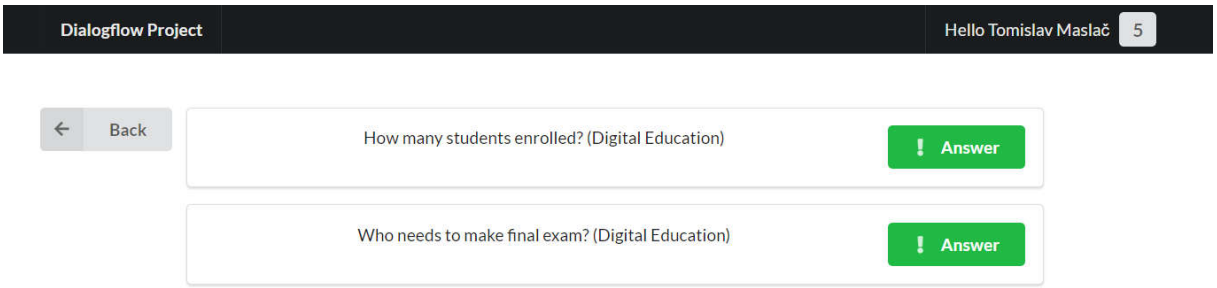
Nakon postavljanja pitanja student dobiva odgovor (slika 2.4). Ukoliko student klikne „Thanks“ odgovor koji je Dialogflow dao označen je kao dobar te se on dodatno trenira pomoću postavljenog pitanja. U suprotnom, student odbija odgovor te može svoje pitanje poslati profesoru.



Slika 2.5 odgovor

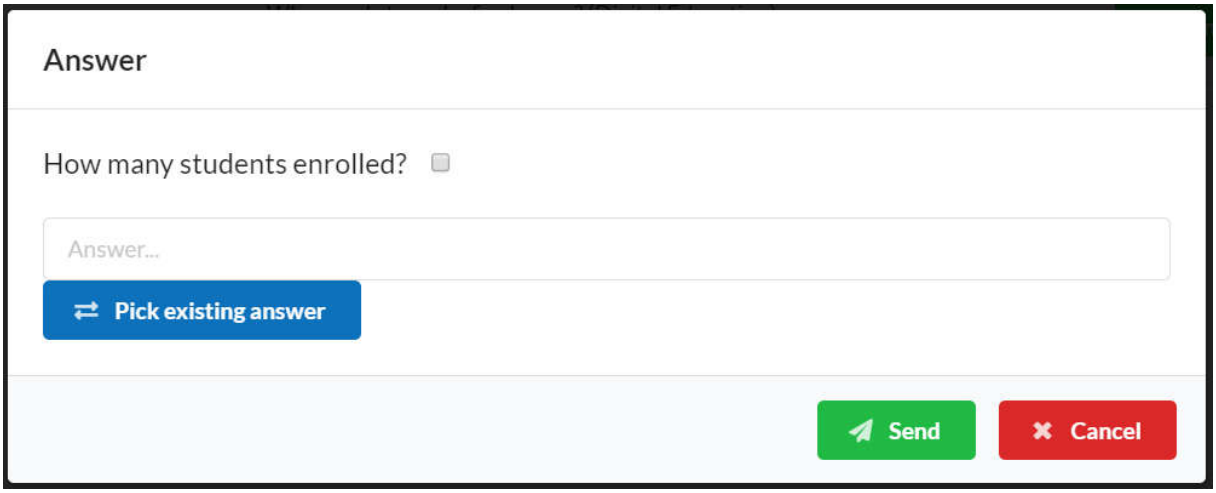
2.5.2 Profesorski panel

Na početnoj stranici profesorovog panela nalaze se neodgovorena pitanja koja su studenti postavili.



Slika 2.6. profesorski panel

Klikom na gumb „Answer“ otvara se dijaloški okvir u kojem profesor može odgovoriti na postavljeno pitanje. Profesor može dati svoj odgovor ili, ako zna da je pitanje već odgovoreno, ali je student svejedno poslao pitanje, odabrati jedno od već postojećih odgovora.



Slika 2.7. odgovaranje na pitanje

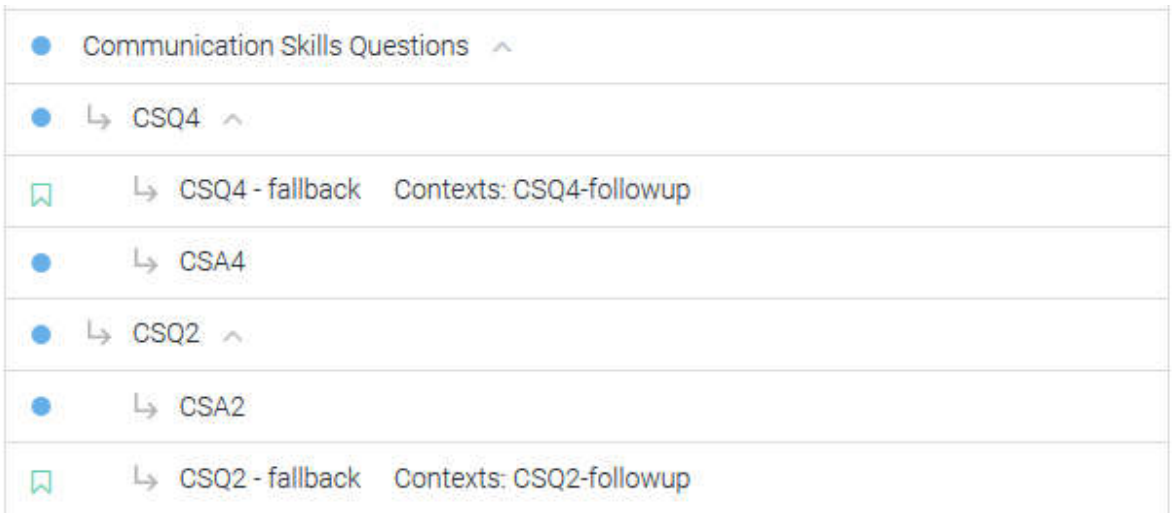
3. Automatsko ocjenjivanje odgovora na otvorena pitanja

Zadatak je bio osmisliti i implementirati računalnu provjeru odgovora na otvorena pitanja korištenjem alata Google Dialogflow. Za implementaciju rješenja uz navedeni alat Google Dialogflow korišteni su za izradu backend-a programski jezik C# i programski okvir (eng. software framework) .NET Framework, a za izradu frontend-a programski jezik JavaScript i programska knjižnica React. Kroz sljedeća poglavlja detaljnije će biti objašnjeno korištenje navedenih alata i programskih jezika, te kako se aplikacija koristi.

3.1 Dialogflow struktura podataka

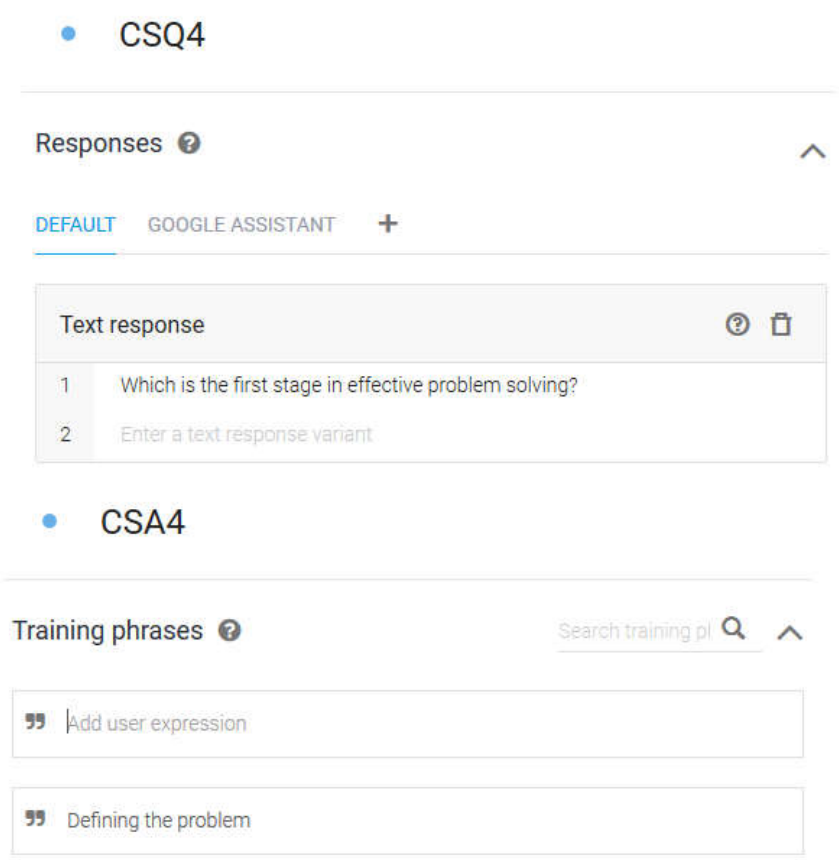
Osnovna namjena alata Dialogflow je ostvariti interakciju između računala i čovjeka kroz razgovor, a to se odvija korištenjem namjera (eng. intent). Svaka namjera sadrži fraze za treniranje i odgovore. U razgovoru namjera čije se fraze za trening najviše podudaraju s unesenim sadržajem pitanja se odabire i Dialogflow vraća jedan od odgovora iz namjere. Do sada su fraze za trening bila moguća pitanja ili izjave sugovornika za koje bi bile već unaprijed pripremljene reakcije, a za provjeru točnosti odgovora fraze za trening će biti točan odgovora za svako pitanje.

Kako bi mogli navigirati po Dialogflow-u i unositi pitanja i odgovore za pojedine predmete koristimo mogućnost grananja namjera (slika 3.1.). Korijen grananja nam je naziv predmeta, njegova djeca su nam pitanja, a svako pitanje ima dvije namjere, ako je odgovor točan ili netočan.



Slika 3.1. Prikaz strukture grananja

Iz korijenske namjere kao povratnu informaciju dobijemo broj pitanja koji sadrži kao djecu. Kada zovemo redni broj nekog pitanja, nakon odabira korijena, povratna informacija nam je pitanje koje se postavlja i na koje je potrebno dati odgovor (slika 3.2.). Kada unesemo odgovor ako je točan odabire se namjera s točnim odgovorom, ako nije odabire se namjera za povlačenje (eng. fallback) koja nam govori da je odgovor netočan.

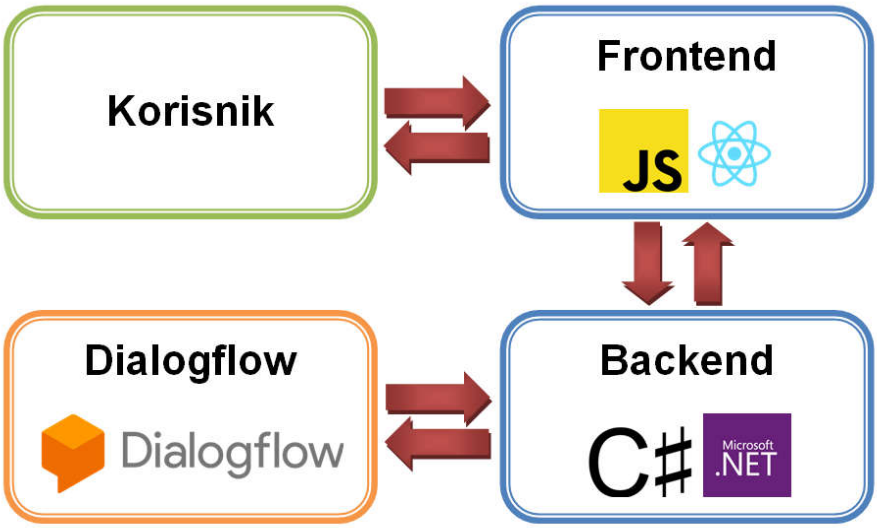


Slika 3.2. Namjera CSQ4 sadrži pitanje, a namjera CSA4 sadrži točan odgovor na pitanje

Kako bi bili sigurni da će Dialogflow prepoznati točan odgovor potrebno je u faze za trening namjere za provjeru odgovora unijeti što više varijacija točnog odgovora. Pogotovo kada na jedno pitanje može biti više različitih točnih odgovora.

3.2 Arhitektura

Aplikacija se sastoji od tri dijela (slika 3.3.): fontend-a preko kojega korisnik dobiva pitanja i unosi odgovore, backend-a koji služi kao komunikacija između frontend-a i Dialogflow i Dialogflow koji služi za skladištenje pitanja i odgovora, te provjerava točnost odgovora.



Slika 3.3. Arhitektura aplikacije

Korisnik svu komunikaciju vrši samo a frontend-om. Frontend je zadužen za prikaz predmeta i svih pitanja koje predmeti imaju. Nakon što korisnik unese odgovore frontend prikazuje povratnu informaciju da li je korisnik unio točne ili netočne odgovore. Fronted prosljeđuje backend-u sve podatke i očekuje odgovore.

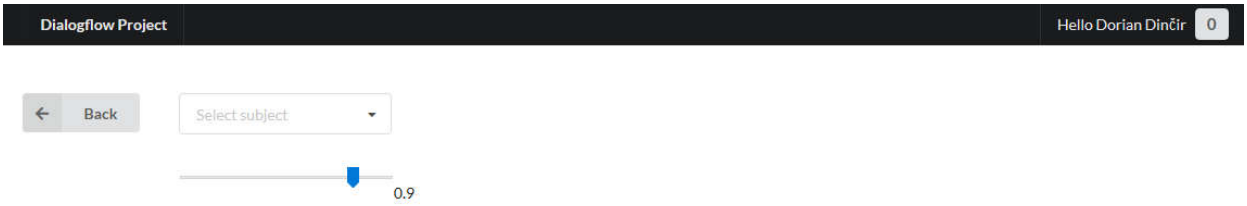
Backend sadrži dva kontrolera: DialogflowController i SubjectsController. SubjectsController sadrži funkciju GetSubjects koja vraća listu predmeta za koje postoje pitanja. DialogflowController ima dvije funkcije: CheckAnswer i GetQuestion. Funkcija GetQuestion kao parameter prima id predmeta. Šalje zahtjev Dialogflow-u koji vraća broj pitanja koji sadrži predmet, ovisno o broju pitanja n funkcija šalje n zahtjeva Dialogflow-u koji vraća pitanja koja funkcija zapisuje u listu. Lista pitanja se prosljeđuje frontend-u. Funkcija CheckAnswer kao parameter prima id predmeta, redni broj pitanja i odgovor. Upit se šalje Dialogflow-u koji iz strukture, koja nam je poznata od prije, vraća

da li je odgovor točan ili netočan. Ako je odgovor točan funkcija vraća koliko je sigurna da je odgovor točan u interval između nula i jedan. U slučaju da je odgovor netočan vraća nulu.

Dialogflow sadrži strukturu objašnjenu u prošlom poglavlju i vrši komunikaciju samo s backend-om. Ima tri funkcije: povrat informacije o broju pitanja, prosljeđivanje pitanja I provjeru ispravnosti odgovora.

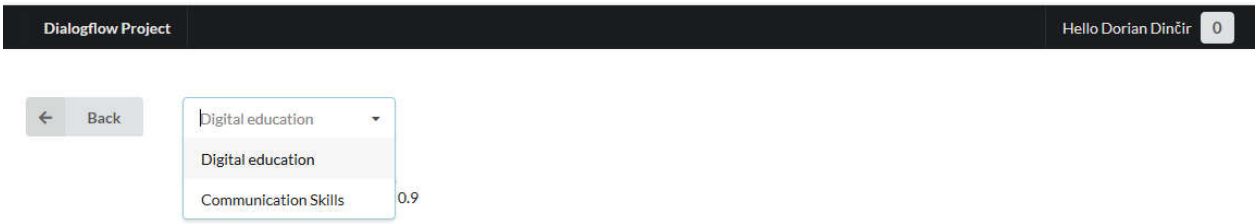
3.3 Korisničke upute

Nakon pokretanja aplikacije na skali koja se nalazi ispod padajućeg izbornik odabiremo postotak za koji odgovor mora biti točan kako bi se priznao (slika 3.4.).



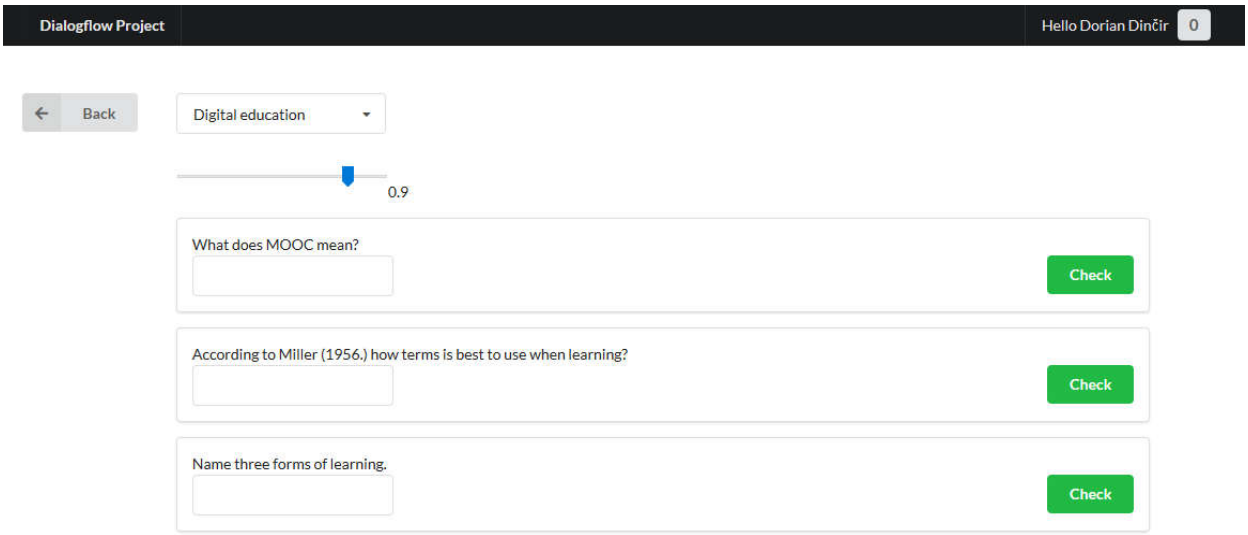
Slika 3.4. Odabir postotka točnosti

Iz padajućeg izbornika potrebno je odabrati predmet za koji želimo dobiti pitanja (slika 3.5.).



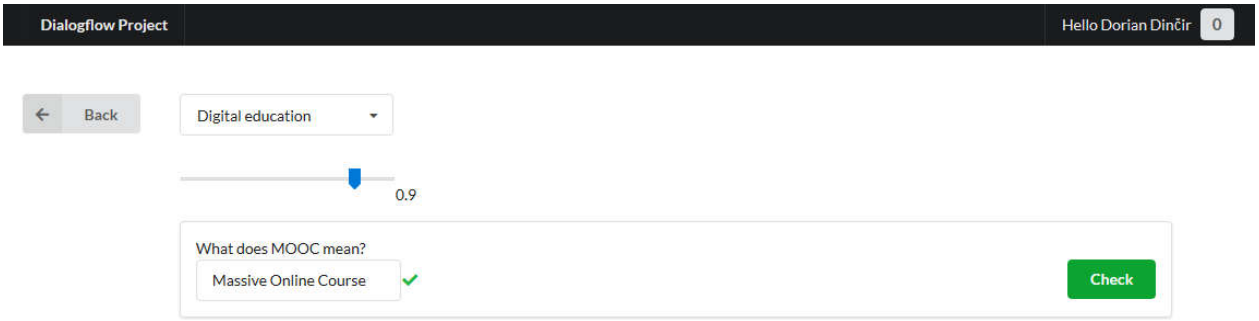
Slika 3.5. Odabir predmeta

Nakon što smo odabrali predmet ispisat će se sva pitanja na koja je potrebno dati odgovor (slika 3.6.).

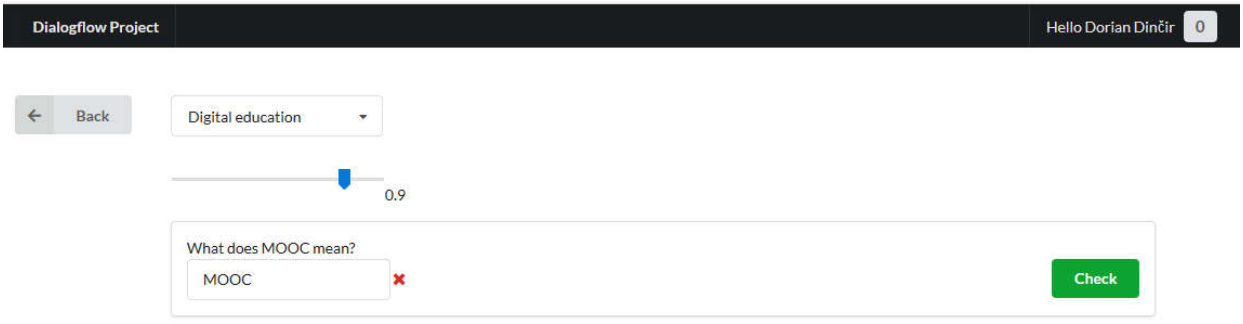


Slika 3.6. Popis svih pitanja

Nakon što upišemo odgovor i pritisnemo gumb Check ako je odgovor točan pojavit će se zelena kvačica (slika 3.7.), u suprotnom ako je netočan pojavit će se crveni iks (slika 3.8.).



Slika 3.7. Ispravan odgovor



Slika 3.8. Neispravan odgovor

4. Konverter stabla odluke u Dialogflow

Zadatak je bio napraviti modul koji pretvara stablo odluke u dijaloški tok. To je napravljeno pomoću programskog jezika Python i library-a dialogflow_v2(https://github.com/googleapis/dialogflow-python-client-v2) .

Napravljen je Dialogflow projekt pomoću vlastitog gmail računa za spremanje dijaloga. Ideja je bila napraviti opću bazu znanja za rješavanje problema koji se mogu pretvoriti u stablo odluke. To se može ostvariti tako da se više stabala odluke pretvori u dijaloge i spremi na Dialogflow projekt. Nakon što bi se napravili potrebni dijalozi, slao bi se upit na već napravljeni Dialogflow projekt koji bi znao razlučiti o kojem se problemu radi te ponudio daljnja pitanja kako bi došao do rješenja. Kako svaki intent ima fraze za treniranje, pri pretvorbi stabla odluke u dijalog korijenskom intentu treba zadati željene fraze pomoću kojih će Dialogflow istrenirati model te kasnije detektirati o kojem se od mogućih više stabala radi. Osmišljena je vlastita struktura stabla odluke koja je korištena pri izradi dijaloga, a prikazana je u nastavku.

```
tv is broken;tv not working;tv has black screen
is cable connected;1
        1is sound on;2
                1take it to the service company;4
                0turn on sound;5
        0connect the cable and try again;3
```

Prvi redak pretstavlja fraze za treniranje koje opisuju kako korisnik može reći da nešto nije uredu s tv-om. Zatim slijede čvorovi stabla od kojih neki mogu biti uvučeni „tabovima“ kako bi se opisala pripadnost drugom čvoru(roditelju). Broj iza znaka „;“ označava redni broj čvora, kao neki id. Broj ispred čvora koji može biti 0 ili 1 predstavlja pozitivan ili negativan odgovor, tj. da ili ne.

Pomoću slijedeće funkcije prethodno opisana struktura se učita i pripremi za stvaranje dijaloga.

```
def_recurse_tree(parent, depth, source, relations):
last_line = source.readline().rstrip()
whilelast_line:
        tabs = last_line.count('\t')
if tabs < depth:
break
        node = last_line.strip()
if tabs >= depth:
if parent isnotNone:
relations.append(parent + ':' + node)
last_line = _recurse_tree(node, tabs+1, source, relations)
returnlast_line
```

Rezultat _recurse_tree funkcije je prikazan u nastavku.

is cable connected;1:1is sound on;2

1is sound on;2:1take it to the service company;4

1is sound on;2:0turn on sound;5

is cable connected;1:0connect the cable and try again;3

Vidi se svaki čvor i njegovi izravni sljedbenici odvojeni znakom „:“.

Nakon toga kreće stvaranje dijaloga. Prvo se stvori korijenski intent, a zatim mu se dodaju follow-up intenti tako da se prati struktura zadanog stabla. Može se i follow-up intentima također dodati njihovi follow-up intenti.

U nastavku su prikazane funkcije za stvaranje intenta i follow-up intenta.

```
defcreate_intent(project_id, session_id, display_name, training_phrases_parts,
message_texts):
"""Create an intent of the given intent type."""
intents_client = df.IntentsClient()

    parent = intents_client.project_agent_path(project_id)
training_phrases = []
fortraining_phrases_partintraining_phrases_parts:
        part = df.types.Intent.TrainingPhrase.Part(
text=training_phrases_part)
# Here we create a new training phrase for each provided part.
training_phrase = df.types.Intent.TrainingPhrase(parts=[part])
training_phrases.append(training_phrase)

    text = df.types.Intent.Message.Text(text=message_texts)
    message = df.types.Intent.Message(text=text)
out_c =
df.types.Context(name='projects/{}/agent/sessions/{}/contexts/{}'.format(project_
id, session_id, 'c' + display_name + '1'),
lifespan_count=1)

    intent = df.types.Intent(
display_name=display_name,
training_phrases=training_phrases,
messages=[message],
output_contexts=[out_c])

    response = intents_client.create_intent(parent, intent)

print('Intent created: {}'.format(response))
```

```
defcreate_followup(project_id, session_id, display_name, training_phrases_parts,
message_texts, inp, out, parent_int):
"""Create an intent of the given intent type."""
intents_client = df.IntentsClient()

    parent = intents_client.project_agent_path(project_id)
training_phrases = []
fortraining_phrases_partintraining_phrases_parts:
        part = df.types.Intent.TrainingPhrase.Part(
text=training_phrases_part)
# Here we create a new training phrase for each provided part.
training_phrase = df.types.Intent.TrainingPhrase(parts=[part])
training_phrases.append(training_phrase)

    text = df.types.Intent.Message.Text(text=message_texts)
    message = df.types.Intent.Message(text=text)


followup_intent=df.types.Intent.FollowupIntentInfo(followup_intent_name=display_n
ame,
parent_followup_intent_name=parent_int)

out_c =
df.types.Context(name='projects/{}/agent/sessions/{}/contexts/{}'.format(project_
id, session_id, 'c' + out),
lifespan_count=1)

p_id = _get_intent_ids(project_id, parent_int)[0]

    intent = df.types.Intent(
display_name=display_name,
training_phrases=training_phrases,
messages=[message],
input_context_names=["projects/{}/agent/sessions/{}/contexts/{}".format(project_i
d, session_id, 'c' + inp)],
output_contexts=[out_c],
parent_followup_intent_name="projects/{}/agent/intents/{}".format(project_id,
p_id),
followup_intent_info=[followup_intent])

    response = intents_client.create_intent(parent, intent)

print('Intent created: {}'.format(response))
```

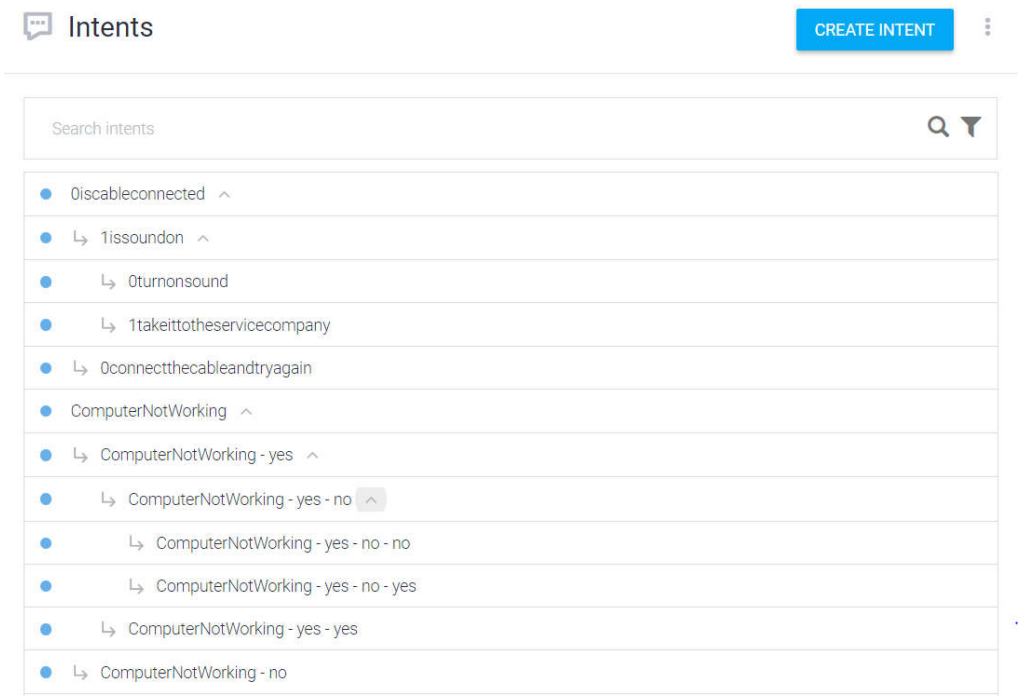
```
def_get_intent_ids(project_id, display_name):
intents_client = df.IntentsClient()

    parent = intents_client.project_agent_path(project_id)
    intents = intents_client.list_intents(parent)
intent_names = [
        intent.name for intent in intents
ifintent.display_name == display_name]

intent_ids = [
intent_name.split('/')[-1] forintent_name
inintent_names]

returnintent_ids
```

Linkovi koje sam koristio za pomoć:

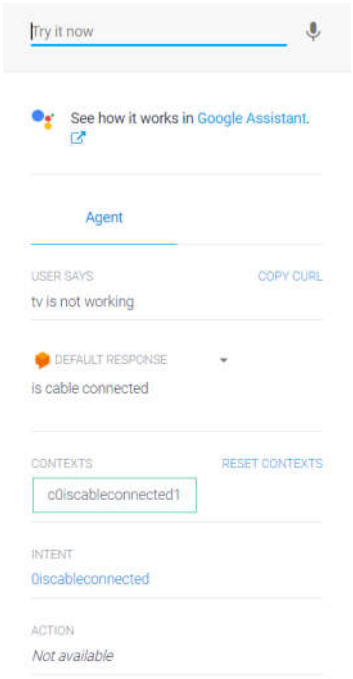
https://github.com/googleapis/dialogflow-python-client-v2/issues/66

https://cloud.google.com/dialogflow-enterprise/docs/reference/rest/v2-overview

Na drugom linku je specifikacija Dialogflow api-ja gdje se vidi što je potrebno slati za pojedinu akciju.



Slika 4.1. Prikaz 2 stabla odluke pretvorena u dijaloge, tvNotWorking(gore je 0iscableconnected) i ComputerNotWorking.



Slika 4.2. Na izjavu da tv ne radi, uspješno je detektiran intent tvNotWorking(0iscableconnected) i počinje razgovor kako bi se rješio problem.