Izrada agenta za razgovor na Web stranici

Miškić, Petar

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Petar Miškić

IZRADA AGENTA ZA RAZGOVOR NA WEB STRANICI

ZAVRŠNI RAD

Varaždin, 2021.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Petar Miškić

JMBAG: 0016140059

Studij: Poslovni sustavi

IZRADA AGENTA ZA RAZGOVOR NA WEB STRANICI

ZAVRŠNI RAD

Mentor:

Izv. prof. dr. sc. Schatten Markus

Varaždin, rujan 2021.

Petar Miškić

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.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

U ovome završnome radu prikazana je izrada i obuka agenta za razgovor uz pomoć modula Django ChatterBot. Početkom rada opisane su metode, tehnike i alati korišteni u izradi ovog rada te potom su detaljno opisani glavni moduli, Django i ChatterBot. Idućim poglavljem upoznajemo se s umjetnom inteligencijom i strojnim učenjem i na koji način oni funkcioniraju. U najvažnijem dijelu rada idemo korak po korak kako bi na što jednostavniji način razradili proces izrade jednoga agenta za razgovor. U „Django ChatterBot“ ulazimo u suštinu rada odnosno integraciju našeg agenta s Djangom. Na kraju rada zaključujemo i rezimiramo kompletan rad.

Ključne riječi: ChatterBot, Django,bot, agent, Python, UI

Sadržaj

1. Uvod

U ovome radu provest ću vas kroz izradu pametnog agenta za razgovor u Django ChatterBot okruženju. Započeti ću s detaljnim opisom metoda i tehnika rada te s alatima, bibliotekama i ostalim modulima koji su korišteni u radu. Zatim ću detaljno objasniti što je to umjetna inteligencija, njezina povijest, trendovi, prednosti i nedostatci. Potom ćemo se dotaći pojma strojnog učenja i zašto je ono popularno u ovome trenutku. Nedugo zatim dolazimo do dijela o ChatterBotu, u kojem ćemo detaljno prikazati njegovu instalaciju, obuku, adaptere i pretprocesore. Na kraju dolazimo do same integracije ChatterBota s Djangom, gdje ćemo napraviti virtualno okruženje, pokrenuti projekt i implementirati agenta za razgovor sa Djangom.

Za pisanje rada najviše sam se koristio online izvorima literature od kojih bih najviše izdvojio ChatterBot i Django dokumentaciju koja su davale odgovore na sva moja pitanja te su se pokazale kao sjajan vodič kroz razumijevanje čitave logike projekta. Također prilikom same izrade agenta koristio sam ChatterBot github koji mi je dao razne ideje i pomogao u načinu izrade implementacije agenta.

2. Metode i tehnike rada

U razvoju agenta za razgovor i web aplikacije u sklopu izrade ovog rada,korišteno je nekoliko alata, modula, biblioteka te programskih jezika. U ovom poglavlju bit će opisani sve korištene tehnologije pri izradi ovog rada.

2.1 HTML

HTML (engl. *Hypertext Markup Language*) je prezentacijski jezik za izradu web stranica. HTML jezikom oblikuje se sadržaj i stvaraju se hiperveze hipertext dokumenta. Temeljna zadaća HTML jezika jest dati uputu web pregledniku o tome kako prikazati hipertext dokument, pri tome da taj dokument izgleda jednako bez obzira o kojemu je web pregledniku, računalu i operacijskog sustavu riječ [4]. U HTML-u se ne primjenjuje nikakva logika te zbog toga se on ne smatra programskim jezikom. Trenutna verzija HTML-a koja je u upotrebi jeste verzija 5.

2.2 CSS

CSS *(engl. Cascading Style Sheets)* je stilski jezik, koji se rabi za opis prezentacije dokumenta napisanog pomoću HTML jezika. CSS je jedan od osnovnih jezika otvorenog weba i standardiziran je u svim web preglednicima. Uz pomoć CSS-a veoma lagano možemo mijenjati boje, font ili izabrati veće razmake između elemenata. Posljednja verzija koje je trenutno u upotrebi jeste treća verzija CSS-a.

2.3 Chatterbot modul

Bot odnosno agent je softverska aplikacija koja pokreće automatizirane zadatke putem interneta. Izvršeni zadaci su jednostavni i ponavljaju se, ti se zadaci obično obavljaju mnogo brže nego što bi čovjek uspio. Primjerice, vremenski bot je programiran tako da nam daje trenutno vrijeme uz pomoć određenog izvora podataka. Ako pitamo vremenskog bota koliko je trenutno sati u Zagrebu, on će nam reći točno vrijeme ali ako ga pitamo kakva je vremenska prognoza u Splitu, zbunit će se i dati će nam slučajne odgovore. Agent u ovome radu je implementiran uz pomoć modula chatterbot. Chatterbot je Python modul koji olakšava generiranje automatiziranih odgovora za unos korisnika. Chatterbot koristi odabir algoritama strojnog učenja za stvaranje različitih vrsta odgovora, to pojednostavljuje chat agenata i automatizaciju razgovora s korisnicima. Važno za spomenuti jeste da chatterbot je jezično neovisan odnosno on omogućuje govor bilo kojeg jezika.

2.4 Django

Django je Python web okvir visoke razine koji omogućuje brzi razvoj sigurnih i održivih web stranica. Izgrađen je od strane iskusnih programera [2]. Django je jedan od najpopularnijih web okvira u Pythonu.

2.5 Ostali alati, biblioteke i programski jezici

U ovom poglavlju će ukratko biti opisani ostali korišteni alati, biblioteke i programski jezici. U radu su još korišteni:

- **Python** je interpretirani programski jezik opće namjene na visokoj razini.
- **Visual Studio Code** je integrirano radno okruženje napravljeno od strane Microsofta te se također koristi i na macOS-u i Linuxu.
- **SQLite** je sustav za upravljanje relacijskim bazama podataka.
- **Anaconda navigator** je grafičko korisničko sučelje koje omogućuje pokretanje aplikacija i jednostavno upravljanje conda paketima, okruženjima bez upotrebe naredbi u naredbenom retku.

3. Umjetna inteligencija

Umjetna inteligencija (eng. Artificial intelligence) opisuje područje računalne znanosti koja se bavi razvojem inteligentnih alata (strojeva, aparata, aplikacija) koji reagiraju i uče kao ljudi [1]. U ovo područje ulaze i pojmovi poput machine learning (strojno učenje) i IOT (internet of things). Tehnološki dizajn AI sustava, između ostalog, uključuje i razumijevanje i analizu jezika, govora, slike, prema čemu sustav uči kako reagirati, planirati ili rješavati određene zadatke [3]. Općenito sustavi umjetne inteligencija rade na način unosa velike količine podataka, analiziraju podatke te ih koriste za predviđanje budućih stanja. Danas se često postavlja pitanje o značajnosti umjetne inteligencije. Umjetna inteligencija je veoma važna jer može dati poduzećima u bilo kojem području djelovanja uvid u njihovo poslovanje za koje možda ranije nisu bili upoznati, također umjetna inteligencija je posebno efikasna kada su u pitanju ponavljajući zadaci, poput analize velikog broja pravnih dokumenata. Prije vala umjetne inteligencije bilo bi teško zamisliti korištenje softvera za povezivanje korisnika s taxi službom, za primjer toga možemo uzeti Uber koji se plasirao na tržište s takvom tehnologijom i postao jedna od najvećih svjetskih tvrtki. Interakcija između običnog čovjeka i umjetne inteligencije je postala svakodnevnica, na primjer komunikacija sa „smart assistant“ kao što su Siri, Alexa, pregledavanje prijedloga Netflixovih filmova, razmjena informacija s raznim chat botovima, korištenje google ili apple maps itd.

3.1 Povijest umjetne inteligencije

Iako se ovih dana mnogo priča o umjetnoj inteligenciji, to zapravo nije nešto novo, ideja je nastala prije nego što je postala popularna danas. Prvo razmišljanje o računalnim mašinama i njihovoj inteligenciji je zapisano u znanstvenom radu Alana Turinga 1950. godine. Nedugo poslije napisan je prvi program temeljen na umjetnoj inteligenciji od strane Allena Newella, Cliffa Shawa i Herberta Simona [4]. Također bitno je spomenuti da je 1964. godine izumljen prvi chatbot po imenu Eliza, prvo autonomno vozilo je napravljeno 1974. godine na univerzitetu Stanford, IBM-ova umjetna inteligencija 1997. godine je pobijedila u šahu najboljeg svjetskog šahista Garryja Kasparova [4].

3.2 Trendovi umjetne inteligencije

3.2.1 Umjetna inteligencija u umjetničkom stvaralaštvu

U velikoj mjeri umjetnici i glazbenici koriste umjetnu inteligenciju za stvaranje umjetnosti i glazbe ovisno o tome što ljudi najčešće slušaju. Time se osigurava da umjetnici slijede moderne trendove i tako zarađuju više novca. Primjer glazbene platforme bazirane na umjetnoj inteligenciji je MuseNet.

3.2.2 Umjetna inteligencija na društvenim mrežama

Naša aktivnost na društvenim mrežama se koristi za preporuku novih stranica, objava, videozapisa koje bi nas mogle zanimati. Primjer umjetne inteligencije i strojnog učenja (eng. machine learning) je aplikacija Facebook koja uz pomoć “face recognition” mogućnosti označava osobu i njegove prijatelje na slikama.

3.2.3 Umjetna inteligencija agenta za razgovor

Virtualni pomoćnici su postali svakodnevnica našeg života. Uz pomoć njih možemo rezervirati taksi, naručiti hranu, upravljati svjetlima u našemu domu, provjeriti raspored itd. Neki poznati primjeri virtualnih pomoćnika današnjice su:

- Siri (Apple)
- Cortana (Microsoft)
- Alexa (Amazon)
- Google Assistant (Google)

3.2.4 Autonomna vozila

Auti umjetne inteligencije danas postaju sve više popularniji jer su pametniji i sigurniji u vožnji. Oni rade tako da prikupljaju podatke s GPS-a, senzora i kamera u automobile te pomoću umjetne inteligencije predviđaju koji su objekti prisutni u blizini vozila i što će učiniti sljedeće. Automobili se sami prilagođavaju ili upozoravaju vozača na moguće pogreške. Tesla je najbolji primjer današnjim autonomnih vozila, kompanija je briljirala do te mjere da automobile voze bez ikakve ljudske intervencije uz pomoć umjetne inteligencije.

3.2.5 Umjetna inteligencija u zdravstvu

Mnogi medicinski centri i bolnice širom svijeta su se počeli oslanjati na umjetnu inteligenciju. Poduzeća su razvila programe uz pomoć umjetne inteligencije koji prikupljaju podatke od pacijenata te im postavljaju dijagnozu bez da pacijent mora dolaziti redovito u bolnicu.

3.3 Prednosti i nedostatci umjetne inteligencije

Prednosti:

1. Efikasnost u poslovima orijentiranim na detalje
2. Ušteda vremena kod rješavanja zadataka s puno podataka
3. Virtualni agenti s UI su uvijek dostupni

Nedostatci:

1. Veoma skupa tehnologija
2. Ograničena ponuda kvalificiranih radnika
3. Nedostatak sposobnosti generaliziranja s jednog zadatka na drugi

4. Strojno učenje

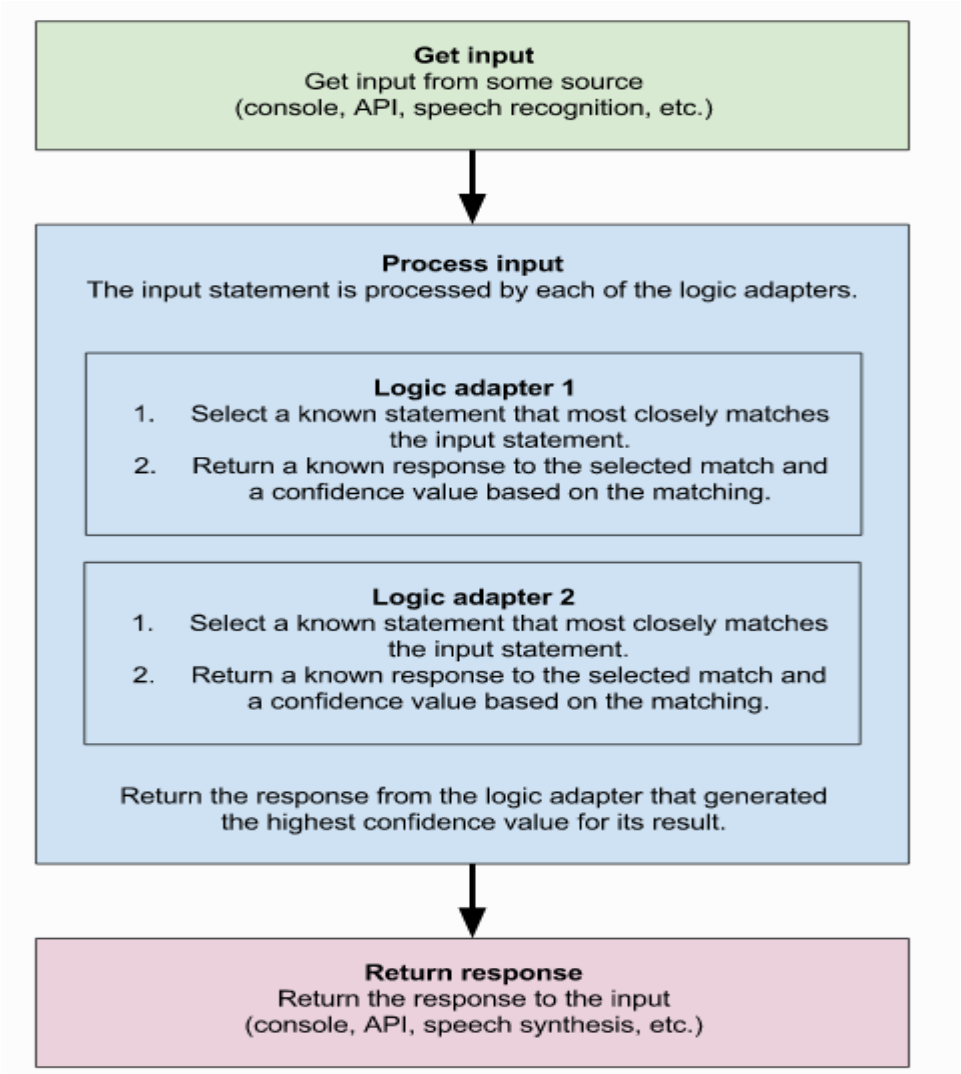
Ono što nas ljude čini posebnima jeste sposobnost da učimo i napredujemo u životu, svaki dan učimo iz svojih iskustava i postajemo sve bolji u svojim zadacima. Tako možemo reći i za računala koji također mogu učiti iz različitih ulaznih podataka koje primaju i baš takav pojam nazivamo strojno učenje (eng. machine learning). Strojno učenje je grana umjetne inteligencije koja se bavi oblikovanjem algoritama koji svoju učinkovitost poboljšavaju na temelju empirijskih podataka [5]. Strojno učenje jedno je od danas najaktivnijih i najuzbudljivijih područja računarske znanosti, ponajviše zbog brojnih mogućnosti primjene koje se protežu od raspoznavanje uzoraka i dubinske analize podataka do robotike, računalnog vida, bioinformatike i računalne lingvistike [5]. Strojno učenje danas se koristi obično za prepoznavanje glasa, teksta i slika. Primjer strojnog učenja je obučavanje računala da može razlikovati automobile i zrakoplove, za početak je potrebno unijeti slike automobila i slike aviona te prikazati računalu što je avion a što automobil. Uz pomoć toga računalo će tražiti uzorke unutar tih slika i to će omogućiti računalu da samostalno prepozna automobil ili avion. Na primjer računalo bi moglo otkriti da zrakoplovi imaju krila, a automobili nemaju. Strojno učenje se uveliko koristi kod prepoznavanja lica, pretvorbe teksta u govor, preporuke oglasa na društvenim mrežama, usavršavanje rezultata tražilice. Naime u strojnom učenju mogu postojati greške, međutim što više podataka računalo primi, njegov algoritam postaje sve više optimiziraniji, pa će sa tom optimizacijom njegova predviđanja povećati točnost.

4.1 Zašto umjetna inteligencija i strojno učenje sada?

U posljednjih nekoliko godina dosta je faktora rezultiralo da strojno učenje i umjetna inteligencija postanu sve više popularniji. Najvažniji čimbenik je velika količina podataka koja se stvara svake minute. Zapravo, 90% svjetskih podataka generirano je u posljednje dvije godine. Zbog ogromnog potencijala mnoga poduzeća ulažu mnogo novca u umjetnu inteligenciju.

5. Kako chatterbot funkcionira

Već gore je u sekciji metode i tehnike rada spomenuto je da chatterbot je Pythonov modul koji je osmišljen kako bi olakšao stvaranje softvera koji može sudjelovati u razgovoru. Komponenta ChatterBot-a počinje bez znanja o komunikaciji. Svaki put kada korisnik unese iskaz, knjižnica sprema tekst koji je unio i tekst na koji je unos odgovoren. Kako ChatterBot prima više unosa, povećava se broj odgovora na koje može odgovoriti i točnost svakog odgovora. Program odabire „najbliži“ odgovor pretraživanjem iz „najbližeg“ poznatog iskaza koji odgovara unosu, a zatim odabire odgovor iz odabira poznatih odgovora na tu izjavu. Detaljniji prikaz se može vidjeti na slici ispod [6].



Slika 1:Dijagram tijeka procesa

5.1 Instaliranje chatterbot modula

Prije samog početka korištenja modula potrebno ga je instalirati. Postoji više načina za implementiranje chatterbot modula.

Za početnike, preporučuje se da započnu s instaliranje**m** najnovije verzije putem Python Indeks paketa (Pypi) uz pomoć sljedeće naredbe:

```
pip install chatterbot
```

- Drugi način instaliranja chatterbot modula jeste izravno preko GitHuba, uz pomoć sljedeće naredbe:

```
pip install git+git://github.com/gunthercox/ChatterBot.git@master
```

Ako već imamo instaliran ChatterBot i želimo provjeriti verziju koju smo instalirali, to radimo uz pomoć sljedeće naredbe:

```
python -m chatterbot --version
```

Kao i svaki softver, chatterbot će se s vremenom mijenjati i donositi nove verzije. Pa tako i postoji sljedeća naredba za nadogradnju postojeće verzije:

```
pip install chatterbot –upgrade
```

5.2 Kreiranje novog agenta

Nakon instaliranja ChatterBot modula prvo je potrebno napraviti novog chat bota.

```
from chatterbot import ChatBot

chatbot = ChatBot("Example bot")
```

Prvo je potrebno iz chatterbot knjižnice „uvesti“ klasu „ChatBot“ uz pomoć koje pravimo našeg agenta za razgovor. Jedini potreban parametar za „ChatBot“ je ime, koje je u ovom slučaju „Example chat bot“.

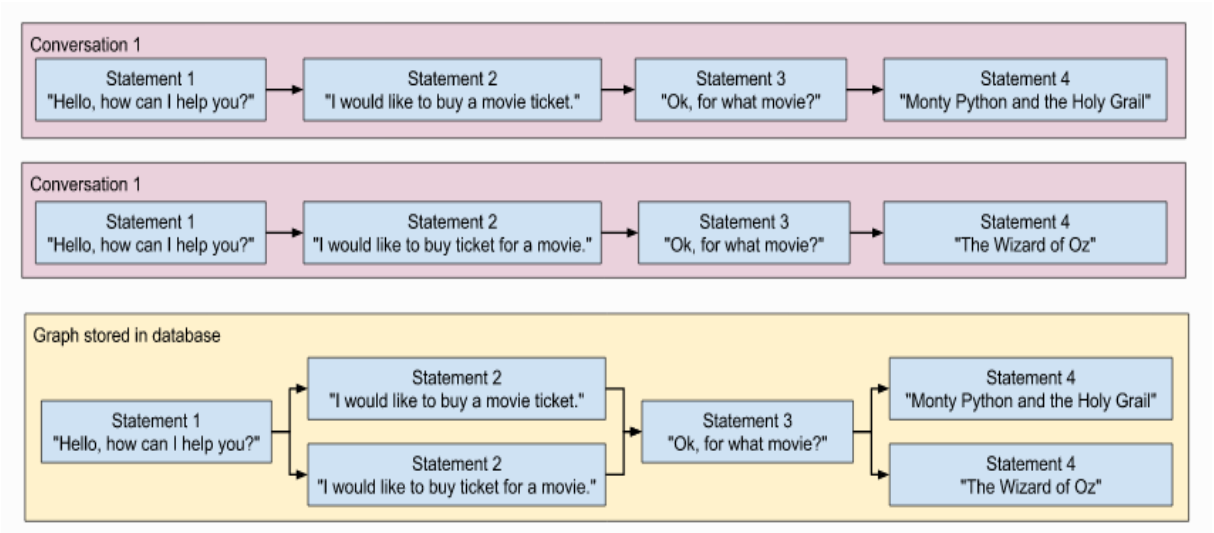
5.3 Obuka agenta za razgovor

Nakon stvaranja nove instance chat bota također je moguće trenirati bota odnosno agenta za razgovor. Obuka nije potrebna ali se preporučuje, ona je dobar način jer omogućuje da bot počne sa znanjem o specifičnim odgovorima.

```
from chatterbot.trainers import ListTrainer

conversation = [
        "Hello, how can I help you.",
        "Hi there.",
        "How are you doing?",
        "Im doing good.",
        "This is good to hear.",
        "Thank you.",
        "You 're welcome."

]

trainer = ListTrainer(chatbot)

trainer.train(conversation)
```

Gore iznad možemo vidjeti česti primjer treniranja chatbota. Trenutna metoda obuke uzima popis izjava odnosno listu izjava koja predstavlja razgovor. Sve to se sprema u varijablu „trainer“ i nakon toga se poziva metoda „trainer.train“.

Na slici ispod možemo vidjeti dva razgovora i njihov tijek, također možemo vidjeti prikaz tih razgovora koji su pohranjeni u bazi podataka [6].



Slika 2: Dijagram obuke agenta

Vrijedno za spomenuti je da ChatterBot također dolazi s korpusnim podacima i pomoćnim modulom koji olakšava brzo osposobljavanje agenta za komunikaciju.

```
from chatterbot import ChatBot
from chatterbot.trainers import ChatterBotCorpusTrainer

trainer = ChatterBotCorpusTrainer(chatbot)

trainer.train(
        "chatterbot.corpus.english"
)
```

Na ovaj način možemo trenirati agenta za razgovor uz pomoć ChatterBot podataka koje smo dobili iz „ChatterBotCorpusTrainer“. Ranije smo gore spomenuli da je ChatterBot jezično neovisan i baš uz pomoć ovog modula možemo obučavati agenta na bilo kojem jeziku, primjerice u ovome primjeru smo uzeli engleski jezik.

5.4 Adapteri za pohranu podataka

ChatterBot dolazi s ugrađenim adapterom koji mu omogućuju povezivanje s različitim vrstama baza podataka. U ovome radu se koristi „SQLStorageAdapter“ koji omogućuje agentu za razgovor povezivanje sa SQL bazama podataka. Prema zadanim postavkama ovaj adapter će stvoriti SQLite bazu podataka. Parametar baze podataka koristi se za navođenje putanje do baze podataka. Na primjeru ispod bazu ćemo nazvati „sqlite:///database.sqlite3“, ova datoteka će se automatski stvoriti ako već ne postoji. Također „SQLStorageAdapter“ je zadani adapter ChatterBot-a i ako ne navedemo adapter u svom konstruktoru, on će se automatski koristiti.

```
bot = ChatBot(
    'Example bot',
    storage_adapter='chatterbot.storage.SQLStorageAdapter',
    database_uri='sqlite:///database.sqlite3'
)
```

5.5 Logički adapteri

Logički adapteri određuju logiku načina na koji ChatterBot odabire odgovor na zadani ulazni izraz. Logički adapteri mogu se odrediti postavljanjem parametra *logic_adapters* gdje se nalazi popis logičkih adaptera. Moguće je unijeti bilo koji broj logičkih adaptera. Primjerice ako se koristi više adaptera, tada će bot vratiti odgovor s najvećom izračunatom vrijednošću pouzdanosti, ako više adaptera vrati isto povjerenje tada će prioritet imati onaj adapter koji se nalazi na prvi na popisu. U ovome primjeru koristit ćemo tri logička adaptera. „BestMatch“ odabire odgovor na temelju najbolje poznatog podudaranja s danim unosom, „MathematicalEvaluation“ rješava matematičke zadatke koji koriste osnovne operacije, „TimeLogicAdapter“ vraća trenutno vrijeme kada to traži ulazna naredba [6].

```
bot = ChatBot('Example bot',
     storage_adapter = 'chatterbot.storage.SQLStorageAdapter',
     logic_adapters = [
          'chatterbot.logic.BestMatch',
          'chatterbot.logic.MathematicalEvaluation',
          'chatterbot.logic.TimeLogicAdapter'
],
database_uri='sqlite:///database.sqlite3'
)
```

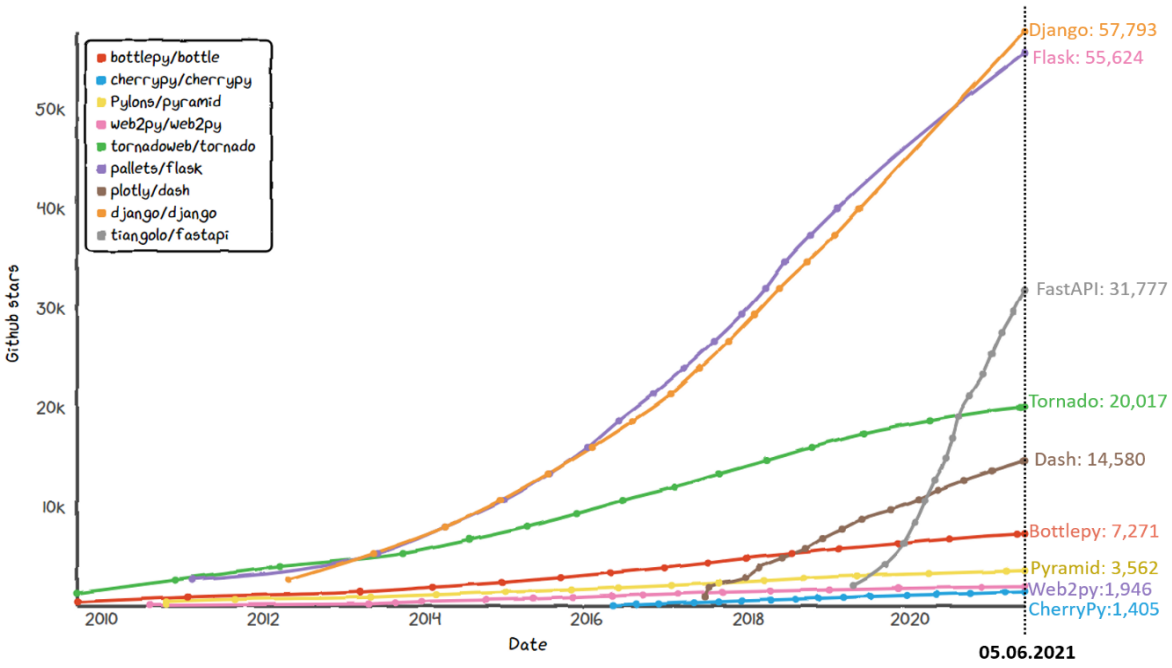
5.6 Pretprocesori

ChatterBot -ovi pretprocesori jednostavne su funkcije koje mijenjaju ulazni izraz koji agenta za razgovor prima prije nego što naredbu obradi logički adapter. Na primjeru ispod može se vidjeti pravilno postavljanje pretprocesora.

```
chatbot = ChatBot(
    'Example bot',
    preprocessors=[
        'chatterbot.preprocessors.clean_whitespace'
    ]
)
```

6. Web okviri za python

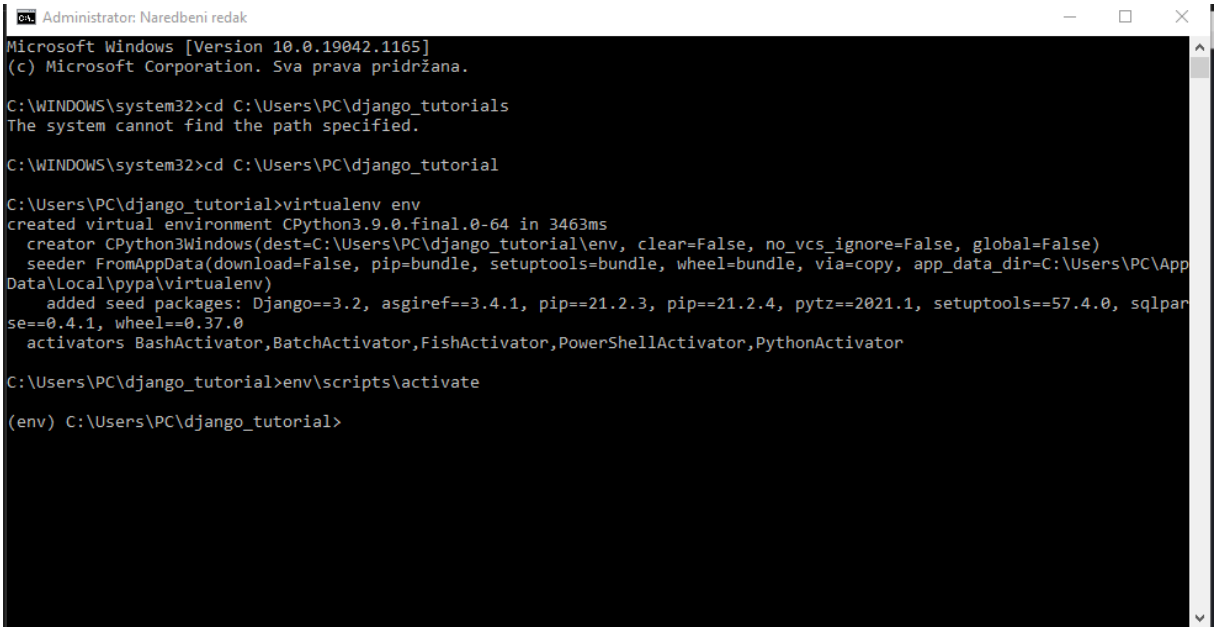
Web okvir je kolekcija ili skup paketa odnosno modula koji korisnicima omogućuju pisanje web aplikacija ili usluga. Općenito, okviri pružaju podršku za brojne aktivnosti, kao što su tumačenje zahtjeva, stvaranje odgovora (prezentiranje podataka u HTML-u), trajno pohranjivanje podataka itd. U Pythonu postoji više web okvira koji su na visokoj razini, u ovome radu najviše će se spominjati Django. Na slici ispod možete vidjeti najpopularnije web okvire na temelju GitHuba u programskome jeziku python. Možemo uočiti da je Django na prvome mjestu s 57793 zvjezdice, drugo mjesto je Flask s 55624 zvjezdice i treće mjesto je novi web okvir FastAPI s 31777 zvjezdica [7].



Slika 3: Grafički prikaz Python web okvira

7. Django ChatterBot

Kao što smo ranije spomenuli u metodama i tehnikama rada Django je Python web okvir visoke razine koji omogućuje brzi razvoj sigurnih i održivih web stranica. Prije samog početka rada s Djangom preporučuje se implementacija virtualnog okruženja. Za početak je potrebno u naredbenom retku otići na lokaciju na kojoj želimo napraviti projekt i upisati „virtualenv env“ i nakon toga će se kreirati virtualno okruženje, komandom „env\scripts\activate“ smo aktivirali naše virtualno okruženje, to se može vidjeti na slici ispod gdje s lijeve strane piše (env).

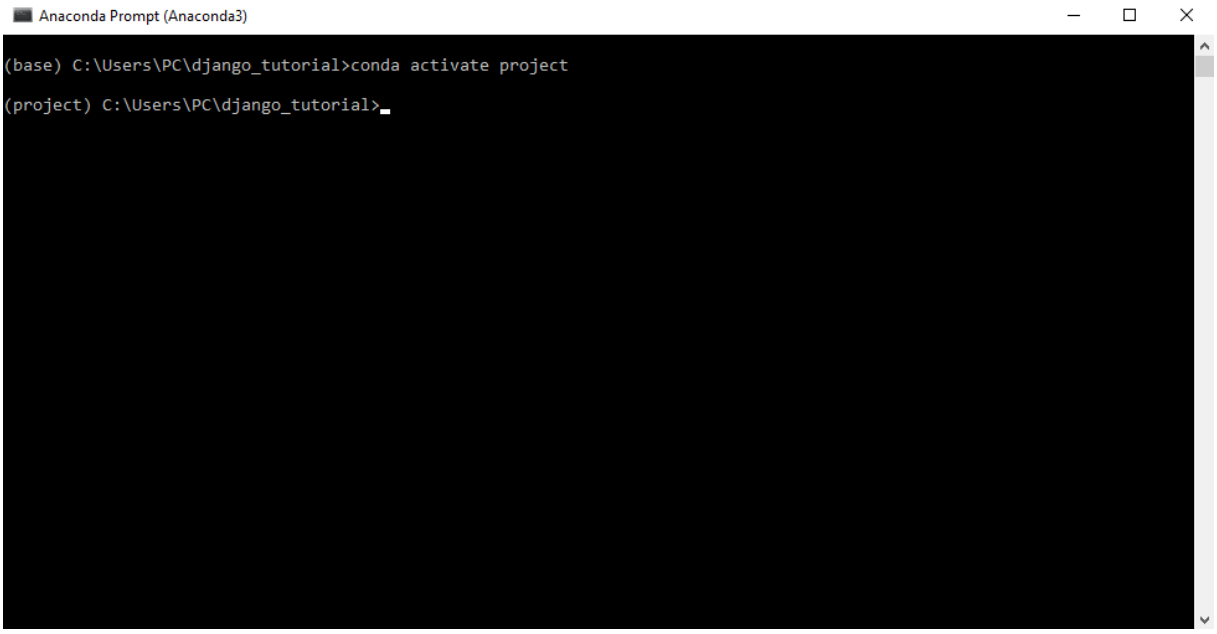


Slika 4: Aktivacija virtualnog okruženja

Drugi način za implementaciju virtualnog okruženja je moguć uz pomoć programa Anaconda Navigator (Anaconda 3) koji prakticira veliki broj python programera. Nakon instalacije programa potrebno je otvoriti „Anaconda prompt“ i isto tako otići na lokaciju na kojoj želimo napraviti projekt i instalirati virtualno okruženje po imenu project.

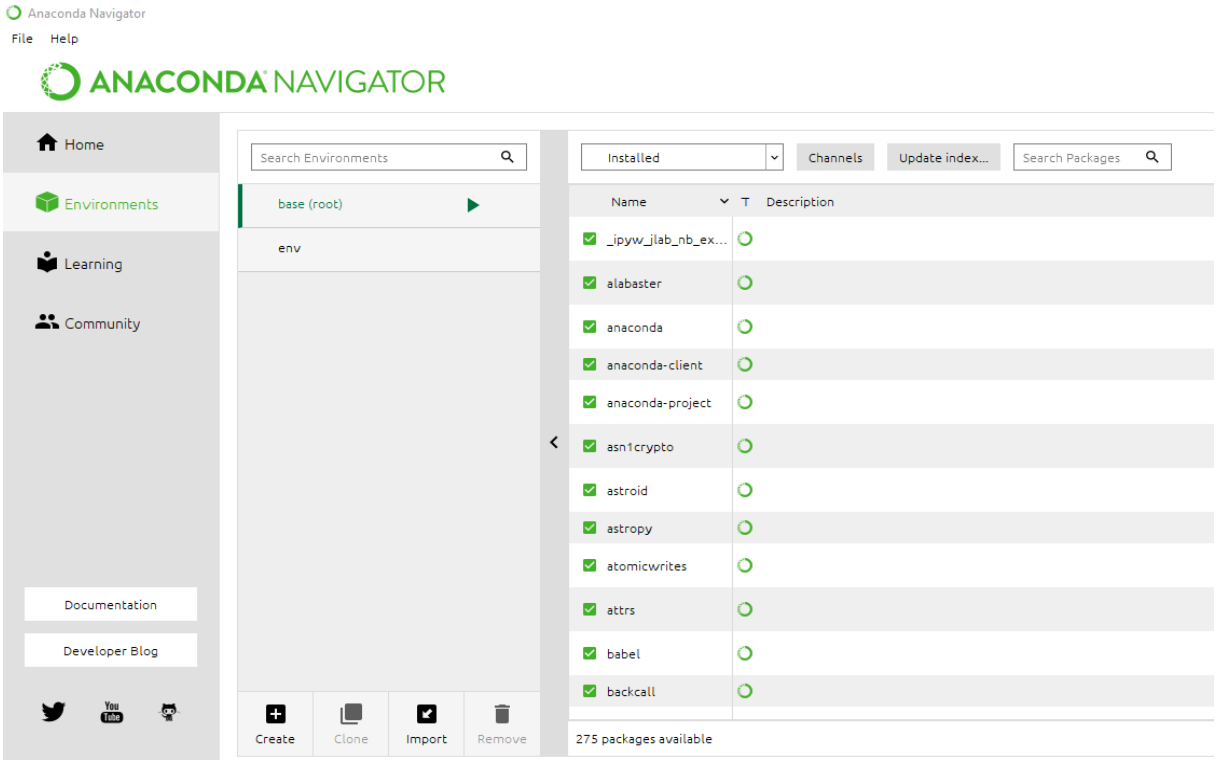
```
conda create -n project python=3.6 anaconda
```

Nakon toga potrebno je unutar naredbenog retka aktivirati novo okruženje.



Slika 5: Anaconda command prompt

Indikator „(project)“ nam pokazuje da smo uspješno aktivirali naše virtualno okruženje i da možemo početi s instaliranjem Djanga i ostalih Python modula. Velika prednost programa Anaconda Navigator programa jeste što možemo vizualno vidjeti sve module koje smo instalirali kao što su chatterbot, spacy, requests itd [8].

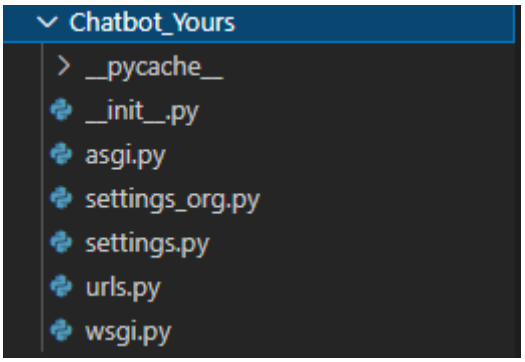


Slika 6: Anaconda Navigator

Nakon što smo namjestili virtualno okruženje, možemo krenuti s instaliranjem Djanga, Django ChatterBota i pokretanjem projekta.

```
pip install Django
pip install django chatterbot
django-admin startproject Chatbot_Yours
```

Nakon pokretanja, naredba „startproject“ je kreirala direktorij projekta „Chatbot_Yours“ koji sadrži sljedeće datoteke.

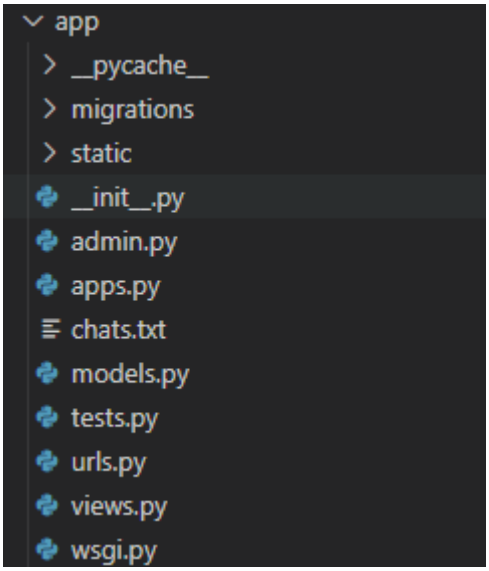


Slika 7: Chatbot_Yours

Sada je potrebno kreirati folder „app“ koji će kreirati direktorij koji će biti na razini aplikacije.

```
python manage.py startapp app
```

Na sljedećoj slici prikazat će se folder „app“ i sve njegove datoteke.



Slika 8: Aplikacija

Na sljedećim slikama možemo vidjeti detaljnije datoteku „settings.py“ koja je automatski generirana prilikom pokretanja projekta.

```python
# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
import os

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/1.8/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'fsch+6!=q+@ol&%0x!nwdl@48^ixbd4clx5f1i!5n^66y+pmn*'


DEBUG = True

ALLOWED_HOSTS = ['localhost', '127.0.0.1', '[::1]']
#'localhost', '127.0.0.1', '[::1]'

# Application definition
INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    'chatterbot.ext.django_chatterbot',
    'app',
)
```

Slika 9: Django settings 1.dio

Aplikacije u Djangu uključuju neku kombinaciju modela, prikaza, predložaka, oznaka predložaka, statičkih datoteka, URL-ova itd. One su općenito povezane u projekte s postavkom „INSTALLED_APPS“.
Zato u „INSTALLED_APPS“ smo dodali „chatterbot.ext.django_chatterbot“ i ime aplikacije „app“ zbog implementacije same aplikacije i ChatterBota u Djangu.

Na sljedećoj slici možemo vidjeti drugi dio datoteke „settings.py“ u kojoj smo dodali ChatterBot postavke odnosno kreirali našeg bota. U sekciji „TEMPLATES“ pod „DIRS“ je dodana putanja gdje se nalazi folder templates, to je učinjeno da bi se prilikom pokretanja servera učitale pripadne HTML datoteke.

```
# ChatterBot settings
CHATTERBOT = {
    'name': 'Django ChatterBot Example',
    'django_app_name': 'django_chatterbot'
}



MIDDLEWARE = (
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    # 'django.contrib.auth.middleware.SessionAuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'django.middleware.security.SecurityMiddleware',
)

ROOT_URLCONF = 'app.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': ['C:/Users/PC/chatbotproject_final/chatbotproject/templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

Slika 10: Django settings 2.dio

Nakon što je završena modifikacija u „settings.py“ potrebno je u „app/views.py“ napraviti funkciju „index“ koja ima parametar „request“u kojoj će se prvo dohvatiti template po imenu „app.html“.

app > views.py > index

```python
from django.shortcuts import render
from django.http import HttpResponse
from django.template import loader
from django.shortcuts import render
from chatterbot import ChatBot
from chatterbot.ext.django_chatterbot import settings
from chatterbot.trainers import ListTrainer
from chatterbot.comparisons import LevenshteinDistance
from chatterbot.trainers import ChatterBotCorpusTrainer


def index(request):
    template = loader.get_template('app.html')
    con = {}

    if request.method == 'POST':

        user_input = request.POST.get('input')
        chat = request.POST.get('content')
        print(user_input)

```

Slika 11: Funkcija index 1.dio

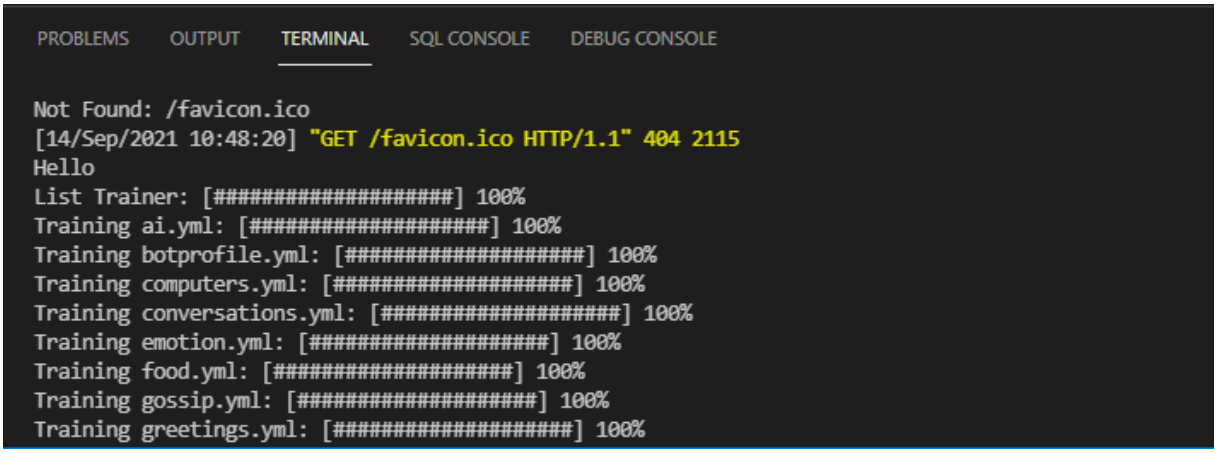
U sljedećem dijelu imamo logičku vrijednost koja je točna kada je trenutni zahtjev korisnika izvršen pomoću „POST“ metode.

Na sljedećoj slici kreirat će se novi bot po imenu „Test“ koji ima logički adapter „chatterbot.logic.BestMatch“ i trenirat će se s određenim podacima iz „chats.txt“ i iz „chatterbot.corpus.english“. Nakon toga ako dobijemo dati odgovor na naše pitanje od bota putem „bot.get_response(user_input)“, to pohranjujemo u varijablu „resp“, ispisujemo i sve to pohranjujemo u con. Na kraju vraćamo „HttpResponse“ koji pruža dolazni HTTP zahtjev Django web aplikaciji s tekstualnim odgovorom.

```python

    bot = ChatBot('Test',logic_adapters=[
        {
            'import_path': 'chatterbot.logic.BestMatch',
            #'threshold': 0.60,
            'maximum_similarity_threshold': 0.90,
            'default_response': "I am sorry, I didn't catched what you meant, but I am constantly learning. Could you please repeat your question in other

        }
],)

    conv = open('app/chats.txt', 'r').readlines()
    #Samo prvi put pokrenuti
    trainer = ListTrainer(bot)
    trainer.train(conv)

    #Samo prvi put pokrenuti
    trainer = ChatterBotCorpusTrainer(bot)
    trainer.train('chatterbot.corpus.english')


    if bot.get_response(user_input):
        resp = bot.get_response(user_input)
        print(resp)
        con = {'resp': resp, 'input': user_input}
    elif not bot.get_response(user_input):
        con = {}
return HttpResponse(template.render(con, request))

```

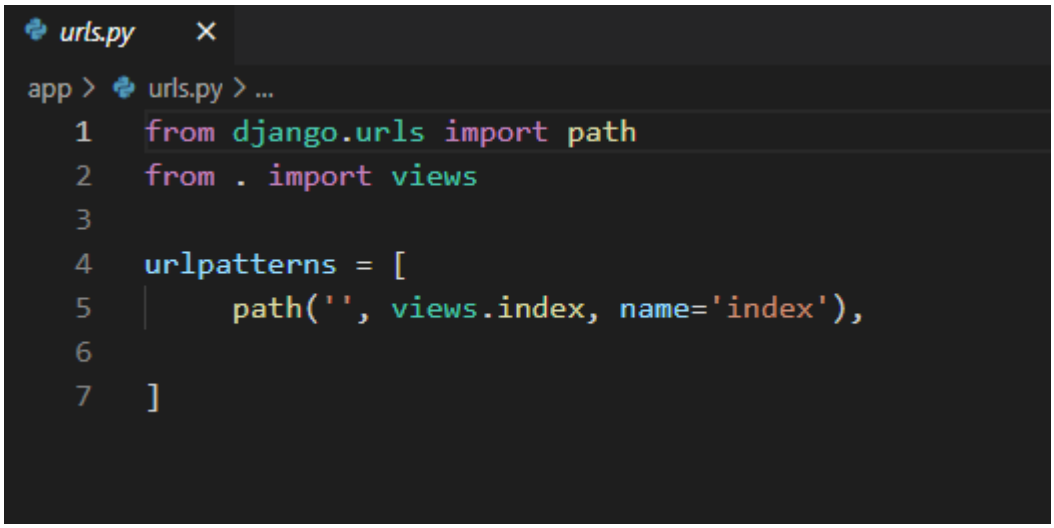
Slika 12: Funkcija index 2.dio

Važno je za napomenuti da treniranje bota je potrebno samo kada se prvi put pokrene server, jer pri pokretanju servera automatski se stvara SQLite baza podataka sa tim podacima, to izgleda ovako.



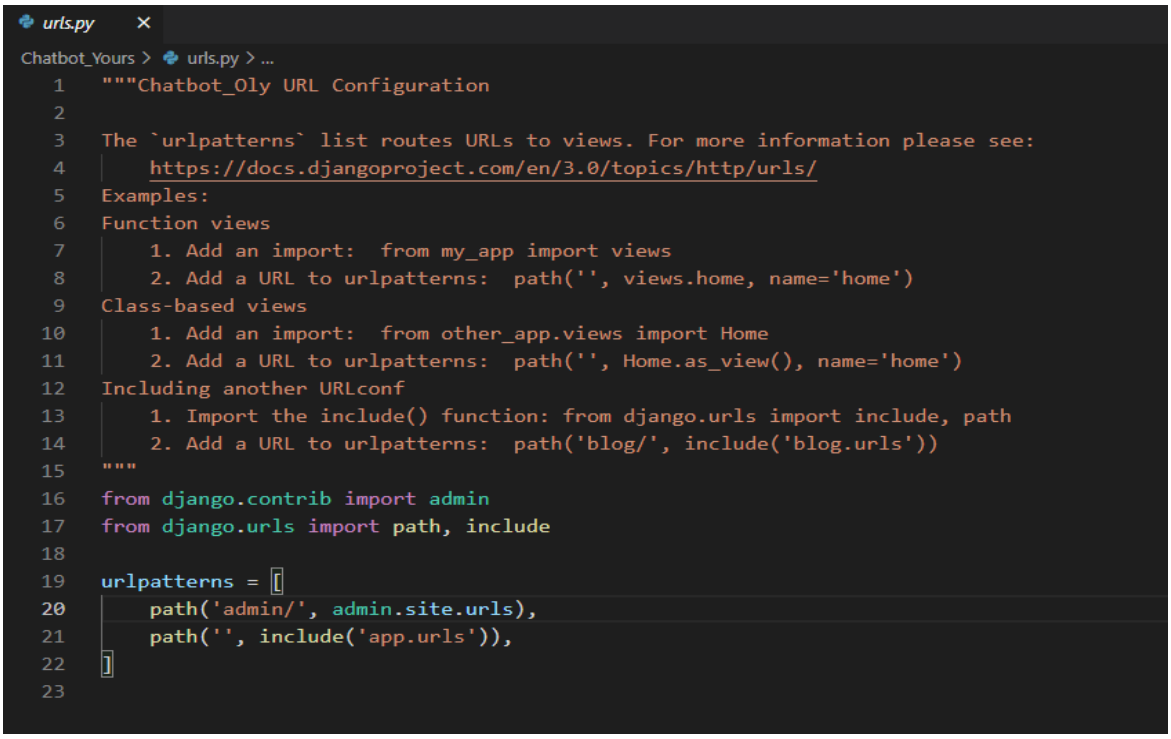
Slika 13: Obuka agenta za razgovor u terminalu

Nakon ispisanog „view.py“ potrebno je dodati „path()“ u „app/urls.py“



Slika 14: app/urls.py

Sada je potrebno „uključiti“ „app.urls“ na projektnoj razini, isto uz pomoć naredbe „path()“



Slika 15: Chatbot_Yours/urls.py

Na slici ispod imamo html datoteku koja se nalazi unutar foldera templates. U datoteci generiramo apsolutni URL statičkih datoteka.U zaglavlju je zadan je naslov „Chat bot“ koji će biti prikazan u tabu browsera. Stranica je stilizirana iz podfoldera „style2.css“. U vidljivom dijelu smo učitali „bot.png“ iz podfoldera „images“, također napravili smo naslov po imenu „Chatbot“. Kada napravimo akciju biti ćemo usmjereni na „app/urls.py“ gdje smo u putanju nazvali index. Nakon toga imamo tri „kojtenjera“, jedan za korisnik na kojem se prikazuje njegov unos, drugi za bota na kojemu se prikazuje njegov odgovor i treći na kojemu upisujemo pitanje koje želimo pitati bota i klikćemo na gumb „Send“.

```html
<!DOCTYPE html>
{% load static %}
<html lang="en">
<head>
    <meta http-equiv="Content-Type" content="text/html" charset="utf8">
    <title>Chat bot</title>
    <link rel="stylesheet" type="text/css" href="static/css/style2.css">

</head>


<body>
    <p>
    <img src="static/images/bot.png">
    </p>

    <h1>Chatbot</h1>

    <form method="post" action="{% url 'index' %}">

        <div id="content">
            <p>You: <br><br>{{ input }}</p>
        </div>

        <div id="content">
            <p>Bot:<br><br>{{ resp }}</p>
        </div>

        <div id="message">
            <textarea name="input" rows="2" cols="34" placeholder="Please enter your question here!"></textarea>
            <br>
            <button name="btn" type="submit">Send</button>
        </div>

        </form>

</body>
</html>
```

Slika 16: app.html datoteka

app > static > css > # style2.css > ...

```css

body {
    background: #222121;

}

#content{
    border: 1px solid #283486;
    background: #FFF;
    margin-left: auto;
    margin-right: auto;
    width: 300px;
    height: 300px;
    overflow: auto;
    border-radius: 3px;
    font-family: 'Times New Roman', Times, serif;
    font-size: 15px;
    color: #727796;
}

#content p{
    padding: 1em;
    margin: 1em;
}


h1{
    color: #283486;
    text-align: center;
}

#message{
    margin-top: 10px;
    margin-left: auto;
    margin-right: auto;
    width: 300px;
    color: #000000;
}

```

Slika 17: style2.css 1.dio

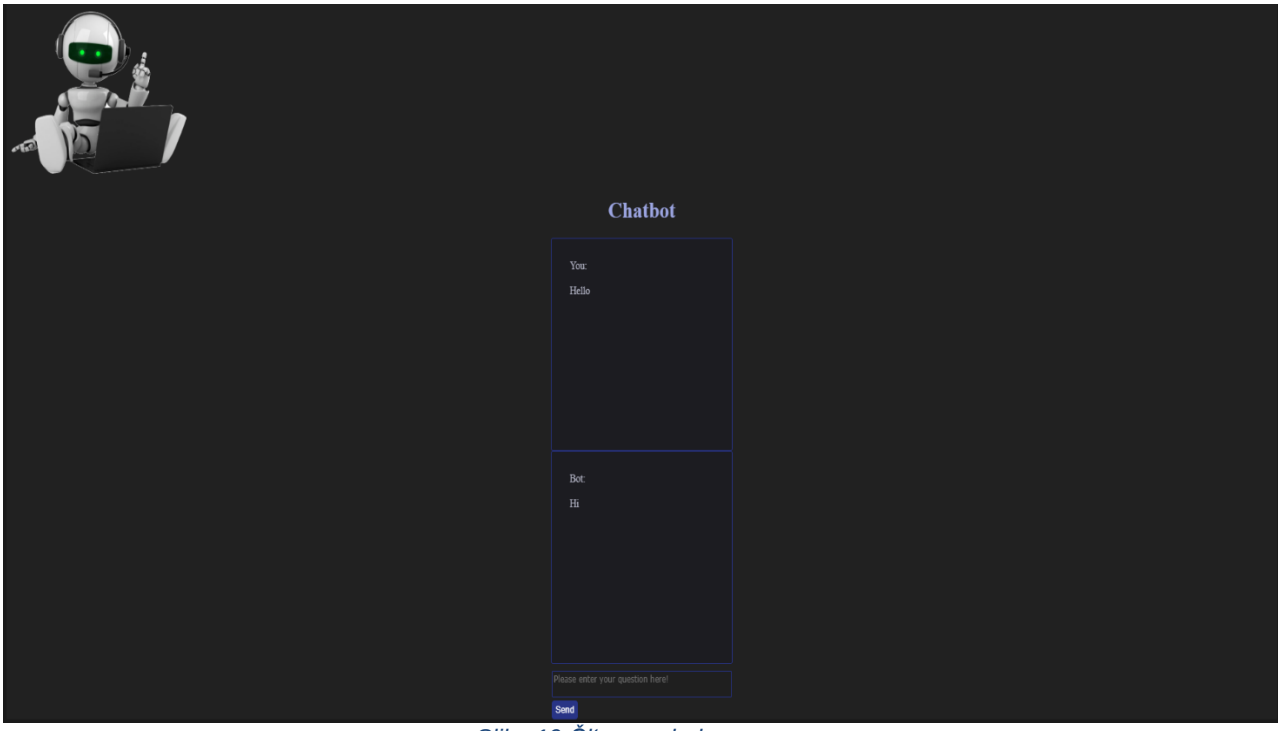
```css
#message textarea{
    border: 1px solid #283486;
    font-family: Tahoma, sans-serif;
    width: 294px;
    resize: none;
}

button{
    border: none;
    padding: 6px;
    text-decoration: none;
    border-radius: 4px;
    background: #283486;
    color: white;
    box-shadow: 0 1px 0 white;
}
```

Slika 18: style2.css 2.dio

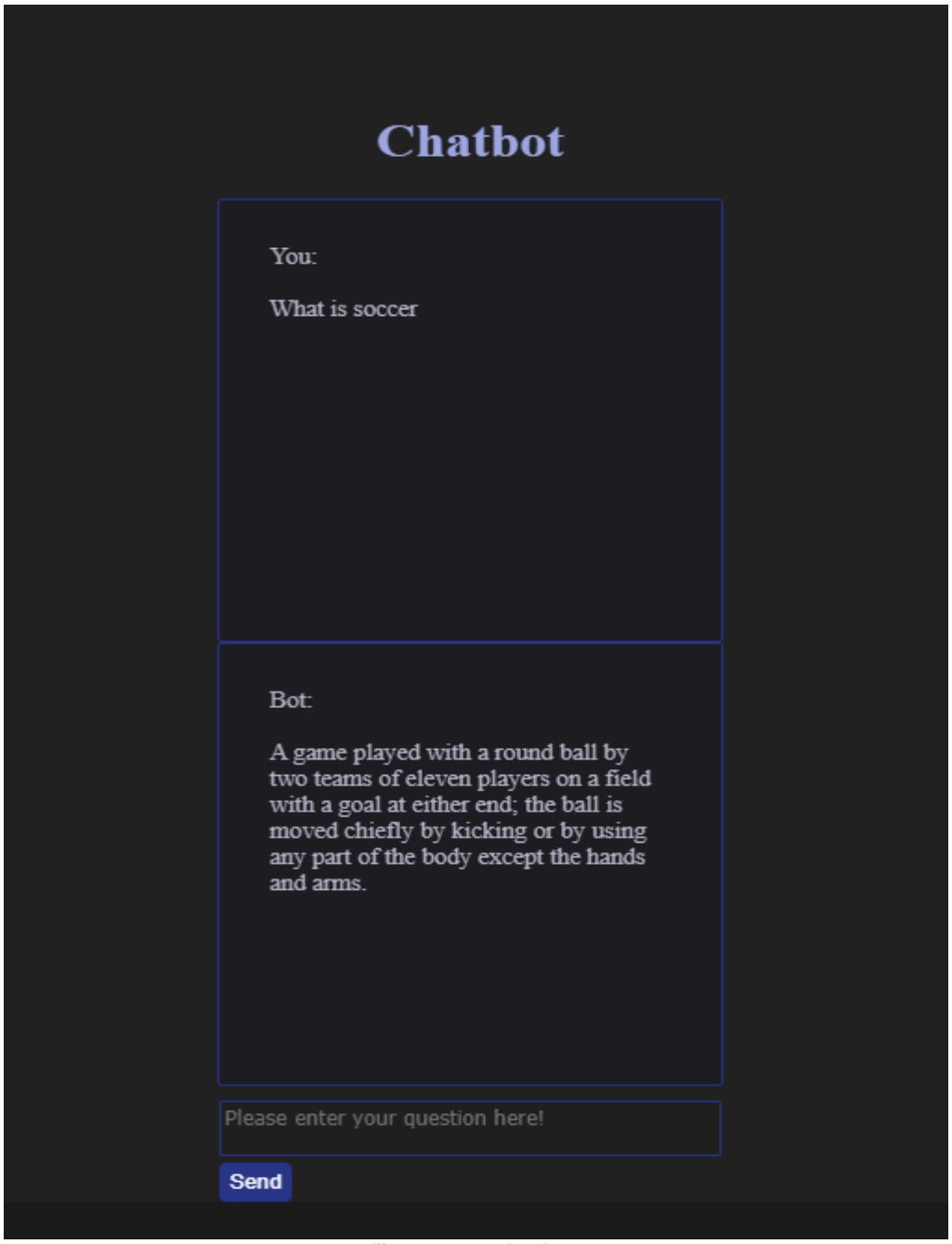
Na gore prikazanoj slici se nalazi detaljan prikaz „style2.css“ datoteke. Uz pomoć ove datoteke stiliziramo našu stranicu, mijenjamo boju pozadine, font, veličinu fonta, širinu i visinu komponenata. Primjerice u „h1“ smo odabrali plavu boju za naš header i centrirali smo naslov koji smo zadali u datoteci „app.html“.

8. Vizualni prikaz projekta

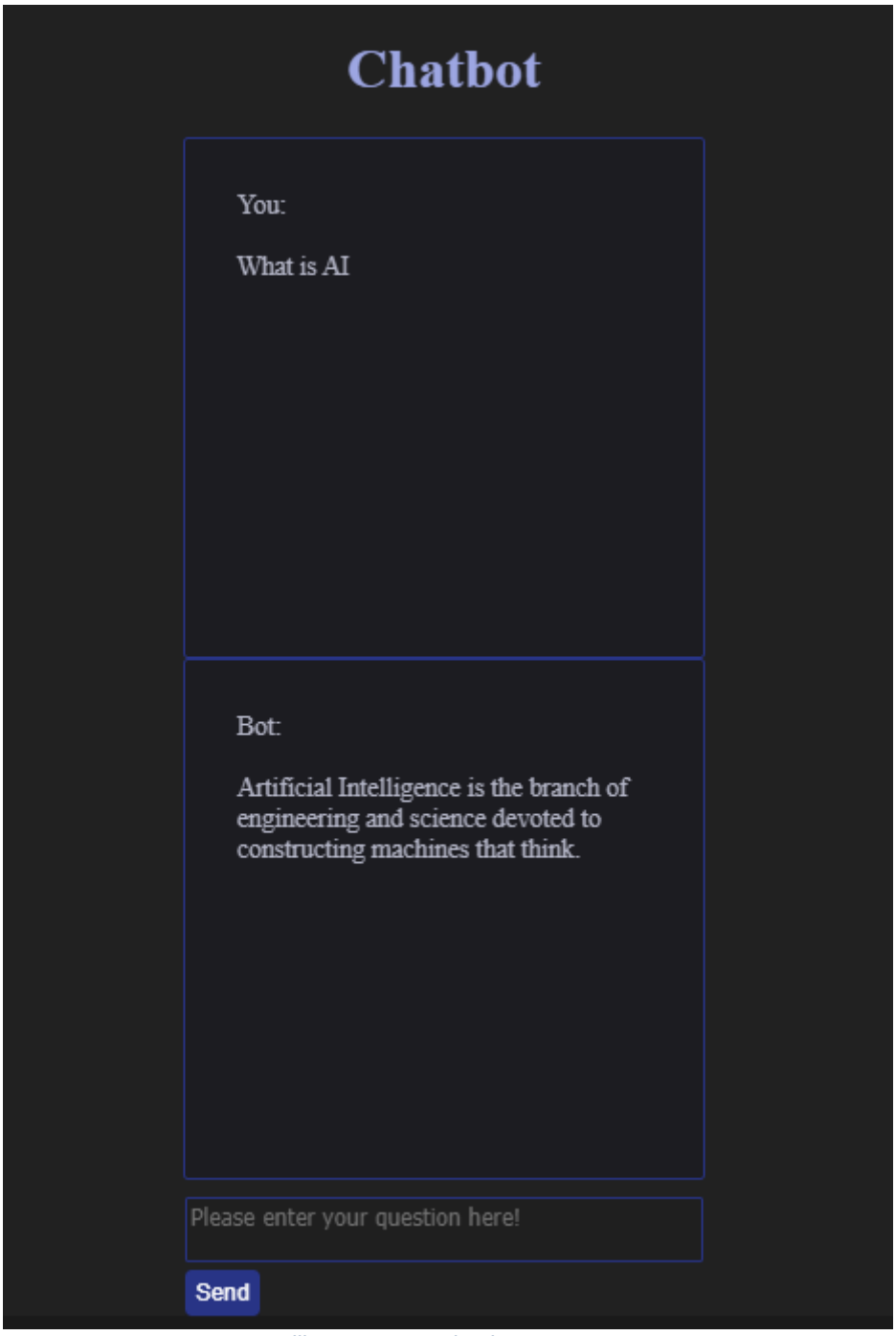
Na slikama ispod se može vidjeti vizualni prikaz agenta za razgovor uz pomoć modula ChatterBot koji je integriran s Djangom.



Slika 19:Čitav pogled



Slika 20:Prvo pitanje



Slika 21: Drugo pitanje

9. Zaključak

Izgradnja inteligentnih agenata za razgovor na raznim platformama postaje sve popularnije nego prije. Činjenica da oko četiri milijarde korisnika ima ovisnost o aplikacijama za razmjenu poruka je dovela do toga da se agenti za razgovor sve više primjenjuju u našoj svakodnevnici. Primjerice, zašto bi trebali skinuti neku aplikaciju za pretraživanje dobrog restorana kad nam bot može pomoći u toj namjeri. Cilj ovog rada bio je vizualno prikazati agenta za razgovor u programskom jeziku Python uz pomoć web okvira Django. Proces obučavanja razvijanja agenta zahtjeva mnogo vremena i pažnje. Važno je napomenuti da je ChatterBot u tom procesu dostao olakšao provedbu raznih dijelova rada.

10. Popis literature

10.1 Literatura korištena za izvor informacija

[1] "HTML" [Na internetu]

Dostupno: https://hr.wikipedia.org/wiki/HTML [Pristupano 28.08.2021.]

[2]

[Pristupano: 29.08.2021.]

[3] "Što je to AI (umjetna inteligencija) i trebamo li je se bojati?" [Na internetu].

Dostupno: https://ec.europa.eu/croatia/basic/what_is_artificial_intelligence_hr

[Pristupano: 02.09.2021.]

[4] "Timeline of artificial intelligence" [Na internetu].

Dostupno: https://en.wikipedia.org/wiki/Timeline_of_artificial_intelligence

[Pristupano 02.09.2021]

[5] "Strojno učenje" [Na internetu].

Dostupno: https://www.fer.unizg.hr/predmet/su [Pristupano 04.09.2021]

[6] "About ChatterBot" [Na internetu].

Dostupno: https://chatterbot.readthedocs.io/en/stable/index.html [Pristupano 08.09.2021]

[7] "Python Web development in 2021: Which web frameworks are the most popular by Github stars?" [Na internetu].

Dostupno:https://gustavwillig.medium.com/python-web-development-in-2021-which-web-frameworks-are-the-most-popular-by-github-stars-e07b1d7ef6f7 [Pristupano 10.09.2021]

[8] "Virtual Environment in Python" [Na internetu].

Dostupno: https://www.datacamp.com/community/tutorials/virtual-environment-in-python

[Pristupano 10.09.2021]

10.2 Literatura korištena u programskom kodu

[9] "Building a chatbot using Chatterbot in Python" [Na internetu].

Dostupno:
https://www.datacamp.com/community/tutorials/building-a-chatbot-using-chatterbot
[Pristupano 09.09.2021]

[10] "ChatterBot“ [Na internetu].

Dostupno: https://github.com/gunthercox/ChatterBot [Pristupano 11.09.2021]

[11] "Your first steps with Django: Set up a Django Project" [Na internetu].

Dostupno: https://realpython.com/django-setup/ [Pristupano 12.09.2021]

[12] "Django documentation" [Na internetu].

Dostupno: https://docs.djangoproject.com/en/3.2/ [Pristupano 13.09.2021]

Popis slika

Prilozi

(views.py, urls.py, wsgi.py, asgi.py, settings.py, app.html, style2.css, manage.py, apps.py)

1. Izvorni kod views.py

```
from django.shortcuts import render
from django.http import HttpResponse
from django.template import loader
from django.shortcuts import render
from chatterbot import ChatBot
from chatterbot.ext.django_chatterbot import settings
from chatterbot.trainers import ListTrainer
from chatterbot.comparisons import LevenshteinDistance
from chatterbot.trainers import ChatterBotCorpusTrainer
from django.views.generic import TemplateView

def index(request):
    template = loader.get_template('app.html')
    con = {}

    if request.method == 'POST':

        user_input = request.POST.get('input')
        chat = request.POST.get('content')
        print(user_input)


        bot = ChatBot('Test',logic_adapters=[
            {
                'import_path': 'chatterbot.logic.BestMatch',
                #'threshold': 0.60,
                'maximum_similarity_threshold': 0.90,
                'default_response': "I am sorry, I didn't catch what you
meant, but I am constantly learning. Could you please repeat your question
in other words?"

            }
    ],)

        conv = open('app/chats.txt', 'r').readlines()
        #Samo prvi put pokrenuti
        #trainer = ListTrainer(bot)
        #trainer.train(conv)

        #Samo prvi put pokrenuti
        #trainer = ChatterBotCorpusTrainer(bot)
        #trainer.train('chatterbot.corpus.english')


        if bot.get_response(user_input):
            resp = bot.get_response(user_input)
            print(resp)
            con = {'resp': resp, 'input': user_input}
        elif not bot.get_response(user_input):
            con = {}
    return HttpResponse(template.render(con, request))
```

2. Izvorni kod urls.py

2.1 App/urls.py

```
from django.urls import path
from . import views

urlpatterns = [
        path('', views.index, name='index'),

]
```

2.2 Chatbot_Yours/urls.py

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
        path('admin/', admin.site.urls),
        path('', include('app.urls')),
]
```

3. Izvorni kod wsgi.py

```
import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "example_app.settings")

application = get_wsgi_application()
```

4. Izvorni kod asgi.py

```
import os

from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'Chatbot_Yours.settings')

application = get_asgi_application()
```

5. Izvorni kod settings.py

```
import os

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))



# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'fsch+6!=q+@ol&%0x!nwdl@48^ixbd4clx5f1i!5n^66y+pmn*'
```

```
DEBUG = True

ALLOWED_HOSTS = ['localhost', '127.0.0.1', '[::1]']
#'localhost', '127.0.0.1', '[::1]'

# Application definition
INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    'chatterbot.ext.django_chatterbot',
    'app',
)


# ChatterBot settings
CHATTERBOT = {
    'name': 'Django ChatterBot Example',
    'django_app_name': 'django_chatterbot'
}




MIDDLEWARE = (
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    # 'django.contrib.auth.middleware.SessionAuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'django.middleware.security.SecurityMiddleware',
)


ROOT_URLCONF = 'app.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS':
['C:/Users/PC/chatbotproject_final/chatbotproject/templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'app.wsgi.application'
```

```
# Database

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}



LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True



STATIC_URL = '/static/'



STATICFILES_DIRS = (
     os.path.join(
         os.path.dirname(__file__),
         'static',
     ),
)
```

6. Izvorni kod app.html

```
<!DOCTYPE html>
{% load static %}
<html lang="en">
<head>
    <meta http-equiv="Content-Type" content="text/html" charset="utf8">
    <title>Chat bot</title>
    <link rel="stylesheet" type="text/css" href="static/css/style2.css">

</head>


<body>
    <p>
    <img src="static/images/bot.png">
    </p>

    <h1>Chatbot</h1>

    <form method="post" action="{% url 'index' %}">

        <div id="content">
```

```
            <p>You: <br><br>{{ input }}</p>
        </div>

         <div id="content">
             <p>Bot:<br><br>{{ resp }}</p>
         </div>

        <div id="message">
            <textarea name="input" rows="2" cols="34" placeholder="Please
enter your question here!"></textarea>
            <br>
            <button name="btn" type="submit">Send</button>
        </div>

        </form>

</body>
</html>
```

7. Izvorni kod style2.css

```
body {
   background: #222121;

}

#content{
   border: 1px solid #283486;
   background: #FFF;
   margin-left: auto;
   margin-right: auto;
   width: 300px;
   height: 300px;
   overflow: auto;
   border-radius: 3px;
   font-family: 'Times New Roman', Times, serif;
   font-size: 15px;
   color: #727796;
}

#content p{
   padding: 1em;
   margin: 1em;
}


h1{
   color: #283486;
   text-align: center;
}

#message{
   margin-top: 10px;
   margin-left: auto;
   margin-right: auto;
   width: 300px;
   color: #000000;
}
```

```
   #message textarea{
      border: 1px solid #283486;
      font-family: Tahoma, sans-serif;
      width: 294px;
      resize: none;
   }

   button{
      border: none;
      padding: 6px;
      text-decoration: none;
      border-radius: 4px;
      background: #283486;
      color: white;
      box-shadow: 0 1px 0 white;
   }
```

8. Izvorni kod manage.py

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'Chatbot_Yours.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()
```

9. Izvorni kod apps.py

```
from django.apps import AppConfig


class AppConfig(AppConfig):
    name = 'app'
```