

UNIVERZITET U BEOGRADU
ELEKTROTEHNIČKI FAKULTET

Bogdan Marković, 2020/0323 i Mirko Krstić, 2020/0194

Preporučivač stručne literature *bookMentor* -
django web aplikacija realizovana korišćenjem
algoritama mašinskog učenja
projekat iz predmeta Principi modernih telekomunikacija

mentor:
prof. dr Milan Bjelica

Beograd, jul 2023.

Sažetak

U vremenu kada je znanje opšte dostupno, a opet toliko ga je da se ne može na efikasan način selektovati, preporučivači pronalaze svoje bitno mesto kao alat u svakodnevnom životu. Preporučivače nesvesno svaki dan koristimo u raznim sferama redovnog života. ***bookMentor*** je napravljen upravo sa ciljem da na efikasan način, poznajući svoje korisnike - studente, i njihove krugove interesovanja ali i dosadašnje iskustvo, preporuči odgovarajuću literaturu kojom bi studenti mogli brže da napreduju na svom profesionalnom putu.

Ključne reči: stručne knjige, preporučivač, web aplikacija, mašinsko učenje, python, django, scikit-learn

Sadržaj

1	Uvod	5
1.1	Cilj projekta	5
1.2	Pozadina projekta	5
1.3	Pregled arhitekture sistema	6
2	Početak rada	7
2.1	Instalacija i podešavanje okruženja	7
2.2	Podešavanje baze podataka	7
3	Korisničko uputstvo	8
3.1	Registracija korisnika	8
3.2	Korisnički profil	9
3.2.1	Pretraga, detalji i ocenjivanje knjiga	10
3.2.2	Prikaz i ocene kurseva	11
3.2.3	Preporučene knjige	11
3.3	Uputstvo za admina	12
4	Sistem preporučivanja i infrastruktura aplikacije	13
4.1	Pregled algoritama preporučivanja - Collaborative filtering + Content-Based	13
4.2	Dobavljanje i skladištenje podataka	14
4.2.1	Dobavljanje knjiga korišćenjem O'Reilly Platform Search API-ja	15
4.2.2	Dobavljanje kurseva Elektrotehničkog fakulteta	15
4.2.3	PostgreSQL Database Management System i punjenje baze podataka	16
4.3	Model preporučivanja	21
4.4	Integracija sistema sa django radnim okvirom - Backend web aplikacije	23
4.4.1	Zašto django?	23
4.4.2	Implementacija backend-a	25

4.5	Frontend web aplikacije	29
5	Budućnost projekta	30
5.1	Buduće funkcionalnosti i poboljšanja	30
6	Zaključak	31
	Bibliografija	32

Spisak slika

3.1	<i>Stranica za registrovanje novog korisnika</i>	8
3.2	<i>Stranica za editovanje privatnih podataka</i>	9
3.3	<i>Dashboard stranica ulogovanog korisnika</i>	9
3.4	<i>Početna stranica - Listing svih knjiga</i>	10
3.5	<i>Detaljan prikaz jedne od knjiga</i>	10
3.6	<i>Detaljan prikaz jednog od kurseva</i>	11
3.7	<i>Preporučene knjige za ulogovanog korisnika</i>	11
3.8	<i>Adminska stranica za kurseve</i>	12
4.1	<i>Izgled GUI pgAdmin aplikacije</i>	17
4.2	<i>Povratna informacija skripte za punjenje baze</i>	19
4.3	<i>Logo django framework-a</i>	24
4.4	<i>MVT Struktura django framework-a</i>	25

Spisak tabela

4.1	Struktura/Tabela Book za opisivanje svake knjige u sistemu	. 15
4.2	Struktura/Tabela Course za opisivanje svakog kursa u sistemu	16

Glava 1

Uvod

1.1 Cilj projekta

Glavni cilj projekta *bookMentor* je da bude koristan studentima svih smerova na Elektrotehničkom fakultetu u Beogradu. Koristnost leži u efikasnosti same web aplikacije, koja na jednostavan način u skladu sa afinitetima studenta, daje odgovarajuće stručne knjige kao preporuku.

1.2 Pozadina projekta

Iako radi naizgled jednostavan posao, implementacija samog projekta odnosno aplikacije iziskivala je korišćenje i kombinaciju nekoliko različitih oblasti Računarske nauke:

- Data engineering
- Database management
- Web development
- Data science
- Machine learning

U narednim poglavljima je detaljno objašnjeno šta je koja oblast predstavljala u samom projektu, i na koji način je doprinela funkcionalnosti sistema. Kako je projekat u potpunosti softversko rešenje, u prilogu odgovarajućih delova biće sadržane i neke od deklaracija funkcija, tipova i promenljivih. Struktura dokumentacije je takva da prati preporučeni tok dokumentovanja, ali i uvaži umetničku slobodu autora.

1.3 Pregled arhitekture sistema

Arhitektura sistema je jednostavna i predstavlja Web aplikaciju koja se pokreće u pretraživaču nezavisno od operativnog sistema. Platformska nezavisnost aplikaciji omogućava pokretanje i na telefonima, tabletima i ostalim sličnim prenosivim uređajima. Aplikacija je dizajnerski potpuno responzivna.

Glava 2

Početak rada

2.1 Instalacija i podešavanje okruženja

Kako je web aplikacija i dalje u ranoj fazi te nije postavljena na javni server, za pokretanje i editovanje koda potrebno je uraditi nekoliko stvari. Da bi se pokrenuo kod sa mašine, potrebno je instalirati programski jezik Python, framework django, te biblioteke Scikit'learn, Pandas i numpy. Što se verzija tiče, korišćena je verzija Python 3.10.11, koja je kompatibilana sa bilo kojom verzijom 3.x, zatim django 4.2. Pandas 1.5.3, Scikit-learn 1.2.2, numpy 1.24.3. Korišćenjem paket menadžera po volji, napraviti virtuelno okruženje sa navedenim verzijama radi idealnog korišćenja. Celokupan kod je otvoren i dostupan na [ovom github profilu](#). Za uređivanje koda se preporučuju korišćenje IDE-a (Integrated Development Environment, srp. Itegrisano Razvojno Okruženje), sa akcentom na JetBrains PyCharm, odnosno Microsoft Visual Studio Code.

2.2 Podešavanje baze podataka

Aplikacija je projektovana za rad sa PostgreSQL DBMS, stoga je neophodno instalirati paket programa Postgres, dostupan na [sajtu](#), uz napomenu da je korišćena verzija PostgreSQL 15. Paket programa podrazumeva program za rad nad bazom u komandnoj liniji, kao i program sa grafičkim okruženjem.

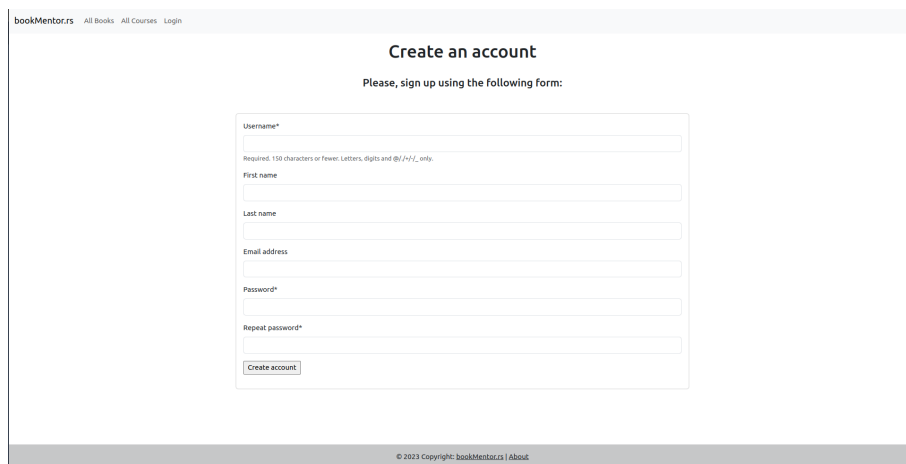
Sama shema baze, ali i realni podaci se ne unose ručno, već se dobijaju na zahtev.

Glava 3

Korisničko uputstvo

3.1 Registracija korisnika

Neregistrovanim korisnicima je dozvoljen pristup svim knjigama i kursevima na pregled. Ipak, radi dobijanja potpune funkcionalnosti sistema, preporučuje im se registrovanje na sistem, koje je poprilično jednostavno i iziskuje unosenje korisničkog imena, imena, prezimena, mejl adrese i šifre. Neophodno je da u sistemu nije zaveden korisnik sa istim korisničkim imenom.



The screenshot shows a web page for creating an account on 'bookMentor.rs'. The page has a light gray header with the site name and navigation links. The main content area is white and contains a centered registration form. The form is titled 'Create an account' and includes a prompt to sign up. It features several input fields: 'Username*' with a character requirement note, 'First name', 'Last name', 'Email address', 'Password*', and 'Repeat password*'. A 'Create account' button is at the bottom of the form. The footer is a dark gray bar with copyright information.

bookMentor.rs All Books All Courses Login

Create an account

Please, sign up using the following form:

Username*

Required: 150 characters or fewer. Letters, digits and @/./+/_ only.

First name

Last name

Email address

Password*

Repeat password*

Create account

© 2023 Copyright: bookMentor.rs | About

Slika 3.1: *Stranica za registrovanje novog korisnika*

3.2 Korisnički profil

Nakon registracije korisnici treba da urede svoje dodatne privatne podatke koji su Usmerenje na Elektrotehničkom fakultetu u Beogradu, i godina studija. Na osnovu ova dva podatka se dobija lista svih kurseva koje je korisnik mogao do tada da sluša. Ukoliko korisnik ne promeni inicijalne podatke, podrazumeva se da je prva godina studijskog programa Elektrotehnika i računarstvo. U prilogu su snimci ekrana na kojima se vidi segmenti o kojima je reč.

The screenshot shows the 'Edit your profile' page. At the top, there is a navigation bar with 'bookMentor.rs', 'All Books', 'All Courses', 'Bogdan's Dashboard', and 'Logout'. The main heading is 'Edit your profile'. Below it is a form with the following fields: 'First name' (filled with 'Bogdan'), 'Last name' (filled with 'Markovic'), 'Email address' (filled with 'bogdan@bookmentor.rs'), 'Smjer*' (a dropdown menu with 'Racunarska tehnika i Informatika' selected), and 'Godina*' (a dropdown menu with 'Trecia godina' selected). There is a 'Save changes' button at the bottom of the form. The footer contains the copyright notice '© 2023 Copyright: bookMentor.rs | About'.

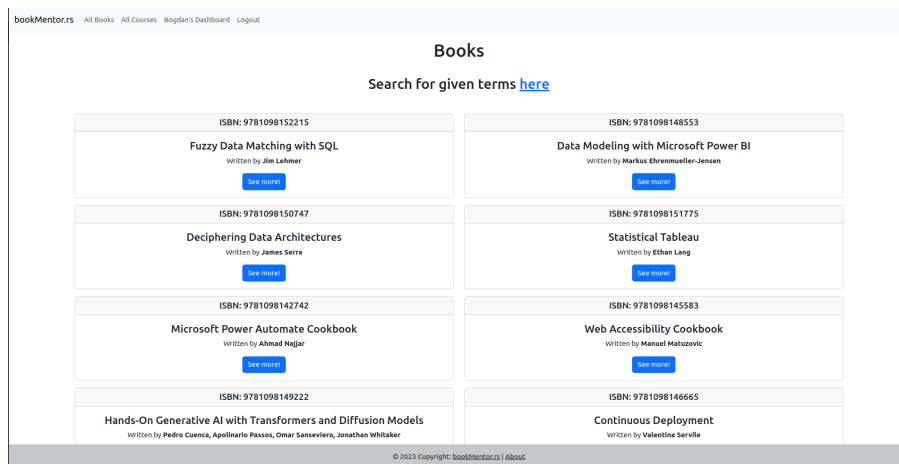
Slika 3.2: Stranica za editovanje privatnih podataka

The screenshot shows the user dashboard. At the top, there is a navigation bar with 'bookMentor.rs', 'All Books', 'All Courses', 'Bogdan's Dashboard', and 'Logout'. The main heading is 'Dashboard'. Below it is a welcome message 'Welcome to your dashboard, Bogdan.' There are four main sections: 'Edit your profile!' with an 'Edit' button, 'Want to see your books?' with a 'See your books' button, 'Courses waiting for your grades...' with a 'View available courses' button, and 'Your graded courses in one place!' with a 'See graded courses' button. At the bottom, there is a message 'Ready for recommendation? [Let's go!](#)'. The footer contains the copyright notice '© 2023 Copyright: bookMentor.rs | About'.

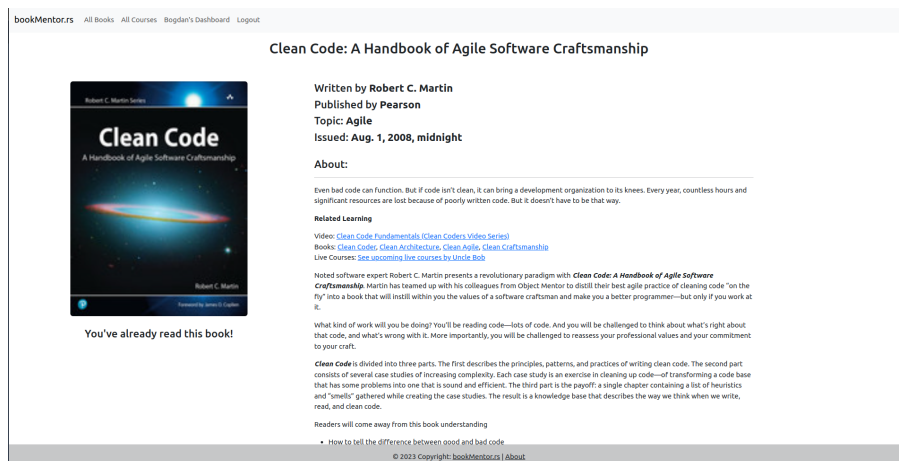
Slika 3.3: Dashboard stranica ulogovanog korisnika

3.2.1 Pretraga, detalji i ocenjivanje knjiga

Pored uređivanja podataka, registrovanom korisniku je omogućeno da pretražuje i da se informiše o knjigama, a zatim ih doda u kolekciju knjiga koje je do tada pročitao. Sistem je jednostavan za korišćenje - knjige koje su već pročitane imaju jasnu naznaku da je tako.



Slika 3.4: Početna stranica - Listing svih knjiga



Slika 3.5: Detaljan prikaz jedne od knjiga

3.2.2 Prikaz i ocene kurseva

Korisnik takode može da ocenjuje kurseve koje je voleo da sluša na skali od 1 do 5. Kursevi koji su već ocenjeni bivaju fizički odvojeni od ostalih kurseva, da ne bi došlo do duplog ocenjivanja.

bookMentor.rs All Books All Courses Bogdan's Dashboard Logout

Konkurentno i distribuirano programiranje

Oblavezan/izborni	Prva godina ER?	Dostupan najranije na SI	Dostupan najranije na IR	Dostupan najranije na OS	Dostupan najranije na OT	Dostupan najranije na OG	Dostupan najranije na OF	Dostupan najranije na OE
Oblavezan	None	3	3	3	Nije dostupan	Nije dostupan	3	Nije dostupan

Topic and coefficient

Java0.3

Parallel Computing0.8

Distributed Systems0.8

Grade(1-5): *

Add course

© 2023 Copyright: bookMentor.rs | About

Slika 3.6: Detaljan prikaz jednog od kurseva

3.2.3 Preporučene knjige

Kada korisnik završi sa unosom preferiranih kurseva i pročitanih knjiga, spreman je za preporuke!

bookMentor.rs All Books All Courses Bogdan's Dashboard Logout

Recommended books for you

ISBN: 9781787125933

Python Machine Learning - Second Edition

Written by Sebastian Raschka, Vahid Mirjalili

See more!

ISBN: 9781784399689

Practical Machine Learning

Written by Sumit Gollapudi

See more!

ISBN: 9780136083238

Clean Code: A Handbook of Agile Software Craftsmanship

Written by Robert C. Martin

See more!

ISBN: 9781119320791

Electronics All-in-One For Dummies, 2nd Edition

Written by Doug Lowe

See more!

ISBN: 9780133755848

Sams Teach Yourself Java™ in 21 Days (Covering Java 8), Seventh Edition

Written by Rogers Cadenhead

See more!

Previous 1 of 3 Next

© 2023 Copyright: bookMentor.rs | About

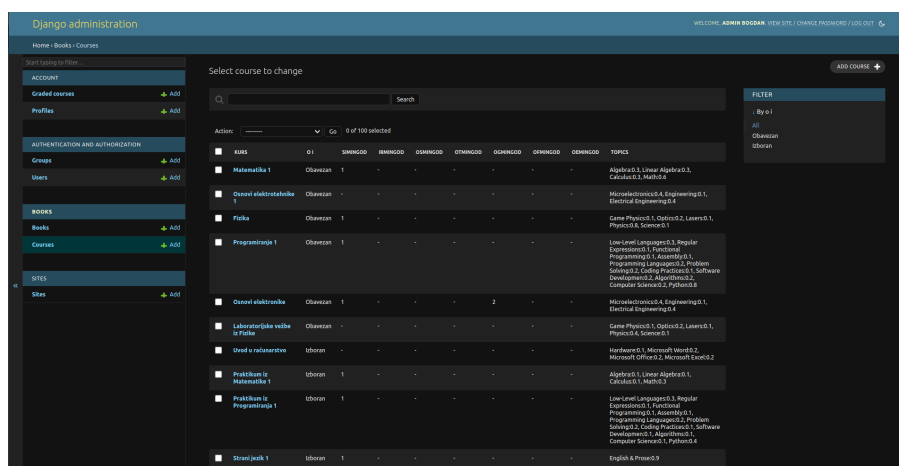
Slika 3.7: Preporučene knjige za ulogovanog korisnika

Autori ohrabruju sve zainteresovane da probaju early phase verziju projekta web aplikacije *bookMentor* i time dodatno unaprede bazu podataka. S povećanjem baze, rezultati preporuke postaju sve kvalitetniji a time i korisničko iskustvo.

3.3 Uputstvo za adminine

Framework django omogućava uređivanje posebne, predefinisane administrativne stranice. Adminska stranica je dozvoljena za pristup samo administratorima odnosno moderatorima aplikacije, jer ova stranica daje pristup bazi podataka na jednostavan a opet pregledan način.

Moderatorima je dozvoljeno da menjanju sve podatke, pregledaju ih u odgovarajućem redosledu i slično. Na slici 3.8 je dat izgled adminske stranice sa svim kursevima u sistemu.



Slika 3.8: Adminska stranica za kurseve

Glava 4

Sistem preporučivanja i infrastruktura aplikacije

U ovom poglavlju se kroz *Project flow* (srp. *Tok izrade projekta*) razmatra proces realizacije projekta u smislu glavnih projektnih odluka, odabranih programskih alata i inženjerskih kompromisa. Uz određene delove je prikazan odgovarajući kod sa deklaracijama klasa i metoda, i komentarima koji predstavljaju pregled višeg nivoa, dok su negde prikazane tabele kao reprezent podataka. Podsećamo da je celokupan kod dostupan na [GitHub profilu](#), i ohrabrujemo njegov pregled. Kako projekat kombinuje različite oblasti Računarske nauke u kojima dominantnu ulogu ima programski jezik **Python**, odlučeno je da upravo on bude korišćen u najvećem delu projekta, i da fokus bude na funkcionalnosti i jednostavnosti.

4.1 Pregled algoritama preporučivanja - Collaborative filtering + Content-Based

Model preporučivanja je fundamentalna komponenta celog projekta, koja će korisnicima preporučiti relevantne knjige na osnovu preferenci i interakcija. Odlučeno je da osnova bude *Collaborative filtering* algoritam, tehnika kod koje je ključno naći slične korisnike (u našem slučaju) ili *iteme*, odnosno stvari koje se preporučuju. U našem sistemu koristimo meru *kosinusne sličnosti* (eng. *cosine similarity*) radi utvrđivanja sličnosti vektora koji opoređuje korisnika sa ostalima. Radi rešavanja problema *hladnog starta* (eng. *cold start*), i obogaćivanja preporuka, u kombinaciju su dodate *preporuke po sadržaju* (eng. *Content-Based recommendation*).

Šta je ono što se preporučuje - knjige stručne literature, koje obrađuju neku oblast. Kome se preporučuje - studentima, od kojih svako voli specifične

oblasti. Kako znamo koje oblasti su studentu zanimljive - tako što su mu interesantniji neki kursevi koje sluša na fakultetu, na kojima se obrađuju baš te teme. Kako znamo da su studenti slični - imaju zajedničke teme koje vole. Tako izgleda ideja povezivanja, a implementacija se nalazi u nekom od sledećih poglavlja.

Ali pre samih preporuka, bilo je potrebno obezbediti podatke, transformisati ih u želenji oblik i iskoristiti korisne attribute... Sve ovo predstavlja posao iz domena *data engineering-a* i *data science-a*, a u narednim sekcijama je opisano rešenje baš za taj deo.

4.2 Dobavljanje i skladištenje podataka

Osnovni problem i prva velika prepreka nakon izbora samog projekta je bio problem dobavljanja i skladištenja podataka. Kako je projekat osmišljen za korišćenje od strane studenata Elektrotehničkog fakulteta u Beogradu a sa ciljem preporučivanja stručnih knjiga, nametnula su se dva ključna pitanja:

1. Kako dobiti knjige i na koji način ih čuvati
2. Kako prikazati sve dostupne kurseve sa Elektrotehničkog fakulteta u Beogradu i u kojoj formi

Kada se prave softverske infrastrukture koje iziskuju korišćenje velikih kolekcija podataka, jedan način za njihovo skladištenje je **interni popis** (ukoliko su stavke iz kolekcije ručno dostupne realizatoru projekta), ili **korišćenje javnih API (*Application Programming Interface*)**, koji na HTTP zahtev (eng. HTTP request) dostavlja podatke u JSON formatu (ovaj način se koristi kada stavke iz kolekcije nisu dostupne ručno, ili ih uopšte ne poseduje realizator projekta). Iz fizički očiglednih razloga, drugi pristup je izabran za korišćenje kod knjiga. Na tržištu je bilo dosta dostupnih javnih API-ja za dobavljanje podataka o knjigama, među kojima su se isticali Amazon Books, Packt Publishing i O'Reilly. S obzirom na to da je za neke od pomenutih API-ja bilo neophodno dobiti virtualni token za korišćenje, odnosno metod autentikacije radi unapređenja brzine slanja i primanja zahteva HTTP protokolom, nakon svih razmatranja odlučeno je da se koristi **O'Reilly Platform Search API**.

4.2.1 Dobavljanje knjiga korišćenjem O'Reilly Platform Search API-ja

Ovaj API jednostavnim HTTP zahtevom sa odgovarajućim parametrima dobavlja podatke u *JSON (JavaScript Object Notation)* formatu. Izabran je zbog obimnosti kolekcije stručne literature koju O'Reilly poseduje, a koja je *publisher-independent*, odnosno kolekcija poseduje knjige različitih izdavača. Ovo je posebno bilo od interesa za sam projekat jer rad sa većom kolekcijom podataka implicira kvalitetnije rezultate u samoj preporuci. Parametri korišćeni za dobijanje dostupnih knjiga su ujedno i podaci koji su bili neophodni za opisivanje svake knjige pojedinačno. Upravo ovi parametri će biti kolone u tabelama buduće Baze podataka, u kojoj će biti smešteno sve vezano za celu infrastrukturu projekta. U tabeli 4.1 je dostupan izgled strukture Book, čija svaka instanca svoje podatke dobija preko API-ja:

Tabela 4.1: Struktura/Tabela Book za opisivanje svake knjige u sistemu

Atribut	Tip podatka	Opis
isbn	Big Int	Identifikacija tabele
issued	DateTime	Trenutak izlaska knjige
authors	Text[]	Lista autora knjige
publishers	Text[]	Lista izdavača knjige
title	Text	Naslov knjige
description	Text	Opis knjige u <html> formatu
average rating	Int	Prosečna ocena na O'Reilly platformi
popularity	Int	Ponderisana vrednost popularnosti knjige
cover url	Text	Hiper veza ka slici korice knjige
topic	Text	Oblast pokrivena knjigom

4.2.2 Dobavljanje kurseva Elektrotehničkog fakulteta

Za dobavljanje kurseva fakulteta izabrana je prva metoda predstavljena na početku poglavlja, a koja se odnosi na interni popis dostupnih resursa. Kako su na [sajtu fakulteta](#) dostupni svi kursevi sa godinama i smerovima na kojima se polažu, proces je mogao da se obavi najjednostavnijim skladištenjem u CSV tip datoteke. Prilikom popisa, kursevima je ručno dodata lista oblasti koji pokrivaju, a u skladu sa listom oblasti dostupnih preko O'Reilly Platform Search API-ja. Na taj način je postignuta osnovna povezanost između kurseva i knjiga. U tabeli 4.2 je dostupan izgled strukture Course koja

jednoznačno opisuje strukturu kurseva interno popisanih, a nakon toga uneti ih u Bazu podataka o kojoj će dodatno biti reči.

Tabela 4.2: Struktura/Tabela Course za opisivanje svakog kursa u sistemu

Atribut	Tip podatka	Opis
id	Int	Identifikator kursa
Kurs	Text	Ime kursa
o/i	Char	Obavezan/Izborni predmet
ER	Boolean	Da li je predmet sa prve godine smeru ER
SIMinGod	Int	Najranija godina slusanja na SI
IRMinGod	Int	Najranija godina slusanja na IR
OSMinGod	Int	Najranija godina slusanja na OS
OTMinGod	Int	Najranija godina slusanja na OT
OGMinGod	Int	Najranija godina slusanja na OG
OFMinGod	Int	Najranija godina slusanja na OF
OEMinGod	Int	Najranija godina slusanja na OE
Topics	Text[]	Lista uređenih parova (Oblast:koeficijent)

Navedene strukture prikazane tabelarno reprezentuju tabele u Bazi podataka. Jasno je da se u svetu tehnologije sve odnosi na podatke i manipulaciju podacima, ali za sve to je potrebno da oni budu negde smešteni. Današnje baze podataka se mogu deliti na više načina, među kojima je osnovna podela na *relacione* i *nerelacione* baze podataka. Kako autori imaju više iskustva u radu sa relacionim bazama podataka koje koriste standardni *SQL (Structured Query Language)*, odlučeno je da se koristi jedna od najpoznatijih i najefikasnijih *open-source (srp. otvorenog koda)* baza podataka - **PostgreSQL**.

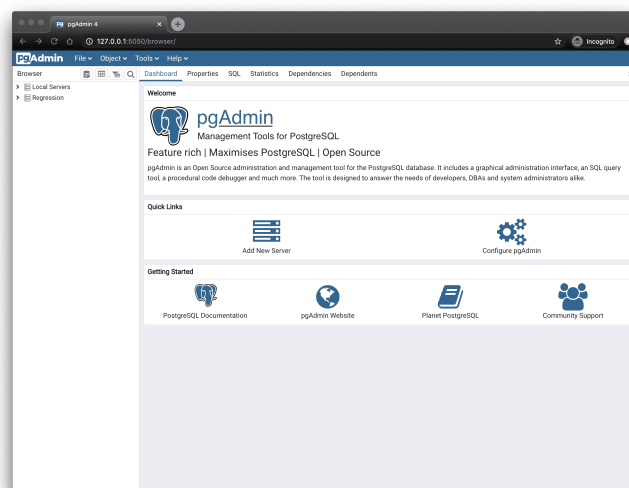
4.2.3 PostgreSQL Database Management System i punjenje baze podataka

PostgreSQL DBMS moderan je sistem za kreiranje i upravljanje relacionim bazama podataka započet na Univerzitetu Berkli 1996. godine. On je **predstavnik softvera otvorenog koda** (eng. open-source software). Implementiran je u niskim jezicima kao što je C/C++ te je vrlo efikasan. Posедуje alate za rad sa različitim tipovima podataka čime ga ističe na tržištu DBMS-ova. Poseduje detaljnu dokumentaciju hijerarhijski struktuiranu radi lakšeg rada. Slogan projekta je *The World's Most Advanced Open Source Relational Database*.

Rad sa PostgreSQL-om (u daljem tekstu PG) moguć je sa nekoliko različitih nivoa pristupa:

- Grafičko korisničko okruženje **pgAdmin** kao najviši pristup
- Pristup iz komandne linije programom **psql**
- Pristup iz programskog koda putem odgovorajucih softverskih **drajvera**

Grafičko okruženje *pgAdmin* korisno je kada je potrebno vršiti ručne promene nad bazom, ili ukoliko je neophodno praviti pojedinačne stavke. Kada je potreban određen nivo automatizacije ili niskog pristupa kako podacima tako i meta podacima (podaci o samoj bazi i tabelama, polise i slično), najbolje je koristiti *psql* iz komandne linije.



Slika 4.1: Izgled GUI *pgAdmin* aplikacije

Treći pristup se odnosi na pristup bazi u programskom kodu višeg reda, kada se koristi tzv DB Driver, odnosno fasada između programera i koda za pristup bazi podataka. Driver omogućava otvaranje konekcije, čitanje podataka, brisanje podataka, menjanje podataka itd. Ovaj pristup je neophodan kada su potrebne sofisticirane operacije nad bazom.

Kao što je već napomenuto, većinski deo projekta je urađen u programskom jeziku Python, te je iz tog razloga korišćen specijalan driver napravljen baš za ovaj jezik - [psycopg2](#). U nastavku je dat pregled funkcija za dovođenje knjiga u bazu, sa kratkim objašnjenjem, kao i deklaracije funkcija za dovođenje kurseva u bazu na osnovu CSV datoteke kurseva:

collectBookData.py

```
import json
import psycopg2
from urllib.request import urlopen
import time

"""
Python script to gather book data from O'Reilly Public Search
↳ API for local database

This data is used for implementation of recommendation system
↳ for electrical engineering students. It is combined
with the manually collected Courses data from faculty's site.
↳ All of data is completely for public use.

"""

def get_connection():
    //Funkcija koja uspostavlja konekciju sa PG bazom

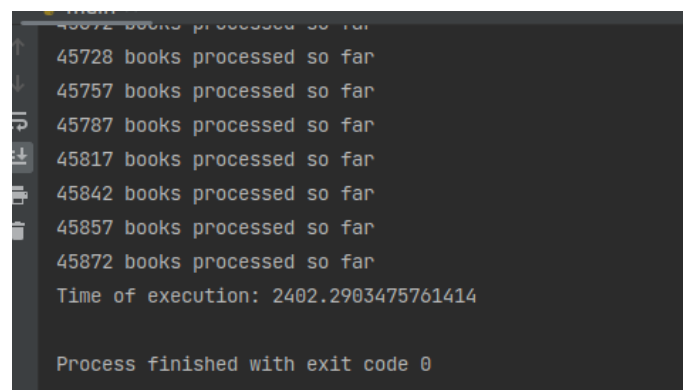
def get_api_name(topic):
    //Funkcija koja obezbeđuje topic name u API friendly
    ↳ formatu

def get_api_url(topic, page):
    //Funkcija koja priprema HTTPS zahtev ka API-ju

def process_book(book):
    //Funkcija koja preuređuje instancu Knjige tako da ona može
    ↳ da se ubaci u bazu podataka. Ovo podrazumeva
    ↳ popunjavanje None polja i rešavanje sličnih
    ↳ nepravilnosti
```

```
def import_books(connection, books_list, page):  
    //Funkcija koja fizički ubacuje knjige u bazu podataka  
  
    //U funkciji main se pokreće petlja u kojoj se generišu HTTPS  
    ↪ API zahtevi za dobijanje knjiga. Knjige se procesuiraju i  
    ↪ zatim ubacuju u bazu.
```

Interesantno je da je za izvršavanje ove iscrpne skripte trebalo čak 2402 sekunde, što je zapravo celih 40 minuta! Na slici 4.2 se može videti povratna informacija ove skripte.



```
45728 books processed so far  
45757 books processed so far  
45787 books processed so far  
45817 books processed so far  
45842 books processed so far  
45857 books processed so far  
45872 books processed so far  
Time of execution: 2402.2903475761414  
  
Process finished with exit code 0
```

Slika 4.2: Povratna informacija skripte za punjenje baze

U narednom listingu 4.2.3 je prikazan kod deklaracija kojima se manualno prikupljeni podaci o Kursovima na fakultetu zapisani u CSV formatu, prebacuju u Bazu podataka.

importCSV.py

```
import psycpg2
import csv
import time

"""
Python script to import all data from Courses csv file. CSV
↪ file was made completely manually, and is available
in github repo.
"""

def getTopics(topicsString):
    //Funkcija koja vraća topics u formatu liste, format
    ↪ pogodan za samu bazu

def getFields(row):
    //Funkcija koja iz reda u CSV datoteci uzima sve relevantne
    ↪ podatke u odgovarajućem obliku

//U funkciji main se prolazi kroz ceo CSV fajl uz pomoć
↪ prethodno objašnjenih funkcija
```

4.3 Model preporučivanja

U ovom trenutku, imamo spremne podatke za naš sistem, i možemo nastaviti dalje sa opisom funkcionalnosti. Bitno je napomenuti da su obezbedjene funkcije koje obavljaju proces dodavanja i ažuriranja koeficijenata za tagove, kroz ocenjivanje kurseva i knjiga. Pre zahteva za preporuke, podrazumeva se da je stanje podataka takvo da su usvojene promene od poslednjeg upisa.

Nakon zahteva ulogovanog korisnika za preporuke, poziva se funkcija *build* koja izvršava ceo proces i poziva odgovarajuće funkcije. Prvo što se radi je normalizacija samih koeficijenata svih korisnika, za svaki tag. Nakon toga, poziva se funkcija kosinusne sličnosti, izdvaja red ulogovanog korisnika, i pamti N id-jeva korisnika sa najvećom sličnošću. N je postavljeno na 30 procenata dužine reda, odnosno pamti se 30 procenata najsličnijih korisnika, što je projektovano za sistem koji ima broj korisnika reda veličine 10^2 . Ukoliko bi sistem imao veći broj korisnika, reda veličine 10^3 , N bi se mogao smanjiti radi efikasnosti.

Zatim se pravi struktura podataka u vidu *pajton rečnika*, koja predstavlja kandidate za preporuke; Iteriramo kroz pročitane knjige sličnih korisnika, kao ključ u rečnik postavljamo objekat knjige, a kao vrednost težinski koeficijent *sim_weight*, koji je postavljen na 1,2. Ukoliko se objekat već nalazi u rečniku, koeficijent se povećava.

Na kraju, poziva se funkcija *description_rec*, koja funkcioniše po sličnom principu. Ovoga puta biramo teme sa najvećom ocenom korisnika, i za svaku od njih vraćamo najpopularnije knjige. Knjige se zatim dodaju u isti rečnik, ali ovoga puta sa manjim težinskim koeficijentom *desc_weight*, koji je postavljen na 1. Na ovaj način smo dali veći prioritet prethodnim preporukama.

Iz rečnika se zatim vade knjige sa najvećim koeficijentom, i prikazuju korisniku kao preporuka, u slučaju da ih već nije pročitao.

U sledećem delu koda napisane su deklaracije funkcija sa komentarima koji se koriste pri prethodno opisanom poslu.

algorithms.py

```
import pandas as pd
import os
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity
from scipy.sparse import coo_matrix
from account.models import Profile
from books.models import Book
```

```

def change_values(tags, coeff):
    //Funkcija koja skalira i vraća koeficijente tagova

def remove_bad(tags, df):
    //Funkcija koja briše tagove koji se ne koriste

def update_csv(csv_path, profile, tags, coeff=5.):
    //Funkcija koja ažurira csv fajl u kome su informacije o
    ↪ koeficijentima

def get_coefficients_from_df(csv_path, profile):
    //Funkcija koja vraća koeficijente iz csv fajla za zadati
    ↪ profil

def normalize(x):
    //Funkcija koja normalizuje vrednosti u nizu

def build(ratings, userID, save=True):
    //Funkcija koja vraća rečnik sa knjigama kandidatima za
    ↪ preporuku

def get_most_popular_books_by_topics(features):
    //Funkcija koja vraća najpopularnije knjige sa zdatim
    ↪ temama

def description_rec(ratings, userID, candidate_books,
    ↪ save=True):
    //Funkcija koja u rečnik sa knjigama kandidatima dodaje
    ↪ knjige po omiljenim temama

```


4.4 Integracija sistema sa django radnim okvirom - Backend web aplikacije

4.4.1 Zašto django?

Nakon obavljenog Data Engineering-a, podaci se nalaze u PG Bazi podataka i spremni su za korišćenje. Naredno pitanje u Project flow-u bilo je kako manipulirati podacima, odnosno koji tip infrastrukture treba koristiti da bi bio što bolji *UX (User Experiance, srp. Korisničko iskustvo)*. Kada se govori o infrastrukturi projekta, govori se o *programu nosiocu*. U ovom slučaju je to aplikacija (Desktop/Web) realizovana u odgovarajućem framework-u. U razmatranju su bile sledeće opcije:

- Native desktop Python GUI pristup (npr. Tkinter)
- MEAN/MERN pristup
- Python based web frameworks

Prva od navedenih varijanti je najteže izvodljiva zbog *niskog nivoa pristupa (eng. low-level)* preko psycopg2 DB Driver-a, ali i činjenice da je u pitanju desktop aplikacija. Iako su desktop aplikacije robusnije, njihova proizvodnja opada već godinama zbog platformske zavisnosti. Upravo platformska zavisnost kao mana implicira platformsku nezavisnost kao vrlinu, što danas imaju apsolutno sve web aplikacije. Iz tog razloga, naredna dva pristupa su bila mnogo interesantnija pri izboru tipa infrastrukture.

MEAN/MERN je skraćenica koja označava pristup Web development-u preko sledećih tehnologija:

1. **MongoDB**, nerelaciona baza podataka
2. **Express.js**, framework za stvaranje RESTful API-ja
3. **Angular / React.js**, frontend web framework
4. **Node.js**, backend web framework

Sa ovim skupom tehnologija je omogućena visoko kvalitetna proizvodnja web aplikacija široke namene. U pitanju su tehnologije široko prihvaćene i poprilično tražene na tržištu Web developmenta. Ipak, jedna stvar se nameće kao problem pri izboru u projektnoj infrastrukturi - glavni jezik koji se koristi u MEAN-u je **JavaScript**, dok je glavni jezik odabran za većinu rešenja u projektu jezik Python. Iako je tehnički izvodljivo koristiti nekoliko

različitih jezika u proizvodnji nekog softverskog rešenja, preporuka je raditi sa onim jezicima i paradigmama koje su lako uparljive kako fizički tako i za programera koji je iza projekta. Iz tog razloga, nametnula se potražnja za **Python based web frameworkom** koji bi obavio isti posao koji bi se odradio korišćenjem MEAN pristupa.

Najpoznatiji takvi framework-ci su ***Flask*** i ***django***. Analiza i biranje "boljeg" između dva frameworka iziskuje veliki napor, jer oba pružaju veliku kontrolu i dosta dobrih mogućnosti. Kombinuju *backend* i *frontend*, odnosno manipulaciju serverskom i klijentskom stranom.

Možda i najvažnija projektna odluka bila je izabrati kvalitetan i odgovarajući framework koji može da podrži sve zahteve navedene u samom početku projektnog teksta. Upravo što je dobar za izradu sofisticiranih web aplikacija, **django** je izabran kao nosilac cele infrastrukture. I to nije slučajno, struktura frameworka je idealna za potrebe projekta upravo zbog skoro idealne povezanosti sa PostgreSQL bazom podataka.



Slika 4.3: *Logo django framework-a*

Primer jedne od viših funkcionalnosti koja je obezbeđena iz ove kvalitetne povezanosti je integracija **PG Text Search-a**, koji predstavlja dubinsku pretragu po nekoliko (specificiranih) polja u bazi, što je u ovom projektu iskorišćeno za detaljnu pretragu knjiga. Ovo je samo jedan od benefita ove veze koji je između ostalih uticao na krajnju odluku.

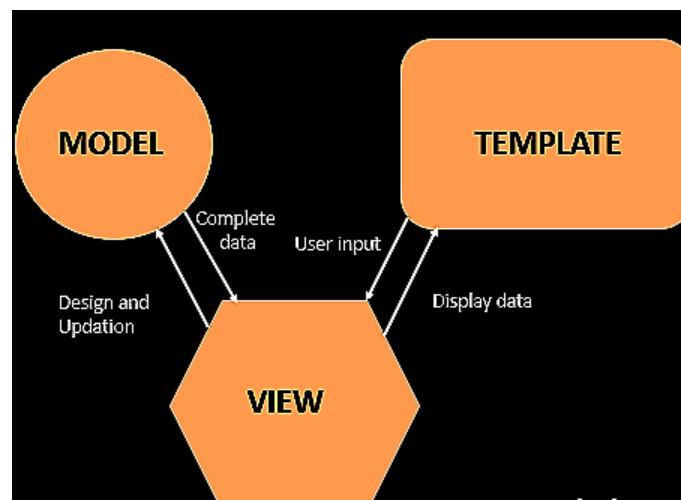
4.4.2 Implementacija backend-a

Kao što je prethodno napomenuto, integrisanost django-a sa PostgreSQL-om stavila ga je na mesto idealnog okvira za rad. Iako složen, ovaj framework ne predstavlja problem da se savlada nakon nekog vremena zbog dokumentacije koja je napisana tako da za svako pitanje postoji odgovor. Zajednica ljudi koji ga koriste je poprilično aktivna čime su dostupni skoro svi odgovori na sva postavljena pitanja.

Struktura framework-a se zasniva na **MVT (Model-View-Template)** pristupu, gde je neophodno spomenuti sledeće stvari:

- **Modeli** -> Klase koje opisuju tabele u bazi podataka na koju je povezan framework
- **Pogledi** -> Interfejsi ka krajnjim korisnicima. Funkcije koje pozivaju odgovarajući HTML sadržaj
- **Templejti** -> Statički HTML+CSS+JS fajlovi koje pretraživač renderuje kada se Pogled izvrši

Način na koji komuniciraju je prikazan na slici:



Slika 4.4: *MVT Struktura django framework-a*

Pored MVT-a, važno je spomenuti da django ima fasadu između stvarnog pristupa bazi podataka (preko psycopg2) i programera koji koristi podatke. Ta fasada se naziva **django ORM (Object Relational Mapping)**, i predstavlja elegantan način da se dobiju podaci, da se njima manipuliše i slično.

Django ORM koristi klase Modele, interfejs Manager i dr. kako bi se podacima pristupalo kao da su najobičniji objekti *OOP (Objektno-orijentisano Programiranje)* paradigme. Na taj način je višestruko ubrzan proces rada, ali i olaksan rad programeru.

Rad u frameworku je maksimalno dekomponovan sa ciljem lakše hijerarhijske organizacije. To znači da svaka zasebna funkcionalnost ide u specijalne podgrupe koje se nazivaju aplikacijama. Može se reći da je framework podeľjen u aplikacije (uz naznaku da su to u stvari podaplikacije) koje međusobno interaguju i koriste podatke. Svaka aplikacija sadrži sledeće fajlove neophodne za rad (ali i još neke koji nisu navedeni):

- Direktorijum **migrations** -> sadrži fajlove koji pamte sve promene nad strukturom modela kako bi u slučaju greške moglo da se pređe na neko prethodno stanje
- **admin.py** -> konfiguracioni fajl u kojem se opisuju modeli u adminskoj stranici
- **forms.py** -> fajl koji sadrži forme koje se koriste u datoj aplikaciji. Forma u ovom slučaju predstavlja klasu (može biti i metoda) koja se translira u HTML formu pri komunikaciji Templejta i Pogleda
- **models.py** -> fajl u kojem se nalaze svi modeli korišćeni za datu aplikaciju. Modeli su klase koje preko django ORM-a mapiraju podatke u bazi podataka
- **urls.py** -> fajl u kojem se nalaze sva mapiranja url adresa za datu aplikaciju
- **views.py** -> fajl u kojem se nalaze svi Pogledi o kojima je bilo reći pri komentarisanju MVT modela koji django koristi

Kako bi se lakše objasnili gorepomenuti pojmovi, najlakše je objasniti ih na primeru u projektu. U okviru projekta su korišćene 2 aplikacije:

- **books** -> sa modelima Book, Course
- **accounts** -> sa modelima Profile, GradedCourse

Neka se posmatra aplikacija *books*. Njena svrha je prikazivanje podataka u obliku pogodnom za korišćenje celokupne infrastrukture. To podrazumeva listing svih knjiga, detaljan pregled pojedinačnih knjiga, listing svih kurseva i detaljan pregled pojedinačnih kurseva. Ukoliko je korisnik autentikovao na

sistemu, omogućeno mu je da knjigu doda u svoju kolekciju knjiga koje je pročitao.

U fajlu **admin.py** su konfigurisani Book i Course modeli kako bi mogli da se u odgovarajućem obliku predstave na adminskoj stranici. Adminska stranica predstavlja built-in funkcionalnost frameworka koji superuser-u (a to je adminski profil) omogućava da na način na koji on to zatraži framework-u (posredstvom admin.py fajla) vidi sve zapise u bazi reprezentovane Modelima.

U fajlu **models.py** su, kao što je već rečeno, svi modeli koji reprezentuju određene tabele iz baze. Konkretno, modeli koji se ovde nalaze su *class Book* i *class Course*. Stvaranje ovih modela je bilo specifično sa tačke gledišta samog framework-a s obzirom na to da ovi modeli nisu prvo napisani u Python-u, već su uvezeni iz same baze. Ova specifična situacija se naziva *Stvaranje modela na osnovu tabela iz legacy baze podataka*, i iziskuje pomoć paketa potprograma django framework-a ali i ručne prepravke. Ova situacija predstavlja tzv. *reverse engineering* (srp. *obrnuti inženjering*) jer se inače tabele stvaraju na osnovu modela a ne obrnuto. Ipak, ovde je situacija bila obrnuta, pa se koristio potprogram **inspectdb**. Naredni listing 4.4.2 pokazuje ključne definicije klasa i metoda koje su korišćene pri ovom zadatku, i njihovo objašnjenje.

```
from django.db import models
from django.urls import reverse

class Book(models.Model):
    //Klasa koja opisuje tabelu Books u bazi

    def get_absolute_url(self):
        //Funkcija vraća punu url adresu ka Knjizi

    class Meta:
        //Podklasa u kojoj se definišu parametri vezani za
        ↪ interakciju modela i samog sistema

    def __str__(self):
        //Funkcija koja vraća String reprezentaciju modela

def get_courses_by_module(module, year, profile):
```

```

        //Vraća kurseve koji se slušaju na zadatom smeru i
        ↪   zadatoj godini

class Course(models.Model):
    //Klasa koja opisuje tabelu Courses u bazi

    def get_topics_dict(self):
        //vraća rečnik sastavljen od topics-a i odgovarajućih
        ↪   koeficijenata

```

U fajlu **urls.py** se nalaze sva mapiranja između Templejta i Pogleda za datu aplikaciju. To znači da mora postojati uređen par (pogled, url) kako bi Templejt znao na čiji HTTP zahtev odgovara. Upravo u fajlu **views.py** se nalaze svi pogledi za datu aplikaciju, pa to u slučaju aplikacije books znači sledeće:

- Pogled za prikaz svih knjiga -> book list
- Pogled za prikaz pojedinačnih knjiga -> book detail
- Pogled za prikaz svih kurseva -> course list
- Pogled za prikaz pojedinačnih kurseva -> course detail

Pogledi koji pokazuju liste koriste tzv. *Paginatore*, koji omogućuju stranični prikaz podataka, pristup mnogo efikasniji i po korisnika i po pretraživač. Pogledi su najobičnije Python funkcije koje obrađuju HTTP zahteve i to dvojako u zavisnosti od toga da li je zahtev *GET* ili *POST*.

Na ovaj način su obrađeni svi scenariji za ovu aplikaciju koji podrazumevaju prikaz podataka. Sličan rezon se koristi za bilo koju drugu aplikaciju u smislu strukture aplikacije - funkcionalnosti se naravno menjaju.

U aplikaciji accounts se pravi podinfrastruktura za sistem registracije i prijave, kao i sve funkcionalnosti koje su planirane da korisnici imaju, a to su:

- Promena ličnih podataka
- Dodavanje knjiga u ličnu kolekciju pročitanih
- Dodavanje i ocenjivanje kurseva dostupnih na osnovu smeru i godine studija korisnika

- **Dobijanje preporučenih knjiga**

Kako je izvorni kod preobiman za projektnu dokumentaciju, ohrabruju se zainteresovani da pogledaju detalje koji su objavljeni na platformi GitHub. Za sve ostale koji su zainteresovani da nauče detaljnije o samoj strukturi django projekata, dostupna je visoko kvalitetna online [dokumentacija](#).

4.5 Frontend web aplikacije

Kako je projekat u svojoj inicijalnoj fazi, izgled stranica nije na nivou na kojem bi trebalo da bude kada aplikacija bude live. Korišćen je [Bootstrap](#) frontend CSS framework, u cilju ulepšavanja stranica. Ipak, autori smatraju da je neophodno uvesti dodatan JavaScript kod uz *AJAX (Asynchronous JavaScript And XML)*, koji bi višestruko poboljšali kvalitet korisničkog iskustva.

JavaScript i AJAX pristup čine ogromnu razliku kod svih softverskih proizvoda realizovanih kao web aplikacije. Projekat u vidu django web aplikacije *bookMentor* bi iskoristio ove softverske alate u cilju poboljšanja korisničkog interfejsa, koji je u trenutnoj fazi poprilično jednostavan iako je potpuno operativan.

U trenutnom formatu, aplikacija ima minimalno JavaScript koda koji je neophodan za potpuno gladak rad. U određenim delovima dodata je funkcija AJAX-a.

Glava 5

Budućnost projekta

5.1 Buduće funkcionalnosti i poboljšanja

Kao što je naznačeno u Uvodnom poglavlju dokumentacije, postoji više različitih ideja na kojima će se raditi na projektu. S obizorm na to da je inicijalna ideja da projekat ne ostane samo projekat, već komercijalni besplatni proizvod koji bi zapravo služio studentima Elektrotehničkog fakulteta u Beogradu, potrebno je da se obrati pažnja i posveti vreme određenim stvarima.

Pre nego projekat može da bude pušten u *Production phase*, neophodno je izvršiti pronalaženje adekvatnog servera (adekvatnog u smislu procesorskih mogućnosti i količine memorije). Određene algoritme treba još unaprediti u smislu vremenske efikasnosti, iako je većina rađena da već bude dovoljno optimizovana.

Jedna od najvažnijih stavki u životnom ciklusu svih projekata je rešavanje postojećih problema kako ne bi došlo do nagomilavanja u budućnosti. Ideja je da se kod što više prilagodi prema konvenciji **Čistog koda**, radi lakšeg dodavanja novih funkcionalnosti. Poželjno je izvršiti dodatnu normalizaciju baze podataka sa ciljem dodavanja novih tabela u budućnosti. Dodatni alati ažuriranja korisnika su takođe neophodna nova stavka.

Prethodno opisana ažuriranja koda direktno omogućuju dodavanje novih funkcionalnosti kao što su forum/blog sistem na kojem bi korisnici mogli da dele iskustva čitanja svojih knjiga.

Razmatrano je uvođenje sistema zapraćivanja, čime bi se projekat podigao na nivo akademske društvene mreže Elektrotehničkog fakulteta u Beogradu. Tu je i ideja automatizacije dodavanja novih knjiga. Naravno, autori su svesni pompeznosti ovakvih zahteva u krakom vremenskom periodu, te ova ideje ostaju jedne od onih koje se mogu realizovati u nekom trenutku u budućnosti.

Glava 6

Zaključak

bookMentor je projekat studenti - studentima, koji za cilj ima da svi jednako kvalitetno napredujemo na svom stručnom putu. Kombinuje nekoliko različitih, a opet povezanih paradigmi i tehnologija Računarske nauke, s ciljem da bude kvalitetan proizvod. Na osnovu ličnih afiniteta ali i afiniteta sličnih korisnika, sistem preporučuje knjige svojim korisnicima. Predstavlja hibridni oblik studentske biblioteke širokog spektra znanja.

Projektna dokumentacija je za cilj imala da na najjednostavniji način prikaže iscrpan proces integracije različitih segmenata s ciljem dobijanja kvalitetnog završnog proizvoda.

Projekat je tek u početnoj fazi svog razvijanja i sklon je minornim ali i onim većim promenama. Ceo programski kod je dostupan, čime se makar malo doprinosi ideji slobodnog softvera, što ovaj projekat i jeste.

Autori su dostupni za sve predloge i kritike, a s ciljem dobijanja još kvalitetnijeg proizvoda.

Literatura

- [1] *PostgreSQL dokumentacija za verziju 15*, elektronski dokument, 2023; dostupno na: <https://www.postgresql.org/docs/15/index.html>;
- [2] *django dokumentacija za verziju 4*, elektronski dokument, 2023; dostupno na: <https://docs.djangoproject.com/en/4.2/>;
- [3] *Python dokumentacija*, elektronski dokument, 2023; dostupno na: <https://docs.python.org/3/>;
- [4] *Pandas dokumentacija*, elektronski dokument, 2023; dostupno na: <https://pandas.pydata.org/docs/>;
- [5] A. Mele, *Django 3 By Example: Build powerful and reliable Python web applications from scratch 3rd Edition*, Packt Publishing, 2020;
- [6] K. Falk, *Practical Recommender Systems*, Manning, 2019;