UNIVERZITET U NOVOM SADU
PRIRODNO-MATEMATIČKI FAKULTET
DEPARTMAN ZA MATEMATIKU I INFORMATIKU

Ivana Milutinović, 22/18

Aplikacija za upravljanje izdavačkom kućom

- seminarski rad iz predmeta Skript jezici-

Novi Sad, 2020/21.

Sadržaj

1. Uvod

1.1. O Python programskom jeziku

Programski jezik Python je jedan od retkih jezika koji su u isto vreme jednostavni i moćni. Jednostavna sintaksa čini ovaj jezik savršenim za početnike u programiranju. Ovaj programski jezik je takođe dobar i vrlo čest izbor pravih eksperata programiranja jer omogućava da se programer posveti razmišljanju o problemu, a ne da se fokusira na samu sintaksu.

Python je interpreterski jezik što znači da se kod izvšava uz pomoć interpretatora, bez potrebe za kompajliranjem. Podržava imperativni, objektno-orijentisan i funkcionalni stil programiranja. Python je nastao početkom devedesetih godina prošlog veka a njegov autor je Gvido van Rosum.

1.2. O samoj aplikaciji

Ova aplikacija je osmišljena kako bi simulirala rad jedne izdavačke kuće. Učesnici koju mogu koristiti funkcionalnosti ove aplikacije su: neregistrovani korisnik, registrovani korisnik i radnik u izdavačkoj kući.

Neregistrovani korisnik, da bi koristio funkcionalnosti koje aplikacija nudi, mora da se registruje i to je jedina funkcionalnost koju će on kao takav imati. Registrovani korisnik i radnik će imati niz funkcionalnosti koje će u nastavku biti navedene i detaljnije opisane.

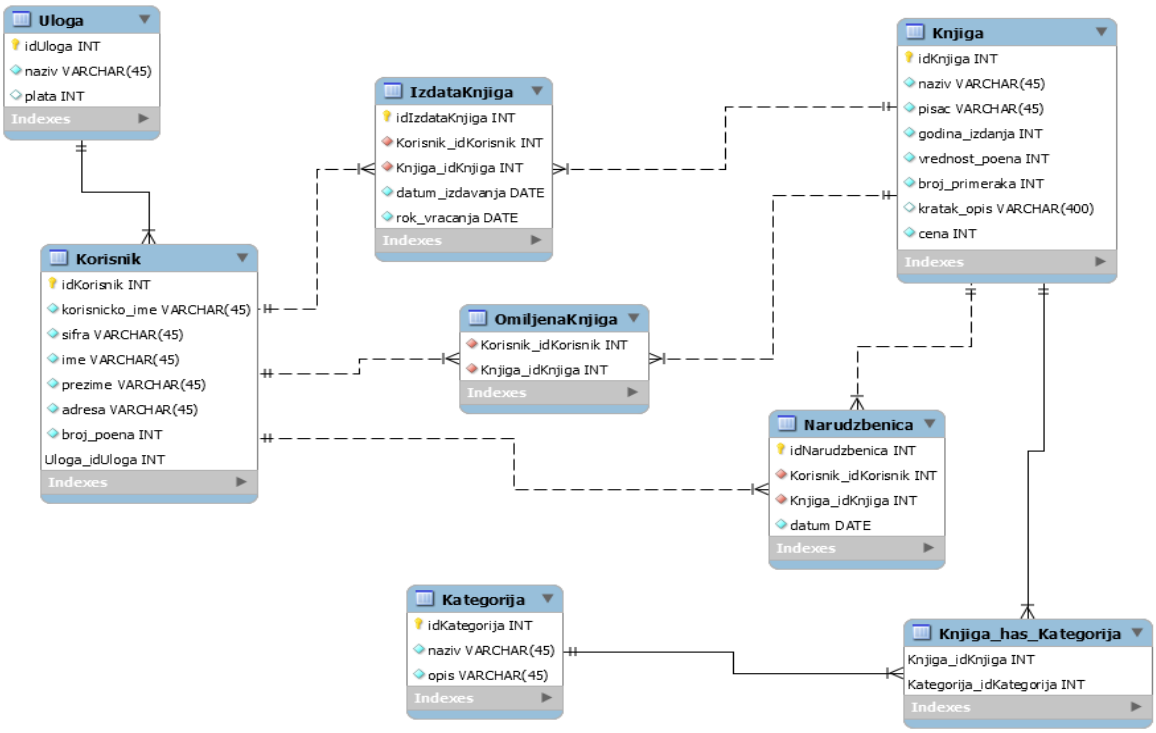
Moduli koji se nalaze u ovom projektu su:

- izdavacka_kuca
- interface
- crud
- globalne_funkc
- interface_user
- interface_admin

U modulu **izdavacka_kuca** nalazi se funkcija main koja se poziva pri svakom pokretanju programa, odnosno ovaj fajl se pokreće kada želimo da startujemo aplikaciju.
U modulu **interface** nalaze se funkcije: pocetni_meni, dobrodoslica, logovanje i registrovanje. Ove funkcije će biti iste za sve korisnike aplikacije (neregistrovani / registrovani korisnik i radnik) odnosno to je ono što će svi učesnici moći da vide i koriste kada se program pokrene, i iz tog razloga su navedene funkcije izdvojene u zaseban modul.
U modulu **crud** se nalaze sve pomoćne funkcije koje će se pozivati u funkcijama koje se nalaze u modulima **interface_user** i **interface_admin** i prvenstveno će služiti za komunikaciju sa bazom podataka putem SQL upita.

U modulu **globalne_funkc** nalaze se funkcije: prikaz_kategorija, prikaz_knjiga, prikaz_knjige i zavrsi_program. Kada je učesnik ulogovan, bilo da je registrovani korisnik ili radnik, ovo su funkcionalnosti koje će njemu biti omogućene.
U modulu **interface_user** nalaze se funkcije: meni, izabrana_opcija, prikaz_knjige, prikaz_omiljenih_knjiga, narucivanje_knjige, iznajmljivanje_knjige, iznajmi, naruci i prikaz_narucenih_knjiga. Registrovanom korisniku će se na početku prikazati meni sa navedenim funkcionalnostima koje mu aplikacija nudi.
U modulu **interface_admin** nalaze se funkcije: meni, izabrana_opcija, prikaz_knjige, azuriranje_informacija, azuriranje, brisanje_knjige, brisanje, prikaz_izdatih_knjiga, dodavanje_knjige, prikaz_svih_korisnika, prihod_knjiga, prikaz_najomiljenijih_knjiga
Radniku će se na početku prikazati meni sa navedenim funkcionalnostima koje mu aplikacija nudi.

Na ***slici 1.2.1.*** prikazana je relaciona šema baze podataka koja će biti korišćenja za potrebe realizacije ovog projekta.



Slika 1.2.1: Relaciona šema baze podataka

Baza podataka sadrži tabele Uloga, Korisnik, Knjiga, Kategorija i presečne tabele Knjiga_has_Kategorija, IzdataKnjiga, OmiljenaKnjiga i Narudžbenica. Veza između tabele Uloga i Korisnik je "1 prema N" odnosno tabela Korisnik kao strani ključ ima ključ tabele Uloga u zavisnosti od uloge korisnika u aplikaciji (registrovani korisnik(1) ili radnik(2)). Veza između tabela Knjiga i Kategorija je "N prema N" što znači da se jedna knjiga može nalaziti u više kategorija i kategorija može imati više knjiga, pa je zato potreba za presečnom tabelom Knjiga_has_Kategorija u kojoj će se čuvati ključevi prethodno dve navedene tabele. Analogno je za presečnu tabelu OmiljeneKnjige, s tim što se radi o vezi između tabela Korisnik i Knjiga. Postoje još dve veze "N prema N" između tabela Korisnik i Knjiga i te veze rezultuju presečnim tabelama IzdataKnjiga koja ima svoj ključ idIzdataKnjiga, a kao strane ključeve dobija idKorisnik i idKnjiga, i Narudžbenica (analogno sa IzdataKnjiga).

2. Opis programa

Kao što je u prethodnoj sekciji navedeno, program se sastoji od šest Python modula: izdavacka_kuca, interface, crud, globalne_funkc, interface_user i interface_admin. Svaki od modula će u nastavku biti posebno opisan u podsekcijama ove sekcije.

2.1. Modul izdavacka_kuca

Ovaj modul služi isključivo za pokretanje programa i sastoji se od poziva funkcije main u kojoj je poziv funkcije pocetni_meni iz modula interface. (***listing 2.1.1***)

```
import interface as i
def main():
    i.pocetni_meni()
if __name__ == "__main__":
    main()
```

Listing 2.1.1

2.2. Modul interface

Iz funkcije *main* modula izdavacka_kuca se poziva funkcija *pocetni_meni* i ona predstavlja ono što se svakom korisniku prikaže pri pokretanju programa. Kod funckije *pocetni_meni* je prikazan na ***listingu 2.2.1***.

```
def pocetni_meni():
    dobrodoslica()
    print("\n{}Početni meni{}".format("-"*17, "-"*18))
    print("\n1) Uloguj se \n2) Registruj se \n3) Završi program\n")
    print("-"*47)
    opc = eval(input("Unesite željenu opciju ---> "))
    print("\n{}".format("-"*47))
    if opc == 1:
        logovanje()
    elif opc == 2:
        registrovanje()
    elif opc == 3:
        gf.zavrsi_program()
    else:
        print("\nUnesite jednu od ponuđenih opcija!!!\n")
        pocetni_meni()
```

Listing 2.2.1

Funkcija koja se na početku poziva u ovoj metodi je *dobrodoslica* i njen kod je prikazan na ***listingu 2.2.2.***

```
def dobrodoslica():
    print("="*47)
    print("{:<46}|".format("|"))
    print("{:<5} DOBRO DOŠLI U IZDAVAČKU KUĆU 'SOFIA' {:>4}".format("|", "|"))
    print("{:<46}|".format("|"))
    print("="*47)
```

Listing 2.2.2

Kada korisnik unese 1, tok programa se prebacuje na funkciju *logovanje* koja omogućava korisniku da unese svoje korisničko ime i šifru kako bi pristupio aplikaciji i bio u mogućnosti da koristi sve funkcionalnosti koje su njemu namenjene. Kod ove funkcije je prikazan na ***listingu 2.2.3.***

```
def logovanje():
    global trenutni_korisnik
    account = input("Unesite korisničko ime: ")
    passw = input("Unesite šifru: ")
    print("\n{}".format("-"*47))
    trenutni_korisnik = crud.pronadji_korisnika(account, passw)
    try:
        uloga = trenutni_korisnik[7]
    except:
        uloga = 3
    if uloga == 1:
        print("\n{}".format("-"*47))
        print("Zdravo, {}! Uspešno ste se ulogovali!".format(trenutni_korisnik[3]))
        print("-"*47)
        gf.inicijalno(trenutni_korisnik)
        user.meni(trenutni_korisnik)
    elif uloga == 2:
        print("\n{}".format("-"*47))
        print("Zdravo, {}! Uspešno ste se ulogovali!".format(trenutni_korisnik[3]))
        print("-"*47)
        gf.inicijalno(trenutni_korisnik)
        admin.meni(trenutni_korisnik)
    else:
        print("{}".format("-"*47))
        print("\nNeuspešno logovanje!\n")
        print("{}".format("-"*47))
        print("\n1) Pokušajte ponovo\n2) Vratite se na početni meni")
        print("\n{}".format("-"*47))
        opc = eval(input("Unesite željenu opciju ---> "))
        print("\n{}".format("-"*47))
        if opc == 1:
            logovanje()
            return
        else:
            pocetni_meni()
```

Listing 2.2.3

Kada korisnik unese 2, tok programa se prebacuje na funkciju *registrovanje* koja omogućuva da korisnik unese potrebne podatke kako bi se registrovao i bio u mogućnosti da koristi sve funkcionalnosti koje su predviđene registrovanim korisnicima. (***listing 2.2.4***)

```
def registrovanje():
    korisnicko_ime = input("Unesite korisničko ime: ")
    sifra1 = input("Unesite šifru: ")
    while not re.match("^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)[a-zA-Z\d]{6,10}$", sifra1):
        print("Šifra mora sadržati velika i mala slova, brojeve i biti \
                dužine bar 6. Pokušajte ponovo.")
        sifra1 = input("Unesite šifru: ");
    sifra2 = input("Potvrdite unetu šifru: ");
    while sifra1 != sifra2:
        print("Nepoklapaju se šifre, probajte ponovo!")
        sifra1 = input("Unesite šifru: ")
        while not re.match("^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)[a-zA-Z\d]{6,10}$", sifra1):
            print("Šifra mora sadržati velika i mala slova, brojeve i biti \
                    dužine bar 6. Pokušajte ponovo.")
            sifra1 = input("Unesite sifru: ");
        sifra2 = input("Potvrdite unetu šifru: ");
    ime_prezime = input("Unesite ime i prezime: ")
    while not ime_prezime.split(" "):
        print("Ime i prezime mora biti odvojeno razmakom!")
        ime_prezime = input("Unesite ime i prezime: ")
    ime, prezime = ime_prezime.split(" ")
    adresa = input("Unesite adresu: ")
    crud.registrovanje_korisnika(korisnicko_ime, sifra1, ime, prezime, adresa)
    print("Uspešno ste se registrovali!")
    pocetni_meni()
```

Listing 2.2.4

Kada korisnik unese 3, poziva se funkcija *zavrsi_program* i tada program prekida sa radom i završava se. Ova funkcija je u modulu *globalne_funkc* i ona će u toj sekciji biti dodatno pojašnjena.

Konzolni prikaz funkcije *pocetni_meni* je prikazan na ***slici 2.2.1***.

```
==================================================
|                                                |
|      DOBRO DOŠLI U IZDAVAČKU KUĆU 'SOFIA'      |
|                                                |
==================================================

------------------Početni meni--------------------

1) Uloguj se
2) Registruj se
3) Završi program

--------------------------------------------------

Unesite željenu opciju --->
```

Slika 2.2.1: Konzolni prikaz funkcije

2.3. Modul crud

U projektu se koristi MySQL Connector/Python koji predstavlja samostalni Python driver za komunikaciju sa MySQL serverima. Objekti veze uspostavljaju konekciju sa bazom podataka i oni se dalje koriste za različite transakcije. Oni takođe predstavljaju sesije baze podataka. Kursor je jedna od moćnijih karakteristika SQL-a i njegovi objekti su odgovorni za prosleđivanje različitih SQL upita serveru baze podataka. Navedeni kod u ***listingu 2.3.1.*** se upravo nalazi u ovom modulu i predstavlja gore opisane stavke koje su neophodne.

```
import mysql.connector
import datetime as dt
db = mysql.connector.connect(host='localhost',user='root',\
                             password='ivanna1109', database='pythondb')
kursor = db.cursor()
```

Listing 2.3.1

Kao što je u uvodu navedeno, ovaj modul sadrži funkcije za komuniciranje sa bazom podataka putem MySQL Connector-a koristeći objekat klase Cursor. Ove funkcije su faktički pomoćne funkcije jer se uvek indirektno pozivaju od strane onih glavnih funkcija koje koristi korisnik. Biće navedene neke karakteristične funkcije iz modula.

Što se tiče dodavanja u bilo koju tabelu baze podataka, uzećemo kao primer funkciju *registrovanje_korisnika* koja dodaje korisnika koji se registrovao u tabelu Korisnik. (***listing 2.3.2***). Slično su realizovane sledeće funkcije uz još neke dodatne provere pre samog dodavanja, ili izmene druge tabele nakon dodavanja: *dodaj_omiljene, iznajmi_knjigu(**listing 2.3.3**), naruci_knjigu,* i *dodavanje_nove_knjige.*

```
def registrovanje_korisnika(korisnicko_ime, sifra1, ime, prezime, adresa):
    vrednosti = (korisnicko_ime, sifra1, ime, prezime, adresa, 100, 1)
    kursor.execute('insert into Korisnik (korisnicko_ime, sifra, ime, \
                    prezime, adresa, broj_poena, Uloga_idUloga) \
                    values (%s, %s, %s, %s, %s, %s, %s)', vrednosti)
    db.commit()
```

Listing 2.3.2

```
def iznajmi_knjigu(idK, poeniKorisnika, idKnjige,  poeni, primerci):
    vrednosti = (idK, idKnjige,\
                 dt.datetime.now(), dt.datetime.now()+dt.timedelta(10))
    kursor.execute('insert into IzdataKnjiga \
                   (Korisnik_idKorisnik, Knjiga_idKnjiga,\
                    datum_izdavanja, rok_vracanja)\
                    values (%s, %s, %s, %s)', vrednosti)
    vrednosti = (poeniKorisnika - poeni, idK)
    kursor.execute('update Korisnik set broj_poena = %s where \
                   idKorisnik = %s', vrednosti)
    vrednosti = (primerci-1, idKnjige)
    kursor.execute('update Knjiga set broj_primeraka = %s where \
                    idKnjiga = %s', vrednosti)
    db.commit()
```

Listing 2.3.3

Što se tiče pretraživanja i pribavljanja željenih podataka iz baze, imamo nekoliko funkcija, kako onih što vraćaju sve podatke iz neke tabele, tako i onih koji vraćaju podatke koji zadovoljavaju prosleđeni uslov. To su funkcije: *pronadji_korisnika, sve_kategorije, sve_knjige_za_kat, proveri_omiljene(**listing 2.3.4**.), omiljene_knjige, narucene, nadji_knjigu_u_omiljenim, nadji_knjigu_u_iznajmljenim, nadji_knjigu_u_narudžbenicama, nadji_knjigu* i *lista_korisnika*.

```
def proveri_omiljene(idK, idKnjige):
    vrednosti = (idK, idKnjige)
    kursor.execute('select * from OmiljenaKnjiga where \
      Korisnik_idKorisnik like %s and Knjiga_idKnjiga like %s', vrednosti)
    if kursor.fetchall():
        return True
    return False
```

Listing 2.3.4

Postoje i funkcije koje samo menjaju vrednosti određenih polja u tabeli i to su sledeće funkcije: *azuriranje_poena, azuriranje_primeraka (**listing 2.3.5**) i azuriranje_cene*.

```
def azuriranje_primeraka(idKnjige, primerci):
    vrednosti = (primerci, idKnjige)
    kursor.execute('update Knjiga set broj_primeraka = %s where\
                    idKnjiga = %s', vrednosti)
    db.commit()
```

Listing 2.3.5

Funkcija koja kaskadno briše prosleđenu knjigu je *izbrisi_knjigu* i prikazana je na ***listingu 2.3.6.*** Ova funkcija će prvo izbrisati prosleđenu knjigu iz tabele Knjiga_has_Kategorija, pa zatim iz glavne tabele Knjiga.

```
def izbrisi_knjigu(idKnjiga):
    vrednost = (idKnjiga, )
    try:
        kursor.execute('delete from Knjiga_has_Kategorija where\
                        Knjiga_idKnjiga = %s', vrednost)
        db.commit()
    except Exception as e:
        print("Exception u brisanju u Knjiga has kategorija: {}".format(e))
    try:
        kursor.execute("delete from Knjiga where \
                        (idKnjiga like '%s')", vrednost)
        db.commit()
    except Exception as e:
        print("Exception u brisanju knjige: {}".format(e))
```

Listing 2.3.6

Funkcije *izdate_chart, omiljene_chart* i *najprodavanije_chart* će biti prikazane i objašnjene u sekciji modula koji sadrži funkcije koje će pozivati ove navedene funkcije.

2.4. Modul globalne_funkc

U ovom modulu se nalaze funkcije koje su, nakon logovanja korisnika na sistem, zajedničke za sve, nebitno od uloge koju ulogovani korisnik ima.

Funkcija koju na početku posebno treba izdvojiti jeste funkcija *inicijalno*. Ona kao parametar prima korisnika koji se ulogovao i koji će u nastavku koristiti aplikaciju. Vrednost prosleđenog parametra se dodeljuje globalnoj promenljivoj *korisnik* koja je na početku inicijalizovana na *None*. ***(listing 2.4.1)***

```
korisnik = None
def inicijalno(k):
    global korisnik
    korisnik = k
```

Listing 2.4.1

U ovom modulu imamo pored navedenog i četiri globalna polja:

- globalne promenljive vezane za usera
  - iznajmljivanjeGlobal = False
  - narucivanjeGlobal = False
- globalne promenljive vezane za admina
  - brisanjeGlobal = False
  - azuriranjeGlobal = False

Vrednost ovih promeljivih će se menjati u zavisnosti od toga koju akciju ulogovani korisnik želi da realizuje i u skladu sa tim će se menjati dalji tok izvršavanja programa koji će dovesti do ostvarivanja željenje radnje.

Funkcije koje će kao parametar primati predefinisane promenljive čija će se vrednost stalno dodeljivati navedenim globalnim poljima, pa se proverati njihova vrednost i u skladu sa tim realizovati dalji tok programa su: *prikaz_kategorija* i *prikaz_knjige*.

Funkcija *prikaz_kategorija* poziva funkciju sve_kategorije iz modula *crud* koja je zadužena za komuniciraju sa bazom i ta metoda će vratiti sve kategorije koje se nalaze u tabeli Kategorija i ispisati ih korisniku. Da bi se tok nastavio dalje, korisnik treba da izabere jednu od ponudjenih kategorija za koju će mu se prikazati knjige koje pripadaju odabranoj kategoriji. Funkcija je napisana tako da faktički primorava korisnika da unese jednu od navedenih opcija, u suprotnom će mu ispisivati određenu poruku i stalno ga vraćati na taj meni. Takođe, korisnik kao poslednju opciju ima mogućnost da se vrati na svoj početni meni u zavisnosti od uloge. (***listing 2.4.2***)

Funkcija *prikaz_knjiga* će na osnovu prethodno odabrane kategorije, pozvati funkciju *sve_knjige_za_kat* iz modula *crud* koja će iz baze na osnovu kategorije, korisniku ispisati knjige koje pripadaju toj kategoriji. Za nastavak programa, korisnik treba da unese redni broj knjige koju želi detaljnije da pregleda, i kao i u prethodnom slučaju ga funkcija primorava da izabere jednu od ponuđenih opcija. Takođe, korisnik ima opciju da se vrati i jedan korak unazad, tj na funkciju *prikaz_kategorija*. (***listing 2.4.3***)

```
def prikaz_kategorija(iznajmljivanje = False, narucivanje = False, \
brisanje_admin = False, azuriranje_admin = False):
    global iznajmljivanjeGlobal
    iznajmljivanjeGlobal = iznajmljivanje
    global narucivanjeGlobal
    narucivanjeGlobal = narucivanje
    global brisanjeGlobal
    brisanjeGlobal = brisanje_admin
    global azuriranjeGlobal
    azuriranjeGlobal = azuriranje_admin
    global korisnik
    kategorije = crud.sve_kategorije();
    i = 1
    print("Prikaz kategorija: ")
    print("{}\n".format("-"*47))
    for k in kategorije:
        print("{}) {}".format(i, k[1]))
        i+=1
    print("{}) Povratak na meni\n".format(i))
    print("-"*47)
    opcija = eval(input("Unesite željenu opciju ---> "))
    print("\n{}".format("-"*47))
    kategorija = None
    if opcija == i:
        if korisnik[7] == 1:
            user.meni(korisnik)
            return
        else:
            admin.meni(korisnik)
            return
    try:
        kategorija = kategorije[opcija-1]
    except:
        print("-"*47)
        print("\nUnesite jednu od ponuđenih opcija!\n")
        print("-"*47)
        prikaz_kategorija()
        return
    try:
        prikaz_knjiga(kategorija[0])
    except Exception as e:
        print(e)
```

Listing 2.4.2

```
def prikaz_knjiga(idKat):
    print("Prikaz knjiga za izabranu kategoriju: ")
    print("{}\n".format("-"*47))
    knjige = crud.sve_knjige_za_kat(idKat)
    i = 1
    for k in knjige:
        print("{})'{}' - {}".format(i, k[1], k[2]))
        i+=1
    print("{}) Vratite se na prikaz kategorija\n".format(i))
    print("{}".format("-"*47))
    opcija = eval(input("Unesite željenu opciju ---> "))
    print("\n{}".format("-"*47))
    if opcija-1 == len(knjige):
        prikaz_kategorija()
        return
    knjiga = None
    try:
        knjiga = knjige[opcija-1]
    except:
        print("-"*47)
        print("\nUnesite jednu od ponudjenih opcija!\n")
        print("-"*47)
        prikaz_knjiga(idKat)
    prikaz_knjige(knjiga, idKat)
```

Listing 2.4.3

Kada korisnik izabere knjigu koju želi da pregleda, u zavisnosti od njegove uloge, prikazuju mu se detalji odabrane knjige kao i opcije koje mu se nude. Obzirom da su mogućnosti koje aplikacija nudi različite u zavisnosti od uloge, poziva se funkcija iz odgovarajućeg modula. (***listing 2.4.4***)

```
def prikaz_knjige(k, idKat):
    print("\n'{}' - {} ({})\n{}\nOpis: {}\n{}\nCena knjige: {}din\n{}"\
              .format(k[1], k[2], k[3],"-"*47, k[6], "-"*47, k[7], "-"*47))
    global iznajmljivanjeGlobal
    global narucivanjeGlobal
    global brisanjeGlobal
    global azuriranjeGlobal
    global korisnik
    if korisnik[7] == 1:
        user.prikaz_knjige(k, idKat, iznajmljivanjeGlobal, \
                                   narucivanjeGlobal)
    else:
        admin.prikaz_knjige(k, idKat, brisanjeGlobal, azuriranjeGlobal)
```

Listing 2.4.4

Kao što je ranije i navedeno, ovde se nalazi i funkcija zavrsi_program koja se poziva svaki put kada korisnik unese opciju koja označava završetak rada programa. Ova funkcija će korisniku samo ispisati da se program završio i on više neće imati mogućnost da koristi aplikaciju sve dok ponovo ne pokrene program iz modula *izdavacka_kuca*. (***listing 2.4.5***)

```
def zavrsi_program():
    print("Završen program.")
```

Listing 2.4.5

2.5 Modul interface_user

Ovaj modul sadrži sve funkcije koje su dostupne korisniku koji ima ulogu registrovanog korisnika.

Funkcija *meni* (***listing 2.5.1***) registrovanom korisniku prikazuje sve mogućnosti koje on ima i primorava ga da unese neku od ponuđenih opcija pozivom funkcije *izabrana_opcija* (***listing 2.5.2***).

```
def meni(trenutni_korisnik):
    global korisnik
    korisnik = trenutni_korisnik
    print("\n---------------------Meni----------------------\n")
    print("1) Prikaz svih kategorija")
    print("2) Prikaz naručenih knjiga i ukupnog iznosa")
    print("3) Prikaz omiljenih knjiga")
    print("4) Iznajmljivanje knjige")
    print("5) Naručivanje knjige")
    print("6) Odjavi se\n")
    print("-----------------------------------------------")
    opc = eval(input("Unesite željenu opciju ---> "))
    print("\n{}".format("-"*47))
    izabrana_opcija(opc)
```

Listing 2.5.1

```
def izabrana_opcija(o):
    switch = {
        1: gf.prikaz_kategorija,
        2: prikaz_narucenih_knjiga,
        3: prikaz_omiljenih_knjiga,
        4: iznajmljivanje_knjige,
        5: narucivanje_knjige,
        6: gf.zavrsi_program
        }
    try:
        return switch[o]()
    except:
        print("\nUnesite jednu od ponuđenih opcija!")
        print("\n{}".format("-"*47))
        meni(korisnik)
        return
```

Listing 2.5.3

Kao što je navedeno u prethodnoj sekciji o modulu *globalne_funkc*, imamo predefinisane vrednosti na osnovu kojih ćemo da znamo koju akciju korisnik želi da preduzme. Funkciji *prikaz_knjige* (***listing 2.5.4***) se prosleđuju te predefinisane vrednosti u slučaju da korisnik želi da iznajmi ili naruči knjigu i u skladu sa tim se nastavlja dalji tok programa. Ako su te obe vrednosti postavljene na False, korisniku se nudi opcija da prikazanu knjigu doda u svoje omiljene knjige i pritom se proverava da li je već tu knjigu dodao u tu listu (poziva se funkcija iz *crud*-a).

```
def prikaz_knjige(k, idKat, iznajmljivanje, narucivanje):
    global iznajmljivanjeGlobal
    iznajmljivanjeGlobal = iznajmljivanje
    global narucivanjeGlobal
    narucivanjeGlobal = narucivanje
    global korisnik
    if iznajmljivanjeGlobal:
        iznajmi(k)
    elif narucivanjeGlobal:
        naruci(k)
    else:
        print("\n1) Dodaj knjigu u omiljene\n2) Povratak na listu \
                knjiga\n")
        print("-"*47)
        opcija = eval(input("Unesite željenu opciju ---> "))
        print("\n{}".format("-"*47))
        if opcija == 1:
            if crud.dodaj_omiljene(korisnik[0], k[0]):
                print("\nUspešno ste dodali knjigu u Vaše omiljene \
                        knjige!\n")
            else:
                print("\nKnjiga je već u Vašim omiljenim knjigama.\n")
            print("-"*47)
            meni(korisnik)
            return
        elif opcija == 2:
            gf.prikaz_knjiga(idKat)
            return
        else:
            print("\nUnesite jednu od ponuđenih opcija!")
            print("\n{}".format("-"*47))
            prikaz_knjige(k,idKat, iznajmljivanje, narucivanje)
            return
```

Listing 2.5.3

Kada korisnik želi da iznajmi knjigu, odmah na početnom meniju kada on izabere tu opciju, poziva se funkcija *iznajmljivanje_knjige* (***listing 2.5.4***) i tu se poziva funkcija iz modula *global_funkc prikaz_kategorija* kojoj se prosleđuje predefinisana vrednost vezana za iznajmljivanje koja se postavlja na True.

```
def iznajmljivanje_knjige():
    gf.prikaz_kategorija(iznajmljivanje = True)
```

Listing 2.5.4

Kada se stigne do funkcije prikaz_knjiga, kao što je gore prethodno navedeno, ako je korisnik na početku izabrao da želi da iznajmi knjigu, kada mu se knjiga prikaže, odmah će se nastaviti sa iznajmljivanjem ako za to postoji mogućnost. Poziva se funkcija *iznajmi* (***listing 2.5.5***) koja ima poziv funkcije iz modula *crud iznajmi_knjigu* koja će podatak o izdatog knjizi dodati u tabelu IzdataKnjiga. Kada se realizuje dodavanje izdate knjige u bazu, korisniku će se prikazati *meni*.

```
def iznajmi(k):
    global korisnik
    if k[4] > korisnik[6]:
        print("\nNemajte dovoljno poena da iznajmite ovu knjigu.")
        input("Za povratak na početni meni, pritisnite ENTER ---> ")
        print("\n{}".format("-"*47))
        meni(korisnik)
        return
    if k[5] == 0:
        print("Nema primeraka knjige na lageru.")
        input("Za povratak na početni meni, pritisnite ENTER ---> ")
        print("\n{}".format("-"*47))
        meni(korisnik)
        return
    input("Za nastavak sa iznajmljivanjem knjige ({} poena),\
          pritisnite ENTER.\nNa Vašem računu imate {} poena.\n\"
          .format(k[4], korisnik[6]))
    crud.iznajmi_knjigu(korisnik[0], korisnik[6], k[0], k[4], k[5])
    print("-"*47)
    print("\nUspešno ste iznajmili knjigu!\n")
    print("-"*47)
    meni(korisnik)
```

Listing 2.5.5

Analogno sa prethodno navedenim funkcijama *iznajmljivanje_knjige* i *iznajmi* se realizuju i funkcije *narucivanje_knjige* (***listing 2.5.6***) i *naruci* (***listing 2.5.7***).

```
def narucivanje_knjige():
    gf.prikaz_kategorija(narucivanje = True)
```

Listing 2.5.6

```
def naruci(k):
    global korisnik
    if k[5] == 0:
        print("Nema primeraka knjige na lageru.")
        input("\nZa povratak na početni meni, pritisnite ENTER ---> ")
        meni(korisnik)
        return
    input("Knjiga će biti dostavljena na kućnu adresu.\n\
          Plaćanje pouzećem.\nCena knjige: {}din\nZa nastavak\
          kupovine knjige, pritisnite ENTER ---> ".format(k[7]))
    crud.naruci_knjigu(korisnik[0], k[0], k[5], k[7])
    print("\n{}".format("-"*47))
    print("\nUspešno ste naručili knjigu!\n")
    print("-"*47)
    meni(korisnik)
```

Listing 2.5.7

Pored navedenih funkcionalnosti, registrovanom korisniku je omogućeno i da pogleda listu svojih omiljenih knjiga pozivom funkcije *prikaz_omiljenih_knjiga* (***listing 2.5.8***). Ova funkcija poziva funkciju *omiljene_knjige* iz modula *crud* koja će iz baze na osnovu id-a registrovanog korisnika izlistati sve njegove omiljene knjige. U slučaju da registrovani korisnik nema nijednu omiljenu knjigu, ispisaće mu se adekvatna poruka o tome i vratiće se na *meni*.

```
def prikaz_omiljenih_knjiga():
    global korisnik
    omiljene = crud.omiljene_knjige(korisnik[0])
    if omiljene:
        print("-"*47)
        print("\nVaše omiljene knjige:")
        print("-"*47)
        i = 1
        for o in omiljene:
            omiljena_naziv = crud.nadji_knjigu(o[1])
            if omiljena_naziv != None:
                print("{}) {}".format(i, omiljena_naziv[1]))
                print("-"*47)
                i+=1
    else:
        print("Nemate nijednu knjigu u omiljenim!")
    input("\nZa povratak na meni, pritisnite ENTER ---> ")
    meni(korisnik)
```

Listing 2.5.8

Analogno je realizovana i funkcija *prikaz_narucenih_knjiga* (***listing 2.5.9***).

```
def prikaz_narucenih_knjiga():
    global korisnik
    narucene_knjige = crud.narucene(korisnik[0])
    if narucene_knjige:
        print("-"*65)
        print("\nVaše naručene knjige: ")
        print("-"*65)
        i = 1
        ukupna_suma = 0
        for k in narucene_knjige:
            knjiga = crud.nadji_knjigu(k[2])
            if knjiga:
                print("{}. {} - {} ({}din) \"
                          .format(i, knjiga[1], knjiga[2], knjiga[7]))
                print("-"*65)
                ukupna_suma += knjiga[7]
        print("\nUkupna suma svih \
              naručenih knjiga je: {}.00".format(ukupna_suma))
        print("-"*65)
    else:
        print("Nemate naručenih knjiga!")
    input("Za povratak na meni, pritisnite ENTER ---> ")
    meni(korisnik)
```

Listing 2.5.9

2.6. Modul interface_admin

Ovaj modul sadrži sve funkcije koje su dostupne korisniku koji ima ulogu radnika.

Analogno sa funkcijama iz modula *interface_user*, i ovde su na sličan način realizovane funkcije *meni* (***listing 2.6.1***) i *izabrana opcija* (***listing 2.6.2***), ali tako da odgovoraju funkcionalnostima koje se nude radniku.

```
def meni(trenutni_korisnik):
    global korisnik
    korisnik = trenutni_korisnik
    print("\n---------------------Meni----------------------\n")
    print("1) Prikaz svih kategorija")
    print("2) Prikaz svih korisnika")
    print("3) Prikaz broja izdatih knjiga po korisnicima")
    print("4) Prikaz 5 najomiljenijih knjiga")
    print("5) Prihod 5 najprodavanijih knjiga")
    print("6) Dodavanje nove knjige")
    print("7) Azuriranje informacije o knjizi")
    print("8) Brisanje knjige")
    print("9) Odjavi se\n")
    print("-"*47)
    opc = eval(input("Unesite željenu opciju ---> "))
    print("\n{}".format("-"*47))
    izabrana_opcija(opc)
```

Listing 2.6. 1

```
def izabrana_opcija(o):
    switch = {
        1: gf.prikaz_kategorija,
        2: prikaz_svih_korisnika,
        3: prikaz_izdatih_knjiga,
        4: prikaz_najomiljenijih_knjiga,
        5: prihod_knjiga,
        6: dodavanje_knjige,
        7: azuriranje_informacija,
        8: brisanje_knjige,
        9: gf.zavrsi_program
        }
    try:
        return switch[o]()
    except:
        print("\nUnesite jednu od ponuđenih opcija!")
        print("\n{}".format("-"*47))
        meni(korisnik)
        return
```

Listing 2.6.2

Radnik može da vidi sve registrovane korisnike iz baze podataka pozivom funkcije *prikaz_svih_korisnika* (***listing 2.6.3***) koja će mu ispisati tražene podatake. Ova funkcija će pozvati metodu *lista_korisnika* iz modula *crud* koja će joj vratiti sve registrovane korisnike koji se nalaze u bazi.

```
def prikaz_svih_korisnika():
    print("Lista svih registrovanih korisnika: ")
    korisnici = crud.lista_korisnika()
    if korisnici:
        for k in korisnici:
            print('{} {}; username: {}'.format(k[3], k[4], k[1]))
```

Listing 2.6.3

Kao što je prethodno navedeno u modulu *interface_user* vezano za iznajmljivanje i narucivanje knjiga, slična situacija je i u ovom modulu, samo se ovde radi o brisanju knjige i ažuriranju informacija o knjizi. Funkcija *prikaz_knjige* (***listing 2.6.4***) se realizuje slično kao u modulu *interface_user*.

```
def prikaz_knjige(k, idKat, brisanje_admin, azuriranje_admin):
    global brisanjeGlobal
    brisanjeGlobal = brisanje_admin
    global azuriranjeGlobal
    azuriranjeGlobal = azuriranje_admin
    if azuriranjeGlobal:
        azuriranje(k, idKat, brisanje_admin, azuriranje_admin)
    elif brisanjeGlobal:
        brisanje(k, idKat, brisanje_admin, azuriranje_admin)
```

Listing 2.6.4

Analogno sa funkcijama iznajmljivanje_knjige i narucivanje_knjige iz modula interface_user, ovde imamo sledeće funkcije: *brisanje_knjige* (***listing 2.6.5***) i *azuriranje_informacija* (***listing 2.6.6***)

```
def brisanje_knjige():
    gf.prikaz_kategorija(brisanje_admin = True)
```

Listing 2.6.5

```
def azuriranje_informacija():
    gf.prikaz_kategorija(azuriranje_admin = True)
```

Listing 2.6.6

Funkcija *brisanje* (***listing 2.6.7***) iz baze briše prosleđenu knjigu u slučaju da je niko nije naručio, iznajmio niti se ijednom korisniku ta knjiga nalazi u omiljenim knjigama.

```
def brisanje(k, idKat, brisanje_admin, azuriranje_admin):
    global korisnik
    if crud.nadji_knjigu_u_omiljenim(k[0]) or
       crud.nadji_knjigu_u_iznajmljenim(k[0]) or
       crud.nadji_knjigu_u_narudzbenicama(k[0]):
        input("Ne možemo izbrisati knjigu.\nZa povratak na meni, \
                    pritisnite ENTER ---> ")
        print("\n{}".format("-"*47))
        meni(korisnik)
        return
    else:
        input("Za nastavak sa brisanjem knjige, pritisnite ENTER ---> ")
        crud.izbrisi_knjigu(k[0])
        print("-"*47)
        print("\nUspešno ste izbrisali knjigu!\n")
        print("-"*47)
        meni(korisnik)
```

Listing 2.6.7

Funkcija *ažuriranje* (***listing 2.6.8***) menja vrednosti određenih kolona u tabeli Knjiga za prosleđenu knjigu. Polja koja se mogu menjati su: vrednost poena, broj primeraka i cena knjige. U zavisnosti koje se polje menja, poziva se određena funkcija iz modula *crud* i pravi izmene u bazi podataka.

```
def azuriranje(k, idKat, brisanje_admin, azuriranje_admin):
    global korisnik
    print("-"*47)
    print("\nOpcije za ažuriranje:")
    print("-"*47)
    print("1) Ažuriranje vrednosti poena")
    print("2) Ažuriranje broja primeraka")
    print("3) Ažuriranje cene")
    print("4) Vratite se na početni meni\n")
    print("-"*47)
    opcija = eval(input("\nUnesite željenu opciju ---> "))
    print("\n{}".format("-"*47))
    if opcija == 1:
        poen = eval(input("\nUnesite novu vrednost poena za knjigu ---> "))
        crud.azuriranje_poena(k[0], poen)
    elif opcija == 2:
        primerci = eval(input("\nUnesite novi broj primeraka knjige ---> "))
        crud.azuriranje_primeraka(k[0], primerci)
    elif opcija == 3:
        cena = eval(input("\nUnesite novu cenu knjige ---> "))
        crud.azuriranje_cene(k[0], cena)
    elif opcija == 4:
        meni(korisnik)
        return
    else:
        print("\nUnesite jednu od ponuđenih opcija!\n")
        prikaz_knjige(k, idKat, brisanje_admin, azuriranje_admin)
        return
    print("\n{}".format("-"*47))
    input("Ažuriranje je uspešno izvršeno!\nZa povratak na meni, \
          pritisnite  ENTER ---> ")
    print("\n{}".format("-"*47))
    meni(korisnik)
    return
```

Listing 2.6.8

Radniku je omogućeno da dodaje nove knjige. Uneće potrebne podatke i pre nego što se pozove funkcija iz modula *crud*, prikazaće mu se kategorije i on treba da izabere onu kategoriju kojoj ta knjiga pripadala. Funkcija koja realizuje dodavanje knjige u bazu je *dodavanje_knjige* (***listing 2.6.9***)

```
def dodavanje_knjige():
    global korisnik
    nova_knjiga = dict()
    nova_knjiga['naziv'] = input("Unesite naziv knjige: ")
    nova_knjiga['pisac'] = input("Unesite pisca: ")
    nova_knjiga['godina_izdanja'] = eval(input("Unesite godinu izdanja: "))
    nova_knjiga['poeni'] = eval(input("Unesite vrednost poena za izdavanje: "))
    nova_knjiga['primerci'] = eval(input("Unesite broj primeraka: "))
    nova_knjiga['opis'] = input("Unesite kratak opis: ")
    nova_knjiga['cena'] = eval(input("Unesite cenu knjige: "))
    kategorije = crud.sve_kategorije()
    okej = False
    kategorija = None
    while not okej:
        i = 1
        print("Sve kategorije: ")
        for k in kategorije:
            print("{}) {}".format(i, k[1]))
            i+=1
        kat = eval(input("Izaberite kategoriju kojoj knjiga pripada---> "))
        try:
            kategorija = kategorije[kat-1]
            okej = True
        except:
            print("Izaberite jednu od ponuđenih kategorija!")
    if crud.dodavanje_nove_knjige(nova_knjiga, kategorija[0]):
        print("Uspešno ste dodali novu knjigu!")
    else:
        print("Knjiga već postoji u bazi podataka!\nOtkazuje se dodavanje.")
    meni(korisnik)
```

Listing 2.6.9

Funkcija *prikaz_izdatih_knjiga* (***listing 2.6.10***) omogućava radniku da na stubičastom grafiku vidi koliko je svaki korisnik iznajmio knjiga. Ova funkcija poziva funkciju *izdate_chart* (***listing 2.6.11***) iz modula *crud* kako bi se iz baze podataka isčitale tražene vrednosti i prikazale na grafiku.

Napomena: kada se grafik iscrta, biće **aktivan 15 sekundi**, nakon čega treba ugasiti taj prozor da bi se program mogao nastaviti (važi za sve grafike u ovom programu).

```
def prikaz_izdatih_knjiga():
    global korisnik

    recnik = crud.izdate_chart()
    korisnici_tmp = crud.lista_korisnika()
    korisnici_imena = map(lambda x: x[3]+"\n"+x[4], korisnici_tmp)
    x = list(korisnici_imena)
    vrednosti = map(lambda k: ( recnik[k[0]] if k[0] in recnik.keys()\
                                else 0), korisnici_tmp)

    y = list(vrednosti)
    plt.bar(x, y)
    plt.xlabel('Korisnici')
    plt.ylabel('Broj izdatih knjiga')
    plt.yticks(arange(0, max(y)+1, 1.0))
    plt.pause(15)
    meni(korisnik)
```

Listing 2.6.10

```
def izdate_chart():
    sql = "select distinct Korisnik_idKorisnik from IzdataKnjiga"
    kursor.execute(sql)
    korisnici = kursor.fetchall()
    recnik = dict()
    for k in korisnici:
        sql = "select count(Knjiga_idKnjiga) from IzdataKnjiga \
                    where Korisnik_idKorisnik like %s"
        kursor.execute(sql, k)
        broj_za_k = kursor.fetchall()[0]
        recnik[k[0]] = broj_za_k[0]
    return recnik
```

Listing 2.6.11

Funkcija *prikaz_najomiljenijih_knjiga* (***listing 2.6.13***) iscrtava pitasti grafik na kom se nalazi 5 knjiga koje su najviše dodavane u omiljene knjige od strane regitrovanih korisnika. Pomoćna funkcija koja služi da se na pitastom grafu prikaže broj knjiga, a ne njihov procenat je *napravi_autopct* (***listing 2.6.12***). Ova funkcija poziva funkciju *omiljene_chart* (***listing 2.6.14***) iz modula *crud* kako bi se isčitale tražene vrednosti.

```
def napravi_autopct(vrednosti):
    def moj_autopct(pct):
        ukupno = sum(vrednosti)
        vrednost = int(round(pct*ukupno/100.0))
        return '{p:.1f}%({v:d})'.format(p=pct,v=vrednost)
    return moj_autopct
```

Listing 2.6.12

```
def prikaz_najomiljenijih_knjiga():
    global korisnik
    recnik = crud.omiljene_chart()

    sortirani_recnik = dict()
    sortirani = sorted(recnik, key=recnik.get, reverse = True)
    sortirani = sortirani[:5]
    for k in sortirani:
        sortirani_recnik[k] = recnik[k]

    nazivi_knjiga = []
    for k in sortirani_recnik:
        pom = crud.nadji_knjigu(k)
        nazivi_knjiga.append(pom[1])

    x = nazivi_knjiga
    y = list(sortirani_recnik.values())
    fig1, ax1 = plt.subplots()
    ax1.pie(y, labels = x, autopct=napravi_autopct(y))

    plt.pause(15)
    meni(korisnik)
```

Listing 2.6.13

```
def omiljene_chart():
    try:
        kursor.execute("select distinct Knjiga_idKnjiga from OmiljenaKnjiga")
        knjige = kursor.fetchall()
    except Exception as e:
        print("Greska u pribavljanju omiljenih knjiga: {}".format(e))
    recnik = dict()
    if knjige:
        for k in knjige:
            try:
                sql = "select count(Korisnik_idKorisnik) from OmiljenaKnjiga \
                          where Knjiga_idKnjiga like %s"
                kursor.execute(sql, k)
                broj_korisnika_knjige = kursor.fetchall()[0]
                recnik[k[0]] = broj_korisnika_knjige[0]
            except Exception as e:
                print("Exception u brojanju korisnika: {}".format(e))
    return recnik
```

Listing 2.6.14

Funkcija *prihod_knjiga* (***listing 2.6.15***) prikazuje linijski grafik sa tačkicama na kom se nalazi 5 najprodavanijih knjiga i ukupan prihod od njihove prodaje. Ova funkcija poziva funkciju *najprodavanije_chart* (***listing 2.6.16***) iz modula *crud* kako bi se iz baze isčitale tražene vrednosti.

```
def prihod_knjiga():
    global korisnik
    recnik = crud.najprodavanije_chart()
    sortirani_recnik = dict()
    sortirani = sorted(recnik, key=recnik.get, reverse = True)
    sortirani = sortirani[:5]
    for k in sortirani:
        sortirani_recnik[k] = recnik[k]
    nazivi_knjiga = []
    for k in sortirani_recnik:
        pom = crud.nadji_knjigu(k)
        nazivi_knjiga.append(pom[1])
    x = list(nazivi_knjiga)
    y = list(sortirani_recnik.values())
    c = list(zip(x,y))
    random.shuffle(c)
    x, y = zip(*c)
    plt.scatter(x, y)
    plt.plot(x,y)
    plt.xlabel("Nazivi knjiga")
    plt.ylabel("Prihod po prodatim knjigama")
    plt.yticks(arange(500, max(y)+100, 100.0))
    plt.ylim(ymin=500, ymax=2000)
    plt.pause(15)
    meni(korisnik)
```

Listing 2.6.15

```
def najprodavanije_chart():
    try:
        kursor.execute('select distinct Knjiga_idKnjiga from Narudzbenica')
        knjige = kursor.fetchall()
    except Exception as e:
        print("Greska u pribavljanju knjiga iz narudzbenice: {}".format(e))
    recnik = dict()
    if knjige:
        for k in knjige:
            try:
                sql = "select sum(cena) from Narudzbenica where Knjiga_idKnjiga = %s"
                kursor.execute(sql, k)
                suma_prihoda = kursor.fetchall()[0]
                recnik[k[0]] = int(suma_prihoda[0])
            except Exception as e:
                print("Exeption u sumiranju prihoda: {}".format(e))
    return recnik
```

Listing 2.6.16

3. Zaključak

U ovom projektu je prikazan program koji simuliran rad jedne izdavačke kuće. Ovaj program se može koristiti u realnom životu za potrebe izdavačkih kuća i takođe je, po potrebi, veoma lako proširiv dodatnim funkcionalnostima.

4. Literatura

1. Zvanični python sajt - https://www.python.org/
2. Python tutorial W3Schools - https://www.w3schools.com/python/
3. Programski jezik Python - https://en.wikipedia.org/wiki/Python_(programming_language)
4. MySQL Connector Developer Guide - https://dev.mysql.com/doc/connector-python/en/
5. Slajdovi sa predavanja i snimljene vežbe sa predmeta Seminarski rad A – Skript jezici