Razvoj web aplikacije korištenjem Python Bottle web okvira

Miloš, Klara

SVEUČILIŠTE U SPLITU
PRIRODOSLOVNO MATEMATIČKI FAKULTET

ZAVRŠNI RAD

RAZVOJ WEB APLIKACIJE KORIŠTENJEM PYTHON BOTTLE WEB OKVIRA

Klara Miloš

Split, rujan 2020.

Temeljna dokumentacijska kartica

Završni rad

Sveučilište u Splitu
Prirodoslovno-matematički fakultet
Odjel za Informatiku
Ruđera Boškovića 33, 21000 Split, Hrvatska

RAZVOJ WEB APLIKACIJE KORIŠTENJEM PYTHON BOTTLE WEB OKVIRA

Klara Miloš

SAŽETAK

U ovom završnom radu, kao uvod, objašnjen je pojam i razlika web aplikacije i web stranice. Dio rada posvećen je i opisu samog Bottle okvira kao i SQLite bazi podataka koja je korištena. Nakon nekoliko riječi o okviru prikazana je instalacija i početak rada u okviru razvojem jednostavne aplikacije. Osnove rada u SQLite sustavu prikazane su naredbama brisanja, ažuriranja i dodavanja podataka. Nakon teorijskog dijela o dotičnom okviru i bazi, opisan je postupak izrade aplikacije praktičnog dijela završnog rada. Pojašnjen je način na koji je aplikacija izrađena kao i demonstriran rad web aplikacije.

Ključne riječi: web aplikacija, Python, web okvir, Bottle, SQLite

Rad je pohranjen u knjižnici Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Rad sadrži: 41 stranica, 18 grafičkih prikaza, 1 tablica i 16 literaturnih navoda. Izvornik je na hrvatskom jeziku.

Mentor: **Dr. sc. Monika Mladenović,** *asistent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Ocjenjivači: **Dr. sc. Monika Mladenović,** *asistent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Dr. sc. Tonći Dadić, *viši predavač Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Dr. sc. Marko Rosić, *redoviti profesor u trajnom zvanju Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Rad prihvaćen: **rujan 2020.**

Basic documentation card

Thesis

University of Split
Faculty of Science
Department of Computer Science
Ruđera Boškovića 33, 21000 Split, Croatia

DEVELOPING WEB APPLICATION USING PYTHON BOTTLE WEB FRAMEWORK

Klara Miloš

ABSTRACT

In this thesis, as an introduction, the concept and difference between a web application and a website is explained. Part of the thesis is dedicated to the description of the Bottle framework itself as well as the SQLite database. After a few words about the framework, we show installation process and development of a simple application as well. The basics of working in the SQLite system are shown by the commands to delete, update and add data. After the theoretical part about the respective framework and database, the procedure of making the application of the practical part of the final work is described. The way the application is made is explained as well as the running the application for demonstration.

Keywords: web application, Python, web framework, Bottle, SQLite

Thesis deposited in library of Faculty of science, University of Split

Thesis consists of: 41 pages, 18 figures, 1 table and 16 references. Original language: Croatian

Mentor: **Monika Mladenović, Ph.D.** *Assistant Professor of Faculty of Science, University of Split*

Reviewers: **Monika Mladenović, Ph.D.** *Assistant Professor of Faculty of Science, University of Split*

Tonći Dadić, Ph.D. *Senior lecturer of Faculty of Science, University of Split*

Marko Rosić, Ph.D. *Full professor of Faculty of Science, University of Split*

Thesis accepted: September 2020.

Sadržaj

Uvod

Razvoj prvih web aplikacija krenuo je 1990.-ih godina. Popularnost interneta uzrokovala je veću potražnju za web aplikacijama, a kako je razvoj interneta pratio i razvoj računalnog sklopovlja (engl. *hardware*), tako je rasla i kompleksnost web aplikacija. Većina web aplikacija dijele neke zajedničke funkcionalnosti, npr. komunikacija s bazom podataka, upravljanje sesijom, ispis HTML-a. Kako programeri ne bi sa svakim novim projektom ponovo razvijali navedene funkcionalnosti, njihove apstrakcije su izdvojene u biblioteke. Kolekcija biblioteka koje zajedno čine jednu cjelinu naziva se aplikacijskim okvirom (engl. *web framework*). Aplikacijski okviri ubrzavaju proces izrade web aplikacija. Danas je u većini programskih jezika napisan barem jedan aplikacijski okvir za razvoj web aplikacija. Jedan od njih je Pythonov Bottle te su njegove značajke opisane u ovom radu. Pohrana, zapisivanje, pamćenje podataka je uvijek bila bitna stvar, tako i danas u modernom i informatički okruženom svijetu. Još krajem 20. i početkom 21. st., tijekom tzv. doba internet boom-a pojavljivalo se sve više i više web stranica i aplikacija koje su sadržavale velik broj informacija. Relevantne informacije je bilo potrebno, logično i strukturirano, organizirati pa se time pojavila velika potražnja za sustavima koji upravljaju bazom podataka. Jedan takav sustav za upravljanje bazom podataka je SQLite.

Ovaj završni rad je podijeljen u nekoliko poglavlja u kojima će se predstaviti spomenuti Bottle okvir i SQLite relacijski sustav, a u posljednjem dijelu će se prikazati izrada web aplikacije koristeći gore navedene elemente.

1. Korištene tehnologije za izradu web aplikacije

U ovom dijelu rada ćemo opisati tehnologije koje smo koristili pri izradi web aplikacije te što je to web aplikacija. Za izradu aplikacije korišten je Pythonov okvir Bottle unutar kojeg je implementirana MVC arhitektura za organizaciju aplikacije i odvajanje podataka od prikaza kao i kontrola.

1.1. Web aplikacije

Web aplikacija je aplikacijski odnosno računalni program koji koristi web preglednike i web tehnologiju za izvršavanje zadataka putem Interneta. Aplikacija se pohranjuje na udaljeni poslužitelj i isporučuje putem Interneta putem sučelja preglednika (Stackpath 2020). Prema Search Software Quality članku Web usluge su po definiciji web aplikacije i mnoge, iako ne sve web stranice, sadrže web aplikacije. Web aplikacije mogu biti dizajnirane za široku upotrebu i mogu ih koristiti svi; od organizacije do pojedinca iz brojnih razloga. Uobičajene web aplikacije mogu obuhvaćati web poštu, mrežne kalkulatore ili online trgovine.

Što se tiče tehničke implementacije, sve web aplikacije su, prije svega, statičke ili dinamične. Pojam 'statičan' dolazi od web aplikacija kojima nedostaje fleksibilnosti. Statičke web aplikacije imaju svoje stranice generirane od strane poslužitelja i nude malo ili ništa interaktivnost (nema JavaScript koda)(Dzone.com, 2020). Obično nema mjesta za personalizaciju i svaka moguća promjena djeluje tek nakon ponovnog učitavanja stranice. Statičke web stranice često je teško održavati i prekomjerna količina podataka koje šalju i primaju stvara rizike od loše izvedbe. Ne treba reći da nisu pogodne za mobilno okruženje. Ipak, statička web aplikacija može biti ispravan izbor kada se dijele vrlo sažete informacije i interakcija nije potrebna. Bilo koja dinamička web-aplikacija temelji se na okviru (engl. *framework*) – softveru web aplikacije koji kontrolira izgradnju web-stranica i olakšava održavanje. Način na koji se takve web aplikacije prikazuju na zaslonu korisnika nije unaprijed određen nego dinamički oblikovan logikom aplikacija koja se implementira na strani poslužitelja ili klijentske strane aplikacije.

1.1.1. Rad web aplikacije

Web aplikacije ne treba preuzeti kao ni instalirati budući da im se pristupa putem mreže. Korisnici mogu pristupiti web aplikaciji putem web preglednika kao što su Google Chrome, Mozilla Firefox ili Safari.

Da bi web-aplikacija funkcionirala, potreban joj je web-poslužitelj (engl. *web server*), poslužitelj aplikacija (engl. *application server*) i baza podataka. Web-poslužitelji upravljaju zahtjevima koji dolaze od klijenta, dok poslužitelj aplikacije izvrši traženi zadatak. Baza podataka može se koristiti za pohranu bilo koje potrebne informacije.

Web aplikacije obično imaju kratke razvojne cikluse i mogu se napraviti s malim razvojnim timovima. Većina web-aplikacija napisana je u JavaScript-u , HTML-u ili CSS-u. Programiranje na klijentskoj strani obično koristi ove jezike, koji pomažu izgraditi *front-end*[1] aplikacije. Programiranje na strani poslužitelja se radi za stvaranje skripti koje će koristiti web-aplikacija. Jezici kao što su Python, Java i Ruby obično se koriste za programiranje na strani poslužitelja.

Evo kako izgleda tipični tijek web aplikacije (Stackpath, 2020):

1. Korisnik putem interneta pokreće zahtjev za web poslužiteljem putem web preglednika ili korisničkog sučelja aplikacije.
2. Web poslužitelj prosljeđuje ovaj zahtjev odgovarajućem poslužitelju web aplikacija.
3. Poslužitelj web aplikacije izvodi traženi zadatak - poput upita baze podataka ili obrade podataka - zatim generira rezultate traženih podataka.
4. Poslužitelj web aplikacije šalje rezultate web poslužitelju s traženim informacijama ili obrađenim podacima.
5. Web poslužitelj odgovara klijentu natrag s traženim informacijama koje se zatim pojavljuju na korisnikovom zaslonu.

[1] Praksa pretvaranja podataka u grafičko sučelje, upotrebom HTML-a, CSS-a i JavaScript-a, tako da korisnici mogu pregledati i komunicirati s tim podacima.

Web aplikacije imaju mnogo različitih koristi, a s tim koristima, dolazi mnoge potencijalne prednosti. Neke uobičajene prednosti web aplikacija uključuju (Stackpath, 2020)(Search Software Quality, 2020):

- Web aplikacije mogu se pokrenuti na više platformi bez obzira na OS ili uređaj sve dok je preglednik kompatibilan.
- Može se pristupiti putem više preglednika.
- Svi korisnici pristupaju istoj verziji, uklanjajući sve probleme s kompatibilnošću.
- Web aplikacije nisu instalirane na tvrdom disku, čime se eliminiraju ograničenja prostora.
- Smanjuju troškove poslovanja i krajnjeg korisnika jer postoji manja podrška i održavanje nego klasične aplikacije.
- Web-aplikacijama se može pristupiti putem različitih uređaja platformi kao što su stolno računalo, prijenosno računalo ili mobilni uređaj.

1.1.2. Web aplikacija i web stranica

PARAMETAR	WEB APLIKACIJA	WEB STRANICA
Stvoreno za	Web aplikacija dizajnirana je za interakciju s krajnjim korisnikom	Web stranica se uglavnom sastoji od statičkog sadržaja. Javno je dostupan svim posjetiteljima.
Interakcija korisnika	U web aplikaciji korisnik ne samo da čita sadržaj stranice već i manipulira ograničenim podacima.	Web stranica pruža vizualni i tekstualni sadržaj koji korisnik može pregledavati i čitati, ali ne utječe na njegovo funkcioniranje.
Autentifikacija	Web aplikacijama je potrebna autentifikacija, jer nude mnogo širi opseg opcija od web stranica.	Autentifikacija nije obavezna za informativne web stranice. Korisnik može zatražiti registraciju za redovito ažuriranje ili pristup dodatnim opcijama. Ova značajka nije dostupna za neregistrirane posjetitelje web mjesta.
Zadatak i složenost	Funkcije web aplikacija prilično su veće i složenije u usporedbi s web stranicama.	Web stranica prikazuje prikupljene podatke i informacije na određenoj stranici.

Vrsta softvera	Razvoj web aplikacija dio je web stranice. To nije cjelovita web stranica.	Web stranica je cjelovit proizvod kojem pristupate uz pomoć preglednika.
Kompilacija	Web mjesto mora biti prethodno kompajlirano prije pokretanja.	Web mjesto nije potrebno unaprijed kompajlirati.
Implementacije	Sve promjene zahtijevaju ponovno kompajliranje i pokretanje cijelog projekta.	Male promjene nikada ne zahtijevaju potpuno ponovno kompajliranje i pokretanje. Samo trebate ažurirati HTML kôd.

1.1 Razlika web aplikacije i web stranice (Guru99, 2020)

1.2. Python web okviri

Python je nesumnjivo jedan od najpopularnijih programskih jezika koje možete pronaći na tržištu i jedna je od najboljih opcija ako ste netko tko se želi baviti web razvojem. Kada je u pitanju razvoj web aplikacije, prema Techno Stacks članku iz 2020., Python će biti među prvim izborima na umu svakog programera. Dostupni su brojni vrhunski Python okviri koji vam mogu olakšati zadatak web razvoja. Postoji valjan razlog zašto većina programera pokazuje interes za Python, a to je jednostavnost ponuđenog programskog jezika. Iako je jednostavan, pokazao se i učinkovitim. Struktura podataka u Pythonu ima visoku razinu, a kada je povezana s dinamičkom semantikom i lako razumljivom sintaksom, to postaje opcija za, danas traženu organizaciju koda. Ovo je razlog zašto se razvio kao idealan jezik za brzu analizu podataka, web razvoj i skriptiranje. Jednostavnost ovog jezika ovdje treba posebno spomenuti. Zbog ove jednostavnosti postalo je prilično lako ponovno koristiti kod i modularno programiranje što skraćuje vrijeme i trošak programiranja. Međutim, postoji zamka. Python sam po sebi nema ugrađene nikakve značajke, a okviri su postali neizbježni za korištenje programerima.

Danas nam je u Pythonu dostupan širok raspon web okvira (Django, Pyramid, Web2py, Flash, Bottle…). Neki su dokazani favoriti u velikim ekosustavima i zajednicama. Drugi se ističu u slučajevima korištenja niša ili za određene vrste razvoja. Ipak, drugi su nadobudni s novim uvjerljivim razlozima koje treba razmotriti. Okviri su dizajnirani za rješavanje različitih problema i postizanje kompromisa kako bi bolje služili svojoj publici. Da svi imaju isti cilj, trebao bi nam samo jedan okvir.

Python okviri mogu se grubo podijeliti u tri grupe, full-stack okvire, mikrookvire (engl. *microframeworks*) i asinkrone okvire (engl. *asynchronous frameworks*).

- Full-stack okviri uglavnom su usmjereni na izgradnju većih aplikacija s potpunim značajkama i nude puno uobičajenih funkcionalnosti već ugrađenih. Ako želite brzo izgraditi nešto složeno, a da sami ne donosite važne odluke, Full-stack okvir dobar je izbor. Ovi okviri obično vam pružaju razumne zadane postavke za komunikaciju s bazama podataka, već zadanih pogleda tj. dizajna web stranica, upravljanje redovima čekanja, pozadinske poslove i druge uobičajene aspekte većih aplikacija (ScoutAPM, 2020).
- Mikrookviri uglavnom su usredotočeni na pružanje male jezgre funkcionalnosti i pozivaju programera da samostalno odluči koje će knjižnice i tehnologije dodati za druge funkcije. To ima za prednost što omogućuje puno veću kontrolu nad dizajnom aplikacije i može rezultirati boljim performansama aplikacije. Oni obično zahtijevaju od programera da odabere vlastiti sloj apstrakcije baze podataka i druge knjižnice. Mikrookvir može biti izvrstan izbor za manje, usko fokusirane aplikacije, razvoj API-ja ili programe u kojima je izvedba važnija (ScoutAPM, 2020).
- Asinkroni okviri su usmjereni na isporuku visoke razine performansi dopuštajući vrlo veliki broj istodobnih veza. Općenito, asinkroni okviri zahtijevaju više strogosti u stilu programiranja i imaju ograničeniji skup dodataka. Asinkroni okviri izvrsni su kada trebate pružiti određenu funkcionalnost na vrlo velikom nivou (ScoutAPM, 2020).

1.2.1. Bottle

Bottle je brz, jednostavan i lagan WSGI(engl. *Web Server Gateway Interface*)[2] mikro web Python okvir. Distribuira se kao jedan datotečni modul i nema drugih ovisnosti osim Python standardne knjižnice (Bottlepy, 2020). Ovaj okvir poznat je po minimalističkim značajkama. Primarna upotreba ovog okvira je izrada web API-ja. Prilično je male veličine; međutim, njegova veličina nema utjecaja na njegove performanse. To je doista jedna od najviših platformi koju ljudi preferiraju. Bottle omogućuje izvršavanje svake web

[2] Sučelje za pristup web poslužitelju jednostavna je konvencija pozivanja za web poslužitelje za prosljeđivanje zahtjeva na web aplikacije ili okvire napisane u programskom jeziku Python.

aplikacije u samo jednoj izvornoj datoteci, bez ovisnosti o bilo kojem drugom izvoru - postoji iznimka za Python Standard Library. Ovaj je okvir ujedno jednostavan i prilično brz. Dolazi s podrškom za adapter za mehanizme predložaka treće strane kao i WSGI / HTTP. Ukratko, Bottle je minimalistički i jednostavan okvir koji se može koristiti za izgradnju aplikacija za osobnu upotrebu. Idealan je i pomaže u otkrivanju organizacije mrežnih okvira, izrade prototipa i savršen je za izgradnju različitih vrsta RESTful (engl. *Representational state transfer*) usluga[3]. Najbolja stvar kod ove vrste Python okvira je što pomaže programerima da usko surađuju s hardverom.

Nekoliko značajki Bottle okvira (Bottlepy, 2020):

- Usmjeravanje: Zahtjevi za mapiranje poziva-funkcija s podrškom za statične i dinamične URL-ove.
- Predlošci: Ugrađeni mehanizam za predloške
- Uslužni programi: Prikladan pristup podacima obrazaca, prijenosima datoteka, kolačićima, zaglavljima i ostalim metapodacima povezanim s HTTP-om.
- Poslužitelj: Ugrađeni HTTP razvojni poslužitelj i podrška za paste, fapws3, bjoern, gae, cherrypy ili bilo koji drugi HTTP poslužitelj sposoban za WSGI.

Osim gore navedenog Bottle karakterizira minimalizam, laganost kako veličine izvorne datoteke tako i pisanja koda, te podržava dodatke za baze podataka.

1.2.1.1 Preuzimanje i instalacija

Instaliranje Bottle okvira prikazana je sljedećih nekoliko linija teksta.

1. Instalirati Python
2. Otvoriti Command prompt (konzolni prozor)
3. Instalirati pip s naredbom: `curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py`

[3] Prijenos predstavničkog stanja softverski je arhitektonski stil koji definira skup ograničenja koja će se koristiti za stvaranje web usluga. Web usluge koje su u skladu s arhitektonskim stilom REST, nazvane RESTful Web usluge, pružaju interoperabilnost između računalnih sustava na Internetu.

Ova naredba će preuzeti python skriptu za instalaciju pip alata

4. Pokrenuti instalacijsku pip skripte naredbom: `python get-pip.py`
5. Instalirati bottlepy server s naredbom: `pip install bottle`
6. Pokrenuti app.py i otvoriti http://localhost:8080/ u web pregledniku

Bottle podržava Python 2.7 i Python 3 dok ostale (starije) verzije ne podržava.

1.2.1.2 Početak: „Hello world“

Nakon instalacije možemo izraditi jednostavan primjer projekta s ispisom "Hello World":

```
from bottle import route, run

@route('/hello')
def hello():
    return "Hello World!"

run(host='localhost', port=8080, debug=True)
```

Pokrenite ovu skriptu ili je zalijepite u Python konzolu, a zatim u svoj preglednik upišite URL http://localhost:8080/hello. Dekorator `route()` veže dio teksta u zagradama na osnovni URL, u ovom slučaju http://localhost:8080. Kako je tekst unutar zagrade „/hello“ u ovom slučaju vezujemo /hello nastavak s funkcijom `hello()`. To se naziva ruta (otuda i naziv dekorator) i najvažniji je koncept ovog okvira. Možete definirati koliko god ruta želite. Kad god preglednik zatraži URL, poziva se pridružena funkcija i vraća se povratna vrijednost pregledniku - to je jednostavno. Pozivom `run()` funkcije u posljednjem retku, pokreće se ugrađeni razvojni poslužitelj. Izvodi se na portalu localhost i portu 8080 i služi zahtjevima dok korisnike ne pritisne [Control]-[C]. Ne zahtijeva nikakvo postavljanje i nevjerojatno je bezbolan način za pokretanje aplikacije za lokalne testove. Ovo je samo demonstracija osnovnog koncepta kako se aplikacije grade s odabranim okvirom.

1.2.1.3 Usmjeravanje zahtjeva i dinamične rute

U posljednjem smo poglavlju izgradili vrlo jednostavnu web aplikaciju sa samo jednom rutom. Naravno, aplikacija sa samo jednom rutom je poprilično dosadna i nemoguća pa ćemo pokazati zanimljivije i kompleksnije slaganje ruta. Donji primjer pokazuje dvije stvari: Na jedan povratni poziv možemo povezati više ruta, a URL-ovima se mogu dodati

zamjenski znakove, zapisani unutar <> zagrada i pristupiti im putem argumenata ključnih riječi.

```
@route('/')
@route('/hello/<name>')
def greet(name='Stranger'):
    return template('Pozdrav {{name}}, kako si ?', name=name)
```

Rute koje sadrže zamjenske znakove nazivaju se dinamičkim rutama (za razliku od statičkih ruta) i istodobno se podudaraju s više URL-ova. Jednostavni zamjenski znak sastoji se od imena zatvorenog u zagradama (npr. <ime>) i prihvaća jedan ili više znakova do slijedeće kose crte (/). Na primjer, ruta /hello/<name> prihvaća zahtjeve za /hello/Klara kao i /hello/Laura, ali ne i za /hello/hello ili /hello/gđica/Miloš. Svaki zamjenski znak pokriveni dio URL-a prosljeđuje kao argument ključne riječi uz povratni poziv zahtjeva. Možete ih odmah upotrijebiti i s lakoćom implementirati RESTful, lijepe i smislene URL-ove. Evo nekoliko primjera takvih URL-ova:

```
@route('/wiki/<pagename>')     #odgovara /wiki/Learning_Python
def show_wiki_page(pagename):
    ...
@route('/<action>/<user>')        # odgovara  /follow/defnull
def user_api(action, user):
    ...
```

Filteri se mogu koristiti za definiranje specifičnijih zamjenskih znakova i/ili transformiranje dijela URL-a prije nego što se proslijedi povratnom pozivu. Filtrirani zamjenski znak deklarira se kao `<ime: filter>` ili `<ime: filter: config>` (obavezno je postojanje „:“ znaka prije filtera). Sintaksa za izborni konfiguracijski dio ovisi o korištenom filtru. Sljedeći su filteri su oni najčešće korišteni i implementirani prema zadanim postavkama (Bottlepy, 2020):

- :int odgovara samo (potpisanim) znamenkama i pretvara vrijednost u cijeli broj.
- :float slično: int, ali za decimalne brojeve.
- :path se podudara sa svim znakovima, uključujući kosu crtu, i može se koristiti za podudaranje s više od jednog segmenta putanje.
- :re omogućuje da odredite prilagođeni regularni izraz u polju za konfiguriranje. Podudarna vrijednost se ne mijenja.

```
@route('/object/<id:int>')
```

```
def callback(id):
    assert isinstance(id, int)

@route('/show/<name:re:[a-z]+>')
def callback(name):
    assert name.isalpha()

@route('/static/<path:path>')
def callback(path):
    return static_file(path, ...)
```

1.2.1.4 HTTP Request metoda

HTTP protokol definira nekoliko metoda zahtjeva za različite zadatke. GET je zadana vrijednost za sve rute bez navođenja druge metode. Ove rute odgovarat će samo GET zahtjevima. Za rukovanje drugim metodama poput POST, PUT, DELETE ili PATCH, potrebno je dodati argument ključne riječi metode u dekorater route() ili upotrijebite jedan od pet alternativnih dekoratera: get(), post(), put(), delete() ili patch(). Metode GET i POST su najkorištenije, POST obično za slanje HTML obrasca.

```
from bottle import get, post, request

@get('/login') # or @route('/login')
def login():
    return '''
        <form action="/login" method="post">
            Username: <input name="username" type="text" />
            Password: <input name="password"
            type="password"/>
            <input value="Login" type="submit" />
        </form>
    '''

@post('/login') # or @route('/login', method='POST')
def do_login():
    username = request.forms.get('username')
    password = request.forms.get('password')
    if check_login(username, password):
        return "<p>Your login information was correct.</p>"
    else:
```

```
return "<p>Login failed.</p>"
```

U ovom je primjeru URL /login povezan s dva različita povratna poziva, jednim za GET zahtjeve i drugim za POST zahtjeve. Prvi korisniku prikazuje HTML obrazac. Drugi povratni poziv poziva se pri predaji obrasca i provjerava podatke za prijavu koje je korisnik unio u obrazac.

1.3. SQLite baza podataka

Spremanje podataka za kasniju upotrebu je centralan dio većine Pythonovih aplikacija. Kad god se mjere podaci u laboratoriju ili gradi Web aplikacija, potrebno je trajno spremiti informacije. Na primjer, ako želite analizirati rezultate nakon izvršenih eksperimenata, ili ako želite sačuvati listu registriranih korisnika na Web stranici te podatke je nužno pohraniti. Svaka aplikacija može zahtijevati različite strategije spremanja podataka. Potrebno je uzeti razne faktore, kao što je veličina generiranih podataka, koliko su podaci samoopisni, kako će se koristiti kasnije itd. U ovom dijeli rada opisati će se spremanje podataka u bazu podataka. Pri tome ćemo koristiti standardne Pythonove module. Python nativno podržava SQLite bazu podataka. SQLite je knjižnica na C jeziku koja implementira mali, brzi, samostalni, visoko pouzdani, cjeloviti, mehanizam baze podataka SQL. SQLite je najčešće korišten mehanizam baze podataka na svijetu. SQLite je ugrađen u sve mobilne telefone i većinu računala i dolazi u paketu s bezbroj drugih aplikacija koje ljudi svakodnevno koriste. Ovo je mala SQL baza podataka koja ne zahtjeva dodatni proces i cijela se nalazi u jednoj datoteci. Za rad s SQLite-om koristi se modul sqlite3 (Branko Žitko, 2020). Prvo je potrebno stvoriti konekciju na bazu kako bi se mogao koristiti sqlite3 modul.

```
import sqlite3
conn = sqlite3.connect("naziv_baze.db")
```

Sada se može stvoriti Cursor objekt koji prvenstveno služi za izvršavanje SQL upita pomoću .execute() metode.

```
cur = conn.cursor()
```

1.3.1. Definiranje tablica i relacija

Za opis rada spomenute baze napravit ćemo relacijsku bazu podataka koja će sadržavati podatke o studentu i studijskim grupama. U pravilu jedan student pripada jednoj studijskog grupi, dok grupama više studenata. Metodom `.executescript()` se pokreću SQL skripte. U donjem primjeru se stvaraju tablice i relacije za gornju relaciju.

```
cur.executescript("""
      CREATE TABLE grupe (
            id text PRIMARY KEY,
            naziv text NOT NULL);
```

```
        CREATE TABLE studenti (
              id integer PRIMARY KEY,
              ime text NOT NULL,
              prezime text NOT NULL,
              grupa_id text DEFAULT NULL,
        FOREIGN KEY (grupa_id) REFERENCES grupe (id));
"""
```

SQL naredba `CREATE TABLE` služi za stvaranje tablice. Često se ova skripta pokreće samo jednom, pošto će SQLite javiti grešku ako tablice postoje. Alternativa je korištenje `CREATE TABLE IF NOT EXISTS` sintakse koja će kreirati tablicu ako ne postoji. Slično, SQL naredba `DROP TABLE` sliži za brisanje tablice, `odnosno DROP TABLE IF EXISTS` za brisanje tablice ako postoji (Branko Žitko, 2020). Polja u SQLite tablici mogu biti:

- text - tekstualni tip
- integer - cijeli brojevi
- real - realni brojevi
- blob - binarni skup podataka
- NULL - null vrijednost

Ako ne želimo da polje ima NULL vrijednost onda ćemo u definiciji polja dopisati `not NULL`. Ako je polje ujedno i jedini primarni ključ, onda se piše `PRIMARY KEY`. U slučaju da više polja čine primarni ključ, onda ih je potrebno definirati na sljedeći način: PRIMARY KEY (polje1, polje2, ...). Strani ključevi se definiraju sljedećom sintaksom: FOREIGN KEY (polje1, polje2, ...) REFERENCES tablica (strano_polje1, strano_polje2, ...) (Branko Žitko, 2020).

1.3.2. Ubacivanje podataka u tablicu

Ako želimo ubaciti podatke u tablicu, koristit ćemo INSERT naredbu.

```
cur.execute("
      INSERT INTO grupe (id, naziv) VALUES (?, ?)", ( "MAT",
                        "Matematika"))
```

Praksa je da se u SQL naredbi koristi oznaka ? na mjesta gdje će se ubacivati vrijednosti. Na ovaj način će Python sam preformatirati Pythonove tipove podataka u SQL tipove podataka.

Ako želimo ubaciti više podataka odjednom, koristit ćemo istu SQL sintaksu, ali ćemo pozvati metodu `.executemany()`.

```
grupe = [("INF", "Informatika"),
      ("BIO", "Biologija"),
      ("KEM", "Kemija")]
cur.execute(" INSERT INTO grupe (id,naziv)VALUES
(?,?)",grupe)
conn.commit()
```

Na kraju je potrebno pozvati `.commit()` metodu konekcije, kako bi se sve transakcije trajno spremile u bazu podataka. Prilikom unosa, ako direktno ne postavimo vrijednost polja id-a, kako u gornjem primjeru, onda će se ono automatski generirati.

1.3.3. Promjena podataka u tablici

Za promjenu podataka se koristi `UPDATE` SQL naredba.

```
cur.execute("UPDATE studenti SET grupa_id = ? WHERE id = ?",
("INF", 2))
conn.commit()
```

1.3.4. Brisanje podataka iz tablice

Za promjenu podataka, odnosno uklanjanje, se koristi `DELETE` SQL naredba.

```
cur.execute("DELETE FROM studenti WHERE id = ?", (3, ))
conn.commit()
```

1.3.5. Dobavljanje podataka

SELECT SQL naredba služi za dohvaćanje podataka. Metoda `.fetchone()` služi za dohvaćanje jednog (prvog) zapisa, a ponovnim pozivanjem će vratiti sljedeći po redu zapis. U slučaju da nema više zapisa, vratit će None.

U donjem SQL upitu smo povezali tablicu studenti i tablicu grupe kako bi za svakog studenta dobili i naziv grupe a metoda `.fetchone()` će vratiti prvi zapis kao n-torku, odnosno(pr. 'Ante', 'Antić', 'Matematika'):

```
cur.execute("""
      SELECT studenti.ime,studenti.prezime,grupe.naziv
      FROM studenti
      JOIN grupe ON studenti.grupa_id = grupe.id
""")
print(cur.fetchone())
```

Ako želimo vratiti sve zapise kao listu, onda ćemo koristiti `.fetchall()` metodu.

```
cur.execute("""
      SELECT studenti.ime,studenti.prezime,grupe.naziv
      FROM studenti
      JOIN grupe ON studenti.grupa_id = grupe.id
""")
print(cur.fetchall())
```

Rezultat ovog upita je lista n-torki gdje svaka n-torka odgovara jednom zapisu(pr. [('Ante', 'Antić', 'Matematika'),] ('Dora', 'Dorić', 'Informatika'), …]

2. Razvoj aplikacije

U ovom dijelu rada opisati ćemo korake izrade jedne web aplikacije, e-Dnevnik. Ideja aplikacije je objediniti i iskoristiti metode i tehnologije opisane na prijašnjim stranicama ovog rada. Osnovna ideja je korištenje Python-ovog modula sqlite3 za spremanje podataka koje aplikacija koristi nakon pokretanja.

2.1. Analiza sustava e-Dnevnik

Ideja za praktični dio završnog rada bila je izrada e-Dnevnika, sustava koji omogućava kako učenicima/roditeljima tako i nastavnicima pristup sustavu. Korisnik se, u pravilu, u aplikaciju može prijaviti kao nastavnik, učenik ili kao administrator. Administrator ima ovlasti dodavanja, uklanjanja, ažuriranja nastavnika i učenika. Nastavnik ima uvid u razrede kojima predaje te može ocjenjivati, brisati i prepravljati ocjene učenika kojima predaje. Prilikom prijave kao učenik, dobiva se ispis podataka o učeniku te o njegovim ocjenama, grupirani po predmetu, ukupnom prosjeku i prosjeku svakog predmeta.

Aplikacija se sastoji od 3 glavna modula:

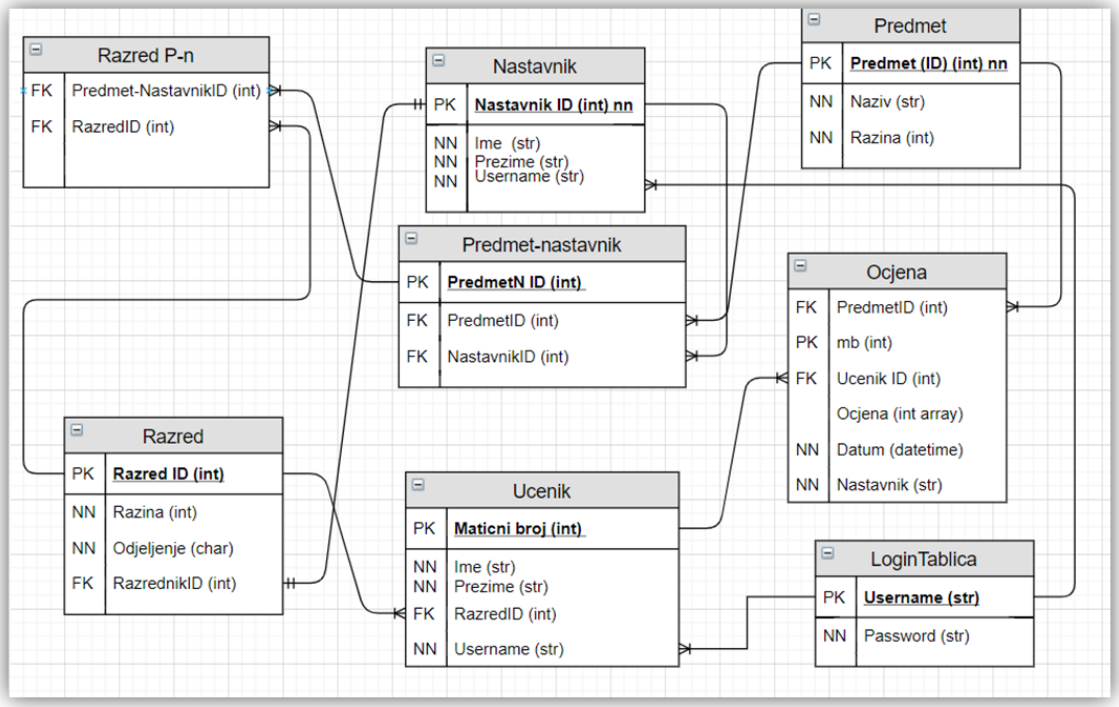
- Administrator
- Nastavnik
- Učenik

Svaki modul se sastoji od 3 Python datoteke: Control, Model i View. Control sadrži routove i logiku za rad aplikacije. Control tehnički nije klasa nego Python datoteka sastavljen od gomilu routova koji nas vode do željenih destinacija. Model dohvaća podatke koji su spremljeni u bazu i šalje pročitane podatke u View (koji vraća .html datoteke) i prikazuju ih korisniku. Html file-ovi vraćaju podatke Control-u koji vrši provjeru nad njima i izvodi daljnje potrebne operacije. Uz 3 glavna tu je login modul koji se služi, u jednu ruku, u svrhu zaštite podataka. Login datoteka nas nakon uspješne prijave prosljeđuje u jedan od tri Control datoteke.

2.2. Model podataka e-Dnevnik

Razvoj baze podataka započinje modeliranjem podataka koje rezultira konceptualnim modelom i ono nam predstavlja osnovni model baze koju ćemo kasnije izraditi u odabranim alatima. Konceptualni model je zapravo osnovna shema baze podataka, a prikazan je kao skica na papiru koja sadrži sve entitete, tj. tablice, atribute, tj. retke u tim tablicama, kao i veze među tablicama. Prije izrade konceptualnog modela se moramo zapitati što sve očekujemo od same baze, a zatim i pripadajuće aplikacije i voditi računa da bi model trebao biti cjelovit.

Prvi korak bila je izrada baze podataka. Kako to biva, ER model na papiru kasnije je implementiran u kod. Skica baze podataka napravljena je u Draw.io online aplikaciji.

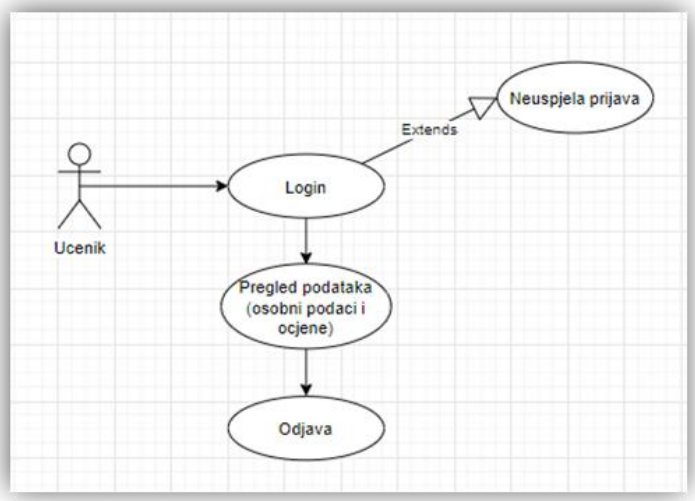


2.1 ER model baze podataka

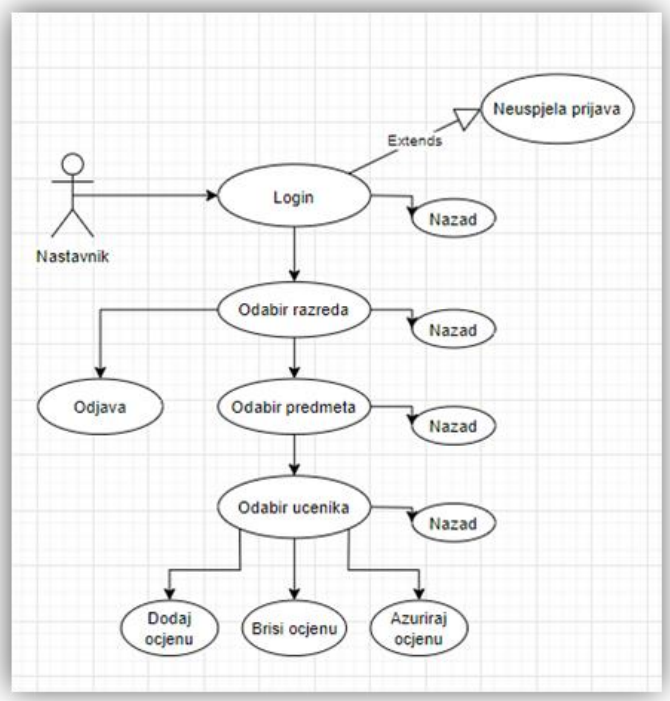
Osnovne tablice su Učenik i Nastavnik sa malim brojem atributa, svega dovoljan za razlikovanje entiteta. Jedan nastavnik predaje većem broju razreda kao i što razred ima veći broj nastavnika koji mu predaje. Kako je to relacija N:M trebalo je stvoriti novu tablicu Razred-Nastavnik sa primarnim ključevima obiju tablica. Isto tako kako nastavnik može i ne mora predavati jedan ili više predmeta (biologija i kemija) tako i za jedan predmet postoji veći broj nastavnika u školi. Kako nastaje N:M relacija opet stvara novu tablicu Predmet-Nastavnik po uzoru na gornji primjer. Jednu od glavnih uloga ima i tablica

učenik. Učenik može pripadati samo jednom razredu a razred ima veći broj učenika – klasična 1:N relacija. Isto tako učenik je vezan za nastavnika ali nema potreba za relacijom jer se preko razreda, a zatim predmeta predavanog tom razredu može doći do nastavnika. Ono što još karakterizira svakog učenika je ocjena. Učenik ima puno ocjena ali ocjena za taj jedan predmet i od tog jednog nastavnika pripada samo tom specifičnom učeniku. Za kraj ostaje još spomenuta Login forma koja prati tko ima pristup kojem dijelu sadržaja. Ona je povezana i sa Učenikom i sa Nastavnikom relacijom 1:N. Očito je da se u bazi ne spominje Administrator. Administrator u bazi podataka ima samo korisničko ime i lozinku - spremljeni u login formu, koja mu omogućuje pristup svom sadržaju.

Prije kodiranja bilo je praktično izraditi *Use-Case* Dijagram. Ovaj dijagram nam daje uvid u to tko ima koje ovlasti unutar aplikacije a prikazuju se grafički. Ovakav način rada danas je sve zastupljeniji a programerima omogućava organizaciju i raspodjelu zadaća i vremena provedenog na izvršavanju jedne funkcionalnosti dijagrama.

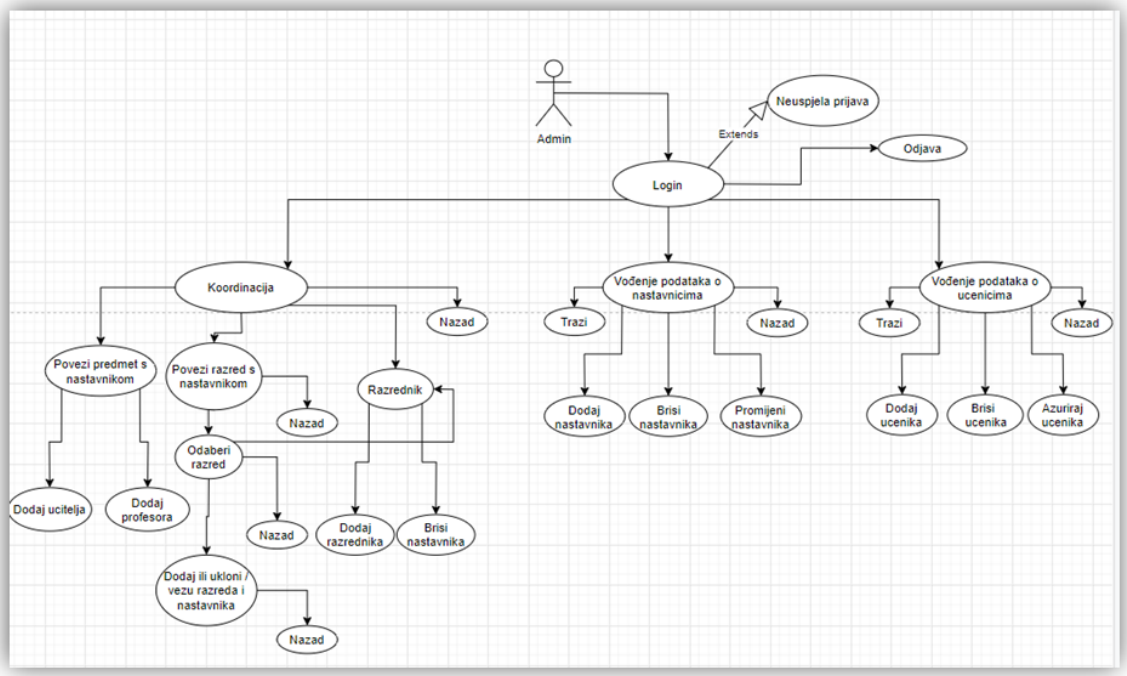


2.2 Use Case dijagram za učenika



2.3 Use Case dijagram za nastavnika

Admin ovlasti su, naravno sveobuhvatne pa zato ima najviše funkcionalnosti, upravo zato Use-Case Dijagram za Admin korisnika ujedno je i nakompleksniji. Na dole prikazanoj slici vidimo funkcionalnosti koje ima superkorisnik.



2.4 Use Case dijagram za Administratora

2.3. Izrada aplikacije

Nakon dobro postavljeih temelja moguće je krenuti sa programiranjem. Ako smo prijašnje korake napravili kako treba, slijedili dobro naučene prakse programiranje, u fazi izrade aplikacije ne bi trebalo naići na veće probleme. Prvi korak je uređenje login.py datoteke koju koristimo za pokretanje aplikacije. Ona mora imati spoj na bazu podataka, sve .html datoteke učitane i definirane rute koje će preusmjeriti na određeni sadržaj. Neke od definiranih ruta imaju, kao metodu, oznake POST što znači da se očekuje da dohvate određene podatke, dok ove druge po *defaultu* rade sa GET metodom. Glavna ruta je ona sa oznakom „/“ i prva se pokreće a u posljednjim redovima koda može se vidjeti i port na lokalnom računalu kojeg aplikacija koristi – 8083.

```
from bottle import route, redirect,
template,request,static_file,run
from adminModel import SQL_LoginModel
import adminControl, nastavnikControl,ucenikControl

dbl=SQL_LoginModel('baza2.db')

@route('/static/<filepath>')
def server_static(filepath):
    return static_file(filepath, root='./')

@route('/')
def login():
    adminControl.username_global=None
    nastavnikControl.username_global=None
    return template('administracija_html/naslovnica')

@route('/', method='POST')
def do_login():
    username = request.forms.get('username')
    password = request.forms.get('password')
    rb=request.forms.get('rb')
    vrsta={'A':0,'N':1,'U':2}
    vr=vrsta[rb]
    ispravan=dbl.login_chek(username,password,vr)
    if ispravan:
```

```
            un = username
            if rb=='A':
                adminControl.username_global=username
                return redirect('/izbornikAdmin')
            elif rb=='N':
                nastavnikControl.username_global=username
                return  redirect('nastavnik/'+un)
            else:
                ucenikControl.username_global=username
                return  redirect('/ucenik/'+un)

        else:
            return "<p>Login failed.""</p>"


if __name__=='__main__':
    run(reloader=True,port=8083)
```

Kod 1 Sadržaj Login.py datoteke

Sljedeći korak je osigurati rad jednom korisniku aplikacije. U ovom radu opisati će se kod napisan za rad korisnika Učenik dok će ostatak koda biti dan kao predložak i moći će se pogledati. Princip rada svih korisnika je u pravilu jednak samo što su neki jednostavniji dok drugi složeniji. Poštujući MVC arhitekturu kod je bilo potrebno podijeliti u 3 dijela. View služi kao predložak, odnosno web stranica koja će se učitat na akciju prijave (engl. *login*) učenika.

```
from bottle import  template

class UcenikView():
    @staticmethod def_osobni_ocjene( podaci, ocjene,
    prosjek_p_p, ukupan_prosjek):
    Return template('ucenik_html/Ucenik_naslovna',
    podaci=podaci,ocjene=ocjene,
    prosjek_p_p=prosjek_p_p,
    ukupan_prosjek=ukupan_prosjek)
```

Kod 2 Sadržaj UcenikView.py datoteke

Model označava bazu podataka gdje spremamo podatke o učeniku, učenik se može prijaviti za pregled ocjena ali samo Admin može dodati novog korisnika. Ova klasa kao glavnu metodu koristi `__init__`. Stvara se tablica i veze ako ne postoje. Osim ove

metode sadrži neke pomoćne za izračun prosjeka ili zaključne ocjene a za što su nam potrebni podaci zapisani u bazi. Kasnije ove metode pozivamo iz kontrolera u trenutku kada su nam potrebne.

```
import sqlite3

class UcenikModel():

    def __init__(self,filename):
        self.conn=sqlite3.Connection(filename)
        self.cur=self.conn.cursor()
        self.filename=filename
        self.cur.executescript("""
            CREATE TABLE IF NOT EXISTS Ucenik(
              mb integer INTEGER PRIMARY KEY,
              ime text NOT NULL,
              prezime text NOT NULL,
              razredID integer NOT NULL,
              username text NOT NULL,
              FOREIGN KEY (razredID) REFERENCES Razred (mb));

            CREATE TABLE IF NOT EXISTS Ocjena (
             mb integer PRIMARY KEY,
             ocjena integer,
             datum DATETIME NOT NULL,
             predmetID integer NOT NULL,
             ucenikID integer NOT NULL,
             nastavnik text NOT NULL,
             FOREIGN KEY (predmetID) REFERENCES Predmet (mb),
             FOREIGN KEY (ucenikID) REFERENCES
             Ucenik(mb));""")

    def zakljuci_ocjenu(self,ocjena):
        if ocjena<1.5:
            return 1
        elif ocjena<2.5:
            return 2
        elif ocjena<3.5:
            return 3
        elif ocjena<4.5:
            return 4
```

```
        else:
            return 5

def ucenik_select_podaci(self,mb):
      self.cur.execute("""SELECT u.ime, u.prezime, u.mb,
       u.username, r.razina, r.odjeljenje FROM Ucenik u INNER
      JOIN Razred r ON u.razredID=r.mb WHERE  u.mb= ?
      """,(mb,))
            return self.cur.fetchall()

def ispis_ocjene_ucenik(self,mb):
             podaci=self.ucenik_select_podaci(mb)[0]
       self.cur.execute("""SELECT
            ocjena,o.datum,o.nastavnik,p.naziv FROM Ocjena o
            INNER JOIN Ucenik u ON u.mb=o.ucenikID  INNER
            JOIN Predmet p ON p.mb=o.predmetID WHERE u.mb= ?
            ORDER BY o.predmetID, o.datum""",
            mb,))
       ocjene=self.cur.fetchall()
       if(len(ocjene)==0):
             return (podaci,{},{},None)
             ocjene_dict={}
             for i in ocjene:
                  if i[3]  not in ocjene_dict:
                      ocjene_dict[i[3]]=[i[:-1]]
                  else:
                      ocjene_dict[i[3]].append(i[:3])
              prosjek_p_p={k: self.prosjek_po_predmetu(k,mb)
      for k in ocjene_dict.keys()}

      ukupni_prosjek=round(sum(self.zakljuci_ocjenu(i) for i
      in prosjek_p_p.values()) /len(prosjek_p_p),2)
             return
      (podaci,ocjene_dict,prosjek_p_p,ukupni_prosjek)


  def prosjek_po_predmetu(self,kljuc,mb):
      self.cur.execute("""SELECT AVG(ocjena) FROM Ocjena o
            INNER JOIN Predmet p ON p.mb=o.predmetID where
            p.naziv=? and o.ucenikID=? ORDER BY datum """,
                        (kljuc, mb))
```

```
        prosjek=self.cur.fetchall()
        return round(prosjek[0][0],2)
```

Kod 3 Sadržaj UcenikModel.py datoteke

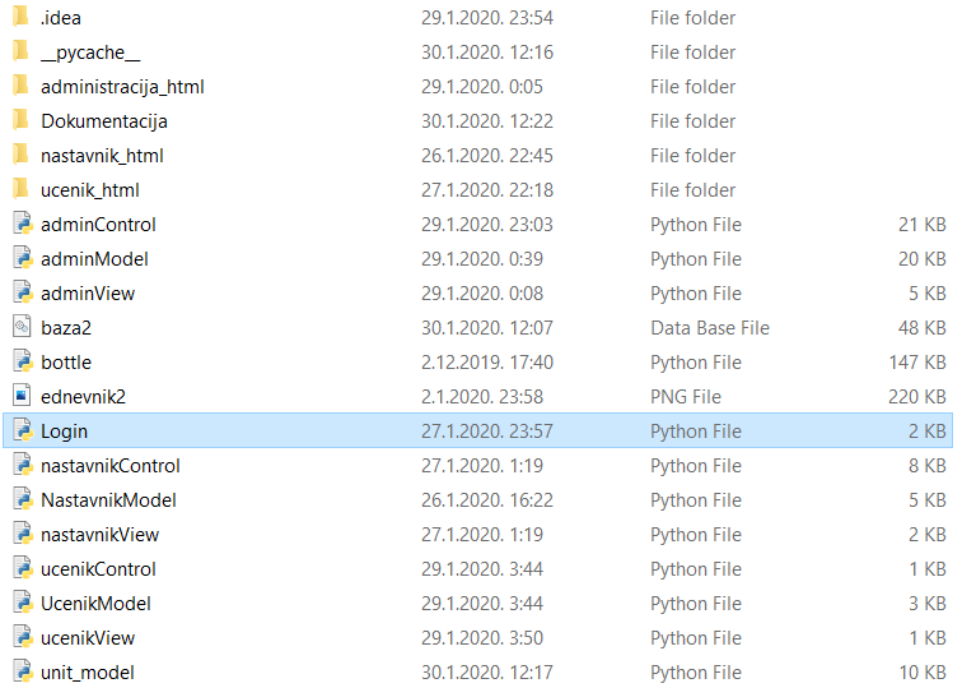
Posljednji korak je pisanje koda za Controler – logika čitave aplikacije. Controler ima reference na relevantni predložak i Model Učenika. Ako je prijava bila neuspješna prikazuje nam se predložak za *Zabranjen pristup*, inače učitava se web stranica sa podacima dohvaćenim iz baze podataka koji se ispisuju korisniku na zaslon.

```
from bottle import route, request, redirect
from UcenikModel import UcenikModel
from ucenikView import  UcenikView

dbu=UcenikModel('baza2.db')
username_global=None
un=None
ui=UcenikView()

def neispravanIdentitet():
    return '<h1> Zabranjen pristup prijavite se: <a
href="/">ovdje</a></h1>'

# OCJENE popis
@route('/ucenik/<username>')
def ucenik_izbornik_ocjene(username):
    global username_global
    if (username_global is None):
       return neispravanIdentitet()
    maticni_ucenika=int(username[:-1])
    podaci,ocjene_rijecnik,prosjek_po_predmetu,
    ukupan_prosjek=dbu.ispis_ocjene_ucenik(maticni_ucenika)
    return
      ui.o_osobni_ocjene(podaci=podaci,ocjene=ocjene_rijecnik
      ,prosjek_p_p=prosjek_po_predmetu,ukupan_prosjek=ukupan_
      prosjek)
```

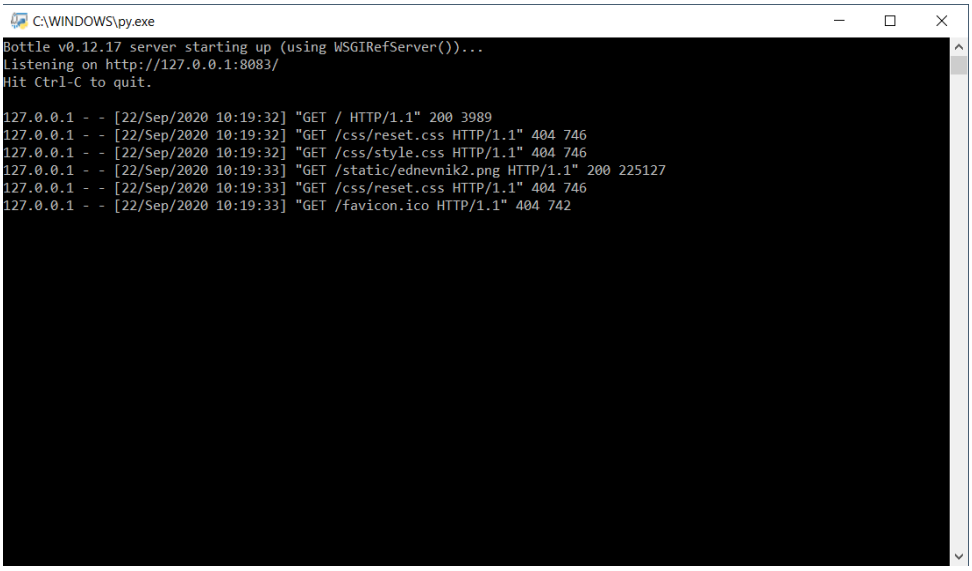
Kod 4 Sadržaj UcenikControler.py datoteke

2.4. Rad aplikacije

Aplikaciju pokrećemo tako da dvaput kliknemo na Login.py datoteku. Što pokreće Command Prompt. Bottle se pokrene, te koristeći WSGIRef Server dobijemo informaciju na kojem portu se osluškuje naša aplikacija. Još je samo potrebno pokrenuti odabrani web preglednik te upisati rutu prikazana na donjoj slici.

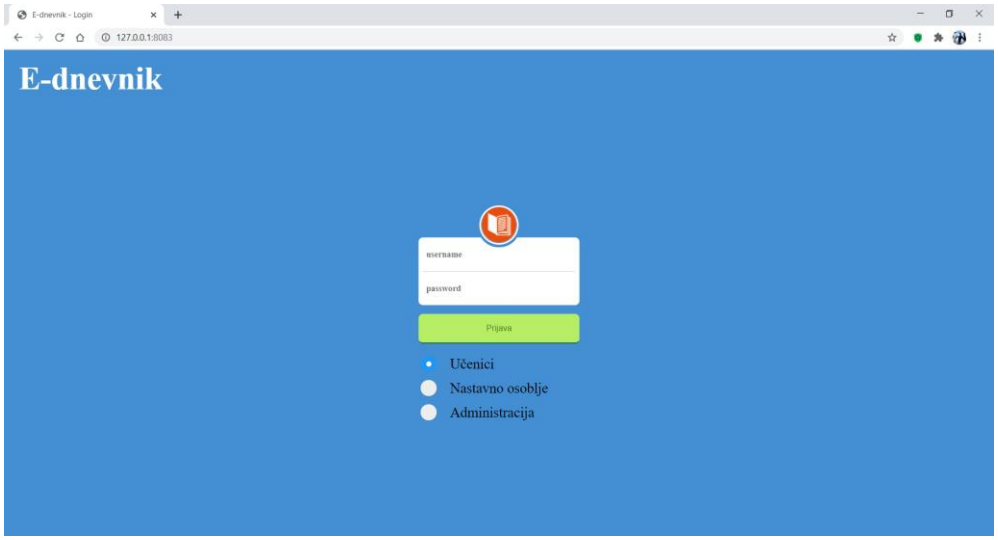


Slika 2.1 Login.py datoteka za pokretanje aplikacije



Slika 2.2 Command Prompt sa informaciji o adresi i portu aplikacije

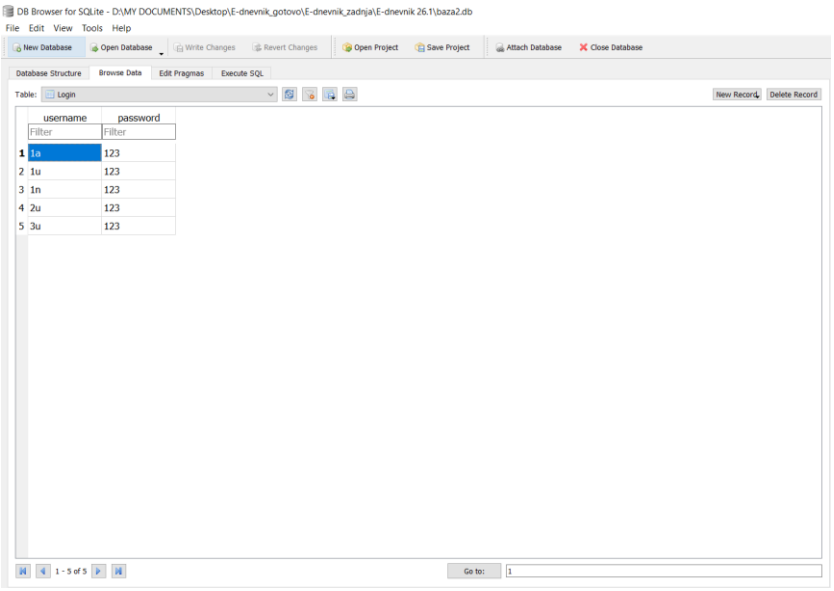
Početna stranica aplikacije koja zahtjeva prijavu za daljnje korištenje. Korisnik bira svoju ulogu i prijavljuje se. Važno je da se `username` i `password` poklapa sa ulogom koju odaberemo inače će javiti grešku. Ukoliko je prijava neuspješna dobivamo prikaz Login failed.



Slika 2.3 Početna stranica aplikacije e-Dnevnik

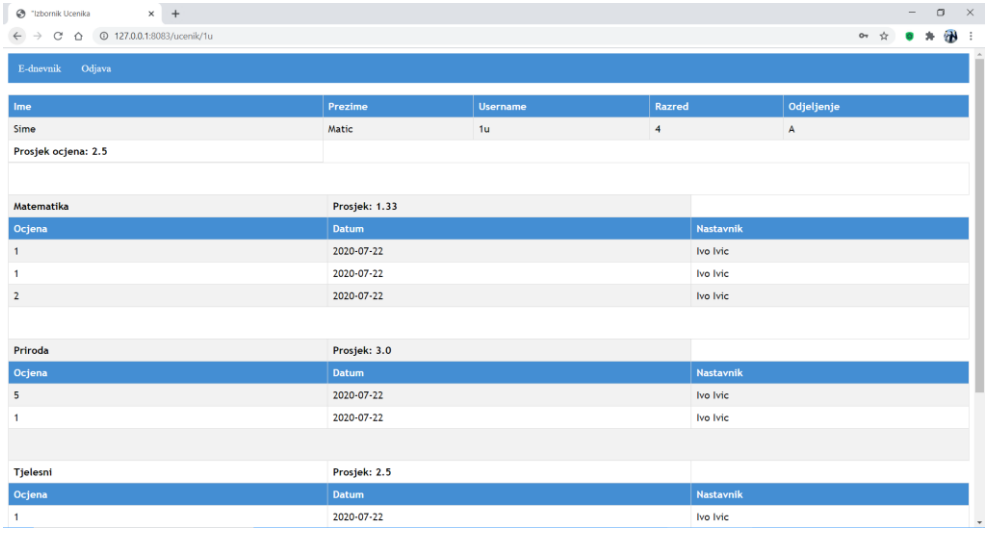
Slika 2.4 Neuspjela prijava

Neuspjela prijava znači da se dogodio jedan od dva moguća scenarija. Ili su nam lozinka i korisničko ime netočni ili smo ih upisali točno ali svoju ulogu nismo točno označili. Samo admin ima ovlasti dodavanja novih korisnika (to će biti prikazano za nekoliko paragrafa i slika) i tada im određuje korisničko ime, lozinku i naravno povezuje to s ulogom - učenik ili nastavnik. Na Slika 2.5 vidljivi su podaci koji se trenutno nalaze u bazi. Korisničko ime ima dva znaka, drugi znak označava o kojem se korisniku radi a-Admin, u-Učenik, n-Nastavnik.



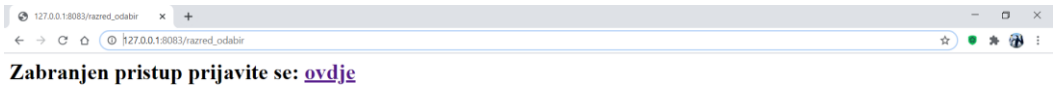
Slika 2.5 Korisnici i njihove lozinke u bazi podataka

Prvo ćemo prikazati rad aplikacije iz uloge Učenika. Kako učenik ima najmanje ovlasti njegov „dio“ aplikacije je ujedno i najjednostavniji. Nakon uspješne prijave prijavljuje se pogled prikazan na Slika 2.6. U samome vrhu, prvo su prikazani osnovni podaci korisnika – ime, prezime, korisničko ime, razred koji učenik pohađa i odjeljenje kojem pripada. Nakon osnovnog, učenik može pregledati koje predmetne u školi trenutno sluša i koje ocjene iz tog predmeta ima. Kako profesori za predmete najčešće nisu isti svim odijeljenima, pored ocjene piše i ime nastavnika koji je zapisao tu ocjenu. U našem slučaju imamo jednog nastavnika koji svima predaje sve predmete ali unosom novim nastavnika od strane Admina imali bi mogućnost odabira i drugih nastavnika ukoliko predaju taj predmet.



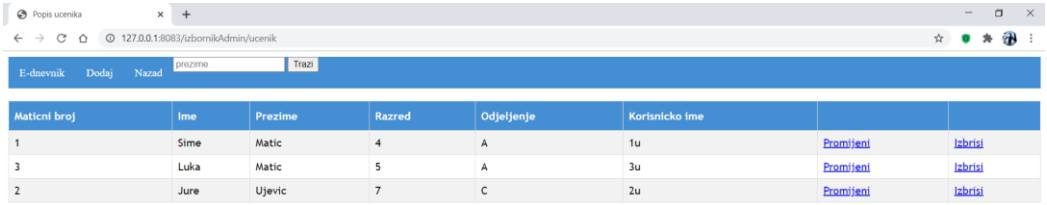
Slika 2.6 Učenički prikaz e-Dnevnik aplikacije

Ukoliko učenik pokuša pristupiti informacija koje mu nisu dostupne prikazuje se pogled na slici ispod. Ukoliko stvarno želimo pristupiti tom sadržaju nudimo mu mogućnost prijave kao admin korisnik.



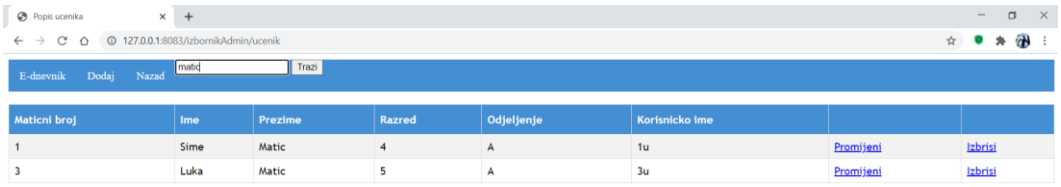
Slika 2.7 Zabranjen pristup sadržaju

Admin je osoba koja ima sve neograničene ovlasti u aplikaciji. Kao što je već spomenuto upisuje nove nastavnike i učenike, radi promjene na već upisanim korisnicima ili ih uklanja iz sustava. Sljedećih nekoliko slika prikazuju funkcionalnosti koje su u ovlasti admina. Na samom početku prikazuje se izbornik sa lijeve strane u kojem odaberemo ono što nas zanima. Za početak prikazujemo popis svih učenika u našem sustavu.



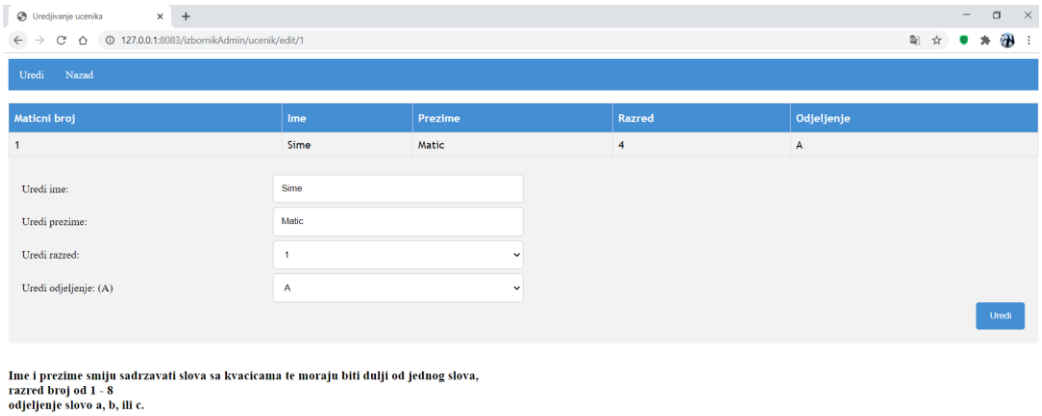
Slika 2.8 Lista svih učenika u sustavu e-Dnevnika

Ukoliko se dohodi da imamo veliki broj učenika, naravno u stvarnosti uistinu jest tako, potrebno je moći na brz način pronaći traženog učenika. Iz spomenutog razloga može se koristiti tražilica odnosno filter za filtriranje podataka.



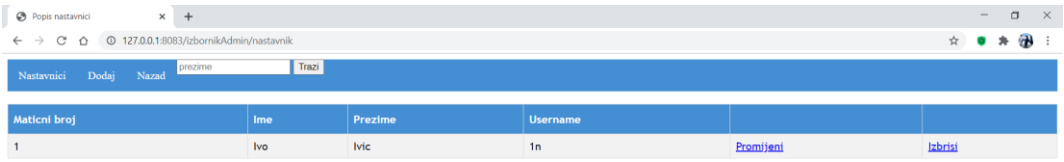
Slika 2.9 Filtriranje podataka

Slika 2.10 prikazuje kako se vrši ažuriranje podataka za odabranog učenika. Posebna pažnja je posvećena validaciji podataka, ukoliko podaci nisu ispravni na samom dnu forme sustav javi greške koje je potrebno ispraviti. Kada smo izmijenili podatke potrebno je potvrditi unos, a izmjene je moguće vidjeti odmah povratkom na prethodni pogled (popis svih učenika).



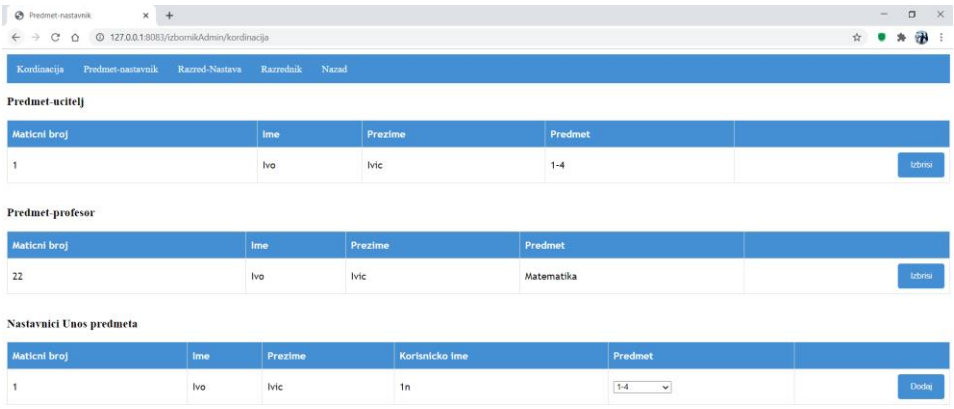
Slika 2.10 Ažuriranje podataka odabranog učenika

Osim rada sa učenicima, admin radi i sa nastavnicima. Funkcionalnosti su gotovo iste. Dodavanje novih nastavnika i povezivanje tih istih sa nekim predmetom kao i brisanje nastavnika iz sustava prikazani su na slici Slika 2.11.

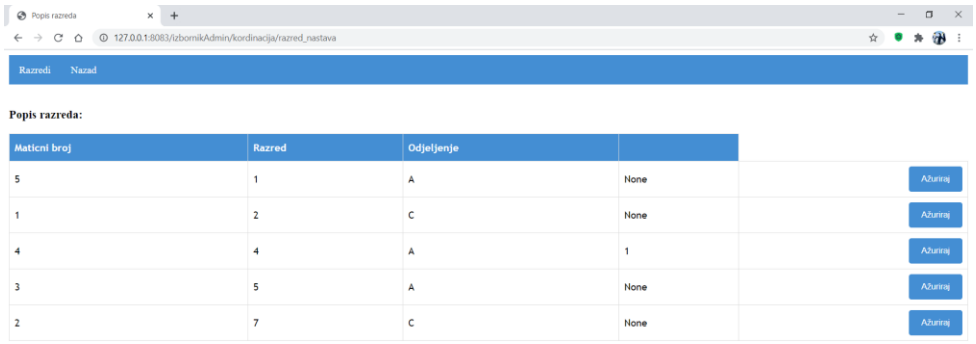


Slika 2.11 Prikaz nastavnika u sustavu e-Dnevnik

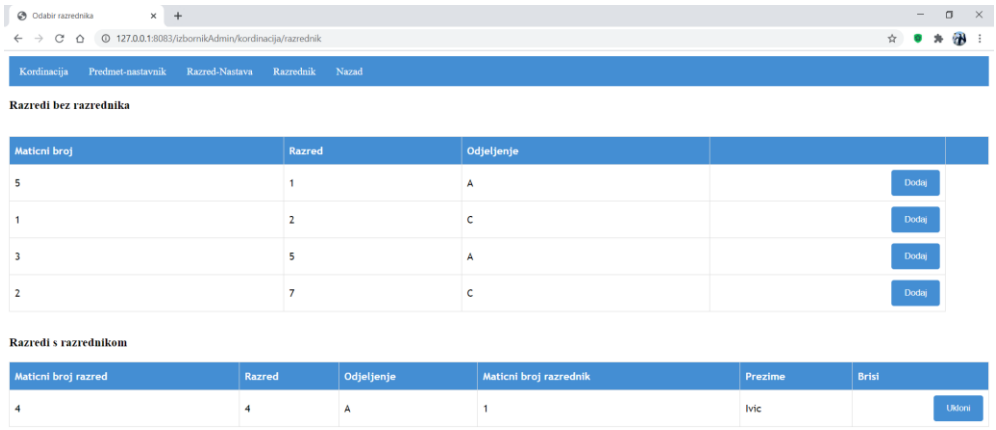
Posljednje ćemo prikazati koordinaciju nastavnika i učenika. Povezivanje razreda sa nastavnikom, pogledavanje i uređivanje koji nastavnik predaje kojem razredu i koji razred nema dodijeljenog nastavnika su problemi koje administrator mora biti u mogućnosti riješiti.



Slika 2.12 Početni prikaz admin koordinacije



Slika 2.13 Prikaz popisa razreda iz uloge Administratora



Slika 2.14 Informacije o postojanju dodijeljenog nastavnika razredu

Zaključak

Razvoj internet tehnologije doveo je do rasprostranjenog korištenja web aplikacija kako tvrtke prelaze s tradicionalnih modela na cloud-based i/ili grid modele. Web-aplikacije tvrtkama pružaju mogućnost da pojednostave svoje poslovanje, povećaju učinkovitost i smanje troškove. Prije početka izrade aplikacije važno je dobro isplanirati izradu aplikacije ako i baze podataka. Potrebno je definirati sve zadatke koje aplikacija mora obavljati jer se time mogu otkriti mnogi potencijalni problemi i stvoriti rani koncept aplikacije. Izrada aplikacije počinje s izradom baze podataka. Izrada baze je bitan korak jer je baza „srce“ aplikacije, dok je ostatak aplikacije samo sučelje za bazu. Loše dizajnirana baza može stvoriti mnogo problema tijekom kasnije izrade aplikacije jer i mala izmjena u bazi može prouzročiti velike prepravke u kodu. Dizajn baze podataka može doslovno značiti razliku između uspjeha i neuspjeha. Usporedba se može povući s temeljima za kuću. Ako je temelj čvrst, možda imate dobru kuću, ovisno o naknadnim radovima. Međutim, manjkav temelj uništit će kuću, gotovo bez obzira na posao koji slijedi. Isto tako, uspjeh ili neuspjeh aplikacije ovisi o kvaliteti baze podataka. Kod samog pisanja koda važno je pravilno organizirati kod u razrede jer se time bitno dobiva na modularnosti te se lakše dodaju novi dijelovi aplikacije i prepravljaju postojeći. Posljednji dio izrade aplikacije je testiranje aplikacije koje služi kako bi se ispravili nedostaci u kodu. Sigurnost aplikacije ne smije se zanemarivati jer bez sigurnosti korisnici mogu lako namjerno ili nenamjerno srušiti aplikaciju. Sav korisnički unos se mora provjeravati prije daljnje obrade na serveru i unosa u bazu podataka. Imajući sve to na umu, možemo zaključiti da je za proces izrade kako ove tako i bilo koje druge web aplikacije, bitno dobro planiranje i razrada. Time se dosta rano može stvoriti nacrt po kojemu se kasnije radi, a čime se izbjegavaju mnogi problemi u kasnijim fazama.

Literatura

[1] StackPath, https://blog.stackpath.com/web-application/. 23.09.2020.

[2] SearchSoftwareQuality, *What Is Web Application (Web Apps) and Its Benefits* https://searchsoftwarequality.techtarget.com/definition/Web-application-Web-app. 23.09.2020.

[3] Guru99, *Difference between Website and Web Application* https://www.guru99.com/difference-web-application-website.html. 23.09.2020.

[4] Dzone.Com, *Types of Web Applications: From a Static Web Page to a Progressive Web App - DZone Web Dev* https://dzone.com/articles/types-of-webapplications-from-a-static-web-page-t. 23.09.2020.

[5] Scout APM Blog, *The Most Popular Python Web Frameworks in 2020* https://scoutapm.com/blog/the-most-popular-python-web-frameworks-in-2020. 23.09.2020.

[6] Technostacks Infotech Pvt. Ltd., *Top 10 Python Frameworks in 2020 For Web Developers* 23.09.2020., https://technostacks.com/blog/python-web-frameworks.

[7] Bottle 0.13-dev documentation, *Tutorial Installation* https://bottlepy.org/docs/dev/tutorial.html#installation. 24.09.2020.

[8] Bottle 0.13-dev documentation, *Tutorial — Hello World* https://bottlepy.org/docs/dev/tutorial.html#quickstart-hello-world. 24.09.2020.

[9] Branko Žitko, *Uvod u programsko inženjerstvo* https://sites.google.com/site/brankozitko/kolegiji/upi. 22.09.2020.

Sažetak

U ovom završnom radu, kao uvod, objašnjen je pojam i razlika web aplikacije i web stranice. Dio rada posvećen je i opisu samog Bottle okvira kao i SQLite bazi podataka koja je korištena. Nakon nekoliko riječi o okviru prikazana je instalacija i početak rada u okviru razvojem jednostavne aplikacije. Osnove rada u SQLite sustavu prikazane su naredbama brisanja, ažuriranja i dodavanja podataka. Nakon teorijskog dijela o dotičnom okviru i bazi, opisan je postupak izrade aplikacije praktičnog dijela završnog rada. Pojašnjen je način na koji je aplikacija izrađena kao i demonstriran rad web aplikacije.

Ključne riječi: web aplikacija, Python, web okvir, Bottle, SQLite

Summary

In this thesis, as an introduction, the concept and difference between a web application and a website is explained. Part of the thesis is dedicated to the description of the Bottle framework itself as well as the SQLite database. After a few words about the framework, we show installation process and development of a simple application as well. The basics of working in the SQLite system are shown by the commands to delete, update and add data. After the theoretical part about the respective framework and database, the procedure of making the application of the practical part of the final work is described. The way the application is made is explained as well as the running the application for demonstration.

Keywords: web application, Python, web framework, Bottle, SQLite

Skraćenice

API	*Asynchronous Transfer Mode*	asinkroni način prijenosa
CSS	*Integrated Services Digital Network*	digitalna mreža integriranih usluga
HTML	*Hyper Text Markup Language*	prezentacijski jezik za izradu web stranica
HTTP	*Hyper Text Transfer Protocol*	hipertekstulani protokol transfera
OS	Operating System	o*perac*ijski sustav
URL	*Uniform Resource Locator*	usklađeni lokator sadržaja

Privitak

Instalacija programske podrške

Opisana I prikazana unutar samog sada

Upute za korištenje programske podrške

Opisane i prikazane unutar samog rada