# Korištenje Pyramid Python okvira za izradu web aplikacije

**Lovrinčević, Mara**

**Undergraduate thesis / Završni rad**

**2021**

*Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj:* **University of Split, University of Split, Faculty of science / Sveučilište u Splitu, Prirodoslovno-matematički fakultet**

*Permanent link / Trajna poveznica:* https://urn.nsk.hr/urn:nbn:hr:166:772086

*Rights / Prava:* In copyright / Zaštićeno autorskim pravom.

*Download date / Datum preuzimanja:* **2024-03-29**

*Repository / Repozitorij:*

Repository of Faculty of Science

SVEUČILIŠTE U SPLITU

**PRIRODOSLOVNO MATEMATIČKI FAKULTET**

ZAVRŠNI RAD

# KORIŠTENJE PYRAMID PYTHON OKVIRA ZA IZRADU WEB APLIKACIJE

Mara Lovrinčević

Split, rujan 2021

# Temeljna dokumentacijska kartica

Završni rad

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

# KORIŠTENJE PYRAMID PYTHON OKVIRA ZA IZRADU WEB APLIKACIJE

Mara Lovrinčević

## SAŽETAK

U ovom završnom radu prvotno su objašnjeni pojmovi kao što su web aplikacija i web okvir. Također i sam opis Pyramid Python okvira i SQLite baze podataka koji su korišteni za izradu ovog rada. Nakon kratkog uvoda o Pyramid okviru prikazan je jedan od načina instalacije okvira i brzi početak rada u samom okviru razvojem jednostavne aplikacije. Nakon teorijskog dijela, opisan je postupak izrade aplikacije praktičnog dijela završnog rada. U radu možete vidjeti i način na koji se u SQLite bazi podataka mogu dodavati, brisati i ažurirati podaci, te na samom kraju kratki postupak izrade aplikacije.

**Ključne riječi**: Python, web okvir, web aplikacija, Pyramid web okvir, SQLite

**Rad sadrži:** 41 stranice, 22 grafička prikaza i 10 literaturnih navoda. Izvornik je na hrvatskom jeziku.

**Mentor:** **izv. prof. dr. sc. Ani Grubišić**, *izvanredni profesor Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

**Ocjenjivači:** **izv. prof. dr. sc. Ani Grubišić,** *izvanredni profesor Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*
**izv. prof. dr. mag. Branko Žitko,** *izvanredni profesor Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*
**Ines Šarić-Grgić, mag. ing. comp.,** *asistent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Rad prihvaćen: **rujan 2021.**

# Basic documentation card

Thesis

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

# USING THE PYRAMID PYTHON WEB FRAMEWORK FOR DEVELOPING WEB APPLICATION

Mara Lovrinčević

## ABSTRACT

In this thesis, concepts such as web application and web framework are initially explained. Also the very description of the Pyramid Python framework and SQLite database used to create this paper. After a short introduction about the Pyramid frame, one of the ways to install the frame and a quick start in the frame itself is shown by developing a simple application. After the theoretical part, the procedure of making the application of the practical part of the thesis is described. In this paper, you can also see the way in which data can be added, deleted and updated in the SQLite database, and at the very end, a short process of creating an web application.

**Keywords**: Python, web framework, web application, Pyramid web framework, SQLite

**Thesis consists of:** 41 pages, 22 figures and 10 references. Original language: Croatian

**Mentor:** **Ani Grubišić, PhD**, *Associate Professor, Faculty of Science, University of Split*

**Reviewers:** **Ani Grubišić, PhD**, *Associate Professor, Faculty of Science, University of Split*
**Branko Žitko, PhD**, *Associate Professor, Faculty of Science, University of Split*
**Ines Šarić-Grgić, mag.ing.comp.** *Assistant, Faculty of Science, University of Split*

Thesis accepted: **September 2021.**

# Sadržaj


1 Uvod ..... 1

2 Web aplikacije ..... 2

2.1 Web okviri ..... 3

2.2 Predlošci ..... 3

2.2.1 Python predlošci ..... 3

2.2.1.1 Jinja ..... 4

2.3 Rad web aplikacija ..... 5

3 Web okviri programskog jezika Python ..... 6

3.1 Django ..... 7

3.2 Bottle ..... 7

3.3 Web2py ..... 7

3.4 Tornado ..... 8

4 Pyramid ..... 9

4.1.1 Brzi početak ..... 9

4.1.2 Konfiguracija ..... 11

4.1.3 Razvijajte se interaktivno ..... 11

4.1.4 Započnite s malim, završite veliko i ostanite završeni ..... 11

4.1.4.1 Započnite s malim ..... 11

4.1.4.2 Završite veliko ..... 12

4.1.4.3 Ostanite završeni ..... 12

5 SQLite baza podataka ..... 13

5.1 Definiranje tablica i relacija ..... 14

5.2 Brisanje i promjena podataka iz tablice ..... 15

5.3 Dobavljanje podataka ..... 15

5.4 Ubacivanje podataka u tablicu ..... 16

6 Razvoj aplikacije ..... 17

6.1 Izrada aplikacije ..... 18
6.2 Pokretanje aplikacije ..... 22
6.3 Rad aplikacije ..... 23
7 Zaključak ..... 31
8 Literatura ..... 32

# 1 Uvod

U posljednjem desetljeću događa se veliki napredak u razvoju web aplikacija koje su dostupne korisnicima putem WWW-a (engl. World Wide Web, skraćeno Web), iako je njihov razvoj započeo 1990.-ih godina. Kako je potražnja za web aplikacijama rasla tako je rasla i njihova složenost. Zbog velike potražnje za web aplikacijama, bilo je potrebno osigurati brži način same izrade. Kraće vrijeme izrade web aplikacije nam omogućuje aplikacijski okvir (engl. *web framework*). Aplikacijski okvir je skup biblioteka koje sadrže funkcionalnosti kao što su npr. komunikacija sa bazom podataka, HTML. Navedene funkcionalnosti i još mnoge druge se nalaze kod većine web aplikacija, te je to najveći razlog uvođenja biblioteka. U današnjim programskim jezicima za razvoj web aplikacije implementiran je minimalno jedan aplikacijski okvir. U Python programskom jeziku ima mnogo implementiranih okvira, kao što su Flask, Pyramid, Bottle i mnogi drugi, koji će biti opisani u ovom radu. Jedna od spomenutih funkcionalnosti je komunikacija sa bazom podataka, što je jako bitna funkcionalnost jer sama pohrana i pamćenje podataka je od ključne važnosti za bilo koju web aplikaciju. Stoga ih je potrebno strukturno i logično organizirati, za što nam je potreban sustav za upravljanjem same baze podataka. Jedan od takvih sustava je SQLite, koji je također opisan u radu.

U ovom završnom radu se nalazi nekoliko poglavlja koji predstavljaju ponajviše Pyramid okvir, uključujući još neke Python okvire, u kojem je izrađena web aplikacija koja je prikazana u posljednjem dijelu rada.

# 2 Web aplikacije

Web aplikacija je aplikacijski softver koji za razliku od računalnih softverskih programa radi na web poslužitelju. Korisnik web-aplikacijama pristupa putem web-preglednika s aktivnom internetskom vezom. Iako razvoj web aplikacija slijedi proces razvoja softvera, tehnologija i arhitektura koja se za to koriste prilično su različite. Softverska aplikacija koja radi na osobnom računalu možda ne ovisi o internetu za razliku od web aplikacije koja ovisi o udaljenim poslužiteljima (Devndra Ghimire [1]).

Moramo napomenuti da web aplikacije mogu biti statičke ili dinamičke. Statičke web aplikacije su vrsta web stranica, kodirane u običnom HTML-u (Hypertext Markup Language). Zahtijevaju skriptiranje samo na strani klijenta, kao što su HTML i CSS (Cascading Style Sheets). Teško ih je održavati, te prekomjerna količina podataka stvara rizike od loše izvedbe. Iako nisu pogodne za mobilna okruženja, statička web aplikacija je možda najbolji izbor ukoliko se dijele sažete količine informacija. Dinamične web stranice temeljene na okviru (engl framework). Karakteriziraju stalno mijenjanje sadržaja kako bi učinio stranicu funkcionalnijom, korisnik uz čitanje može i komunicirati. To zahtijeva više nego samo skriptiranje na strani klijenta, potreban nam je poslužitelj preko kojeg korištenjem skriptiranih jezika kao što je JavaScript kako bi odradili taj dio web stranice. (weblogographic [2])

Aplikacije su podijeljene u „slojeve“ koji imaju svoje uloge. Većina aplikacija imaju samo jedan sloj koji se nalazi na klijentu, dok web aplikacije se gledaju na n-slojnom pristupu. Najčešće je to troslojni oblik. Prvi sloj je web preglednik zatim dinamički sadržaj implementiran u nekom programskom jeziku (npr. Python, JavaScript, itd.) i zadnji sloj su baze podataka. Web preglednik šalje zahtjeve srednjem sloju koji ih servisira postavljanjem upita i ažuriranja prema bazi podataka i generira korisničko sučelje. U ovom radu ćemo proći kroz sve dijelove web aplikacije. Za dinamički sadržaj ćemo koristiti web okvir Pythona, a za bazu podataka SQLite, o kojima ćemo malo detaljnije u nastavku.

## 2.1 Web okviri

Web okvir je biblioteka kodova koja čini brži i lakši razvoj weba pružajući uobičajene obrasce za izgradnju pouzdanih i održivih web aplikacija. Web okvir softverski je okvir koji je dizajniran za podršku razvoj web aplikacija, uključujući web usluge, web resurse i web API (engl. Application Programming Interface). Web okviri nam omogućavaju brži razvoj aplikacija, ali se moramo zapitati, zašto su nam web okviri korisni? Web okviri sažimaju ono što su programeri naučili u posljednjih dvadeset godina prilikom programiranja web stranica i aplikacija za web. Okviri olakšavaju ponovnu uporabu koda za uobičajene HTTP (Hyper Text Transfer Protocol) operacije i strukturiranje projekata tako da drugi programeri koji poznaju okvir mogu brzo izgraditi i održavati aplikaciju. Web okviri su koncepti implementirani od strane Djanga , Flaska , Bottla , Pyramida i nekoliko drugih biblioteka. Iako su najpoznatiji web okviri danas Django i Flask odlučili smo se obraditi Pyramid web okvir.

## 2.2 Predlošci

Predlošci (engl. Tamplate) se obično koriste kao posrednički format koji su napisali programeri za programsku izradu jednog ili više željenih izlaznih formata, obično HTML, XML ili PDF. Omogućuju programerima da generiraju željene vrste sadržaja, poput HTML -a, dok koriste neke podatke i konstrukcije programiranja, poput uvjeta, i za petlje za manipulaciju izlazom. Datoteke predložaka koje su stvorili programeri, a zatim ih obrađuje stroj za predloške, sastoje se od unaprijed napisanih oznaka i blokova oznaka predloška u koje se ubacuju podaci. U tipičnoj WSGI aplikaciji , predložak bi generirao HTML izlazni odgovor kada dođe HTTP zahtjev za određeni URL.

### 2.2.1 Python predlošci

Postoji nekoliko popularnih Python predložaka. Implementacija stroja za predloške bit će negdje na spektru između dopuštanja izvođenja proizvoljnog koda i davanja samo ograničenog skupa mogućnosti putem oznaka predloška.

Neki od Python predložaka:

- Django predložak
- Jinja
- Chameleon je HTML/XML predložak

- Diazo
- Airspeed

Gore navedeni predlošci su neki od najpoznatijih ali sa potrebe ovoga rada ćemo detaljnije o Jinji jer ima konzistentnu sintaksu i projekt je čisto izdvojen kao neovisan projekt otvorenog koda.

#### 2.2.1.1 Jinja

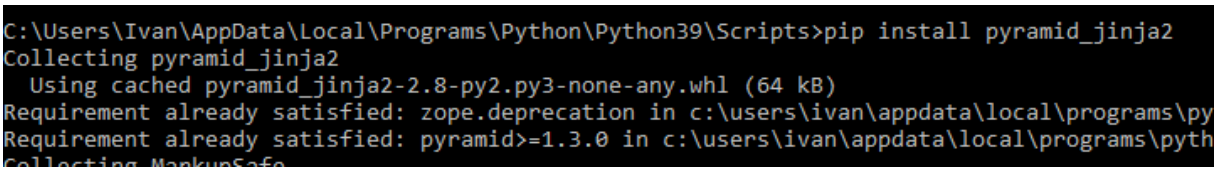
Jinja, koja se također naziva " Jinja2 " za specifikaciju najnovije verzije izdanja, motor je predloška Python koji se koristi za stvaranje HTML, XML ili drugih formata označavanja koji se vraćaju korisniku putem HTTP odgovora. Jinja2 je koristan jer ima dosljednu sintaksu oznake predložaka, a projekt je čisto izdvojen kao neovisni projekt otvorenog koda pa ga druge knjižnice koda mogu koristiti kao ovisnost. Postiže promišljenu ravnotežu u spektru predložaka gdje na jedan kraj možete umetnuti proizvoljan kod u predloške, a na drugi kraj programer može kodirati što god želi. [3]

Jinja je brz, izražajan i proširiv predložak. Posebna rezervirana mjesta u predlošku omogućuju pisanje koda sličnog Python sintaksi. Zatim se predlošku prosljeđuju podaci za iscrtavanje konačnog dokumenta.

Jinja uključuje:

- Nasljeđivanje predloška i uključivanje.
- HTML predlošci mogu koristiti automatsko preslikavanje
- Okruženje u sandboxu može sigurno iscrtati nepouzdane predloške.
- Podrška za sinkronizaciju za generiranje predložaka koji automatski obrađuju funkcije
- Predlošci se sastavljaju u optimizirani Python kod pravovremeno i spremaju u predmemoriju ili se mogu sastaviti unaprijed.
- Iznimke ukazuju na ispravan redak u predlošcima kako bi se olakšalo ispravljanje pogrešaka.
- Proširivi filtri, testovi, funkcije, pa čak i sintaksa.

Jinja se instalira na potpuno isti način kao i Pyramid. U CMD na lokaciji pip-a pozovemo metodu *pip install Jinja2,* sačekamo instalaciju i počnemo sa kodiranjem. Podržava Python 3.6 i novije verzije Pythona.




*Slika 1. Instalacija jinja2*

## 2.3 Rad web aplikacija

Kada govorimo o web aplikacijama govorimo o aplikacijama koje u napravljene da ih podržavaju svi web preglednici. Najčešće su to aplikacije razvijene u JavaScrip i HTML jezicima, budući da se ti jezici oslanjaju na preglednik za izvršavanje programa. U prošlom poglavlju su navedeni pojmovi dinamičke i statičke web stranice. Najveća razlika između takvih stranica je ta da statičke stranice ne zahtijevaju procesiranje koda na serveru, dok dinamičke jesu.

Iz razloga što web aplikacije rade na serveru, server upravlja sa zahtjevima i upitima. Kada korisnik zatraži nešto na stranici taj se upit šalje serveru gdje se izvršava bilo to sa samom bazom podataka ili sa aplikacijskim dijelom, te vraća rezultat korisniku na ekran.(Search Software Quality [4])

Web aplikacije su se kroz godine dosta mijenjale, rješavale su se mnoge mane koje su bile uočene, ali uz sve mane web aplikacije imaju svoje prednosti kao što su (StackPath [5]):

- Bez obzira na OS (operacijski sustav) web aplikacije se mogu pokrenuti na više platformi
- Može se pristupiti putem više preglednika, te svi korisnici pristupaju istoj verziji
- Ne instaliraju se čime se eliminiraju ograničenja prostora.
- Možemo im pristupiti sa bilo kojeg uređaja koji ima pristup internetu.
- Smanjuju troškove poslovanja postoji manja podrška i održavanje.

# 3 Web okviri programskog jezika Python

Možemo reći da je Python danas jedan od najpopularnijih programskih jezika te ima brojne vrhunske okvire koji olakšavaju zadatak web razvoja. Sama njegova jednostavnost i učinkovitost je razlog zašto većina programera pokazuje interes za Python. Sama struktura podataka u Pythonu igra veliku ulogu, a ukoliko se poveže sa lako razumljivom sintaksom, postaje jako tražen programski jezik za današnju organizaciju koda. Ovo je razlog zašto se razvio kao idealan jezik za web razvoj.

Django, Flask, Bottle su samo od nekih najpoznatijih Python web okviri. Svaki od web okvira imaju različite vrste razvoja i različite načine rada. Svaki od njih je dizajniran za rješavanje različitih problema jer da svi web okviri imaju isti cilj ne bi nam trebalo mnoštvo okvira.

Python okviri se dijele na asinkrone okvire (engl. *asynchronous frameworks*), full-stack okvire i mikrookvire (engl. *microframeworks*) (Scout APM [6]):

- Asinkroni okviri su vrlo visokih performansi i mogu dopustiti veliki broj istodobnih veza. Općenito, oni zahtijevaju više programskog stila i obično nisu toliko robusni kao tradicionalniji okviri.
- Full-stack okviri usmjereni su na stvaranje velikih, složenih aplikacija s punim mogućnostima. Obično pružaju razne unaprijed izgrađene aplikacije uz minimalan napor.
- Razvoj mikrookvira uglavnom je usmjeren na pružanje ograničenog skupa funkcionalnosti. Omogućuje programerima da odaberu vlastite knjižnice i tehnologije, a to može rezultirati boljim performansama.

Ukratko ćemo navesti i opisati nekoliko najpoznatijih web okvira, ali ćemo se fokusirati na Pyramid Python web okvir u kojem je odrađen praktični dio ovoga rada.

## 3.1 Django

Django je Python web okvir na visokoj razini koji potiče brzi razvoj i čisti, pragmatičan dizajn. Besplatan je i osmišljen je kako bi programerima pomogao u što bržem preuzimanju aplikacija od koncepta do završetka. Django uključuje desetke dodataka koje možete koristiti za rješavanje uobičajenih zadataka razvoja weba. Brine za autentifikaciju korisnika, te administraciju sadržaja.

Django ozbiljno shvaća sigurnost i pomaže programerima u izbjegavanju mnogih uobičajenih sigurnosnih grešaka, poput ubrizgavanja SQL-a, skriptiranja na različitim web-lokacijama, krivotvorenja zahtjeva na različitim web-mjestima i prekrajanja klikova. Njegov sustav autentifikacije korisnika pruža siguran način upravljanja korisničkim računima i lozinkama. Neka od najprometnijih web mjesta na planeti koriste Djangovu sposobnost da brzo i fleksibilno skalira kako bi zadovoljila najveće prometne zahtjeve.

## 3.2 Bottle

Bottle je brz, jednostavan i lagan WSGI(engl. Web Server Gateway Interface)[1] mikrookvir. Cijeli bootle kod se nalazi u jednoj datoteci i nema vanjskih ovisnosti osim standardne Python biblioteke. Bottle okvir je dobar izbor za ideje za izradu nekog prototipa, za naučiti kako se izrađuju web okviri te za izradu i pokretanje jednostavnih osobnih web aplikacija. Sve u svemu možemo reći da je Bottle minimalističan, brz i jednostavan Python okvir.

## 3.3 Web2py

Web2py je besplatni web-okvir otvorenog koda, objavljen je pod licencom LGPL verzije 3, za razvoj sigurnih web aplikacija koje se temelje na bazama podataka; napisan je na Pythonu i programibilan na Pythonu. Web2py je full-stack okvir, što znači da sadrži sve komponente koje su vam potrebne za izradu potpuno funkcionalnih web aplikacija. Razlikuje od ostalih web okvira po tome što je jedini okvir koji u potpunosti prihvaća paradigmu Web 2.0, gdje je web računalo. Osmišljen kako bi vodio web razvojnog programera da slijedi dobre prakse softverskog inženjeringa, kao što je korištenje uzorka kontrolera pogleda modela MVC (engl.

[1] WSGI je sučelje pristupnika web poslužitelja. To je specifikacija koja opisuje kako web poslužitelj komunicira s web aplikacijama i kako se web aplikacije mogu povezati zajedno kako bi obradile jedan zahtjev.

Model-View-Controller)[2]. web2py odvaja prikaz podataka (model) od prikaza podataka (view), a također i od logike aplikacije i tijeka rada (controler). web2py pruža knjižnice koje pomažu razvojnim programerima da osmisle, implementiraju i testiraju svaki od ova tri dijela zasebno te ih tjera da rade zajedno.

## 3.4 Tornado

Tornado je Python web okvir i knjižnica asinkronih mreža, izvorno razvijena na FriendFeedu. Koristeći mrežni I/O koji ne blokira, Tornado se može skalirati na desetke tisuća otvorenih veza, što ga čini idealnim za dugo prozivanje, WebSockets i druge aplikacije koje zahtijevaju dugotrajnu vezu sa svakim korisnikom. Tornado web okvir i HTTP poslužitelj zajedno nude potpunu alternativu WSGI-ju. Iako je moguće koristiti Tornado HTTP poslužitelj kao spremnik za druge WSGI okvire (WSGIContainer), ova kombinacija ima ograničenja i da biste u potpunosti iskoristili Tornado, morat ćete zajedno koristiti Tornadov web okvir i HTTP poslužitelj.

[2] MVC je uzorak softverskog dizajna koji se obično koristi za razvoj korisničkih sučelja koji dijele povezanu programsku logiku na tri međusobno povezana elementa.

# 4 Pyramid

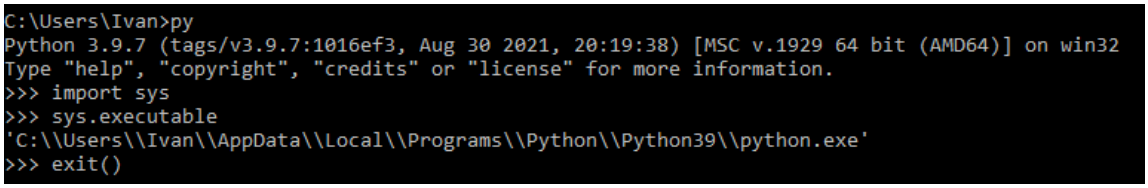
Pyramid je otvorenog koda, mal je stoga i brz Python web okvir. Osmišljen je kako bi olakšao izradu samih web aplikacija.

Slijedi određena načela dizajna i inženjeringa (Pyramid [7]):

- Jednostavnost – jednostavan za koristiti, te možete započeti sa radom čak i ukoliko ne razumijete sve.
- Minimalizam – daje samo osnovne alate potrebne za izradu gotovo svih web aplikacija.
- Dokumentacija – zalaže za opsežnu i ažuriranu dokumentaciju.
- Ubrzati - Piramida je osmišljena tako da bude izrazito brza.
- Pouzdanost - Piramida je razvijena konzervativno i iscrpno testirana.
- Otvorenost – kao i kod Pythona, softver Pyramid distribuira se pod dopuštenom licencom otvorenog koda .

### 4.1.1 Brzi početak

Pyramid podržavaju sve verzije Pythona. Nakon same instalacije Pythona potrebno je kroz CMD[3] (engl. Command Prompt) pronaći gdje je pohranjena Python aplikacija, te na toj lokaciji preuzeti i instalirati pip.




*Slika 2. Provjer lokacije Python aplikacije*

[3] Kratica za naredbu, cmd je naredba sustava Microsoft Windows koja otvara prozor naredbenog retka sustava Windows.

```
C:\Users\Ivan>cd C:\Users\Ivan\AppData\Local\Programs\Python\Python39

C:\Users\Ivan\AppData\Local\Programs\Python\Python39>curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 1911k  100 1911k    0     0  1911k      0  0:00:01  0:00:01 --:--:-- 1747k

C:\Users\Ivan\AppData\Local\Programs\Python\Python39>py get-pip.py
Collecting pip
  Using cached pip-21.2.4-py3-none-any.whl (1.6 MB)
Collecting wheel
  Downloading wheel-0.37.0-py2.py3-none-any.whl (35 kB)
Installing collected packages: wheel, pip
```

*Slika 3. Instaliranj pip-a*

Nakon što je pip instaliran javiti će gdje je pip pohranjen. Kroz CMD na lokaciji gdje je instaliran pip instalira se Pyramid i svi ostali dodaci kao što je Jinja2.

```
C:\Users\Ivan\AppData\Local\Programs\Python\Python39\Scripts>pip install pyramid
Collecting pyramid
  Using cached pyramid-2.0-py3-none-any.whl (246 kB)
Collecting plaster-pastedeploy
  Using cached plaster_pastedeploy-0.7-py2.py3-none-any.whl (7.8 kB)
Collecting zope.deprecation>=3.5.0
```

*Slika 4. Instalacija Pyramida*

Pyramid web okvir olakšava izradu web stranica. Za minimalan zadatak/odgovor „hello world“, koji je prikazan u donjem primjeru, možete započeti vrlo jednostavno, ali vas može dovesti jako daleko. Kako web aplikacija raste, Pyramid vam nastoji učiniti pisanje složenijih softvera mnogo lakšim, odnosno s manje napora (Try Pyramid [8])

```python
from wsgiref.simple_server import make_server
from pyramid.config import Configurator
from pyramid.response import Response

def hello_world(request):
    return Response('Hello World!')

if __name__ == '__main__':
    with Configurator() as config:
        config.add_route('hello', '/')
        config.add_view(hello_world, route_name='hello')
        app = config.make_wsgi_app()
    server = make_server('0.0.0.0', 6543, app)
    server.serve_forever()
```

*Slika 5. Brzi početak "hello world"*

### 4.1.2 Konfiguracija

Pyramid vam omogućuje da zadržite svoju konfiguraciju tik uz kod. Na taj način ne morate mijenjati datoteke da biste vidjeli svoju konfiguraciju. Na primjer:

```
from pyramid.view import view_config
from pyramid.response import Response

@view_config(route_name='fred')
def fred_view(request):
    return Response('fred')
```

*Slika 6. Konfiguracija*

Međutim, korištenje dekoratera konfiguracije Pyramid ne mijenja vaš kod. Ostaje ga lako produžiti, testirati ili ponovno upotrijebiti. Možete testirati svoj kod kao da dekorateri nisu tu. Okviru možete uputiti da zanemaruje neke dekoratere. Možete čak upotrijebiti imperativni stil za pisanje svoje konfiguracije, potpuno preskačući dekoratere.

### 4.1.3 Razvijajte se interaktivno

Pyramid može automatski otkriti promjene koje unesete u datoteke predloška i kod, tako da su vaše promjene odmah dostupne u vašem pregledniku. Možete otkloniti pogreške pomoću običnih starih `print()` poziva koji će se prikazati na vašoj konzoli. Ima alatnu traku za otklanjanje pogrešaka koja vam omogućuje da vidite informacije o tome kako vaša aplikacija radi izravno u vašem pregledniku. Kad vaša aplikacija ima pogrešku, interaktivni program za otklanjanje pogrešaka omogućuje vam da pregledate preglednik kako biste saznali što se dogodilo.

### 4.1.4 Započnite s malim, završite veliko i ostanite završeni

Započnite s malim, završite veliko i ostanite završeni (engl. The Start Small, Finish Big Stay Finished Framework) je moto Pyramid web okvira. Stoga ćemo navesti što svaki od njih zapravo znači.

#### 4.1.4.1 Započnite s malim

Brz i jednostavan početak ključna je stavka lakih okvira. Jednako tako, možete izabrati koje ćete pristupe koristiti za predloške, baze podataka, sigurnost i drugo. Piramida se ističe u smanjenju do prvog sata učenja.

- Počnite kao modul s jednom datotekom s malo složenosti u prvom satu
- Upotrijebite prikladnu metodu za generiranje uzorka projekta
- Birajte među raznim predlošcima, bazama podataka, sigurnosnim rješenjima i još mnogo toga koristeći kvalitetu i praktičnost dodatnog sustava Pyramid
- Iskoristite raznoliku dokumentaciju visoke kvalitete za procjenu, isprobavanje ili napredni razvoj s Pyramidom

#### 4.1.4.2 Završite veliko

Ambiciozni projekti teže velikom rastu bez gubitka dostojanstva. Piramida je jedinstveno opremljena za mjerenje s vama. Njegov sustav konfiguracije, proširenja i dodataka daje kostur da podrži vaše ambicije, a njegova arhitektura osigurava da ne izgubite one performanse i značajke kvalitete s kojima ste započeli.

- Uključite i konfigurirajte dodatke Pyramid unutar svoje aplikacije
- Nadjačajte i prilagodite osnovni kod i dodatke izvana, bez račvanja
- Izgradite i isporučite podsustave za višekratnu upotrebu unutar i izvan naše organizacije
- Brojni prošireni objekti ugrađeni u okvir
- Koristite Pyramid kao "okvirni okvir" za izradu vlastitog web sustava posebne namjene i domene

#### 4.1.4.3 Ostanite završeni

Jednostavan prvi sat Pyramid -a pomaže vam da započnete, a njegova proširivost pomaže vam da dovršite svoje ambicije. Pyramid pomaže u održavanju vaše aplikacije dovršenom razumijevanjem cijelog životnog ciklusa profesionalne web aplikacije.

- Duboka predanost stabilnosti API -ja i ispravljanje grešaka u više od 120 izdanja softvera
- Kultura pokrivenosti testova i dokumentacije čini Pyramid izborom osiguranim za budućnost
- Držanje konfiguracije izvan koda znači manje mogućih grešaka u budućnosti
- Vrhunske performanse čak i s rastom okvira
- Duboka proširivost i veliki uzorci dizajna znače da ga nećete prerasti

# 5 SQLite baza podataka

Najjednostavnije rečeno, SQLite je programski paket u javnoj domeni koji omogućuje upravljanje relacijskom bazom podataka. Sustav relacijskih baza podataka koristi se za pohranu korisnički definiranih zapisa u velike tablice. Osim pohrane podataka i upravljanja, mehanizam baze podataka može obraditi složene naredbe upita koje kombiniraju podatke iz više tablica u generirana izvješća i sažetke podataka. SQLite je biblioteka na C jeziku koja implementira samostalni, pouzdani, brzi, mali cjeloviti, mehanizam SQL baze podataka. Ugrađen je u sve mobilne uređaje i većinu računala. SQLite je lagan po pitanju složenosti postavljanja, administrativnih troškova i korištenja resursa.

SQLite je definiran sljedećim značajkama (Using SQLite, 2010 [9]):

- Serverless – ne zahtijeva zaseban poslužiteljski proces ili sustav za rad. SQLite izravno pristupa datotekama za pohranu.
- Zero Configuration – nema poslužitelja znači nema postavljanja. Stvaranje instance baze podataka SQLite jednostavno je poput otvaranja datoteke.
- Cross-Platform – cijela instanca baze podataka nalazi se u jednoj datoteci na više platformi koja ne zahtijeva administraciju
- Self-Contained – jedna biblioteka sadrži cijeli sustav baze podataka koji se izravno integrira u aplikaciju domaćina
- Small Runtime Footprint – početna datoteka manja je od megabajta koda i zahtijeva samo nekoliko megabajta memorije. Uz neke prilagodbe, veličina biblioteke i upotreba memorije mogu se značajno smanjiti
- Transactional – SQLite transakcije potpuno su usklađene i omogućuju siguran pristup za više procesa ili niti
- Full-Featured – SQLite podržava većinu značajki jezika upita pronađenih u standardu SQL92
- Highly Reliable – razvojni tim SQLitea ozbiljno shvaća testiranje i provjeru koda

Za rad s SQLite-om koristi se modul sqlite. Prvo što je potrebno je to da stvorite poveznicu na bazu podataka kako bi se mogao koristiti sqlite3 modul (Branko Žitko [10])

```
import sqlite3

conn = sqlite3.connect('tesna_baza_podataka.db')
```

Nakon što se uspostavi konekcija stvara se objekt Cursor koji služi za izvršavanje SQL upita pomoću .execute() metode.

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

## 5.1 Definiranje tablica i relacija

Kako bi opisali rad baze podataka napravit ćemo relacijsku bazu koja će sadržavati podatke o fakultetu i studentima. Jedan student pripada jednom fakultetu, dok fakultetu pripada više studenata. SQL skripte se pokreću metodom *.executescript()*. U donjem primjeru e prikazano stvaranje tablica i relacija za gornje navedeni primjer.

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

      CREATE TABLE studenti ( id integer
            PRIMARY KEY, ime text NOT
            NULL, prezime text NOT NULL,
            fakultet_id text DEFAULT
            NULL,

      FOREIGN KEY (fakultet_id) REFERENCES fakultet (id));
""")
```

SQLite naredbe za stvaranje i brisanje tablica:

- *CREATE TABLE* – služi za stvaranje tabilca (pokreće se samo jedan put jer će SQLite javiti grešku ako tablica postoji)
- *CREATE TABLE IF NOT EXISTS* - kreirati tablicu ako ona ne postoji
- *DROP TABLE* - služi za brisanje tablice
- *DROP TABLE IF EXISTS* - za brisanje tablice ako postoji

Unutar naredbi za stvaranje tablica imamo pojam text. Text je jedan od mogućih vrijednosti koje se mogu pronaći unutar same tablice. Još neki od njih bi bili:

- real - realni brojevi
- integer - cijeli brojevi

- NULL - null vrijednost
- blob - binarni skup podataka

PRIMARY KEY je polje ili skup polja s jedinstvenim vrijednostima u tablici. Vrijednosti primarnog ključa mogu se koristiti za referiranje na sve zapise jer svaki zapis ima različitu vrijednost ključa. Svaka tablica može imati samo jedan primarni ključ.

FOREIGN KEY (vanjski ili strani ključ) je ključ koji se za razliku od primarnog ključa može ponavljati na više mjesta.

## 5.2 Brisanje i promjena podataka iz tablice

Za brisanje podataka, odnosno uklanjanje, iz tablice koristi se SQLite naredba *DELETE.*

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

SQLite naredba *UPDATE* služi za promjenu podataka u bazi.

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

## 5.3 Dobavljanje podataka

Pomoću SQLite naredbe *SELECT* dohvaćamo podatke iz tablica. Metoda *.fetchone()* služi za dohvaćanje prvog elementa, ukoliko se pozove ponovo vraća sljedećeg po redu. U slučaju da nema više što vratiti, vratit će None.

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

Ukoliko želimo ispis svih, kao liste, koristimo metodu *.fetchall().*

```
cur.execute("""
      SELECT studenti.ime,studenti.prezime,fakulteti.naziv
```

```
        FROM studenti
        JOIN fakulteti ON studenti.fakultet_id = fakultet.id
    """)
    print(cur.fetchall())
```

## 5.4 Ubacivanje podataka u tablicu

Kako bi ubacili novi podatak u tablicu koristimo SQLite naredbu *INSERT*.

```
cur.execute("
    INSERT INTO fakulteti (id, naziv) VALUES (?, ?)", ( "PFST",
    "Pomorski fakultet u Splitu"))
```

U praksi ? unutar naredbi označava mjesta na koja će se ubaciti vrijednost, stoga Python na vrlo jednostavan način će pretvoriti Pythonove tipove podataka u SQLite tipove podataka.

Ukoliko želimo ubaciti dva ili više podataka odjednom moramo uvesti SQLite sintaksu i to pozivom metode *.executemany().*

```
    fakulteti = [("EFST", "Ekonomski fakultet u Splitu"),
        ("PMFST", "Prirodoslovno matematički fakultet u Splitu"),
        cur.execute(" INSERT INTO fakulteti (id,naziv) VALUES
        (?,?)",fakulteti) conn.commit()
```

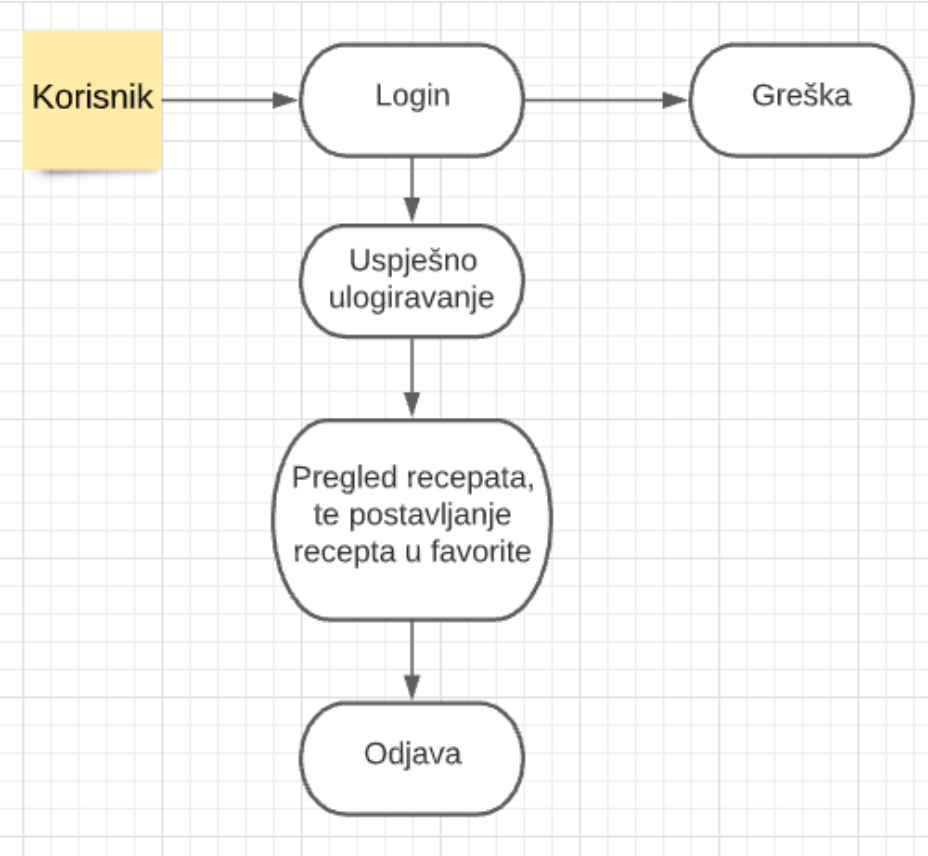
Kako bi sve promjene na bazi podataka bile trajne na kraju moramo pozvati metodu *.comit().* Ako se prilikom unosa direktno ne postavi vrijednost polja id-a, kao u gornjem primjeru on se automatski generira.

# 6 Razvoj aplikacije

U ovom poglavlju ćemo prikazati korake izrade web aplikacije recepti. Ideja je bila napraviti vrlo jednostavnu aplikaciju u kojoj korisnik može pretraživati razne recepte. Kroz samu izradu aplikacije cilj je bio iskoristiti sve navedene tehnologije kroz ovaj rad.

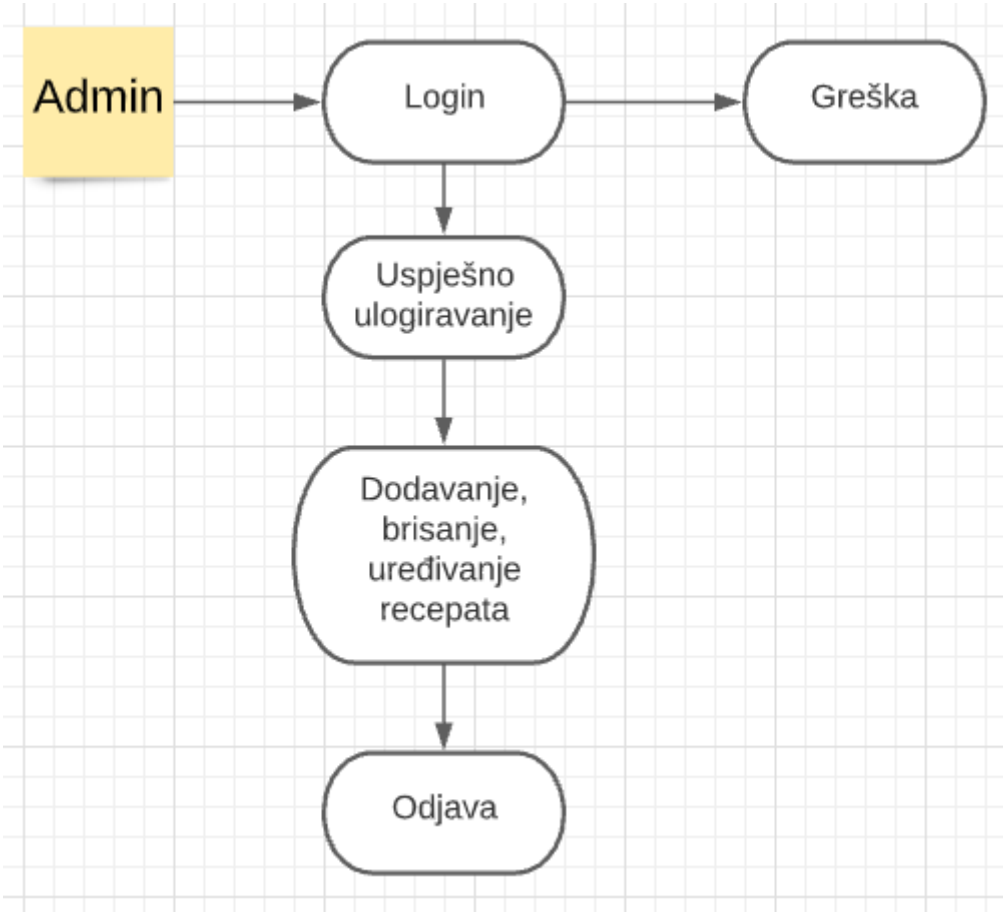
Prije same izrade aplikacije bilo je potrebno definirati što će aplikacija moći raditi. Pošto je u pitanju vrlo jednostavna web aplikacija, aplikacija ima i vrlo malo mogućnosti kao što su prijavljivanje kao korisnik ili admin. Kako bi se korisnik mogao prijaviti prvo je potrebna registracija, te se korisnik sa registriranim podacima svaki sljedeći put ponovno prijavljuje. Korisnik ima samo mogućnosti pregleda recepata, te spremanja određenog recepta u favorite, dok admin ima veće mogućnosti i ovlasti kao što su dodavanje, brisanje i uređivanje samih recepata.

Najpraktičnije je prije samoga programiranja napraviti Use-Case-Diagram (UCD) kako bi bolje vidjeli sve pojedinosti aplikacije. Ovi dijagrami nam daju grafički uvid u sve ovlasti unutar aplikacije.




*Slika 7. Use case dijagram za korisnika*

Use case dijagram prikazan na slici gore je grafički prikaz funkcionalnosti za korisnika aplikacije. Na dijagramu se može vidjeti da se korisnik može prijaviti, odjaviti, te pregledavati recepte. Dijagram na slici dolje nam prikazuje isto tako funkcionalnosti ali za administratora koji kao što se može isčitati iz dijagrama im puno više funkcionalnosti nego sam korisnik. Administrator može brisati, uređivati i dodavati nove recepte.




*Slika 8. Use case dijagram za admin-a*

## 6.1 Izrada aplikacije

Nakon što definiramo što želimo da nam aplikacija radi možemo započeti sa samim programiranje web aplikacije. Pogledajmo za početak izradu dizajna web aplikacije, odnosno kako će nam sve to izgledati kao otvorimo stranicu u pregledniku. Cijeli ovaj dio se izvodi kroz Jinja2 predložak. Unutar same aplikacije postoji nekoliko datoteka sa nastavkom *.jinja2* za uređenje svake pojedine stranice, bilo to favoriti, login, registracija, itd. Ispod je prikazan kod za izradu početne stranice *index.jinja2*.

```
<!DOCTYPE html>
<html>
    <body onload="ucitajNavbar(document.getElementById('session').value)">
      <div id="navbar-html"></div>
      <input type="hidden" id="session" value="{{session}}">
      <div class="jumbotron jumbotron-fluid">
```

```
        <div class="container">
          <h1 class="display-4">Recepti</h1>
          <p class="lead">Svi recepti na jednom mjestu.</p>
        </div>
      </div>
      {% for id, naslov, sastojci, slika, recept, isInFavorite in data %}
        <div id="{{id}}" class="alert alert-info" role="alert">
          <div class="post-container">
            <h2 class="font-weight-bold" >{{naslov}}</h2> </div>
          <div class="post-container">
            <div class="post-thumb"><img height="250px"alt ="Slika je trenutno
nedostupna" src="{{slika}}" /></div>
            <div class="post-content">
                <h3 class="post-title">Sastojci</h3>
                <ul class="list-group">
                {% for sastojak in sastojci %}
                    <li class="list-group-item">{{sastojak}}</li>
                {% endfor %}
                </ul>
          </div>
        </div>
        <div class="post-container">
        <h3 class="post-title">Opis</h3> <p></p>{{recept}} </p>
        </div>
          {% if session == 1 %}
            <div>
              <button value = "{{id}}" type="button"
onclick='deleteRecipe(document.getElementById(this.value))' class="btn btn-
primary">Izbrisi</button>
              <button value = "{{id}}" type="button"
onclick='window.location.replace(`/admin/update/${this.value}`)'  class="btn btn-
primary">Uredi</button>
            </div>
          {% endif %}
          {% if session == 2 %}
            <div>
              {% if not isInFavorite %}
                <button  value = "{{id}}" type="button" onclick='addFavorite(this)'
class="btn btn-success">Dodaj u favorite</button>
            {% else %}
                <button value = "{{id}}" type="button"
onclick='removeFavorite(this)'  class="btn btn-danger">Izbrisi iz favorita</button>
            {% endif %}
            </div>
          {% endif %}
        </div>
    {% endfor %}
    </body>
</html>
```

Svi ostali jinja2 predlošci u malo izmijenjeni s obzirom što taj dio stranice zahtijeva. Nakon dizajna same web stranice napravljena je mala baza podataka, pomoću Pytonovog sqlite3 modula, za spremanje svih dodanih recepata unutar *DbModel.py*. Datoteka također sadrži metode koje nam služe za  odabir recepta, brisanje, dodavanje i uređivanje recepata.

```
import sqlite3

from enum import Enum

class DbModel():
    def __init__(self,filename):
        self.lokacija_slike = "img/"
        self.conn=sqlite3.Connection(filename)
        self.cur=self.conn.cursor()
        self.filename=filename
        self.cur.executescript("""
            CREATE TABLE IF NOT EXISTS User(
              id INTEGER PRIMARY KEY AUTOINCREMENT,
              ime text NOT NULL,
              prezime text NOT NULL,
              username text NOT NULL,
              password text NOT NULL);

            CREATE TABLE IF NOT EXISTS Recipe(
              id INTEGER PRIMARY KEY AUTOINCREMENT,
              naslov text NOT NULL,
              sastojci text NOT NULL,
              slika text NOT NULL,
              recept text NOT NULL);

            CREATE TABLE IF NOT EXISTS Favorite(
              id INTEGER PRIMARY KEY AUTOINCREMENT,
              user_id INTEGER,
              recept_id INTEGER,
              FOREIGN KEY(user_id) REFERENCES User(id),
              FOREIGN KEY(recept_id) REFERENCES Recipe(id));""")

def selectRecipes(self, userId = None):
            self.cur.execute( """SELECT * FROM Recipe""")
            return [(id,naslov,sastojci,slika,recept, False if userId
is None else
                     self.checkIsInFavorites(userId,id))
                     for id, naslov, sastojci, slika, recept in
self.cur.fetchall()]


    def selectRecipe(self, id):
        self.cur.execute( """SELECT * FROM Recipe where id=
?""",(id,))
        data = self.cur.fetchall()
        if(len(data) > 0):
            data = data[0]
            return {'id':data[0], 'naslov':data[1], 'sastojci':
data[2], 'slika': data[3].replace('img/',""), 'recept': data[4] }
        return None

    def addRecipe(self, naslov, sastojci, slika, recept):
        slika = self.lokacija_slike + slika
        self.cur.execute("""INSERT INTO Recipe (naslov, sastojci,
slika, recept)
        VALUES (?, ?, ?, ?)""", (naslov, sastojci, slika, recept))
        self.conn.commit()

    def removeRecipe(self, id):
```

```
            self.removeFavorite(id)
            self.cur.execute("""DELETE FROM Recipe
                WHERE id = ?""", (id,))
            self.conn.commit()

        def updateRecipe(self,id, naslov, sastojci, slika, recept):
            slika = self.lokacija_slike + slika
            self.cur.execute("""UPDATE Recipe SET
                naslov = ?, sastojci = ?, slika = ?, recept = ? WHERE id =
? """,(naslov, sastojci, slika, recept, id))
            self.conn.commit()
```

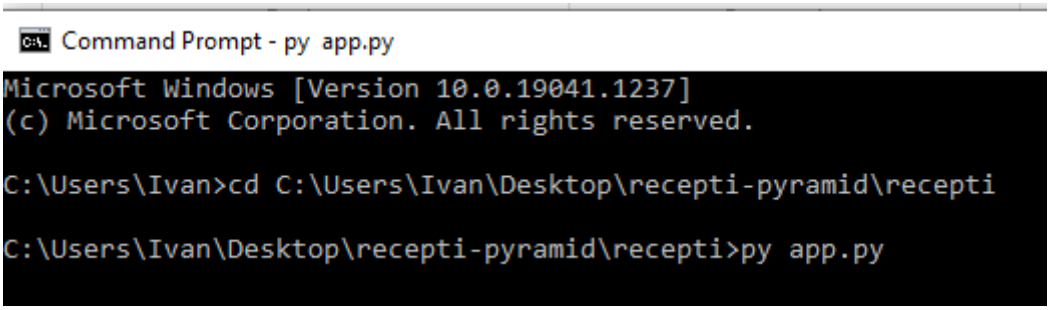
Glavna datoteka koja i služi za samo pokretanje aplikacije je *app.py* koja ima najbitnije funkcije, a to su provjera podataka za prijavljivanje. Kada kažem da su najbitnije funkcije baš one za prijavu, svi bi se složili jer tko bi htio da se svi mogu prijaviti npr. u njihov mail sa neispravnim podacima. Ispod je prikazan dio kod iz *app.py* za provjeru login korisnika.

```
@view_config(route_name = 'userLogin', renderer ='views/login.jinja2')

def admin_user_view(request):

    session = isSessionActive(request.session)

    if session:
        url = request.route_url('home')
        return HTTPFound(location=url)
    else:
        return {'session': session, 'naslov':'Korisnik', 'message':""}

@view_config(route_name = 'userLogin', request_method = 'POST')
def admin_user_post(request):
    data = json.loads(request.body)
    username,password = data['username'],data['password']
    session = isSessionActive(request.session)
    if session:
        url = request.route_url('home')
        return HTTPUnauthorized('Niste ulogirani kao admin')
    user = db_context.selectUser(username, password)
    if (user is not None):
        request.session['role'] = Role.USER.value
        request.session['uid'] = user[0]
        request.session['uName'] =user[1]
        return Response(body=json.dumps({'status':'ok'}))
    else:

        return Response(body=json.dumps({'session': session,
'Naslov':'Korisnik','status':'notOk', 'message':"Neispravni podaci
pokusajte ponovno"}))
```

## 6.2 Pokretanje aplikacije

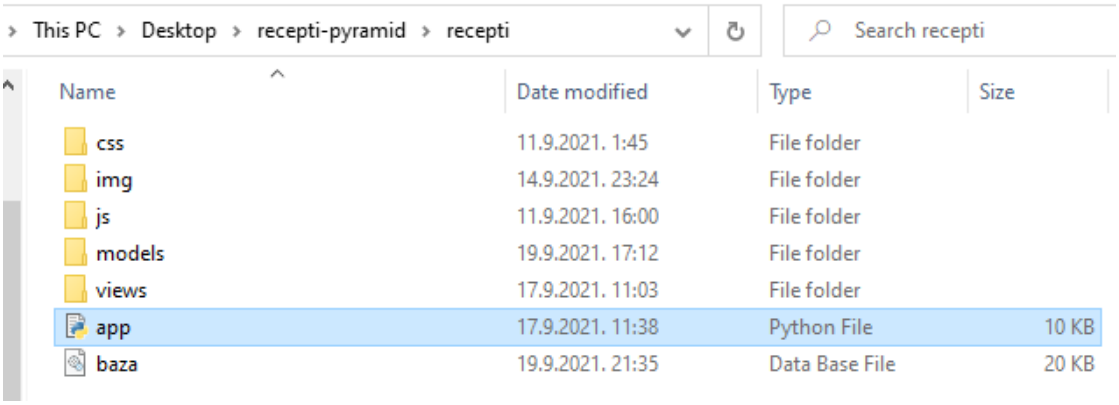
Aplikacija se može pokrenuti na dva načina:

1. Pokretanje kroz Command Prompt (CMD) – prvo trebate otvoriti CMD na računalu, te pomoću naredbe *cd 'putanja do lokacije'* doći do lokacije projekta, te pokrenuti sa te lokacije aplikaciju sa naredbom *py naziv*, te upisom u preglednik localhost:6543




*Slika 9. Pokretanje aplikacije kroz CMD*

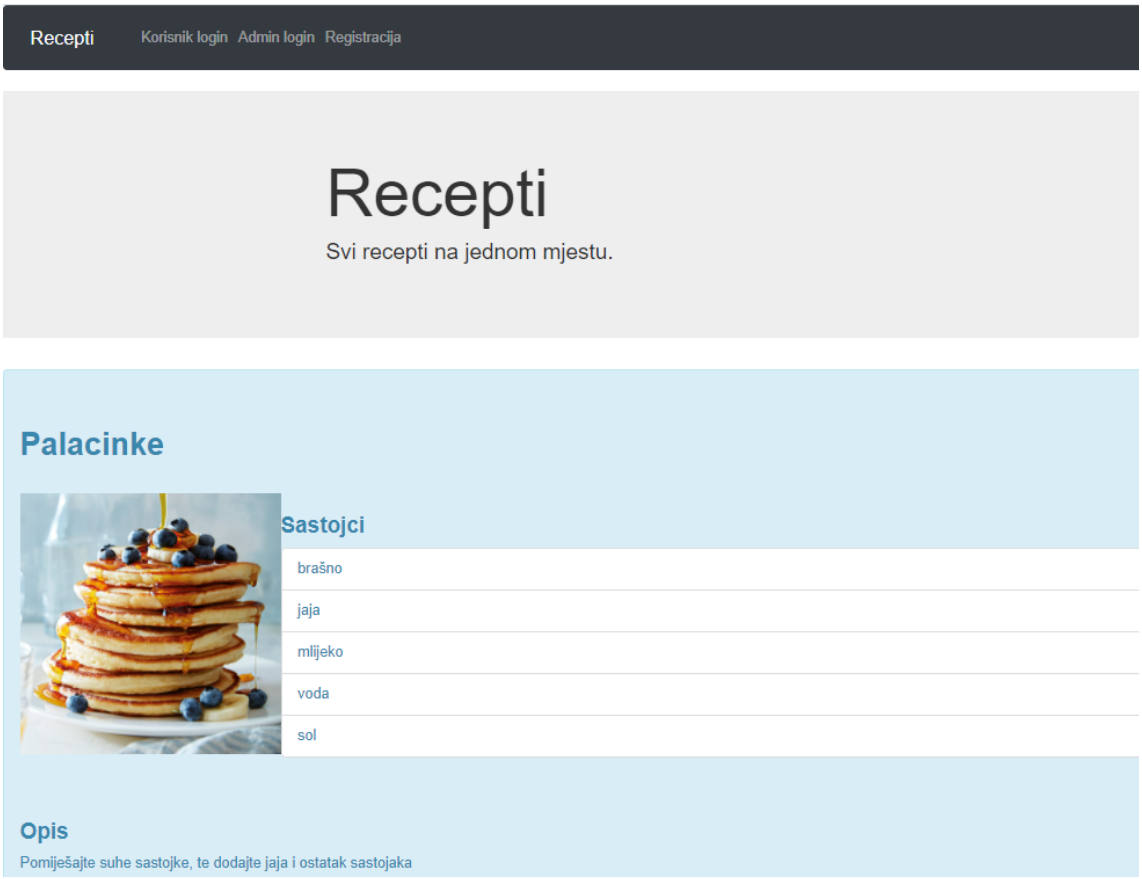
2. Pokretanje sa dvostrukim klikom na *app.py*, te upisom na preglednik localhost:6543




*Slika 10. app.py za pokretanje aplikacije*

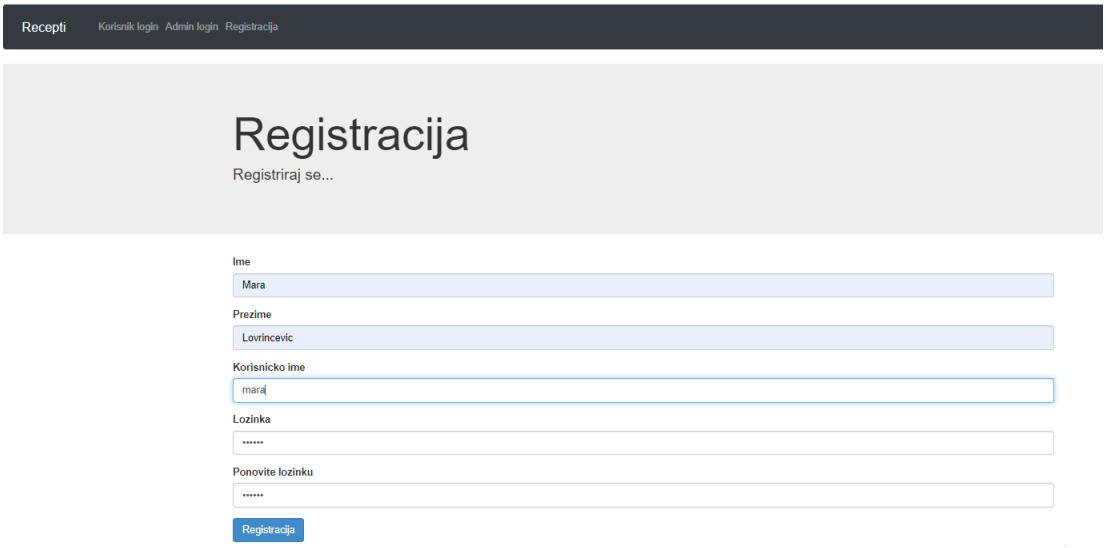
## 6.3 Rad aplikacije

Na početnoj stranici web aplikacije su vidljivi svi recepti. Također se preko ove stranice korisnik ili admin može ulogirati, te se novi korisnik može registrirati.

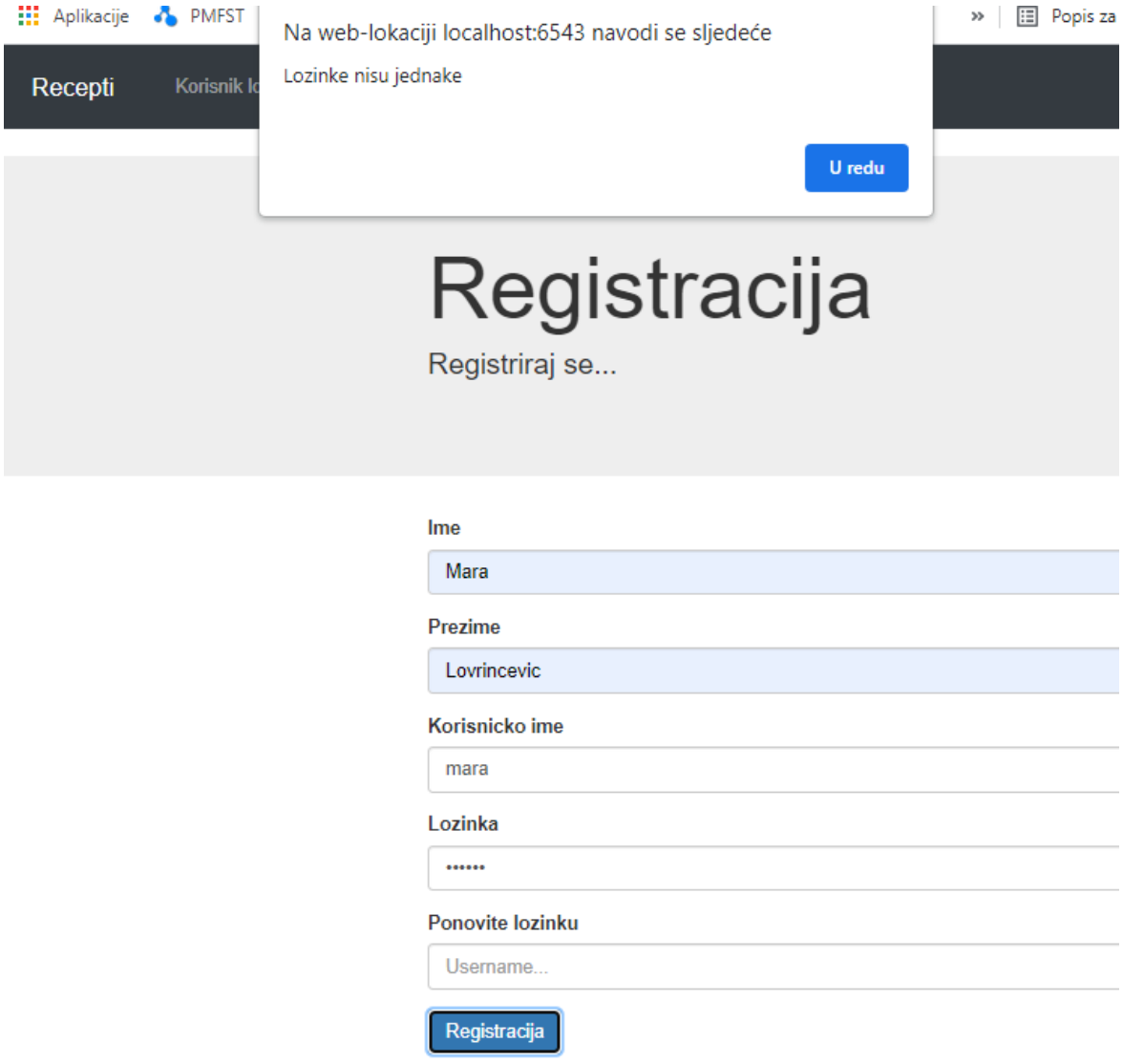



*Slika 11. Početna stranica*

Na stranici za registraciju se od korisnika traži unos podataka kao što su ime, prezime korisničko ime i lozinka. Ukoliko se neki od podataka ne unese javlja se greška.

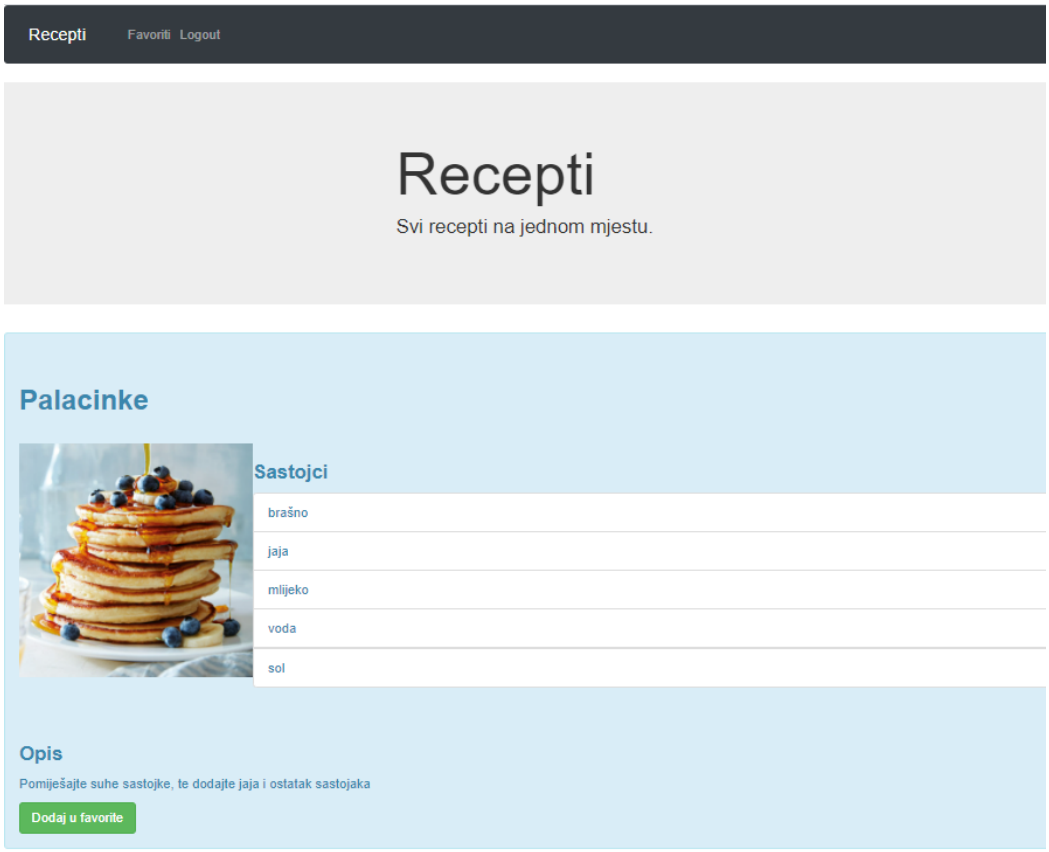



*Slika 12. Registracija*

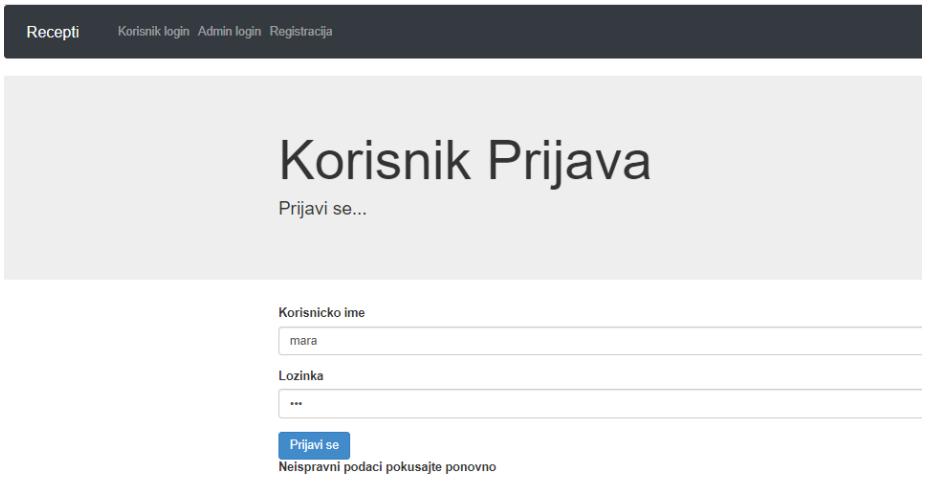



*Slika 13. Jedna od grešaka koja se javlja ukoliko se ne unesu svi podaci*

Korisnika stranica nakon uspješne registracije vodi direktno na dio stranice za prijavu. Ukoliko se unesu ispravni podaci korisnik je ulogiran te se otvara stranica sa receptima, a ukoliko su podaci neispravni javlja grešku da su podaci neispravni što znači da je korisnik neispravno unio ili korisničko ime ili lozinku.

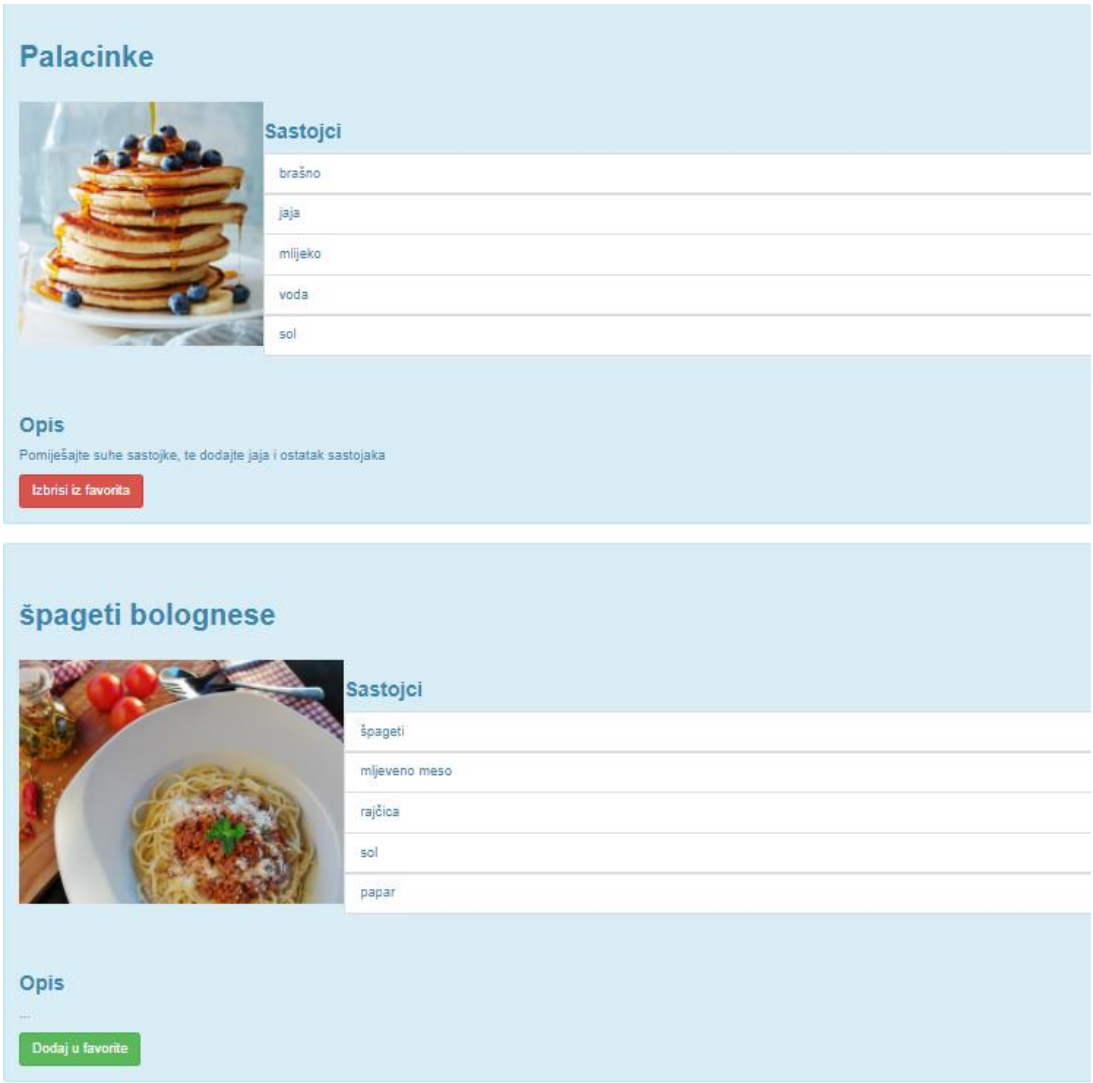



*Slika 14. Stranica nakon uspješne prijave*

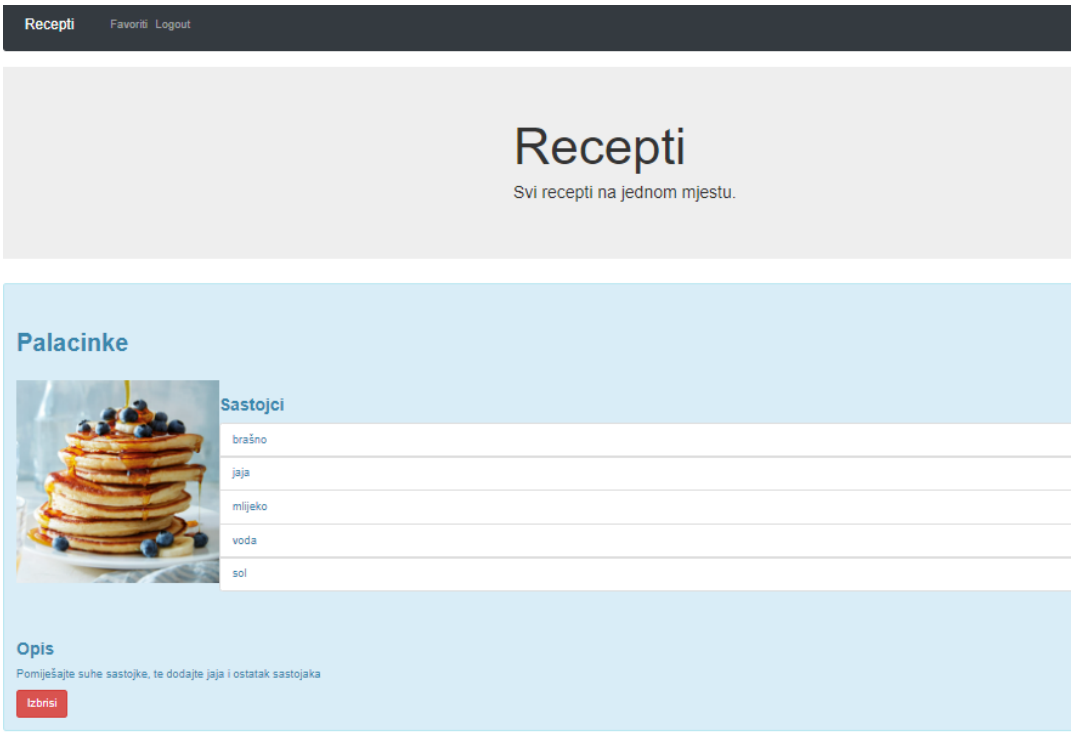



*Slika 15. Greška kod prijave korisnika*

Ukoliko se korisnik ispravno ulogirao može pregledavati sve recepte, dodati ih u favorite te u alatnoj traci ući na taj dio stranice i od tu čak i micati određene recepte iz favorita.

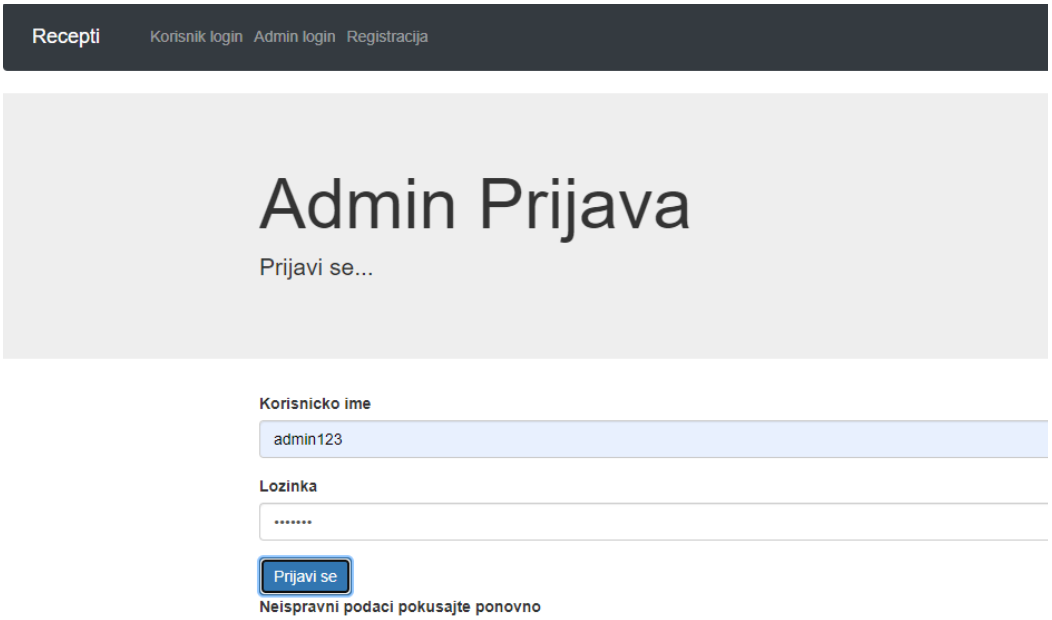



*Slika 16. Dodavanje u favorite*



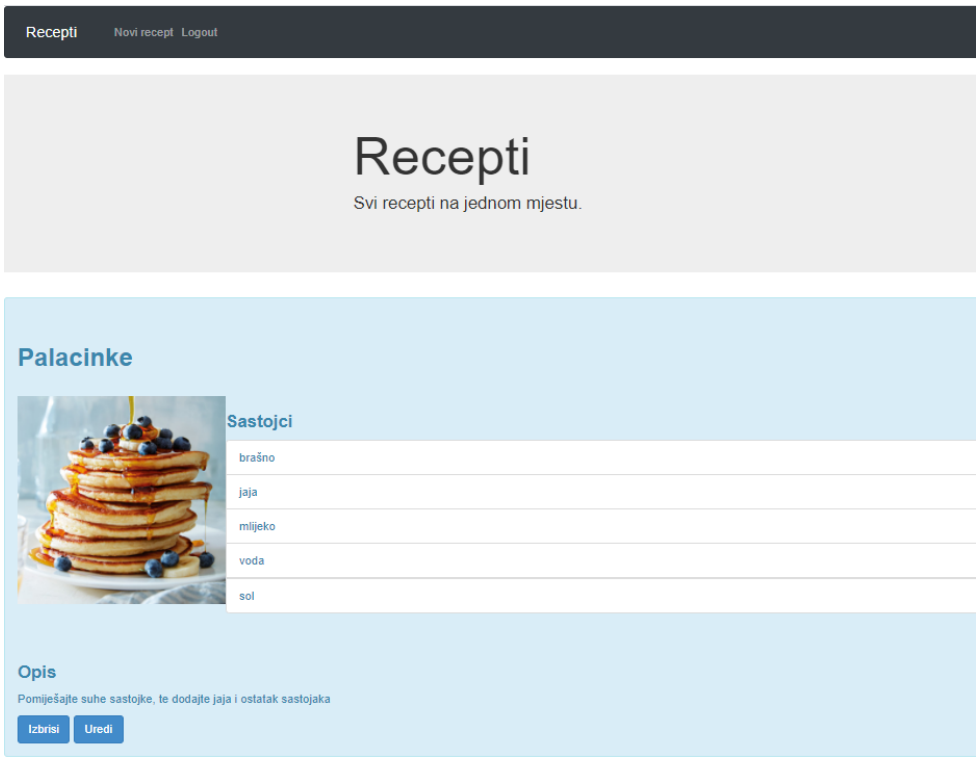

*Slika 17. Favoriti*

Korisnik se na kraju može odjaviti, te ga vraća na početnu stranicu. Admin ima jedinstveno korisničko ime i lozinku. Sa ispravnim podacima se samo administrator stranice može ulogirati.



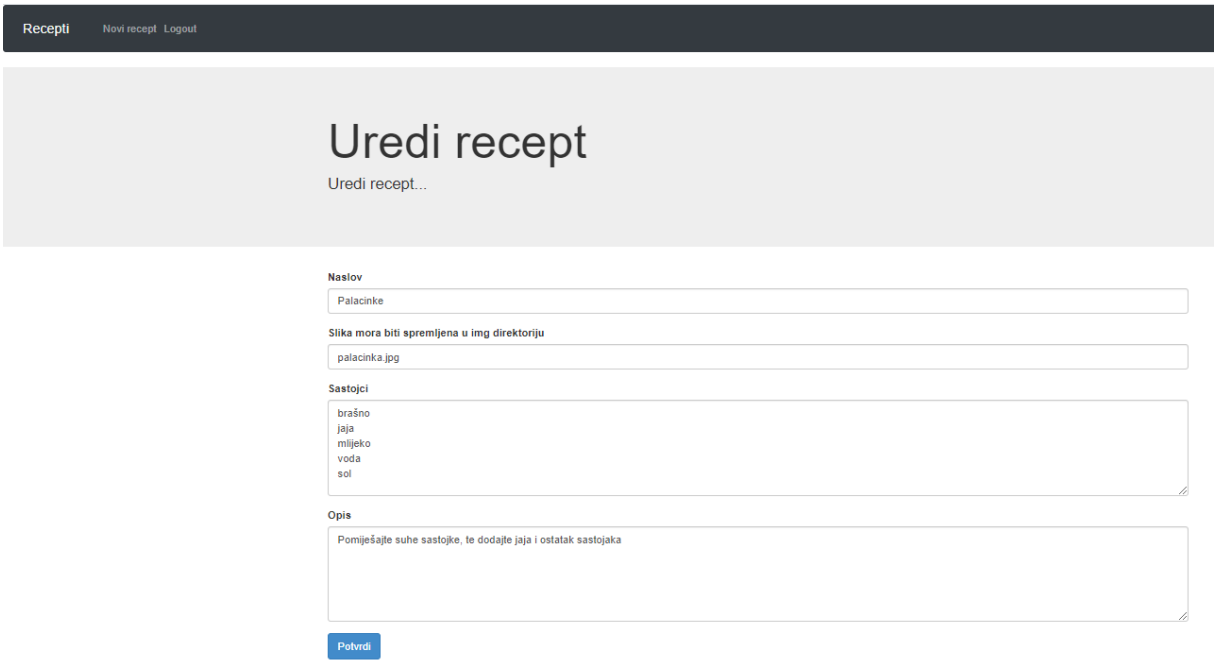

*Slika 18. Greška kod prijave*

Nakon ispravnog prijavljivanja administrator ima neograničene ovlasti na stranici, odnosno može uređivati i brisati recepte (na slici dolje u donjem lijevom kutu), te dodavati nove recepte (gornji lijevi kut).



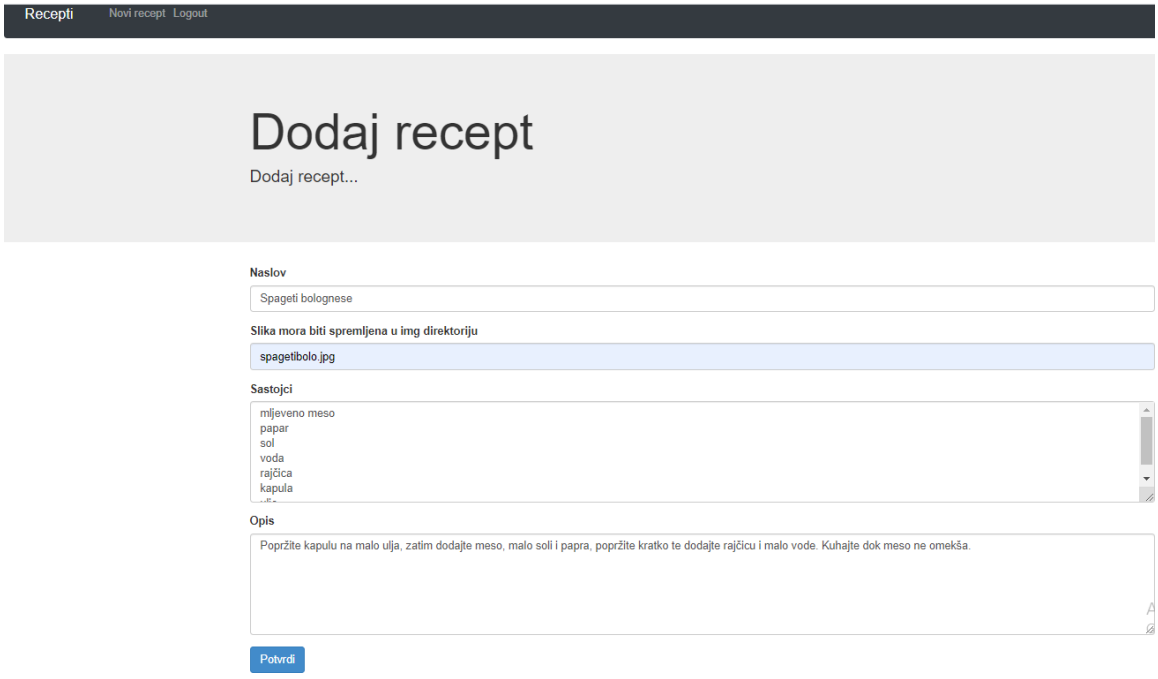

*Slika 19. Mogućnosti koje admin može raditi na stranici*

Ukoliko admin odluči izbrisati recept stisne na botun *izbriši* te se javlja poruka da je recept izbrisan, a ukoliko odluči da želi taj recept urediti otvara se novi dio stranice za uređivanje recepta koji je prikazan na donjoj slici.



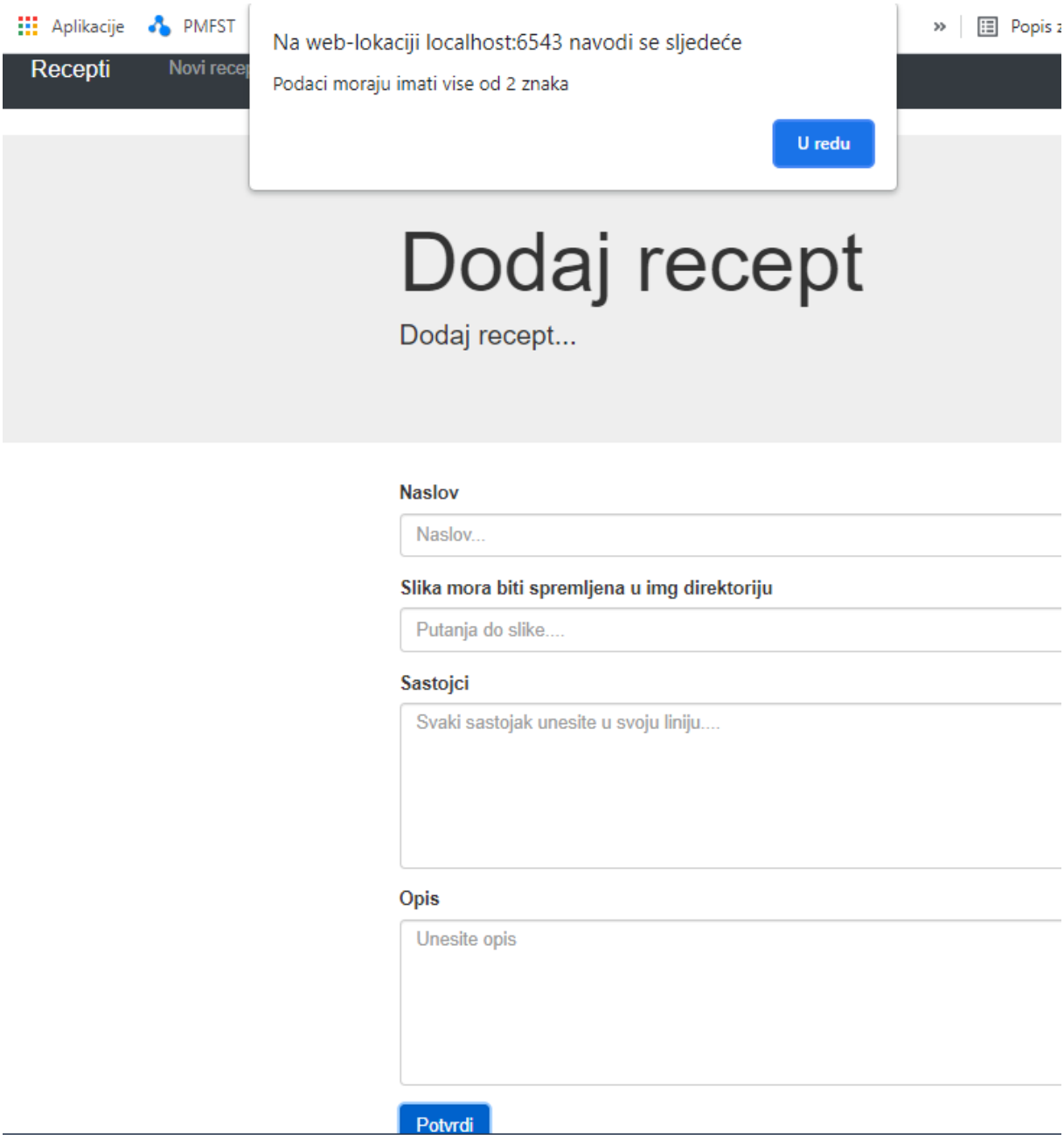

*Slika 20. Uređivanje recepta*

Nakon što je admin zadovoljan sa promjenom stisne na botun *Potvrdi* te ga stranica vraća na sve recepte. Kao što smo već spomenuli admin može i dodavati nove recepte klikom na *Novi recept* u alatnoj traci.




*Slika 21. Unos novog recepta*

Kao što možete vidjeti na gornjoj slici uz naziv, sastojke i kratki opis nekog recepta potrebno je staviti i „putanju“ do slike tog jela. Ta putanja je zapravo naziv slike koja se nalazi u folderu *img* naše aplikacije. Ukoliko su svi podaci uneseni recept je vidljiv među svim receptima. Ukoliko nešto nije ispravno uneseno javlja se greška.




*Slika 22. Greška kod unosa novog recepta*

Admin se isto kao i korisnik na kraju rada odjavljuje na botun *Logout* na alatnoj traci.

# 7 Zaključak

U ovom završnom radu prikazana je izrada web aplikacije s kojom nam je bio cilj prikazati rad Pyramid web okvira, baze podataka i samih web aplikacija. Web aplikacije su danas postale jako važne karike za poslovanja mnogih poduzeća stoga su njihova izrada i održavanje jako bitne.

Prije same izrade neke web aplikacije morate biti upoznati sa samim tim pojmom i sa pozadinom njenog rada. Sljedeći korak je upoznati se za programima koji imaju mogućnost izrade web aplikacija, te znati izabrati pravi program ili okvir u kojem ćete kasnije izraditi aplikaciju.

Prije izrade aplikacije trebalo bi pomno isplanirati svaki korak izrade. To uključuje sve zadatke koje aplikacija mora moći obavljati, bila to prijava, registracija ili možda nešto drugo. Svaka izrada aplikacije započinje izradom baze podataka koja je ključan korak jer se u njoj pohranjuju svi podaci (npr. korisnička imena i lozinke korisnika). Loše dizajnirana baza podataka će vam stvarati mnogo problema kod izrade aplikacije. Naravno kako bi vam bilo lakše kodirati aplikaciju i u budućnosti raditi nekakve promjene ili dodavanje dodatnih opcije na web aplikaciji važno je obratiti pažnju na organizaciju samog koda. I naravno na kraju je samo testiranje aplikacije kako bi se uvidjeli nedostaci ili greške u aplikaciji.

# 8 Literatura


[1] D. Ghimire, »Comparative study on Python web frameworks: Flask and Django,« 2020.

[2] »weblogographic,« 2018. [Mrežno]. Available: https://hr.weblogographic.com/difference-between-static-website-and-dynamic-website-7361.

[3] »Jinja Documentation (3.0.x),« [Mrežno]. Available: https://jinja.palletsprojects.com/en/3.0.x/intro/#installation.

[4] S. S. Quality, »What Is Web Application (Web Apps) and Its Benefits,« [Mrežno]. Available: https://searchsoftwarequality.techtarget.com/definition/Web-application-Web-app .

[5] R. Gibb, »StackPath,« 31 05 2016. [Mrežno]. Available: https://blog.stackpath.com/web-application/ .

[6] M. Sanders, »Scout APM,« 02 07 2020. [Mrežno]. Available: https://scoutapm.com/blog/the-most-popular-python-web-frameworks-in-2020.

[7] »The Pyramid Web Framework v2.0,« [Mrežno]. Available: https://docs.pylonsproject.org/projects/pyramid/en/latest/narr/introduction.html.

[8] »Try Pyramid,« [Mrežno]. Available: https://trypyramid.com/.

[9] J. A. Kreibich, Using SQLite, NY, America: O'Reilly Media, 2010.

[10] B. Žitko. [Mrežno]. Available: https://sites.google.com/site/brankozitko/kolegiji/upi.

# 9 Popis slika

Slika 1. Instalacija jinje2 ..... 5
Slika 2. Provjer lokacije Python aplikacije ..... 9
Slika 3. Instaliranj pip-a ..... 10
Slika 4. Instalacija Pyramida ..... 10
Slika 5. Brzi početak "hello world" ..... 10
Slika 6. Konfiguracija ..... 11
Slika 7. Use case dijagram  za korisnika ..... 17
Slika 8. Use case dijagram za admin-a ..... 18
Slika 9. Pokretanje aplikacije kroz CMD ..... 22
Slika 10. app.py za pokretanje aplikacije ..... 22
Slika 11. Početna stranica ..... 23
Slika 12. Registracija ..... 24
Slika 13. Jedna od grešaka koja se javlja ukoliko se ne unesu svi podaci ..... 24
Slika 14. Stranica nakon uspiješne prijave ..... 25
Slika 15. Greška kod prijave korisnika ..... 25
Slika 16. Dodavanje u favorite ..... 26
Slika 17. Favoriti ..... 27
Slika 18. Greška kod prijave ..... 27
Slika 19. Mogućnosti koje admin može raditi na stranici ..... 28
Slika 20. Uređivanje recepta ..... 29
Slika 21. Unos novog recepta ..... 29
Slika 22. Greška kod unosa novog recepta ..... 30

# Skraćenice

WWW – engl. World Wide Web

HTML – engl. HyperText Markup Language

CSS – engl. Cascading Style Sheets

HTTP – engl. HyperText Transfer Protocol

XML – engl. eXtensible Marcap Language

PDF – engl. Portable Document Format

CMD – engl. Command Prompt

OS – engl. Operating System

WGSI – engl. Web Server Getaway Interface

API – engl. Asynchronous Transfer Mode

SQL – engl. Structured Query Language

MVC – engl. Model/View/Controller

URL – engl. Uniform Resource Locator

# Sažetak

U ovom završnom radu prvotno su objašnjeni pojmovi kao što su web aplikacija i web okvir. Također i sam opis Pyramid Python okvira i SQLite baze podataka koji su korišteni za izradu ovog rada. Nakon kratkog uvoda o Pyramid okviru prikazan je jedan od načina instalacije okvira i brzi početak rada u samom okviru razvojem jednostavne aplikacije. Nakon teorijskog dijela, opisan je postupak izrade aplikacije praktičnog dijela završnog rada. U radu možete vidjeti i način na koji se u SQLite bazi podataka mogu dodavati, brisati i ažurirati podaci, te na samom kraju kratki postupak izrade aplikacije.

**Ključne riječi**: Python, web okvir, web aplikacija, Pyramid web okvir, SQLite

# Summary

In this thesis, concepts such as web application and web framework are initially explained. Also the very description of the Pyramid Python framework and SQLite database used to create this paper. After a short introduction about the Pyramid frame, one of the ways to install the frame and a quick start in the frame itself is shown by developing a simple application. After the theoretical part, the procedure of making the application of the practical part of the thesis is described. In this paper, you can also see the way in which data can be added, deleted and updated in the SQLite database, and at the very end, a short process of creating an web application.

**Keywords**: Python, web framework, web application, Pyramid web framework, SQLite