Izrada web aplikacije za oglašavanje nekretnina koristeći MVC obrazac softverske arhitekture

Zakarija, Jura

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Jura Zakarija

Zagreb, 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentori:

Dr. sc. Tomislav Stipančić

Student:

Jura Zakarija

Zagreb, 2021.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem mentoru Dr. sc. Tomislavu Stipančiću na iskazanoj pomoći, savjetima i uloženom vremenu tijekom izrade rada.

Jura Zakarija

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske radove studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala te mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum:	Prilog:
Klasa: 602-04/21-6/1	
Ur. broj: 15-1703-21	

DIPLOMSKI ZADATAK

Student: **JURA ZAKARIJA** Mat. br.: 0035194472

Naslov rada na hrvatskom jeziku: **Izrada web aplikacije za oglašavanje nekretnina koristeći MVC obrazac softverske arhitekture**

Naslov rada na engleskom jeziku: **Development of a web application for real estate advertising using the MVC form of software architecture**

Opis zadatka:

Model-pogled-kontrola (engl. Model–View–Controller - MVC) je obrazac softverske arhitekture koji se koristi kod razvoja softvera za odvajanje pojedinih dijelova aplikacije u komponente ovisno o njihovoj namjeni. U radu je potrebno razviti web aplikaciju za oglašavanje nekretnina koristeći MVC obrazac implementiran kroz odabranu web platformu. Aplikacija mora sadržavati funkcionalnosti prijave i registracije korisnika, te postavljanja i pretraživanja oglasa. Osim toga, potrebno je izraditi prikladno administracijsko sučelje za upravljanje sadržajem web aplikacije. Aplikaciju je potrebno postaviti na produkcijsko okruženje.

Osim toga potrebno je:

1. objasniti funkcionalnost modernih web aplikacija (REST, HTTP, Frontend/Backend, baze podataka, programski jezici)
2. dati pregled i usporedbu najčešće korištenih web okvira (Django, Express/Laravel/.NET)
3. izraditi web aplikaciju te razvijenu aplikaciju dovesti u vezu s MVC obrascem
4. opisati funkcionalnosti razvijene aplikacije.

U radu je potrebno navesti korištenu literaturu te eventualno dobivenu pomoć.

Zadatak zadan: 30. rujna 2021.

Rok predaje rada: 2. prosinca 2021.

Predviđeni datum obrane: 13. prosinca do 17. prosinca 2021.

Zadatak zadao: doc. dr. sc. Tomislav Stipančić

Predsjednica Povjerenstva: prof. dr. sc. Biserka Runje

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS KRATICA

Kratica	Opis
ASGI	Asynchronous Server Gateway Interface
CRUD	Create, Read, Update, and Delete
CSS	Cascading Style Sheets
DOM	Document Object Model
DTL	Django Template Language
HTML	Hyper Text Markup Language
JSON	JavaScript Object Notation
MVC	Model – View - Controler
MVT	Model – View – Template
NoSQL	Not only SQL
ORM	Object-Relational Mapper
RDBMS	Relational database management system
REST	Representational state transfer
SQL	Structured Query Language
VPS	Virtual Private Server
WSGI	Web Server Gateway Interface
WYSIWYG	What You See Is What You Get

SAŽETAK

U ovom radu opisan je rad modernih web aplikacija. Dan je kratak pregled arhitekture web aplikacija uključujući dizajn serverske strane i strane poslužitelja. Opisani su najpopularniji tipovi relacijskih i nerelacijskih baza podataka te aplikacijska programska sučelja. Zatim je dan pregled najkorištenijih programskih jezika te programskih okvira za izradu web aplikacija. Spomenuti programski okviri su Django, Express.js, Laravel i ASP.NET Core. Opisan je MVC obrazac izrade web aplikacija te je izražena je web aplikacija za prodaju nekretnina prema spomenutom obrascu. Dan je detaljan opis arhitekture spomenute aplikacije te je opisana uloga pojedinih datoteka u aplikaciji. Izrađena aplikacija ima funkcionalnost registracije i prijave korisnika te pretraživanja i postavljanja oglasa sa nekretninama. Spomenuta web aplikacija postavljena je na produkcijsko okruženje koristeći uslugu virtualnog privatnog servera(VPS)

Ključne riječi: , web aplikacija, Django, JavaScript, nekretnine, VPS

SUMMARY

This paper describes the work of modern web applications. A brief overview of the architecture of web applications including server side and client side design is given. The most popular types of relational and non-relational databases and application programming interfaces are described. Then an overview of the most used programming languages and programming frameworks for creating web applications was given. The aforementioned programming frameworks are Django, Express.js, Laravel and ASP.NET Core. MVC design pattern for web application creation is described and the web application for the sale of real estate is created according to the aforementioned MVC pattern. A detailed description of the architecture of the application is given and the role of individual files in the application is described. The created application has the functionality of registering and signing in of users and searching and placing real estate ads. The aforementioned web application is put on a production environment using a virtual private server (VPS) service

Keywords: , web application, Django, JavaScript, real estate, VPS

1. UVOD

S porastom broja korisnika interneta raste i broj novih web aplikacija i usluga koje se nude putem interneta. Danas je moguće kupovati putem interneta, rezervirati hotel za godišnji odmor, steći online diplome iz sve većeg broja različitih područja ili komunicirati s ljudima sa svih dijelova svijeta. Sve to omogućeno je zahvaljujući najmodernijim web tehnologijama i programskim jezicima.

Trenutno najpopularnije tehnologije su one napisane u programskim jezicima Python i JavaScript. Programski okviri kao što su Django, Angular, React ili Vue omogućavaju izradu brzih i skalabilnih web aplikacija koje svima olakšavaju život. Tome u prilog govori da najbogatije i najmoćnije tvrtke kao što su Facebook ili Google temelje svoju moć na web aplikacijama izrađenim u tim tehnologijama.

U ovom radu se stoga objašnjava arhitektura modernog weba te je izrađena web aplikacija putem spomenutog Django programskog okvira.

2. MODERNE WEB APLIKACIJE

Moderne web aplikacije sastoje se od više različitih dijelova ili slojeva. Najvažniji su baza podataka koja služi za pohranu korisničkih informacija, serverske strane ili strane poslužitelja(tzv. *backend*) koji služi za obradu informacija i klijentske strane(tzv. *frontend*) koji služi za prikazivanje informacija. Za komunikaciju između serverske i korisničke stran web aplikacije zadužen je API(*application programming interface* – aplikacijsko programsko sučelje). U nastavku su detaljnije pojašnjeni navedeni koncepti.

Slika 1. Moderne web tehnologije[1]

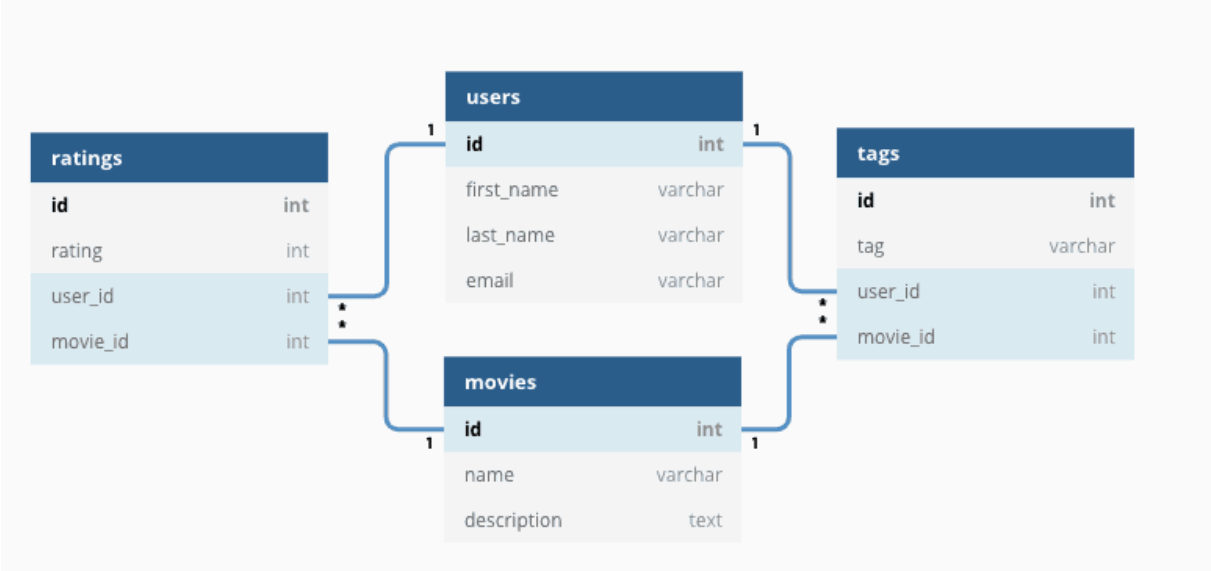
2.1. Baze podataka

Podaci su sastavni dio svake moderne web aplikacije. Najčešće su pohranjeni na poslužitelju, unutar baze podataka. Ovisno o tipu, količini i namjeni podataka, spremaju se u relacijske(SQL) ili nerelacijske(NoSQL) baze podataka.

Relacijske baze podataka služe za pohranu strukturiranih podataka u obliku tablica, a svaka tablica sastoji se od velikog broja zapisa i redaka međusobno povezanih primarnim ključevima sa drugim tablicama. Postoje 3 tipa veze između podataka:

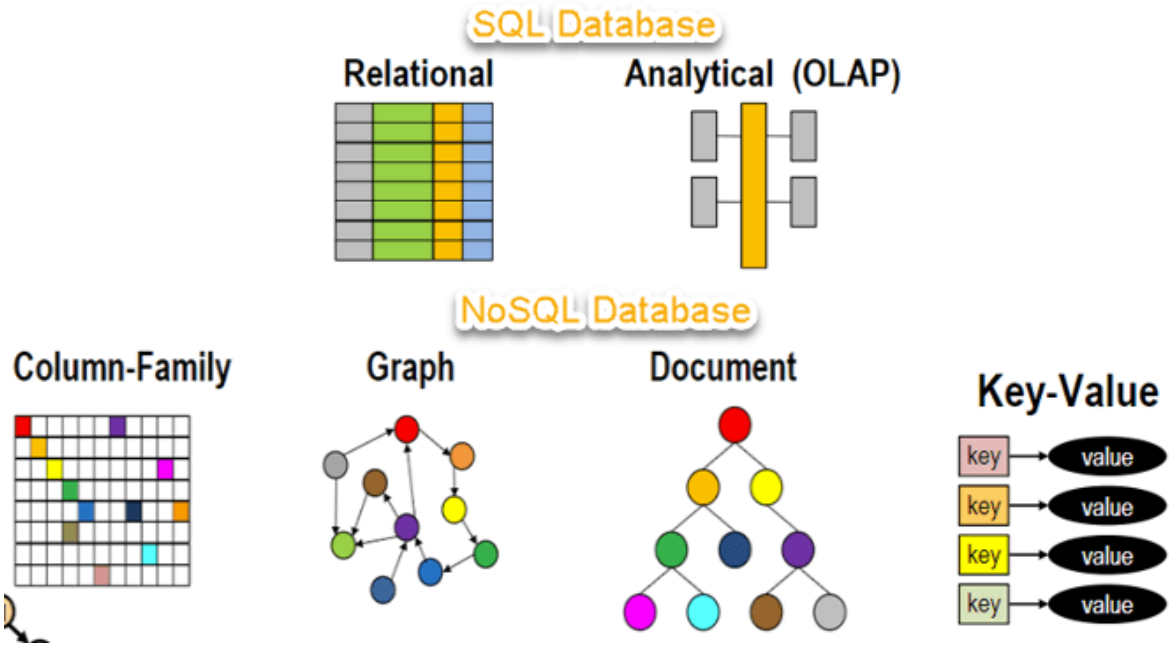
- Veza 1 naprema 1 – podatak iz jedne tablice može biti povezan samo sa jednim podatkom iz druge tablice
- Veza 1 naprema N – podatak iz jedne tablice može biti povezan sa više podataka iz druge tablice
- Veza N naprema M – veza više naprema više, odnosno svaki podatak može biti povezan s više podataka iz druge tablice.

Najčešće se koriste u raznim poslovnim sustavima u kojima je potrebno baratati velikom količinom visoko strukturiranih podataka. Na slici 2 prikazana je jednostavna struktura podataka unutar relacijske baze podataka. Neke od poznatijih relacijskih baza podataka su MySQL, Oracle, Microsoft SQL Server i Postgresql.



Slika 2. Prikaz relacijske baze podataka[2]

Nerelacijske baze podataka(NoSQL) koriste se za pohranu nestrukturiranih podataka. Ovisno o potrebi i namjeni ovakve baze podataka, postoji više načina pohrane podataka unutar NoSQL baza. Najpopularnije su dokument baze, stupac baze, ključ-vrijednost baze i graf baze. Popularne NoSQL baze podataka su MongoDB, Cassandra, Redis, HBase, Neo4j, Amazon DynamoDB, Couchbase i Memcached. Na slici 2 prikazana je usporedba između relacijskih i nerelacijskih baza podataka. NoSQL baze podataka se najčešće koriste kada je potrebno baratati nestrukturiranim podacima te se najviše koriste za pohranu podataka sa društvenih mreža i sličnih stranica.



Slika 3. Usporedba relacijskih i nerelacijskih baza podataka[3]

2.2. Strana poslužitelja(Backend)

Strana poslužitelja ili backend služi prvenstveno za obradu podataka. Programski kod ovog sloja izvršava se na poslužitelju. Backend se ovih dana najčešće piše u nekom od popularnih programskih okvira koji uvelike ubrzavaju razvoj aplikacija. Ovaj sloj komunicira sa bazom podataka putem procedura i transakcija ili pomoću programskog alata ORM(Object–relational mapping). ORM je alat koji služi za apstrakciju podataka unutar baze podataka te olakšava pristupanje i upravljanje podatcima putem programskog koda na poslužitelju. Postoji mnogo vrsta ORM-ova te gotovo svaki programski okvir koristi vlastiti. Najčešće korišteni programski jezici na strani poslužitelja su Java, C#, Python, PHP, Golang i JavaScript.

Java je objektno orijentirani programski jezik visoke razine. Razvili su ga Jamse Gosling i Patrik Naughton, a objavljen je 1995. godine. Najveća prednost ovog jezika je ta što se može izvoditi na svim operacijskim sustavima unutar programskog rješenja JVM(Java Virtual Machine). Trenutno je 3. najkorišteniji programski jezik. Najpoznatiji programski paketi napisani u Javi su Spring i Hibernate

C# je objektno orijentirani programski jezik razvijen 2000. godine od strane Microsoft-a i Andreasa Hejlsberga. Sintaksom je sličan Javi. Najpoznatiji programski okviri napisani u C#-u su .Net Core i Blazor.

Python je interpretirani jezik opće namjene. Nastao je 1991 godine, a njegov tvorac Guido van Rossum počeo je raditi na njemu još u kasnim 1980-ima. Odlikuje ga jednostavna sintaksa i velik broj programskih paketa te je trenutno najpopularniji te 2 najkorišteniji programski jezik, odmah iza Javascripta. Najpoznatiju programski okviri za izradu web aplikacija u Pythonu su Django, Flask i FastAPI.

PHP je skriptni jezik opće namjene orijentiran prema razvoju web aplikacija. Nastao je 1994. godine, a osmislio ga je dansko-kanadski programer Rasmus Lerdorf. Najveći broj web stranica trenutno objavljenih na internetu napisana je u PHP-u. Najpoznatiji programski okviri za PHP su Laravel i Symfony. Ovdje valja spomenuti i WordPress, koji je upravljački sustav(CMS) za web aplikacije temeljen na PHP-u i MySQL bazi podataka. Danas ova tehnologija pogoni 42% svih web stranica, a u prošlosti taj broj je bio još veći.

Golang je moderan programski jezik dizajniran za brzinu . Osmislili su ga Google-ovi inženjeri Robert Griesemer, Rob Pike i Ken Thompson. Jedan je od najbrže rastućih programskih jezika po popularnosti. Najpoznatiji programski okviri za Golang su Fiber, Martini i Gin Gonic.

JavaScript se interpretirani programski jezik koji se uglavnom koristi za izradu klijentske strane web aplikacija. No u zadnje vrijeme može se sve češće koristi i za izradu poslužiteljske strane u takozvanom MERN/MEAN *stacku*. MERN/MEAN označavaju tehnologije MongoDB, Expres.js, React/Angular i Node.js. Prednost ove grupe tehnologija je u to što su sve bazirane na JavaScriptu pa je time olakšan razvoj buduće da je cijela aplikacija napisana u istom jeziku. Mana je to što JavaScript nije dizajniran za ovakvu primjenu pa su prisutna određena ograničenja.

Slika 4. Popularni programski jezici[4]

2.3. Strana klijenta(Frontend)

Strana klijenta ili frontend služi za prikazivanje podataka koji dolaze s backend-a. Za to se koristi kombinacija 3 jezika: HTML, CSS i JavaScript. Kako bi se ubrzao razvoj modernih web stranica, često se koriste programski okviri pisani u JavaScriptu. Oni omogućuju istovremeno korištenje navedena 3 jezika te dolaze sa vlastitim alatima za pojednostavljivanje izrade web aplikacija. Trenutno su najpopularniji frontend programski okviri Angular, React i Vue.

HTML(Hyper Text Markup Language) je prezentacijski jezik za izradu web stranica. Jezik se sastoji od različitih tagova okruženim uglatim zagradama. Glavna svrha HTML-a je definiranje strukture web stranica. Prva verzija HTML-a predstavljena je 1991. a trenutno je aktivna verzija HTML5 iz 2014.

CSS(Cascading Style Sheets) je programski jezik namijenjen stiliziranju web stranica. Koristi se u kombinaciji sa HTML-om i JavaScript-om. Najčešće korištene funkcionalnosti su pozicioniranje elemenata, izrada animacija, te upravljanje prikazom sadržaja. Trenutno se koristi verzija CSS3. Postoje modernije inačice kao što su SASS/SCSS te one posjeduju neke dodatne funkcionalnosti, ali se one prilikom puštanja aplikacije na produkciju automatski prevode u standardni CSS.

JavaScript je interpretirani programski jezik koji radi prema ECMAScript strandardu. Nastao je 1995 godine, a trenutno je najpopularniji programski jezik te se koristi u preko 97% aktiwnih web stranica. Glavna namjena mu je upravljanje logikom prikazivanja sadržaja na klijentskom sučelju. Zbog sve kompleksnijih zahtjeva korisnika primjena JavaScripta je sve šira te se svaki dan pojavljuju novi alati, programski okviri i biblioteke. Tako je danas moguće koristi JavaScript i na serverskoj strani, za komunikaciju s bazama podataka, za izradu mobilnih aplikacija ili za strojno učenje. Posljedica toga je da je ekosustav JavaScript alata iznimno kompleksan te sklon čestim promjenama. Na slici su prikazani logotipovi samo dijela JavaScript ekosustava.

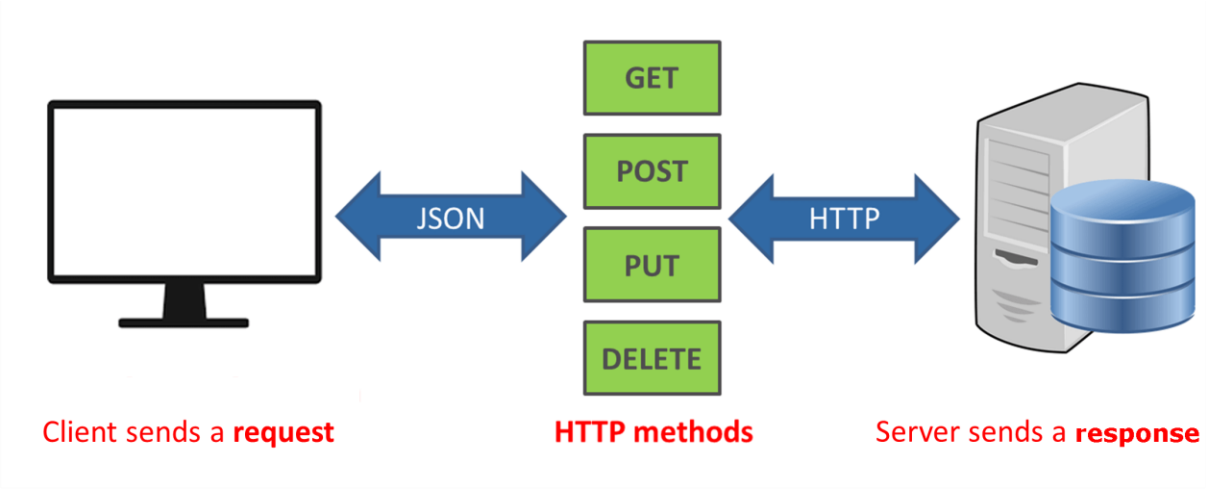
Slika 5. JavaScript ekosustav[5]

Na slici je moguće prepoznati programske okvire Angular, React i Vue, TypeScript koji je ustvari novija varijacija JavaScripta koju je napisao Microsoft, alat npm za upravljanje JavaScript paketima, alati Webpack i Babel za kompajliranje i optimiziranje JavaScript koda za produkciju i mnogi drugi. Također se može vidjeti brzina razvoja novih alata u relativno kratkom razdoblju.

2.4. Aplikacijsko programsko sučelje(API)

API(*Application Programming Interface*) je software koji omogućuje međusobnu komunikaciju između 2 ili više programa preko unaprijed definiranog standarda i u unaprijed definiranom formatu podataka. Za web aplikacije se danas najčešće koristi REST (*Representational state transfer*) standard, a u porastu je i GraphQL standard kojeg je osmislio Facebook.

REST API radi na HTTP protokolu na način da klijent zatraži neku informaciju od poslužitelja putem neke od HTTP metoda(GET, POST, PUT, DELETE). Poslužitelj vraća traženu informaciju klijentu u unaprijed određenom formatu, najčešće JSON ili XML. Proces je ilustriran na slici.

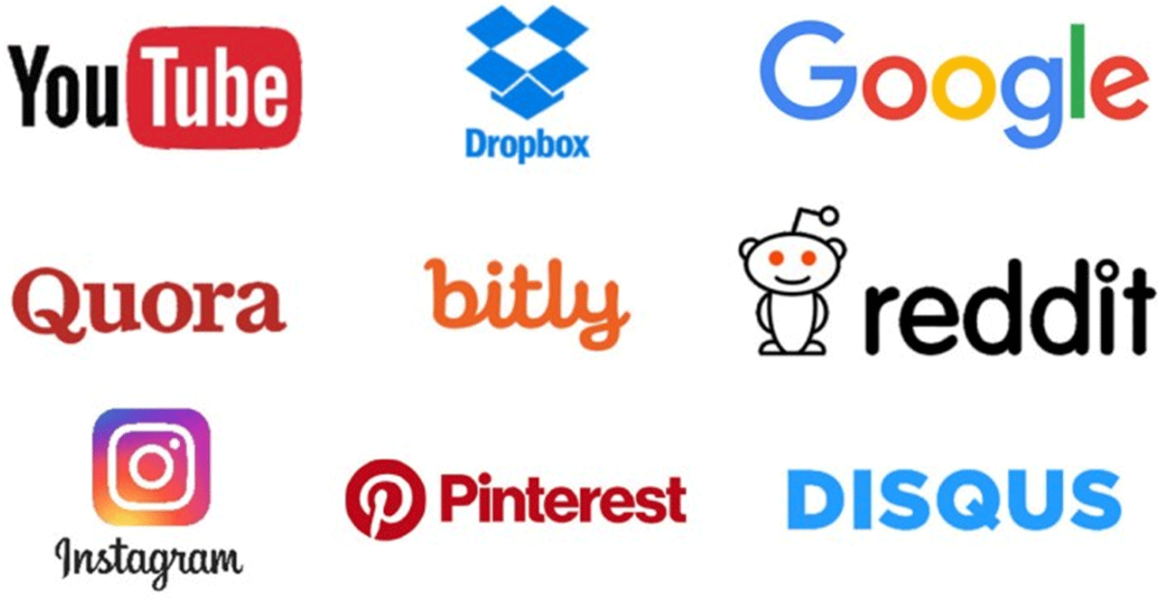


Slika 6. REST API[6]

3. PREGLED NAJČEŠĆE KORIŠTENIH WEB OKVIRA

3.1. Django

Django je programski okvir napisan u Python-u. Nastao je 2003. godine, ali je javno objavljen tek 2005. godine. Jedan je od najpopularnijih programskih okvira za izradu web aplikacija. Glavne karakteristike Djanga su brza izrada aplikacije, velika količina besplatnih biblioteka za proširivanje osnovne funkcionalnosti, solidna brzina nastalih aplikacija te dobra mogućnost ponovne upotrebe istog koda na više aplikacija. Neke od poznatijih web aplikacija koje koriste Django su Instagram, Pinterest, Bitbucket, Disqus, Udemy, Spotify, NASA, National Geographic, Mozila, Dropbox, Quora, Bitly, Reddit i druge.



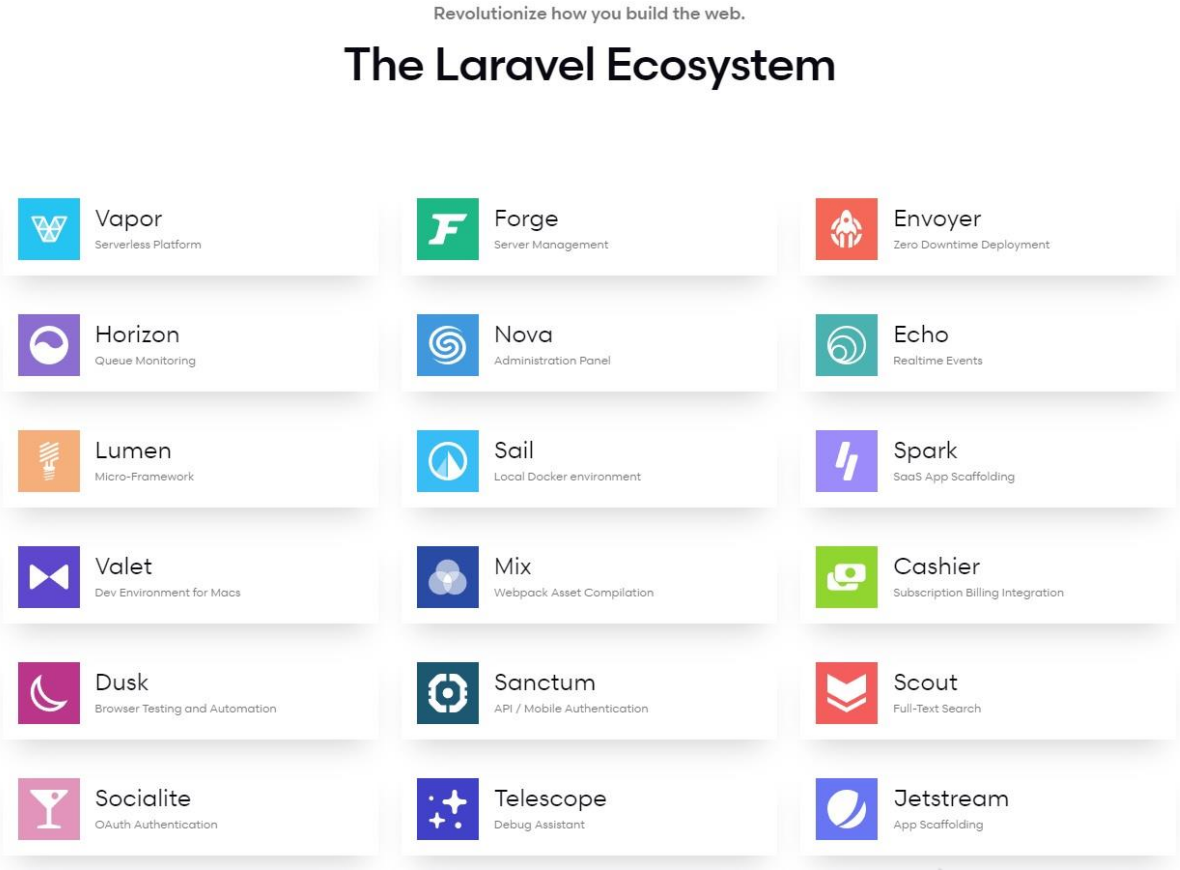
Slika 7. Web aplikacije koje koriste Django[7]

3.2. Express.js

Express.js je programski okvir napisan za Node.js. Node.js je runtime okruženje otvorenog koda za JavaScript. Node.js omogućuje da se kod napisan u JavaScriptu izvršava na strani poslužitelja, a ne samo unutar browsera na strani poslužitelja. Express.js iz tog razloga služi za razvoj serverske strane web aplikacija te je ključni element MERN/MEAN stacka. Neke od poznatijih aplikacija napisanih u ovoj tehnologiji su Twitter, Groupon, GoDaddy, Uber, LinkedIn, Trello, Netflix, PayPal, Clockwise i druge.

3.3. Laravel

Laravel je programski okvir napisan u PHP-u, a baziran je na nešto starijem programskom okviru Simfony. Koristi se za izradu serverske strane web aplikacija. Jedan je od najčešće korištenih programskih okvira zbog jednostavnosti izrade aplikacija i velikog broja dodatnih biblioteka, no u zadnje vrijeme gubi popularnost. Poznatije web aplikacije napisane u Laravelu su 9GAG, Pfizer, BBC, Crowdcube, TourRadar, CBS Interactive, Stitch Labs, Spark Hire, Paxful i GoFundMe.

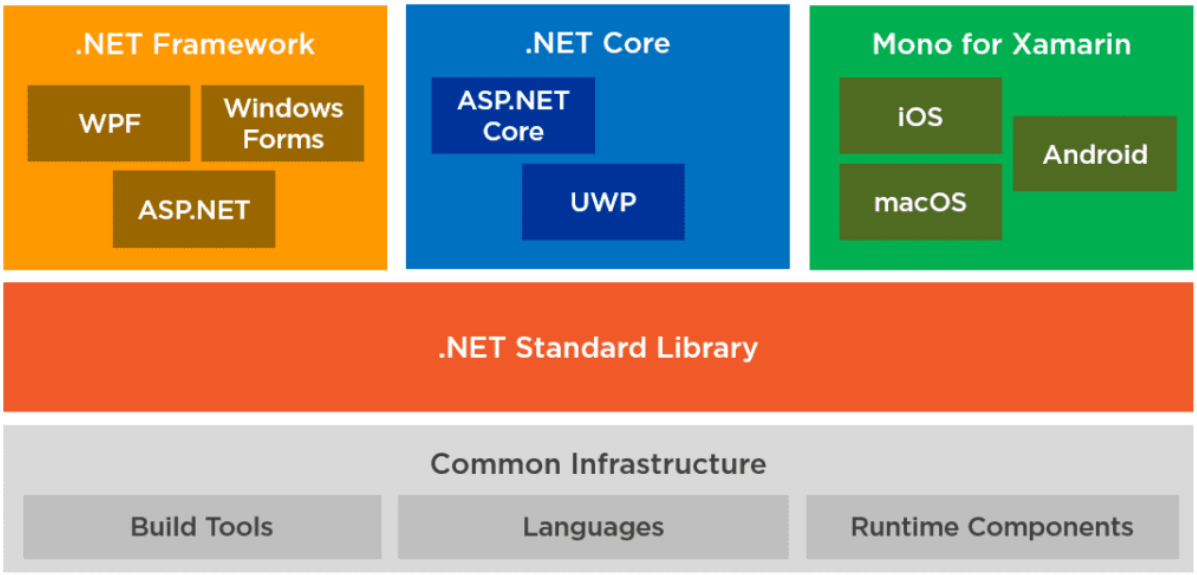


Slika 8. Laravel ekosustav[8]

3.4. ASP.NET Core

ASP.NET je programski okvir napisan u programskom jeziku C# za Microsoft-ovu .NET Framework platformu. Jedan je od najzastupljenijih programskih okvira prisutnih za izradu web aplikacija te ujedno i najstariji na ovoj listi. ASP.NET je samo mali dio .NET ekosustava te uz njega valja još spomenutu .NET Core, Windows Forms, WPF i Xamarin. Pogodan je za izradu većih i kompleksnijih web aplikacija, a najpoznatije web stranice

napravljene pomoću ovog okvira su Microsoft, Dell, Visual Studio, Ancestry.com, Diply.com, Marketwatch.com i druge.



Slika 9. .NET Framework ekosustav[9]

4. IZRADA WEB APLIKACIJE

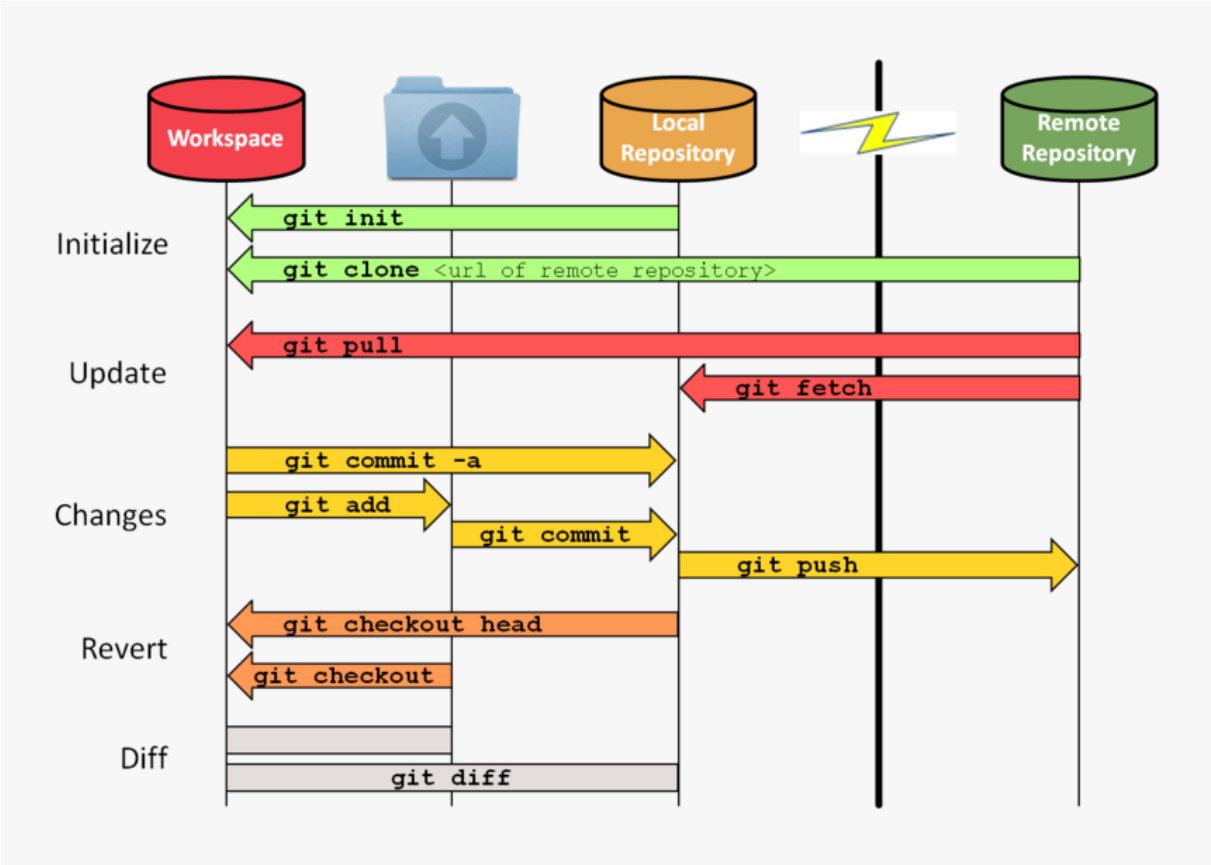
Za izradu web aplikacije odabran je Django programski okvir. Django omogućuje brzu izradu skalabilnih web aplikacija te postoji velik broj besplatnih biblioteka napisanih za ovaj programski okvir. Django je trenutno najpopularniji programski okvir otvorenog koda napisan u Python-u, dok je sam Python trenutno najpopularniji programski jezik. Kao baza podataka koristi se Postgresql, relacijska baza podataka otvorenog koda koja omogućuje relativno jednostavnu integraciju s Django programskim okvirom.

4.1. Korišteni alati

Za izradu modernih web aplikacija potrebno je koristiti različite programske alata kako bi se ubrzao razvoj. Neki od korištenih alata opisani su u nastavku.

4.1.1. Git

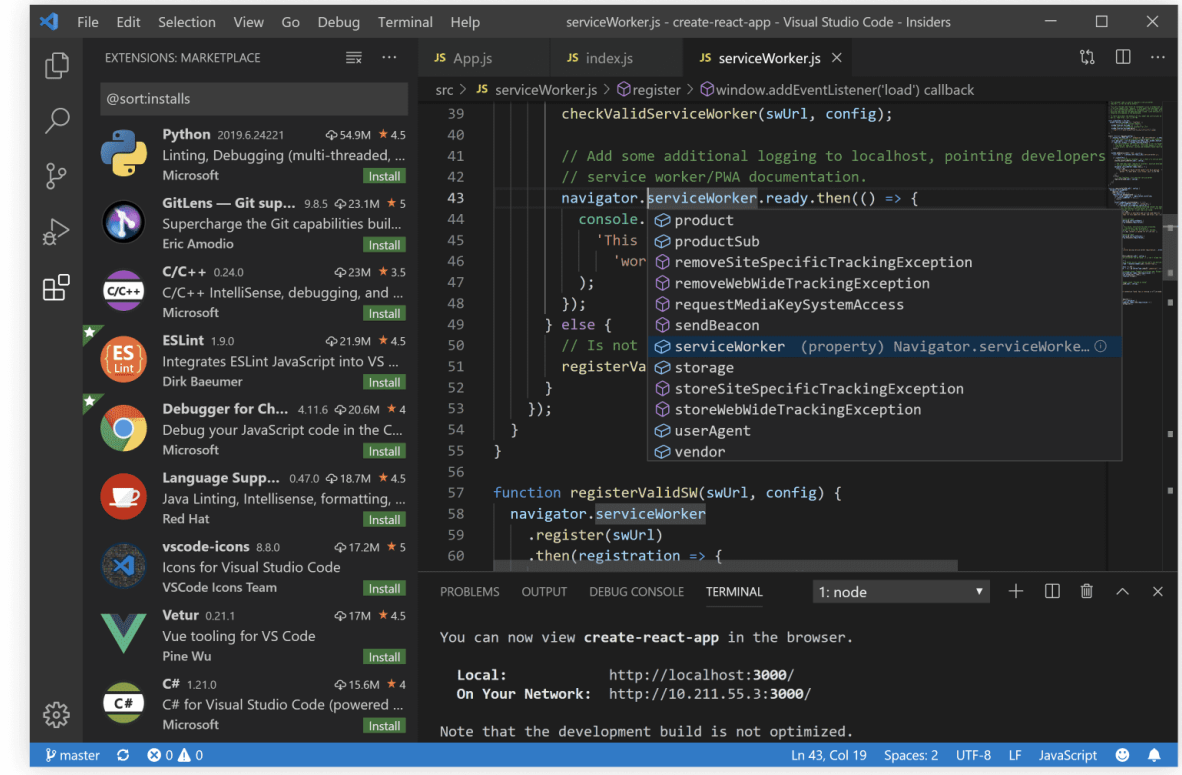
Git je distribuirani sustav za verzioniranje datoteka. Koristi se u programerskim firmama za poboljšanje suradnje između različitih programera. Omogućava veću brzinu razvoja, integritet podataka te potporu za razvoj aplikacija sa nelinearnim tokom, tj. razvoj više paralelnih dijelova aplikacije istovremeno. Git je stvorio Linus Torvalds 2005 godine za razvoj Linux karnela. Postoji više alternativa Git-a, na primjer Subversion, TFVC ili Mercurail, ali Git je daleko najpopularniji i najzastupljeniji. Git funkcionira na sljedeći način: na lokalnom stroju inicijalizira se lokalni Git repozitorij za praćenje promjena pomoću naredbe git init. Nakon što se naprave potrebne promjene na lokalnom kodu potrebno je dodati datoteke čiji se sadržaj želi pratiti pomoću naredbe git add. Zatim je potrebno sačuvati navedene promjene pomoću naredbe git commit te ih poslati na udaljeni repozitorij pomoću naredbe git push. Ostale korisne naredbe su prikazane na slici ispod. Kao udaljeni repozitorij najčešće se koriste platforme kao što su GitHub, GitLab, Bitbucket, SourceForge ili Google Cloud Source Repositories.



Slika 10. Princip rada Git-a[10]

4.1.2. VS Code

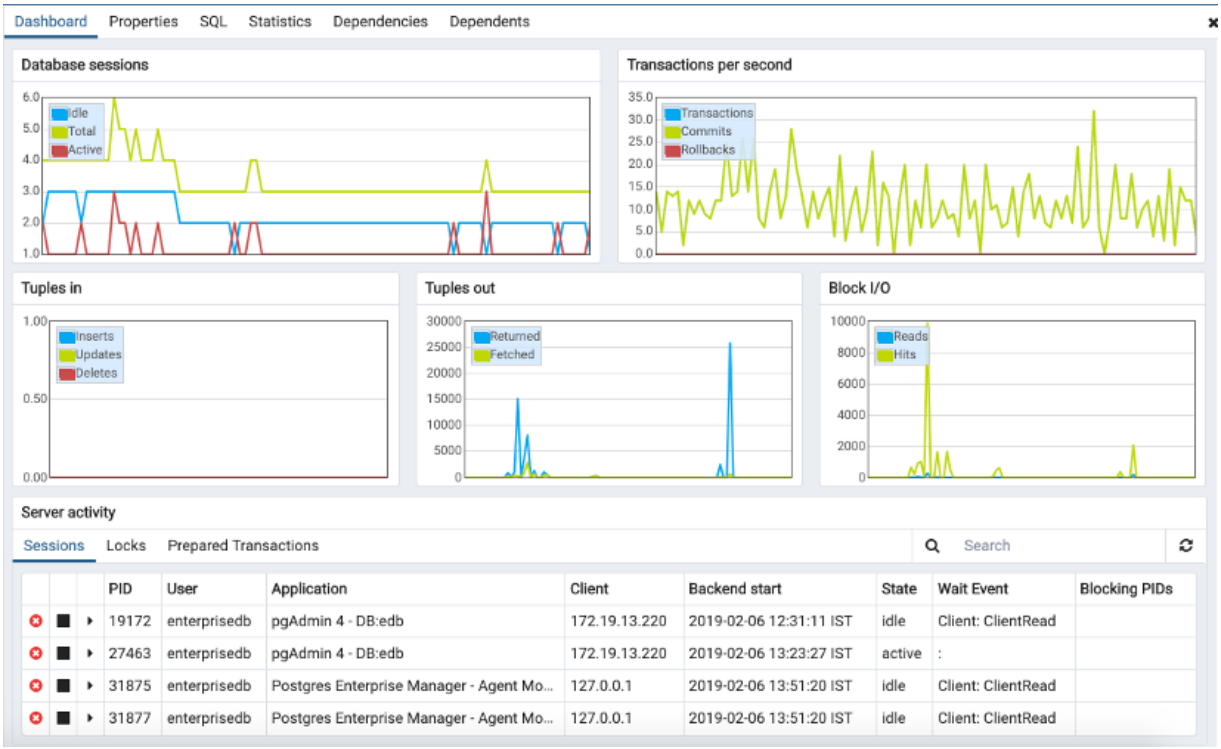
VS Code je najpopularniji editor za pisanje programskog koda. Razvio ga je Microsoft, a temeljen je na poznatom Atom editoru. VS Code je moguće instalirati na svim popularnim operativnim sustavima(Windows, Mac OS i Linux) te omogućava rad sa svim popularnim programskim jezicima. Nudi napredne mogućnosti rada s datotekama, provjeru ispravnosti sintakse, funkcionalnost formatiranja i testiranja programskog koda te instalaciju velikog broja paketa za olakšavanje pisanja programskog koda. Odabran je radi jednostavnosti korištenja, a poznatije alternative su Visual Studio, Atom, Sublime Text, Notepad++ i Pycharm



Slika 11. Sučelje VS Code-a[11]

4.1.3. Postgresql

Postgresql je relacijska baza otvorenog koda. Koristi se u velikom broju komercijalnih web aplikacija zbog svoje brzine i jednostavnosti, te ja najčešće korištena baza podataka za web aplikacije napisane u Django programskom okviru. Dolazi sa vlastitim grafičkim sučeljem zvanom pgAdmin. Koristi se kao baza podataka prilikom razvoja i testiranja aplikacije te kao produkcijska baza podataka.



Slika 12. Sučelje pgAdmin[12]

4.1.4. Virtualno okruženje(Venv)

Prilikom razvoja programa napisanih u Pythonu korisno je odvojiti instancu pythona korištenu za razvoj i onu instaliranu na operativni sustav. Razlog za to je da bi se izbjegli konflikti prilikom instalacije različitih biblioteka i dodataka te da bi se omogućio istovremeni rad sa više verzija Pythona na istom računalu. Za ovu aplikaciju koristi se biblioteka venv koja je dio Python standardne biblioteke. Prvo je potrebno pozicionirati se unutar direktorija u kojem se planira razviti program te kreirati virtualno okruženje sljedećom naredbom:

```
$ python -m venv {venv}
```

Ovu naredbu potrebno je pokrenuti unutar konzole operativnog sustava te će se tako kreirati direktorij sa svim datotekama virtualnog okruženja. Umjesto uglatih zagrada potrebno je napisati ime virtualnog okruženja. Nakon što je virtualno okruženje kreirano potrebno ga je i aktivirati naredbom:

```
$ source venv/scripts/activate
```

Nakon pokretanja ove naredbe u komandnom sučelju će se pojaviti naziv virtualnog okruženja ispred svake sljedeće naredbe. Zatim je potrebno instalirati sve pakete koji su potrebni za izradu web aplikacije.

4.1.5. Pip

Pip je program za instalaciju python biblioteka. Pomoću ovog alata moguće je instalirati sve javno dostupne Python biblioteke sa PyPI(Python Package Index). Za instalaciju bilo koje datoteke potrebno je unutar virtualnog okruženja izvršiti naredbu:

```
$ pip install {ime_biblioteke}
```

Za instalaciju svih potrebnih biblioteka praktičnije je pročitati imena svih potrebnih biblioteka iz neke datoteke. Kod razvoja Python aplikacija nepisano pravilo je da se ta datoteka zove requirements.txt te naredba za njihovu instalaciju izgleda ovako:

```
$ pip install -r requirements.txt
```

Ako se pak biblioteke instaliravaju paralelno s razvojem aplikacije, poželjno je nazive novoinstaliranih biblioteka dodati u datoteku requirements.txt kako bi drugim programerima kasnije bilo lakše raditi na istoj aplikaciji. To se postiže ovako:

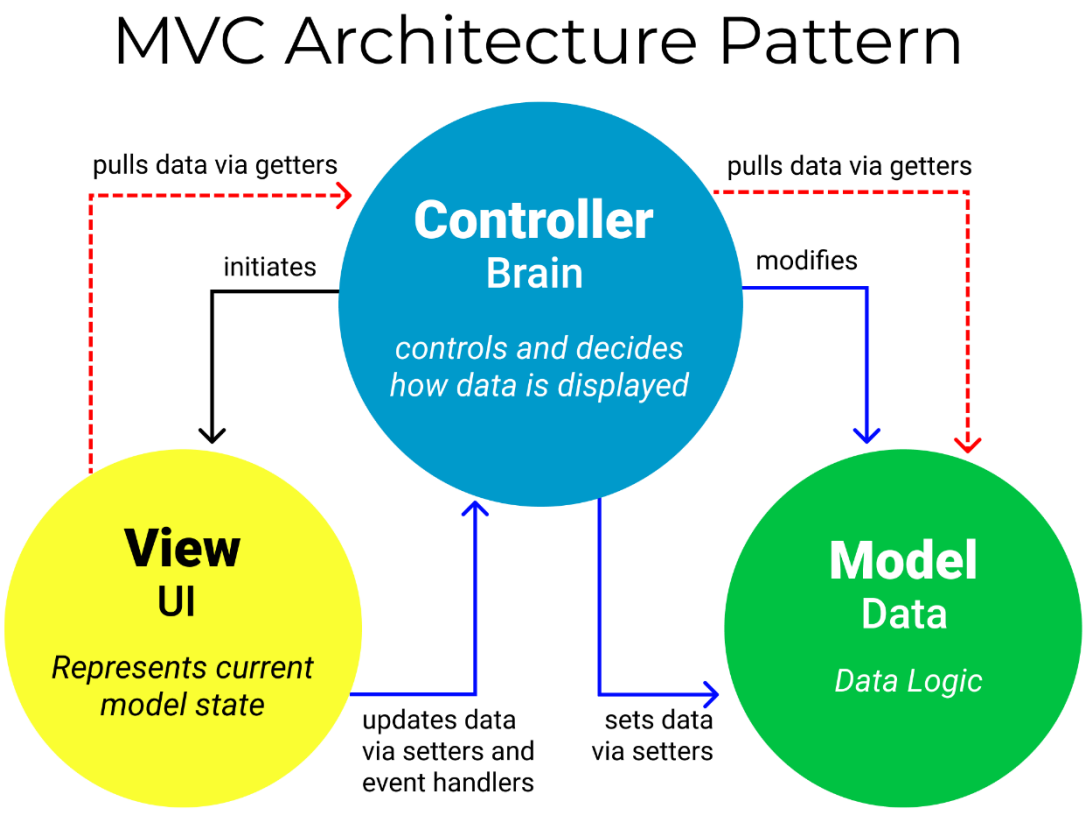
```
$ pip freeze > requirements.txt
```

Za provjeru instaliranih biblioteka unutar virtualnog okruženja koristi se naredba:

```
$ pip list
```

4.2. MVC(Model-View-Controler) obrazac

MVC je obrazac softverske arhitekture. Koristi se u gotovo svim modernim web aplikacijama i programskim okvirima. Valja napomenuti da svaki programski okvir ima svoj način implementacije MVC obrasca, no razlike su uglavnom minimalne. Funkcioniranje MVC obrasca prikazano je na slici ispod.

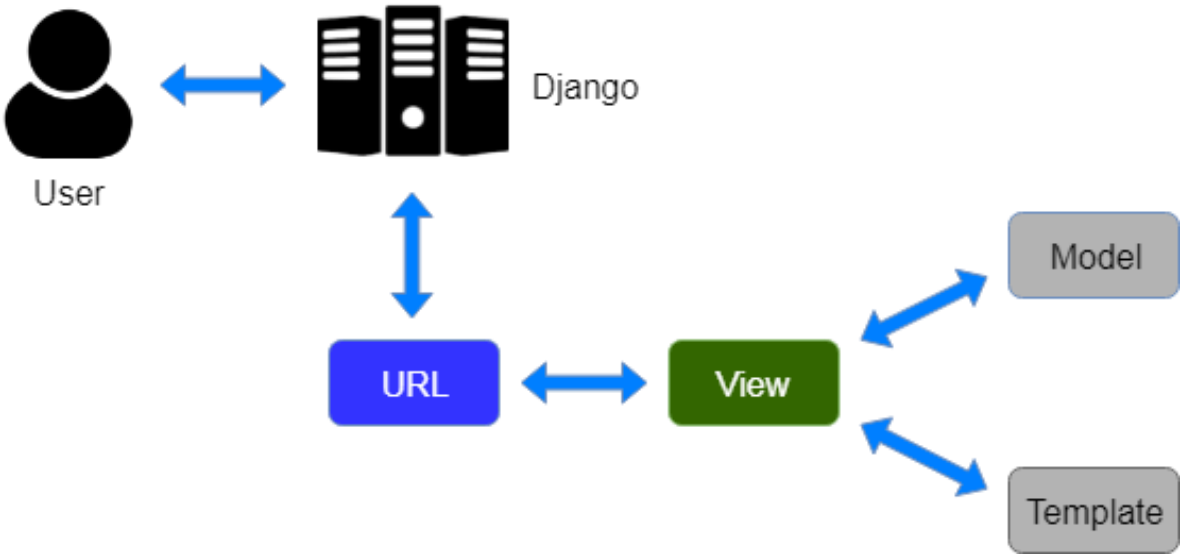


Slika 13. MVC obrazac[13]

Unutar sloja modela definira se struktura podataka aplikacije. Generalno vrijedi pravilo da svaki model predstavlja jednu tablicu unutar baze podataka. Također se definira interakcija između različitih modela te tipovi podataka za svaki atribut modela, tj. tablice. Unutar sloja kontrolera definira se koje će se informacije poslati na korisničku stranu aplikacije te kako će one biti strukturirane. Najčešće se koristi JSON format te aplikacija komunicira sa korisničkom stranom aplikacije putem REST, SOAP ili nekog drugog API-ja. Sloj pogleda(View) definira kako će se podaci koje je kontroler poslao na korisničku stranu prikazati. Ovdje se najčešće radi ili o nekom od JavaScript programskih okvira za korisničku

stranu(Angular, React ili Vue) ili se koristi tzv. *Template* sustav odabranog programskog okvira za serversku stranu. To je slučaj s ovom aplikacijom te se za razvoj sloja pogleda koristi DTL(Django Template Language).

Django ima vlastitu varijantu MVC obrasca koja se naziva MVT(Model-View-Template). Funkcionalno je vrlo slična MVC obrascu. Sloj modela je isti, tj. definira se struktura i tipovi podataka. Sloj kontrolera zvanično ne postoji te njegovu ulogu preuzima sloj pogleda(View). Sloj predložaka(Template) koristi se za prikazivanje informacija na korisničkoj strani.



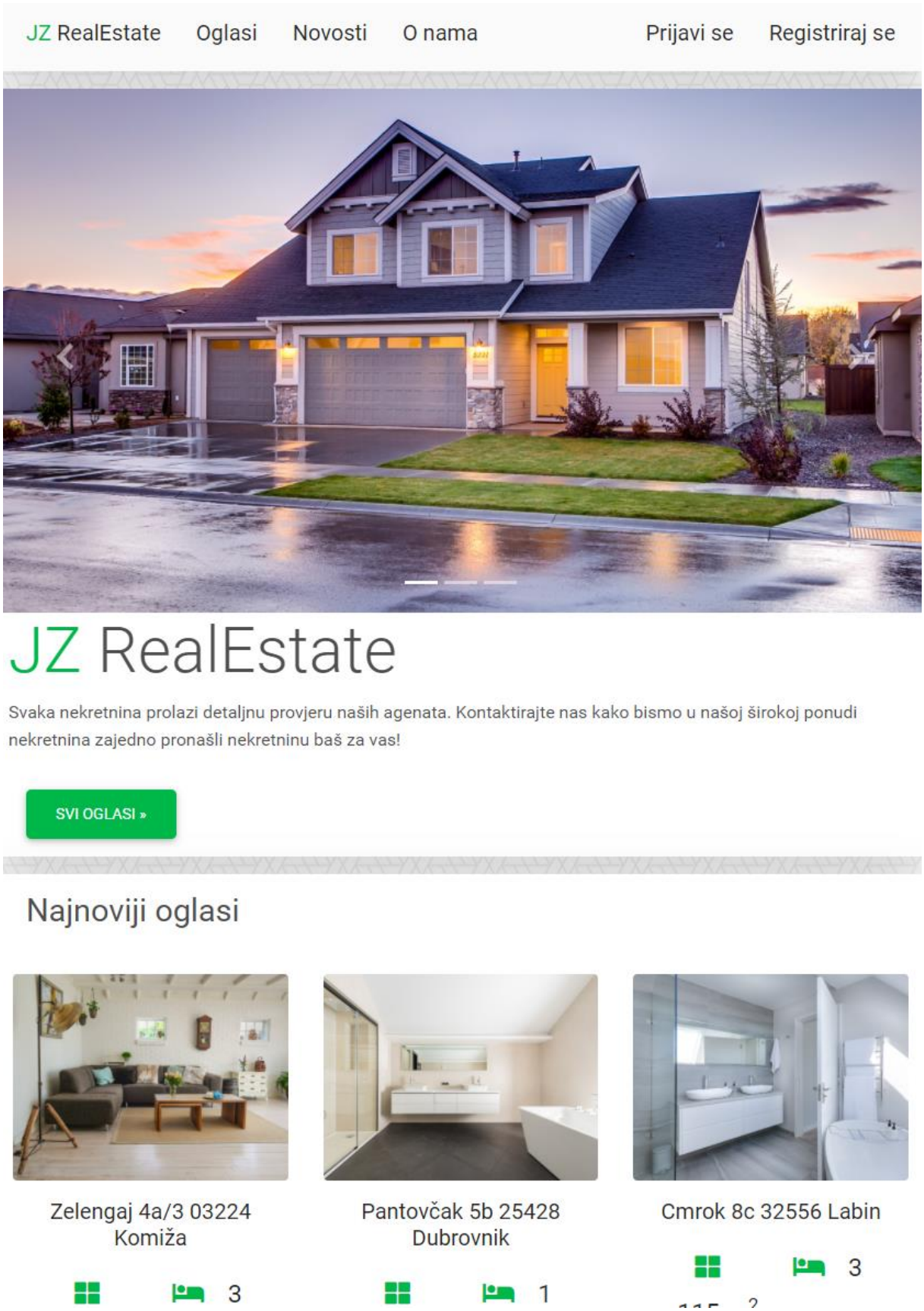
Slika 14. Django MVT obrazac[14]

4.3. Struktura web aplikacije

Web aplikacija sastojat će se od više zasebnih stranica do kojih se može doći preko linkova na navigacijskoj traci. Na lijevoj strani navigacijske trake nalaze se linkovi:

- Naslovnica(JZ RealEstate)
- Oglasi
- Novosti
- O nama

Naslovnica je prva stranica koja se prikazuje korisnika prilikom dolaska na ip adresu web aplikacije. Na njoj se nalaze karusel sa slikama, dodatni link na stranicu Oglasi te sekcija koja prikazuje 3 najnovija oglasa. Izgled naslovnice zajedno s navigacijom prikazan je na slici ispod.



Slika 15. Naslovna stranica s navigacijom

Na stranici oglasi nalaze se kao što i ime govori svi objavljeni oglasi za nekretnine. S lijeve strane nalazi se forma za pretraživanje oglasa, dok se s desne strane nalaze oglasi posloženi jedan ispod drugog. Klikom na ime oglasa dolazi se do posebne stranice na kojojse nalazi detaljni prikaz spomenutog oglasa sa svim informacijama i slikama. Na stranici se prikazuje maksimalno 5 oglasa istovremeno te se ispod njih nalaze linkovi za paginaciju preko kojih se dolazi do oglasa koji trenutno nisu prikazani.

Stranica Novosti služi za prikazivanje novosti tj. članaka vezanih za tržište nekretnina. Na stranici se prikazuju 4 najnovija članka sa označenom kategorijom i kratkim opisom, a ispod njih nalaze linkovi za paginaciju. Klikom na naziv članka korisnik se upućuje na zasebnu stranicu pojedinog članka na kojoj se nalazi puni sadržaj članka.

Na stranici O nama nalazi se statički sadržaj, tj. sadržaj koji nije moguće mijenjati preko administracijskog sučelja. Ideja je da se na ovoj stranici nalaze osnovne informacije o agenciji te osnovne kontakt informacije sa lokacijom prikazanom preko Google Maps-a.

Na desnoj strani nalaze se linkovi vezani za korisnika te njegov profil. Za korisnike koji trenutno nisu prijavljeni na stranicu pokazuju se linkovi:

- Prijavi se
- Registriraj se

Za korisnike koji su prijavljeni pokazuje se padajući izbornik sa korisničkim imenom te sljedeći linkovi:

- Moj profil
- Promijeni lozinku
- Predaj oglas
- Moji oglasi
- Odjavi se

Stranica Moj profil omogućava korisniku promjenu osnovnih informacija, kao što su korisničko ime, email adresa ili broj telefona za kontakt.

Stranica Promijeni lozinku omogućava, kao što i ime govori promjenu lozinke. Od korisnika se traži unos stare, tj. trenutne lozinke te unos nove lozinke 2 puta radi provjere. Klikom na gumb Promijeni lozinku nova lozinka za tog korisnika se sprema u bazu.

Stranica Predaj oglas omogućava prijavljenim korisnicima predaju novih oglasa. Klikom na link korisnika se vodi na formu u koju ubacuje sve potrebne podatke i slike za predaju oglasa.

Stranica Moji oglasi omogućava korisniku lakše snalaženje među vlastitim oglasima te se klikom na link korisnika vodi na novu stranicu na kojoj su izlistani svi njegovi oglasi.

Klikom na link Odjavi se korisniku se prikazuje forma koja ga pita želi li se stvarno odjaviti sa stranice.

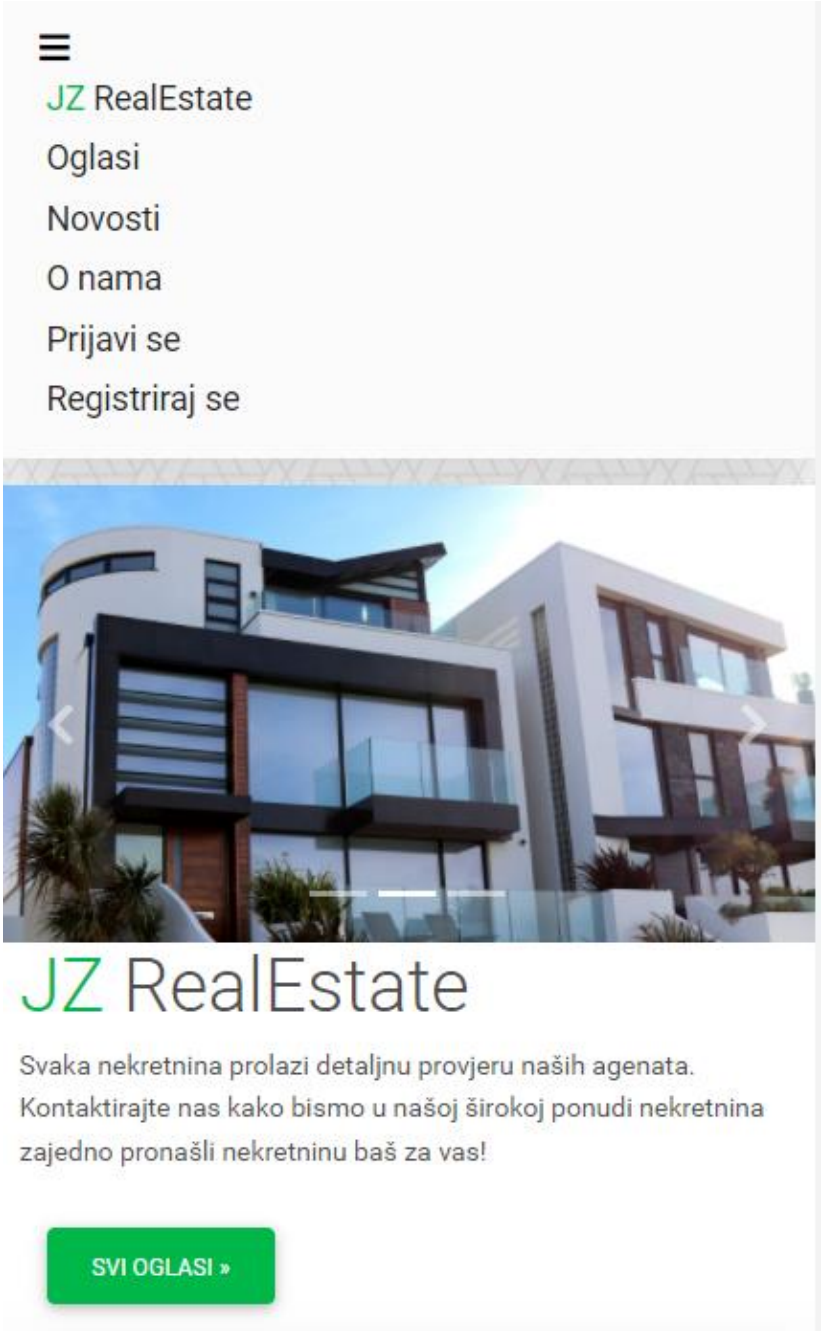
Do navedenih stranica dolazi se preko linkova te svaki link vodi na zasebnu stranicu.

Tablica 1. Popis linkova web aplikacije

Naziv stranice	**Link**	**HTTP request**
JZ RealEstate	http://172.104.241.98/	GET
Oglasi	http://172.104.241.98/listings/	GET
Oglas	http://172.104.241.98/listings/slug/	GET
Predaj oglas	http://172.104.241.98/listings/create/	CREATE
Uredi oglas	http://172.104.241.98/listings/slug/update_listing/	POST
Obriši oglas	http://172.104.241.98/listings/slug/	DELETE
Novosti	http://172.104.241.98/news/	GET
Novost	http://172.104.241.98/news/slug/	GET
O nama	http://172.104.241.98/about/	GET
Prijavi se	http://172.104.241.98/accounts/login/	POST
Registriraj se	http://172.104.241.98/accounts/signup/	POST
Moj profil	http://172.104.241.98/accounts/profile/update_profile/	POST
Promjeni lozinku	http://172.104.241.98/accounts/password/change/	POST
Odjavi se	http://172.104.241.98/accounts/logout/	POST
Admin	http://172.104.241.98/admin/	GET

Pojam slug koji se nalazi na nekim linkovima označava da se do tih stranica dolazi preko varijabilne rute koja ovisi o polju slug koje se nalazi u modelima pojedinih aplikacija. Slug je podatak tekstualnog tipa koji se generira iz naziva objekta kako bi se do svakog objekta koji se nalazi na vlastitoj stranici moglo doći zasebnom rutom. Do stranica koje su uvučene udesno nije moguće doći direktno s navigacijske trake nego je potrebno prvo doći na njima nadređenu stranicu, a tek se onda klikom na jedan od linkova dolazi do navedenih stranica.

Dizajn web aplikacije prilagođen je i mobilnim uređajima, uključujući izgled navigacijske trake, veličine slika i fontova te način prikazivanja oglasa i novosti. Primjer je dan na slici ispod.



Slika 16. Izgled stranice na mobilnim uređajima

4.4. Izrada web aplikacije

Nakon kreiranja virtualnog okruženja(postupak opisan ranije) potrebno je instalirati Django programski okvir pomoću pip-a:

```
$ pip install Django
```

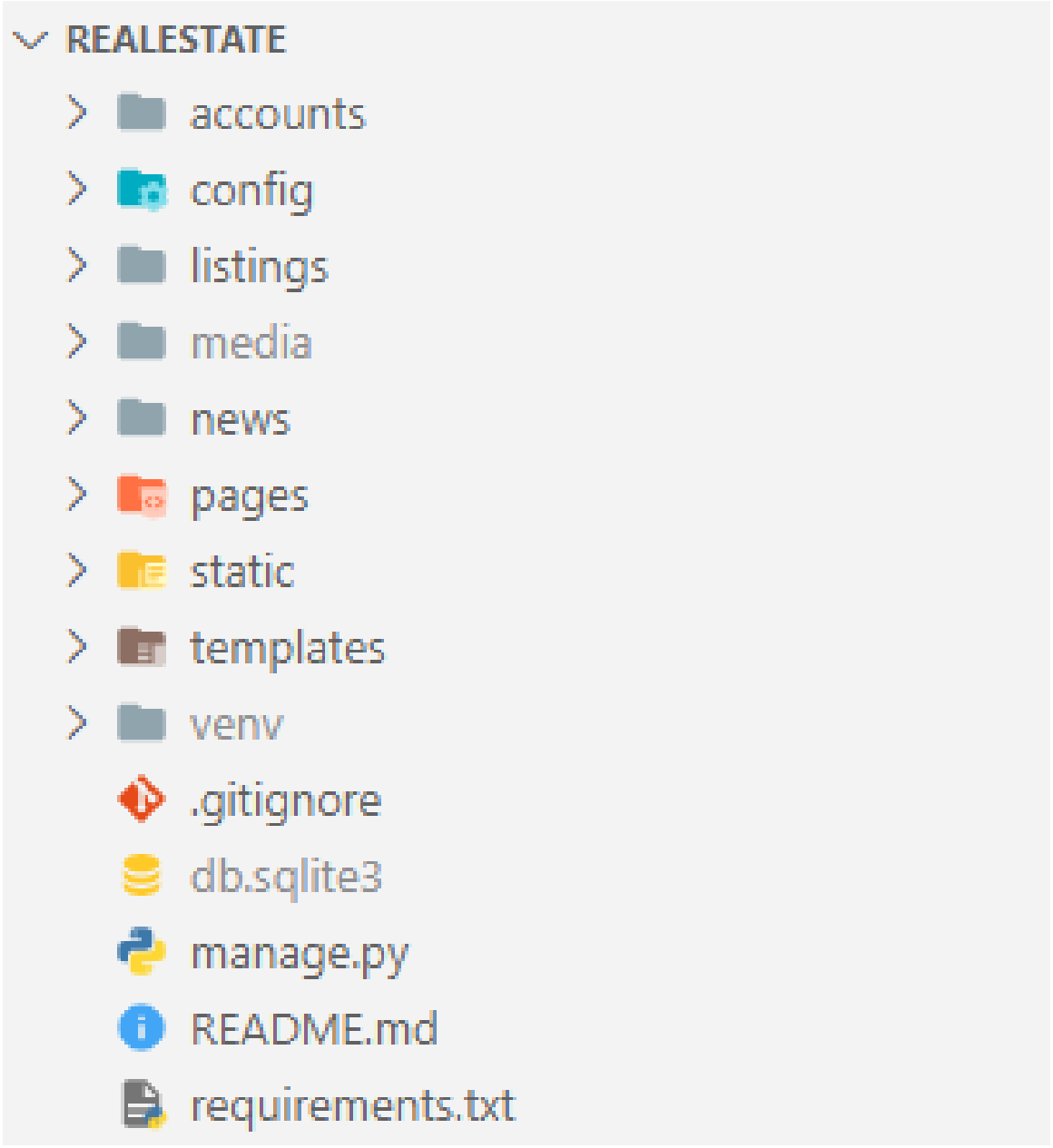
Zatim je potrebno kreirati Django projekt:

```
$ django-admin startproject config .
```

Ova naredba kreirati će glavnu aplikaciju config unutar direktorija s virtualnim okruženjem. Django projekti se sastoje od više izoliranih aplikacija. Aplikacije su ustvari direktoriji sa unaprijed definiranom strukturom i nazivima datoteka. U drugim programskim jezicima i okvirima ovakve strukture se češće zovu moduli, ali ovdje je zadržana nomenklatura iz Djanga. Ovaj pristup omogućava brži razvoj aplikacije te garantira ujednačenu strukturu aplikacije. Za potrebe aplikacije kreirane su još 4 dodatne aplikacije:

- Accounts
- Listings
- News
- Pages.

Nakon kreiranja svih potrebnih aplikacija(modula), datoteka i direktorija, struktura projekta izgleda ovako:



Slika 17. Struktura Django projekta

Direktorij accounts je prije spomenuta aplikacija i u njoj se nalazi sva logika vezana za korisnike web aplikacije. Definiraju se potrebni podatci koji će se pratiti o korisnicima, kao što su njihovo korisničko ime, lozinka, email adresa, ime i prezime, broj telefona, razina korisničkih prava i slično. Definira se izgled i sadržaj formi za upravljanje navedenim podacima, definiraju se sve moguće rute povezane s korisnicima te koji će se podaci o korisnicima prikazivati na klijentskoj strani

Direktorij config je glavni direktorij ove Django aplikacije. U njemu se nalaze 4 bitne datoteke:

- asgi.py
- settings.py
- urls.py
- wsgi.py

Datoteke asgi.py i wsgi.py koriste se tek prilikom postavljanja web aplikacije u produkciju.

Datoteka settings.py sadrži sve postavke koje se koriste unutar Django aplikacije. Neke od postavki koje se definiraju ovdje su podaci za spajanje s bazom podataka, podaci o korištenim aplikacijama i bibliotekama, podaci za autentifikaciju i kriptiranje podataka, podaci o strukturi projekta i drugi.

Datoteka urls.py služe za umrežavanje svih ruta koje Django projekt koristi. Svaka aplikacija ima zasebnu datoteku urls.py

Direktorij listings je aplikacija koja definira upravljanje oglasima. Ovdje je definirana struktura podataka za svaki oglas te su definirane forme za predavanje oglasa. Definirana je i logika pretraživanja oglasa.

Direktorij media koristi se za pohranu slika koje korisnici postave na poslužitelj prilikom predavanja oglasa.

Direktorij news je aplikacija za objavu raznih članaka na web stranicu. Objave mogu raditi isključivo korisnici koji imaju pristup administracijskom sučelju, ali je zato čitanje članaka dopušteno svim korisnicima.

Direktorij pages je aplikacija u kojoj je definirana struktura stranica sa statičkim sadržajem, kao što je naslovna stranica.

Direktorij static sadrži datoteke koje se koriste za prikazivanje sadržaja na stranici. Preciznije tu se nalaze datoteke za stiliziranje stranice(CSS), datoteke korištenih fontova i statičkih slika te dijelovi JavaScript koda.

Direktorij templates funkcionira kao sloj predložaka(Template) u Django MVT strukturi. Ovdje je definiran izgled svake zasebne stranice uz pomoć HTML-a, DTL-a(Django Template Language), CSS-a i JavaScripta.

Unutar direktorija venv nalaze se datoteke virtualnog okruženja i o njima je bilo govora u ranijem poglavlju.

Datoteka .gitignore definira koji će se Direktoriji i datoteke pratiti u sustavu za verzioniranje. Važno je definirati ovu datoteku buduće da se prilikom razvoja web aplikacije na lokalnom stroju često koriste datoteke koje su ili specifične za korišteni operativni sustav ili se mogu kreirati automatski pa nema potrebe da budu unutar git repozitorija. Na taj način se osigurava čišća struktura projekata.

Datoteka db.sqlite3 služi kao priručna baza podataka te se koristi sam tijekom ranog stadija razvoja web aplikacije te prilikom testiranja. Kod puštanja aplikacije u produkcijsko okruženje aplikacija se spaja na pravu bazu podataka.

Datoteka manage.py služi za pokretanje Django aplikacije te je automatski generirana prilikom pokretanja projekta.

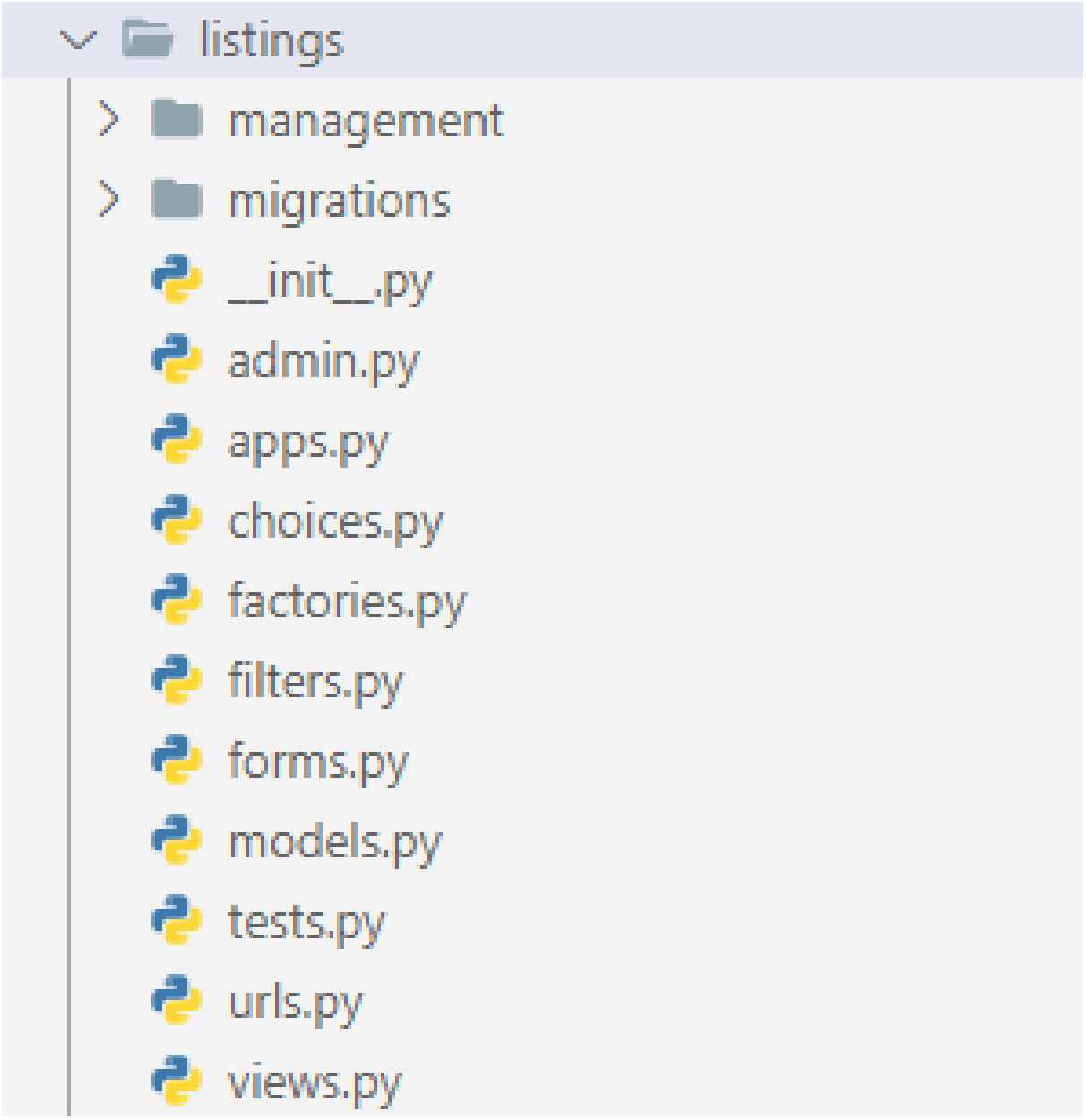
Datoteka README.md sadrži kratki opis web aplikacije i najčešće se koristi za davanje uputa o pokretanju projekta na lokalnom stroju te služi kao svojevrsni priručnik drugim programerima.

Datoteka requirements.txt sadrži nazive i verzije potrebnih Django biblioteka kako bi se pokrenula web aplikacija.

U nastavku je detaljno objašnjena struktura modula(aplikacija) accounts, listings, news i pages.

4.4.1. Modul listings

Modul listings sadrži datoteke i direktorije za upravljanje programskom logikom izrade, pretraživanja i uređivanja objavljenih oglasa za nekretnine.. Struktura aplikacije prikazana je na slici te je identična za sve aplikacije. Modul listings je najkompleksniji modul u projektu te je stoga on odabran za prikaz strukture.



Slika 18. Struktura Django aplikacije

Unutar direktorija management nalaze se komande za upravljanje većom količinom podataka, kao na primjer za kreiranje testnih podataka kako bi se izbjeglo ručno kreiranje svakog korisničkog računa i oglasa.

Direktorij migrations sadrži datoteke migracija koje se koriste za ažuriranje baze podataka. Prilikom svake promjene strukture podataka u modelima, točnije u datotekama models.py pojedinih aplikacija, potrebno je kreirati nove migracije te ih primijeniti na korištenu bazu podataka.

Datoteka __init__.py se automatski kreira prilikom kreiranja svake aplikacije i govori da tretira ovaj direktorij u kojem se nalazi kao modul, tj. omogućuje da se dijelovi koda iz ovog direktorija(aplikacije) koriste u drugim aplikacijama.

U datoteci admin.py definira se koja polja će se prikazivati na administracijskom sučelju te u kojem formatu. Da bi se podaci iz nekog modela prikazivali, potrebno ih je registrirati.

```
from django.contrib import admin
from .models import Listing

@admin.register(Listing)
class ListingAdmin(admin.ModelAdmin):
    class Meta:
        model = Listing

    readonly_fields = ("id", "slug", "created", "modified")
```

Datoteka apps.py definira podatke o aplikaciji koji će se koristiti u drugim dijelovima projekta.

U datoteci choices.py definiraju se konstante koje će se koristiti u aplikaciji, kao što su nazivi županija. Na ovaj način se postiže bolja struktura projekta.

Datoteka factories.py služi za automatsko kreiranje testnih podataka. Definira se struktura svakog objekta koji se želi kreirati za testiranje te se nakon toga pomoću neke od management komandi pokrene kreiranje tih objekata. Može se reći da svaki factory ili tvornica ima ulogu nacrta za stvaranje podataka. U kodu ispod prikazan je način kreiranja testnih podataka.

```
import random
import os

from faker import Faker
from factory.django import DjangoModelFactory, ImageField
from factory import LazyFunction
from factory.fuzzy import FuzzyChoice
from accounts.models import CustomUser

from .models import Listing
from .choices import County, PropertyType

fake = Faker(locale='hr_HR')

def get_username():
    return f'{fake.user_name()}{random.randint(0,999)}'

def get_title():
    return f'{fake.address()}'

def get_price():
    return f'{random.randint(700000, 2500000)}'

def get_area():
    return f'{random.randint(30, 150)}'

def get_int():
    return random.randint(1, 3)

def get_images():
    path = os.path.join(os.getcwd(), 'static/random_images')
    images = os.listdir(path)
    images = [f'static/random_images/{image}' for image in images]
    return images
```

```
class UserFactory(DjangoModelFactory):
    class Meta:
        model = CustomUser

    username = LazyFunction(get_username)
    email = LazyFunction(fake.email)
    password = 'testing321'
    phone = LazyFunction(fake.phone_number)

class ListingFactory(DjangoModelFactory):
    class Meta:
        model = Listing

    title = LazyFunction(get_title)
    author = FuzzyChoice(CustomUser.objects.all())
    location = LazyFunction(fake.address)
    price = LazyFunction(get_price)
    county = FuzzyChoice(x for x, _ in County.get_choices())
    area = LazyFunction(get_area)
    bedrooms = LazyFunction(get_int)
    bathrooms = LazyFunction(get_int)
    car_spaces = LazyFunction(get_int)
    property_type = FuzzyChoice(x for x, _ in
PropertyType.get_choices())
    year = LazyFunction(fake.year)
    photo_main = ImageField(from_path=FuzzyChoice(img for img in
get_images()))
    photo_1 = ImageField(from_path=FuzzyChoice(img for img in
get_images()))
    photo_2 = ImageField(from_path=FuzzyChoice(img for img in
get_images()))
    photo_3 = ImageField(from_path=FuzzyChoice(img for img in
get_images()))
    photo_4 = ImageField(from_path=FuzzyChoice(img for img in
get_images()))
```

```
    photo_5 = ImageField(from_path=FuzzyChoice(img for img in
get_images()))
    photo_6 = ImageField(from_path=FuzzyChoice(img for img in
get_images()))
```

Datoteka filters.py definira po kojim atributima modela će se vršiti pretraživanje oglasa i na koji način.

Datoteka forms.py definira izgled formi za unos i izmjenu oglasa. Definirani su tipovi podataka koji se prihvaćaju u formu te način prikaza forme.

Unutar datoteke models.py definirana je struktura modela oglasa te tipovi podataka koji će biti spremljeni u bazu podataka. Svaki oglas mora imati definiran naziv, adresu nekretnine, cijenu, županiju, osnovne podatke o nekretnini kao što su stambena površina, broj kupaonica/spavaćih soba/garažnih mjesta, godinu izgradnje te nekoliko slika. Kod za ovaj model je prikazan ispod.

```
import uuid
import math

from django.db import models
from django.urls import reverse
from django.conf import settings
from django.utils.translation import gettext_lazy as _

from django_extensions.db.models import TimeStampedModel
from django_extensions.db.fields import AutoSlugField

from .choices import County, PropertyType

class Listing(TimeStampedModel, models.Model):
    class Meta:
        ordering = ["-created"]

    id = models.UUIDField(primary_key=True, default=uuid.uuid4,
editable=False)
```

```
    slug = AutoSlugField(populate_from=["title", "id"])
    title = models.CharField(max_length=255)
    location = models.CharField(max_length=200)
    price = models.PositiveIntegerField()
    county = models.CharField(max_length=50,
choices=County.get_choices())
    area = models.PositiveIntegerField()
    bedrooms = models.PositiveIntegerField()
    bathrooms  = models.PositiveIntegerField()
    car_spaces = models.PositiveIntegerField()
    property_type = models.CharField(max_length=50,
choices=PropertyType.get_choices())
    year = models.PositiveSmallIntegerField()
    photo_main = models.ImageField(upload_to="photos/%Y/%m/%d/")
    photo_1 = models.ImageField(upload_to="photos/%Y/%m/%d/",
blank=True)
    photo_2 = models.ImageField(upload_to="photos/%Y/%m/%d/",
blank=True)
    photo_3 = models.ImageField(upload_to="photos/%Y/%m/%d/",
blank=True)
    photo_4 = models.ImageField(upload_to="photos/%Y/%m/%d/",
blank=True)
    photo_5 = models.ImageField(upload_to="photos/%Y/%m/%d/",
blank=True)
    photo_6 = models.ImageField(upload_to="photos/%Y/%m/%d/",
blank=True)


    author = models.ForeignKey(
        settings.AUTH_USER_MODEL,
        verbose_name=_("Author"),
        on_delete=models.SET_NULL,
        blank=True,
        null=True,
    )


    def __str__(self):
```

```
        return self.title

    def get_absolute_url(self):
        return reverse("listing-detail", kwargs={"slug": self.slug})

    def get_price_euros(self):
        return math.ceil(self.price / (100.0 * 7.51)) * 100

    def get_readable_county(self):
        return self.county.replace("_", "-")
```

Datoteka test.py služi za definiranje automatskih testova. Kod izrade velikih i kompleksnih aplikacija bitno je napraviti testove koji se mogu automatski pokretati kako bi se izbjegle greške u kodu prilikom izmjena koda.

Datoteka urls.py definira sve rute na kojima se mogu dobiti resursi(odgovori poslužitelja) unutar pojedine aplikacije. Svaka aplikacija ima svoju datoteku s rutama, a sve su povezane s istoimenom datotekom urls.py unutar glavne aplikacije config. Za aplikaciju listing rute su definiranu kao što je prikazano u kodu ispod.

```
from django.urls import path
from .views import (
    ListingListView,
    ListingDetailView,
    ListingCreateView,
    ListingUpdateView,
    ListingDeleteView,
    UserListingsView,
)

urlpatterns = [
    path("", ListingListView.as_view(), name="listings"),
    path("<str:username>", UserListingsView.as_view(), name="user-
listings"),
```

```
    path("create/", ListingCreateView.as_view(), name="listing-
create"),
    path("<slug:slug>/", ListingDetailView.as_view(), name="listing-
detail"),
    path(
        "<slug:slug>/update_listing/",
        ListingUpdateView.as_view(),
        name="listing-update",
    ),
    path(
        "<slug:slug>/delete_listing/",
        ListingDeleteView.as_view(),
        name="listing-delete",
    ),
]
```

Unutar datoteke views.py definiran je sloj pogleda(View) u MVC obrascu. Ovdje se definira koji podaci će se prikazati na kojoj ruti te se definiraju dodatne postavke kao što su paginacija i naziv HTML predloška za taj pogled. Programski kod prikazan je ispod.

```
from django.urls import reverse_lazy
from django.shortcuts import get_object_or_404

from django.contrib.auth.mixins import LoginRequiredMixin,
UserPassesTestMixin
from django.views.generic import (
    ListView,
    DetailView,
    CreateView,
    UpdateView,
    DeleteView,
)

from django_filters.views import BaseFilterView
```

```
from django.template.response import TemplateResponse

from .models import Listing
from .forms import ListingForm
from .filters import ListingFilter

class ListingListView(BaseFilterView, ListView):
    template_name = "listings/listing_list.html"
    model = Listing
    paginate_by = 5
    context_object_name = "listings"
    filterset_class = ListingFilter

    def get_context_data(self, *args, **kwargs):
        _request_copy = self.request.GET.copy()
        parameters = _request_copy.pop("page", True) and
_request_copy.urlencode()
        context = super().get_context_data(*args, **kwargs)
        context["parameters"] = parameters
        return context

class UserListingsView(ListView, LoginRequiredMixin):
    model = Listing
    template_name = "listings/user_listings.html"
    context_object_name = "listings"
    paginate_by = 5

    def get_queryset(self):
        return Listing.objects.filter(author=self.request.user)

class ListingDetailView(DetailView):
    model = Listing
    template_name = "listings/listing_detail.html"
```

```
    context_object_name = "listing"


class ListingCreateView(LoginRequiredMixin, CreateView):
    model = Listing
    templete_name = "listings/listing_form.html"
    form_class = ListingForm
    success_url = "/listings/"

    def form_valid(self, form):
        form.instance.author = self.request.user
        return super().form_valid(form)


class ListingUpdateView(LoginRequiredMixin, UserPassesTestMixin,
UpdateView):
    model = Listing
    form_class = ListingForm

    def form_valid(self, form):
        form.instance.author = self.request.user
        return super().form_valid(form)

    def test_func(self):
        listing = self.get_object()
        return self.request.user == listing.author


class ListingDeleteView(LoginRequiredMixin, UserPassesTestMixin,
DeleteView):
    model = Listing
    success_url = "/listings/"

    def test_func(self):
        listing = self.get_object()
        return self.request.user == listing.author
```

4.4.2. Modul accounts

Modul accounts definira logiku prijave i registracijske korisnika. Kao model korisnika koristi se Djangov model *AbstractUser* te je proširen tako da prima podatke o telefonskom broju korisnika. Za ostalu logiku korišten je poznati Django paket *allauth*. Bilo je potrebno redefinirati forme za kreiranje i prijavu korisnika te formu za promjenu korisničkih podataka preko stranice moj profil. Programski kod datoteke forms.py je dan ispod.

```
from django import forms
from django.forms import ModelForm
from django.contrib.auth.forms import UserCreationForm,
UserChangeForm
from .models import CustomUser

class CustomUserCreationForm(UserCreationForm):
    phone = forms.CharField()

    class Meta(UserCreationForm.Meta):
        model = CustomUser
        fields =(
            "email",
            "username",
        )

class CustomUserChangeForm(UserChangeForm):
    class Meta:
        model = CustomUser
        fields = (
            "email",
            "username",
        )

class ProfileForm(ModelForm):
    class Meta:
```

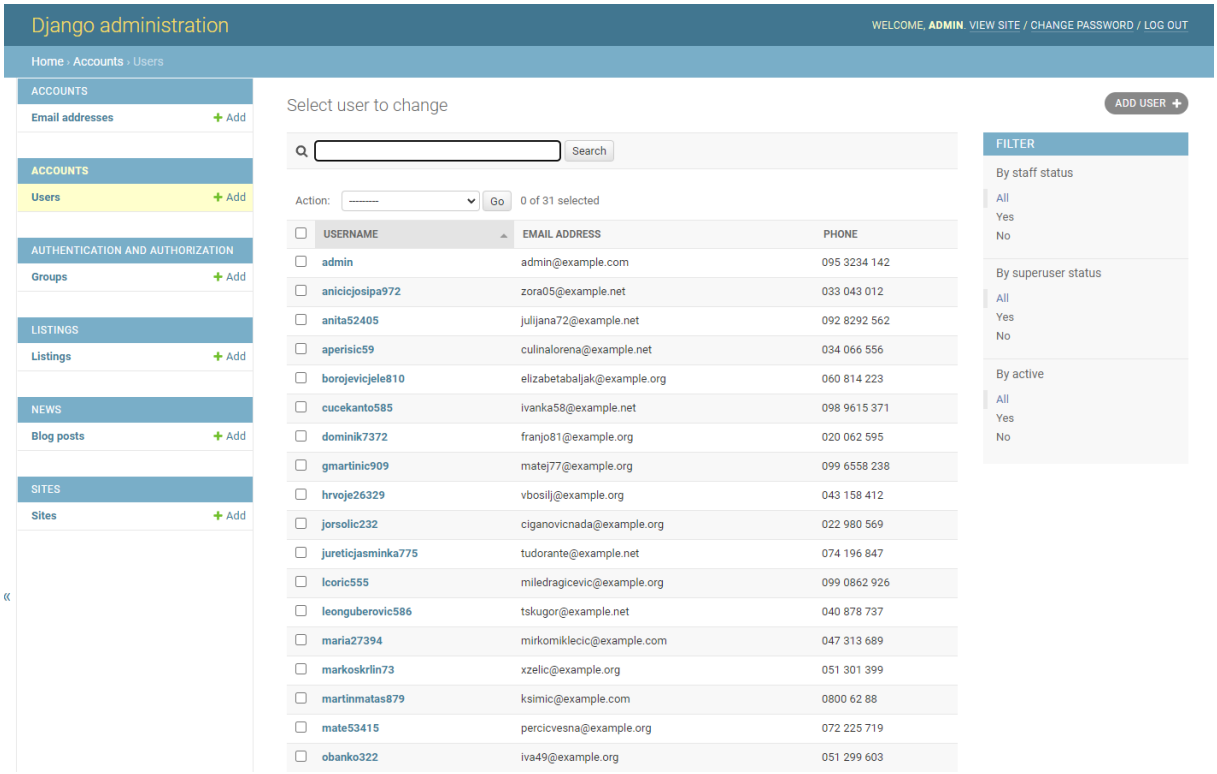
```
        model = CustomUser
        fields = ("email", "first_name", "last_name", "username",
"phone")
```

Da bi se omogućio detaljniji prikaz korisničkih podataka preko administracijskog sučelja napravljene su sljedeća promjene u datoteci admin.py.

```
from django.contrib import admin
from django.contrib.auth import get_user_model
from django.contrib.auth.admin import UserAdmin

from .forms import CustomUserCreationForm, CustomUserChangeForm
from .models import CustomUser

class CustomUserAdmin(UserAdmin):
    add_form = CustomUserCreationForm
    form = CustomUserChangeForm
    model = CustomUser
    list_display = ["username", "email", "phone"]
    fieldsets = (
        (None, {"fields": ("username", "password")}),
        (
            "Personal info",
            {"fields": ("first_name", "last_name", "email",
"phone")},
        ),
        (
            "Permissions",
            {
                "fields": (
                    "is_active",
                    "is_staff",
                    "is_superuser",
                    "groups",
```

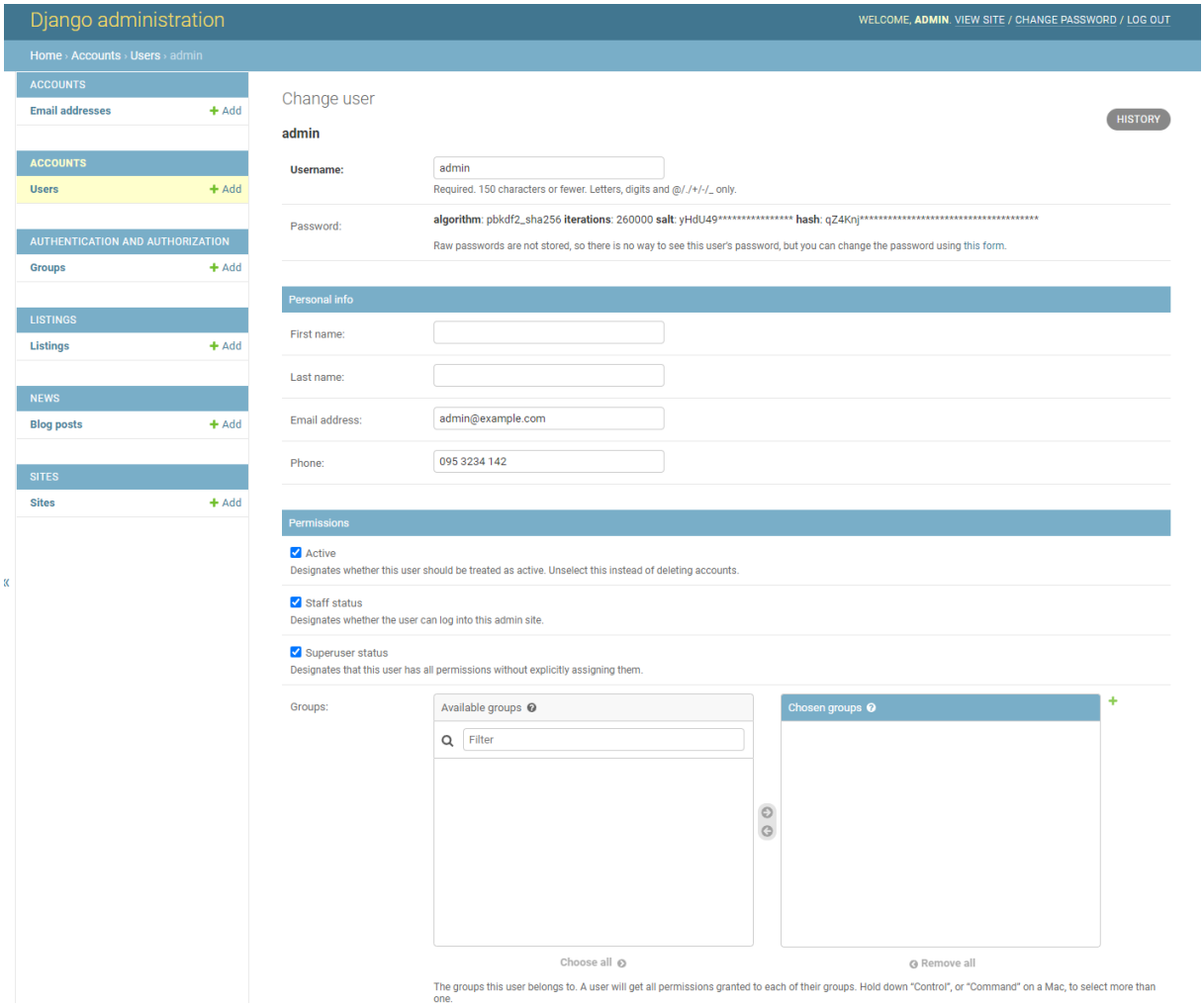
```
                    "user_permissions",
                )
            },
        ),
        ("Important dates", {"fields": ("last_login",
"date_joined")}),
    )

admin.site.register(CustomUser, CustomUserAdmin)
```

Nakon ovih promjena omogućeno je prikazivanje korisničkog imena, email adrese te broj telefona preko administracijskog sučelja.



Slika 19. Prikaz korisnika unutar administracijskog sučelja

Klikom na korisničko ime administrator može vidjeti detaljnije informacije o korisniku kao što su zadnji datum prijave ili korisnička prava. Unutar ove stranice korisničkog sučelja administrator može upravljati korisničkim pravima svakog korisnika bilo da im dodjeljuje pojedinačna prava za neku akciju ili da ih dodaje u predefinirane grupe korisnika.



Slika 20. Administracijsko sučelje za pojedinog korisnika

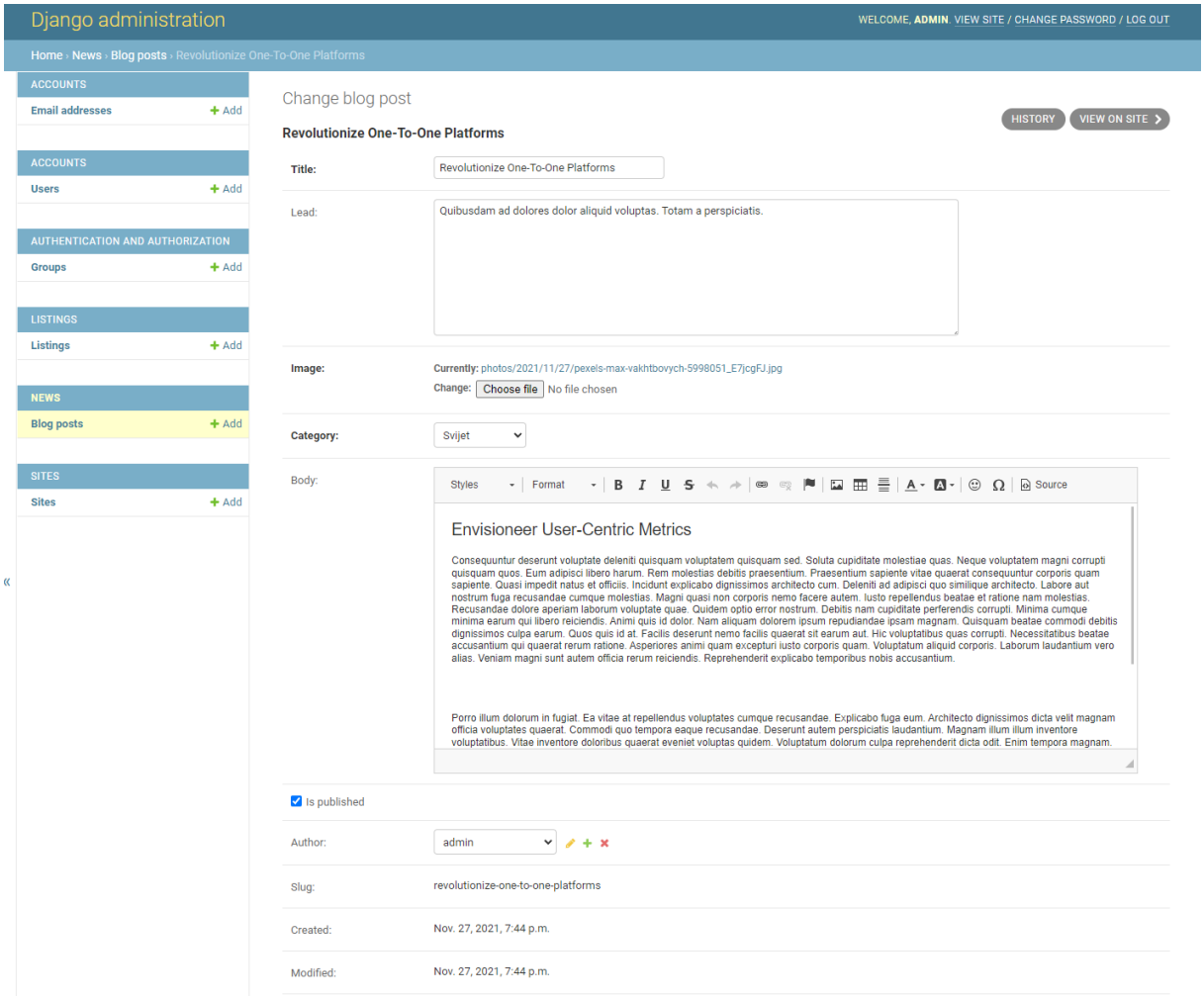
4.4.3. Modul news

Unutra modula news definirana je struktura podataka za pojedine članke objavljene na stranici. Članke mogu objavljivati samo korisnici koji imaju ili status superuser-a ili status staff s eksplicitnim dopuštenjem za objavu članaka: Navedena struktura definirana je unutar datoteke models.py.

```
from django.db import models
from django.urls import reverse
from django.conf import settings
from django.utils.translation import gettext_lazy as _
```

```
from django_extensions.db.models import TimeStampedModel
from django_extensions.db.fields import AutoSlugField

from ckeditor.fields import RichTextField

from .choices import Category

class BlogPost(TimeStampedModel, models.Model):
    title = models.CharField(max_length=200)
    lead = models.TextField(blank=True)
    image = models.ImageField(upload_to="photos/%Y/%m/%d/")
    slug = AutoSlugField(populate_from=["title"])
    category = models.CharField(max_length=50,
choices=Category.get_choices())
    body = RichTextField(blank=True, null=True)
    is_published = models.BooleanField(default=True)
    author = models.ForeignKey(
        settings.AUTH_USER_MODEL,
        verbose_name=_("Author"),
        on_delete=models.SET_NULL,
        blank=True,
        null=True,
    )

    class Meta:
        ordering = ["-created"]

    def __str__(self):
        return self.title

    def get_absolute_url(self):
        return reverse("news-detail", kwargs={"slug": self.slug})
```

Za unos teksta oglasa korišten je paket CKEditor. CKEditor je Django biblioteka koja omogućava korištenje takozvanog WYSIWYG uređivača teksta. Ovaj tip editora omogućava unos stiliziranog teksta koristeći funkcije kao što su bold ili italic, umetanje slika i linkova u tekst, promjenu boja i mnoge druge. Tako unesen tekst će se onda prikazati i na stranici pojedinačnog članka. Važno je napomenuti da je ova funkcionalnost omogućena isključivo unutar administracijskog sučelja te je mogu koristiti samo korisnici od povjerenja, jer u protivnom ova funkcionalnost predstavlja potencijalni sigurnosni rizik za web aplikaciju, budući da je tip podataka koji se unosi u sučelje ustvari HTML, a samim time moguće je unijeti i maliciozni JavaScript kod. Slika CKEditor sučelja prikazana je na slici ispod.



Slika 21. CKEditor sučelje

4.4.4. Modul pages

Unutar modula pages definirane su strukture statičkih stranica Naslovnica i O nama. Za statičke stranice nije potrebno definirati datoteku models.py, ali je potrebno napraviti promjene u datoteci views.py kako bi se definirali pogledi za navedene stranice. Programski kod datoteke views.py dan je ispod.

```
from typing import List
from django.views.generic import TemplateView
from listings.models import Listing

class HomePageView(TemplateView):
    template_name = 'pages/home.html'

    def get_context_data(self, **kwargs):
        context = super(HomePageView,
self).get_context_data(**kwargs)
        context['latest_listings'] = Listing.objects.all()[:3]
        return context

class AboutPageView(TemplateView):
    template_name = 'pages/about.html'
```

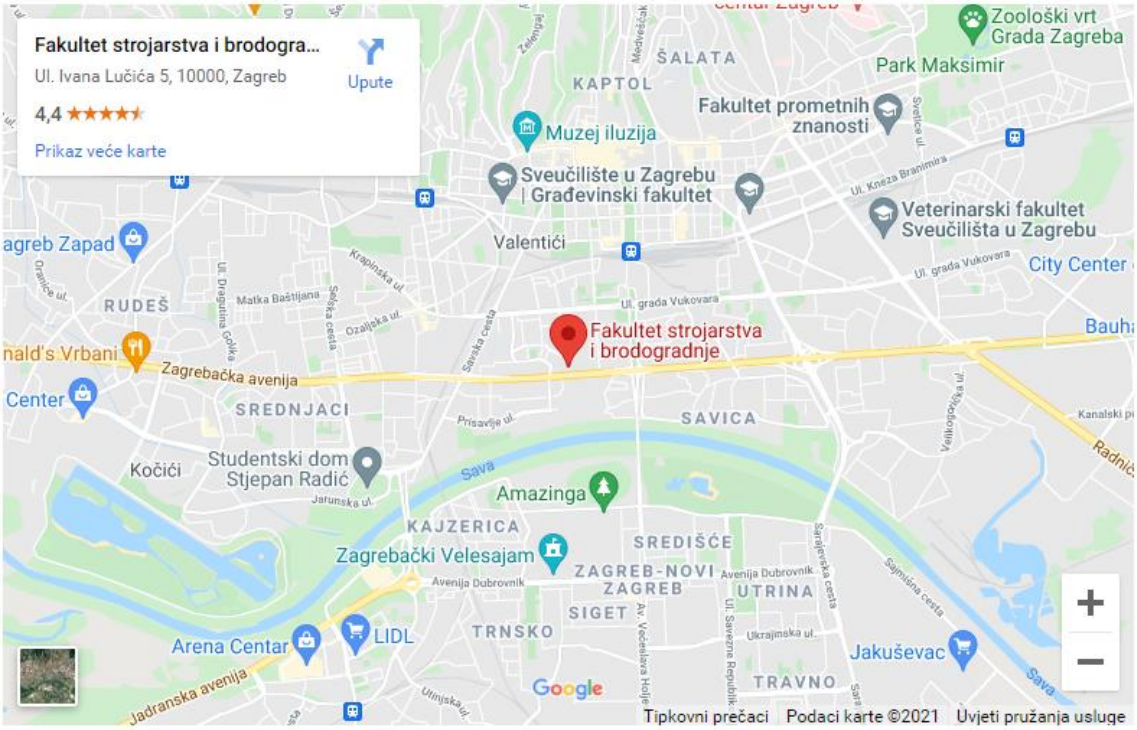
Unutar datoteke urls.py potrebno je definirati rute do navedenih stranica.

```
from django.urls import path

from .views import HomePageView, AboutPageView

urlpatterns = [
    path('', HomePageView.as_view(), name='home'),
    path('about/', AboutPageView.as_view(), name='about'),
]
```

Na stranici O nama nalazi se prikaz adrese agencije te ostale informacije. Kao adresa uzeta je adresa Ulica Ivana Lučića 5. Slika stranice o nama dana je ispod.

Naša lokacija

Prostorije naše Agencije nalaze se u Zagrebu, Ulica Ivana Lučića 5. Radno vrijeme je po dogovoru, a za dolazak u agenciju molimo Vas da se unaprijed najavite.



Naši agenti

Margaret Sotillo Escala

Sed porttitor lectus nibh, Cras ultricies ligula sed magna dictum porta two.

+54 356 945234

Stiven Spilver Darw

Sed porttitor lectus nibh, Cras ultricies ligula sed magna dictum porta two.

+54 356 945234

Emma Toledo Cascada

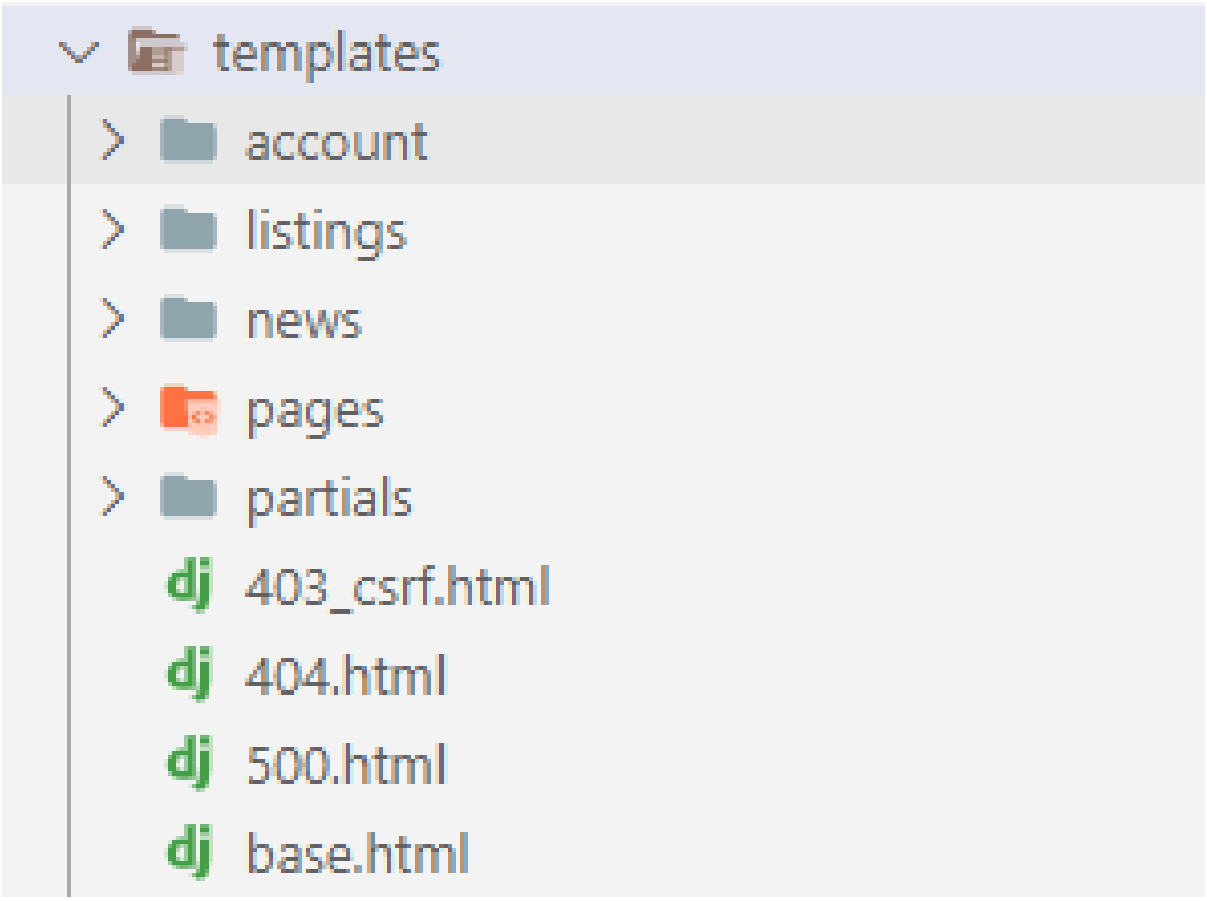
Sed porttitor lectus nibh, Cras ultricies ligula sed magna dictum porta two.

+54 356 945234

Slika 22. Stranica O nama

4.5. Sloj Predložaka(Template)

Na prethodnim stranicama opisana je struktura modela(M) i pogleda(V) Djangove MVT strukture. Sloj predložaka zadužen je za prikazivanje sadržaja na klijentskoj strani. Za razliku od sloja modela i sloja pogleda, koji su napisani u Pythonu, sloj predložaka napisan je u kombinaciji HTML-a, CSS-a i JavaScripta. Struktura podataka unutar direktorija templates dana je na slici ispod.



Slika 23. Struktura podataka direktorija templates

Direktoriji account, listing, news i pages sadrže predloške koji se koriste za kreiranje stranica definiranim u istoimenim aplikacijama.

Datoteka base.html je najvažnija datoteka za sloj predložaka budući da su ovdje definirana struktura ostalih predložaka uključujući korišteni CSS i JavaScript kod iz drugih datoteka pohranjenih unutar direktorija static. Programski kod ove datoteke dan je ispod.

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
    <meta
      name="viewport"
      content="width=device-width,minimum-scale=1,initial-scale=1"
    />
    <title>JZ RealEstate{% block title %}{% endblock title
%}</title>
    <link
      rel="shortcut icon"
      type="image/x-icon"
      href="{% static 'images/favicon.ico' %}"
    />

    {% block css %}
    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="{% static 'css/mdb.min.css' %}" />

    <!-- Font Awesome -->
    <link rel="stylesheet" href="{% static 'css/all.css' %}"/>

    <!-- Lightbox -->
    <link rel="stylesheet" href="{% static 'css/lightbox.min.css'
%}">

    <!-- Custom CSS -->
    <link rel="stylesheet" href="{% static 'css/base.css' %}"/>

    {% endblock %}
  </head>
```

```
  <body>
    <main class="d-flex flex-column min-vh-100">

      <!-- Navbar -->
      {% include 'partials/navbar.html' %}

      <div>
        {% block content %}
        <p>Default content...</p>
        {% endblock content %}
      </div>

      <!-- Footer -->
      {% include 'partials/footer.html' %}

    </main>

    <!-- Bootstrap JavaScript -->
    <script src="{% static 'js/jquery-3.5.1.min.js' %} "></script>
    <script src="{% static 'js/mdb.min.js' %} "></script>

    <!-- 3rd party JavaScript -->
    <script src="{% static 'js/lightbox.min.js' %}"></script>

    <!-- Project JS -->
    <script src="{% static 'js/base.js' %}"></script>
  </body>
</html>
```

Unutar datoteka 403_csrf.html, 404.html i 500.html definiran je izgled stranica do kojih dolazi ako se desi greška u aplikaciji izazvana na serverskoj ili klijentskoj strani. Stranica definirana u datoteci 403_csrf.html koristi se kada dođe do greške prilikom unosa podataka u formu. Stranica definirana u datoteci 404.html prikazuje se kada korisnik zatraži

prikaz neke stranice koja trenutno ne postoji ili je izbrisana. Stranica definirana u datoteci 500.html prikazuje se kada se desi neka greška na serverskoj strani.

Direktorij partials sadrži dijelove web stranice koji se prikazuju na više različitih stranica te se po potrebi uključuju u njih pomoću naredbe include. Ovakve datoteke se često nazivaju i komponente. Unutar ovog direktorija definirane se 3 komponente:

- navbar.html
- footer.html
- pagination.html

Komponente navbar.html i footer.html koriste se na svakoj stranici. Komponenta navbar definira izgled i ponašanje navigacijske trake, dok komponenta footer definira izgled linkova na dnu svake stranice. Programski kod komponente navbar prikazan je ispod.

```
{% load static %}

<nav class="navbar navbar-expand-sm bg-light shadow-lg py-3 mb-3">
  <div class="container">
    <button
      class="navbar-toggler"
      type="button"
      data-mdb-toggle="collapse"
      data-mdb-target="#navbarSupportedContent"
      aria-controls="navbarSupportedContent"
      aria-expanded="false"
      aria-label="Toggle navigation"
    >
      <i class="fas fa-bars"></i>
    </button>
    <div class="collapse navbar-collapse justify-content-between"
id="navbarSupportedContent">

        <div>
            <ul class="navbar-nav">
                <li class="nav-item py-1">
```

```
                    <a class="h5 mx-3 text-dark nav-hover text-
nowrap" href="{% url 'home' %}"><span class="text-success">JZ</span>
RealEstate</a>
                </li>
                <li class="nav-item py-1">
                    <a class="h5 mx-3 text-dark nav-hover" href="{%
url 'listings' %}">Oglasi</a>
                </li>
                <li class="nav-item py-1">
                    <a class="h5 mx-3 text-dark nav-hover" href="{%
url 'news' %}">Novosti</a>
                </li>
                <li class="nav-item py-1">
                    <a class="h5 mx-3 text-dark nav-hover text-
nowrap" href="{% url 'about' %}">O nama</a>
                </li>
            </ul>
        </div>

        <div>
        {% if user.is_authenticated %}
        <div class="dropdown">
            <a
                class="h5 mx-3 text-dark nav-hover dropdown-toggle"
                href="#"
                role="button"
                id="dropdownMenuLink"
                data-mdb-toggle="dropdown"
                aria-expanded="false"
            >
                {{ user.username }}
            </a>
            <ul class="dropdown-menu" aria-
labelledby="dropdownMenuLink"
            >
                <div class="dropdown-divider"></div>
```

```
                <a class="dropdown-item text-dark" href="{% url
'user-profile' %}"
                >Moj profil</a>
                <div class="dropdown-divider"></div>

                <a class="dropdown-item text-dark" href="{% url
'account_change_password' %}"
                >Promjeni lozinku</a>
                <div class="dropdown-divider"></div>

                <a class="dropdown-item text-dark" href="{% url
'listing-create' %}"
                >Predaj oglas</a>
                <div class="dropdown-divider"></div>

                <a class="dropdown-item text-dark" href="{% url
'user-listings' user.username %}"
                >Moji oglasi</a>
                <div class="dropdown-divider"></div>

                <a class="dropdown-item text-dark" href="{% url
'account_logout' %}"
                >Odjavi se</a>
            </ul>
        </div>
        {% else %}
            <ul class="navbar-nav">
                <li class="nav-item py-1">
                    <a class="h5 mx-3 text-dark nav-hover text-
nowrap" href="{% url 'account_login' %}">Prijavi se</a>
                </li>
                <li class="nav-item py-1">
                    <a class="h5 mx-3 text-dark nav-hover text-
nowrap" href="{% url 'account_signup' %}">Registriraj se</a>
                </li>
            </ul>
```

```
        {% endif %}
        </div>
    </div>
  </div>
</nav>
```

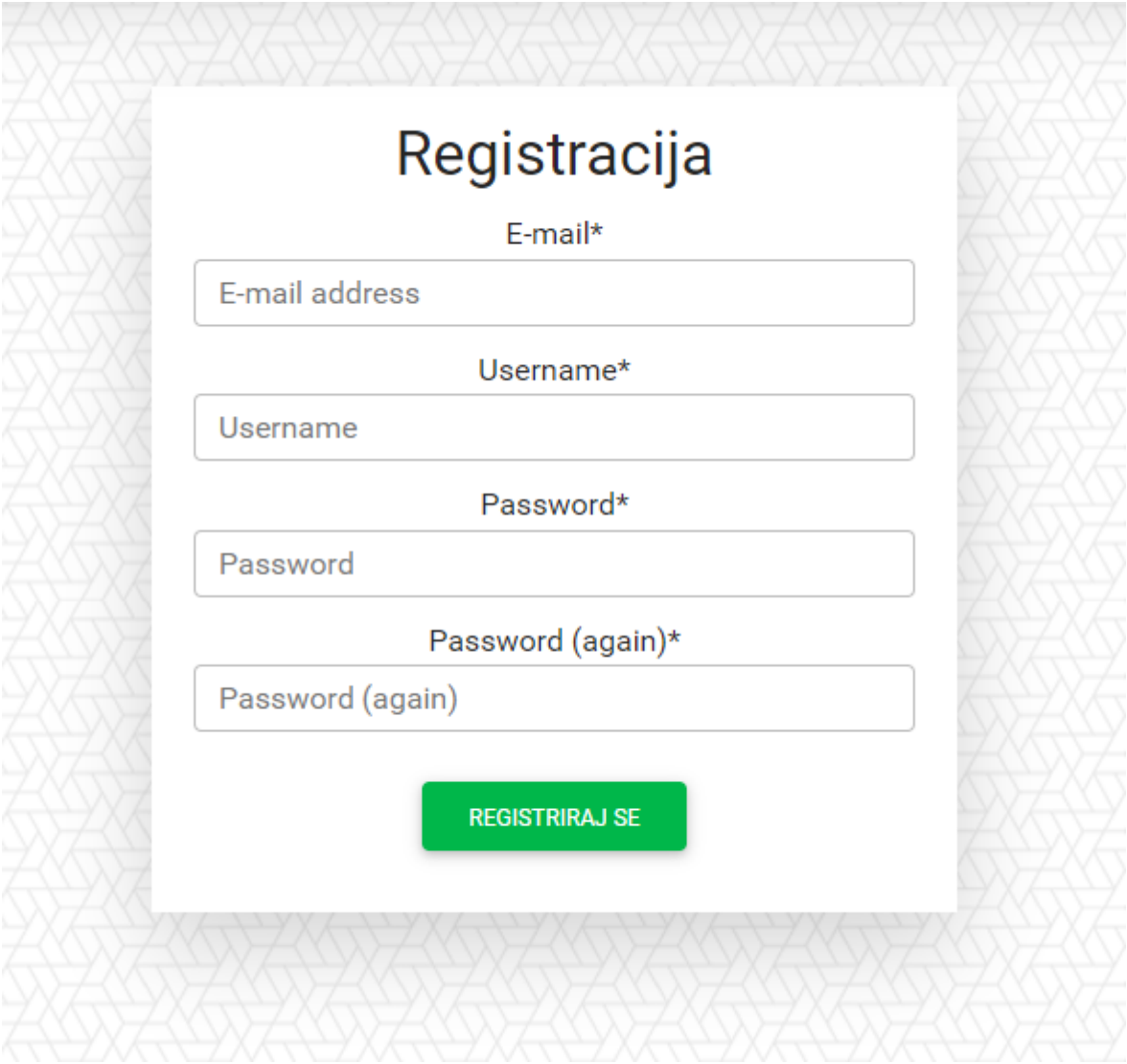
Komponenta pagination koristi za na stranicama Oglasi i Novosti za navigaciju te se mijenja dinamički ovisno o broju elemenata i stranici na kojoj se korisnik nalazi. Programski kod je dan ispod.

```
{% load static %}
<section class="text-center">

  {% if is_paginated %}

    {% if page_obj.has_previous %}
    <a class="btn btn-outline-success mb-2 px-2 py-1"
href="?page=1&{{ parameters }}"><i class="fas fa-angle-double-
left"></i></a>
    <a class="btn btn-outline-success mb-2 px-2 py-1" href="?page={{
page_obj.previous_page_number }}&{{ parameters }}"><i class="fas fa-
angle-left"></i></a>
    {% endif %}

    {% for num in page_obj.paginator.page_range %}
      {% if page_obj.number == num %}
      <a class="btn btn-success mb-2" href="?page={{ num }}&{{
parameters }}"><strong>{{ num }}</strong></a>
      {% elif num > page_obj.number|add:'-3' and num <
page_obj.number|add:'3' %}
      <a class="btn btn-outline-success mb-2 px-2 py-1"
href="?page={{ num }}&{{ parameters }}"><strong>{{ num
}}</strong></a>
      {% endif %}
    {% endfor %}

    {% if page_obj.has_next %}
```

```
    <a class="btn btn-outline-success mb-2 px-2 py-1" href="?page={{
page_obj.next_page_number }}&{{ parameters }}"><i class="fas fa-
angle-right"></i></a>

    <a class="btn btn-outline-success mb-2 px-2 py-1" href="?page={{
page_obj.paginator.num_pages }}&{{ parameters }}"><i class="fas fa-
angle-double-right"></i></a>

    {% endif %}


  {% endif %}


  <p class="text-success">Stranica {{ page_obj.number }} od {{
page_obj.paginator.num_pages }}</p>

</section>
```

5. OPIS FUNKCIONALNOSTI WEB APLIKACIJE

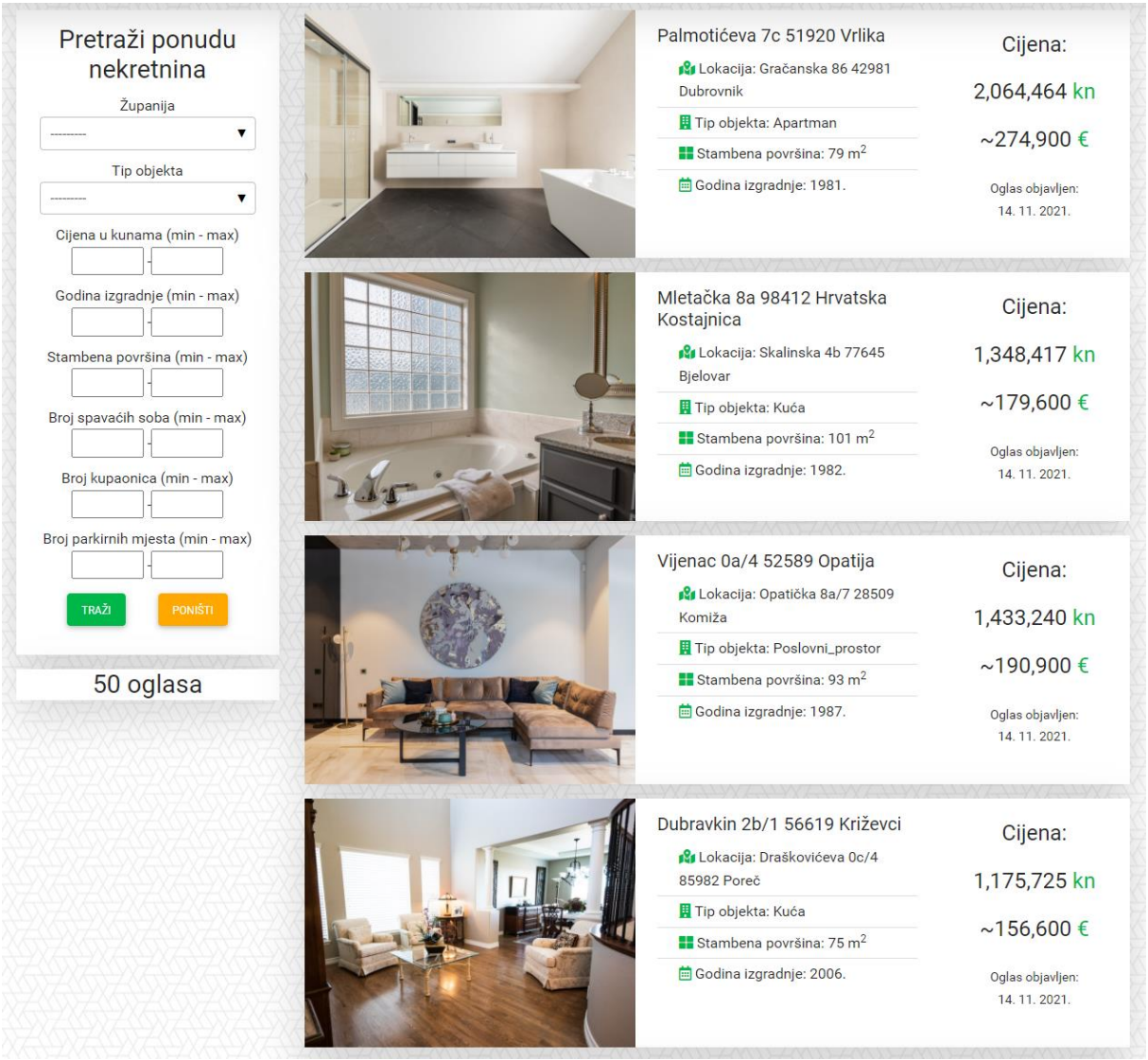
Izrađena web aplikacija pripada hipotetskoj agenciji za prodaju nekretnina. Aplikacija omogućava registraciju novih korisnika, prijavu postojećih korisnika te upravljanjem vlastitim korisničkim podacima, kao što su promjena email adrese, lozinke ili broja telefona. Na slici ispod dan je prikaz forme za registraciju korisnika.



Slika 24. Forma za registraciju korisnika

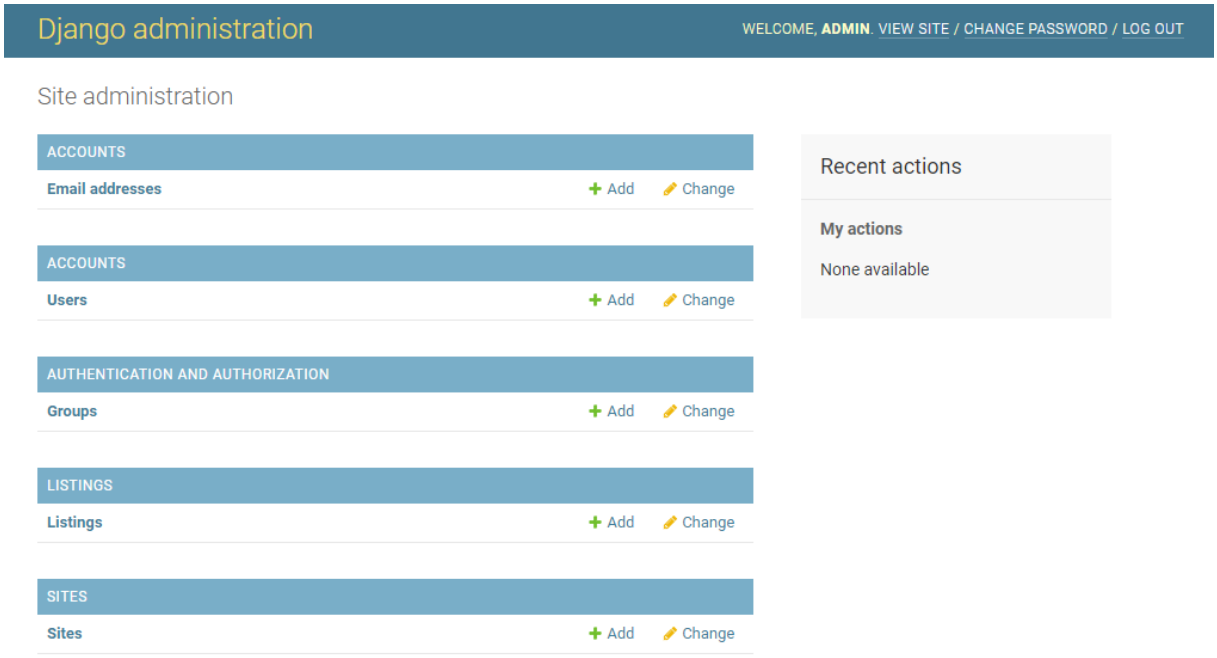
Promjena lozinke kao i ostalih korisničkih podataka odvija se preko zasebnih formi. Samo prijavljeni korisnici mogu objavljivati nove oglase za nekretnine te upravljati vlastitim oglasima

Najbitnija funkcionalnost aplikacije je svakako mogućnost pretraživanja oglasa. Svi korisnici imaju pravo pretraživati nekretnine, a omogućeni kriteriji pretraživanja su Županija u kojoj se nekretnina nalazi, tip nekretnine, cijena nekretnine u kunama, godina izgradnje nekretnine, stambena površina, broj spavaćih soba, broj kupaonica i broj garažnih mjesta s kojima dolazi nekretnina



Slika 25. Prikaz stranice za pretragu nekretnina

Za aplikaciju je izrađeno i aplikacijsko sučelje koje administratorima omogućava upravljanje korisničkim računima i oglasima. Pristup administracijskom sučelju je ograničen te samo korisnici sa statusom administratora(tzv. *superuser*) ili editora(tzv *staff member*) imaju pravo pristupu. Dodatno je moguće upravljanje s pravima svakog korisnika te je moguće dodijeliti prava pristupu administratorskom sučelju ili prava upravljanja sadržajem na administracijskom sučelju.

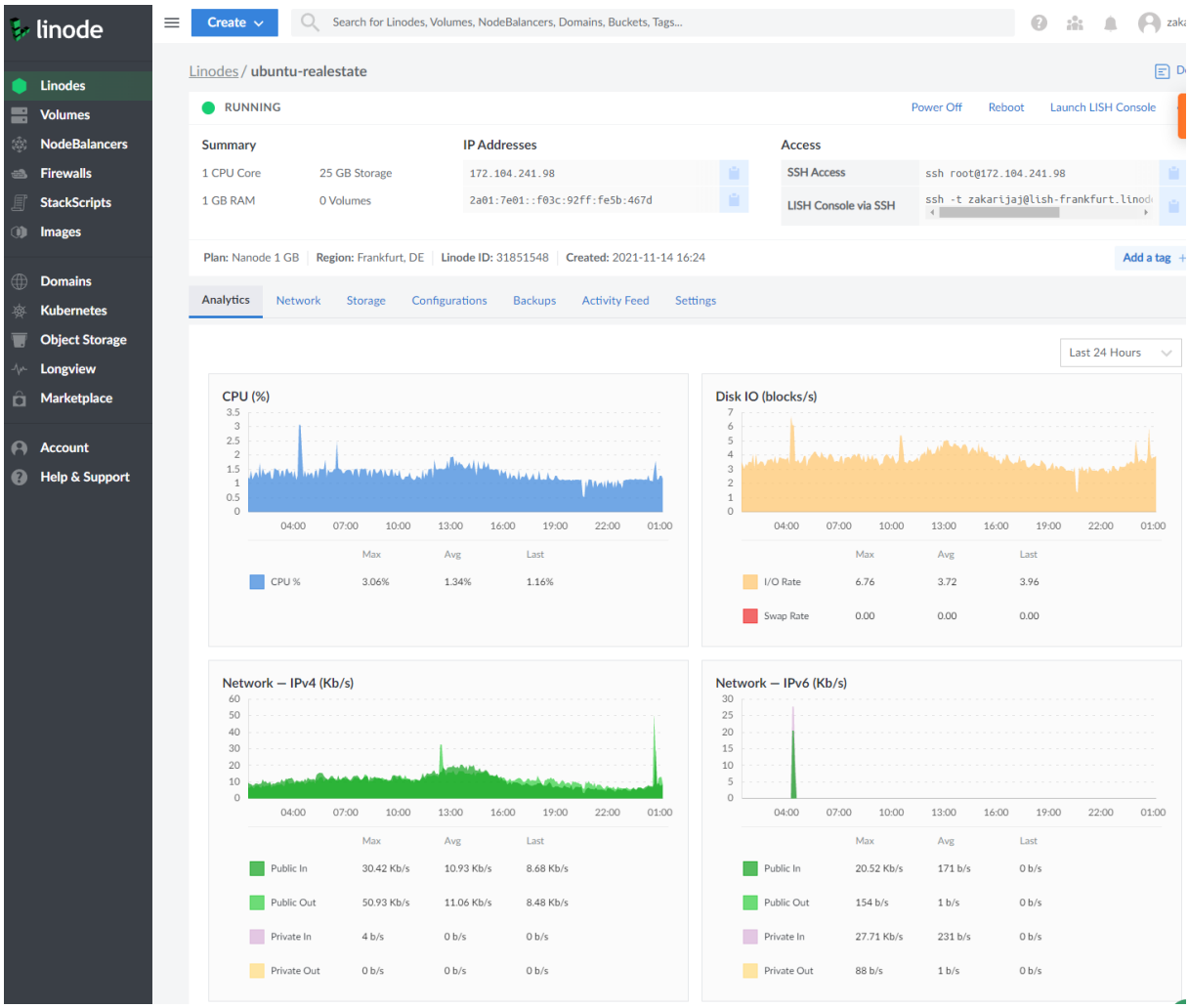


Slika 26. Prikaz administratorskog sučelja

6. POSTAVLJANJE APLIKACIJE NA PRODUKCIJSKO OKRUŽENJE

6.1. VPS(Virtualni privatni server)

Za postavljanje aplikacije na produkcijsko okruženje korištena je usluga VPS(Virtualni privatni server). Pomoću ove usluge moguće je zakupiti dio servera kod udaljenog poslužitelja. Neke od poznatijih kompanija koje nude ovakve usluge su Linode, Digital Ocean i Hetzner. Za potrebe ove aplikacije odabrana je kompanija Linode te usluga Nanode. Nanode je najmanja i najjeftinija usluga te se s njom dobije virtualno računalo s jednom procesorskom jezgrom, jednim gigabajtom RAM-a te 25 gigabajta diskovnog prostora. Na virtualno računalo instaliran je operativni sustav Ubuntu 20.04 LTS.



Slika 27. Linode administracijsko sučelje

6.2. Postavljanje aplikacije na server

Pristup serveru moguć je isključivo putem komandnog sučelja. Proces postavljanja aplikacije se sastoji od nekoliko koraka:

- Ažuriranje postojećih programa i paketa na operativnom sustavu stroja
- Instalacija PostgreSQL baze podataka te kreiranje baze i korisnika koji će koristiti tu bazu iz aplikacije
- Postavljanje vatrozida
- Postavljanje virtualnog okruženja za python
- Prebacivanje programskog koda aplikacije na server
- Kompresija CSS i JavaScript datoteka za brže posluživanje u produkciji
- Povezivanje aplikacije sa bazom podataka
- Instalacija i povezivanje paketa gunicorn sa aplikacijom za serviranje statičkih datoteka i slika
- Instalacija i povezivanje Nginx web servera
- Pokretanje aplikacije

Tek nakon izvršenja navedenih koraka moguće je pristupiti aplikaciji na IP adresi 172.104.241.98.

7. ZAKLJUČAK

U radu je izrađena web aplikacija za oglašavanje i prodaju nekretnina. Korišten je Django programski okvir te PostgreSQL baza podataka. Aplikacija je postavljena na produkcijsko okruženje koristeći uslugu virtualnog privatnog servera tvrtke Linode. Izrađena aplikacija daje dobar uvid u funkcioniranje modernih web tehnologija.

Velik broj korisnika interneta osigurava da su ovakvi tipovi aplikacija sve traženiji budući da omogućavaju ljudima bržu i jednostavnu kupovinu nekretnina, ali uz određene promjene ova aplikacija mogla bi poslužiti za prodaju i oglašavanje bilo kojeg tipa proizvoda i usluga.

S obzirom na ograničeno vrijeme i resurse neke napredne tehnologije kao što je automatsko slanje email-ova nisu mogle biti implementirane.

LITERATURA

[1] Moderne web tehnologije: https://www.davrous.com/2019/01/11/4-ways-to-create-cross-platforms-apps-using-web-technologies/,Pristupljeno: 13. studenog 2021.

[2] Relacijske baze: https://www.omnisci.com/technical-glossary/relational-database, Pristupljeno: 13. studenog 2021.

[3] Nerelacijske baze podataka: https://medium.com/@anuragbhattacharjee/why-choose-mongodb-over-sql-323b8783d4ef, Pristupljeno 13. studenog 2021.

[4] Popularni programski jezici: https://www.asynclabs.co/blog/software-development/how-to-choose-the-right-backend-technology-for-your-app/, Pristupljeno: 13. studenog 2021.

[5] JavaScript ekosustav: https://www.linkedin.com/pulse/javascript-ten-year-challenge-ahamad-milazi/, Pristupljeno: 13. studenog 2021.

[6] REST API: https://phpenthusiast.com/blog/what-is-rest-api, Pristupljeno: 13. studenog 2021.

[7] Web aplikacije u Djangu: https://www.quora.com/Why-is-Django-Web-Framework-so-popular-and-preferred-frameworks-among-all-available-frameworks, Pristupljeno: 14. studenog 2021.

[8] Laravel ekosustav: https://laravel.com/, Pristupljeno 15. studenog 2021.

[9] .NET Framework ekosustav: https://stackify.com/net-ecosystem-demystified/, Pristupljeno: 15. studenog 2021.

[10] Princip rada Git-a: https://git-scm.com/, Pristupljeno 16. studenog 2021.

[11] VS Code: https://code.visualstudio.com/, Pristupljeno 16. studenog 2021.

[12] Postgresql i pgAdmin: https://www.postgresql.org/, Pristupljeno 16. studenog 2021.

[13] MVC obrazac: https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/, Pristupljeno 17. studenog 2021.

[14] Django MVT obrazac: https://www.quora.com/Is-Django-an-MVC-or-an-MVT-framework, Pristupljeno 17. studenog 2021.

PRILOZI

I. CD-R disk