Razvoj back-end dijela web aplikacije pomoću Django razvojnog okvira

Oblaković, Nemanja

VELEUČILIŠTE SUVREMENIH INFORMACIJSKIH TEHNOLOGIJA

STRUČNI PRIJEDIPLOMSKI STUDIJ INFORMACIJSKIH TEHNOLOGIJA

Nemanja Oblaković

ZAVRŠNI RAD

RAZVOJ BACK END DIJELA WEB APLIKACIJE POMOĆU DJANGO RAZVOJNOG OKVIRA

Zagreb, listopada 2024.

Veleučilište suvremenih informacijskih tehnologija
10000 Zagreb, Ulica Vjekoslava Klaića 7

Studij:	Stručni prijediplomski studij informacijskih tehnologija smjer programiranje
Student:	**Nemanja Oblaković**
Matični broj:	2018021

Zadatak završnog rada

Predmet:	Uvod u programiranje
Naslov:	**Razvoj back end dijela web aplikacije pomoću Django razvojnog okvira**
Zadatak:	Opisati specifičnosti i posebnosti Python programskog jezika i Django razvojnog okvira. Opisati proces instalacije Django programskog okvira. Navesti prednosti i nedostatke Django razvojnog okvira. Izraditi web aplikaciju korištenjem Python programskog jezika i Django razvojnog okvira.
Mentor:	Dragana Čulina, pred.
Zadatak uručen kandidatu:	12.10.2023.
Rok za predaju rada:	31.10.2024.
Rad predan:	_____

Povjerenstvo:

Marijan Čančarević, v. pred.	član predsjednik	_____
Dragana Čulina, pred.	mentor	_____
Edmond Krusha, v. pred.	član	_____

Sadržaj

POPIS SLIKA

POPIS TABLICA

POPIS KODOVA

1. UVOD

Razvojem digitalnog svijeta i sve širom primjenom računala, od svakodnevnog života do stručnih istraživanja, programiranje postaje ključna vještina jer omogućuje razvoj potrebnih programa. Usavršavanjem prvih programskih rješenja i nastojanjem da se isprave poteškoće nastale u razvoju modernih programa, razvio se veliki broj programskih jezika i tehnologija koji nastoje optimizirati rad i standardizirati komunikaciju između različitih softvera. Neki od tih jezika postali su specijalizirani samo za primjenu unutar jedne tehnologije, dok se ostali mogu koristiti za razvoj rješenja različitih primjena. Jedan od poznatijih i najčešće korištenih programskih jezika je Python.

Svoju popularnost može zahvaliti svojoj širokoj primjeni, jasnoći i sažetosti koda, jednostavnosti učenja, te mogućnosti primjene i u naprednim programskim rješenjima. Najčešće se koristi za razvoj internet aplikacija, analizu podataka i strojno učenje, automatizaciju i skriptiranje, razvoj desktop aplikacija i drugo. Zbog toga što se kod korištenja programa korisnika ne zamara tehnologijom kojom je razvijen pojedini proizvod, većina osnovnih korisnika nije svjesna kojom se tehnologijom razvijao određeni proizvod. Korisnik će kroz dan boravkom na internetu ili koristeći neku od svojih desktop aplikacija, vrlo vjerojatno koristiti kod napisan u Pythonu ili u nekom od njegovih razvojnih okvira. Python programski jezik i Django razvojni okvir postali su popularni alati među programerima i razvojnim timovima diljem svijeta zbog svoje jednostavnosti, fleksibilnosti i moćnih mogućnosti.

U ovom radu istražit ćemo osnove Python programskog jezika, kao i napredne koncepte i primjene Django razvojnog okvira u izradi internet aplikacija. Kroz izrađenu internet aplikaciju pružit će se uvid u prednosti i nedostatke ovih tehnologija u izradi internet aplikacija, opisati proces instalacije potrebnih programskih paketa, korištenje naredbenog retka u svrhu postavljanja inicijalnih postavki samog projekta te manipulacije projektom kroz naredbeni redak. Opisat će se i prikazati tehnologije koje je potrebno koristiti za razvoj baze podatka i smještaj projekta i baze na server. Potrebno je napomenuti kako se Django razvojni okvir koristi za razvoj backend dijela aplikacije, te će se zbog toga detaljnije obraditi, s ciljem da kroz analizu njihove sintakse, funkcionalnosti i primjera prikaže se kako ovi alati omogućuju brz i učinkovit razvoj sofisticiranih softverskih rješenja.

Arhitektura koju se koristi za razvoj ove internet aplikacije je klijent-server arhitektura koja se zasniva na distribuciji funkcija između dva tipa nezavisnih i autonomnih procesa, servera i klijenta. Klijent je aktivna strana, on je taj koji zahtjeva neke specifične usluge od servera, dok je server pasivan i čeka upite klijenta. Puni smisao ove arhitekture se postiže kada se server i

klijent ne nalaze na istom računalu, iako je moguć rad ove arhitekture i kada su na istom računalu. Kod razdvajanja ove arhitekture na više uređaja ili postavljanjem na server u oblaku, omogućujem se serveru da poslužuje više različitih korisnika u isto vrijeme.

Ova aplikacija ima cilj prezentirati proizvode široj publici putem interneta, te omogućiti subjektu koji će se koristiti njome lakše upravljanje poslovanjem. S obzirom na to da sama po sebi ne može obavljati posao reklamiranja, postavljanje hiperlinka kao poveznice s profilima na društvenim mrežama. Omogućuje lakšu promidžba sadržaja aplikacije i olakšava pronalaženje iste.

2. PYTHON

Python programski jezik je visoko razumljiv i intuitivan jezik koji se često koristi u razvoju različitih vrsta softvera, od web aplikacija do znanstvenih računalnih simulacija. Ovaj jezik ističe se svojom jednostavnom sintaksom koja omogućuje brzo i učinkovito pisanje koda, što ga čini popularnim izborom među programerima svih razina iskustva.

2.1. Uvod u Python

Python je objektno orijentirani, interpretirani programski jezik koji je osmislio Guido van Rossum i prvi put objavljen je 1991. godine. Jedna od glavnih filozofija Pythona je čitljivost koda, što se postiže naglaskom na jasnoći i jednostavnosti. Python podržava razne paradigme programiranja, uključujući proceduralno, objektno orijentirano i funkcionalno programiranje.

2.2. Instalacija Python interpretera

Prije nego što počnete programirati u Pythonu, morate instalirati Python interpreter na svoje računalo. Python interpreter je program koji čita vaš Python kod i izvršava ga. Možete preuzeti Python interpreter s internet stranice python.org i slijediti upute za instalaciju koje su prikladne za vaš operativni sustav.

2.3. Osnovne sintaktičke strukture

Python ima jednostavnu i čistu sintaksu koja olakšava čitanje i pisanje koda. Varijable se deklariraju prilikom dodjele vrijednosti, bez potrebe za eksplicitnim definiranjem tipova podataka. Na primjer, x = 5 definira varijablu x s vrijednošću 5, 5 je tipa int pa je automatski i x int. Python podržava različite tipove podataka, uključujući cjelobrojne brojeve, decimalne brojeve, nizove, liste, n-torka, rječnike itd.

2.4. Kontrola toka programa

Python omogućuje kontrolu toka programa kroz uvjetne izjave i petlje. Uvjetne izjave poput if, elif i else omogućuju izvršavanje određenih dijelova koda samo ako su zadovoljeni određeni uvjeti. Na primjer:

```
if x > 0:
   print("x je pozitivan broj")
```

```
elif x == 0:
    print("x je nula")
else:
    print("x je negativan broj")
```
Kod 1. Kontrola toka (autor)

Petlje poput for i while omogućuju izvršavanje određenog bloka koda više puta. Primjer petlje for:

```
for i in range(5):
    print(i)
```
Kod 2. Primjer for petlje (autor)

2.5. Varijable i tipovi podataka

Varijable u Pythonu se koriste za pohranu podataka. Python ne zahtijeva eksplicitnu deklaraciju varijabli prije njihove upotrebe – dovoljno je dodijeliti im vrijednost. Na primjer:

```
x = 10
y = "Python"
z = 3.14
```
Kod 3. Deklaracija varijabli (autor)

U ovom primjeru, x je cijeli broj (integer), y je niz (string), a z je decimalni broj (floating-point). Python automatski prepoznaje tip podatka na temelju dodijeljene vrijednosti.
Python podržava nekoliko osnovnih tipova podataka:

- **int**: cijeli brojevi (npr. 10, -5)
- **float**: decimalni brojevi (npr. 3.14, -0.001)
- **str**: nizovi (tekstualni podaci) (npr. "Python", "Programiranje")
- **bool**: logički tipovi (npr. True, False)
- **list**: liste su uređene kolekcije podataka (npr. [1, 2, 3])
- **tuple**: nepromjenjive kolekcije podataka (npr. (1, 2, 3))
- **dict**: rječnici koji pohranjuju parove ključ-vrijednost (npr. {"ime": "Ivan", "dob": 25})

Sintaksa Python koda izuzetno je jednostavna i izravna, što olakšava rad s raznim tipovima

podataka.

2.6. Operatori u Pythonu

Python podržava razne operatore za obavljanje matematičkih, logičkih i usporedbenih operacija. Osnovni matematički operatori su:

- **+** (zbrajanje): a + b
- **-** (oduzimanje): a - b
- ***** (množenje): a * b
- **/** (dijeljenje): a / b
- **%** (modul): a % b (ostatak nakon dijeljenja)
- ****** (eksponencija): a ** b (a na b-tu potenciju)
- **//** (cjelobrojno dijeljenje): a // b (rezultat dijeljenja bez decimala)

Osim matematičkih operatora, Python podržava i operatore za usporedbu (==, !=, >, <, >=, <=) te logičke operatore kao što su and, or i not.

2.7. Funkcije u Pythonu

Funkcije su ključni dio programiranja u Pythonu jer omogućuju organizaciju koda u modularne blokove koji mogu obavljati specifične zadatke. Funkcija se definira pomoću ključne riječi def, a može primati ulazne argumente te vraćati rezultat pomoću ključne riječi return. Korištenjem funkcija, kod postaje pregledniji, lakše održiv i ponovno upotrebljiv. Primjer jednostavne funkcije:

```
def pozdrav(ime):
    return f"Pozdrav, {ime}!"

print(pozdrav("Ivan"))
```

Kod 4. Deklaracija funkcije i poziv funkcije (autor)

Ova funkcija pozdrav prima argument ime i vraća personaliziranu poruku. Funkcije mogu biti i složenije, podržavati više parametara, imati zadane vrijednosti parametara te omogućavati fleksibilan način pozivanja, što doprinosi čitljivosti i efikasnosti programskog koda.

3. DJANGO

Django je jedan od najpopularnijih internet razvojnih okruženja u svijetu, razvijen kako bi ubrzao i pojednostavio izradu složenih internet aplikacija. Radi se o razvojnom okruženju otvorenog koda napisanom u Pythonu, a njegova glavna prednost je pružanje "baterija uključenih" (eng. "batteries-included") pristupa, što znači da dolazi s unaprijed ugrađenim funkcionalnostima potrebnim za izradu kompletnih internet aplikacija. Ovaj pristup omogućuje programerima da se više fokusiraju na logiku i poslovne zahtjeve aplikacije, umjesto na tehničke detalje poput sigurnosti, upravljanja sesijama ili povezivanja s bazom podataka.

3.1. Povijest Django frameworka

Django je izvorno razvijen 2003. godine od strane novinara i programera u medijskoj kući "Lawrence Journal-World" u Kansasu. Njihov cilj bio je kreirati alat koji će omogućiti brzo kreiranje internet aplikacija, a da se pritom osigura visoka razina sigurnosti i održavanja. Razvojno okruženje je nazvano po jazz gitaristu Djangu Reinhardtu, a 2005. godine postao je projekt otvorenog koda.

Od tada, Django se brzo proširio među programerima i postao jedan od najkorištenijih Internet razvojnih okruženja na svijetu, zahvaljujući svojoj jednostavnosti, učinkovitosti i skalabilnosti. Danas ga koriste velike kompanije poput Instagrama, Pinteresta i Mozille, ali i mnogi mali razvojni timovi i startupi.

3.2. Karakteristike i prednosti Django frameworka

Django je dizajniran s nekoliko ključnih principa na umu, a svi oni pridonose njegovoj popularnosti među programerima:

1. **Brzina razvoja**: Django omogućuje brzi razvoj aplikacija zahvaljujući velikom broju ugrađenih funkcionalnosti. Mnogi uobičajeni zadaci, poput autentifikacije korisnika, upravljanja sesijama, slanja e-mailova i povezivanja s bazama podataka, već su unaprijed riješeni unutar razvojnog okvira.
2. **Sigurnost**: Django nudi napredne mehanizme sigurnosti koji automatski štite aplikacije od uobičajenih sigurnosnih prijetnji poput SQL injekcija, skriptiranja na strani klijenta (XSS) i prijenosa CSRF napada. Programeri su često zaštićeni od sigurnosnih propusta, čak i bez dodatnog rada na ovom aspektu.
3. **Skalabilnost**: zhvaljujući svojoj modularnosti, Django je izuzetno skalabilan. Velike

aplikacije mogu se lako podijeliti u manje, neovisne module, dok sustav "middlewarea" omogućuje prilagodbu funkcionalnosti u skladu s potrebama projekta.

4. **Kompatibilnost s različitim bazama podataka**: Django koristi ORM (Object-Relational Mapping) za interakciju s bazama podataka, što omogućuje jednostavnu promjenu baza podataka bez potrebe za velikim promjenama u kodu. Osim popularnih baza podataka poput PostgreSQL, MySQL i SQLite, Django također podržava NoSQL baze.

Django je stoga idealan izbor za projekte koji zahtijevaju brzi razvoj, visoku razinu sigurnosti i lako održavanje.

Tablica 1. Prednosti i nedostatci Django frameworka (autor)

Django Framework	
Prednosti	Nedostatci
Brzina razvoja	Pristup “Baterije uključene”
Sigurnost	ORM (Object-Relational Mapping)
Skalabilnost	Monolitan
Kompatibilnost s različitim bazama podataka	Sporija zajednica u uvođenju podrške za novije tehnologije

3.3. Nedostatci Django Frameworka

Iako je Django vrlo popularan i moćan razvojni okvir, ima i nekoliko nedostataka. Prvo, zbog pristupa "baterije uključene" (eng. batteries-included), Django može biti previše složen za jednostavnije projekte jer dolazi s mnogim ugrađenim funkcionalnostima koje možda neće biti potrebne. To može dovesti do opterećenja aplikacije, pogotovo u manjim projektima. Drugo, Djangov ORM (Object-Relational Mapping) i način na koji upravlja bazama podataka nisu uvijek najučinkovitiji za složene upite, što može ograničiti performanse kod većih aplikacija. Treće, Django je prilično monolitan, što znači da može biti manje fleksibilan od "lakših"

razvojnih okvira kada su u pitanju specifične prilagodbe na razini sustava. Na kraju, iako je dokumentacija iznimno dobra, zajednica može ponekad biti sporija u uvođenju podrške za najnovije tehnologije u usporedbi s nekim drugim razvojnim okvirima.

3.4. Instalacija i postavljanje Django projekta

Jedna od glavnih prednosti Django razvojnog okvira je jednostavnost postavljanja novog projekta. Da biste započeli raditi s Djangom, prvo je potrebno instalirati Python (ako već nije instaliran) i Django razvojni okvir. Možete provjeriti je li Python instaliran korištenjem naredbenog retka i naredbe python –version. Ukoliko ova naredba ne radi, mora se dodati Python u Windows PATH.

```
python --version
```

Slika 1. Provjera instalirane verzije Pythona (W3Schools, 2024.)

3.4.1. Instalacija Djanga

Preporučljivo je imati virtualno okruženje za svaki Django projekt, a jedan način za to je venv, paket uključen u Python. Više o postavljanju samog virtualnog okruženja bit će prikazano u praktičnom radu.

Korištenjem naredbenog retka, koji smo koristili za provjeru instalirane verzije Pythona, možemo instalirati najnoviju verziju Djanga uz pomoć alata pip, Pythonovog sustava za upravljanje paketima. Za to nam je potrebna sljedeća naredba.

```
pip install django
```

Slika 2. Naredba za instalaciju Djanga (autor)

Nakon završene instalacije Djanga, možemo napraviti provjeru uspješnosti instalacije i odmah saznati o kojoj je verziji riječ unosom naredbe

```
django-admin --version
```

Slika 3. Naredba za provjeru verzije Djanga (autor)

3.4.2. Kreiranje projekta i pokretanje servera

Nakon instalacije Djanga, spremni ste kreirati novi projekt. Proces započinje kreiranjem direktorija projekta pomoću naredbe django-admin startproject.

```
django-admin startproject moj_projekt
```

Slika 4. Kreiranje Django projekta (autor)

Ovo će generirati strukturu direktorija s osnovnim datotekama potrebnim za pokretanje Django projekta, uključujući manage.py, glavnu skriptu za upravljanje projektom. Kako biste provjerili je li vaš projekt ispravno postavljen, možete pokrenuti ugrađeni Django razvojni server s pomoću naredbe python manage.py runserver.

```
python manage.py runserver
```

Slika 5. Pokretanje Django servera (autor)

Ova naredba će pokrenuti lokalni server i pružiti vam URL (obično http://127.0.0.1:8000/) na kojem možete pristupiti svojoj aplikaciji u pregledniku. Ako se pojavi stranica s porukom "It worked!", to znači da je vaš Django projekt uspješno postavljen i spreman za daljnji razvoj.

3.5. MVC (Model-View-Controller) arhitektura u Django

Jedan od ključnih koncepata na kojima se temelji Django je **MVC arhitektura** (Model-View-Controller). Ovo je obrazac koji se koristi za razdvajanje poslovne logike aplikacije od njenog prikaza i interakcije s korisnicima.

3.5.1. Model (M)

Model predstavlja sloj aplikacije koji je odgovoran za interakciju s bazom podataka. U Djangu, model je Python klasa koja mapira na bazu podataka i definira strukturu podataka s kojima aplikacija radi. Svaki model predstavlja tablicu u bazi podataka, dok svaki atribut unutar modela

predstavlja stupac te tablice.

```python
from django.db import models

class Autor(models.Model):
    ime = models.CharField(max_length=100)
    datum_rodenja = models.DateField()

    def __str__(self):
        return self.ime
```

Slika 6. Primjer modela (autor)

3.5.2. View (V)

View je odgovoran za obradu korisničkih zahtjeva i vraćanje odgovarajućeg odgovora, bilo da je to HTML stranica, JSON podatak ili nešto drugo. U Djangu, view je Python funkcija ili klasa koja definira logiku odgovora na određeni URL zahtjev.

```python
from django.http import HttpResponse

def pozdrav(request):
    return HttpResponse("Pozdrav svijete!")
```

Slika 7. Primjer viewa (autor)

3.5.3. Controller (C)

U Djangu, komponenta "Controller" zapravo je zastupljena u sustavu **URL ruta**. Ove rute povezuju korisničke zahtjeve s odgovarajućim view funkcijama. Svaki URL obrazac definiran je unutar datoteke urls.py. Primjer rute koja povezuje URL s view-om:

```python
from django.urls import path
from . import views

urlpatterns = [
    path('pozdrav/', views.pozdrav),
]
```

Slika 8. Primjer controllera (autor)

U ovom primjeru, svaki put kada korisnik pristupi URL-u http://localhost:8000/pozdrav/, Django poziva funkciju pozdrav i vraća odgovarajući odgovor.

3.6. Rad s bazama podataka u Django projektima

Jedna od ključnih značajki Django razvojnog okvira je integrirani sustav za rad s bazama podataka putem **ORM-a** (Object-Relational Mapping), koji omogućuje rad s bazama podataka na visokoj razini apstrakcije. Djangov ORM omogućuje programerima interakciju s bazom podataka putem Python objekata i klasa, eliminirajući potrebu za pisanjem SQL koda. Ovaj pristup pojednostavljuje upravljanje podacima, osobito u aplikacijama koje koriste relacijske baze podataka.

3.6.1. Migracije – upravljanje promjenama u bazi podataka

Jedna od najvažnijih značajki Django ORM-a je sustav za migracije. **Migracije** su skripte koje prate promjene u modelima i automatski prilagođavaju bazu podataka novim strukturama podataka bez potrebe za ručnim pisanjem SQL naredbi. Nakon što definiramo ili promijenimo model, mogu se generirati migracije s pomoću sljedeće naredbe:

```
python manage.py makemigrations
```

Slika 9. Naredba za generiranje migracije (autor)

A zatim primijeniti te migracije na bazu podataka:

```
python manage.py migrate
```

Slika 10. Naredba za primjenu migracija (autor)

Ovim postupkom, Django automatski kreira tablice u bazi podataka ili ih ažurira u skladu s promjenama u modelima.

3.6.2. Upiti nad bazom podataka

Djangov ORM omogućuje jednostavno izvršavanje upita nad bazom podataka. Na primjer, za dohvaćanje svih korisnika iz tablice Korisnik, možemo koristiti sljedeći upit:

```
korisnici = Korisnik.objects.all()
```

Slika 11. Upit nad bazom podataka (autor)

Za složenije upite, Django nudi različite metode poput filter(), exclude(), order_by() i mnoge druge, omogućujući fleksibilno i efikasno pretraživanje baze podataka. ORM također podržava povezane modele, omogućujući jednostavne upite nad relacijskim podacima.

3.7. Sigurnost i autentikacija korisnika

Sigurnost je ključna komponenta svake internet aplikacije, a Django pruža snažne mehanizme zaštite aplikacija. Jedna od najvažnijih značajki Djanga je ugrađeni sustav za **autentifikaciju i autorizaciju korisnika**, koji omogućuje jednostavno upravljanje korisničkim prijavama i dozvolama pristupa.

3.7.1. Ugrađeni sustav za autentifikaciju

Django dolazi s gotovim modulom za autentifikaciju korisnika, koji omogućuje brzo postavljanje prijave, registracije i autentifikacije. Model User, koji se automatski dodaje prilikom kreiranja projekta, omogućuje spremanje informacija o korisnicima kao što su korisničko ime, lozinka, e-mail, te status aktivacije i dozvole.

Autentifikacija se često provodi kroz sljedeće korake:

1. **Prijava korisnika** – Korisnici unose svoje podatke putem web forme, a Django ih provjerava u bazi podataka.
2. **Registracija novih korisnika** – Korisnici mogu kreirati nove račune putem formulara,

a njihove informacije automatski se spremaju u bazu.

3. **Upravljanje lozinkama** – Django omogućuje resetiranje lozinki i slanje e-mailova korisnicima sa sigurnosnim kodovima za potvrdu promjena.

Primjer prijave korisnika može izgledati ovako:

```
from django.contrib.auth import authenticate, login

def prijava(request):
    korisnicko_ime = request.POST['username']
    lozinka = request.POST['password']
    korisnik = authenticate(request, username=korisnicko_ime, password=lozinka)

    if korisnik is not None:
        login(request, korisnik)
        return HttpResponse("Prijava uspješna")
    else:
        return HttpResponse("Neispravni ↓ aci")
```

Slika 12. Primjer prijave korisnika (autor)

Django nudi i ugrađene funkcionalnosti za zaštitu lozinki korištenjem naprednih algoritama za hashiranje, čime se osigurava da lozinke nikada nisu pohranjene kao čisti tekst.

3.7.2. Zaštita od uobičajenih napada

Django automatski pruža zaštitu od mnogih uobičajenih sigurnosnih prijetnji, uključujući:

1. **CSRF (Cross-Site Request Forgery)** – Django ugrađuje CSRF token u svaku POST formu, osiguravajući da samo legitimni korisnici mogu slati zahtjeve.
2. **SQL injekcije** – Korištenjem ORM-a, Django automatski sanitizira ulaze i sprječava napade SQL injekcijama.
3. **XSS (Cross-Site Scripting)** – Django automatski obrađuje podatke koji dolaze iz korisničkih unosa prilikom renderiranja HTML-a, čime sprječava izvršenje zlonamjernog JavaScript koda.

Uz ove mjere, Django potiče upotrebu HTTPS protokola za šifriranje podataka koji se razmjenjuju između klijenata i servera, što dodatno osigurava aplikaciju.

3.8. Integracija s različitim servisima I API-ima

Moderne internet aplikacije često se oslanjaju na integraciju s različitim servisima i API-ima za proširenje funkcionalnosti. Django omogućuje jednostavnu integraciju s vanjskim servisima, bilo putem RESTful API-ja, SOAP servisa ili drugih metoda razmjene podataka.

Integracija s REST API-ima

REST (Representational State Transfer) API je najčešće korišten oblik API-ja koji omogućuje interakciju između različitih sustava putem HTTP protokola. Django ima ugrađene alate i biblioteke, poput **Django REST Frameworka** (DRF), koji olakšava kreiranje i integraciju s RESTful API-ima.

Django REST razvojno okruženje pruža fleksibilnu platformu za kreiranje API-ja u Djangu, omogućujući lako definiranje endpointa, serijalizaciju podataka i upravljanje autentifikacijom. Primjer jednostavnog API endpointa za dohvaćanje liste korisnika može izgledati ovako:

```
from rest_framework import viewsets
from .models import Korisnik
from .serializers import KorisnikSerializer

class KorisnikViewSet(viewsets.ModelViewSet):
    queryset = Korisnik.objects.all()
    serializer_class = KorisnikSerializer
```

Slika 13. Primjer API endpointa (autor)

Ovaj endpoint omogućuje dohvaćanje, dodavanje i ažuriranje korisnika putem API-ja, uz automatsku serijalizaciju podataka u JSON format.

3.8.1. Autentifikacija putem vanjskih servisa

Django se lako integrira s vanjskim servisima za autentifikaciju poput **Googlea**, **Facebooka**, i **GitHuba**. Korištenje ovih servisa omogućuje korisnicima prijavu putem svojih postojećih računa na ovim platformama. Biblioteke poput **django-allauth** pružaju alate za jednostavno postavljanje autentifikacije putem društvenih mreža ili drugih OAuth servisa.

Za integraciju s Google prijavom, na primjer, koriste se OAuth protokoli za sigurno preuzimanje korisničkih podataka i provjeru njihovog identiteta, bez potrebe za ručnim

upravljanjem korisničkim podacima.

3.8.2. Integracija s vanjskim servisima putem Webhookova

Webhookovi su još jedan način integracije s vanjskim sustavima. Django može primati obavijesti iz drugih sustava putem HTTP zahtjeva kada dođe do određenih događaja. Na primjer, može se integrirati Django aplikacija s platnim sustavom poput **Stripea** koji šalje obavijesti o uspješnim ili neuspješnim transakcijama. Webhook obrade se mogu postaviti unutar Django view-a, kao što je prikazano u primjeru:

```
from django.http import HttpResponse

def stripe_webhook(request):
    event = json.loads(request.body)

    if event['type'] == 'payment_intent.succeeded':
        # Obrada uspješne uplate
        return HttpResponse(status=200)

    return HttpResponse(status=400)
```

Slika 14. Primjer Webhook obrade (autor)

4. PRAKTIČNI RAD – TODO APLIKACIJA

Kroz prethodni dio rada obradio se teoretski dio upotrebe Django razvojnog okruženja za razvoj backend dijela internet aplikacije. Kroz praktični dio će se na primjeru jedne internet aplikacije prikazati kako uz pomoć Djanga dobiti funkcionalni backend dio aplikacije.

4.1. Todo aplikacija

Prije početka razvoja same aplikacije, potrebno je kreirati virtualno okruženje u kojem ćemo razvijati aplikaciju. Koristimo naredbeni redak i sljedeće naredbe:

```
C:\Users\Nemanja>py -m venv VirEnv

C:\Users\Nemanja>VirEnv\Scripts\activate.bat

(VirEnv) C:\Users\Nemanja>
```

Slika 15. Kreiranje virtualnog okruženja (autor)

Nakon kreiranja virtualnog okruženja potrebno ga je aktivirati kako bi se moglo u njemu raditi.Virtualno okruženje je aktivno kada se naziv okruženja pojavi u zagradama na početku naredbenog retka, u ovom slučaju „(VirEnv)“. Važno je za napomenuti kako je virtualno okruženje potrebno aktivirati svaki put kad se počinje raditi na projektu nakon zatvaranja samog okruženja.

Prije kreiranja samog projekta unutar virtualnog okruženja, potrebno je instalirati Django razvojno okruženje u virtualnom okruženju. Najjednostavnije je to uraditi s pomoću naredbenog retka u kojem se trenutno nalazimo, pozicioniramo se unutar virtualnog okruženja.

```
(VirEnv) C:\Users\Nemanja\VirEnv>py -m pip install Django
Collecting Django
  Using cached Django-5.1.1-py3-none-any.whl (8.2 MB)
Collecting asgiref<4,>=3.8.1
  Using cached asgiref-3.8.1-py3-none-any.whl (23 kB)
Collecting sqlparse>=0.3.1
  Using cached sqlparse-0.5.1-py3-none-any.whl (44 kB)
Collecting tzdata
  Using cached tzdata-2024.1-py2.py3-none-any.whl (345 kB)
Collecting typing-extensions>=4
  Using cached typing_extensions-4.12.2-py3-none-any.whl (37 kB)
Installing collected packages: typing-extensions, tzdata, sqlparse, asgiref, Django
Successfully installed Django-5.1.1 asgiref-3.8.1 sqlparse-0.5.1 typing-extensions-4.12.2 tzdata-2024.1
WARNING: You are using pip version 21.2.4; however, version 24.2 is available.
You should consider upgrading via the 'C:\Users\Nemanja\VirEnv\Scripts\python.exe -m pip install --upgrade pip' command.


(VirEnv) C:\Users\Nemanja\VirEnv>
```

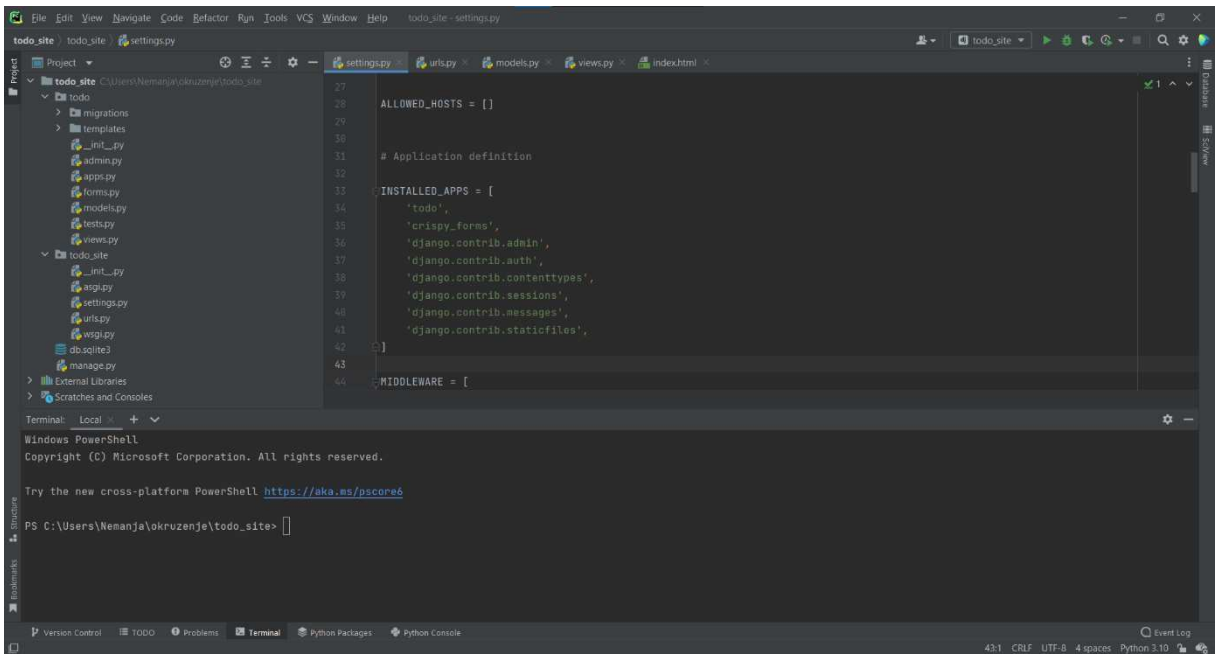
Slika 16. Instalacija Django frameworka (autor)

U aplikaciji koja će se raditi koristimo Django Crispy Forms. Paket koji olakšava upravljanje izgledom i stilizacijom formi u Django aplikacijama, omogućavajući jednostavnu integraciju sa CSS razvojnim okruženjima poput Bootstrapa. Pruža fleksibilnost u prilagođavanju i poboljšava korisničko iskustvo kroz čist i organiziran prikaz formi.

```
(VirEnv) C:\Users\Nemanja\VirEnv>py -m pip install django-crispy-forms
Collecting django-crispy-forms
  Using cached django_crispy_forms-2.3-py3-none-any.whl (31 kB)
Requirement already satisfied: django>=4.2 in c:\users\nemanja\virenv\lib\site-packages (from django-crispy-forms) (5.1.
1)
Requirement already satisfied: asgiref<4,>=3.8.1 in c:\users\nemanja\virenv\lib\site-packages (from django>=4.2->django-
crispy-forms) (3.8.1)
Requirement already satisfied: sqlparse>=0.3.1 in c:\users\nemanja\virenv\lib\site-packages (from django>=4.2->django-cr
ispy-forms) (0.5.1)
Requirement already satisfied: tzdata in c:\users\nemanja\virenv\lib\site-packages (from django>=4.2->django-crispy-form
s) (2024.1)
Requirement already satisfied: typing-extensions>=4 in c:\users\nemanja\virenv\lib\site-packages (from asgiref<4,>=3.8.1
->django>=4.2->django-crispy-forms) (4.12.2)
Installing collected packages: django-crispy-forms
Successfully installed django-crispy-forms-2.3
WARNING: You are using pip version 21.2.4; however, version 24.2 is available.
You should consider upgrading via the 'C:\Users\Nemanja\VirEnv\Scripts\python.exe -m pip install --upgrade pip' command.
```

Slika 17. Instalacija Django Crispy Forms (autor)

Nakon instalacije potrebnih paketa, može se kreirati projekt unutar virtualnog okruženja. Sam projekt će se zvati „todo_site“, dok će se aplikacija koju kreiramo unutar njega nazvati „todo“. Treba izbjegavati projekt i aplikaciju nazivati jednako, zbog lakšeg snalaženja u kasnijem radu na projektu.



Slika 18. PyCharm IDE (autor)

Potrebno je otići u todo_site\settings.py i u INSTALLED_APPS dodati todo aplikaciju i Django Crispy Forms.

```
27
28  ALLOWED_HOSTS = []
29
30
31  # Application definition
32
33  INSTALLED_APPS = [
34      'todo',
35      'crispy_forms',
36      'django.contrib.admin',
37      'django.contrib.auth',
38      'django.contrib.contenttypes',
39      'django.contrib.sessions',
40      'django.contrib.messages',
41      'django.contrib.staticfiles',
42  ]
43
44  MIDDLEWARE = [
```

Slika 19. Installed_Apps (autor)

Potrebno je kreirati modele, viewove i controllere za aplikaciju kako bi imali osnovne slojeve za rad aplikacije. Unutar ove aplikacije modeli će se smjestiti u todo\models.py.

```
from django.db import models
from django.utils import timezone
```

```
class Todo(models.Model):
    title = models.CharField(max_length=100)
    details = models.TextField()
    date = models.DateTimeField(default=timezone.now)

    def __str__(self):
        return self.title
```

Kod 5. Model (autor)

Viewove smo kreirali kao Python funkcije. Funkcija index() vraća HTML file, a funkcija remove obrađuje model Todo, tj briše ga.

```
from django.shortcuts import render, redirect
from django.contrib import messages
from .forms import TodoForm
from .models import Todo

def index(request):

    item_list = Todo.objects.order_by("-date")
    if request.method == "POST":
        form = TodoForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('todo')
    form = TodoForm()

    page = {
        "forms": form,
        "list": item_list,
        "title": "TODO LIST",
    }
    return render(request, 'todo/index.html', page)
```

```
# function to remove item
def remove(request, item_id):
    item = Todo.objects.get(id=item_id)
    item.delete()
    messages.info(request, "item removed !!!")
    return redirect('todo')
```

Kod 6. View (autor)

Controlleri u django aplikaciji su URL putevi.

```
from django.contrib import admin
from django.urls import path
from todo import views

urlpatterns = [

    path('', views.index, name="todo"),

    path('del/<str:item_id>', views.remove, name="del"),

    path('admin/', admin.site.urls),
]
```

Kod 7. URLs (autor)

Nakon završetka potrebnih modela, viewova i controllera potrebno je odraditi migracije kako bi se ažurirala baza podataka. Za migraciju koristimo Terminal unutar PyCharma i naredbe:

```
python manage.py makemigrations
python manage.py migrate
```

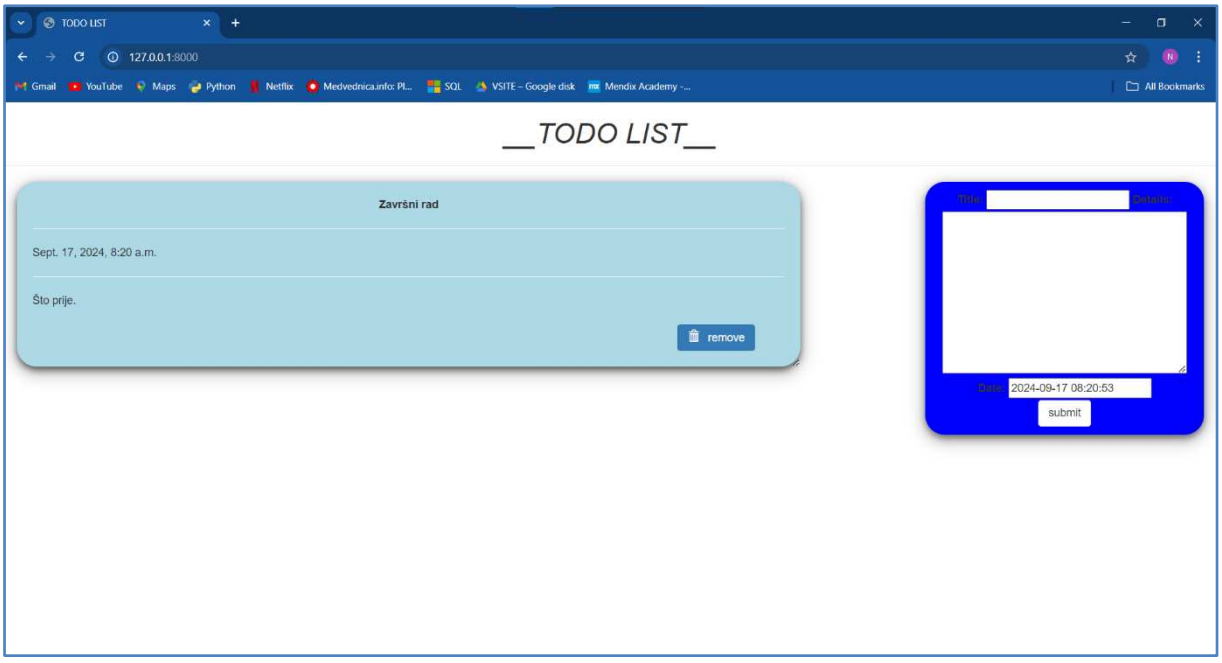
Slika 20. Naredbe za migracije(W3Schools, 2024.)

Nakon izvršenih migracija, može se pokrenuti server na lokalnom računalu, ali će se aplikacija pokrenuti u Internet pregledniku.

```
PS C:\Users\Nemanja\okruzenje\todo_site> py manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
September 17, 2024 - 10:19:36
Django version 4.2, using settings 'todo_site.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Slika 21. Pokretanje servera (autor)



Slika 22. Todo aplikacija (autor)

5. ZAKLJUČAK

Python i Django, kao jedan od najpopularnijih internet razvojnih alata, predstavljaju iznimno moćnu kombinaciju za izradu modernih, sigurnih i skalabilnih web aplikacija. Kroz dosadašnju analizu, prikazano je kako Python, sa svojom jednostavnošću, čitljivošću i bogatim ekosustavom biblioteka, pruža čvrstu osnovu za razvoj različitih vrsta softverskih rješenja, uključujući internet aplikacije, znanstvene projekte, automatizaciju i razvoj umjetne inteligencije.

Python je posebno istaknut po svojoj fleksibilnosti i dostupnosti ogromnog broja biblioteka, čime se omogućuje brzo rješavanje specifičnih problema bez potrebe za izradom složenih algoritama od nule. Njegova jednostavna sintaksa čini ga pristupačnim i novim programerima i stručnjacima, dok njegovu svestranost dokazuje široka primjena u različitim industrijama. Između ostalog, Pythonov napredni rad s funkcijama omogućava jednostavnu implementaciju funkcionalnosti kroz modularni pristup, čime se povećava održivost i skalabilnost aplikacija.

Django razvojni okvir, razvijen na Pythonu, dodatno proširuje mogućnosti za internet razvoj, nudeći moćne alate za brzi razvoj i sigurnost. Njegov pristup "baterije uključene" osigurava da programeri odmah imaju na raspolaganju ključne komponente potrebne za izradu funkcionalnih aplikacija, uključujući ORM za upravljanje bazama podataka, ugrađene sigurnosne mehanizme, autentifikaciju korisnika i sustav za upravljanje migracijama. Sve te značajke čine Django jednim od najpoželjnijih izbora za razvoj složenih internet aplikacija koje zahtijevaju robusnu backend logiku.

Unatoč svojim prednostima, Django nije bez nedostataka. Svojim opsežnim setom funkcionalnosti, može biti previše "težak" za manje projekte, gdje jednostavnija razvojna okruženja mogu biti efikasnija. Osim toga, ORM, iako vrlo koristan za jednostavne upite, može postati neefikasan u radu sa složenijim bazama podataka i zahtjevima koji uključuju napredne SQL upite. Unatoč tim ograničenjima, Django nudi jasnu prednost u razvoju aplikacija kojima je potrebna brzina i sigurnost, kao što su platforme za e-trgovinu, društvene mreže i upravljački sustavi sadržaja (CMS).

Sigurnost, kao jedan od ključnih izazova u modernom web razvoju, iznimno je dobro pokrivena unutar Django razvojnog okvira. Ugrađena zaštita od CSRF, XSS i SQL injekcija, kao i podrška za HTTPS, čine Django vrlo sigurnim okruženjem za izradu aplikacija. Autentifikacija

korisnika, bilo putem tradicionalnih metoda prijave ili putem integracije s vanjskim servisima poput Googlea ili Facebooka, pojednostavljuje upravljanje korisnicima i osigurava visok stupanj sigurnosti njihovih podataka.

Osim toga, Djangova podrška za rad s vanjskim servisima i API-ima putem Django REST Frameworka omogućava jednostavnu integraciju s vanjskim sustavima, čime se značajno proširuje funkcionalnost aplikacija. Webhookovi i integracija s platnim sustavima, kao što su Stripe i PayPal, omogućavaju Djangu da bude osnovna platforma za razvoj cjelovitih rješenja, od korisničkog sučelja do složenih backend operacija.

U konačnici, Python i Django zajedno nude pouzdano, skalabilno i sigurno rješenje za web razvoj. Dok Python pruža čvrstu osnovu u pogledu programiranja općenito, Django donosi specifične alate koji omogućuju jednostavno upravljanje bazama podataka, sigurnost, te jednostavnu integraciju s vanjskim servisima. Razumijevanje oba alata ključno je za svakog modernog programera koji želi razvijati složene web aplikacije na učinkovit i siguran način.

Uzevši u obzir sve prednosti i mane, Python i Django ostaju među najvažnijim alatima za internet razvoj. Njihova prilagodljivost, sigurnost i široka primjena u industriji čine ih idealnim izborom za projekte svih veličina, dok njihova zajednica i dokumentacija pružaju podršku i resurse potrebne za kontinuirani razvoj i napredak.

LITERATURA

1. Eric Matthes, (2016), Python Crash Course, *No Starch Pres*
2. William S. Vincent, (2024), Django for beginners, WelcomeToCode
3. Daniel Rubio,(2017.), Beginning Django, Apress Berkeley, CA
4. W3Schools, Django Tutorial, https://www.w3schools.com/django/index.php (pristupljeno 16.9.2024.)
5. GeeksforGeeks, Django Tutorial | Learn Django Framework, https://www.geeksforgeeks.org/django-tutorial/?ref=shm, (pristupljeno 16. 9. 2024.)

SAŽETAK

Ovaj rad prikazuje proces razvoja web aplikacije koristeći Python programski jezik i Django razvojno okruženje. Python, zbog svoje jednostavne sintakse i svestranosti, odabran je kao temeljni programski jezik, dok Django, kao jedan od najpopularnijih internet rauvojnih okruženja, pruža potrebne alate za brzo, sigurno i skalabilno razvijanje aplikacija. Kroz rad je detaljno objašnjen razvoj aplikacije, uključujući ključne korake kao što su kreiranje virtualnog okruženja, kreiranja projekta, definiranje modela podataka s pomoću Django ORM-a, migracije modela podataka u bazu podataka. Konačni rezultat je funkcionalna i sigurna internet aplikacija koja koristi moderne standarde u razvoju softvera, s naglaskom na skalabilnost, održivost i jednostavno upravljanje podacima.

Ključne riječi: Python programski jezik, Django, MVC arhitektura, web aplikacija, SQL lite

SUMMARY

This paper presents the process of developing a web application using the Python programming language and the Django framework. Python, due to its simple syntax and versatility, was chosen as the core programming language, while Django, as one of the most popular web frameworks, provides the necessary tools for fast, secure, and scalable application development. The paper explains in detail the development of the application, including key steps such as creating a virtual environment, creating the project, defining data models using Django ORM, and migrating data models to the database. The final result is a functional and secure web application that adheres to modern software development standards, with a focus on scalability, maintainability, and simple data management.

Keywords: Python programming language, Django, MVC architecture, web application, SQLite