

**SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN**

Nina Krsnik

**APLIKACIJA ZA UPRAVLJANJE
KORISNIČKIM RAČUNIMA**

PROJEKT

TEORIJA BAZA PODATAKA

Varaždin, 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Nina Krsnik

Matični broj: 0016155048

Studij: Baze podataka i baze znanja

APLIKACIJA ZA UPRAVLJANJE KORISNIČKIM RAČUNIMA

PROJEKT

Mentor:

Prof. dr. sc. Markus Schatten

Varaždin, Siječanj 2025.

Nina Krsnik

Izjava o izvornosti

Izjavljujem da je ovaj projekt izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autorica potvrdila prihvatanjem odredbi u sustavu FOI Radovi

Sažetak

Seminarski rad bavi se razvojem aplikacije za upravljanje korisničkim računima koja uključuje različite uloge korisnika, prava pristupa te pohranu meta podataka. Aplikacija koristi Flask framework za razvoj web sučelja i PostgreSQL za pohranu podataka. Korisnicima su dodijeljene različite uloge (admin, super admin, moderator, novosti) s odgovarajućim pravima pristupa, dok meta podaci o korisnicima omogućuju bolju kontrolu i sigurnost sustava. Rad također prikazuje tehničku implementaciju funkcionalnosti kao što su prijava korisnika, dodavanje i brisanje postova te upravljanje meta podacima. Kroz aplikaciju je osigurana efikasnost u upravljanju korisničkim podacima i zaštiti informacija.

Ključne riječi: upravljanje korisničkim računima, uloge korisnika, prava pristupa, meta podaci, Flask, PostgreSQL, sigurnost podataka, web aplikacija, administracija korisnika

Sadržaj

1. Uvod	1
2. Teorijski uvod	2
2.1. Poopćene baze podataka	2
2.2. Objektno-relacijske baze podataka	2
2.3. PostgreSQL	3
2.4. Flask	3
2.5. JSON	3
2.6. HTML	3
3. Model baze podataka	5
4. Implementacija	6
4.1. Kreiranje tablica u PostgreSQL	6
4.2. Implementacija aplikacije Python i Flask app.py	9
4.3. Opis HTML i CSS stranica	11
5. Primjeri korištenja	14
6. Zaključak	20
7. Literatura	21

1. Uvod

Upravljanje korisničkim računima danas je postala ključna komponenta svakog informacijskog sustava. Svaka aplikacija koja obrađuje osjetljive podatke, poput osobnih informacija, mora imati efikasan način upravljanja korisnicima, njihovim pravima pristupa i pohranom meta podataka. U ovom seminarskom radu istražit će se razvoj aplikacije za upravljanje korisničkim računima koja omogućuje različite uloge korisnika i različite razine pristupa podacima. Aplikacija je dizajnirana za sigurno upravljanje korisničkim podacima i omogućava administrativne funkcionalnosti kao što su dodavanje, brisanje korisnika i upravljanje njihovim pravima pristupa.

Sustav koristi mikro web framework za Python Flask te za upravljanje bazama podataka PostgreSQL sustav. Aplikacija omogućuje pristup različitim razinama prava korisnicima, uključujući administratore, super administratore, moderatore i novosti, čime se osigurava da podaci budu zaštićeni i da se pristup njima može pratiti i kontrolirati.

Pored osnovnih funkcionalnosti, aplikacija također omogućuje pohranu i prikaz meta podataka korisnika. Kroz ovaj rad, bit će prikazano kako je aplikacija razvijena, koji su tehnički izazovi prepoznati tijekom izrade, kao i prednosti korištenja takvog sustava za upravljanje korisnicima i njihovim podacima.

Motivacija za odabir ove teme je želja za razumijevanjem i razvojem naprednih tehnika za upravljanje korisničkim računima s naglaskom na sigurnost i pristup njihovim pravima. Na taj način omogućuje se daljnje usavršavanje u području web sigurnosti i aplikacijskog razvoja. Također, izradom ovog projekta produbljuje se znanje u radu s tehnologijama Flask i PostgreSQL kako bi se stvorile web stranice, baze podataka i sigurnosne mjere koje će biti međusobno povezane.

2. Teorijski uvod

Baze podataka predstavljaju ključni element u mnogim aplikacijama jer one omogućuju pohranu, pretragu i manipulaciju podacima i to na način koji je veoma učinkovit i siguran pogotovo kada se radi o velikoj količini podataka. Različiti pristupi bazama podataka razvijeni su tijekom vremena kako bi zadovoljili specifične potrebe aplikacija i njihovih korisnika. U ovom teoretskom uvodu razmotrit će se osnovni pristupi bazama podataka koji su relevantni za implementaciju aplikacije za upravljanje korisničkim računima, a to su poopćene i objektno-relacijske baze podataka. [1]

2.1. Poopćene baze podataka

Poopćene baze podataka omogućuju rad s različitim vrstama podataka unutar istog sustava, kombinirajući karakteristike relacijskih, objektnih i drugih pristupa. One omogućuju veću fleksibilnost u radu s podacima jer mogu pohraniti podatke različitih tipova i struktura. Ovaj pristup može biti koristan u aplikacijama koje zahtijevaju pohranu podataka u različitim formatima, kao što su tekstualni podaci, slike, video ili čak objektno-orijentirani podaci. Njihova prednost je velika fleksibilnost u radu s različitim vrstama podataka. Međutim, zbog toga javlja se povećana složenost implementacije te se kod njih javlja potreba za većim resursima za održavanje i optimizaciju sustava. [3] U ovom projektu, pristup poopćenim bazama podataka može se primijeniti u kontekstu fleksibilnosti u načinu pohrane i pristupa podacima. Na primjer, tablica meta podaci može sadržavati podatke o korisnicima koji nisu striktno povezani s drugim tablicama, omogućujući proširivost i fleksibilnost podataka.

2.2. Objektno-relacijske baze podataka

Objektno-relacijske baze podataka kombiniraju snagu relacijskih baza podataka s prednostima objektno-orijentiranog pristupa. Ovaj pristup omogućuje pohranu složenih objekata u bazama podataka, dok istovremeno zadržava relacijsku strukturu za jednostavnije upite i administraciju podataka. Objektno-relacijske baze podataka omogućuju pohranu podataka koji sadrže referencijalne veze, nasljeđivanje i složene podatkovne tipove, što je korisno za aplikacije koje rade s objektima i njihovim atributima. [2] [3] Za aplikaciju za upravljanje korisničkim računima, ovaj pristup može omogućiti pohranu korisničkih podataka u obliku objekata koji se mogu manipulirati na objektno-orijentiran način, dok istovremeno zadržavaju strukturalnu prednost relacijskih baza podataka. Njihova prednost je ta što kombiniraju dobre strane od oba pristupa, dakle uzimaju prednosti objektnog i relacijskog pristupa. Time se omogućuje složeno modeliranje podataka. S druge strane, javlja se veća složenost u implementaciji zbog toga može biti sporiji u odnosu na čiste relacijske baze podataka. U ovom projektu, PostgreSQL kao objektno-relacijski sustav omogućava nam da pohranimo podatke u relacijskim tablicama (kao što su korisnik i post) i istovremeno iskoristimo napredne značajke kao što su korisnički definirani tipovi podataka. Na primjer, podaci poput meta podaci mogu koristiti proširene tipove

podataka koji nisu ograničeni na osnovne SQL tipove, već omogućuju složenije strukture.

2.3. PostgreSQL

PostgreSQL je sustav za upravljanje odnosnim bazama podataka koji koristi SQL (Structured Query Language) za pohranu i manipulaciju podacima. To je objektno-relacijski sustav, što znači da podržava i tradicionalne relacijske modele podataka (tablice, redove, stupce) i dodatne objektne značajke poput korisnički definiranih tipova podataka, funkcija i proširenja. Podaci su organizirani u tablicama koje su povezane putem veza što je relacijski model. Zatim omogućava izvođenje kompleksnih upita i optimizaciju uz pomoć indeksa. [5] // U ovom projektu, PostgreSQL je korišten za pohranu korisničkih podataka, meta podataka, postova u relacijskom formatu, gdje se podaci pohranjuju u tablicama s definiranim vezama između njih (npr. tablica korisnik povezana je s tablicom post).

2.4. Flask

Flask je mikro web framework za Python koji omogućava brzo razvijanje web aplikacija. Flask pruža samo osnovne komponente za razvoj aplikacija, kao što su rute, upravljanje sesijama, obrada obrazaca i prikazivanje HTML predložaka, ali ostavlja korisnicima slobodu da koriste druge alate i biblioteke prema potrebi. Lagan je i fleksibilan framework, koji omogućava brzi razvoj aplikacija. Moguće je razvijati aplikacije koje komuniciraju s podacima putem HTTP-a. Omogućava brzi razvoj s minimalnim postavkama. U ovom projektu, Flask je korišten za razvoj aplikacije koja omogućava interakciju s korisnicima putem web stranice. Aplikacija koristi rute koje omogućuju pregled postova, dodavanje korisnika i prikaz meta podataka. [4]

2.5. JSON

JSON je format za razmjenu podataka koji je lagan, čitljiv za ljude i jednostavan za parsiranje i generiranje u većini programskih jezika. Često se koristi za prenošenje podataka između klijenta i servera u web aplikacijama. JSON je jednostavan za čitanje i prenošenje podataka. Može se lako parsirati u gotovo svim programskim jezicima, uključujući Python i JavaScript. U ovom projektu JSON se koristi za razmjenu podataka između klijenta i servera u pogledu meta podataka. [7]

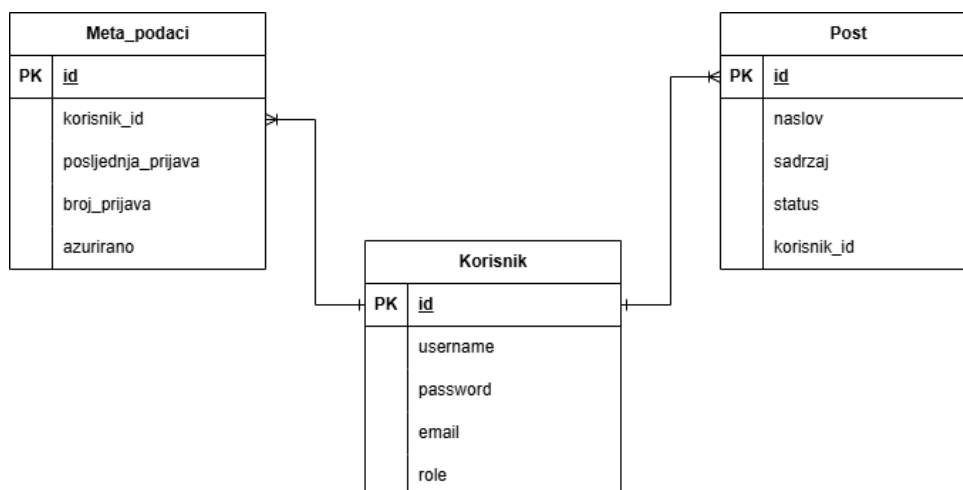
2.6. HTML

HTML je standardni jezik za označavanje i strukturiranje sadržaja na webu. HTML dokumenti sadrže oznake koje definiraju elemente kao što su tekst, slike, veze, forme i drugi sadržaji koji čine web stranicu. HTML je osnovni jezik za izradu web stranica. HTML se često koristi u kombinaciji s CSS-om i JavaScript-om za razvoj interaktivnih web aplikacija. U ovom

projektu, HTML je korišten za prikazivanje sadržaja na web stranici, uključujući listu postova, dodavanja uloga korisnicima, logiranja, prikaza podataka.

3. Model baze podataka

Za projekt su u PostgreSQL bazi podataka kreirane sljedeće tablice. Tablica Korisnik sadrži informacije o korisnicima sustava kao što su id, username, password, email, role. Veze koje su kod nje su: 1:N s tablicom Meta podaci. Jedan korisnik može imati više meta podataka. 1:N s tablicom Post. Jedan korisnik može postaviti više postova. Tablica Meta podaci sadrži dodatne informacije o korisnicima, poput podataka o njihovim prijavama, broju prijava i posljednjih akcija. Povezana je s tablicom korisnik N:1, odnosno više meta podataka može pripadati jednom korisniku. Tablica Post pohranjuje informacije o postovima koje korisnici stvaraju tako da sadrži atribut id, naslov, sadržaj, status koji jedino može stvarati superadmin i korisnik id koji povezuje post s korisnikom koji je kreirao post. Veze ovdje su N:1 s tablicom Korisnik, odnosno više postova može biti povezano s jednim korisnikom. 1:N s tablicom Komentar, jedan post može imati više komentara.



Slika 1: ERA dijagram (draw.io, vlastiti rad)

4. Implementacija

Za implementaciju baze podataka i odgovarajuće aplikacije, važno je uspostaviti osnovne dijelove sustava: od same baze podataka (kreiranje tablica i veza među njima) do aplikacije koja koristi te podatke.

4.1. Kreiranje tablica u PostgreSQL

```
CREATE FUNCTION public.update_azurirano_column() RETURNS trigger
LANGUAGE plpgsql
AS $$
BEGIN
    NEW.azurirano = CURRENT_TIMESTAMP;
    RETURN NEW;
END;
$$;

ALTER FUNCTION public.update_azurirano_column() OWNER TO nina;

SET default_tablespace = '';

SET default_table_access_method = heap;

--
-- Name: korisnici; Type: TABLE; Schema: public; Owner: nina
--

CREATE TABLE public.korisnici (
    id integer NOT NULL,
    username character varying(50) NOT NULL,
    password character varying(255) NOT NULL,
    email character varying(100) NOT NULL,
    role character varying(50) NOT NULL,
    inherits integer,
    CONSTRAINT korisnici_role_check CHECK (((role)::text = ANY ((ARRAY['admin'::character varying, 'super_admin'::character varying, 'novosti'::character varying, 'moderator'::character varying])::text[])))
);
```

Slika 2: Funkcija i tablica korisnici (snimka zaslona, vlastiti rad)

```

CREATE TABLE public.moderatori (
    can_manage_comments boolean DEFAULT true
)
INHERITS (public.korisnici);

ALTER TABLE public.moderatori OWNER TO nina;

--
-- Name: novosti; Type: TABLE; Schema: public; Owner: nina
--

CREATE TABLE public.novosti (
    can_publish_articles boolean DEFAULT true
)
INHERITS (public.korisnici);

ALTER TABLE public.novosti OWNER TO nina;

--
-- Name: postovi; Type: TABLE; Schema: public; Owner: nina
--

CREATE TABLE public.postovi (
    id integer NOT NULL,
    naslov character varying(255) NOT NULL,
    sadrzaj text NOT NULL,
    autor_id integer,
    datum_kreiranja timestamp without time zone DEFAULT CURRENT_TIMESTAMP
);

```

Slika 3: Inherits i tablica postovi(snimka zaslona, vlastiti rad)

```

CREATE FUNCTION public.update_azurirano_column() RETURNS trigger
    LANGUAGE plpgsql
    AS $$
BEGIN
    NEW.azurirano = CURRENT_TIMESTAMP;
    RETURN NEW;
END;
$$;

ALTER FUNCTION public.update_azurirano_column() OWNER TO nina;

SET default_tablespace = '';

SET default_table_access_method = heap;

--
-- Name: korisnici; Type: TABLE; Schema: public; Owner: nina
--

CREATE TABLE public.korisnici (
    id integer NOT NULL,
    username character varying(50) NOT NULL,
    password character varying(255) NOT NULL,
    email character varying(100) NOT NULL,
    role character varying(50) NOT NULL,
    inherits integer,
    CONSTRAINT korisnici_role_check CHECK (((role)::text = ANY ((ARRAY['admin'::character varying, 'super_admin'::character varying, 'novosti'::character varying, 'moderator'::character varying])::text[])))
);

```

Slika 4: Funkcija i korisnici (snimka zaslona, vlastiti rad)

```
CREATE TABLE public.admini (  
    can_manage_users boolean DEFAULT true  
)  
INHERITS (public.korisnici);  
  
ALTER TABLE public.admini OWNER TO nina;
```

Slika 5: Inherits (snimka zaslona, vlastiti rad)

```
CREATE TABLE public.postovi_status (  
    status character varying(50) DEFAULT 'draft'::character varying NOT NULL  
)  
INHERITS (public.postovi);  
  
ALTER TABLE public.postovi_status OWNER TO nina;  
  
--  
-- Name: super_admins; Type: TABLE; Schema: public; Owner: nina  
--  
CREATE TABLE public.super_admins (  
    admin_privileges boolean DEFAULT true  
)  
INHERITS (public.korisnici);
```

Slika 6: Inherits (snimka zaslona, vlastiti rad)

```

CREATE TABLE public.meta_podaci (
    id integer NOT NULL,
    korisnik_id integer NOT NULL,
    atribut character varying(255) NOT NULL,
    vrijednost text,
    kreirano timestamp without time zone DEFAULT CURRENT_TIMESTAMP,
    azurirano timestamp without time zone DEFAULT CURRENT_TIMESTAMP
);

ALTER TABLE public.meta_podaci OWNER TO nina;

--
-- Name: meta_podaci_id_seq; Type: SEQUENCE; Schema: public; Owner: nina
--

CREATE SEQUENCE public.meta_podaci_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE

```

Slika 7: Meta podaci (snimka zaslona, vlastiti rad)

```

-- Umetanje korisnika
INSERT INTO korisnik (username, password, email, role) VALUES
('admin', 'admin_pass', 'admin@example.com', 'admin'),
('user1', 'password1', 'user1@example.com', 'korisnik'),
('user2', 'password2', 'user2@example.com', 'moderator');

-- Umetanje meta podataka
INSERT INTO meta_podaci (korisnik_id, posljednja_prijava, broj_prijava, azurirano) VALUES
(1, '2025-01-16 10:00:00', 5, '2025-01-17 10:30:00'),
(2, '2025-01-16 12:00:00', 2, '2025-01-17 12:00:00');

-- Umetanje postova
INSERT INTO post (naslov, sadržaj, status, korisnik_id) VALUES
('Post 1', 'Ovo je prvi post', 'published', 1),
('Post 2', 'Ovo je drugi post', 'draft', 2);

-- Umetanje komentara
INSERT INTO komentar (sadržaj, korisnik_id, post_id, datum_kreiranja) VALUES
('Ovo je komentar na prvi post', 2, 1, '2025-01-17 13:00:00'),
('Ovo je komentar na drugi post', 3, 2, '2025-01-17 14:00:00');

```

Slika 8: Početno ručno uneseni podaci u tablice (snimka zaslona, vlastiti rad)

4.2. Implementacija aplikacije Python i Flask app.py

Flask je mikro web framework za Python koji omogućuje razvoj web aplikacija. U kontekstu ovog projekta, Flask se koristi za kreiranje web aplikacije koja omogućava interakciju s korisnicima putem internetskog preglednika. Flask obuhvaća osnovne komponente potrebne za razvoj web aplikacija, kao što su definiranje ruta, obrada HTTP zahtjeva i prikazivanje HTML stranica. [4]

U aplikaciji su definirane rute koje omogućuju korisnicima interakciju s aplikacijom putem URL-a. Svaka ruta predstavlja funkciju koja se izvršava kada korisnik posjeti određeni

URL putem HTTP zahtjeva (najčešće GET ili POST). Na primjer, ruta koja odgovara na zahtjev za prikazivanje svih postova u aplikaciji omogućuje korisniku da pregleda popis postova pohranjenih u bazi podataka.

Flask omogućuje povezivanje s bazom podataka putem SQLAlchemy, što je biblioteka koja olakšava rad s bazama podataka. Podaci o korisnicima, postovima i meta podacima pohranjuju se u relacijsku bazu podataka PostgreSQL. Flask koristi SQLAlchemy za izvođenje SQL upita koji dohvaćaju, dodaju ili ažuriraju podatke u bazi. Na primjer, kada korisnik doda novi post, Flask će obraditi zahtjev i ažurirati bazu podataka s novim podacima. Flask se koristi za implementaciju aplikacije koja upravlja korisničkim računima, meta podacima i pristupnim pravima. Aplikacija je dizajnirana tako da korisnicima omogućuje interakciju s bazom podataka putem internetskog sučelja. Flask pruža osnovne funkcionalnosti za upravljanje zahtjevima, povezivanje s bazom podataka, obrada HTML predložaka i rad s JSON podacima. [4]

Flask aplikacija u ovom projektu se sastoji od nekoliko ključnih komponenata:

- **Rute:** Definiraju način na koji aplikacija reagira na korisničke zahtjeve putem URL-ova.
- **Kontekst aplikacije:** Flask omogućuje dijeljenje podataka i konfiguracija unutar aplikacije.
- **Template engine (Jinja2):** Omogućuje dinamičko generiranje HTML stranica na temelju podataka iz aplikacije.
- **SQLAlchemy:** Biblioteka koja omogućuje povezivanje s relacijskom bazom podataka (u ovom slučaju PostgreSQL) i obavljanje SQL operacija.
- **JSON i HTML:** Format za prikazivanje podataka korisnicima putem internetskog sučelja.

Jedna od osnovnih funkcionalnosti Flask aplikacije je definiranje ruta koje omogućuju aplikaciji da reagira na različite vrste HTTP zahtjeva. Svaka ruta odgovara određenoj funkciji koja obrađuje zahtjev. Na primjer, kada korisnik posjeti početnu stranicu aplikacije, Flask odgovara na zahtjev za tu stranicu slanjem HTML predloška koji sadrži osnovne informacije o aplikaciji.

- **GET zahtjevi:** Ovi zahtjevi omogućuju korisnicima da pristupe određenim stranicama aplikacije. Na primjer, korisnik može posjetiti stranicu za prijavu, pregled postova ili upravljanje svojim računom.
- **POST zahtjevi:** Ovi zahtjevi se koriste za slanje podataka na server. Na primjer, kada korisnik šalje obrazac za prijavu podaci se šalju na server putem POST zahtjeva. Flask zatim procesira podatke, pohranjuje ih u bazu podataka i vraća korisniku odgovor, poput uspješne prijave ili pogreške u unosu.

Flask omogućuje definiranje ruta koje odgovaraju na različite vrste zahtjeva. Na primjer, rute za prijavu i registraciju obrađuju korisničke ulaze, dok rute za administrativne funkcije omogućuju administraciju korisničkih računa i meta podataka.

Flask koristi SQLAlchemy kao ORM (Object-Relational Mapping) biblioteku koja omogućuje povezivanje s relacijskom bazom podataka i obavljanje SQL operacija u Pythonu. U ovom projektu, baza podataka koristi PostgreSQL, a SQLAlchemy omogućuje aplikaciji da pohranjuje i dohvaća podatke u formi objekata.

U aplikaciji je definirana struktura modela koja odgovara različitim entitetima u bazi podataka, poput korisničkih računa, postova i meta podataka. Na primjer, model korisnika može sadržavati atribut kao što su ime, email, lozinka i uloga. SQLAlchemy omogućuje obavljanje operacija poput:

- **Kreiranje novih korisnika** kada se korisnik prijavi ili registrira.
- **Dohvaćanje podataka** za prikazivanje informacija o korisnicima ili postovima.
- **Ažuriranje podataka** kada korisnik mijenja svoje postavke ili dodaje novi post.
- **Brisanje podataka** ako korisnik odluči obrisati svoj račun ili post.

Za korisnike s posebnim ulogama, što je u ovom projektu super admin, Flask omogućuje pristup dodatnim informacijama o korisnicima i njihovoj aktivnosti u aplikaciji. Super administratori mogu pristupiti detaljnim meta podacima, poput broja postova i aktivnosti drugih korisnika.

- **Meta podaci o korisnicima:** Aplikacija može prikazati informacije o korisničkim računima, uključujući datum registracije, broj postova, broj komentara i druge aktivnosti.
- **Upravljanje korisnicima:** Administratori mogu dodavati, uređivati ili brisati korisničke račune, dok moderatori mogu odobravati ili uklanjati postove. Super administratori i kod kreiranja postova imaju dodatnu mogućnost odabira vrste posta.

Osim HTML-a, aplikacija koristi JSON (JavaScript Object Notation) za razmjenu podataka između klijenta i servera. Na primjer, kada korisnik šalje zahtjev za ažuriranje svog profila, podaci se mogu poslati u JSON formatu. Flask zatim obrađuje te podatke, ažurira bazu podataka i vraća odgovor u JSON formatu, što omogućuje jednostavno upravljanje podacima na strani klijenta. [7]

JSON se također koristi za komunikaciju s drugim servisima ili za slanje podataka u API pozivima, čime se omogućuje fleksibilna razmjena podataka u aplikaciji. [7]

4.3. Opis HTML i CSS stranica

HTML i CSS stranice zajedno omogućuju korisnicima jednostavan i intuitivan način interakcije s aplikacijom. HTML pruža strukturu i organizaciju podataka, dok CSS osigurava da stranice budu privlačne i jednostavne za korištenje. [6] Stranice su dizajnirane tako da omoguće lako kretanje kroz aplikaciju, omogućujući korisnicima da brzo pristupe funkcijama poput prijave, registracije, pregleda korisničkih podataka i postavki. [5]

- **Navigacija i izbornici:** Svaka stranica aplikacije sadrži navigacijski izbornik (`<nav>`) koja prikazuje mogućnosti pristupa koje svaki korisnik ima. Kao što su kreiranje postova, meta podaci ili odjava.
- **Forme za unos podataka:** HTML forme (`<form>`) omogućuju korisnicima unos podataka, kao što su prijava u sustav, dodavanje novih korisnika, dodavanje posta. Svaka forma sadrži odgovarajuće inpute (`<input>`) za unos informacija poput korisničkog imena, e-mail adrese, lozinke i drugih podataka. Forme su dizajnirane tako da omogućuju jednostavno unos podataka i validaciju prije nego što korisnik pošalje informacije.
- **Tablice za organizaciju podataka:** Za prikazivanje podataka u strukturiranom formatu koriste se HTML tablice (`<table>`). Tablice omogućuju korisnicima pregled svih relevantnih informacija u aplikaciji, kao što su popisi korisnika, njihove uloge, statusi i drugi podaci. Korištenje `<tr>`, `<td>` i `<th>` omogućava razvrstavanje podataka u redove i stupce, čineći informacije lako dostupnima i preglednima.
- **Struktura stranice:** Korištenjem osnovnih HTML elemenata kao što su `<header>`, `<footer>`, `<section>` i `<article>`, stranice su organizirane na način koji omogućuje jasno razdvajanje različitih dijelova stranice, kao što su zaglavlja, sadržaj i navigacija.

CSS je korišten za stiliziranje HTML elemenata i poboljšanje vizualnog identiteta aplikacije. Cilj CSS-a je učiniti stranice estetski privlačnima i funkcionalnima, uz poštivanje principa responzivnosti kako bi stranice bile optimizirane za različite uređaje. Korišteni su različiti CSS pristupi i tehnike za postizanje željenih rezultata [6]:

- **Responzivni dizajn:** Korištenjem `@media` upita, CSS omogućuje responzivni dizajn, čime stranice postaju prilagodljive različitim veličinama ekrana i uređajima, kao što su mobilni telefoni, tableti i desktop računala. Ovaj pristup osigurava da se sadržaj stranice prilagođava ovisno o veličini zaslona, čime se optimizira korisničko iskustvo.
- **Tipografija:** CSS je korišten za postavljanje fontova, veličina fontova, razmaka između slova, linija i odlomaka, čineći tekst lako čitljivim i vizualno privlačnim. Također su korištene boje i kontrasti kako bi se istaknuli važni dijelovi stranice, kao što su naslovi, podnaslovi i interaktivni elementi poput poveznica i gumba.
- **Boje i pozadine:** U projektu su korištene palete boja koje su usklađene s vizualnim identitetom aplikacije. Boje pozadine, teksta, gumbova i drugih elemenata postavljene su kako bi osigurale dobar kontrast i učinile sadržaj lako dostupnim. Pozadine mogu biti jednoboje ili koristiti slike, ovisno o dizajnu stranice.
- **Raspored elemenata:** CSS je korišten za postavljanje elemenata na stranici tako da se stranice pravilno prikazuju na svim uređajima. Korištenjem fleksibilnog rasporeda (Flexbox) i mrežnog rasporeda (CSS Grid), stranice omogućuju dinamičko

pozicioniranje elemenata u odnosu na veličinu prozora preglednika. Ovo uključuje raspored navigacijskih traka, tablica, formi i ostalih sadržajnih blokova.

- **Interaktivni efekti:** CSS efekti poput prijelaza i animacija dodaju dinamičnost stranici. Na primjer, gumbi mogu imati animacije prilikom hover stanja, a prijelazi mogu biti korišteni za glatke promjene boja ili pozicija. Ovi efekti poboljšavaju korisničko iskustvo i čine aplikaciju vizualno atraktivnijom.[6]

5. Primjeri korištenja

Prijava

Molimo unesite svoje podatke za prijavu kako biste pristupili sustavu.

User: superadmin

Lozinka: ----

Neppravno korisničko ime ili lozinka

Prijava se

Slika 9: Login(snimka zaslona, vlastiti rad)

Kada korisnik otvori web stranicu prvo će se pojaviti Login koji će od korisnika tražiti da unese user odnosno korisničko ime koje je prethodno određeno i može biti superadmin, admin, moderator ili novost. Također, od korisnika će se tražiti unos password odnosno šifre. Ukoliko se unesu krivi podaci pojaviti će se poruka da su uneseni pogrešni podaci.

Dobrodošli, novost!

Odjava Dodaj post

Postovi

Naslov	Sadržaj
pozdrav	bok
kako si	dobro
proba	ovo je proba
proba 2	test
1	11
test1	test1
1	1

Korisnici

ID	Korisničko ime	Uloga	Inherits
----	----------------	-------	----------

Slika 10: Prikaz stranice za user novost(snimka zaslona, vlastiti rad)

Ukoliko je korisnik unio podatke za usera novost tada će se prikazati web stranica prikazana na slici. Dakle, novost jedino može čitati trenutne postove. U navigacijskoj traci u desnom dijelu vidimo opciju Odjave. Možemo vidjeti dolje da je ispisana prazna tablica za prikaz korisnika. To znači da user novost nema ovlasti pregledavati ili mijenjati podatke u tablici korisnici.

Dobrodošli, moderator!

Odjava Dodaj post

Postovi

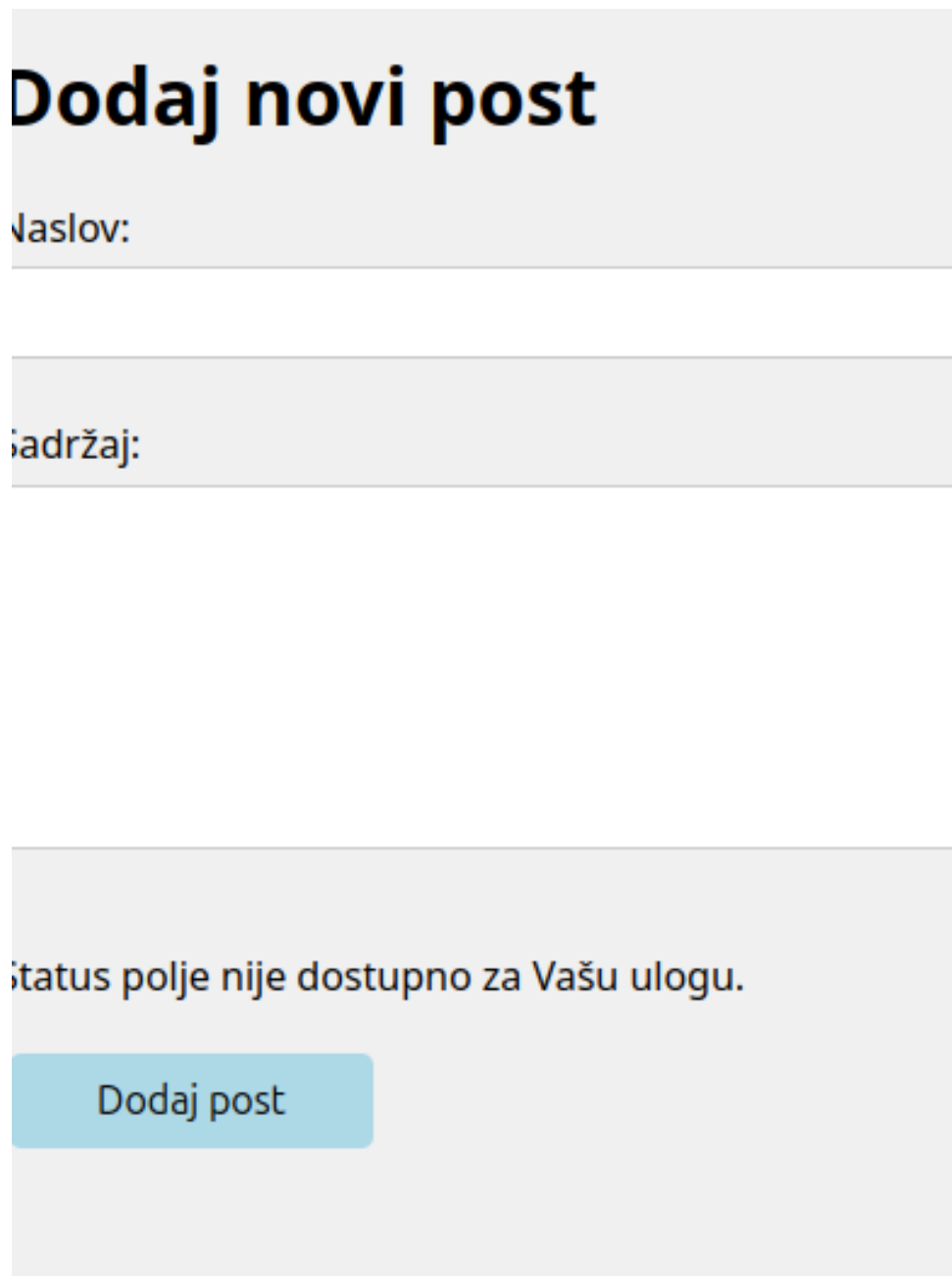
Naslov	Sadržaj
pozdrav	bok
kako si	dobro
proba	ovo je proba
proba 2	test
1	11
test1	test1
1	1

Korisnici

ID	Korisničko ime	Uloga	Inherits
----	----------------	-------	----------

Slika 11: Prikaz stranice za moderator(snimka zaslona, vlastiti rad)

Za moderatora prikazuje se web stranica u kojoj se može vidjeti da on može čitati postove, ali isto tako u navigacijskoj traci možemo vidjeti opciju Dodaj post koja označava da ovaj user može kreirati nove postove. Moderator kao i novost nema mogućnost pregleda tablice korisnici. Ukoliko odaberemo opciju Dodaj post otvorit će se stranica koja je vidljiva na sljedećoj slici.



Dodaj novi post

Naslov:

Sadržaj:

Status polje nije dostupno za Vašu ulogu.

Dodaj post

Slika 12: Dodaj običan post(snimka zaslona, vlastiti rad)

Ovdje možemo vidjeti da korisnik može unijeti naslov posta i sadržaj te da opcija status polja nije dostupna za ovog korisnika tako da kada korisnik unese naslov i sadržaj može pritisnuti gumb Dodaj post te će se post objaviti na početnoj stranici gdje su izlistani svi postovi.

Dobrodošli, admin!

Odjava

Dodaj post

Postovi

Naslov	Sadržaj	Brisanje
pozdrav	bok	Obrisi
kako si	dobro	Obrisi
proba	ovo je proba	Obrisi
proba 2	test	Obrisi
1	11	Obrisi
test1	test1	Obrisi
1	1	Obrisi

Korisnici

ID	Korisničko ime	Uloga	Inherits
----	----------------	-------	----------

Slika 13: Prikaz stranice za admin(snimka zaslona, vlastiti rad)

Ukoliko se prijavimo s admin korisničkim podacima pojavit će nam se ova stranica. Ona nam prikazuje da admin ima ovlasti pregleda postova, ali isto tako on ima i mogućnost obrisati ih. On isto ima opciju dodavanja običnog posta.

Dobrodošli, superadmin!			
		Odjava	Dodaj korisnika Meta podaci Dodaj post
Postovi			
Naslov	Sadržaj	Brisanje	Status
pozdrav	bok	Obrisi	
kako si	dobro	Obrisi	
proba	ovo je proba	Obrisi	
proba 2	test	Obrisi	
1	11	Obrisi	
test1	test1	Obrisi	draft
1	1	Obrisi	archived
Korisnici			
ID	Korisničko ime	Uloga	Inherits

Slika 14: Prikaz stranice za super admina(snimka zaslona, vlastiti rad)

Korisnici				
ID	Korisničko ime	Uloga	Inherits	
3	novost	novosti	1	Obrisi
4	moderator	moderator	1	Obrisi
1	admin	admin	2	Obrisi
5	clan	novosti	2	Obrisi
2	superadmin	super_admin	None	Obrisi

Slika 15: Prikaz stranice za super admina(snimka zaslona, vlastiti rad)

Ako se na web stranicu prijavimo sa super admin korisničkim podacima tada možemo vidjeti da ćemo imati više opcija od onih osnovnih kao što je pregled posta ili brisanje istog. Možemo vidjeti da sada imamo pristup pregledu tablice korisnici to jest da su sada u tablici ispisani podaci. U navigacijskoj traci uz klasičnu odjavu imamo i opcije meta podataka, dodaj korisnika i dodaj post. Opcija dodaj post ovoga puta se malo razlikuje što možemo vidjeti na sljedećoj slici.

Dodaj novi post

Naslov:

Sadržaj:

Status:

Draft

Dodaj post

Slika 16: Dodaj post s dodatnom opcijom (snimka zaslona, vlastiti rad)

Sada uz klasičan unos naslova i sadržaja posta možemo vidjeti da se prikazuje i dodatna opcija Status koja se pojavljuje kao skočni izbornik gdje se može birati Drafts, Archived ili Published.

Dodaj novog korisnika

Korisničko ime:

Lozinka:

Email:

Uloga:

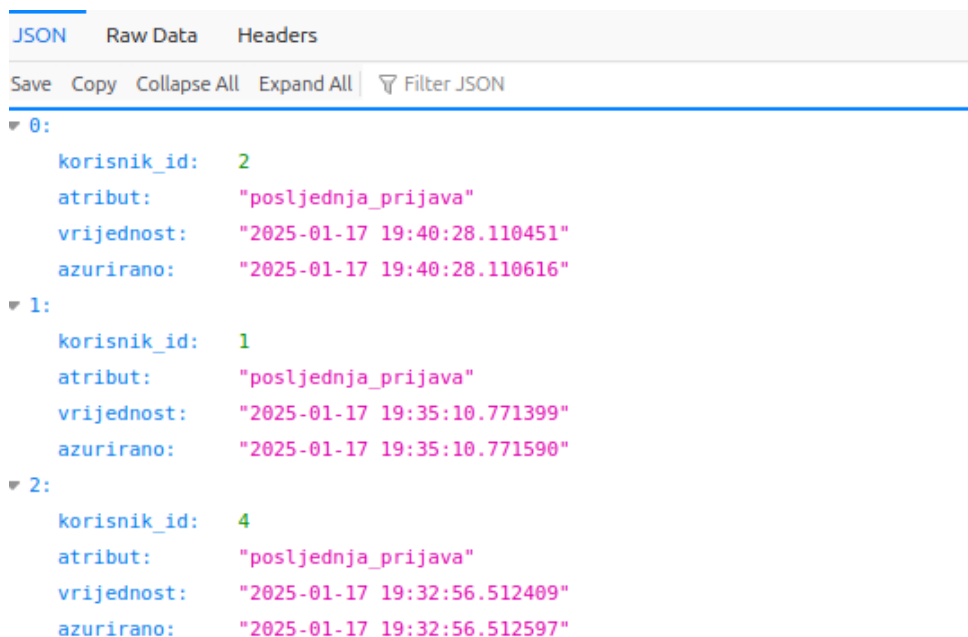
Admin

Inherits:

Dodaj korisnika

Slika 17: Dodaj novog korisnika(snimka zaslona, vlastiti rad)

Super admin ima opciju kreiranja novog korisnika, unosom potrebnih podataka super admin odmah kreira korisnika koji će se pojaviti i spremi u tablici korisnici.



Slika 18: Sadržaj json datoteke(snimka zaslona, vlastiti rad)

Kako se svaki korisnik logira u stranicu to se bilježi u JSON datoteku čiji podaci su vidljivi na ovoj slici.

Korisnik ID	Atribut	Vrijednost	Azurirano
2	posljednja_prijava	2025-01-17 19:45:08.415086	2025-01-17 19:45:08.415227
1	posljednja_prijava	2025-01-17 19:35:10.771399	2025-01-17 19:35:10.771590
4	posljednja_prijava	2025-01-17 19:32:56.512409	2025-01-17 19:32:56.512597

Slika 19: Ispis meta podataka(snimka zaslona, vlastiti rad)

Takvi podaci iz json datoteke prikazuju se na web stranici kod super admina pod opcijom Meta podaci. Možemo vidjeti kako su se u tablici prikazali isti podaci koji su ispisani u JSON datoteci.

6. Zaključak

U ovom projektu uspješno su implementirane funkcionalnosti potrebne za upravljanje različitim vrstama korisnika unutar baze podataka i web-aplikacije. Kroz korištenje tehnologija PostgreSQL za upravljanje bazom podataka te Flask, HTML, CSS i JSON za izgradnju korisničkog sučelja i povezivanje s bazom, ostvarena je funkcionalna i fleksibilna platforma prilagođena specifičnostima aplikacijske domene.

Odabrane tehnologije pokazale su se iznimno prikladnima za implementaciju aplikacije ove vrste. PostgreSQL pružio je pouzdano rješenje za obradu i pohranu podataka s naprednim mogućnostima poput podrške za relacijski i objektno-relacijski pristup. Njegova fleksibilnost omogućila je jednostavno modeliranje odnosa među tablicama, kao i implementaciju složenijih funkcionalnosti poput provjera prava pristupa i pohrane meta podataka. Flask se pokazao kao lagan i moćan okvir za razvoj web-aplikacija. Njegova modularnost i podrška za integraciju s različitim bazama podataka omogućila je jednostavnu implementaciju RESTful API-ja za interakciju korisnika s podacima. HTML i CSS osigurali su korisničko sučelje prilagođeno različitim vrstama korisnika, dok je JSON bio ključan za prijenos podataka između klijentske strane i poslužitelja.

Glavne prednosti odabrane platforme su prilagodljivost, jednostavnost razvoja, PostgreSQL i Flask omogućuju implementaciju sigurnosnih mehanizama, poput autentifikacije, autorizacije i kontrole pristupa. Međutim, iako je aplikacija funkcionalna uočena su određena ograničenja. Jedan od njih je da Flask pri povećanju broja korisnika može postati ograničavajući faktor i može doći do pterećenja aplikacije. Trenutna implementacija koristi osnovne metode autentifikacije i autorizacije, ali bi za produkcijsko okruženje bilo potrebno implementirati naprednije mjere zaštite, poput enkripcije podataka i dvofaktorske autentifikacije.

Sve u svemu PostgreSQL i Flask pokazali su se kao idealna kombinacija za brzu i učinkovitu implementaciju aplikacija u domeni upravljanja podacima, dok HTML, CSS i JSON omogućuju kvalitetnu interakciju s krajnjim korisnicima

7. Literatura

- 1 Steiner A. (1998.) "A Generalisation Approach to Temporal Data Models and their Implementations" Preuzeto 17.1.2025. s <https://www.research-collection.ethz.ch/bitstream/handle/20.500.11850/143598/1/eth-22598-01.pdf>
- 2 De Tre G., De Caluwe R. (1999.) "A generalized object-oriented database model with generalized constraints" Preuzeto 17.1.2025. s <https://ieeexplore.ieee.org/abstract/document/781719>
- 3 Okreša Đurić B., Schatten M. Materijali s predmeta Teorije baza podataka, 2024.
- 4 Grinberg M. (2018.) "Flask Web Development" United States of America O'Reilly Media, Inc. [Online] https://books.google.fi/books?hl=hr&lr=&id=cVlPDwAAQBAJ&oi=fnd&pg=PT25&dq=flask&ots=xPEZcl2rbY&sig=tGTGVVGD1JpkRBzxPqCkgonpJws&redir_esc=y#v=onepage&q&f=false
- 5 Worsley J.C., Drake J.D. (2002.) "Practical PostgreSQL" United States of America O'Reilly Media, Inc. https://books.google.fi/books?hl=hr&lr=&id=G8dh95j5NgcC&oi=fnd&pg=PR4&dq=postgresql+database&ots=8U54GSGSj8&sig=GqlWqgu3WEr_a43nssdFdhSTpJQ&redir_esc=y#v=onepage&q=postgresql%20database&f=false
- 6 Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification [PDF Online] Preuzeto 17.1.2025. s <https://www.w3.org/TR/CSS2/css2.pdf>
- 7 Introducing JSON [Online] Preuzeto 17.1.2025. s <https://www.json.org/json-en.html>

Popis slika

1.	ERA dijagram (draw.io, vlastiti rad)	5
2.	Funkcija i tablica korisnici (snimka zaslona, vlastiti rad)	6
3.	Inherits i tablica postovi(snimka zaslona, vlastiti rad)	7
4.	Funkcija i korisnici (snimka zaslona, vlastiti rad)	7
5.	Inherits (snimka zaslona, vlastiti rad)	8
6.	Inherits (snimka zaslona, vlastiti rad)	8
7.	Meta podaci (snimka zaslona, vlastiti rad)	9
8.	Početno ručno uneseni podaci u tablice (snimka zaslona, vlastiti rad)	9
9.	Login(snimka zaslona, vlastiti rad)	14
10.	Prikaz stranice za user novost(snimka zaslona, vlastiti rad)	14
11.	Prikaz stranice za moderator(snimka zaslona, vlastiti rad)	14
12.	Dodaj običan post(snimka zaslona, vlastiti rad)	15
13.	Prikaz stranice za admin(snimka zaslona, vlastiti rad)	16
14.	Prikaz stranice za super admina(snimka zaslona, vlastiti rad)	16
15.	Prikaz stranice za super admina(snimka zaslona, vlastiti rad)	16
16.	Dodaj post s dodatnom opcijom(snimka zaslona, vlastiti rad)	17
17.	Dodaj novog korisnika(snimka zaslona, vlastiti rad)	18
18.	Sadržaj json datoteke(snimka zaslona, vlastiti rad)	19
19.	Ispis meta podataka(snimka zaslona, vlastiti rad)	19