SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Roberto Šandro

APLIKACIJA ZA UPRAVLJANJE KORISNIČKIM RAČUNIMA U OBJEKTNO-RELACIJSKOJ BAZI (POSTGRESQL) S WEB SUČELJEM

PROJEKT

TEORIJA BAZA PODATAKA

Varaždin, 2026.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Roberto Šandro

Matični broj: 0016159806

Studij: Baze podataka i baze znanja

APLIKACIJA ZA UPRAVLJANJE KORISNIČKIM RAČUNIMA U OBJEKTNO-RELACIJSKOJ BAZI (POSTGRESQL) S WEB SUČELJEM

PROJEKT

Mentor:

doc. dr. sc. Bogdan Okreša Đurić

Varaždin, siječanj 2026.

Roberto Šandro

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.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI Radovi

Sažetak

U radu je razvijen sustav za upravljanje korisničkim računima koji obuhvaća uloge, prava pristupa i evidenciju aktivnosti (audit). Baza podataka implementirana je u PostgreSQL-u kao objektno-relacijskom sustavu, pri čemu su korišteni pogledi (views) i okidači (triggers) za automatizaciju bilježenja događaja. Aplikacijski dio razvijen je u Pythonu korištenjem Flask okvira, a komunikacija s bazom ostvarena je putem psycopg2 adaptera. Implementiran je RBAC model (Role-Based Access Control) kroz relacije korisnik–uloga i uloga–dozvola, uz autentikaciju s hashiranjem lozinki. Rad prikazuje konceptualni i logički model, ključne dijelove implementacije te primjere korištenja i provjeru funkcionalnosti.

Ključne riječi: PostgreSQL; objektno-relacijske baze; RBAC; uloge i dozvole; okidači; pogledi; Flask; psycopg2; audit log

Sadržaj

1. Uvod

Upravljanje korisničkim računima i kontrola pristupa predstavlja temeljnu funkcionalnost većine suvremenih informacijskih sustava. Čim aplikacija sadrži više od jednog korisnika, pojavljuje se potreba da se jasno definira tko smije pristupiti određenim podacima i tko smije izvršavati pojedine operacije. Neadekvatno postavljena prava pristupa često su uzrok sigurnosnih incidenata, neovlaštenih izmjena podataka i narušavanja integriteta sustava, a posljedično i gubitka povjerenja korisnika.

U okviru ovog projekta razvijena je web aplikacija za upravljanje korisničkim računima koja obuhvaća:

- evidenciju korisnika (kreiranje i brisanje),
- dodjelu uloga korisnicima,
- definiranje dozvola i dodjelu dozvola ulogama,
- provjeru prava pristupa prilikom izvođenja operacija (RBAC),
- automatsko bilježenje ključnih događaja kroz audit zapis.

Aplikacijski dio razvijen je u programskom jeziku Python korištenjem Flask okvira, dok je baza podataka implementirana u PostgreSQL-u. Poseban naglasak stavljen je na korištenje mogućnosti baze podataka koje nadilaze minimalni relacijski sloj, poput pogleda (engl. *views*) i okidača (engl. *triggers*). Takvim pristupom dio poslovne logike i kontrole konzistentnosti podataka smješta se izravno u bazu, čime se povećava pouzdanost sustava neovisno o aplikacijskom sloju.

Motivacija za izbor PostgreSQL-a proizlazi iz njegove stabilnosti, potpore naprednim SQL mehanizmima i česte primjene u praksi [1]. Flask je odabran kao lagani web okvir koji omogućuje brzu izradu pregledne aplikacije uz jasan i modularan kod [2]. U sklopu rada prikazani su teorijski koncepti, model baze podataka, ključni dijelovi implementacije te primjeri korištenja, uz evaluaciju prednosti i ograničenja odabranog rješenja.

Cilj rada je razviti sustav koji jasno demonstrira kontrolu pristupa kroz RBAC model i primjenu objektno-relacijskih značajki PostgreSQL-a na domenu upravljanja korisničkim računima.

2. Opis aplikacijske domene

Aplikacijska domena ovog projekta obuhvaća sustav za upravljanje korisničkim računima s jasno definiranim razinama ovlasti. Takvi sustavi predstavljaju temelj gotovo svake web aplikacije koja omogućuje rad većem broju korisnika, osobito kada postoji potreba za razlikovanjem administrativnih i korisničkih funkcionalnosti.

Razvijena aplikacija namijenjena je centraliziranom upravljanju korisnicima, njihovim ulogama i pripadajućim pravima pristupa. Sustav je zamišljen kao jednostavno, ali funkcionalno rješenje koje demonstrira osnovne koncepte kontrole pristupa, uz naglasak na pouzdanost i preglednost.

2.1. Temeljni koncepti domene

Osnovni koncepti aplikacijske domene su:

- **Korisnik** – entitet koji se autentificira u sustav korištenjem korisničkog imena i lozinke. Korisnik predstavlja aktivnog sudionika sustava i izvršava radnje sukladno dodijeljenoj ulozi.
- **Uloga** – skup ovlasti koji definira razinu pristupa korisnika sustavu. Model sustava omogućuje da jedan korisnik može imati jednu ili više uloga (relacija korisnik–uloga realizirana je kao više-prema-više). U implementiranom web sučelju, prilikom dodavanja novog korisnika dodjeljuje se jedna uloga radi jednostavnosti, dok se model baze i autorizacijska logika mogu koristiti i za scenarij višestrukih uloga.
- **Dozvole** – implicitna prava koja proizlaze iz dodijeljene uloge. Dozvole nisu izravno vidljive korisniku, već se koriste unutar aplikacije za kontrolu dostupnih funkcionalnosti.
- **Audit zapis** – zapis o važnim događajima u sustavu, prvenstveno o kreiranju i brisanju korisničkih računa.

U sustavu su definirane tri osnovne uloge:

- **Administrator** – ima pravo dodavanja i brisanja korisnika te pregled svih podataka u sustavu.
- **Obični korisnik** – ima ograničen pristup sustavu i može koristiti samo osnovne funkcionalnosti.
- **Auditor** – nema pravo izmjene podataka, ali ima mogućnost pregleda audit zapisa.

2.2. Relacije i pravila domene

Veza između korisnika i uloga realizirana je kao relacija više-prema-više, čime je omogućeno da jedan korisnik može imati jednu ili više uloga, dok ista uloga može biti dodijeljena većem broju korisnika. U implementiranom web sučelju, prilikom dodavanja novog korisnika administrator odabire jednu ulogu radi jednostavnosti korištenja, dok model baze i autorizacijska logika podržavaju scenarij višestrukih uloga.

2.3. Audit i praćenje aktivnosti

Posebna pažnja u domeni posvećena je praćenju aktivnosti u sustavu. Svako dodavanje ili brisanje korisničkog računa automatski se bilježi u tablicu audit zapisa. Audit zapis sadrži informaciju o vrsti događaja i vremenu kada se događaj dogodio.

Audit zapis služi za evidenciju ključnih događaja nad korisničkim računima. U implementaciji se automatski bilježe događaji kreiranja i brisanja korisnika, pri čemu se u tablicu *audit_log* upisuje tip događaja (npr. *USER_CREATED*, *USER_DELETED*) i vrijeme izvršenja. Audit zapis referencira korisnika nad kojim se promjena dogodila.

3. Teorijski uvod

3.1. Objektno-relacijske baze podataka

Objektno-relacijske baze podataka predstavljaju proširenje klasičnih relacijskih baza koje, uz standardne tablice i relacije, podržavaju dodatne mehanizme za implementaciju složenije poslovne logike. Takav pristup omogućuje zadržavanje formalnih svojstava relacijskog modela, uz istovremenu veću fleksibilnost u radu s podacima [3].

PostgreSQL je primjer objektno-relacijskog sustava koji pruža širok skup naprednih mogućnosti, uključujući funkcije, okidače, poglede i proceduralni jezik PL/pgSQL. U kontekstu ovog projekta, te su mogućnosti iskorištene za automatizaciju određenih zadataka i povećanje pouzdanosti sustava.

3.2. Primjena objektno-relacijskog pristupa u projektu

U razvijenom sustavu objektno-relacijski pristup koristi se prvenstveno za:

- implementaciju audit zapisa pomoću okidača,
- centralizaciju pravila vezanih uz integritet podataka,
- izlaganje strukturiranih pogleda nad podacima aplikaciji.

Korištenjem okidača omogućeno je da se određeni događaji automatski bilježe u bazi podataka, bez potrebe da se takva logika ponavlja u aplikacijskom kodu. Time se smanjuje mogućnost pogreške i osigurava da su svi relevantni događaji evidentirani.

3.3. RBAC model kontrole pristupa

Kontrola pristupa u projektu temelji se na RBAC modelu (Role-Based Access Control). U tom modelu prava pristupa nisu dodijeljena izravno korisnicima, već proizlaze iz uloge koju korisnik posjeduje [4].

U implementaciji se RBAC provodi tako da se korisniku mogu pridružiti jedna ili više uloga, a dozvole se izračunavaju kao unija dozvola svih uloga koje korisnik posjeduje. U korisničkom sučelju pri kreiranju korisnika odabire se jedna uloga, no autorizacijski mehanizam podržava i scenarij višestrukih uloga.

3.4. Audit kao sigurnosni mehanizam

Audit zapis predstavlja važan sigurnosni mehanizam jer omogućuje praćenje ključnih događaja u sustavu. U ovom projektu audit se provodi na razini baze podataka pomoću okidača, koji se aktiviraju prilikom dodavanja i brisanja korisničkih računa.

Svaki audit zapis sadrži informaciju o vrsti događaja i vremenu izvršenja, čime se omogućuje naknadna analiza aktivnosti. Takav pristup povećava transparentnost rada sustava i predstavlja dobru praksu u razvoju aplikacija koje upravljaju korisničkim računima [5].

4. Model baze podataka

Model baze podataka osmišljen je s ciljem podrške funkcionalnostima upravljanja korisničkim računima, ulogama, dozvolama i evidencijom aktivnosti. Pri izradi modela naglasak je stavljen na konzistentnost podataka, jasno definirane relacije te mogućnost proširenja sustava bez potrebe za promjenom osnovne strukture baze.

Baza podataka implementirana je u sustavu PostgreSQL kao objektno-relacijska baza, pri čemu su korištene standardne relacijske strukture uz dodatne mehanizme poput pogleda i okidača.

4.1. Konceptualni model

Konceptualni model sustava temelji se na ERA pristupu (Entity–Relationship–Attribute). Model obuhvaća temeljne entitete potrebne za implementaciju kontrole pristupa i praćenje aktivnosti korisnika.

Glavni entiteti u modelu su:

- **Korisnik (User)**
- **Uloga (Role)**
- **Dozvola (Permission)**
- **Audit zapis (Audit_Log)**

Uz navedene entitete, u modelu su prisutne i pomoćne relacije koje omogućuju realizaciju veza više-prema-više između korisnika i uloga te uloga i dozvola.

4.2. Opis entiteta

4.2.1. Korisnik

Entitet *Korisnik* predstavlja osobu ili sustav koji se autentificira u aplikaciju. Svaki korisnik identificiran je jedinstvenim korisničkim imenom te posjeduje lozinku pohranjenu u obliku kriptografskog sažetka (hash). Uz osnovne identifikacijske podatke, za korisnika se bilježi i vrijeme kreiranja zapisa.

4.2.2. Uloga

Entitet *Uloga* definira razinu ovlasti unutar sustava. Uloga predstavlja skup dozvola koje određuju koje su funkcionalnosti dostupne korisniku. Primjeri uloga u sustavu su administrator, obični korisnik i auditor.

4.2.3. Dozvola

Entitet *Dozvola* predstavlja pojedinačno pravo pristupa određenoj funkcionalnosti sustava, poput pregleda korisnika, dodavanja korisnika ili pregleda audit zapisa. Dozvole se ne dodjeljuju izravno korisnicima, već isključivo ulogama.

4.2.4. Audit zapis

Entitet *Audit zapis* služi za evidenciju važnih događaja u sustavu. U ovoj implementaciji audit zapis bilježi događaje vezane uz kreiranje i brisanje korisničkih računa. Svaki audit zapis sadrži informaciju o vrsti događaja i vremenu kada se događaj dogodio, uz referencu na korisnika nad kojim je akcija izvršena.

4.3. Relacije između entiteta

Između entiteta u modelu definirane su sljedeće relacije:

- **Korisnik – Uloga**: relacija više-prema-više. Jedan korisnik može imati jednu ili više uloga, dok ista uloga može biti dodijeljena većem broju korisnika. Relacija je realizirana pomoćnom tablicom *user_roles*.
- **Uloga – Dozvola**: relacija više-prema-više. Jedna uloga može sadržavati više dozvola, a ista dozvola može biti dodijeljena više uloga. Relacija je realizirana pomoćnom tablicom *role_permissions*.
- **Korisnik – Audit zapis**: relacija jedan-prema-više. Jedan korisnik može imati više audit zapisa, dok se svaki audit zapis odnosi na najviše jednog korisnika.

4.4. Logički (relacijski) model

Na temelju konceptualnog modela definiran je relacijski model baze podataka koji se sastoji od sljedećih tablica:

- *users*
- *roles*
- *permissions*
- *user_roles*
- *role_permissions*
- *audit_log*

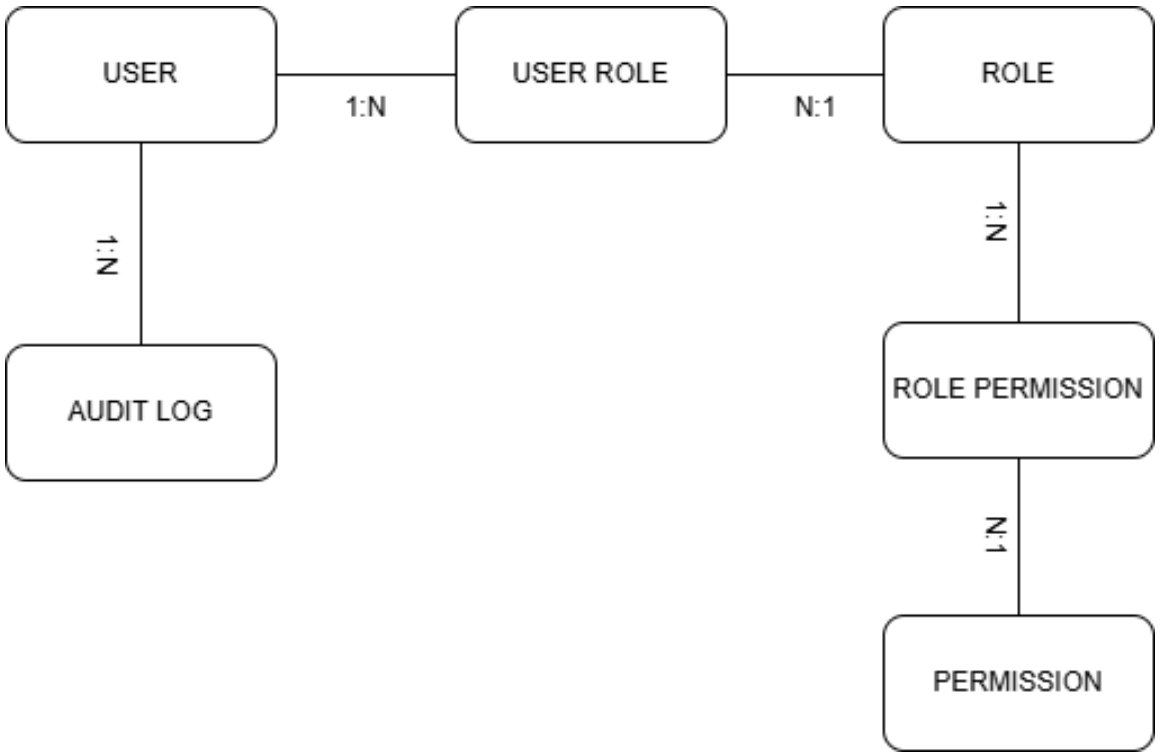
Tablice su povezane primarnim i stranim ključevima, čime se osigurava referencijalni integritet podataka. Pomoćne tablice koriste kompozitne primarne ključeve kako bi se spriječilo dupliciranje veza.

4.5. Pogledi i automatizacija

Za potrebe pojednostavljenja pristupa podacima i povećanja čitljivosti upita implementirani su pogledi (*views*). Pogledi omogućuju dohvat agregiranih i strukturiranih podataka, primjerice pregled korisnika s pripadajućim ulogama ili pregled audit zapisa s korisničkim imenima.

Automatizacija evidencije aktivnosti ostvarena je korištenjem okidača (*triggers*) koji se aktiviraju prilikom umetanja i brisanja zapisa u tablici korisnika. Na taj način osigurano je da se audit zapis vodi konzistentno i neovisno o aplikacijskom sloju.

Opisani model baze podataka predstavlja čvrstu osnovu za implementaciju kontrole pristupa i praćenje aktivnosti te omogućuje jednostavno proširenje sustava u budućnosti.



Slika 1: ERA dijagram sustava za upravljanje korisničkim računima

Na slici 1 prikazan je ERA dijagram sustava za upravljanje korisničkim računima. Dijagram obuhvaća temeljne entitete korisnik, uloga, dozvola i audit zapis, uz pripadajuće asocijativne entitete koji omogućuju realizaciju veza više-prema-više.

Entitet korisnik predstavlja subjekte koji se autentificiraju u sustav i koriste njegove funkcionalnosti. Uloga definira razinu ovlasti korisnika, dok dozvola predstavlja pojedinačno pravo pristupa određenoj funkcionalnosti sustava. Veza između korisnika i uloga realizirana je asocijativnim entitetom *user_role*, čime je omogućena dodjela jedne ili više uloga pojedinom koris-

niku.

Autorizacija je dodatno razrađena kroz vezu između uloga i dozvola, koja je ostvarena asocijativnim entitetom *role_permission*. Time je omogućeno fleksibilno upravljanje pravima pristupa bez izravnog vezivanja dozvola uz korisnike.

Entitet audit zapis povezan je s korisnikom relacijom jedan-prema-više te služi za evidenciju ključnih događaja u sustavu, poput kreiranja i brisanja korisničkih računa. Opisani ERA model predstavlja temelj za relacijski model baze podataka i implementaciju kontrole pristupa u aplikaciji.

5. Implementacija

Implementacija sustava sastoji se od dva glavna dijela: baze podataka implementirane u sustavu PostgreSQL i web aplikacije razvijene u programskom jeziku Python korištenjem Flask okvira. Sustav je osmišljen tako da se dio logike realizira na razini baze podataka, dok aplikacijski sloj služi za upravljanje korisničkim sučeljem, autentikacijom i autorizacijom.

5.1. Implementacija baze podataka

Baza podataka implementirana je kao objektno-relacijska baza u PostgreSQL-u. Struktura baze definirana je SQL skriptom koja sadrži tablice, primarne i strane ključeve te ograničenja integriteta. Posebna pažnja posvećena je očuvanju referencijalnog integriteta i dosljednosti podataka.

Za potrebe kontrole pristupa definirane su tablice korisnika, uloga i dozvola, uz pomoćne tablice koje omogućuju realizaciju veza više-prema-više. Time je omogućena fleksibilna dodjela uloga korisnicima i dozvola ulogama.

5.1.1. Pogledi (views)

Kako bi se pojednostavio dohvat podataka i smanjila složenost SQL upita u aplikacijskom sloju, implementirani su pogledi.

Primjerice, pogled za pregled korisnika objedinjuje podatke o korisnicima i njihovim ulogama, dok pogled za audit zapise omogućuje čitljiv prikaz zabilježenih događaja zajedno s korisničkim imenima. Korištenjem pogleda povećava se čitljivost koda i olakšava održavanje sustava.

5.1.2. Audit i okidači

Audit mehanizam implementiran je na razini baze podataka korištenjem PL/pgSQL funkcije i okidača. Okidači se aktiviraju automatski prilikom umetanja i brisanja zapisa u tablici korisnika te upisuju odgovarajući zapis u tablicu audit zapisa.

Ovakav pristup osigurava da se evidencija aktivnosti vodi neovisno o aplikacijskom sloju, čime se smanjuje mogućnost pogreške ili zaobilaženja audit mehanizma. Audit zapisi sadrže informaciju o vrsti događaja i vremenu izvršenja, što omogućuje naknadnu analizu aktivnosti u sustavu.

```sql
CREATE OR REPLACE FUNCTION log_user_action()
RETURNS TRIGGER AS $$
BEGIN
  IF TG_OP = 'INSERT' THEN
    INSERT INTO audit_log(user_id, action)
    VALUES (NEW.id, 'USER_CREATED');
    RETURN NEW;
  ELSIF TG_OP = 'DELETE' THEN
    INSERT INTO audit_log(user_id, action)
    VALUES (OLD.id, 'USER_DELETED');
    RETURN OLD;
  END IF;
  RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

Isječak koda 1: Audit funkcija u bazi podataka (bilježenje kreiranja i brisanja korisnika)

5.2. Implementacija aplikacijskog sloja

Aplikacijski sloj sustava razvijen je korištenjem Flask web okvira. Flask je odabran zbog svoje jednostavnosti i fleksibilnosti, što omogućuje jasnu organizaciju aplikacijske logike i brz razvoj funkcionalnosti.

Komunikacija s bazom podataka ostvarena je pomoću psycopg2 biblioteke, koja omogućuje izvođenje SQL upita i rad s transakcijama. Za dohvat podataka korišten je *RealDictCursor*, čime se rezultati upita mapiraju u rječnike, što pojednostavljuje rad s podacima u aplikaciji.

5.2.1. Autentikacija korisnika

Autentikacija korisnika temelji se na provjeri korisničkog imena i lozinke. Lozinke se u bazu podataka ne pohranjuju u izvornom obliku, već se prije pohrane kriptografski sažimaju (hashiraju) korištenjem odgovarajućih funkcija. Prilikom prijave, unesena lozinka uspoređuje se s pohranjenim sažetkom.

Nakon uspješne autentikacije, u korisničkoj sesiji pohranjuju se osnovni podaci o korisniku, uključujući identifikator korisnika i pripadajuće dozvole.

5.2.2. Autorizacija i RBAC

Autorizacija u sustavu implementirana je primjenom RBAC modela. Svakom korisniku dodijeljene su jedna ili više uloga, a svaka uloga ima pripadajući skup dozvola. Efektivne dozvole korisnika izračunavaju se kao unija dozvola svih uloga koje korisnik posjeduje.

U aplikaciji je implementiran mehanizam provjere dozvola pomoću dekoratora, koji prije izvršavanja pojedine funkcionalnosti provjerava ima li korisnik odgovarajuće pravo pristupa. Time je osigurano da su pojedine funkcije dostupne isključivo korisnicima s odgovarajućim ovlastima.

```python
def permission_required(permission):
    def decorator(f):
        @wraps(f)
        def wrapper(*args, **kwargs):
            if "user_id" not in session:
                return redirect(url_for("login"))
            if permission not in session.get("permissions", []):
                return "Access denied", 403
            return f(*args, **kwargs)
        return wrapper
    return decorator
```

Isječak koda 2: Dekorator za provjeru dozvola (RBAC autorizacija) u aplikacijskom sloju

5.2.3. Korisničko sučelje

Korisničko sučelje aplikacije implementirano je korištenjem HTML predložaka. Sučelje omogućuje prijavu korisnika, pregled korisnika i njihovih uloga, dodavanje i brisanje korisnika te pregled audit zapisa.

Prikaz pojedinih dijelova sučelja uvjetovan je dozvolama korisnika. Na taj način se već na razini korisničkog sučelja onemogućuje pristup funkcionalnostima za koje korisnik nema ovlasti, čime se dodatno povećava sigurnost i preglednost sustava.

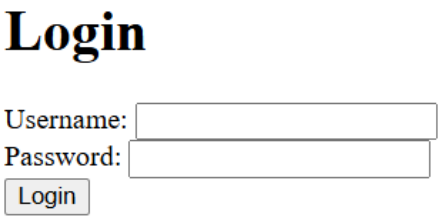
6. Primjeri korištenja

U ovom poglavlju prikazani su tipični primjeri korištenja razvijenog sustava za upravljanje korisničkim računima. Primjeri ilustriraju osnovni tijek rada aplikacije te demonstriraju ispravnost implementacije autentikacije, autorizacije i audit mehanizma.

6.1. Prijava korisnika u sustav

Prvi korak u korištenju aplikacije je prijava korisnika u sustav. Korisnik unosi svoje korisničko ime i lozinku putem obrasca za prijavu. Nakon slanja podataka, aplikacija provjerava vjerodajnice u bazi podataka usporedbom unesene lozinke s pohranjenim kriptografskim sažetkom.

U slučaju uspješne autentikacije, korisniku se otvara početna stranica aplikacije, a u korisničkoj sesiji pohranjuju se identifikator korisnika i pripadajuće dozvole. Ako su uneseni podaci neispravni, pristup sustavu je odbijen.



Slika 2: Prijava korisnika u sustav

6.2. Pregled početne stranice

Nakon prijave korisniku se prikazuje početna stranica aplikacije. Sadržaj navigacijskog izbornika prilagođen je dozvolama prijavljenog korisnika. Na taj način korisnici vide samo one funkcionalnosti za koje imaju pravo pristupa.

Primjerice, korisnici s administrativnim ovlastima imaju pristup upravljanju korisnicima i ulogama, dok korisnici s ulogom auditora imaju mogućnost pregleda audit zapisa.

6.3. Pregled korisnika

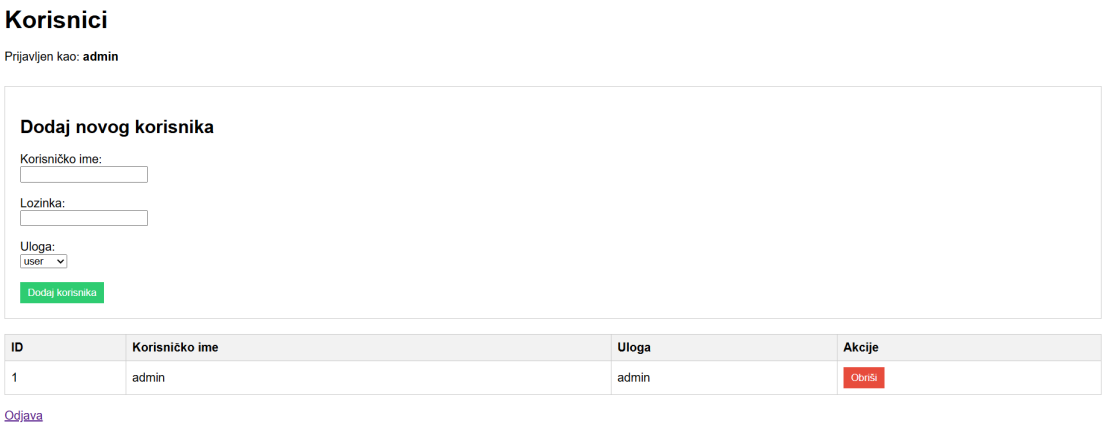
Korisnik koji posjeduje dozvolu za pregled korisnika može pristupiti popisu svih korisničkih računa u sustavu. U tabličnom prikazu prikazani su osnovni podaci o korisnicima, uključujući korisničko ime i pripadajuće uloge.

Uloge korisnika dohvaćaju se iz baze podataka agregiranim upitom, čime se omogućuje prikaz jedne ili više uloga po korisniku u čitljivom obliku.

6.4. Dodavanje novog korisnika

Administrator sustava ima mogućnost dodavanja novog korisnika putem odgovarajućeg obrasca. Prilikom dodavanja unose se korisničko ime, lozinka i uloga korisnika. Prije pohrane u bazu podataka, lozinka se kriptografski sažima.

Nakon uspješnog dodavanja korisnika, u bazi podataka automatski se generira audit zapis koji bilježi događaj kreiranja korisničkog računa. Time je osigurano da se sve važne promjene evidentiraju bez dodatne intervencije aplikacijskog koda.



Slika 3: Administratorsko sučelje za pregled i upravljanje korisnicima

6.5. Brisanje korisnika

Administrator ima mogućnost brisanja postojećeg korisničkog računa. Prije izvršavanja operacije brisanja, aplikacija zahtijeva potvrdu korisnika kako bi se spriječilo nenamjerno uklanjanje podataka.

Brisanjem korisnika iz baze podataka automatski se aktivira okidač koji generira audit zapis o brisanju korisničkog računa. Na taj način zadržava se evidencija o izvršenoj radnji, čak i nakon uklanjanja korisnika.

6.6. Pregled uloga i dozvola

Korisnici s odgovarajućim dozvolama mogu pregledavati popis definiranih uloga u sustavu. Za svaku ulogu prikazan je popis pripadajućih dozvola, čime je omogućen jasan uvid u strukturu ovlasti unutar sustava.

Ovakav prikaz olakšava administraciju i omogućuje jednostavnu provjeru koja prava su dodijeljena pojedinim ulogama.

6.7. Pregled audit zapisa

Korisnici s ulogom auditora ili administratori imaju mogućnost pregleda audit zapisa. Audit zapisi prikazani su u kronološkom redoslijedu, uz informaciju o vrsti događaja i vremenu izvršenja.

Pregled audit zapisa omogućuje nadzor nad aktivnostima u sustavu te predstavlja važan mehanizam za sigurnosnu analizu i provjeru ispravnosti rada aplikacije.

7. Zaključak

U okviru ovog projekta razvijen je funkcionalan sustav za upravljanje korisničkim računima temeljen na objektno-relacijskoj bazi podataka PostgreSQL i web aplikaciji izrađenoj u Flask okviru. Sustav omogućuje upravljanje korisnicima, ulogama i dozvolama te osigurava evidenciju aktivnosti putem audit zapisa.

Korištenje objektno-relacijskog pristupa pokazalo se pogodnim za implementaciju domene koja zahtijeva jasno definirane odnose i dodatna pravila na razini baze podataka. Prednost ovakvog pristupa očituje se u mogućnosti implementacije dijela poslovne logike izravno u bazi, čime se povećava sigurnost i konzistentnost podataka.

Ograničenja implementacije odnose se prvenstveno na skalabilnost i sigurnosne mehanizme koji bi u produkcijskom okruženju zahtijevali dodatne nadogradnje, poput naprednijeg upravljanja sesijama i detaljnije kontrole pristupa. Unatoč tome, razvijeni sustav uspješno demonstrira primjenu RBAC modela i objektno-relacijskih mogućnosti PostgreSQL-a u realnoj aplikacijskoj domeni.

Projekt može poslužiti kao temelj za daljnje proširenje sustava, primjerice dodavanjem novih uloga, integracijom s vanjskim autentikacijskim servisima ili razvojem složenijeg korisničkog sučelja.

Popis literature

[1] PostgreSQL Global Development Group, *PostgreSQL Documentation*, `https://www.postgresql.org/docs/`, Accessed: 2026-01-10, 2023.

[2] M. Grinberg, *Flask Web Development*, 2. izdanje. O'Reilly Media, 2018.

[3] R. Elmasri i S. B. Navathe, *Fundamentals of Database Systems*, 7. izdanje. Boston: Pearson, 2016.

[4] R. S. Sandhu, E. J. Coyne, H. L. Feinstein i C. E. Youman, „Role-Based Access Control Models," *IEEE Computer*, sv. 29, br. 2, str. 38–47, 1996.

[5] A. Behl, *Cybersecurity and Cyberwar*. Oxford University Press, 2017.

Popis slika

Popis isječaka koda