SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Jakov Malović

WEB APLIKACIJA ZA GEOGRAFSKU EVIDENCIJU POSJETA NACIONALNIH PARKOVA

PROJEKT

TEORIJA BAZA PODATAKA

Varaždin, 2026.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Jakov Malović

Matični broj: 0035238861

Studij: Baze podataka i baze znanja

WEB APLIKACIJA ZA GEOGRAFSKU EVIDENCIJU POSJETA NACIONALNIH PARKOVA

PROJEKT

Mentor:

prof. dr. sc. Markus Schatten

Varaždin, siječanj 2026.

Jakov Malović

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

Ovaj rad predstavlja razvoj web aplikacije za geografsku evidenciju posjeta nacionalnih parkova koristeći prostorne baze podataka. Sustav kombinira PostgreSQL bazu podataka s PostGIS ekstenzijom za pohranu i obradu geografskih podataka, Streamlit framework za korisničko sučelje te Folium biblioteku za interaktivnu kartografsku vizualizaciju. Implementiran je sustav autentifikacije korisnika s razlučivanjem uloga, gdje obični korisnici mogu evidentirati vlastite posjete parkovima, a administratori imaju pristup agregiranim statistikama o svim posjetama. Aplikacija koristi GeoJSON format za razmjenu geografskih podataka i SQL prostorne funkcije za analizu geografskih relacija. Rad detaljno opisuje arhitekturu sustava, implementaciju prostornih upita, integraciju kartografskih komponenti te evaluaciju performansi pri radu s velikim skupovima prostornih podataka.

Ključne riječi: GIS; PostGIS; PostgreSQL; Streamlit; Folium; prostorne baze podataka; web kartografija; geografska evidencija

Sadržaj

1. Uvod

1.1. Motivacija i kontekst

Geografski informacijski sustavi predstavljaju jedno od najbrže rastućih područja primijenjene informatike, s širokom primjenom u urbanom planiranju, okolišnoj zaštiti, turizmu i mnogim drugim domenama. Tradicionalno, GIS aplikacije bile su kompleksni desktop sustavi koji su zahtijevali specijaliziranu obuku i skupu programsku opremu. Međutim, razvojem web tehnologija i otvorenih prostornih baza podataka, postalo je moguće razvijati pristupačne web aplikacije koje donose GIS funkcionalnosti širem krugu korisnika.

Evidencija posjeta nacionalnih parkova predstavlja idealnu domenu za primjenu GIS tehnologija. Geografska komponenta je inherentna problemu jer svaki park ima točno definirane geografske granice i poziciju. Korisnici su zainteresirani ne samo za popise parkova koje su posjetili, već i za vizualni prikaz tih posjeta na karti. Administratori pak trebaju prostornu analizu podataka kako bi razumjeli obrasce posjeta i optimizirali upravljanje parkovima.

Tradicionalni pristupi evidenciji posjeta koristili su jednostavne relacijske baze podataka bez geografske komponente. Informacije o lokacijama pohranjivale su se kao tekstualni opisi ili, u najboljem slučaju, kao par koordinata u zasebnim stupcima. Ovakav pristup otežava prostornu analizu jer standardni SQL nema ugrađenu podršku za geografske operacije poput izračuna udaljenosti, provjere preklapanja područja, ili pronalaženja entiteta unutar određenog radijusa.

PostGIS ekstenzija za PostgreSQL rješava ove probleme dodajući podršku za geografske tipove podataka i funkcije. Omogućuje pohranu kompleksnih geometrijskih oblika poput poligona koji predstavljaju granice parkova, te pruža stotine funkcija za prostornu analizu. Integracija PostGIS-a s modernim web framework-ovima omogućuje razvoj sofisticiranih GIS aplikacija s relativno jednostavnim kodom.

1.2. Cilj rada

Glavni cilj ovog rada je razviti funkcionalnu web aplikaciju za geografsku evidenciju posjeta nacionalnih parkova koja demonstrira primjenu prostornih baza podataka u praktičnom kontekstu. Aplikacija mora omogućiti korisnicima intuitivnu interakciju s geografskim podacima kroz kartu, bez potrebe za razumijevanjem tehničkih detalja GIS-a.

Specifični ciljevi rada obuhvaćaju projektiranje i implementaciju prostorne baze podataka koja pohranjuje geografske granice županija i parkova koristeći PostGIS tipove podataka. Baza mora podržavati efikasno pohranjivanje i dohvaćanje kompleksnih poligona s tisućama točaka. Zatim, cilj je razviti sustav autentifikacije korisnika s razlučivanjem uloga gdje obični korisnici imaju pristup samo vlastitim podacima, dok administratori mogu vidjeti agregirane statistike svih korisnika.

Također, potrebno je implementirati interaktivno kartografsko sučelje koristeći Folium

biblioteku koje prikazuje županije, parkove i markere s različitim vizualnim indikatorima ovisno o statusu posjete. Sučelje mora podržavati zoom, pan i tooltip funkcionalnosti. Dodatno, cilj je optimizirati performanse aplikacije pri radu s velikim geometrijama korištenjem tehnika poput simplifikacije geometrije i cacheiranja rezultata upita.

Konačno, rad nastoji analizirati prednosti i nedostatke PostGIS pristupa u odnosu na alternative kao što su samostojeći GIS serveri ili cloud GIS servisi, te evaluirati skalabilnost sustava.

1.3. Struktura rada

Rad je organiziran u šest poglavlja koja sistematski pokrivaju sve aspekte razvoja aplikacije. Nakon ovog uvodnog poglavlja, drugo poglavlje obrađuje teorijski okvir potreban za razumijevanje geografskih informacijskih sustava. Objašnjavaju se osnovni koncepti GIS-a, prostorni tipovi podataka, standardni formati za razmjenu geografskih podataka, te principi web kartografije.

Treće poglavlje opisuje korištene alate i tehnologije. Detaljno se objašnjava PostgreSQL baza podataka i PostGIS ekstenzija, Streamlit framework za razvoj web aplikacija, Folium biblioteka za kartografsku vizualizaciju, te psycopg2 adapter za komunikaciju s bazom podataka. Također se prikazuje arhitektura sustava i dizajn baze podataka.

Četvrto poglavlje predstavlja središnji dio rada i detaljno opisuje implementaciju svih komponenti aplikacije. Obrađuje se projektiranje i kreiranje prostorne baze podataka, implementacija sustava autentifikacije, razvoj funkcija za dohvaćanje i obradu geografskih podataka, kreiranje kartografskog sučelja, te implementacija funkcionalnosti za evidentiranje posjeta.

Peto poglavlje analizira performanse sustava i evaluira različite optimizacijske tehnike. Mjere se vremena izvršavanja prostornih upita za različite veličine geometrija, testira se utjecaj simplifikacije na performanse i vizualnu kvalitetu, te se analizira skalabilnost sustava.

Šesto poglavlje daje zaključak rada, sažimajući ključne doprinose i evaluirajući postignute ciljeve. Također se predlažu moguća buduća proširenja i poboljšanja sustava.

2. Teorijski okvir

2.1. Geografski informacijski sustavi

2.1.1. Definicija i komponente GIS-a

Geografski informacijski sustav može se definirati kao računalni sustav za prikupljanje, pohranu, analizu i vizualizaciju geografskih podataka. GIS integrira geografske podatke s atributnim podacima, omogućujući korisnicima da postavljaju prostorna pitanja poput gdje se nešto nalazi, što se nalazi na određenoj lokaciji, ili kako su različiti entiteti prostorno povezani.

Svaki GIS sustav sastoji se od pet osnovnih komponenti. Prva komponenta je hardver, odnosno računalna oprema na kojoj GIS softver radi. Moderna web GIS aplikacija može raditi na raznolikim platformama od mobilnih uređaja do moćnih servera. Druga komponenta je softver koji pruža funkcionalnosti za pohranu, analizu i prikaz geografskih podataka. U našem slučaju, to uključuje PostgreSQL s PostGIS ekstenzijom, Streamlit aplikaciju, i Folium biblioteku.

Treća komponenta su podaci koji uključuju i geografske i atributne informacije. Geografski podaci opisuju gdje se nešto nalazi, dok atributni podaci opisuju što je to i koje karakteristike ima. Na primjer, za nacionalni park geografski podaci uključuju granice parka, dok atributni podaci uključuju ime parka, površinu, broj posjetitelja, i slično. Četvrta komponenta su metode i procedure koje definiraju kako se podaci prikupljaju, obrađuju i analiziraju. Peta komponenta su ljudi koji koriste sustav, od običnih korisnika do GIS analitičara i administratora. [1]

2.1.2. Prostorni tipovi podataka

U GIS sustavima, geografski objekti predstavljaju se pomoću geometrijskih primitiva. Najjednostavniji tip je točka koja predstavlja jednuznačnu lokaciju u prostoru definiranu koordinatama. Točke se koriste za predstavljanje objekata čija veličina nije značajna u danom mjerilu, poput gradova na karti države ili stabala u parku.

Linija ili polyline je niz povezanih točaka koji predstavlja linearne značajke poput cesta, rijeka ili granica. Svaki segment linije definiran je početnom i završnom točkom, a kompleksne linije sastoje se od mnoštva takvih segmenata. Poligon je zatvorena površina definirana vanjskim prstenom i opcionalnim unutarnjim prstenovima koji predstavljaju rupe. Poligoni se koriste za predstavljanje područja poput parcela, jezera, šuma ili, u našem slučaju, granica županija i parkova.

PostGIS podržava sve ove tipove kroz geometry i geography tipove podataka. Geometry tip koristi Kartezijanski koordinatni sustav i pogodan je za planarnu geometriju na manjim područjima. Geography tip koristi sferični koordinatni sustav i precizniji je za velike udaljenosti jer uzima u obzir zakrivljenost Zemlje. U našoj implementaciji koristimo geometry tip jer radimo s relativno malim područjem Hrvatske gdje je razlika zanemariva.

2.1.3. Koordinatni sustavi i projekcije

Svaki geografski podatak mora biti definiran u nekom koordinatnom sustavu. Geografski koordinatni sustav koristi geografsku širinu i dužinu za definiranje pozicija na Zemljinoj površini. Širina mjeri se u stupnjevima sjever ili jug od ekvatora, dok se dužina mjeri istok ili zapad od primarnog meridijana. Ovaj sustav je intuitivan ali ima nedostatak što udaljenosti i površine nisu konzistentne - jedan stupanj dužine na ekvatoru predstavlja drugačiju udaljenost nego na polu.

Projekcijski koordinatni sustav transformira sferične koordinate u ravninski sustav. Svaka projekcija uključuje iskrivljenja jer nije moguće prikazati sfernu površinu na ravnini bez distorzije. Različite projekcije optimizirane su za očuvanje različitih svojstava poput površine, oblika, udaljenosti ili smjera. Za Hrvatsku, standardna projekcija je HTRS96 koja je definirana EPSG kodom pet sedam šest sedam tri.

Web kartografske aplikacije tipično koriste Web Mercator projekciju, označenu EPSG kodom tri osam pet osam. Ova projekcija je optimizirana za prikaz na web mapama jer očuvava oblike i kutove, što čini navigaciju intuitivnom. Međutim, značajno iskrivljuje površine posebno blizu polova. U našoj aplikaciji, PostGIS automatski upravlja transformacijama između koordinatnih sustava kada je potrebno.

2.2. PostGIS i prostorne baze podataka

2.2.1. Osnove PostGIS-a

PostGIS je prostorna ekstenzija za PostgreSQL bazu podataka koja dodaje podršku za geografske objekte. Ekstenzija uvodi nove tipove podataka za pohranu geometrija, indekse za efikasno pretraživanje prostornih podataka, i stotine funkcija za prostornu analizu. PostGIS implementira specifikacije Open Geospatial Consortium-a što osigurava interoperabilnost s drugim GIS sustavima.

Instalacija PostGIS-a vrši se kao ekstenzija postojeće PostgreSQL baze. Nakon instalacije, ekstenzija se aktivira u bazi korištenjem CREATE EXTENSION naredbe. Ovo dodaje nove tipove podataka i funkcije u shemu baze. PostGIS također kreira posebne tablice metapodataka koje prate geografske stupce i njihove koordinatne sustave.

Osnovni tip podataka je geometry koji može pohraniti različite geometrijske oblike. Svaka geometrija ima pridružen SRID koji identificira njezin koordinatni sustav. PostGIS omogućuje miješanje različitih geometrijskih tipova i SRID-ova u istoj bazi, s automatskim transformacijama po potrebi. Također pruža validation funkcije koje osiguravaju da su geometrije valjane prema OGC specifikacijama. [2]

2.2.2. Prostorni indeksi

Efikasno pretraživanje prostornih podataka zahtijeva specijalizirane indeksne strukture. Standardni B-tree indeksi koji se koriste za skalarne podatke nisu prikladni za višedimenzionalne prostorne podatke. PostGIS koristi R-tree indekse implementirane kroz GiST mehanizam PostgreSQL-a.

R-tree indeks funkcionira tako da grupira prostorno bliske objekte u hijerarhijsku strukturu. Svaki node u tree-u predstavlja minimalni pravokutnik koji obuhvaća sve objekte u tom node-u. Pretraživanje počinje od korijena i spušta se samo u one node-ove čiji pravokutnici se preklapaju s traženim područjem. Ovo drastično smanjuje broj geometrija koje moraju biti detaljno provjerene.

Kreiranje prostornog indeksa vrši se CREATE INDEX naredbom s USING GIST klauzulom. Nakon kreacije, PostgreSQL query planer automatski koristi indeks kada evaluira prostorne uvjete u WHERE klauzuli. Indeks značajno ubrzava operacije poput pronalaženja svih parkova unutar određene županije ili pronalaženja najbližeg parka od date točke.

2.2.3. Prostorne funkcije

PostGIS pruža opsežan skup funkcija za rad s geografskim podacima. Geometrijske funkcije konstruiraju nove geometrije iz postojećih. Na primjer, ST Centroid vraća centroid geometrije, ST Buffer stvara buffer zonu oko geometrije, a ST Union kombinira više geometrija u jednu.

Prostorne relacijske funkcije provjeravaju odnose između geometrija. ST Contains provjerava sadržava li jedna geometrija drugu, ST Intersects provjerava preklapaju li se dvije geometrije, a ST Distance izračunava najkraću udaljenost između dvije geometrije. Ove funkcije koriste prostorne indekse za efikasno izvršavanje.

Funkcije za obradu geometrija modificiraju geometrije na različite načine. ST Simplify smanjuje broj točaka u geometriji očuvavajući njezin opći oblik, što je korisno za optimizaciju performansi. ST Transform pretvara geometriju iz jednog koordinatnog sustava u drugi. ST AsGeoJSON konvertira PostGIS geometriju u GeoJSON format pogodan za web aplikacije.

2.3. GeoJSON format

2.3.1. Struktura GeoJSON-a

GeoJSON je otvoreni standard za kodiranje geografskih struktura podataka koristeći JSON format. Standard definira nekoliko tipova geometrijskih objekata uključujući Point, LineString, Polygon, MultiPoint, MultiLineString, i MultiPolygon. Svaki geometrijski objekt specificira svoj tip i koordinate.

Feature je objekt koji kombinira geometriju s atributima. Feature sadrži geometry svojstvo s geometrijskim objektom i properties svojstvo s proizvoljnim atributima. Na primjer, feature

može predstavljati park s poligonom koji definira granice i svojstvima poput imena, površine i datuma osnivanja.

FeatureCollection je niz feature-a. Ovo je najčešći top-level objekt u GeoJSON dokumentima jer omogućuje predstavljanje više geografskih entiteta u jednoj strukturi. U našoj aplikaciji, županije i parkovi dohvaćaju se kao FeatureCollection objekti koji se zatim prosljeđuju Folium biblioteci za vizualizaciju.

2.3.2. Prednosti GeoJSON-a

GeoJSON je postao de facto standard za razmjenu geografskih podataka na webu iz nekoliko razloga. Prvi razlog je jednostavnost formata. JSON je široko razumljiv i podržan u svim programskim jezicima. GeoJSON proširuje JSON s minimalnom dodatnom kompleksnošću. Drugi razlog je kompaktnost. GeoJSON je tekstualni format koji je razumljiv ljudima ali i dovoljno kompaktan za mrežni prijenos.

Treći razlog je široka podrška alata. Gotovo sve moderne kartografske biblioteke, od Leaflet-a preko Folium-a do OpenLayers-a, imaju ugrađenu podršku za GeoJSON. PostGIS pruža funkcije za konverziju između internog formata i GeoJSON-a. Četvrti razlog je mogućnost direktne upotrebe u JavaScript-u. GeoJSON dokument je valjan JavaScript objekt koji se može direktno koristiti bez parsiranja. [3]

2.4. Web kartografija

2.4.1. Slippy maps koncept

Moderne interaktivne web karte koriste koncept poznat kao slippy maps gdje se karta sastoji od tile-ova. Umjesto renderiranja cijele karte kao jedne velike slike, karta se dijeli na male kvadratne slike zvane tile-ovi. Svaki zoom level ima svoj skup tile-ova s odgovarajućom rezolucijom. Na zoom level-u nula, cijela svijet stane u jedan tile. Svaki sljedeći zoom level udvostručuje broj tile-ova u svakom smjeru.

Kada korisnik gleda kartu, browser učitava samo tile-ove koji su trenutno vidljivi. Kako korisnik pomiče kartu ili zoomira, novi tile-ovi se učitavaju dinamički. Ovo omogućuje glatku navigaciju čak i za karte cijelog svijeta jer se u svakom trenutku učitava samo mali podskup podataka.

Tile-ovi se tipično serviraju s tile servera koji mogu biti komercijalnog ili open source karaktera. Popularne opcije uključuju OpenStreetMap tile servere, Mapbox, Google Maps, i CartoDB. U našoj aplikaciji koristimo CartoDB Positron tile-ove koji imaju svijetlu, minimalističku estetiku koja dobro prikazuje granice županija.

2.4.2. Kartografska vizualizacija

Efikasna kartografska vizualizacija zahtijeva pažljivo dizajniranje vizualnih elemenata. Boje moraju biti odabrane tako da pružaju dobar kontrast i jasno komuniciraju informacije. U našoj aplikaciji, županije su prikazane bijelom bojom s tamno plavim granicama što ih jasno razlikuje. Parkovi su prikazani narančastom bojom što ih vizualno ističe.

Markeri koriste boje za kodiranje informacija o statusu. Zeleni markeri označavaju parkove koje je korisnik posjetio, crveni označavaju neposjećene parkove, a plavi markeri pokazuju administratorima ukupan broj posjeta. Ikone na markerima također nose značenje - kvačica za posjećene parkove, X za neposjećene, i grupa ljudi za administrator prikaz.

Interaktivnost je ključna komponenta moderne web kartografije. Tooltips prikazuju dodatne informacije kada korisnik pređe mišem preko elementa. Popups pružaju detaljnije informacije kada korisnik klikne na element. Zoom i pan kontrole omogućuju eksploraciju karte. Sve ove funkcionalnosti implementirane su kroz Folium biblioteku koja ih automatski genrira. [4]

3. Alati i tehnologije

3.1. PostgreSQL i PostGIS

3.1.1. PostgreSQL baza podataka

PostgreSQL je objektno-relacioni sustav za upravljanje bazama podataka otvorenog koda koji se razvija od devedesetih godina. Odabran je za ovaj projekt zbog nekoliko ključnih karakteristika. Prva je robusnost i pouzdanost. PostgreSQL ima reputaciju iznimno stabilnog sustava s ACID svojstvima koja garantiraju integritet podataka čak i u slučaju hardverskog kvara ili pada sustava.

Druga karakteristika je napredne mogućnosti upita. PostgreSQL podržava kompleksne upite s podupitima, window funkcijama, rekurzivnim upitima, i mnoge druge napredne SQL feature-e. Podrška za JSON tipove podataka omogućuje fleksibilnu pohranu polustrukturiranih podataka. Treća karakteristika je ekstenzibilnost. PostgreSQL ima moćan extension mehanizam koji omogućuje dodavanje novih tipova podataka, funkcija, operatora i indeksa bez modifikacije core koda.

Četvrta karakteristika je skalabilnost. PostgreSQL može efikasno rukovati bazama od nekoliko gigabajta do nekoliko terabajta. Podržava različite tehnike optimizacije uključujući tablespaces, particioniranje tablica, i paralelno izvršavanje upita. Peta karakteristika je aktivna zajednica i bogata ekosustav alata. Postoji mnoštvo biblioteka za pristup PostgreSQL-u iz različitih programskih jezika, grafičkih alata za administraciju, i dodatnih ekstenzija.

Za instalaciju na Linux sustavima, PostgreSQL je dostupan u standardnim package repository-ima. Na Ubuntu i Debian distribucijama instalacija uključuje instalaciju PostgreSQL servera i klijentskih alata. Nakon instalacije, potrebno je kreirati bazu podataka i korisnika koji će imati pristup. PostgreSQL koristi peer autentifikaciju za lokalne konekcije što znači da Unix korisnik automatski ima pristup PostgreSQL korisničkom računu istog imena. [5]

3.1.2. PostGIS ekstenzija

PostGIS je prostorna ekstenzija za PostgreSQL koja dodaje potpunu GIS funkcionalnost. Instalacija PostGIS-a vrši se kroz package manager na Linux sustavima. Nakon instalacije paketa, ekstenzija se aktivira u bazi pomoću CREATE EXTENSION postgis naredbe. Ovo dodaje više od tisuću novih funkcija u bazu, zajedno s novim tipovima podataka.

PostGIS se kontinuirano razvija s redovitim izdanjima koja donose nova poboljšanja performansi i funkcionalnosti. Verzija tri točka nula donijela je značajne optimizacije za velike geometrije i bolje performanse prostornih indeksa. Podrška za geography tip podataka omogućuje precizne kalkulacije na globalnoj razini. Integracija s GDAL bibliotekom omogućuje import i export različitih geografskih formata.

U našem projektu, PostGIS se koristi za pohranu granica županija i parkova kao polygoni.

Svaki poligon može sadržavati tisuće točaka koje definiraju kompleksne granice. PostGIS efikasno pohranjuje i indeksira ove podatke. Funkcije poput ST Centroid koriste se za izračun centara parkova gdje se postavljaju markeri. ST AsGeoJSON pretvara geometrije u format pogodan za slanje web aplikaciji.

3.2. Streamlit framework

3.2.1. Uvod u Streamlit

Streamlit je Python framework otvorenog koda za brzi razvoj web aplikacija za data science i machine learning projekte. Framework omogućuje kreiranje interaktivnih web aplikacija bez potrebe za znanjem HTML-a, CSS-a ili JavaScript-a. Programer piše običan Python kod, a Streamlit automatski generira responzivni web interface.

Osnovna filozofija Streamlit-a je jednostavnost. Umjesto da se eksplicitno upravlja stanjem aplikacije, Streamlit automatski rerenderira cijeli script od vrha do dna svaki put kada se nešto promijeni. Ovo eliminira mnoge kompleksnosti tradicionalnog web razvoja ali zahtijeva pažljivo dizajniranje kako bi se izbjegle nepotrebne rekalkulacije.

Streamlit pruža bogat skup widget-a za kreiranje korisničkog sučelja. Osnovni widgeti uključuju text input za unos teksta, slider za numeričke vrijednosti, selectbox za odabir iz liste, checkbox za boolean vrijednosti, i button za pokretanje akcija. Svi ovi widgeti automatski čuvaju svoje stanje između rerun-ova.

Za vizualizaciju podataka, Streamlit ima ugrađenu podršku za različite biblioteke. Može prikazati Pandas DataFrame-ove kao interaktivne tablice, Matplotlib i Plotly grafove, te prilagođene komponente poput mapa. Session state mehanizam omogućuje persistiranje podataka kroz multiple rerun-ove što je ključno za implementaciju autentifikacije i praćenja stanja korisnika.

3.2.2. Deployment i performanse

Streamlit aplikacije mogu se deploy-ati na različite načine. Za razvoj, aplikacija se pokreće lokalno koristeći streamlit run komandu. Za produkciju, Streamlit nudi cloud hosting servis Streamlit Cloud koji omogućuje besplatno hostanje javnih aplikacija. Alternativno, aplikacija se može deploy-ati na vlastitom serveru koristeći Docker kontejnere ili tradicionalne web servere.

Performanse Streamlit aplikacija ovise o efikasnosti Python koda. Svaki widget interaction pokreće rerenderiranje cijelog script-a što može biti sporo za kompleksne aplikacije. Streamlit pruža cache decorator koji omogućuje cachiranje rezultata skupih operacija. Funkcije označene ovim decorator-om izvršavaju se samo kada se njihovi argumenti promijene.

U našoj aplikaciji, funkcije za dohvaćanje županija i parkova iz baze bile bi idealni kandidati za cachiranje. Međutim, cachiranje je trenutno onemogućeno jer podaci o posjetama

se mogu mijenjati. Za produkcijske aplikacije s većim brojem korisnika, potrebno je pažljivo implementirati cachiranje uz invalidaciju cache-a kada se podaci promijene.

3.3. Folium biblioteka

3.3.1. Osnove Folium-a

Folium je Python biblioteka za kreiranje interaktivnih mapa koristeći Leaflet JavaScript biblioteku. Folium omogućuje kreiranje kompleksnih karti s markerima, poligonima, linijama i drugim geografskim elementima koristeći jednostavan Python API. Biblioteka automatski generira HTML i JavaScript kod potreban za prikaz karte.

Osnovna jedinica je Map objekt koji predstavlja osnovnu kartu. Map prima parametre poput početne pozicije, zoom level-a i tile provider-a. Različiti tile provideri nude različite kartografske stilove od satelitskih snimaka preko topografskih karti do minimalističkih dizajna. U našoj aplikaciji koristimo CartoDB Positron koji ima čistu, svijetlu estetiku prikladnu za prikaz administrativnih granica.

Na mapu se dodaju layeri koristeći različite Folium klase. Marker dodaje točkastu oznaku na određenoj poziciji s opcionalnom ikonom i popup-om. GeoJson layer prikazuje GeoJSON podatke s mogućnošću stiliziranja kroz style function. CircleMarker kreira kružne markere s definiranim radijusom. Svaki layer može imati pridružene event handlere koji reagiraju na klikove ili hover događaje.

3.3.2. Integracija sa Streamlit-om

Integracija Folium-a sa Streamlit-om vrši se kroz streamlit folium package. Ovaj package pruža st folium funkciju koja renderira Folium mapu unutar Streamlit aplikacije. Funkcija prima Folium Map objekt i vraća dictionary s informacijama o interakcijama korisnika poput kliknutih markera ili zoom level-a.

Kreiranje mape u Streamlit aplikaciji počinje konstrukcijom Folium Map objekta. Zatim se dodaju različiti layeri koristeći Folium API. Konačno, st folium funkcija renderira mapu. Svaka interakcija korisnika s kartom pokreće Streamlit rerun koji omogućuje aplikaciji da reagira na korisničke akcije.

U našoj implementaciji, mapa prikazuje tri glavna layer-a. Prvi layer prikazuje granice županija kao poligone. Drugi layer prikazuje parkove također kao poligone. Treći layer dodaje markere na centroidima parkova s bojama i ikonama koje označavaju status posjete. Ova hijerarhijska organizacija omogućuje jasnu vizualnu separaciju različitih geografskih elemenata.

3.4. psycopg2 biblioteka

3.4.1. Povezivanje s bazom

psycopg2 je najpoznatiji PostgreSQL adapter za Python. Biblioteka implementira Python DB-API specifikaciju i pruža dodatne PostgreSQL specifične feature-e. Instalacija se vrši preko pip package manager-a.

Osnovna upotreba počinje kreiranjem konekcije s bazom pomoću connect funkcije. Funkcija prima parametre poput host-a, imena baze, korisničkog imena i lozinke. Konekcija se može koristiti za kreiranje cursor-a koji izvršava SQL upite. Važno je pravilno zatvarati konekcije i cursor-e kako bi se oslobodili resursi.

U našoj aplikaciji, DB CONFIG dictionary definira parametre konekcije. Host je postavljen na localhost što znači da se aplikacija i baza nalaze na istom računalu. Username i password koriste standardne PostgreSQL credentials. Svaka funkcija koja pristupa bazi kreira novu konekciju, izvršava upit, i zatvara konekciju. Za produkcijske aplikacije bilo bi efikasnije koristiti connection pooling.

3.4.2. Izvršavanje upita

Izvršavanje SQL upita vrši se kroz cursor objekte. Cursor execute metoda prima SQL string i opcionalnu listu parametara. Parametri se interpoliraju u upit koristeći siguran mehanizam koji sprječava SQL injection napade. Rezultati upita dohvaćaju se pomoću fetchone, fetchall ili fetchmany metoda.

Za upite koji vraćaju geometrije, psycopg2 vraća geometrije kao bytea stringove. PostGIS funkcija ST AsGeoJSON konvertira geometrije u GeoJSON string koji se zatim parsira u Python dictionary koristeći json loads. Ova konverzija omogućuje jednostavnu integraciju s Folium bibliotekom koja direktno prihvaća GeoJSON strukture.

Transakcijski control vrši se kroz commit i rollback metode na konekciji. U našoj aplikaciji, većina upita su SELECT upiti koji ne modificiraju bazu pa ne zahtijevaju eksplicitan commit. Funkcija za unos posjete poziva commit nakon INSERT upita kako bi promjena bila trajno pohranjena. Error handling implementiran je kroz try except finally blokove koji osiguravaju da se konekcija zatvara čak i kada dođe do greške.

3.5. Arhitektura sustava

Aplikacija je dizajnirana kao troslojna arhitektura koja jasno odvaja prezentacijski sloj, poslovnu logiku i sloj podataka. Prezentacijski sloj implementiran je u Streamlit-u i odgovoran je za prikaz korisničkog sučelja i obradu korisničkog inputa. Ovaj sloj ne sadrži poslovnu logiku već samo kôd za renderiranje widgeta i mapa.

Sloj poslovne logike implementiran je kroz funkcije definirane na početku script-a. Ove

funkcije enkapsuliraju svu logiku za autentifikaciju, dohvaćanje podataka, i procesiranje geografskih informacija. Funkcije primaju parametre, izvršavaju potrebne operacije, i vraćaju rezultate. Ova separacija omogućuje jednostavno testiranje i ponovno korištenje koda.

Sloj podataka je PostgreSQL baza s PostGIS ekstenzijom. Baza pohranjuje sve podatke uključujući korisnike, parkove, županije i posjete. Prostorne tablice sadrže geometrijske stupce koji pohranjuju geografske granice. Sloj podataka također uključuje stored procedure i triggere koji implementiraju kompleksniju logiku direktno u bazi.

Komunikacija između slojeva odvija se kroz dobro definirane interface-e. Prezentacijski sloj poziva funkcije poslovne logike prosljeđujući potrebne parametre. Poslovni sloj komunicira s bazom kroz SQL upite izvršene preko psycopg2-a. Rezultati propagiraju natrag kroz slojeve do prezentacijskog sloja koji ih formatira za prikaz.

4. Implementacija

4.1. Projektiranje baze podataka

4.1.1. Shema baze

Baza podataka sastoji se od četiri glavne tablice koje modeliraju domenu aplikacije. Tablica korisnici pohranjuje informacije o korisnicima sustava. Stupci uključuju id kao primarni ključ, username za jedinstveno korisničko ime, password hash za sigurnu pohranu lozinke, i role koji označava je li korisnik obični korisnik ili administrator.

Tablica Parkovi pohranjuje informacije o nacionalnim parkovima. Stupac id je primarni ključ. Stupac name sadrži naziv parka. Stupac geom tipa geometry sa Polygon constraint-om pohranjuje geografske granice parka. SRID geometrije je četiri dva šest tri što odgovara WGS84 koordinatnom sustavu koji koriste GPS uređaji i web kartografske aplikacije.

Tablica zupanije pohranjuje informacije o hrvatskim županijama. Struktura je slična tablici Parkovi s nazivom u stupcima naziv i geometrijama u stupcu geom. Ova tablica koristi se primarno za vizualizaciju administrativnih granica na karti i ne sudjeluje u poslovnoj logici evidentiranja posjeta.

Tablica posjete povezuje korisnike s parkovima i pohranjuje informacije o posjetama. Stupac id je primarni ključ. Stupac user id je vanjski ključ koji referencira tablicu korisnici. Stupac park id je vanjski ključ koji referencira tablicu Parkovi. Stupac datum posjeta automatski se postavlja na trenutni datum kada se kreira zapis. Stupac ocjena pohranjuje numeričku ocjenu od jedan do pet. Stupac biljeska omogućuje korisnicima unos proizvoljnih tekstualnih bilješki.

4.1.2. Kreiranje prostornih tablica

Kreiranje tablice Parkovi zahtijeva specijalne PostGIS konstrukcije. Prvo se kreira obična tablica s ne-geografskim stupcima. Zatim se dodaje geometrijski stupac koristeći PostGIS funkciju AddGeometryColumn. Ova funkcija prima ime tablice, ime stupca, SRID, geometrijski tip, i dimenzionalnost.

```sql
CREATE TABLE "Parkovi" (
    id SERIAL PRIMARY KEY,
    name VARCHAR(255)
);

SELECT AddGeometryColumn(
    'Parkovi',
    'geom',
    4326,
    'POLYGON',
    2
);
```

Isječak koda 1: Kreiranje tablice Parkovi

Nakon kreiranja tablice, potrebno je kreirati prostorni indeks na geom stupcu. Indeks se kreira koristeći GIST metodu koja je optimizirana za prostorne podatke. Indeks značajno ubrzava prostorne upite poput pronalaženja svih parkova unutar određene udaljenosti od točke ili pronalaženja parkova koji se preklapaju s određenom županijom.

```sql
CREATE INDEX parkovi_geom_idx
ON "Parkovi"
USING GIST (geom);
```

Isječak koda 2: Kreiranje prostornog indeksa

Tablica zupanije kreira se na sličan način. Nakon kreiranja tablica, podaci se mogu importirati iz različitih geografskih formata koristeći GDAL alate ili direktno koristeći PostGIS funkcije za parsiranje GeoJSON, KML ili drugih formata.

4.2. Sustav autentifikacije

4.2.1. Provjera korisničkih podataka

Funkcija provjeri login implementira osnovnu autentifikaciju korisnika. Funkcija prima username i password kao argumente i provjerava postoje li u bazi podataka. Upit SELECT dohvaća id i role korisnika čiji se username i password hash podudaraju s proslijeđenim vrijednostima.

```python
def provjeri_login(username, password):
    conn = psycopg2.connect(**DB_CONFIG)
    cur = conn.cursor()
    query = """
        SELECT id, role
        FROM korisnici
        WHERE username = %s AND password_hash = %s
    """
    cur.execute(query, (username, password))
    user = cur.fetchone()
    conn.close()
    return user
```

Isječak koda 3: Funkcija za provjeru logina

Funkcija koristi parametriziran upit gdje su username i password interpolirani koristeći psycopg2 mehanizam. Ovo je ključno za sigurnost jer sprječava SQL injection napade. Napadač ne može ubaciti zlonamjerni SQL kod kroz username ili password polja jer se svi specijalni znakovi automatski escape-aju.

Rezultat upita je tuple koji sadrži id i role ako je pronađen korisnik, ili None ako korisnik ne postoji ili je lozinka kriva. Pozivajući kod provjerava je li rezultat None i na temelju toga odlučuje dopustiti ili odbiti pristup. Kada je autentifikacija uspješna, id i role se pohranjuju u Streamlit session state za korištenje kroz aplikaciju.

4.2.2. Upravljanje sesijom

Streamlit session state je dictionary struktura koja persistira podatke kroz multiple rerunove aplikacije. Ključevi session state-a mogu se postaviti i dohvaćati kao atributi st session state objekta. U našoj aplikaciji koristimo session state za pohranu informacija o prijavljenom korisniku.

Kada se korisnik uspješno prijavi, postavljamo tri vrijednosti u session state. user id sadrži primarni ključ korisnika iz baze. role sadrži ulogu korisnika bilo admin ili user. username sadrži korisničko ime za prikaz u sučelju. Ove informacije ostaju dostupne dok god je sesija aktivna.

Provjera je li korisnik prijavljen vrši se testiranjem postoji li user id u session state-u. Ako ne postoji, prikazuje se login forma. Ako postoji, prikazuje se glavno sučelje aplikacije s mapom i funkcionalnostima. Logout funkcionalnost implementirana je brisanjem user id iz session state-a i pozivom st rerun koji ponovno učitava aplikaciju.

```python
if 'user_id' not in st.session_state:
    # Prikazi login formu
    st.title("Prijava")
    username = st.text_input("Username")
    password = st.text_input("Password", type="password")

    if st.button("Login"):
        user = provjeri_login(username, password)
        if user:
            st.session_state['user_id'] = user[0]
            st.session_state['role'] = user[1]
            st.session_state['username'] = username
            st.rerun()
        else:
            st.error("Krivi podaci")
else:
    # Prikazi glavno sucelje
    st.title("Karta parkova")
```

Isječak koda 4: Provjera autentifikacije

4.3. Dohvaćanje geografskih podataka

4.3.1. Dohvaćanje županija

Funkcija dohvati zupanije dohvaća geometrije svih županija iz baze i pretvara ih u GeoJSON format. Upit koristi PostGIS funkciju ST AsGeoJSON za konverziju geometrija. Također koristi ST Simplify funkciju za smanjenje broja točaka u geometriji.

```python
def dohvati_zupanije():
    conn = psycopg2.connect(**DB_CONFIG)
    cur = conn.cursor()

    query = """
        SELECT naziv,
               ST_AsGeoJSON(ST_Simplify(geom, 0.001))
        FROM zupanije
    """
    cur.execute(query)
    rows = cur.fetchall()
    conn.close()

    features = []
    for naziv, geom_str in rows:
        features.append({
            "type": "Feature",
            "properties": {"naziv": naziv},
            "geometry": json.loads(geom_str)
        })

    return {
        "type": "FeatureCollection",
        "features": features
    }
```

Isječak koda 5: Dohvaćanje županija

ST Simplify funkcija prima geometriju i tolerance parametar. Tolerance određuje koliko geometrija može odstupati od originala. Vrijednost nula točka nula nula jedan odabrana je eksperimentalno kao dobar balans između vizualne točnosti i performansi. Simplifikacija može smanjiti broj točaka za pedeset posto ili više bez vidljivog gubitka kvalitete na tipičnim zoom levelima.

Rezultat upita procesira se u Python loop-u gdje se svaki redak pretvara u GeoJSON Feature objekt. Naziv županije postavlja se kao property što omogućuje Folium-u da prikaže ime u tooltip-u. Geometrija se parsira iz JSON string-a u Python dictionary. Svi feature-i se skupljaju u listu i vraćaju kao FeatureCollection.

4.3.2. Dohvaćanje parkova za korisnika

Funkcija dohvati parkove za korisnika implementira kompleksniju logiku koja ovisi o ulozi korisnika. Za obične korisnike, funkcija dohvaća sve parkove zajedno s informacijom o tome je li ih korisnik posjetio. Za administratore, funkcija dohvaća ukupan broj posjeta za svaki park.

```python
def dohvati_parkove_za_korisnika(user_id, role):
    conn = psycopg2.connect(**DB_CONFIG)
    conn.set_client_encoding('UTF8')
    cur = conn.cursor()

    if role == 'admin':
        query = """
            SELECT
                p.id,
                COALESCE(p.name, 'Park') as name,
                ST_Y(ST_Centroid(p.geom)) as lat,
                ST_X(ST_Centroid(p.geom)) as lon,
                (SELECT COUNT(*)
                 FROM posjete
                 WHERE park_id = p.id) as ukupno_posjeta
            FROM "Parkovi" p
        """
        cur.execute(query)
    else:
        query = """
            SELECT
                p.id,
                COALESCE(p.name, 'Park') as name,
                ST_Y(ST_Centroid(p.geom)) as lat,
                ST_X(ST_Centroid(p.geom)) as lon,
                v.datum_posjeta
            FROM "Parkovi" p
            LEFT JOIN posjete v
                ON p.id = v.park_id AND v.user_id = %s
            ORDER BY p.name
        """
        cur.execute(query, (user_id,))

    podaci = cur.fetchall()
    conn.close()
    return podaci
```

Isječak koda 6: Dohvaćanje parkova s posjetama

Za administratore, upit koristi skalarni podupit koji broji sve posjete za svaki park. COUNT agregacija vraća broj zapisa u tablici posjete čiji park id odgovara trenutnom parku. Ova vrijednost vraća se kao peti stupac rezultata.

Za obične korisnike, upit koristi LEFT JOIN kako bi uključio sve parkove čak i one koje korisnik nije posjetio. Join uvjet uključuje i park id i user id kako bi dohvatio samo posjete specifičnog korisnika. Ako korisnik nije posjetio park, datum posjeta će biti NULL.

ST Centroid funkcija izračunava geometrijski centar poligona parka. ST X i ST Y funkcije ekstraktiraju x i y koordinate centroida. Ove koordinate koriste se za pozicioniranje markera na karti. COALESCE funkcija osigurava da će se prikazati tekst Park ako je name stupac NULL.

4.4. Kreiranje kartografskog sučelja

4.4.1. Inicijalizacija mape

Kreiranje Folium mape počinje konstrukcijom Map objekta. Konstruktor prima parametre koji definiraju početni prikaz karte. Location parametar je lista s geografskom širinom i dužinom koje definiraju centar karte. Za Hrvatsku, pozicija četrdeset četiri točka pet, šesnaest odgovara otprilike geografskom centru zemlje.

```
m = folium.Map(
    location=[44.5, 16.0],
    zoom_start=7,
    tiles="CartoDB positron"
)
```

Isječak koda 7: Inicijalizacija Folium mape

Zoom start parametar postavlja početni zoom level. Level sedam pruža dobar pregled cijele Hrvatske uz dovoljno detalja za raspoznavanje županija. Tiles parametar određuje koji tile provider će se koristiti. CartoDB Positron odabran je jer ima svijetlu, minimalističku estetiku koja ne ometa prikazivanje granica županija i parkova.

Folium podržava različite tile providere uključujući OpenStreetMap, Stamen Terrain, Stamen Toner i druge. Svaki provider nudi drugačiji kartografski stil optimiziran za različite svrhe. CartoDB Positron posebno dobro funkcionira za aplikacije koje prikazuju administrativne granice jer koristi neutralne boje koje omogućuju jasno isticanje podataka.

4.4.2. Dodavanje GeoJSON layera

GeoJSON layer-i dodaju se na mapu koristeći folium GeoJson klasu. Konstruktor prima GeoJSON strukturu i opcije za stiliziranje. Style function je callback funkcija koja prima feature i vraća dictionary sa CSS stilovima.

```
zupanije_geojson = dohvati_zupanije()

folium.GeoJson(
    zupanije_geojson,
    name="Zupanije",
    style_function=lambda x: {
        'fillColor': 'white',
        'color': "#1001DF",
        'weight': 2,
        'fillOpacity': 0.4
    },
    tooltip=folium.GeoJsonTooltip(
        fields=['naziv'],
        labels=False
    )
).add_to(m)
```

Isječak koda 8: Dodavanje županija na mapu

Style function u ovom primjeru postavlja bijelu boju ispune, tamno plavu boju granice, debljinu linije od dva piksela, i četrdeset posto opacity što omogućuje vidljivost tile-ova ispod. Lambda funkcija prima feature kao parametar ali ga ne koristi jer su svi poligoni istog stila. Za dinamičko stiliziranje, funkcija može pristupiti feature properties i vratiti različite stilove ovisno o atributima.

Tooltip se dodaje koristeći GeoJsonTooltip klasu. Fields parametar specificira koja svojstva feature-a treba prikazati. Labels parametar postavljen na False znači da se neće prikazivati labele samo vrijednosti. Kada korisnik pređe mišem preko županije, prikazat će se ime županije u malom popup-u pored kursora.

4.4.3. Dodavanje markera

Markeri se dodaju individualno za svaki park koristeći podatke dohvaćene iz baze. Boja i ikona markera ovise o statusu posjete i ulozi korisnika. Za svaki park kreira se Folium Marker objekt koji se dodaje na mapu.

```
parkovi = dohvati_parkove_za_korisnika(
    st.session_state['user_id'],
    st.session_state['role']
)

for park in parkovi:
    p_id, p_ime, lat, lon, info_vrijednost = park

    if st.session_state['role'] == 'admin':
        broj_posjeta = info_vrijednost
        if broj_posjeta > 0:
            boja_marker = "blue"
            ikona_tip = "users"
            status_txt = f"Ukupno posjeta: {broj_posjeta}"
        else:
            boja_marker = "red"
            ikona_tip = "times"
            status_txt = "Nitko jos nije posjetio"
    else:
        datum = info_vrijednost
        if datum:
            boja_marker = "green"
            ikona_tip = "check"
            status_txt = f"Posjetili ste: {datum}"
        else:
            boja_marker = "red"
            ikona_tip = "times"
            status_txt = "Niste jos posjetili"

    folium.Marker(
        [lat, lon],
        popup=folium.Popup(
            f"<h5>{p_ime}</h5>{status_txt}",
            max_width=300
        ),
        icon=folium.Icon(
            color=boja_marker,
            icon=ikona_tip,
            prefix='fa'
        ),
        tooltip=p_ime
    ).add_to(m)
```

Isječak koda 9: Dodavanje markera za parkove

Logika za određivanje boje i ikone razlikuje se za administratore i obične korisnike. Administratori vide plavi marker za parkove s posjetama i crveni za neposjećene. Ikona users iz Font Awesome seta prikazuje grupu ljudi što vizualno komunicira da se radi o statitici svih korisnika.

Obični korisnici vide zeleni marker s kvačicom za parkove koje su posjetili i crveni marker s X znakom za neposjećene. Ova jasna vizualna razlika omogućuje brzo skaniranje karte i

identifikaciju parkova koje još nije posjetio.

Popup se kreira koristeći HTML markup. H5 tag prikazuje ime parka kao naslov, a status tekst pruža dodatne informacije. Max width parametar ograničava širinu popup-a na tristotinije piksela što osigurava da se popup pravilno prikazuje na svim veličinama ekrana. Tooltip prikazuje samo ime parka pri hover-u.

4.5. Unos posjeta

4.5.1. Filtriranje neposjećenih parkova

Lista parkova koje korisnik nije posjetio kreira se tijekom iteriranja kroz rezultate upita. Za svakog parka koji nema datum posjete, dodaje se tuple s imenom i id-om u listu ne posjeceni parkovi. Ova lista zatim se koristi za populiranje selectbox widget-a.

```
1 ne_posjeceni_parkovi = []

2 for park in parkovi:
3     p_id, p_ime, lat, lon, datum = park

4     if st.session_state['role'] != 'admin':
5         if datum is None:
6             ne_posjeceni_parkovi.append((p_ime, p_id))
```

Isječak koda 10: Kreiranje liste neposjećenih parkova

Format func parametar selectbox widget-a specificira kako se elementi prikazuju korisniku. Lambda funkcija prima tuple i vraća prvi element što je ime parka. Ovo omogućuje prikazivanje čitljivih imena dok se interno čuva tuple s id-om koji je potreban za insert upit.

4.5.2. Spremanje posjete u bazu

Funkcija upisi posjetu kreira novi zapis u tablici posjete. Funkcija prima četiri parametra: user id, park id, ocjenu i bilješku. INSERT upit dodaje novi redak s ovim vrijednostima plus automatski generirani id i trenutni timestamp.

```python
def upisi_posjetu(user_id, park_id, ocjena, biljeska):
    conn = psycopg2.connect(**DB_CONFIG)
    cur = conn.cursor()
    try:
        query = """
            INSERT INTO posjete
                (user_id, park_id, ocjena, biljeska)
            VALUES (%s, %s, %s, %s)
        """
        cur.execute(query, (user_id, park_id, ocjena, biljeska))
        conn.commit()
        st.success("Posjeta zabilježena!")
    except psycopg2.Error:
        st.error("Greška ili već postoji zapis.")
    finally:
        conn.close()
```

Isječak koda 11: Funkcija za unos posjete

Try except finally blok osigurava pravilno upravljanje greškama i resursima. Try blok izvršava upit i poziva commit. Ako upit uspije, prikazuje se success poruka. Catch blok hvata sve psycopg2 greške uključujući unique constraint violations ako korisnik pokušava dodati duplikat posjetu. Finally blok osigurava da se konekcija uvijek zatvori čak i ako dođe do greške.

Nakon uspješnog unosa, aplikacija poziva st rerun što osvježava cijeli interface. Ovo povlači nove podatke iz baze i ažurira mapu s novim markerom. Novi park više neće biti u listi neposjećenih parkova i marker će promijeniti boju iz crvene u zelenu.

4.6. Administratorski prikaz

4.6.1. Dohvaćanje statistike

Funkcija dohvati admin statistiku agregira posjete po parkovima i vraća broj posjeta za svaki park. Upit koristi LEFT JOIN kako bi uključio i parkove bez posjeta. GROUP BY agregira posjete po imenu parka, a COUNT broji zapise.

```python
def dohvati_admin_statistiku():
    conn = psycopg2.connect(**DB_CONFIG)
    cur = conn.cursor()
    query = """
        SELECT
            COALESCE(p.name, 'Nepoznati park') as naziv,
            COUNT(v.id) as broj_posjeta
        FROM "Parkovi" p
        LEFT JOIN posjete v ON p.id = v.park_id
        GROUP BY p.name
        ORDER BY broj_posjeta DESC
    """
    cur.execute(query)
    podaci = cur.fetchall()
    conn.close()
    return podaci
```

Isječak koda 12: Dohvaćanje administratorske statistike

LEFT JOIN osigurava da se svi parkovi uključe u rezultat čak i oni bez posjeta. Za parkove bez posjeta, COUNT vraća nula jer nema povezanih zapisa u tablici posjete. ORDER BY DESC sortira rezultate tako da se najposjećeniji parkovi prikazuju prvi.

COALESCE funkcija osigurava da se prikaže Nepoznati park ako je name stupac NULL. Ovo je defensive programming praksa koja osigurava da rezultati upita uvijek imaju smislene vrijednosti čak i ako podaci u bazi nisu potpuni.

4.6.2. Prikazivanje stupčastog grafikona

Rezultati statistike prikazuju se koristeći Streamlit bar chart widget. Widget prima dictionary gdje ključevi predstavljaju nazive osi. Data transformira se iz liste tuple-ova u dictionary s listama vrijednosti.

```python
if st.session_state['role'] == 'admin':
    st.subheader("Admin Statistika")
    statistika = dohvati_admin_statistiku()
    chart_data = {
        "Park": [r[0] for r in statistika],
        "Posjeta": [r[1] for r in statistika]
    }
    st.bar_chart(data=chart_data, x="Park", y="Posjeta")
```

Isječak koda 13: Prikazivanje statistike grafom

List comprehensions ekstraktiraju imena parkova i brojeve posjeta iz tuple-ova. X parametar specificira koja vrijednost ide na horizontalnu os, a y parametar specificira vertikalnu os. Streamlit automatski kreira interaktivan grafikon s tooltip-ovima i zoom mogućnostima.

Grafikon pruža administratorima brz vizualni pregled popularnosti različitih parkova.

Najposjećeniji parkovi lako se identificiraju po visokim stupcima. Ova informacija može biti korisna za planiranje resursa, marketing kampanja ili analize trendova.

5. Analiza i evaluacija

5.1. Performanse prostornih upita

5.1.1. Mjerenje vremena izvršavanja

Performanse prostornih upita kritične su za korisničko iskustvo web aplikacija. Dugotrajni upiti rezultiraju sporim učitavanjem stranice što frustrira korisnike. Mjerenja su vršena na development računalu s Intel i5 procesorom i šesnaest gigabajta RAM-a. PostgreSQL baza je konfigurana s default postavkama.

Osnovni upit za dohvaćanje županija bez simplifikacije traje prosječno dva stotine pedeset milisekundi. Ovaj upit dohvaća četrnaest županija gdje svaka geometrija sadrži tisuće točaka. Većina vremena odlazi na konverziju geometrija u GeoJSON format. Mrežni prijenos GeoJSON podataka dodaje dodatnih sto milisekundi.

Dodavanje ST Simplify funkcije s tolerance nula točka nula nula jedan smanjuje vrijeme na sto trideset milisekundi. Simplifikacija smanjuje broj točaka za otprilike pedeset posto što proporcionalno smanjuje veličinu podataka i vrijeme konverzije. Vizualna razlika na zoom levelu sedam je neprimjetna, ali je performansno poboljšanje značajno.

Upit za dohvaćanje parkova s posjetama traje između pedeset i sto milisekundi ovisno o broju korisnikovih posjeta. JOIN operacija je relativno brza jer tablica posjete ima manje od stotinu zapisa. Izračun centroida za svaki park dodaje dodatnih deset do dvadeset milisekundi.

5.1.2. Utjecaj prostornih indeksa

Testiranja su pokazala drastičnu razliku u performansama s i bez prostornih indeksa. Prostorni upit koji pronalazi sve parkove unutar određene županije izvršava se u pet milisekundi s indeksom. Bez indeksa, isti upit traje dvije stotine milisekundi.

Indeks omogućuje PostgreSQL-u da brzo odbaci većinu geometrija koje ne mogu zadovoljavati prostorni uvjet. Umjesto testiranja svakog parka, provjeravaju se samo oni čiji bounding box se preklapa s bounding boxom županije. Za tabicu s pedeset parkova, indeks smanjuje broj detaljnih geometrijskih testova s pedeset na tri ili četiri.

Veličina indeksa je proporcionalna kompleksnosti geometrija. Za naše tablice, prostorni indeksi zauzimaju otprilike petnaest posto veličine same tablice. Ovo je prihvatljiv overhead s obzirom na performansna poboljšanja koja pruža. Indeksi također zahtijevaju ažuriranje kada se dodaju ili modificiraju geometrije, ali ovaj overhead je minimalan za aplikacije koje se primarno koriste za čitanje.

5.2. Optimizacija mrežnog prijenosa

5.2.1. Kompresija GeoJSON podataka

GeoJSON podatci mogu biti prilično veliki posebno za geometrije s mnogo točaka. Kompletni GeoJSON za sve županije bez simplifikacije je približno sto pedeset kilobajta. Za korisnike s sporim internet konekcijama, ovo može značajno usporiti učitavanje aplikacije.

Simplifikacija geometrija je prva linija obrane. Kako je već spomenuto, ST Simplify sa tolerance nula točka nula nula jedan smanjuje veličinu podataka za polovinu bez primjetne vizualne degradacije. Za veće tolerance poput nula točka nula jedan, smanjenje može biti do sedamdeset posto ali uz vidljiv gubitak detalja.

HTTP kompresija je druga tehnika optimizacije. Većina web servera podržava gzip kompresiju gdje se odgovor komprimira prije slanja klijentu. GeoJSON tekstualni format komprimira se izuzetno dobro - gzip tipično postiže compression ratio između pet do jedan i deset do jedan. Ovo znači da sto pedeset kilobajta nekomprimiranih podataka postaje petnaest do trideset kilobajta komprimiranih.

Kombinacija simplifikacije i HTTP kompresije smanjuje transfer s sto pedeset kilobajta na otprilog deset do petnaest kilobajta. Ovo je više nego deseterostruko poboljšanje koje značajno ubrzava učitavanje aplikacije posebno na mobilnim mrežama.

5.2.2. Cachiranje rezultata

Cachiranje je moćna tehnika za smanjivanje redundantnih upita bazi. Streamlit pruža cache decorator koji automatski cacheira rezultate funkcija. Funkcija se izvršava samo prvi put kada je pozvana s određenim argumentima. Sljedeći pozivi s istim argumentima vraćaju cached rezultat.

U našoj aplikaciji, funkcija dohvati zupanije idealan je kandidat za cachiranje. Geometrije županija se ne mijenjaju često pa nema potrebe dohvaćati ih iz baze svaki put. Dodavanje cache decorator-a funkciji znači da će se upit izvršiti samo jednom po session-u.

Međutim, cachiranje mora biti pažljivo primijenjeno. Funkcija dohvati parkove za korisnika ne može se cachirati jer rezultati ovise o trenutnim posjetama korisnika. Ako bi se ova funkcija cacheirala, nove posjete ne bi bile odmah vidljive jer bi se prikazivali cached rezultati.

Za produkcijske aplikacije s mnogo korisnika, potreban je sofisticiraniji caching mehanizam. Redis ili Memcached mogu se koristiti za distributed caching. Cached podaci mogu imati TTL koji osigurava da se cache invalidira nakon određenog vremena. Query rezultati mogu se cachirati s ključevima koji uključuju user id tako da svaki korisnik ima svoj cache.

5.3. Skalabilnost sustava

5.3.1. Broj korisnika

Trenutna implementacija dizajnirana je za mali do srednji broj concurrent korisnika. Svaka funkcija kreira novu database konekciju što je prihvatljivo za nekoliko korisnika ali postaje bottleneck s desetinama ili stotinama istovremenih korisnika. PostgreSQL ima limit na broj simultanih konekcija definiran max connections parametrom.

Connection pooling je standardno rješenje za ovaj problem. Umjesto da svaka funkcija kreira novu konekciju, sve funkcije dijele pool od pre-kreiranih konekcija. Psycopg2 pool modul pruža jednostavnu implementaciju connection pooling-a. Za web aplikacije, middleware poput PgBouncer može upravljati pooling-om na razini servera.

Streamlit server također ima ograničenja skalabilnosti. Default konfiguracija podržava nekoliko stotina istovremenih korisnika. Za veće opterećenje, potrebno je deploy-ati više Streamlit instanci iza load balancer-a. Svaka instanca servisira podskup korisnika, a load balancer distribuira zahtjeve.

5.3.2. Veličina podataka

Trenutna baza sadrži četrnaest županija i pedeset parkova što je trivijalna količina podataka. Međutim, kako broj parkova raste, potrebno je razmotriti strategije optimizacije. Sa stotinama ili tisućama parkova, dohvaćanje svih odjednom postaje neefikasno.

Spatial clustering je jedna tehnika gdje se geometrije grupiraju u regije. Umjesto dohvaćanja svih parkova, aplikacija dohvaća samo one u trenutno vidljivom području karte. PostGIS funkcija ST MakeEnvelope može kreirati bounding box od trenutnog viewport-a karte, a ST Intersects može filtrirati parkove unutar tog box-a.

Level of detail je druga tehnika gdje se različite verzije geometrija koriste za različite zoom levele. Na niskim zoom levelima gdje je cijela Hrvatska vidljiva, parkovi se prikazuju kao pojednostavljeni poligoni ili čak samo kao točke. Na visokim zoom levelima, prikazuju se detaljne granice. Ovo smanjuje količinu podataka koja mora biti transferirana i renderirana.

Particioniranje tablica može poboljšati performanse upita za vrlo velike tablice. PostgreSQL podržava horizontal particioniranje gdje se redovi podijele u više fizičkih tablica na temelju nekog kriterija. Na primjer, parkovi bi mogli biti particionirani po županijama. Upiti koji filtriraju po županiji mogu pristupiti samo relevantnoj particiji.

5.4. Prednosti i nedostaci pristupa

5.4.1. Prednosti PostGIS rješenja

Korištenje PostGIS-a za prostorne podatke donosi nekoliko značajnih prednosti. Prva prednost je integracija s postojećom relacijskom bazom. Prostorni i ne-prostorni podaci nalaze se u istoj bazi što pojednostavljuje backup, replication, i access control. Transakcijska konzistentnost osigurava da prostorni i atributni podaci ostaju sinkronizirani.

Druga prednost je moć SQL-a za kompleksne upite. Mogućnost kombiniranja prostornih operacija s JOIN-ovima, agregatima i podupitima omogućuje sofisticirane analize koje bi bile teške ili nemoguće s standalone GIS alatima. Na primjer, jednostavno je napisati upit koji pronalazi sve korisnike koji su posjetili parkove unutar određene županije s ocjenom višom od četiri.

Treća prednost je otvoreni standard format. PostGIS implementira OGC Simple Features specifikaciju što osigurava interoperabilnost. Podaci se mogu lako exportirati u različite formate i koristiti s drugim GIS alatima. Funkcije poput ST AsGeoJSON, ST AsKML, i ST AsGML omogućuju konverziju u različite formate.

Četvrta prednost je skalabilnost i performanse. PostgreSQL je dokazano skalabilan za terabajte podataka. Prostorni indeksi omogućuju brze upite čak i nad milijunima geometrija. Podrška za paralelizaciju upita dodatno poboljšava performanse na modernom hardveru.

5.4.2. Nedostaci i ograničenja

Unatoč prednostima, PostGIS pristup ima određene nedostatke. Prvi nedostatak je kompleksnost postavljanja i konfiguracije. Instalacija PostgreSQL-a i PostGIS-a zahtijeva više koraka nego korištenje cloud GIS servisa. Programeri moraju razumjeti geografske koncepte poput koordinatnih sustava i projekcija.

Drugi nedostatak je ograničena vizualizacijska funkcionalnost. PostGIS je izvrsna za pohranu i analizu ali ne pruža map rendering. Potrebne su dodatne biblioteke poput Folium-a ili Leaflet-a za vizualizaciju. Za napredne kartografske potrebe poput stiliziranja na temelju zoom level-a ili interaktivnih editing-a, potrebni su specializirani alati.

Treći nedostatak je potencijalna over-engineering za jednostavne use case-ove. Ako aplikacija samo treba prikazati nekoliko točaka na karti, puni PostGIS stack je možda pretjeran. Cloud servisi poput Google Maps API ili Mapbox mogu biti jednostavniji za takve scenarije.

Četvrti nedostatak je administracijski overhead. PostgreSQL baza zahtijeva redovito održavanje uključujući vacuum, analyze, backup, i monitoring. Za male projekte ili prototype-ove, ovo može biti dodatno opterećenje. Managed database servisi mogu smanjiti ovaj overhead ali uz dodatni trošak.

5.5. Usporedba s alternativama

5.5.1. Cloud GIS servisi

Cloud GIS servisi poput Google Maps Platform, Mapbox, ili HERE nude alternativni pristup gdje se prostorni podaci pohranjuju i procesiraju u cloud-u. Prednost ovih servisa je jednostavnost integracije. Developer može početi koristiti servise s nekoliko linija JavaScript koda bez potrebe za postavljanjem infrastrukture.

Drugi prednost cloud servisa je skalabilnost. Servisi automatski skaliraju s opterećenjem bez potrebe za ručnim upravljanjem serverima ili bazama. Podržavaju milijune zahtjeva dnevno s garantiranom dostupnošću. Treća prednost je bogat feature set koji uključuje geocoding, routing, elevation data, i satelitske snimke.

Nedostatak cloud servisa je trošak. Većina servisa naplaćuje po broju zahtjeva što može postati skupo za aplikacije s velikim prometom. Google Maps cijena može dosegnuti stotine dolara mjesečno za aplikacije s tisućama dnevnih korisnika. Drugi nedostatak je vendor lock-in. Aplikacija postaje ovisna o specifičnom provideru što otežava migraciju.

Za našu aplikaciju, cloud servis bi bio preskup jer zahtijevamo punu kontrolu nad prostornim podacima i analitikama. PostGIS omogućuje pohranu svih podataka lokalno bez ongoing troškova. Međutim, za aplikacije koje prvenstveno prikazuju postojeće tile-ove bez potrebe za prostornim upitima, cloud servisi mogu biti jeftinije i jednostavnije rješenje.

5.5.2. Standalone GIS serveri

GeoServer i MapServer su standalone serveri specijalizirani za servisiranje geografskih podataka. Ovi serveri mogu čitati podatke iz PostGIS-a ili drugih izvora i servisirati ih kroz standardne protokole poput WMS, WFS, i WCS. Prednost ovih servera je advanced rendering mogućnosti uključujući SLD styling, labeling, i multi-layer compositing.

Druga prednost je compliance sa OGC standardima što olakšava integraciju s desktop GIS alatima poput QGIS-a ili ArcGIS-a. Treća prednost je caching mehanizmi poput GeoWebCache koji mogu dramatično ubrzati servisiranje tile-ova. Četvrta prednost je podrška za različite output formate uključujući raster tile-ove, vector tile-ove, i raw data formata.

Nedostatak standalone servera je dodatna kompleksnost infrastrukture. Potrebno je deployment-ati i održavati dodatni server pored baze i web aplikacije. Konfiguracija može biti kompleksna posebno za napredne feature-e. Za naš use case gdje web aplikacija direktno pristupa PostGIS-u, standalone server bi bio over-engineering.

6. Zaključak

6.1. Sažetak rada

Ovaj rad predstavio je razvoj kompletne web aplikacije za geografsku evidenciju posjeta nacionalnih parkova koristeći PostGIS prostornu bazu podataka i moderne Python web tehnologije. Aplikacija uspješno demonstrira integraciju različitih tehnologija uključujući PostgreSQL, PostGIS, Streamlit i Folium u koherentan sustav.

Implementacija obuhvaća sve ključne komponente funkcionalnog GIS sustava. Prostorna baza podataka pohranjuje kompleksne geometrije županija i parkova s efikasnim indeksiranjem. Sustav autentifikacije omogućuje razlučivanje uloga gdje obični korisnici evidentiraju vlastite posjete, a administratori pristupaju agregiranim statistikama. Interaktivna karta pruža intuitivnu vizualizaciju geografskih podataka s markerima koji jasno komuniciraju status posjete.

Prostorni upiti demonstriraju moć PostGIS-a za geografsku analizu. Funkcije poput ST Centroid, ST AsGeoJSON, i ST Simplify omogućuju efikasnu obradu i konverziju geografskih podataka. Optimizacijske tehnike uključujući simplifikaciju geometrija i prostorne indekse osiguravaju prihvatljive performanse.

Evaluacija sustava pokazala je da PostGIS pristup pruža odličan balans između funkcionalnosti, performansi i fleksibilnosti. Vrijeme izvršavanja upita je u milisekundama čak i za kompleksne geometrije. Skalabilnost je zadovoljavajuća za manje do srednje aplikacije, s jasnim putevima za optimizaciju kada je potrebno.

6.2. Evaluacija ciljeva

Pregledom originalnih ciljeva postavljenih na početku rada, možemo zaključiti da su svi uspješno realizirani. Prvi cilj bio je projektirati i implementirati prostornu bazu podataka koja pohranjuje geografske granice županija i parkova. Baza je uspješno implementirana s PostGIS geometrijskih tipovima i prostornim indeksima. Podaci se efikasno pohranjuju i dohvaćaju.

Drugi cilj bio je razviti sustav autentifikacije korisnika s razlučivanjem uloga. Sustav je potpuno funkcionalan s jasnom separacijom između običnih korisnika koji vide vlastite posjete i administratora koji vide agregirane statistike. Session state mehanizam pravilno održava autentifikaciju kroz session.

Treći cilj bio je implementirati interaktivno kartografsko sučelje. Folium mapa uspješno prikazuje tri layer-a podataka - županije, parkove i markere. Interaktivne mogućnosti uključujući zoom, pan i tooltip funkcioniraju glatko. Vizualni indikatori jasno komuniciraju status posjete.

Četvrti cilj bio je optimizirati performanse pri radu s velikim geometrijama. ST Simplify funkcija smanjuje broj točaka bez vidljivog gubitka kvalitete. Prostorni indeksi omogućuju brze upite. Vrijeme učitavanja aplikacije je prihvatljivo čak i za kompleksne geometrije.

Peti cilj bio je analizirati prednosti i nedostatke PostGIS pristupa. Analiza je pokazala

da PostGIS pruža izvrsnu integraciju s relacijskim podacima, moćne prostorne funkcije, i dobre performanse. Nedostaci uključuju kompleksnost postavljanja i administracijski overhead koji mogu biti preveliki za najjednostavnije use case-ove.

6.3. Buduća proširenja

Iako je trenutna implementacija funkcionalna, postoji mnoštvo mogućnosti za buduća proširenja i poboljšanja sustava. Prvo moguće proširenje je implementacija naprednijih prostornih upita i analiza. Moglo bi se dodati funkcionalnosti poput pronalaženja najbližih parkova od korisnikove trenutne lokacije, izračuna udaljenosti između parkova, ili identifikacije parkova unutar određenog radijusa.

Drugo proširenje je bogata korisnička profilna stranica. Korisnici bi mogli vidjeti statistiku svojih posjeta uključujući ukupan broj posjećenih parkova, prosječnu ocjenu, najdraže parkove, ili vizualizaciju posjeta kroz vrijeme. Heatmap bi mogao prikazati koje regije Hrvatske korisnik najviše posjećuje.

Treće proširenje je socijalna komponenta gdje korisnici mogu dijeliti svoje posjete, komentare i fotografije. Mogla bi se implementirati funkcionalnost praćenja drugih korisnika i prikazivanja njihovih aktivnosti. Leaderboard bi prikazivao najaktivnije korisnike ili one s najviše posjeta.

Četvrto proširenje je offline funkcionalnost pomoću Progressive Web App tehnologija. Korisnici bi mogli preuzeti karte za offline korištenje i evidentirati posjete dok nemaju internet konekciju. Posjete bi se automatski sinkronizirale kada se vrati konekcija.

Peto proširenje je integracija s vanjskim API-ima. Mogao bi se dohvatiti meteorološke informacije za svaki park, trenutna zauzetost parkirališta, ili informacije o događanjima. Wikipedia ili Wikidata API-i mogli bi pružiti dodatne informacije o historiji i značajkama parkova.

Šesto proširenje je napredna administratorska kontrolna ploča. Umjesto samo stupčastog grafikona, administratori bi mogli vidjeti vremenske serije posjeta, geografsku distribuciju korisnika, korelacije između ocjena i karakteristika parkova, ili prediktivne analize trendova.

6.4. Zaključna razmatranja

Kroz razvoj ovog projekta, PostGIS se pokazao kao izuzetno moćan alat za implementaciju GIS funkcionalnosti unutar web aplikacija. Integracija prostornih podataka s relacijskom bazom omogućuje kompleksne upite koji bi bili teški ili nemoguće s drugim pristupima. Deklarativna priroda SQL-a čini prostorne upite razumljivim i održivim.

Jedna od ključnih lekcija iz projekta je važnost pažljivog dizajniranje prostornih indeksa. Performanse prostornih upita drastično se razlikuju s i bez indeksa. Simplifikacija geometrija je također kritična za performanse web aplikacija gdje se geometrije moraju transferirati preko mreže.

Streamlit se pokazao kao odličan izbor za brzi razvoj proof-of-concept aplikacija. Mogućnost kreiranja interaktivnog sučelja s minimalnim kodom ubrzava razvoj. Međutim, za produkcijske aplikacije s velikim brojem korisnika, tradicionalniji web framework poput Django-a ili Flask-a bio bi prikladniji.

Kombinacija PostGIS-a i Folium-a pruža moćan stack za web GIS aplikacije. PostGIS rješava backend izazove pohrane i analize, dok Folium pruža frontend vizualizaciju. Jasna separacija između ovih odgovornosti rezultira čistom arhitekturom koja je laka za razumijevanje i održavanje.

Projekt također ilustrira važnost razumijevanja geografskih koncepata poput koordinatnih sustava, projekcija, i geometrijskih tipova. Developer mora biti upoznat s ovim konceptima kako bi efikasno koristio GIS alate. Međutim, sa solidnim razumijevanjem osnova, PostGIS pruža dostupan entry point u svijet GIS-a.

Na kraju, ovaj rad uspješno demonstrira da je moguće razviti funkcionalne GIS aplikacije koristeći otvorene tehnologije bez potrebe za skupim komercijalnim softverom. PostGIS, zajedno s Python ekosistemom, pruža sve što je potrebno za većinu GIS use case-ova. Za organizacije s ograničenim budžetima ili programere koji žele naučiti GIS, ovo je izvrstan početak.

Popis literature

[1] N. Schuurman, *Geographic Information Systems (GIS): A Short Introduction*. Wiley-Blackwell, 2004.

[2] R. O. Obe i L. S. Hsu, *PostGIS in Action*, 3rd. Manning Publications, 2021.

[3] H. Butler, M. Daly, A. Doyle, S. Gillies, S. Hagen i T. Schaub, „The GeoJSON Format," Internet Engineering Task Force (IETF), RFC 7946, 2016.

[4] I. Muehlenhaus, *Web Cartography: Map Design for Interactive and Mobile Devices*. CRC Press, 2014.

[5] P. Corti, S. V. Mather, T. J. Kraft i B. Park, *PostGIS Cookbook*, 2nd. Packt Publishing, 2018.

Popis isječaka koda