TripPy

Arhitekturni projekat

Verzija 1.0

Pregled izmena

| **Datum** | **Verzija** | **Opis** | **Autor** |
| --- | --- | --- | --- |
| 18.04.2024. | 1.0 | Inicijalna verzija | Marko, Bogdan, Milan |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Sadržaj

[**1. Cilj dokumenta 5**](#_gjdgxs)

[2. Opseg dokumenta 5](#)

[3. Reference 5](#)

[**4. Predstavljanje arhitekture 5**](#_3znysh7)

[**5. Ciljevi i ograničenja arhitekture 5**](#_tyjcwt)

[**6. Pogled na slučajeve korišćenja 5**](#_3dy6vkm)

[6.1 Dijagrami slučajeva korišćenja 6](#_1t3h5sf)

[6.2 Kratak opis slučajeva korišćenja 7](#_4d34og8)

[6.2.1 Prijavljivanje 7](#)

[6.2.2 Pregled oglasa po imenu 7](#_edjqo5xqmrlo)

[6.2.3 Pregled spiska oglasa 7](#_gnq2eloc62vn)

[6.2.4 Sortiranje oglasa po imenu 7](#)

[6.2.5 Pregled određenog oglasa 7](#_26in1rg)

[6.2.6 Dodavanje novog oglasa 7](#)

[6.2.7 Ažuriranje podataka o oglasu 8](#)

[6.2.8 Brisanje postojećeg oglasa 8](#)

[6.2.9 Pregled maršruta 8](#)

[6.2.10 Sastavljanje nove maršrute 8](#)

[6.2.11 Ažuriranje sastavljene maršrute 8](#)

[6.2.12 Brisanje postojeće maršrute 8](#)

[6.2.13 Prihvatanje/odbijanje oglasa 8](#)

[6.2.14 Kreiranje novog korisničkog naloga 8](#)

[6.2.15 Brisanje postojećeg korisničkog naloga 8](#)

[6.2.16 Ažuriranje podataka postojećeg korisnika 8](#)

[6.2.17 Pregled spiska svih korisničkih naloga 8](#)

[**7. Pogled na logičku arhitekturu sistema 8**](#_35nkun2)

[7.1 Pregled arhitekture – organizacija paketa i podsistema u slojeve 9](#_1ksv4uv)

[7.1.1 Korisnički interfejs 9](#_44sinio)

[7.1.2 Aplikaciona logika 9](#_2jxsxqh)

[7.1.3 Pristup podacima 9](#_z337ya)

[7.1.4 HTML 10](#_3j2qqm3)

[7.1.5 JavaScript 10](#)

[7.1.6 Python (Flask) 10](#)

[7.1.7 PostgreSQL 10](#_1y810tw)

[**8. Pogled na procese 10**](#_4i7ojhp)

[8.1 Procesi 10](#_2xcytpi)

[8.1.1 Web čitač 10](#_1ci93xb)

[8.1.2 Web server 10](#_3whwml4)

[8.1.3 Server (Flask) 10](#)

[8.1.4 PostgreSQL Server 11](#_2bn6wsx)

[**9. Pogled na raspoređivanje sistema 11**](#_qsh70q)

[9.1 Klijent 11](#_3as4poj)

[9.2 Web server 11](#_1pxezwc)

[9.3 DBMS server 11](#_49x2ik5)

[**10. Pogled na implementaciju sistema 11**](#_2p2csry)

[10.1 Model domena 11](#_147n2zr)

[10.2 Šema baze podataka 13](#)

[10.3 Komponente sistema 13](#_23ckvvd)

[10.3.1 Komponente korisničkog interfejsa 13](#_ihv636)

[10.3.2 Komponente aplikacione logike 14](#_32hioqz)

[10.3.3 Komponente za pristup podacima 15](#_1hmsyys)

[**11. Performanse 16**](#_41mghml)

[**12. Kvalitet 16**](#_2grqrue)

Arhitekturni projekat

# Cilj dokumenta

Cilj ovog dokumenta je specifikacija zahteva u pogledu detaljnog opisa slučajeva korišćenja TripPy sistema.

# Opseg dokumenta

Dokument se odnosi na TripPy aplikaciju koja će biti razvijena od strane eBora-e. Namena sistema je efikasno i uspešno planiranje i organizovanje putovanja.

# Reference

Spisak korišćene literature:

1. TripPy – Predlog projekta, eBora-TripPy-01, V1.0, 2024, eBora.
2. TripPy – Planirani raspored aktivnosti na projektu, V1.0, 2024, eBora.
3. TripPy – Plan realizacije projekta, V1.0, 2024, eBora.
4. TripPy – Vizija sistema, V1.0, 2024, eBora.
5. TripPy – Specifikacija zahteva, V1.0, 2024, eBora.

# Predstavljanje arhitekture

Arhitektura sistema u dokumentu je prikazana kao serija pogleda na sistem: pogled na slučajeve korišćenja, pogled na logičku arhitekturu sistema, pogled na procese, pogled na razmeštaj komponenti sistema i pogled na implementaciju. Ovi pogledi su predstavljeni odgovarajućim UML dijagramima.

# Ciljevi i ograničenja arhitekture

Ključni zahtevi i sistemska ograničenja koja imaju značajan uticaj na izbor arhitekture i projektovanje sistema su:

1. TripPy aplikacija će biti implementirana kao Web aplikacija zasnovana na JavaScript skripting jeziku, Python jeziku i PostgreSQL bazi podataka [4].
2. Klijentski deo TripPy aplikacije će biti optimizovan za sledeće Web čitače: Microsoft Edge, Opera, Firefox (Mozilla) kao i Google Chrome. [4].
3. Svi zahtevi u pogledu performansi dati u [5] moraju biti uzeti u obzir pri izboru arhitekture i razvoju sistema.

# Pogled na slučajeve korišćenja

U ovom odeljku je dat pogled na slučajeve korišćenja definisane u specifikaciji zahteva [5].

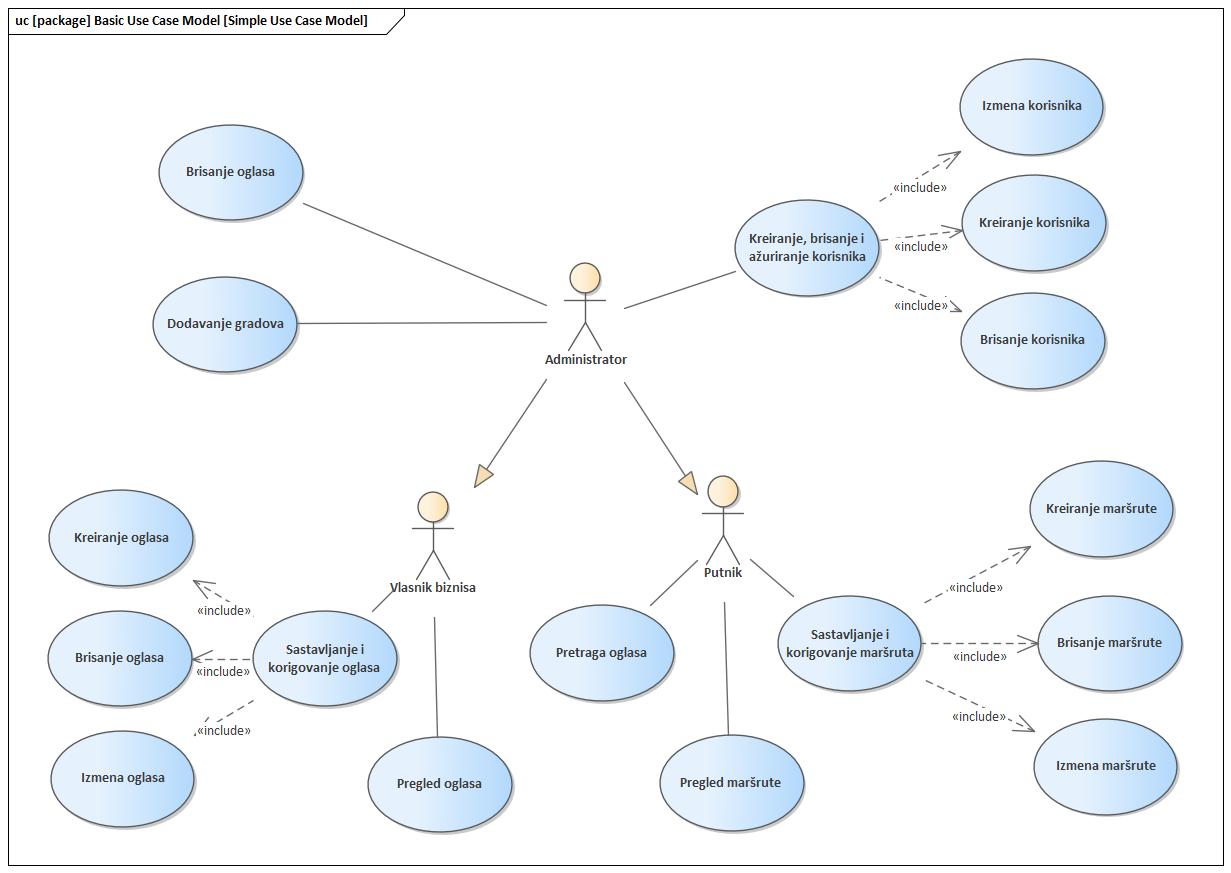
Slučajevi korišćenja TripPy aplikacije su:

* Prijavljivanje
* Pregled spiska oglasa
* Sortiranje oglasa po imenu
* Pregled odredjenog oglasa
* Pretraga oglasa po imenu
* Dodavanje novog oglasa
* Ažuriranje podataka o oglasu
* Brisanje oglasa
* Prihvatanje/odbijanje oglasa
* Pregled maršrute
* Sastavljanje nove maršrute
* Korigovanje sastavljene maršrute
* Brisanje postojeće maršrute
* Kreiranje novog korisničkog naloga
* Brisanje postojećeg korisničkog naloga
* Ažuriranje podataka postojećeg korisnika
* Pregled spiska svih korisnika

Ove slučajeve korišćenja mogu da iniciraju putnik, vlasnik biznisa, administrator.

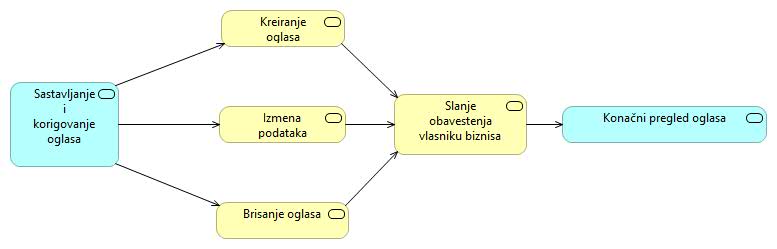
## Dijagrami slučajeva korišćenja

Osnovni UML dijagram koji prikazuje korisnike i slučajeve korišćenja TripPy aplikacije prikazan je na sledećoj slici:

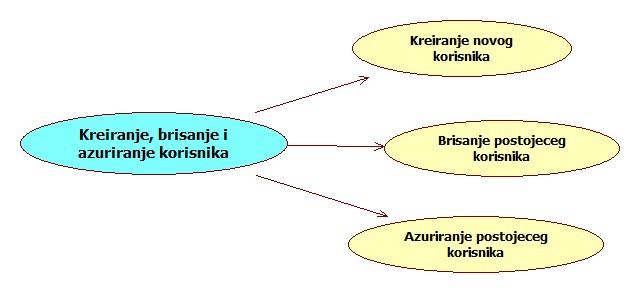


Slučajevi korišćenja *sastavljanje i korigovanje oglasa*, *kreiranje, brisanje i ažuriranje korisnika, sastavljanje i korigovanje maršrute* obuhvataju složenije radnje koje se mogu dalje razložiti na pojedinačne slučajeve korišćenja.

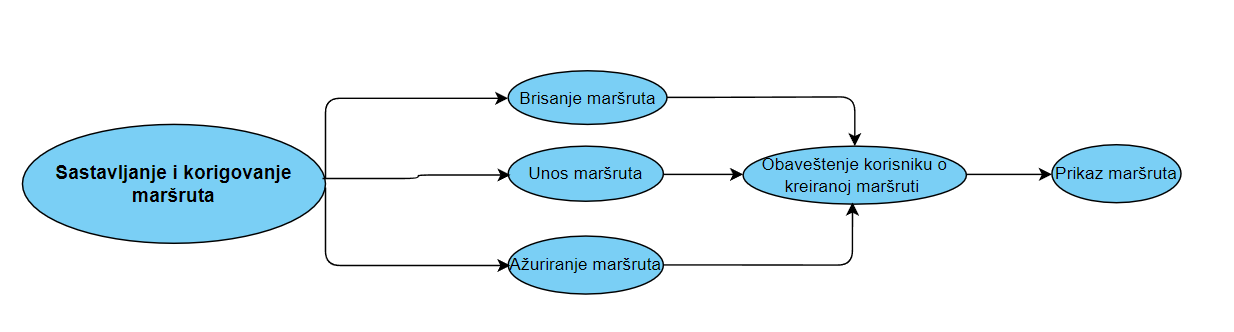
Detaljni UML dijagram za slučaj korišćenja *sastavljanje i korigovanje oglasa* je prikazan na sledećoj slici:



Detaljni UML dijagram za slučaj korišćenja *kreiranje, brisanje i ažuriranje korisnika* je prikazan na sledećoj slici:



Detaljni UML dijagram za slučaj korišćenja *sastavljanje i korigovanje maršruta* je prikazan na sledećoj slici:

****

## Kratak opis slučajeva korišćenja

### Prijavljivanje

Kratak opis: Prijavljivanje zaposlenih u cilju pristupa funkcionalnostima aplikacije.

Akteri koji iniciraju slučaj korišćenja: Putnik, Vlasnik biznisa, Administrator.

### Pregled oglasa po imenu

Kratak opis: Prikaz stranice sa oglasima čije ime sadrži uneti tekst.

Akteri koji iniciraju slučaj korišćenja: Putnik, Vlasnik biznisa, Administrator.

### Pregled spiska oglasa

Kratak opis: Prikaz stranice sa oglasima.

Akteri koji iniciraju slučaj korišćenja: Putnik, Vlasnik biznisa, Administrator.

### Sortiranje oglasa po imenu

Kratak opis: Prikaz stranice sa oglasa sortiranim po imenu.

Akteri koji iniciraju slučaj korišćenja: Putnik, Vlasnik biznisa, Administrator.

### Pregled određenog oglasa

Kratak opis: Prikaz stranice sa informacijama o određenom oglasu.

Akteri koji iniciraju slučaj korišćenja: Putnik, Vlasnik biznisa, Administrator.

### Dodavanje novog oglasa

Kratak opis: Dodavanje novog oglasa u bazu oglasa od strane vlasnika biznisa.

Akteri koji iniciraju slučaj korišćenja: Vlasnik biznisa.

### Ažuriranje podataka o oglasu

Kratak opis: Izmena podataka oglasa.

Akteri koji iniciraju slučaj korišćenja: Vlasnik biznisa.

### Brisanje postojećeg oglasa

Kratak opis: Brisanje postojećeg oglasa iz baze oglasa.

Akteri koji iniciraju slučaj korišćenja: Vlasnik biznisa, Administrator.

### Pregled maršruta

Kratak opis: Prikaz stranice sa maršrutama.

Akteri koji iniciraju slučaj korišćenja: Putnik, Vlasnik biznisa, Administrator.

### Sastavljanje nove maršrute

Kratak opis: Sastavljanje nove maršrute od strane putnika.

Akteri koji iniciraju slučaj korišćenja: Putnik.

### Ažuriranje sastavljene maršrute

Kratak opis: Izmena podataka maršrute.

Akteri koji iniciraju slučaj korišćenja: Putnik.

### Brisanje postojeće maršrute

Kratak opis: Brisanje postojeće maršrute iz baze podataka.

Akteri koji iniciraju slučaj korišćenja: Putnik, Administrator.

### Prihvatanje/odbijanje oglasa

Kratak opis: Prihvatanje/odbijanje sastavljenih oglasa od strane Administratora.

Akteri koji iniciraju slučaj korišćenja: Administrator.

### Kreiranje novog korisničkog naloga

Kratak opis: Kreiranje korisničkog naloga za novog zaposlenog.

Akteri koji iniciraju slučaj korišćenja: Putnik, Vlasnik biznisa.

### Brisanje postojećeg korisničkog naloga

Kratak opis: Brisanje korisničkog naloga i podataka za postojećeg korisnika.

Akteri koji iniciraju slučaj korišćenja: Administrator.

### Ažuriranje podataka postojećeg korisnika

Kratak opis: Ažuriranje osnovnih podataka postojećeg korisničkog naloga.

Akteri koji iniciraju slučaj korišćenja: Administrator, putnik, vlasnik biznisa.

### Pregled spiska svih korisničkih naloga

Kratak opis: Prikaz stranice sa listom svih putnika i vlasnika biznisa.

Akteri koji iniciraju slučaj korišćenja: Administrator.

# Pogled na logičku arhitekturu sistema

U ovom odeljku je dat pregled logičke arhitekture sistema. Ovaj pogled sadrži opis najznačajnijih klasa, njihove organizacije u pakete i podsisteme, i organizacija podsistema u slojeve. U cilju opisivanja dinamičkih aspekata arhitekture, ovaj odeljak može da uključi opise realizacije najznačajnijih slučajeva korišćenja. Da bi se ilustrovala veza između arhitekturno značajnih klasa, podsistema, paketa ili slojeva moguće je uključiti i odgovarajuće dijagrame klasa.

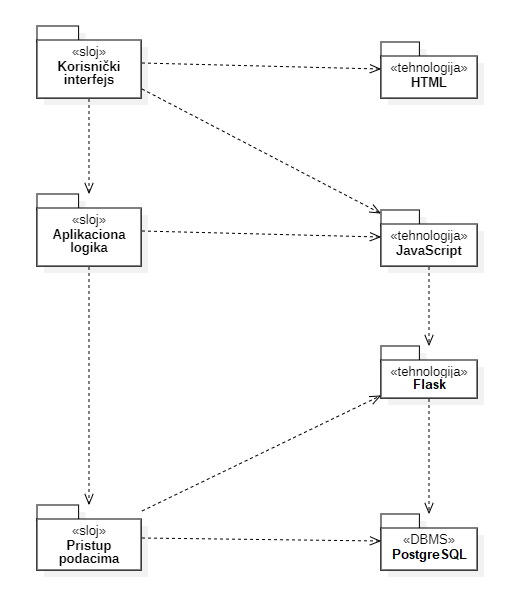
Logički pogled na TripPy sistem obuhvata 3 glavna paketa: Korisnički interfejs, Aplikaciona logika, Pristup podacima.

Paket *Korisnički interfejs* sadrži Web stranice, skripte(Javascript) i multimedijalni sadržaj koji realizuju grafički dizajn i forme preko kojih korisnici sistema komuniciraju sa sistemom.

Paket *Aplikaciona logika* predstavlja serverski deo sistema koji je zadužen za realizaciju funkcionalnosti specifičnih za domen sistema koji se razvija.

Paket *Pristup podacima* sadrži funkcije koje predstavljaju interfejs za pristup, dodavanje i ažuriranje podataka koji se čuvaju u bazi podataka.

## Pregled arhitekture – organizacija paketa i podsistema u slojeve



### Korisnički interfejs

Ovaj sloj realizuje korisnički interfejs aplikacije. U njemu su sadržane sve HTML, multimedijalni sadržaji i JS skripte koje generišu HTML stranice preko kojih korisnici komuniciraju sa sistemom.

Sloj korisničkog interfejsa zavisi od sloja aplikacione logike, kao i paketa HTML i JS.

### Aplikaciona logika

Sloj aplikacione logike predstavlja serverski deo sistema. Sadrži Python kod koji realizuje funkcionalnost karakterističnu za domen primene sistema i uspostavljaju vezu između korisničkog interfejsa i sloja za pristup podacima.

Ovaj sloj zavisi od sloja za pristup podacima i Python serverskog dela.

### Pristup podacima

Sloj za pristup podacima se nalazi na dnu troslojne arhitekture i sadrži Python kod zadužen za pribavljanje, dodavanje i ažuriranje podataka koji se čuvaju u PostgreSQL bazi podataka.

Ovaj sloj ne zavisi od drugih slojeva, ali je zavisan od Python-a i PostgreSQL baza podataka.

### HTML tehnologija

Tehnologija HTML-a definiše gradivne elemente stranica koje se prikazuju u Web čitaču i koje omogućavaju prikaz formatiranih informacija i realizaciju formi za unos i ažuriranje podataka.

### JavaScript tehnologija

Tehnologija JS obezbeđuje mehanizam za pisanje i izvršavanje skripti. Ove skripte mogu da generišu HTML kod koji realizuje korisnički interfejs i služe za komunikaciju sa serverskim delom aplikacije.

### Python (Flask) tehnologija

Tehnologija Flask obezbeđuje mehanizam za pisanje i izvršavanje serverskog dela koda. Ovaj kod odgovara na zahteve klijenata i pristupa bazi podataka u cilju pribavljanja, unosa i ažuriranja podataka.

### PostgreSQL - DBMS

PostgreSQL predstavlja sistem za upravljanje bazama podataka koji će se koristiti za realizaciju TripPy sistema.

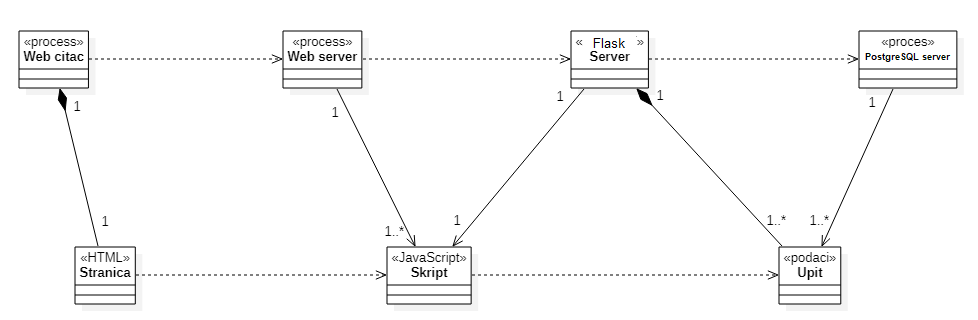
# Pogled na procese

U ovom odeljku je sadržan pogled na procesnu arhitekturu sistema. Ovaj opis treba da sadrži specifikaciju različitih zadataka (procesa i niti) uključenih u rad sistema. Takođe je potrebno dati dijagrame koji pokazuju njihovu interakciju i konfiguraciju. Dodela objekata i klasa na određene zadatke takođe spada u opis procesne arhitekture.

Ilustracije radi u nastavku je dat opis procesa uključenih u izvršenje TripPy sistema kao Web aplikacije.

## Procesi

Na sledećem UML dijagramu klasa prikazani su procesi koji učestvuju u izvršenju TripPy sistema. Dijagram je opšteg tipa i može se primeniti na bilo koju Web aplikaciju zasnovanu na klijent-server (JS – Python) arhitekturi i PostgreSQL bazi podataka.



### Web čitač

Web čitač je proces koji izvršava funkcionalnost aplikacije za prikaz HTML stranica dobijenih od nekog Web servera. U najopštijem slučaju Web čitač u jednom trenutku može da prikazuje samo jednu HTML stranicu.

Web čitač zavisi od Web servera koji generiše i vraća odgovarajuću HTML stranicu na zahtev.

### Web server

Web server je proces koji izvršava funkcionalnost opsluživanja zahteva prispelih sa više Web čitača i predstavlja interfejs između čitača i serverskog dela aplikacije.

### Server (Flask)

Server proces obavlja posao obrade zadatog zadatka i generiše odgovarajući sadržaj koji Web server šalje Web čitaču. Ovaj proces može da zahteva usluge PostgreSQL servera-a. Komunikacija između serverskog procesa i PostgreSQL servera se obavlja preko prosleđivanja upita i vraćanja rezultat.

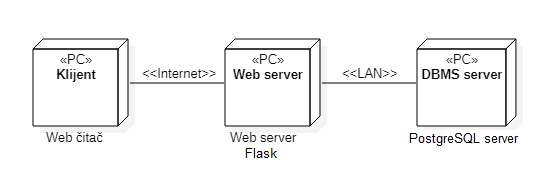
### PostgreSQL Server

PostgreSQL Server je proces koji izvršava funkcionalnost PostgreSQL sistema za upravljanje bazama podataka. Ovaj proces može konkurentno da prihvati određen broj upita, izvrši ih nad bazom podataka i vrati rezultate procesu koji je upite postavio.

# Pogled na raspoređivanje sistema

Pogled na raspoređivanje sistema prikazuje različite fizičke čvorove za najopštiju konfiguraciju sistema. Fizičkim čvorovima koji predstavljaju procesore vrši se dodeljivanje identifikovanih procesa.

Na sledećoj slici dat je UML dijagram raspoređivanja TripPy sistema:



## Klijent

Pristup aplikaciji se obavlja preko klijentskih računara na kojima se izvršava Web čitač. Za povezivanje između klijenta i Web servera koristi se Internet infrastruktura tako da nema ograničenja u pogledu lokacije klijenta.

## Web server

Računar na kome se izvršava Web server opslužuje više klijenata koji pristupaju preko Interneta. Pored osnovnog procesa koji realizuje funkcionalnost Web servera, na ovom računaru mogu da se izvršavaju i procesi servera koji vrše obradu zahteva. U najopštioj konfiguraciji DBMS se izvršava na posebnoj mašini koja je sa Web serverom u lokalnoj mreži (LAN) ali može da se izvršava i na istoj mašini na kojoj se izvršava i serverski deo aplikacije.

## DBMS server

DBMS server je računar na kome se izvršava PostgreSQL Server proces koji realizuje funkcionalnost sistema za upravljanje bazama podataka. Zbog sigurnosti podataka koji se na ovom računaru čuvaju pristup bazi je ograničen samo na računare iz lokalne mreže (LAN).

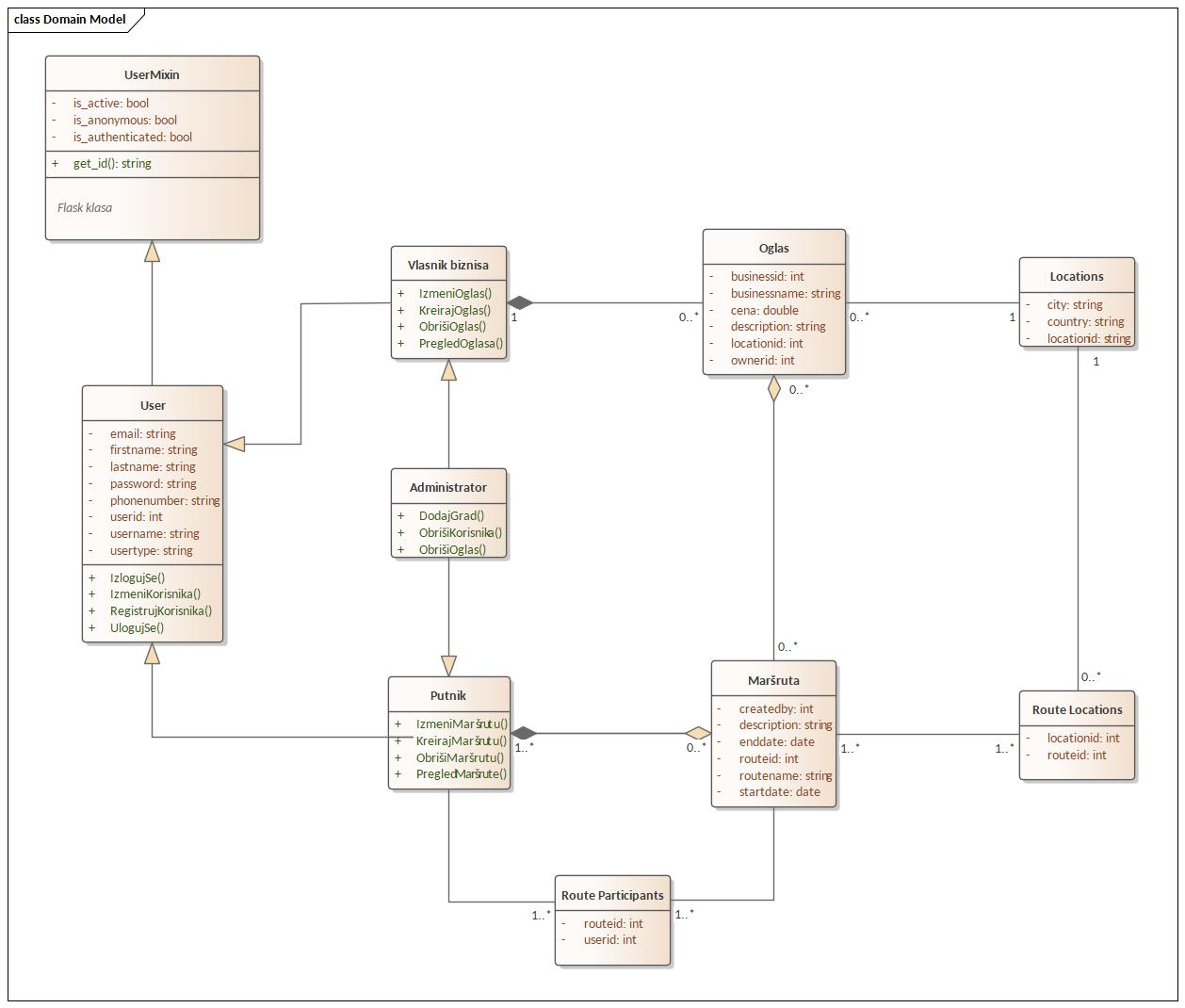
# Pogled na implementaciju sistema

Pogled na implementaciju prikazuje različite aspekte bitne za implementaciju sistema. U slučaju TripPy aplikacije ovaj odeljak sadrži model domena, šemu baze podataka i prikaz komponenti sistema razvrstanih u ranije identifikovane pakete.

## Model domena

Model domena za koji se TripPy sistem projektuje je ilustrovan UML dijagramom klasa. U njemu su prikazane domenske klase, neki od njihovih atributa, kao i veze koje se mogu identifikovati između njih.

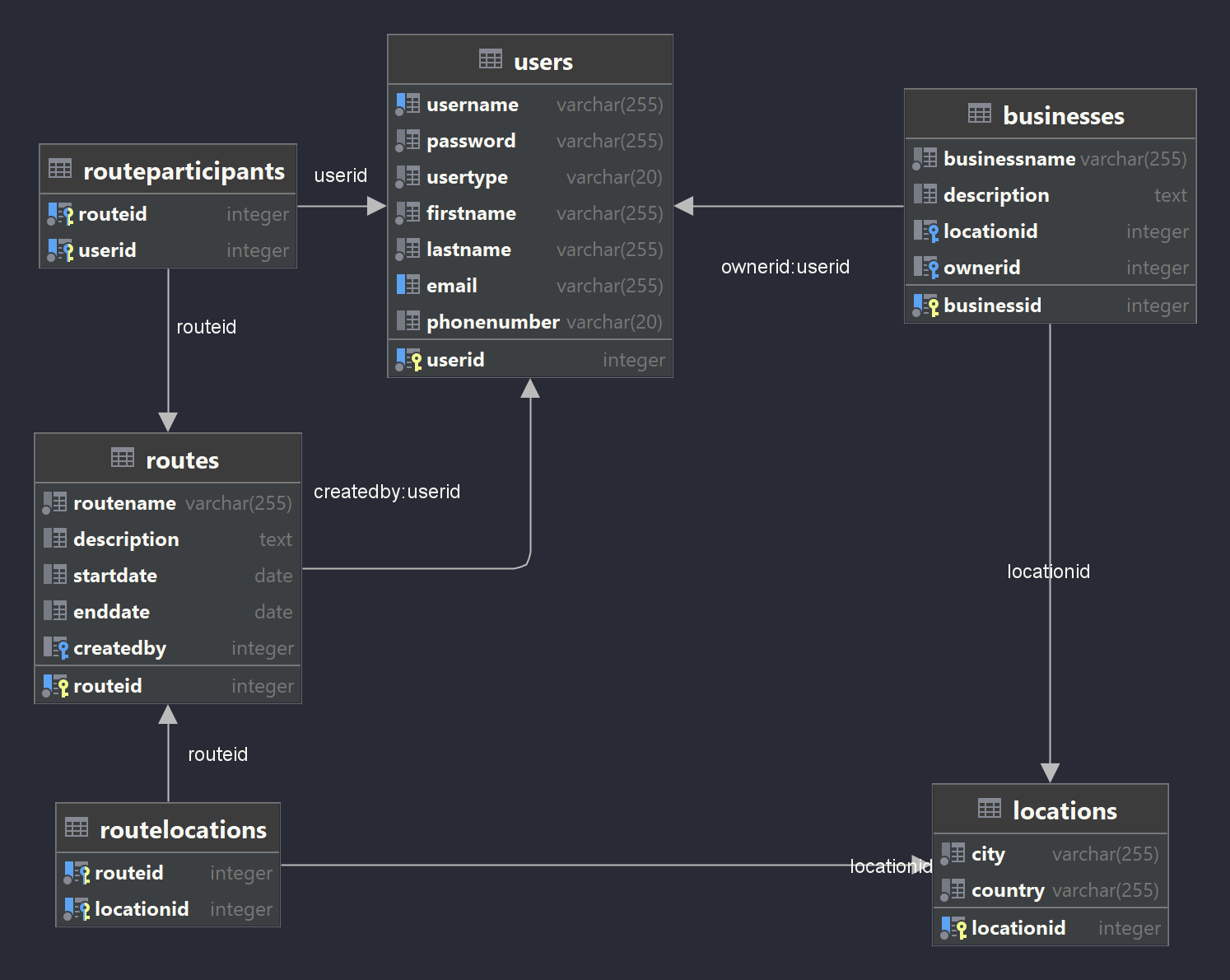
Model domena predstavlja osnovu za projektovanje baze podataka, ali i identifikaciju nekih od komponenti koje će biti implementirane.



## Šema baze podataka

Detaljna šema baze podataka je prikazana na sledećem dijagramu.

Prikazana šema je eksportovana pomoću alata *DataGrip*.



## Komponente sistema

Komponente sistema TripPy aplikacije su JS skripte i Python serverski deo čiji će pregled biti dat po arhitekturnim slojevima. Za ilustraciju će biti korišćeni UML dijagrami komponenti, ali i dijagrami klasa.

### Komponente korisničkog interfejsa

Dizajn korisničkog interfejsa je obuhvaćen dvema komponentama:

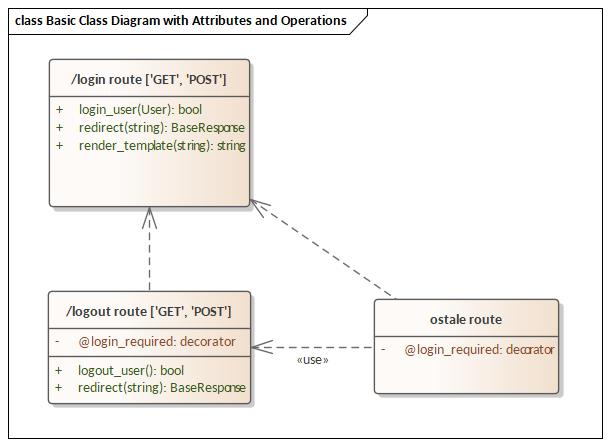


Komponenta **Route** implementira stranicu aplikacije čiji sadržaj može da varira od parametra koji joj se proslede u zahtevu.

Komponenta **Template** implementira tehnologiju Jinja2 aplikacije čiji sadržaj može da varira od parametra koji joj se proslede u zahtevu.

Komponenta **main.css** predstavlja opis stilova za pojedine HTML elemente koji se javljaju na različitim stranicama.

Sistem za logovanje korisnika ilustrovan je na sledećem dijagramu ruta:



Ovaj sistem pamti kredencijale u cookies pretraživača.

Značenje atributa i metoda je sledeće:

/login route [‘GET’, ‘POST’]

* login\_user(User)– koristi se da prijavi korisnika, tako što postavlja sesiju korisnika
* redirect(string)- koriste se za preusmeravanja korisnika na drugu lokaciju(URL).
* render\_template(string)- koristi se za prikazivanje šablona, tj HTML datoteke.

/logout route [‘GET’, ‘POST’]

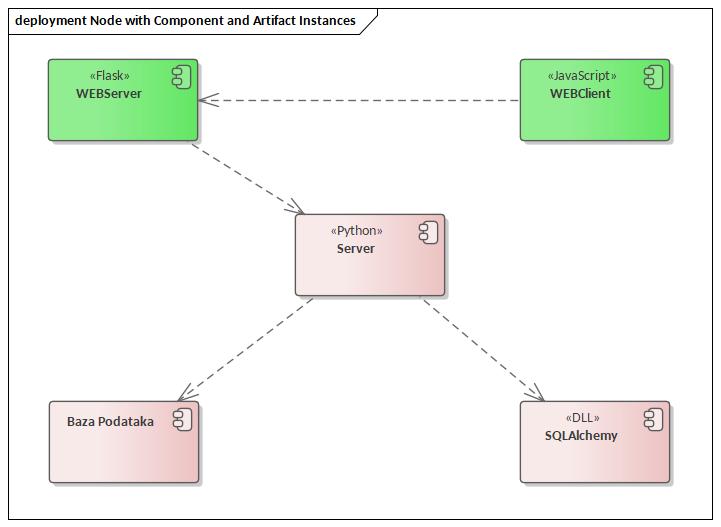
* @login\_required- dekorator se koristi za proveru da li je korisnik trenutno prijavljen.
* logout\_user()- čisti korisnikovu sesiju, čime korisnik biva izlogovan.
* redirect(string)- koriste se za preusmeravanja korisnika na drugu lokaciju(URL).

ostale rute

* @login\_required- dekorator se koristi za proveru da li je korisnik trenutno prijavljen.

### Komponente aplikacione logike

Komponente koje realizuju domen problema se uključuju isključivo prekokomponente **Flask Route**. Flask se stara o zahtevima. Na sledećem dijagramu su prikazane komponente ovog sloja i njihove međusobne zavisnosti:

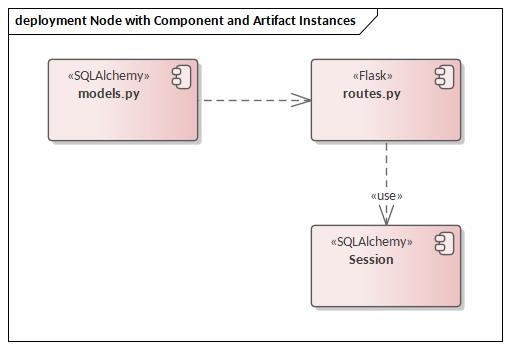


Opis komponenti:

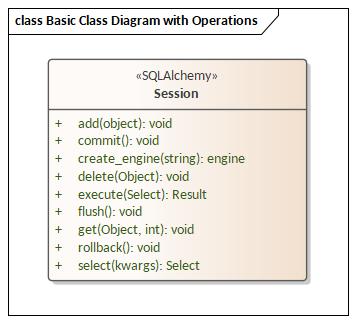
* WEBClient – Šalje zahteve do servera putem WEBServera, obrađuje i prikazuje pristigle podatke.
* WEBServer – Omogućava komunikaciju klijentskog i serverskog dela aplikacije.
* Server – Odgovara i obrađuje zahteve klijenta. Upisuje, ažurira i briše podatke iz baze podataka.
* Baza podataka – Skladišti podatke aplikacije.
* SQLAlchemy – Omogućava komunikaciju serverskog dela aplikacije sa bazom podataka.

### Komponente za pristup podacima

Pristup bazi podataka je u potpunosti zatvoren u funkcije koje su definisane u okviru routes.py. Date klase se mapiraju u models.py, instanciraju u routes.py i putem routes.py instanciranjem sesije pristižu upiti.



Na sledećem UML dijagramu klasa pobrojane su funkcije za pristup podacima iz baze:



Navedene funkcije obavljaju sledeće zadatke:

* add(object) - Dodaje objekat u sesiju, i on je spreman da se čuva u bazi podataka kada se sesija komituje.
* commit() - Potvrđuje sve promene napravljene u trenutnoj sesiji i trajno ih čuva u bazi podataka.
* create\_engine(string) - Kreira novi Engine objekat koji predstavlja konekciju sa bazom podataka.
* delete(Object) - Briše objekat iz baze podataka.
* execute(Select) - Izvršava SQL upit koji je prosleđen kao argument i vraća rezultat u obliku Result objekta.
* flush() - Sinhronizuje stanje objekata u sesiji sa stanjem u bazi podataka, ali ne komituje promene.
* get(Object) - Dobavlja objekat iz baze podataka na osnovu primarnog ključa (id) koji je prosleđen kao argument.
* select(kwargs) - Generiše SELECT upit na osnovu prosleđenih kriterijuma (kwargs) i vraća Select objekat koji može biti izvršen kako bi se dobio rezultat.

U okviru ovih funkcija se izvršava i povezivanje na bazu podataka pomoću datih atributa klase.

# Performanse

Izabrana arhitektura softvera podržava zahteve u pogledu broja korisnika koji mogu simultano pristupati sistemu i vremena odziva za pristup bazi podataka specificirane u zahtevima u pogledu performansi [5]:

1. Sistem će da podrži do 100 simultanih pristupa korisnika aplikaciji.
2. Vreme potrebno za pristupanje bazi podataka u cilju izvršenje nekog upita ne sme da bude veće od 5 sekundi.

Zahtevane performanse su zadovoljene izborom tehnologija na kojima će sistem biti razvijen i definisane hardverske platforme [5].

# Kvalitet

Izabrana arhitektura softvera podržava zahteve u pogledu dostupnosti i srednjeg vremena između otkaza specificirane u zahtevima u pogledu pouzdanosti [5]:

1. TripPy aplikacija će biti dostupna 24 časa dnevno, 7 dana u nedelji. Vreme kada aplikacija nije dostupna ne sme da pređe 10%.
2. Srednje vreme između dva sukcesivna otkaza ne sme da padne ispod 120 sati.