Fakultet strojarstva,računarstva i elektrotehnike  
Projektiranje infromacijskih sustava

Analiza i dizajn:

Gradski i prigradski prijevoz

„CityLink“

Verzija: 1.0

Voditelj projekta: Luka Vasilj

Mostar, Lipanj 2024.

Sadržaj

[1. Dijagram komponenti s reprezentativnim klasama 3](#_Toc169194348)

[1.1 Detaljan opis veza 3](#_Toc169194349)

[2. Dijagram ugradnje (razmještaja, rasporeda) s komponentama 4](#_Toc169194350)

[2.1 Objašnjenje veza i komponenti 5](#_Toc169194351)

[3. Implementacijski/instalacijski zahtjevi 6](#_Toc169194352)

[3.1. Klijentska aplikacija 6](#_Toc169194353)

[3.2. Aplikacijski poslužitelj 6](#_Toc169194354)

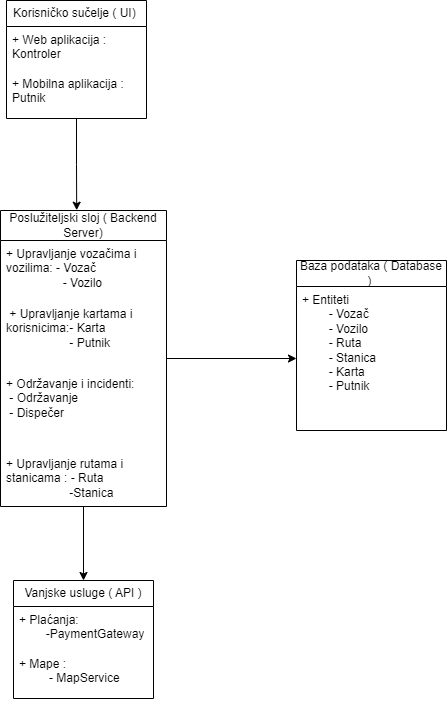
[3.3. DB poslužitelj 6](#_Toc169194355)

[3.4. Vanjske usluge 7](#_Toc169194356)

[3.5. Dodatni komentari i opis implementacijskih/instalacijskih zahtjeva 7](#_Toc169194357)

[4. Prilozi 8](#_Toc169194358)

# 1. Dijagram komponenti s reprezentativnim klasama

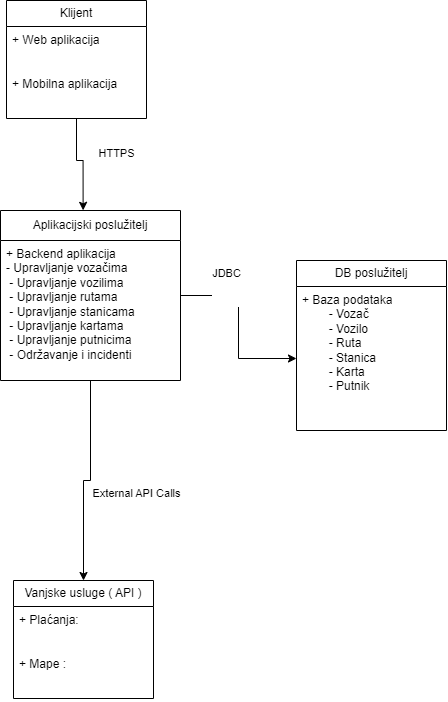


Slika 1. Dijagram komponenti s reprezentativnim klasama

## 1.1 Detaljan opis veza

* **Korisničko sučelje (UI)** koristi **REST API** za komunikaciju s **Poslužiteljskim slojem (Backend Server)**. Mobilna i web aplikacija šalju zahtjeve za različite operacije (kupovina karata, provjera ruta, registracija korisnika itd.) i dobivaju odgovore koje koriste za prikaz informacija korisnicima.
* **Poslužiteljski sloj (Backend Server)** koristi **SQL upite** za interakciju s **Bazom podataka (Database)** kako bi pohranio i dohvaćao potrebne podatke za različite entitete (vozači, vozila, rute, stanice, karte, putnici).
* **Poslužiteljski sloj (Backend Server)** također koristi **vanjske API pozive** za komunikaciju s **Vanjskim uslugama (API)**, poput plaćanja i navigacijskih servisa, kako bi obradio transakcije i pružio navigacijske podatke.
* **Baza podataka (Database)** može uključivati povratne mehanizme poput trigera i procedura koje komuniciraju s backend serverom za složenije operacije ili automatizacije.

# 2. Dijagram ugradnje (razmještaja, rasporeda) s komponentama



Slika 2. Dijagram ugradnje (razmještaja, rasporeda) s komponentama

## 2.1 Objašnjenje veza i komponenti

1. **Klijent**
   * **Mobilna aplikacija**: Instalirana na korisničkim mobilnim uređajima. Povezuje se s aplikacijskim poslužiteljem putem HTTPS protokola.
   * **Web aplikacija**: Dostupna putem web preglednika. Povezuje se s aplikacijskim poslužiteljem putem HTTPS protokola.
2. **Aplikacijski poslužitelj**
   * Hostira backend aplikaciju koja upravlja poslovnom logikom sustava (vozači, vozila, rute, stanice, karte, putnici, održavanje i incidenti).
   * Komunicira s DB poslužiteljem putem JDBC protokola za pristup bazi podataka.
   * Komunicira s vanjskim uslugama putem vanjskih API poziva.
3. **DB poslužitelj**
   * Sadrži bazu podataka koja pohranjuje sve potrebne informacije (vozači, vozila, rute, stanice, karte, putnici).
   * Baza podataka je dostupna aplikacijskom poslužitelju putem JDBC protokola.
4. **Vanjske usluge**
   * **PaymentGateway**: Vanjska usluga za obradu plaćanja, s kojom aplikacijski poslužitelj komunicira putem vanjskih API poziva.
   * **MapService**: Vanjska usluga za kartografske i navigacijske informacije, s kojom aplikacijski poslužitelj komunicira putem vanjskih API poziva.

# 3. Implementacijski/instalacijski zahtjevi

## 3.1. ****Klijentska aplikacija****

* **Mobilna aplikacija**:
  + **Platforme**: Dostupna na Android i iOS uređajima.
  + **Instalacija**: Aplikacija treba biti dostupna za preuzimanje iz Google Play Store-a i Apple App Store-a.
  + **Sigurnost**: Implementirati SSL/TLS za sigurnu komunikaciju s aplikacijskim poslužiteljem.
  + **Ažuriranja**: Redovita ažuriranja putem trgovina aplikacija za ispravljanje grešaka i dodavanje novih značajki.
* **Web aplikacija**:
  + **Pristupačnost**: Dostupna putem modernih web preglednika (Chrome, Firefox, Safari, Edge).
  + **Sigurnost**: Implementirati SSL/TLS za sigurnu komunikaciju s aplikacijskim poslužiteljem.
  + **Responzivni dizajn**: Omogućiti prilagodbu korisničkog sučelja za različite veličine zaslona i uređaje.

## 3.2. ****Aplikacijski poslužitelj****

* **Poslužiteljska infrastruktura**:
  + **Operacijski sustav**: Preporučeno korištenje Linux distribucije (npr. Ubuntu Server) zbog stabilnosti i sigurnosti.
  + **Resursi**: Dovoljno CPU, RAM i diskovnog prostora za podršku broju korisnika i obrade podataka.
  + **Skalabilnost**: Mogućnost horizontalnog skaliranja za podršku povećanom broju korisnika i opterećenju.
* **Sigurnost**:
  + **SSL/TLS**: Implementirati za sve komunikacije s klijentima i vanjskim uslugama.
  + **Firewall**: Konfigurirati firewall za zaštitu od neovlaštenog pristupa.
  + **Autentifikacija i autorizacija**: Koristiti robustne mehanizme autentifikacije (npr. OAuth2) za sigurnost korisničkih podataka.
* **Razvoj i implementacija**:
  + **Kontinuirana integracija**: Postaviti CI/CD pipeline za automatsku izgradnju, testiranje i implementaciju aplikacije.
  + **Praćenje i nadzor**: Implementirati alate za praćenje performansi aplikacije (npr. Prometheus, Grafana) i logiranja (npr. ELK stack).

## 3.3. ****DB poslužitelj****

* **Baza podataka**:
  + **RDBMS**: Preporučeno korištenje PostgreSQL ili MySQL.
  + **Resursi**: Dovoljno CPU, RAM i diskovnog prostora za pohranu i obradu podataka.
  + **Sigurnost**:
    - **Enkripcija**: Koristiti enkripciju za podatke u tranzitu i podatke u mirovanju.
    - **Backup**: Redoviti backup podataka s planom za oporavak u slučaju katastrofe.
    - **Pristup**: Ograničiti pristup bazi podataka samo na aplikacijski poslužitelj i ovlašteno osoblje.
* **Performanse**:
  + **Indeksiranje**: Koristiti indekse za optimizaciju upita.
  + **Replikacija**: Postaviti replikaciju za visoku dostupnost i skalabilnost.
  + **Monitoring**: Praćenje performansi baze podataka i automatsko upozoravanje na probleme.

## 3.4. ****Vanjske usluge****

* **Integracija s vanjskim API-ima**:
  + **PaymentGateway**:
    - **Autentifikacija**: Korištenje API ključeva ili OAuth za sigurnu komunikaciju.
    - **Testno okruženje**: Implementirati i testirati integraciju u sigurnom testnom okruženju prije produkcijskog rada.
  + **MapService**:
    - **Autentifikacija**: Korištenje API ključeva.
    - **Rate Limiting**: Implementirati rješenja za rukovanje ograničenjima broja poziva (rate limiting).
* **Sigurnost**:
  + **SSL/TLS**: Osigurati da sve komunikacije s vanjskim uslugama budu šifrirane.
  + **Error handling**: Implementirati robusne mehanizme za rukovanje greškama i ponovni pokušaj neuspjelih poziva.
* **Performanse**:
  + **Caching**: Koristiti keširanje rezultata vanjskih API poziva gdje je moguće kako bi se smanjilo opterećenje i ubrzale performanse.
  + **Timeouts**: Postaviti razumne timeout postavke za API pozive kako bi se spriječilo dugotrajno čekanje na odgovor.

## 3.5. Dodatni komentari i opis implementacijskih/instalacijskih zahtjeva

1. **Planiranje kapaciteta**:
   * Procijeniti očekivano opterećenje sustava (broj korisnika, broj transakcija) i planirati kapacitet poslužitelja i baze podataka u skladu s tim.
2. **Testiranje**:
   * Provesti sveobuhvatno testiranje sustava uključujući jedinicne testove, integracijske testove, testove performansi i sigurnosne testove.
   * Implementirati automatizirano testiranje kao dio CI/CD pipelinea.
3. **Dokumentacija**:
   * Održavati ažuriranu dokumentaciju za sve dijelove sustava uključujući API dokumentaciju, korisničke priručnike i administrativne vodiče.
4. **Sigurnosne kopije i oporavak**:
   * Postaviti redovite sigurnosne kopije svih kritičnih podataka i razviti planove za oporavak u slučaju katastrofe.
5. **Kontinuirano poboljšanje**:
   * Prikupljati povratne informacije od korisnika i redovito ažurirati i poboljšavati sustav na temelju tih povratnih informacija.

# 4. Prilozi

* Dijagram komponenti s reprezentativnim klasama.drawio
* Dijagram komponenti s reprezentativnim klasama.drawio.png
* Dijagram ugradnje (razmještaja, rasporeda) s komponentama.drawio
* Dijagram ugradnje (razmještaja, rasporeda) s komponentama.drawio.png