FAZA I – ARHITEKTURA

Naziv projekta: **Kolaborativni Travel Planner**

1. Kontekst i cilj softverskog projekta

Moderni putnici cesto planiraju grupna putovanja putem neefikasnih kanala (grupe, tabele). Ovaj sistem omogucava centralizovano mesto za planiranje. Cilj projekta je razvoj visekorisnicke web aplikacije za kolaborativno planiranje putovanja u realnom vremenu. Sistem omogucava vise korisnika da istovremeno rade nad istim planom putovanja, uz podrsku za perzistenciju podataka i sinhronizaciju izmena. Arhitektura koristi message-passing mehanizme i primenu projektnih obrazaca.

2. *Arhitekturni zahtevi*

2.1 Arhitekturno znacajni slucajevi koriscenja

- Kreiranje novog plana putovanja
- Dodavanje, izmena i brisanje aktivnosti u okviru plana
- Istovremeni rad vise korisnika nad istim planom
- Zakljucavanje plana putovanja (samo jedan korisnik ima pravo izmene)
- Podrska za vise nezavisnih planova putovanja

2.2 Atributi kvaliteta (ne-funkcionalni zahtevi)

- POUZDANOST - podaci o planovima putovanja moraju biti trajno sacuvani
- SKALABILNOST – podrska za veci broj korisnika i planova
- BEZBEDNOST – korisnici mogu pristupati samo planovima za koje imaju dozvolu
- PERFORMANSE – izmene se moraju propagirati klijentima u realnom vremenu
- ODRZIVOST – mogucnost lakog prosirenja funkcionalnosti

2.3 Tehnicka i poslovna ogranicenja

- Upotreba message-passing mehanizma za komunikaciju

- Upotreba ORM alata za rad sa bazom podataka
- Implementacija

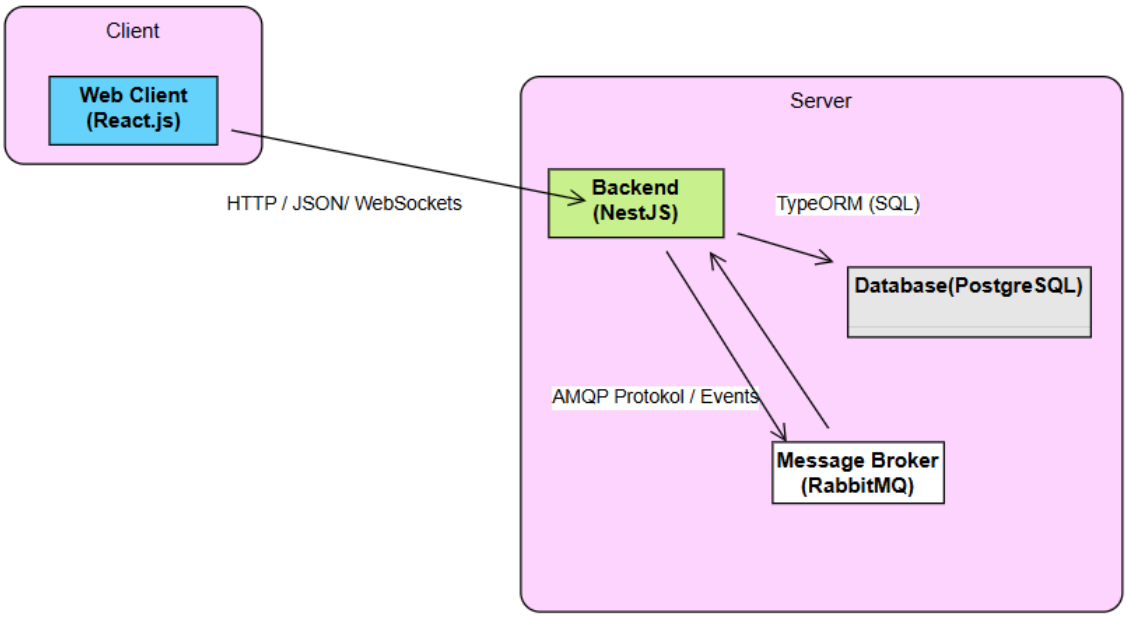
3. Arhitekturni dizajn

3.1 Arhitekturni obrasci

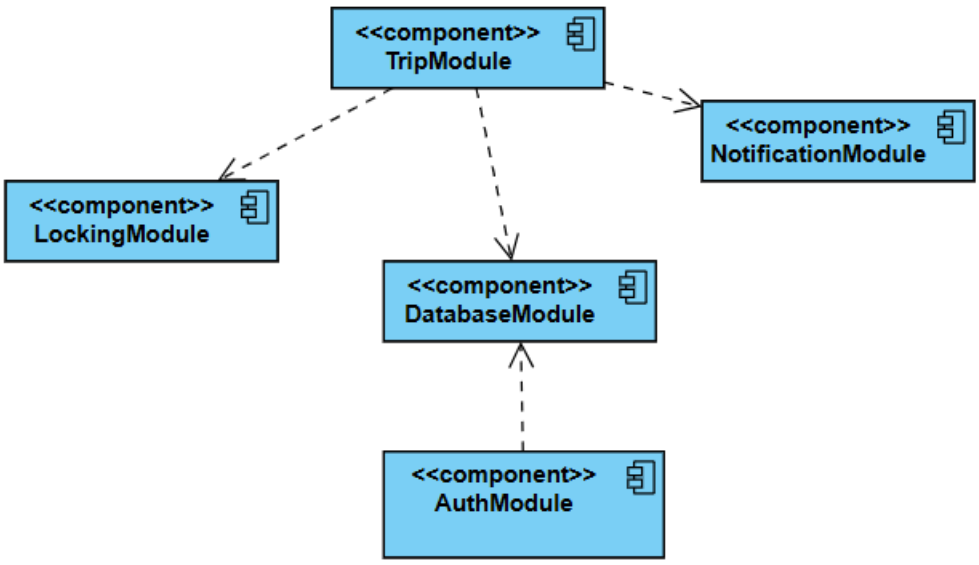
Za ovaj sistem koristimo:

- Slojevita arhitektura: Primenjena na backendu (NestJS) kroz slojeve kontrolera, servisa i repozitorijuma.
- Pub/Sub: Koriscen za real-time sinhronizaciju putem RabbitMQ-a.
- Client-Server: Jasna podela sistema na React frontend i NestJS backend.
- Repository: Izmedju servisa i baze podataka (TypeORM) radi apstrakcije pristupa podacima.

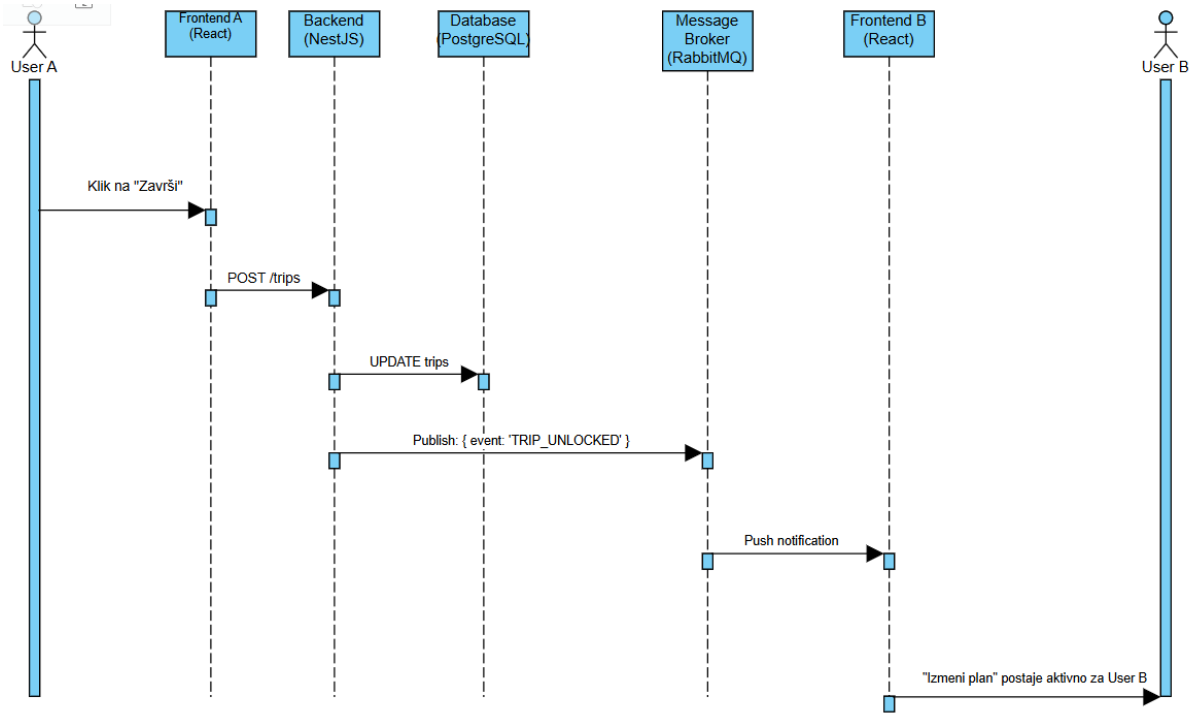
3.2 Generalna arhitektura (Box-line dijagram)

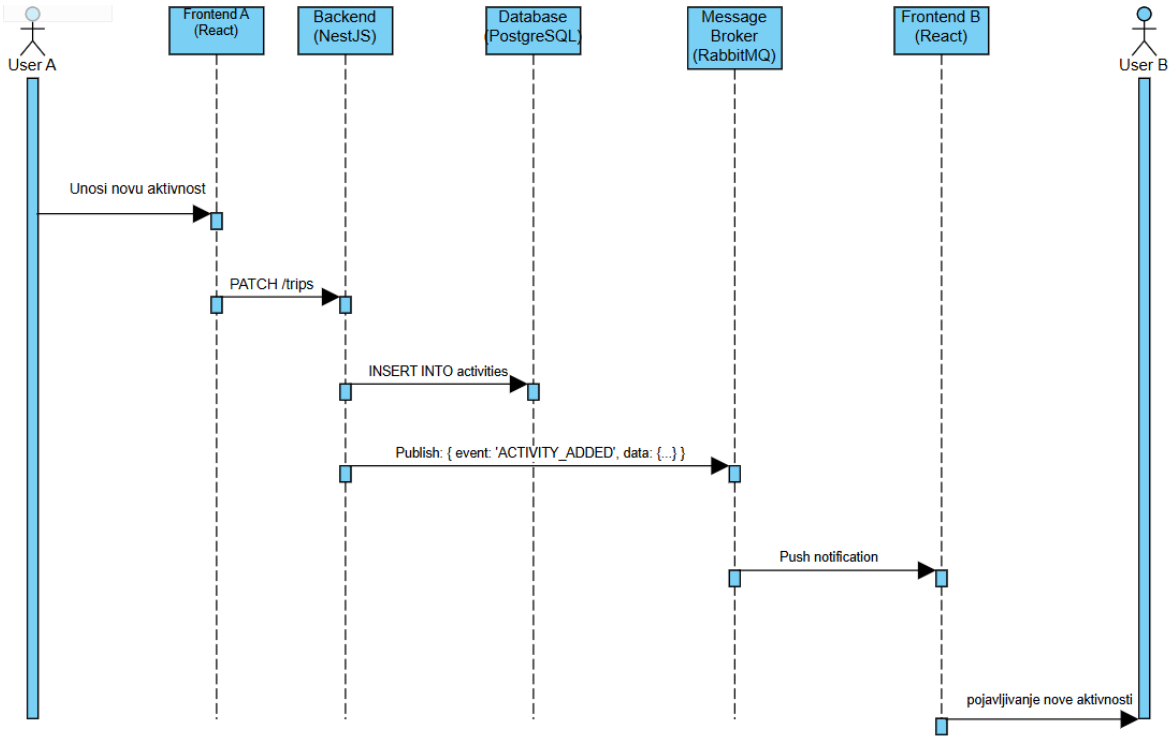


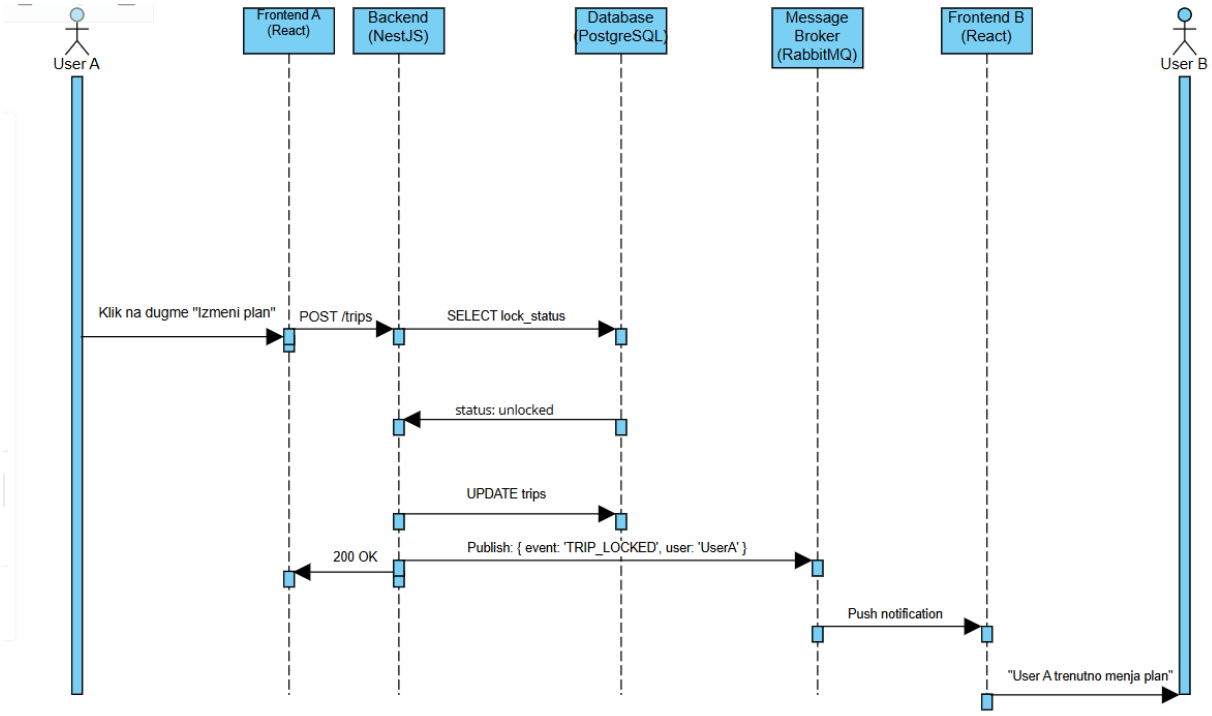
3.3 Strukturni pogledi



3.4 Bihevioralni pogledi







3.5 Implementaciona pitanja

Sledeca 3 obrasca se implementiraju u domenskoj oblasti:

- State Pattern (pracenje stanja plana puta)
- Strategy Pattern (razlicite strategije u zavisnosti od uloge korisnika)
- Command Pattern (svaka instrukcija kao komanda)

U nasem projektu se koristi sledece:

- **Frameworks:** NestJS (Backend), React.js (Frontend).
- **ORM:** TypeORM za mapiranje objekata u PostgreSQL bazu.
- **Messaging:** RabbitMQ .
- **Biblioteke:** socket.io za WebSocket komunikaciju, passport za autentifikaciju.

4. Analiza arhitekture

4.1 Potencijalni rizici i strategije prevazilazenja

- Rizik: Korisnik zakljuci plan i izgubi konekciju.
  Strategija: Implementacija timeout mehanizma.

- Rizik: RabbitMQ padne u toku rada.
  Strategija: Persistent redovi i potvrda o prijemu (ACK).

- Rizik: Vise korisnika istovremeno pristupa planu.
  Strategija: Optimizacija; salju se male izmene, a ne ceo plan puta.

19101 Nadja Dinic
18549 Nadja Stosic