Event Driven

Studenti: Predmetni profesor:

Kristina Joksimović, 18203 Bratislav Predić

Anđelija Mijajlović, 18247 Dragan Stojanović

Mihajlo Bencun, 18081

Komunikacija

Korišćenje RabbitMQ i WebSocket tehnologija, zajedno sa dizajnerskim obrascima kao što su Singleton, Observer, Memento i Factory, omogućilo nam je da izgradimo skalabilan, efikasan i održiv sistem za obaveštavanje korisnika o porukama, notifikacijama i dokumentima. Ovi obrazci ne samo da poboljšavaju strukturu i modularnost koda, već i olakšavaju njegovo održavanje i proširivanje.

U našem projektu koristimo RabbitMQ i WebSocket za komunikaciju između različitih komponenti sistema. Cilj ovog izveštaja je da detaljno objasni kako koristimo ove tehnologije za obaveštavanje korisnika o porukama, notifikacijama i dokumentima.

# RabbitMQ

RabbitMQ je robustan alat za razmenu poruka koji omogućava asinkronu komunikaciju između različitih delova sistema. Koristimo RabbitMQ za slanje i primanje poruka između različitih servisa u našoj aplikaciji.

Primer: U našoj aplikacije iskorišćen je, između ostalog, notifications\_queue u okviru RabbitMQ koji pamti sve notifikacije koje stu stigle, za izmenu dokumenta, za priključivanje nekom dokumentu i tome slicno..

# WebSocket

WebSocket omogućava real-time, dvosmernu komunikaciju između klijenata i servera. Koristimo WebSocket za slanje real-time obaveštenja korisnicima.

# Design patterns

#### Singleton Pattern

Singleton pattern osigurava da će klasa imati samo jednu instancu i pruža globalnu tačku pristupa toj instanci. U našem projektu, Singleton pattern je korišćen za Nest.js servise, kao što je servis za povezivanje sa bazom podataka.

#### Observer Pattern

Observer pattern omogućava objektima da budu obavešteni o promenama stanja drugih objekata. U našem projektu, Observer pattern je korišćen za obaveštavanje korisnika o novim porukama i notifikacijama putem WebSocket-a.

Subject: Subjekt je objekat koji održava listu svojih posmatrača (observersa) i obaveštava ih o promenama stanja. U našem slučaju, RabbitMQ može biti subjekt jer generiše događaje (notifikacije I poruke) koje treba preneti posmatračima.

Observer: Posmatrač je objekat koji prima obaveštenja od subjekta o promenama stanja. U našem slučaju, WebSocket Gateway i WebSocket klijenti su posmatrači koji primaju obaveštenja o događajima.

ConcreteObserver; Konkretni posmatrači su objekti koji implementiraju interfejs posmatrača i izvršavaju akcije kada prime obaveštenje o promeni stanja. U našem slučaju, React komponente koje prikazuju notifikacije I poruke su konkretni posmatrači.

#### Memento Pattern

Memento pattern omogućava snimanje i vraćanje prethodnog stanja objekta bez otkrivanja njegove interne strukture. U našem projektu, Memento pattern je korišćen za implementaciju undo/redo funkcionalnosti za dokumente.

#### Factory Pattern

Factory pattern omogućava kreiranje objekata bez eksplicitnog navođenja tačne klase objekta koji će biti kreiran. U našem projektu, Factory pattern je korišćen za kreiranje različitih tipova dokumenata.

# 