Arhitektura i projektovanje softvera

FindYourPlace

Arhitekturni projekat

Aleksandar Petković 15814

Nenad Nikolić 15783

Stefan Krstić 15690

Opis sistema

FindYourPlace je socijalna mreza za deljenje i trazenje razlicitih mesta i dogadjaja. Dogadjaji mogu biti koncerti, zurke, meetup-ovi, predavanja, seminari, takmicenja itd. Aplikacija omogucava korisnicima dodavanje dogadjaja i mesta kao i njihovo pretrazivanje prema zadatim kriterijumima i prikazivanje na mapi. Takodje omogucava da se klijenti pretplate na neki dogadjaj i dobijaju informacije kada dodje do izmene tog dogadjaja. Aplikacija omogucava i povezivanje i komunikaciju izmedju korisnika (chat).

Funkcionalni zahtevi

1. Povezivanje klijenta i servera.
2. Registracija korisnika
3. Obeležavanje i deljenje događaja.
4. Skladištenje podataka o događajima.
5. Prijavljivanje na događaj.
6. Komunikacija između korisnika.
7. Pregled profila korisnika.
8. Prikaz lokacije događaja na mapi.

Nefunkcionalni zahtevi

1. Pouzdanost (reliability)
2. Lakoća korišćenja (usability)
3. Proširljivost (extensibility)
4. Performanse
5. Modifikabilnost
6. Skalabilnost (scalability)
7. Otpornost na greške (fault tolerance)
8. Lako održavanje (maintainability)

Arhitekturni zahtevi

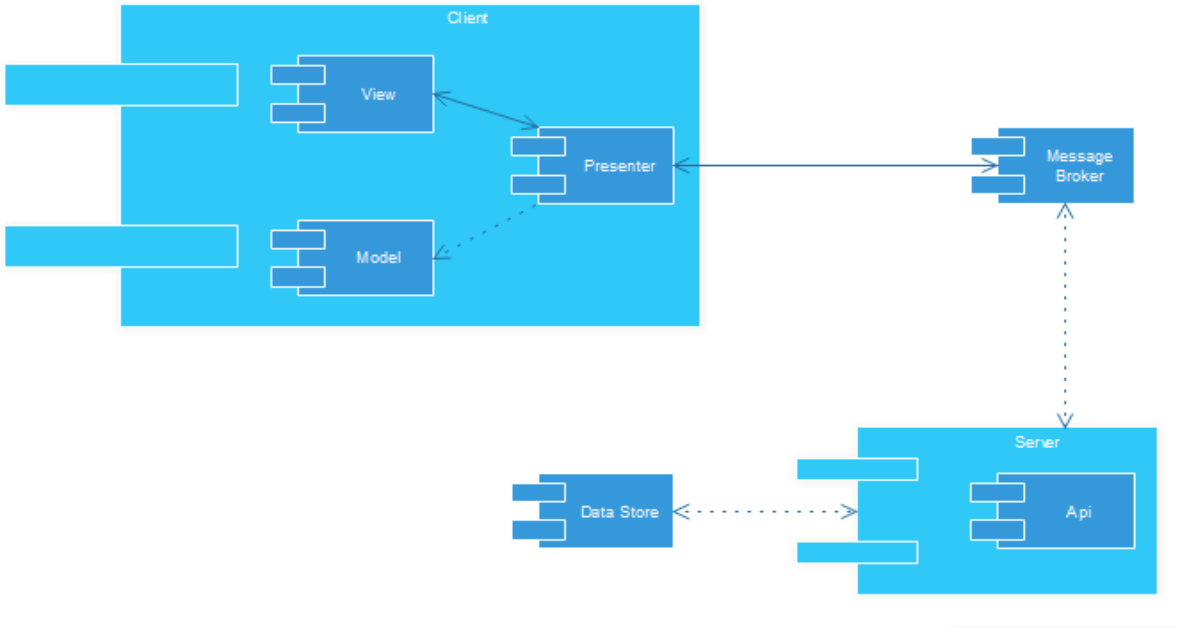
Na osnovu navedenih nefunkcionalnih zahteva i ograničenja, imamo sledeće arhitekturne zahteve:

1. Pouzdanost - Aplikacija ne sme korisnicima da prikaže nevalidne. Takođe ne sme doći do gubljenja poruka koje se razmenjuju između korisnika i servera. Šema baze podataka je skrivena.
2. Lakoća korišćenja - Korisnički interfejs treba da bude intuitivan, jednostavan i lak za korišćenje.
3. Proširljivost - Arhitektura aplikacije mora da bude takva da omogući lako dodavanje novih funkcionalnosti.
4. Performanse - Aplikacija treba novonastale promene dogadjaja da propagira korisnicima koji nadgledaju taj dogadjaj u roku od 3 sekunde.
5. Modifikabilnost- laka promena sistema
6. Skalabilnost- Aplikacija mora da radi i kada se poveća broj korisnika.
7. Otpornost na greške - U slučaju loše akcije korisnika ili u slučaju da je sistem nedostupan treba korisniku prikazati odgovarajuću poruku.
8. Lako održavanje - Komponente sistema treba da imaju jednostavna i lako razumljiva imena koja opisuju njihovu svrhu u cilju lakšeg održavanja koda u budućnosti.

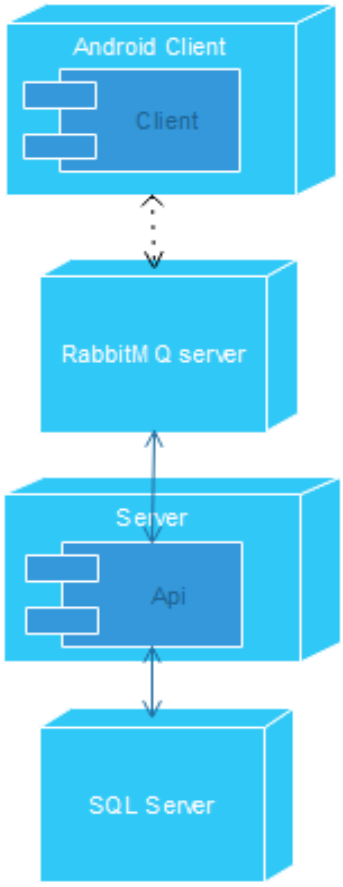
Arhitekturni dizajn

Strukturni pogled

Uprošćen prikaz arhitekture sistema dat je na slici ispod. Osnovnu arhitekturu sistema čini Model-View-Presenter arhitekturni obrazac implementiran u klijentu.Za potrebe real-time komunikacije koristi se Message Broker RabbitMQ koji implementira publisher-subscriber arhitekturni obrazac. Na serveru se izvršava Api koji komunicira sa bazom podataka.

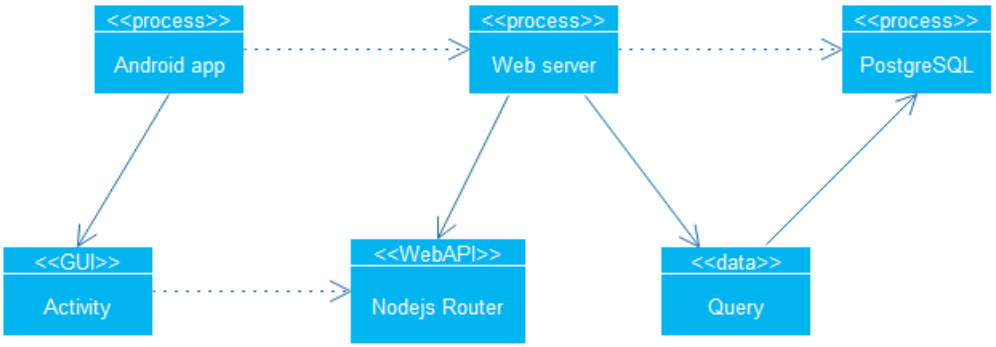


Alokacioni pogled

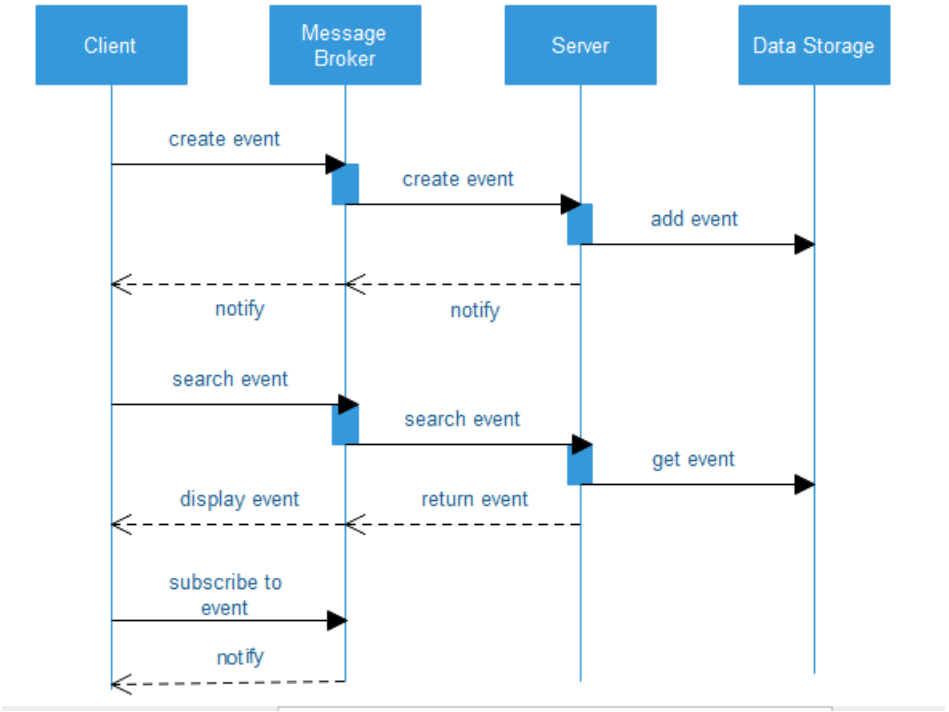


Pogled na procese

Dat je pregled najvaznijih procesa



Bihevioralni pogled



Arhitekturni obrasci

1. Layered pattern

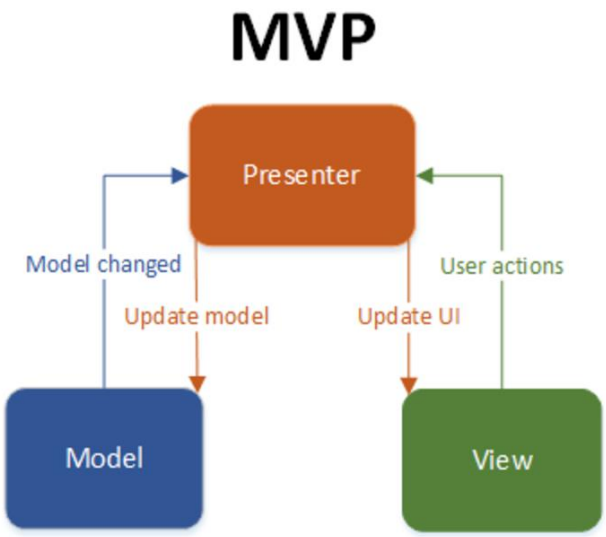
Aplikacija ce se sastojati od:

- Sloja perzistencije gde se nalazi baza podataka ,

- Serverskog sloja na kome se nalazi biznis model podataka i realizovana logika aplikacije,

- Klijentskog sloja u kome je realizovana celokupna prezentaciona logika pomoću koje korisnici sistema komuniciraju sa Serverskim slojem, odnosno poslovnom logikom i modelom podataka. Klijentski sloj sadrži i logiku za obradu podataka i rad sa podacima koji se čuvaju lokalno .

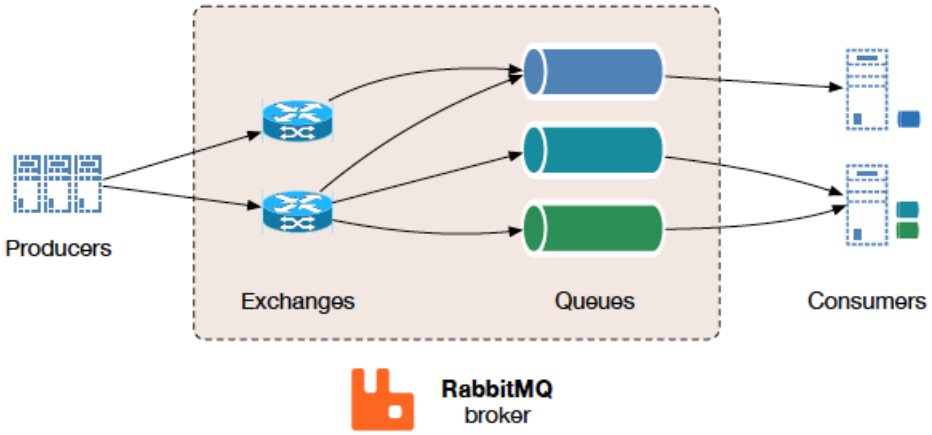
2.MVP (model-view-presenter) pattern



Predstavlja modifikaciju MVC pattern-a gde ne komuniciraju direktno View i Model.

3.Publish-Subscriber pattern

Koristi se za komunikaciju izmedju klijenta i servera. Ovaj obrazac je sadržan u RabbitMQ Message Broker-u. Njegova uloga je da pojednostavi razmenu poruka izmedju aplikacija.Obezbedjuje aplikacijama jednostavnu platformu za slanje i primanje poruka i sigurnost da će poruke biti isporučene. RabbitMQ razmenu poruka omogucava korišćenjem redova.Osnovna arhitektura RabbitMQ-a je data na sledećoj slici:



Specifikacija biblioteka i programskih okvira

Klijent:

Za izradu klijentskog dela aplikacije koristiće se **Android studio IDE.**

Server:

Za izradu servera koristiće se **Node.js framework** i **Express.js framework.**

Baza podataka:

Za skladištenje podataka koristiće se **PostgreSQL** objektno-relacioni sistem za upravljanje bazama podataka.

Komunikacija:

Za komunikaciju će se koristiti **RabbitMQ** - message broker zadužen za asinhronu komunikaciju.
Java AMPQ library - biblioteka za povezivanje sa Message Brokerom koja će se koristiti za klijentski deo aplikacije.
Amqplib - biblioteka za povezivanje sa Message Brokerom iz node.js-a

Objektno relaciono mapiranje:

Za mapiranje objekata u relacije koristiće se **Sequelize.** Sequelize je promise-based ORM za node.js sa podrškom za dijalekt PostgreSQL.