ARHITEKTURA I PROJEKTOVANJE SOFTVERA

FamilyConnect

Arhitektura projekta

Student:

Stefan Petković 18325

1. Kontekst i cilj projekta

FamilyConnect je mobilna android aplikacija za organizaciju porodičnih aktivnosti i informisanje svih članova korišćenjem zajedničkog kalendara aktivnosti, zajedničke liste za poslove, zajedničke liste za kupovinu namirnica, praćenje finansija i mogućnost komunikacije između članova putem poruka.

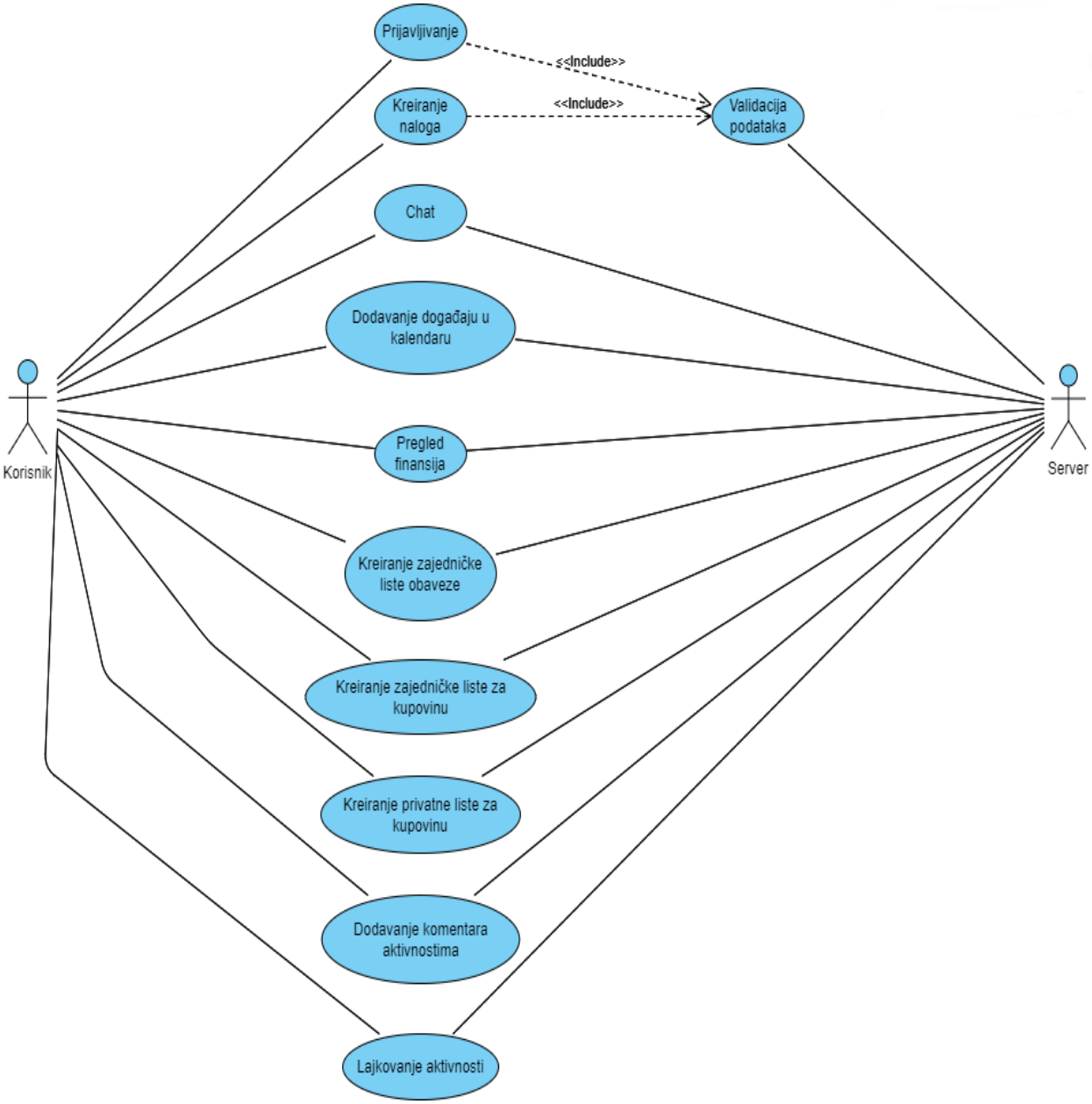
Komunikacija i interakcija sa svim funkcionalnostima unutar aplikacije je organizovana unutar grupe koju čine korisnici. Korisnici imaju mogućnost kreiranja sopstvene privatne liste za kupovinu namirnica nevezano za njihovu aktivnost unutar neke grupe.

2. Arhitekturni zahtevi

Ovde ćemo definisati arhitekturno značajne slučajeve korišćenja, glavne funkcionalne i ne-funkcionalne zahteve, kao i tehnička i poslovna ograničenja vezana za realizaciju projekta FamilyConnect.

Funkcionalni zahtevi aplikacije:

- Kreiranje korisničkog naloga, logovanje i editovanje profila
- Postojanje zajedničkog kalendara na kome članovi grupe mogu da dodaju i uređuju aktivnosti koje svi članovi mogu da vide, takođe postoji i mogućnost postavljanja podsetnika u vidu notifikacija
- Postojanje zajedničke liste obaveza unutar grupe kao i kreiranje privatne liste , mogućnost dodele obaveze nekom od članova grupe
- Postojanje zajedničke liste za kupovinu namirnica , kao i mogućnost kreiranja privatne liste
- Mogućnost praćenja finansija unutar grupe
- Mogućnost komunikacije između korisnika u aplikaciji u vidu poruka ondnosno preko chata
- Mogućnost dodavanja komentara i lajkovanje aktivnosti koje postavljaju članovi grupe



3. Nefunkcionalni zahtevi

Nefunkcionalni zahtevi definišu atribute sistema poput bezbednosti, performansi, održavanja , korisćenja i sl. Nefunkcionalni zahtevi sami po sebi unose određena ograničenja u dizajnu sistema. Nefunkcionalni zahtevi u okviru projekta FamilyConnect su :

- **Funkcionalnost:** Sistem treba da bude funkcionalan i da na zadovoljavajuć način izvršava sve ono za šta je namenjen.
- **Pouzdanost**: U okviru aplikacije je potrebno obezbediti da aplikacija bude dovoljno pouzdana kako ne bi došlo do gubitka

podataka u slučaju gubitka konekcije sa serverom. Takođe je potrebnu da aplikacija korisnicima bude dostupna 24h dnevno.

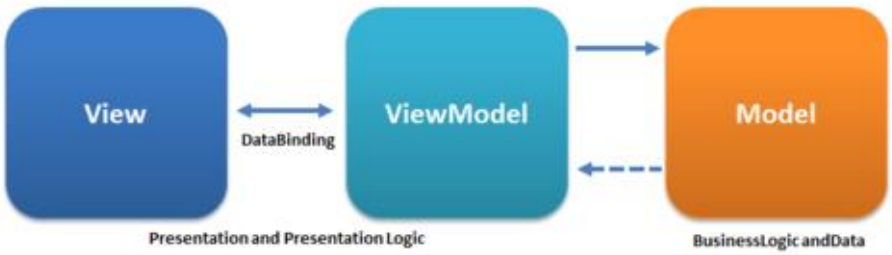
- **Održavanje**: Sistem je potrebno blagovremeno analizirati i održavati i ukoliko se pronađe neka greška u funkcionalnostima blagovremeno je ukloniti iz sistema.
- **Bezbednost**: Sistem treba da čuva podatke u bazi , i korisnički podaci takođe moraju biti zaštićeni.
- **Upotrebljivost**: Aplikacija treba da ima intuitivan interfejs koji je jednostavan za korišćenje većini korisnika.
- **Skalabilnost**: Ukoliko aplikacija ima povećan broj korisnika potrebno je obezbediti normalan rad aplikacije prilikom velikog broja korisnika i potrebno je obezbediti da ne dođe do pada sistema usled velikog broja korisnika.

4. Arhitekturni dizajn

U nastavku će biti pretstavljeni arhitekturni obrasci koji će biti korišćeni u izradi projekta. Pored ovih obrazaca, moguće je da se u toku implementacije dodaju još neki obrasci ukoliko se za njima ukaže prilika.

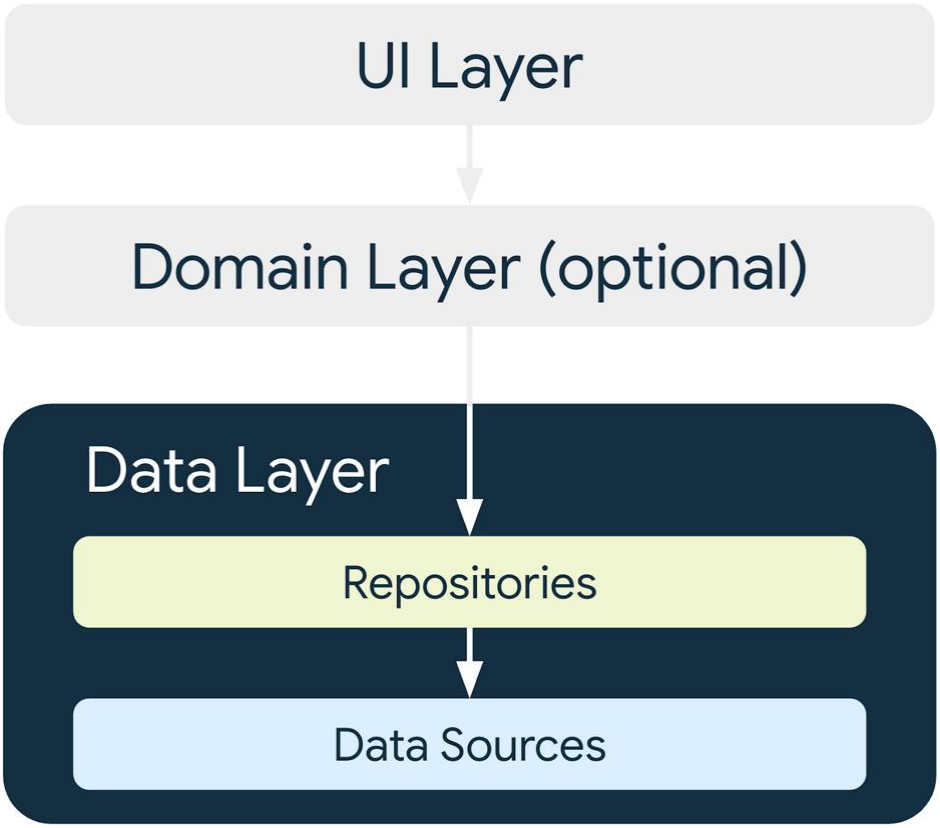
4.1 Model – View – ViewModel

Model-View-ViewModel se u našem projektu prvenstveno koristi zbog radvajanja korisničkog interfejsa o poslovne logike aplikacije. Takođe ovaj obrazac deluje kao posrednik između pogleda i modela i služi za prikaz podataka koji se mogu vezati za odrešen pogled. Ovakav vid radvajanja pruža mogućnosti boljeg testiranja, olakšano održavanje aplikacije i njenu fleksibilnost.



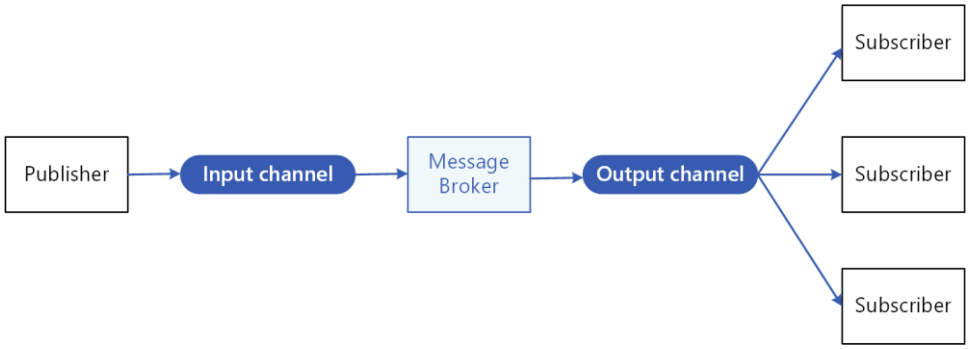
4.2 Repository

Repository obrazac će se u našoj aplikaciji koristiti za izolaciju sloja podataka od ostatka aplikacije. Sloj podataka se u našem slučaju odnosi na deo aplikacije koji je odvojen od korisničkog interfejsa. Ovo odvajanje se omogućava preko Api-ja koji će služiti za komunikaciju između naše aplikacije i baze podataka , a to se postiže mapiranjem podataka nekim ORM , a sam proces mapiranja korisnik ne vidi, već se njemu prikazuju samo izmene koje se dešavaju nad odgovarajućim podacima.



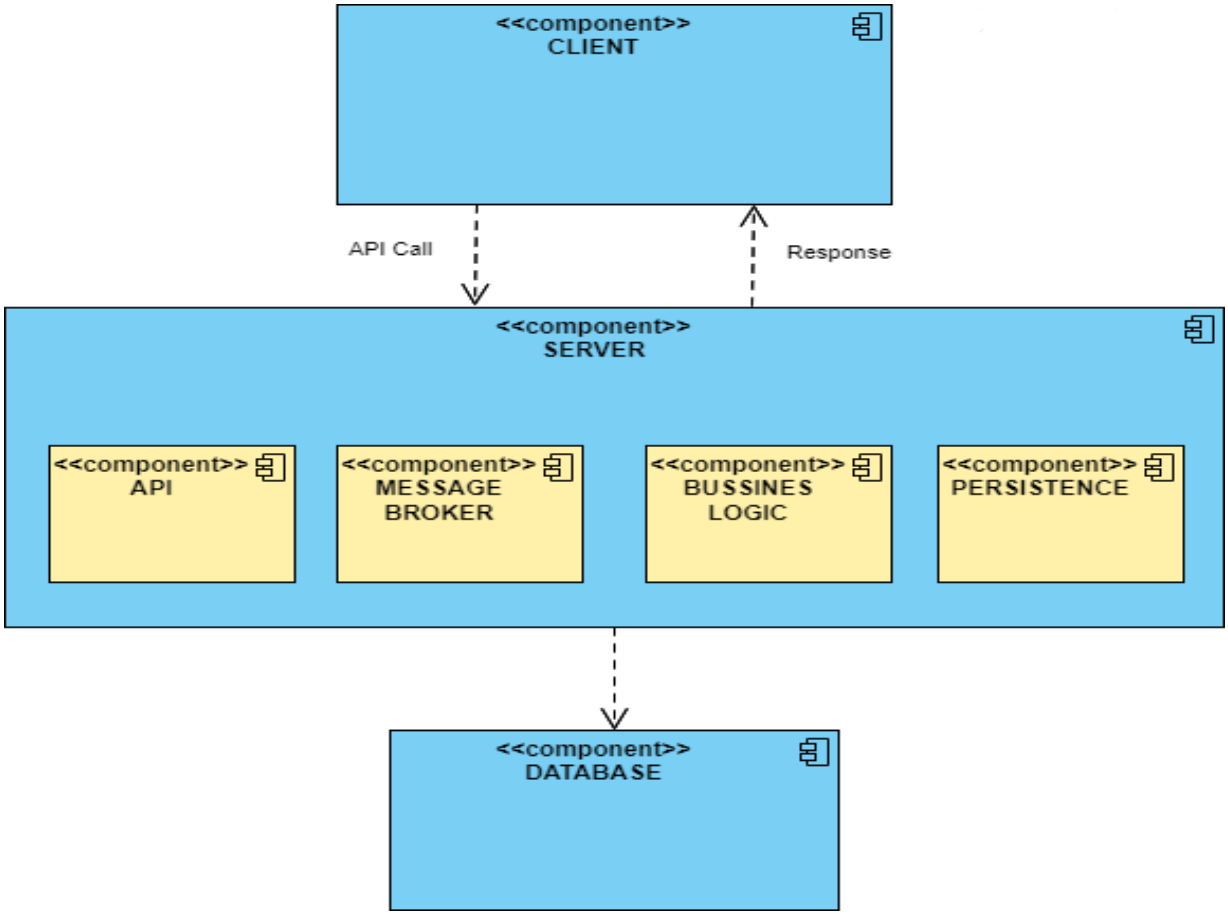
4.3 Publish – Subscribe

U okviru razmene poruka biće implementiran ovaj obrazac tako što će se pomoću obaveštenja o izmeni nečega na sta je subscribe-ovan korisnik , odnosno svi korisnici koji su subscribe-ovani. Ovaj obrazac će se u našoj aplikaciji oslanjati na MB(Message Broker) koji služi za prenos poruka od “proizvođača” do pretplatnika.



5. Generalna arhitektura

Na slici ispod je prikazana generalna arhitektura projekta:



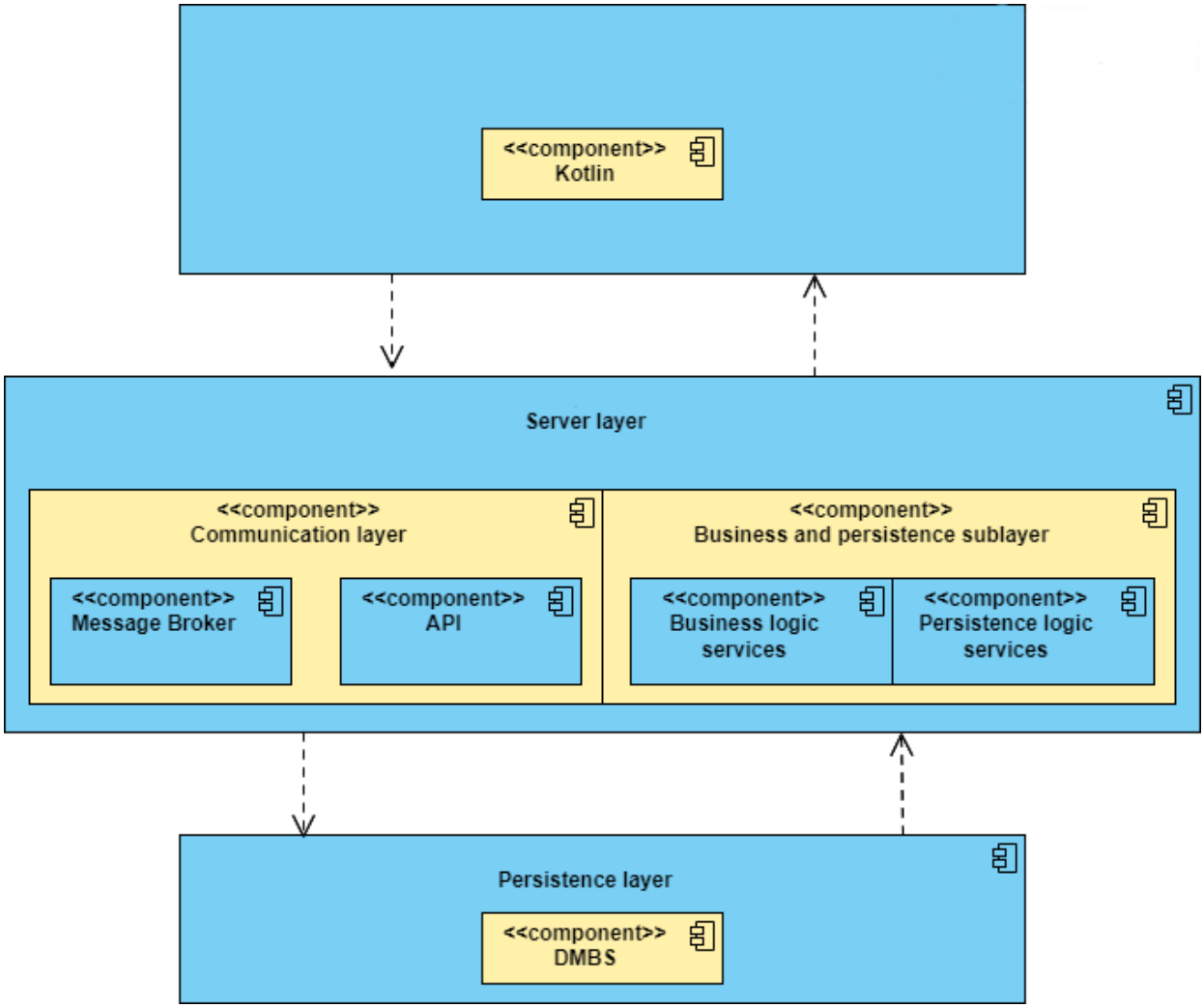
6. Strukturni pogledi

Dijagram prikazuje strukturu komponenti sistema i njihovu povezanost.

Klijentski sloj, koji predstavlja deo aplikacije sa kojim korisnici interaguju, se sastoji od activity-a i fragmenata, koji su zaduženi za prikaz, dok je drugi deo zadužen za komunikaciju sa serverom. Serverski sloj se sastoji od komunikacionog podsloja i podsloja za logiku aplikacije tj. biznis logiku sa podslojem perzistencije.

Komunikacioni podsloj obuhvata RESTful Web API za sinhronu komunikaciju i Message Broker za asinhronu komunikaciju sa klijentom.

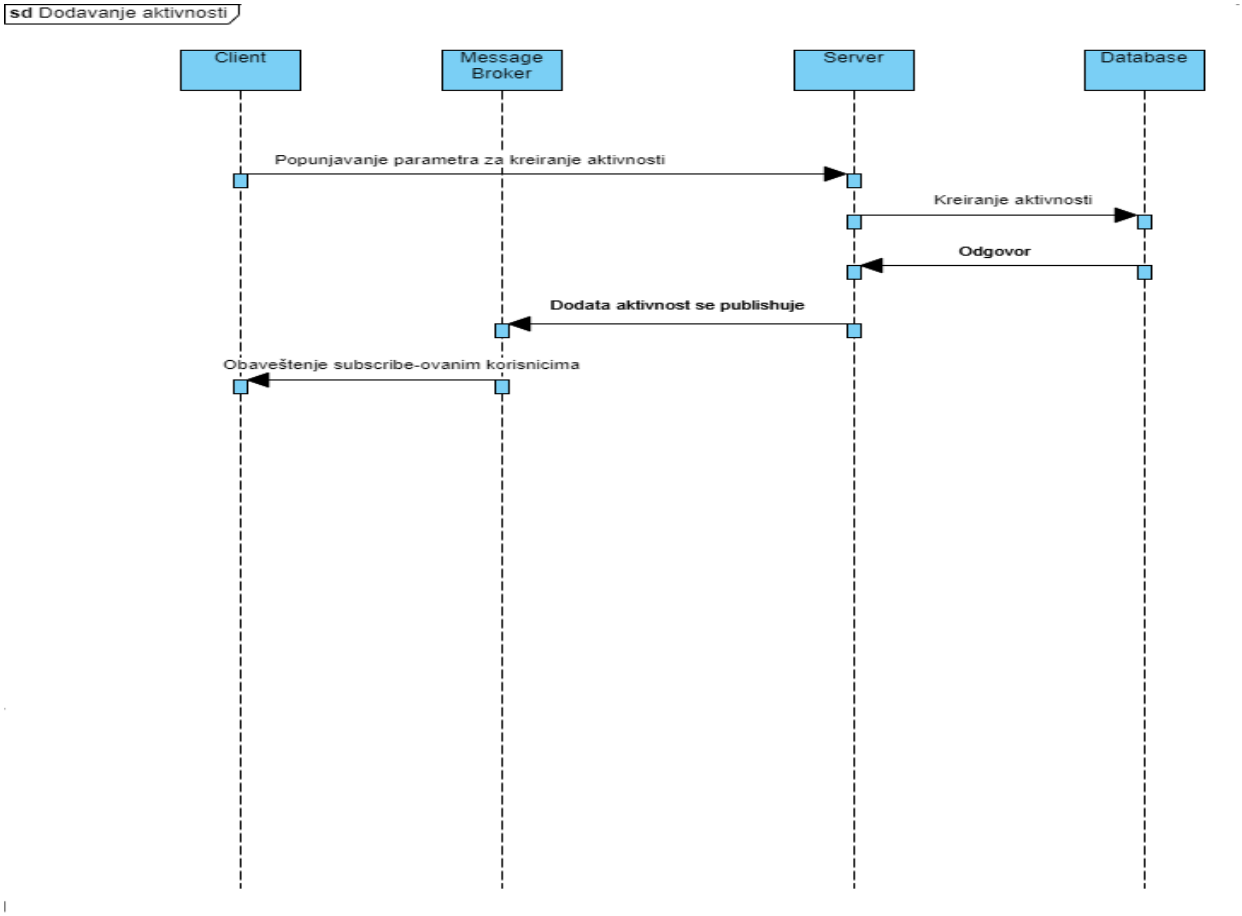
Na sloju perzistencije se nalazi DBMS kao konekcija sa bazom podataka.



7. Bihevioralni pogledi

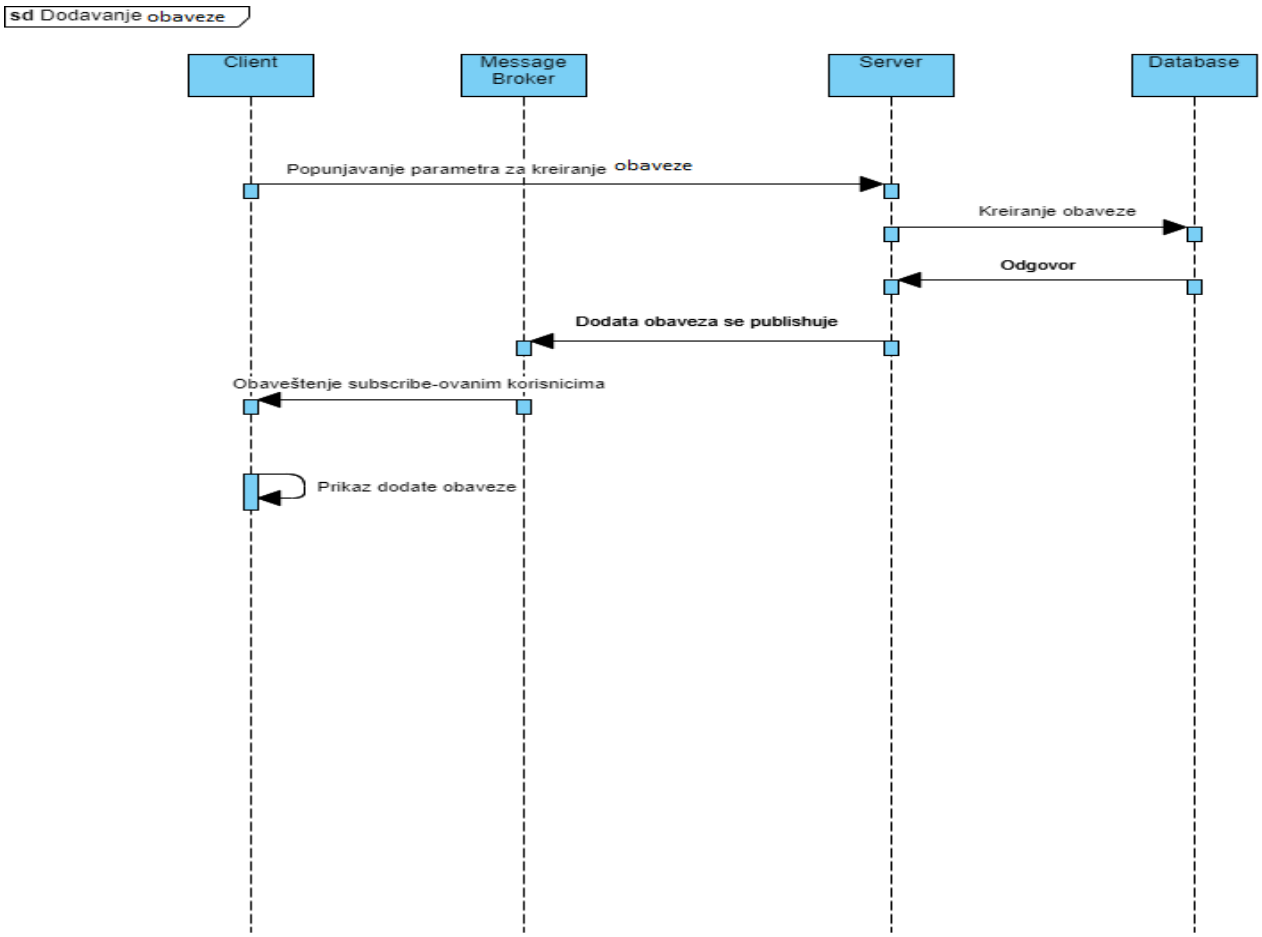
1. Kreiranje nove aktivnosti unutar kalendara

Sekvencijalni dijagram koji prikazuje kreiranje aktivnosti



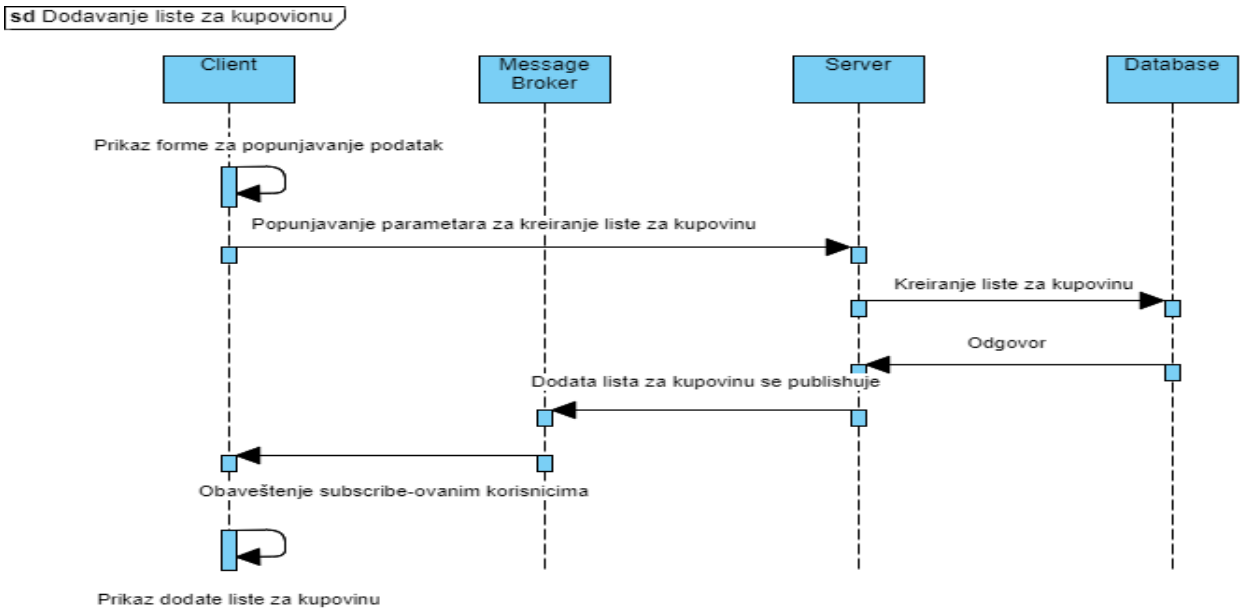
2. Kreiranje nove obaveze unutar grupe

Sekvencijalni dijagram koji prikazuje kreiranje nove obaveze



3. Kreiranje nove liste za kupovinu unutar grupe

Sekvencijalni dijagram koji prikazuje kreiranje nove liste za kupovinu



8. Implementacija projekta

Ovde su definisane tehnologije i okviri koji će se koristiti u izradi projekta:

- Kotlin – klijentska strana aplikacije, zato što se radi o native android aplikaciji
- Node.js + Express.js - backend aplikacije
- MongoDB – baza podataka
- Mongoose – ODM (Object Document Modeling)
- RabbitMQ - Message Broker