Arhitektura i projektovanje softvera

Faza 1

eRebel

Arhitektura projekta

Student:
Stefan Prvulović,
16297

Predmetni profesor:
Dragan Stojanović

Sadržaj

1. Kontekst i cilj projekta

eRebel predstavlja veb-aplikaciju za međusobnu komunikaciju između različitih planeta u galaksiji koje su deo Pobunjeničkog saveza. Aplikacija je inspirisana franšizom „Ratovi zvezda“ (engl. *Star Wars*) i u osnovi predstavlja aplikaciju za komunikaciju korisnika na različitim mestima i razmenjivanje međusobnih informacija o sopstvenom stanju nekih parametara, pri čemu se obaveštavaju svi ostali korisnici o stanjima i izmenama tih parametara. U sledećem pasusu je ilustrativno opisano kako bi komunikacija i izmena parametara bila izvršena.

Period je vladavine Galaktičke Imperije. Pobunjenički savez, sastavljen od nekolicine hrabrih boraca, pokušava na sve načine da sabotira autoritativnu i terorističku vladavinu Imperije. Da bi u tome uspeli, oni uspostavljaju sedišta na udaljenim planetama galaksije. Sedište Saveza može biti na jednoj ili na više planeta. Svako sedište vodi evidenciju o ljudima, brodovima, resursima (komunikatori, delovi brodova), zarobljenicima (imperijski vojnici na ispitivanjima), oružju i sl. Logovanje na aplikaciju je dozvoljeno vođama Pobunjeničkog saveza i logovanjem mogu unositi i brisati podatke o sopstvenim paramterima, kao i proveravanje podataka drugih sedišta na drugim planetama. Svako sedište ima opciju da pošalje poruku ukoliko im je potreban neki resurs ili su u opasnosti od razotkrivanja i svim korisnicima se šalje poruka i omogućava da pošalju određene resurse koji su potrebni. Prilikom pristizanja novih resursa, svako sedište dodaje u svoju bazu podataka resurse. Moguće je i kreirati sedište na novoj planeti od već postojećih raspoređujući resurse sa već postojećih planeta na novu.

Cilj aplikacije jeste funkcionalno vođenje evidencije svakog sedišta i obaveštavanje svih korisnika o izmenama u bazi podataka i slanje poruka o tim izmenama, kao i dodavanje ili brisanje parametara. U toku projektovanja će možda biti dodate još neke funkcionalnosti.

2. Arhitekturni zahtevi

eRebel predstavlja veb-aplikaciju koja poseduje bazu podataka koja služi za čuvanje, razmenu i deljenje informacija između više korisnika omogućavajući evidenciju pojedinačne baze svakog entiteta i time potpomaže u efikasnoj i strukturiranoj organizaciji više pojedinačnih sistema koji čine celinu.

Poseduje komponente koje se izvršavaju na računarima pojedinačnih korisnika, kao i sistem razmene poruka koji drugim korisnicima daje uvid u izmene pojedinačnih baza podataka. Server prikuplja podatke i može da ih filtrira na određeni način korisnicima.

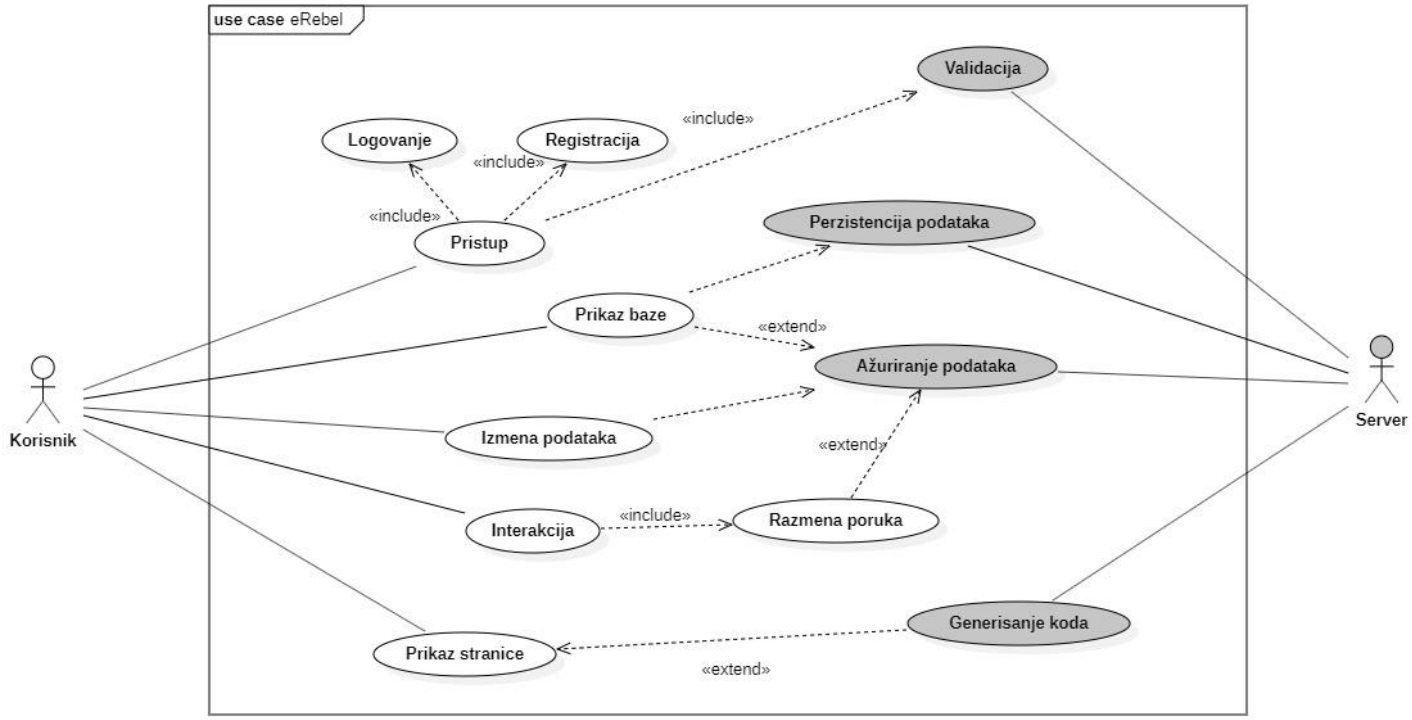
2.1. Glavni funkcionalni zahtevi

Funkcionalni zahtevi ove veb-aplikacije predstavljaju način na koji korisnici mogu da joj pristupe, manipulišu određenim podacima sopstvene baze i kako da interaguju sa drugim korisnicima. Interakcija je omogućena slanjem poruka i mogućnošću izmene ili ažuriranja podataka drugim korisnicima, nakon čega su svi ostali korisnici o tome obavešteni. Inicijalno, cilj aplikacije jeste komunikacija i efektivno evidentiranje podataka pojedinačnih korisnika, ali tokom projektovanja će možda biti dodate još neke funkcionalnosti.

Osnovni funkcionalni zahtevi ove veb-aplikacije su:

- **Registracija i/ili logovanje**: Korisnik ima pristupno korisničko ime i šifru. Registracija je omogućena samo ako za tim ima potrebe.
- **Prikaz sopstvene baze podataka**: Korisnik može da vidi svoju bazu podataka organizovanu pregledno i intuitivno gde lako može da proceni da li je potrebna izmena.
- **Interakcija sa drugim korisnicima**: Korisnik može da interaguje sa drugim korisnicima preko mehanizama definisanih u okviru aplikacije.
- **Razmena poruka**: Korisnici međusobno mogu razmenjivati poruke.
- **Perzistencija podataka**: Prilikom interakcije korisnika definisanim u okviru aplikacije gde je omogućena izmena podataka iz baza drugih korisnika, ti podaci moraju da budu ažurirani i odmah vidljivi i u bazi.

Na sledećoj slici prikazan je dijagram slučajeva korišćenja veb-aplikacije i način na koji korisnik svojim interakcijama utiče na serversku stranu.



Slika 2-1 Dijagram slučajeva korišćenja funkcionalnih zahteva

2.2. Nefunkcionalni zahtevi

Nefunkcionalni zahtevi definišu atribute sistema poput bezbednosti, performansa, održavanja, korišćenja i sl. Oni predstavljaju određena ograničenja u dizajnu sistema.

Nefunkcionalni zahtevi u okviru projekta ***eRebel*** su:

- **Funkcionalnost**: Sistem treba da bude funkcionalan i da na dobar način izvršava ono za šta je namenjen.
- **Kompatibilnost**: Sistem bi trebalo da bude multi-platformski, tj. da se veb-aplikacija bez problema izvršava na različitim platformama.
- **Pouzdanost**: U okviru pouzdanosti spada perzistencija podataka, tj. ako dođe do nekakvog prekida ili gubitka interneta u trenutku kada treba da dođe do promene podataka da korisnik može da nastavi gde je stao.
- **Održavanje**: Sistem treba da se blagovremeno analizira i održava, ukoliko se u korišćenju pronađu neke greške u funkcionalnosti, da se te greške isprave i učine proizvod dugoročnijim.
- **Bezbednost**: Sistem treba da čuva podatke u bazi, kao i podatke o korisnicima.
- **Upotrebljivost**: Aplikacija treba da bude laka za korišćenje, pregledna i da korisnicima na jednostavan način omogući ono čemu je namenjena.
- **Skalabilnost**: Ukoliko dođe do velike količine novih korisnika, potrebno je podržati taj rast i obezbediti da ne dođe do pada.

2.3. Tehnička i poslovna ograničenja

Tehnička ograničenja u okviru projekta su:

- **Komunikacija**: Aplikacija treba da podrži sinhronu (komunikacija između klijenta i servera) i asinhronu (razmena poruka između klijenata) komunikaciju.
- **Prikrivenost šeme baze podataka**: Korisnicima su vidljivi oni podaci koji su određeni za prikaz, a skriven je način na koji su ti podaci predstavljeni u bazi i povezani.
- **Pristup preko interneta**: Potrebno je da se aplikacija izvršava preko veba i stoga da se koriste veb-tehnologije koje omogućavaju nesmetano korišćenje aplikacije na vebu.

3. Arhitekturni dizajn

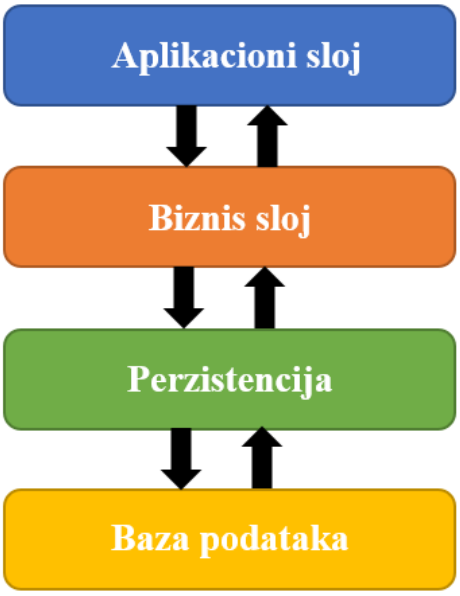
3.1. Arhitekturni obrasci

U nastavku će biti predstavljeni obrasci koji će biti korišćeni u toku realizacije veb-aplikacije ***eRebel***. Pored ovih obrazaca, moguće je da se u toku implementacije inkorpiraju još neki ako se za njima ukaže potreba.

3.1.1. *Layered*

Aplikacija će implementirati *Layered* arhitekturni obrazac. Ovaj obrazac definiše da aplikacija ima više slojeva. Osnovni obrazac deli aplikaciju na tri sloja: prezentacija, aplikacija i skladištenje. U okviru ove aplikacijce, biće iskorišćen modifikovan obrazac sa dodatkom biznis sloja u okivru kog će se implementirati komunikacija između klijenata. Stoga, Layered obrazac za ovu aplikaciju imaće 4 sloja:

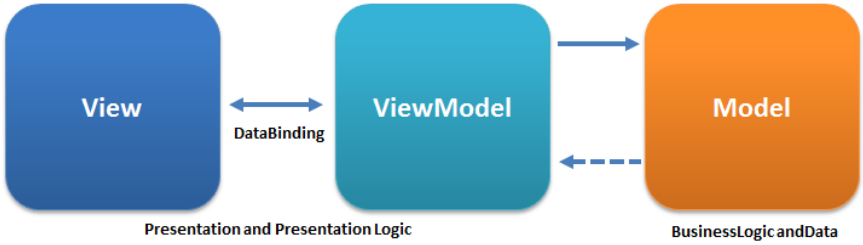
1. **Aplikacija**: Prikaz aplikacije klijentu preko kog je on koristi.
2. **Biznis**: Deo aplikacije preko kog se izvršava komunikacija između klijenata metodom poruka.
3. **Perzistencija**: ORM alati za mapiranje domenskih entiteta preko kojih serverska aplikacija ostvaruje komunikaciju sa bazom podataka.
4. **Baza podataka**: Sama baza podataka sa entitetima i vezama između njih.



Slika 3-1 *Layered* obrazac

3.1.2. *Model–view–viewmodel*

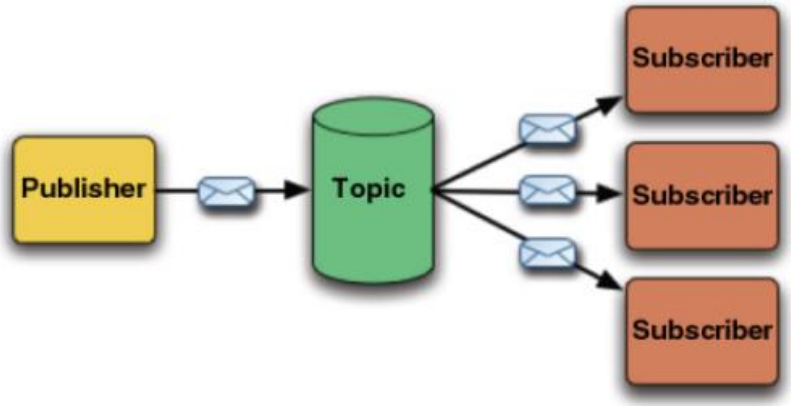
Model–view–viewmodel (MVVM) je arhitekturni obrazac koji predstavlja modifikaciju osnovnog obrasca *Model-view-controller* (MVC). U okviru aplikacije, ovaj obrazac će biti implementiran određenim tehnologijama tako što odvaja prikaz aplikacije korisniku od serverske strane koja omogućava prikaz određenih podataka.



Slika 3-2 MVVM obrazac

3.1.3. *Publish-subscribe*

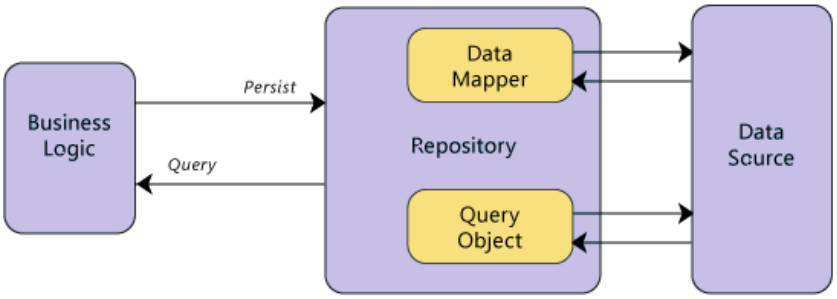
U okviru razmene poruka, biće implementiran Publish-subscribe obrazac tako što će se obaveštavanjem o izmeni jednog korisnika obaveštavati svi ostali korisnici. Omogućavanje slanja nekih posebnih poruka određenom korisniku, takođe se primenjuje pravilo ovog obrasca da svaki korisnik koji je „preplaćen“ na određenu temu, prima poruke o toj temi.



Slika 3-3 *Publish-subsribe* obrazac

3.1.4. *Repository*

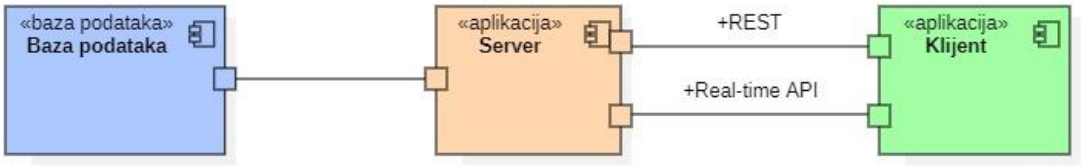
Repository obrazac će se ogledati u načinu na koji se podaci prikazuju korisniku i na koji način je predstavljena baza. Korisnik će preko prikaza aplikacije moći da menja svoju ili bazu drugih korisnika, kao i dobijati pregled onih podataka koji su važni, ali će se mapiranje i izmene u bazi dešavati preko određenih ORM alata i taj proces korisnik ne vidi, već se njemu samo prikazuju izmene ako do njih dođe.



Slika 3-4 *Repository* obrazac

3.2. Generalna arhitektura

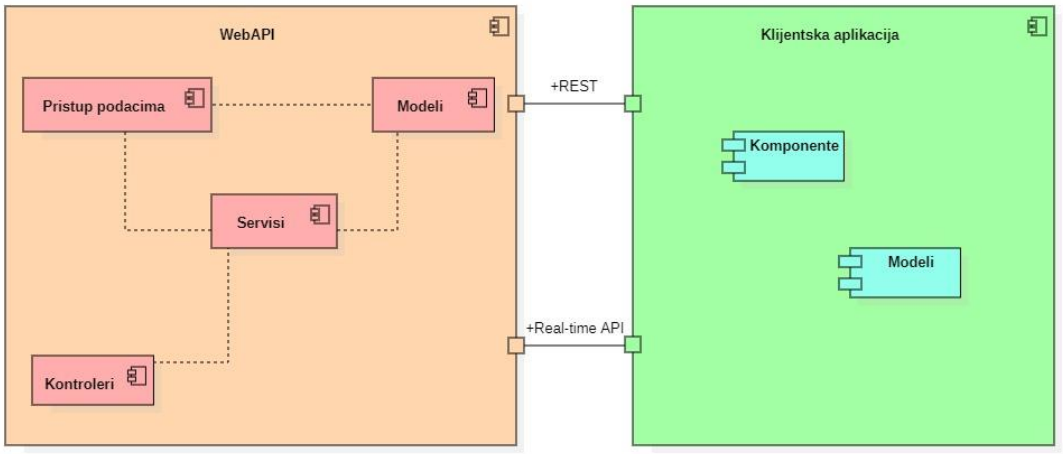
Generalna arhitektura predstavlja način na koji će aplikacija biti realizovana. U okviru aplikacije biće tri glavna dela arhitektura: klijent, server i baza podataka. Komunikacija između klijenta i servera biće omogućena slanjem HTTP zahteva i poštujući pravila REST servisa, tj. biće RESTful. Drugi način komunikacije biće razmena poruka koja će biti omogućena nekom od biblioteka pogodnoj za tehnologije kojima će projekat biti realizovan. Veza između servera i baze podataka će biti realizovana preko određenih obrazaca i time omogućiti perzistentnost podataka.



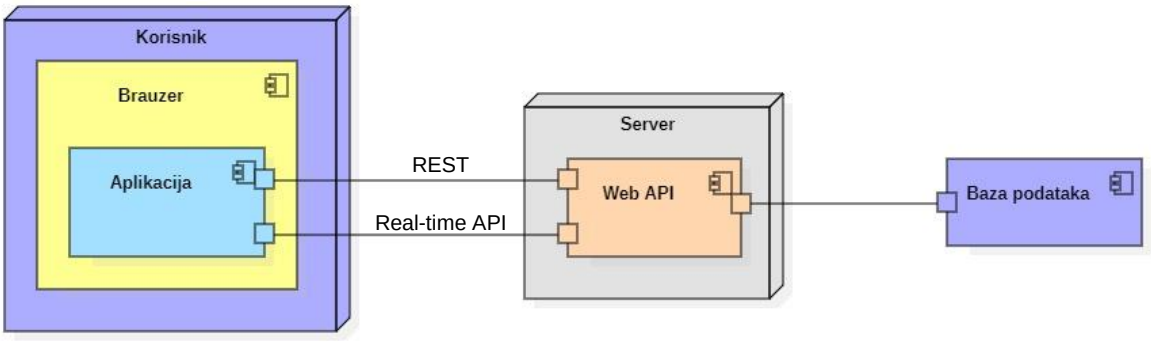
Slika 3-5 Generalna arhitektura

3.3. Strukturni pogledi

Slika 3-6 predstavlja reprezentaciju Web API-ja i klijentske aplikacije sa odgovarajućim delovima. Web API, koji predstavlja deo servera, sa klijentskom aplikacijom komunicira preko REST servisa i razmenom poruka preko odgovarajućeg API-ja predodređenog za to. U okviru Web API-ja se nalaze delovi koji su mapirani iz baze podataka u vidu modela, kontrolera i načina na koji se pristupa bazi, a u klijentskoj aplikaciji imamo modele podataka pogodne za prikaz, kao i sam prikaz stranice. Na slici 3-7 prikazan je dijagram raspoređivanja ova dva dela u arhitekturi celog sistema.



Slika 3-6 Strukturni pogled

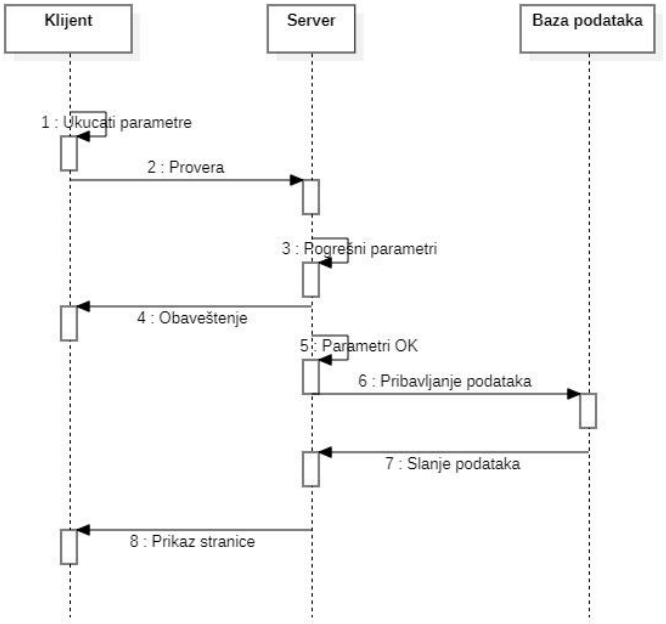


Slika 3-7 Dijagram raspoređivanja

3.4. Bihevioralni pogledi

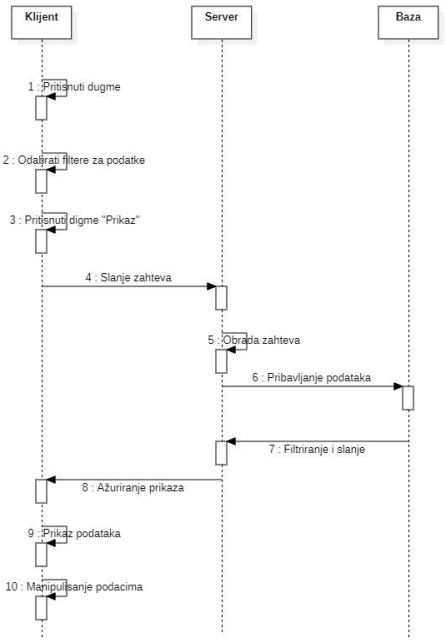
Bihevioralni pogledi bliže određuju arhitekturu prilikom izvršenja neke akcije. U nastavku će biti prikazani sekvencijalni dijagrami za osnovne akcije koje korisnik može da izvršava tokom korišćenja aplikacije. Tokom realizacije, možda će biti dodate akcije.

3.4.1. Prijava korisnika

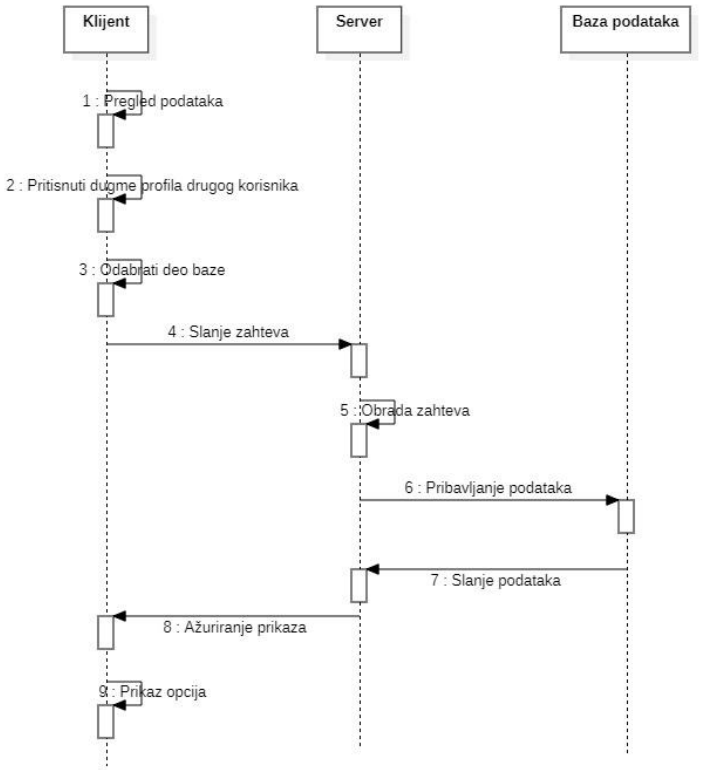


Slika 3-8 Sekvencijalni dijagram za prijavu korisnika

3.4.2. Pregled podataka



3.4.3. Interakcija s drugim korisnicima



3.5. Implementaciona pitanja

Biblioteke i okviri koji će se koristiti za realizaciju ovog projekta su:

- **Angular** – JavaScript okvir za pisanje front-end dela veb-aplikacije,
- **NestJS** – JavaScript okvir za pisanje back-end dela veb-aplikacije,
- **RabbitMQ** – softver koji omogućava real-time komunikaciju.

Ukoliko se pri inicijalnom projektovanju neki od ovih okvira ne pokažu pogodnim za realizaciju projekta, moguće je zameniti ih pogodnijim.

5. Analiza arhitekture

Neki od potencijalnih rizika u razvoju arhitekture ove aplikacije mogu biti:

- **Loše performanse u toku real-time komunikacije**: Neki korisnici mogu aplikaciju koristiti na sporim mrežama i stoga može doći do kašnjenja u komunikaciji između njih. Jako je teško osmisliti strategiju za rešavanje ovog problema jer većinom zavisi od mašina klijentskih korisnika.
- **Preopterećenost servera**: Ukoliko u određenom trenutku dođe do prevelikog broja korisnika. Ovo možemo rešiti testiranjem za vreme projektovanja aplikacije i s vremenom povećavati server i održavati ga da ne dođe do pada.