2026

Bomb Rush

ARHITEKTURNI DOKUMENT

STEFAN CVETKOVIĆ 19461 | LAZAR STANKOVIĆ 19375

Contents

Kontekst i cilj projekta

Bomb Rush je online video igra u kojoj se dva ili više igrača takmiče dok ne ostane samo jedan. U samoj sredini igre se nalazi bomba koja otkucava u kojoj se nalazi kombinacija slova, cilj igrača je da napiše reč koja će u sebi sadržati taj niz slova na bilo kojoj poziciji. Ideja je da se pri svakom dobrom odgovoru pređe na sledećeg igrača, restartuje tajmer na bombi i kombinacija slova na njoj. U suštini, svaki igrač ima dva života i odredjeno vreme da napiše odgovore. Bomba ide od igrača do igrača dok ne ostane samo jedan igrač.

Korisnici ove web aplikacije mogu da kreiraju svoje profile uz mogućnost dodavanja profilnih slika i prikaznog imena. Pored toga svaki korisnik može kreirate svoje sobe, privante ili javne, gde se mogu pridružiti drugi korisnici. Kreator sobe može praviti svoja podešavanja (težina slova, broj života, maksimalno i minimalno vreme tajmera bombe).

Arhitekturni zahtevi

Glavni funkcionalni zahtevi

- **Registracija i prijava korisnika**
- **Pridruživanje i kreiranje sobe (lobby-a)**
- **Podešavanja igre**
- **Chat u okviru svake sobe**
- **Igranje igre**

Ne-funkcionalni zahtevi

- **Pouzdanost** – obezbediti perzistenciju podataka
- **Dostupnost** – aplikacija mora uvek biti dostupna
- **Performanse** – potrebno je obezbediti što manje vreme odziva i što lakše izvršenje same igre s obzirom na to da je aplikacija koja radi u realnom vremenu.
- **Sigurnost** – potrebno je obezbediti autentifikaciju i autorizaciju podataka
- **Skalabilnost** – podržati rast broja koristnika
- **Modifikabilnost** – obezbediti lako dodavanje novih funkcionalnosti i izmenu već postojećih
- **Upotrebljivost** – aplikacija treba da bude intuitivna i laka za korišćenje

Tehnička i poslovna ograničenja

Jedno od glavnih tehničkih ograničenja je upotreba Web browser-a za pristup aplikaciji. Potrebno je obezbediti dobru autorizaciju i autentifikaciju kako bi svako video ono što treba da vidi. Takođe je potrebno „naterati“ korisnika da ostane na samom sajtu. S obzirom na to da se aplikaciji pristupa preko web browser-a korisnik može lako da pređe na nešto drugo. Pored ovoga moramo i obezbediti konstantu komunikaciji između korisnika i samog message broker-a. SQL server je primarno napravljen za Windows operativni sistem i zbog toga postoji šansa da dođe do malih problema ako se odlučimo da koristimo Linux server. S obzirom na to da naša web aplikacija koristi React.JS potrebno je da korisnik ima uključen JavaScript kako bi mogao da koristi našu aplikaciju.

Arhitekturni dizajn

Arhitekturni obrasci

Layered obrazac

Aplikacija će imati troslojnu klijent-server arhitekturu. Prezentacioni sloj aplikacije će se izvršavati na samom klijentu. Preko prezentacionog sloja će klijent vršiti interakciju sa samom aplikaciju i slati/primati podatke. Prezentacioni sloj će samo komunicirati sa serverskim slojem, dok će serverski sloj komunicirati sa slojem perzistencije podataka.

Serverski sloj ima ulogu manipulacije podacima i čuvanjem/čitanjem iz sloja perzistencije. U ovom sloju će se izvršavati poslovna logika igre, kreiraće se kombinacija slova koja će se slati igračima i proveravati ispravnost odgovora igrača. Na ovom sloju će se takođe nalaziti Message broker koji će biti implementiran pomoću SignalR-a. Na ovom sloju će se čuvati kolokcija svih mogućih reči kao HashSet.

Sloj perzistencije će biti sama baza podataka u kojoj će se skladištiti podaci o korisnicima i sobama koje korisnici kreiraju.

Repository

Aplikacija će koristiti ovaj obrazac za komunikaciju sa bazom podataka.

Publish-Subscribe

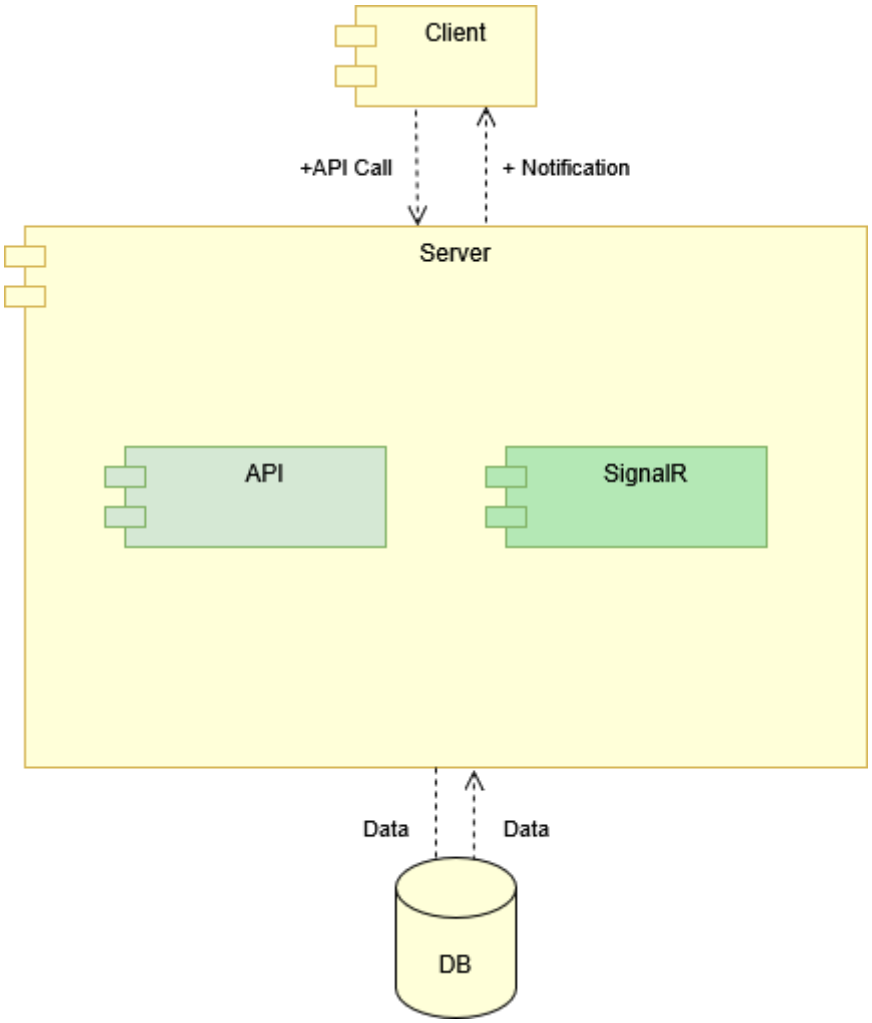
Aplikacija će koristiti ovaj obrazac za komunikaciju sa podacim u realnom vremenu. Biće implementiran kao SignalR koji će u realnom vremenu komunicirati sa klijent stranom aplikacije. On će igraču vraćati trenutnu reč i ostalim igračima slati pokušaje drugih igrača. Svaki igrač će se pretplatitit na Lobby koji kreira osnivač sobe i preko toga dobijati poruke koje su za njega i slati poruke za druge.

Model-View-Controller

MVC razdvaja komponenta sistema aplikacije radi lakšeg skaliranja i održavanja. Model ima ulogu predstavljanja podataka i logike poslovanja aplikacije. On komunicira sa bazom i nema uvid u to za šta se ti podaci koriste. View je taj koji igraču prikazuje podatke iz modela, on ne sadrži nikakvu poslovnu logiku. Controller obrađuje podatke koje šalje View i vrši interakciju sa modelom. On međusobno ažurira View i Model.

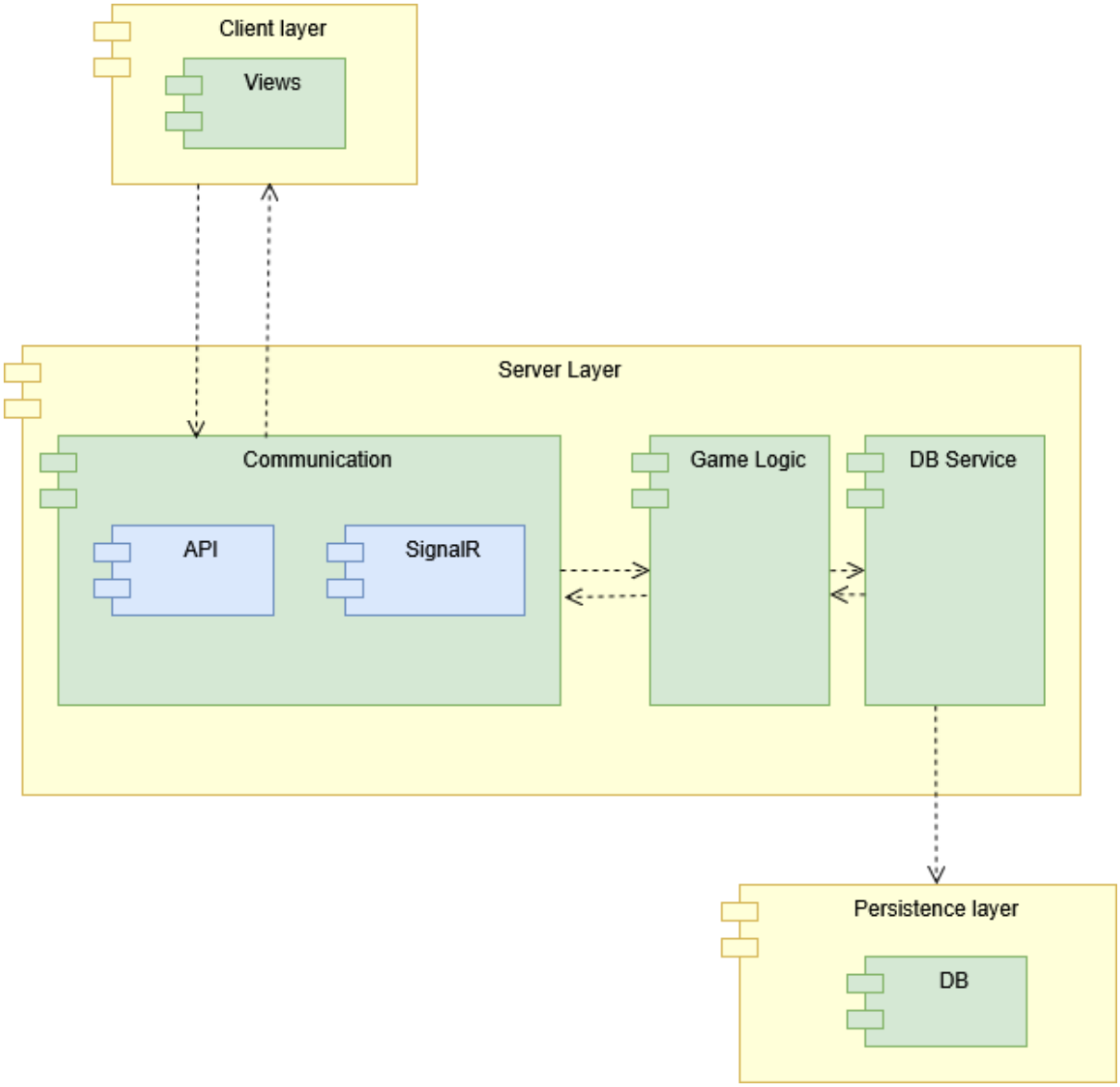
Generalna arhitektura

Generalna arhitektura sistema aplikacije podrazumeva postojanje klijenta, servera i baze podataka. Klijent salje zahteve serveru, server čita ili upisuje podatke u bazu i vraća rezultat nazad klijentu. Pored toga server sadrži SignalR koji je zadužen za komunikaciju između klijenta i servera u realnom vremenu.



Strukturni dijagram

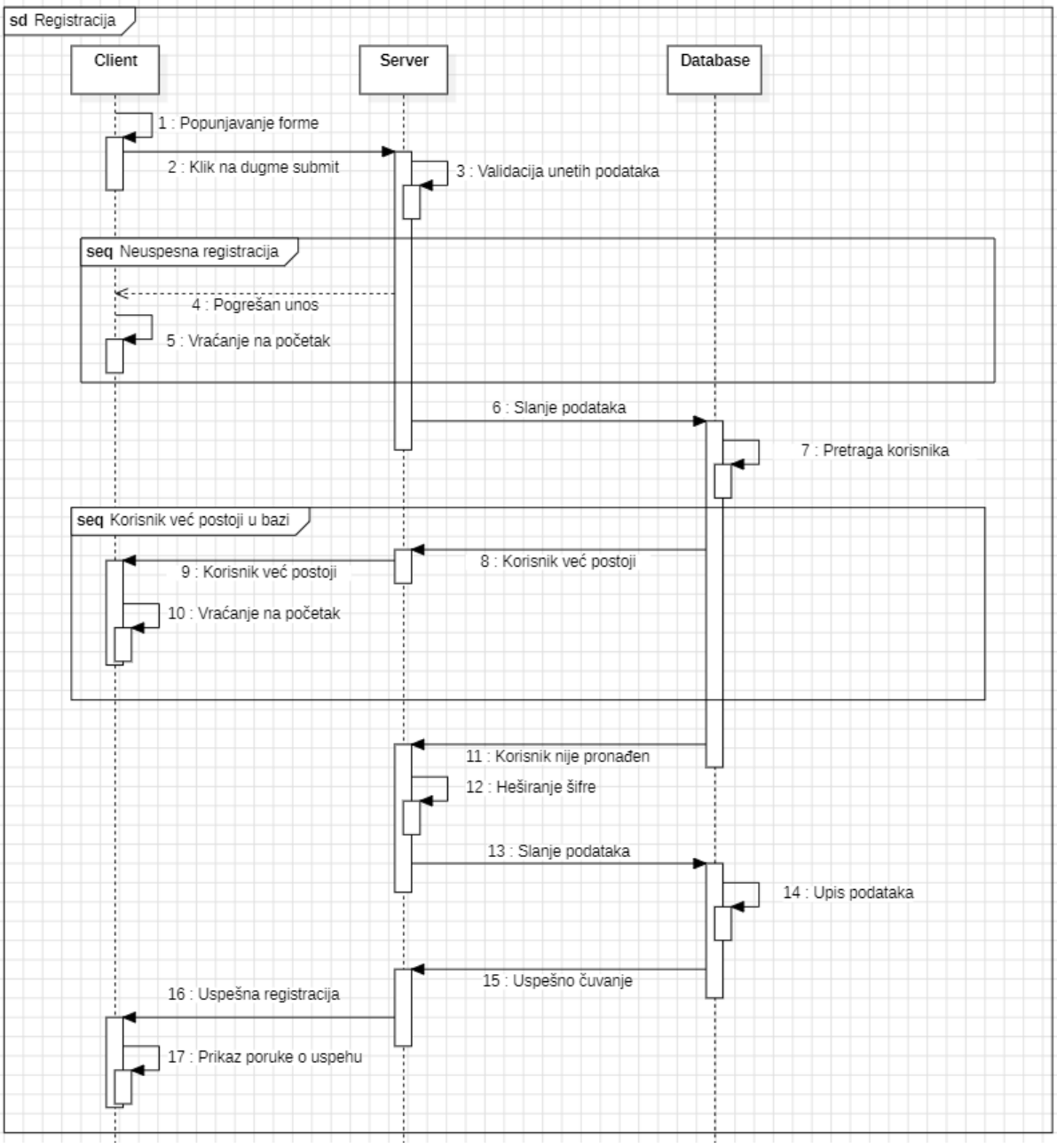
Aplikacija je podeljena u tri sloja. Klijentski sloj komunicira sa Serverom koristeći komunikacioni sloj koji se sastoji iz API-a i Message broker-a (SignalR). Server to obrađuje i pomoću servisa baze pristupa sloju perzistencije gde se nalazi baza podataka.



Bihevioralni dijagram

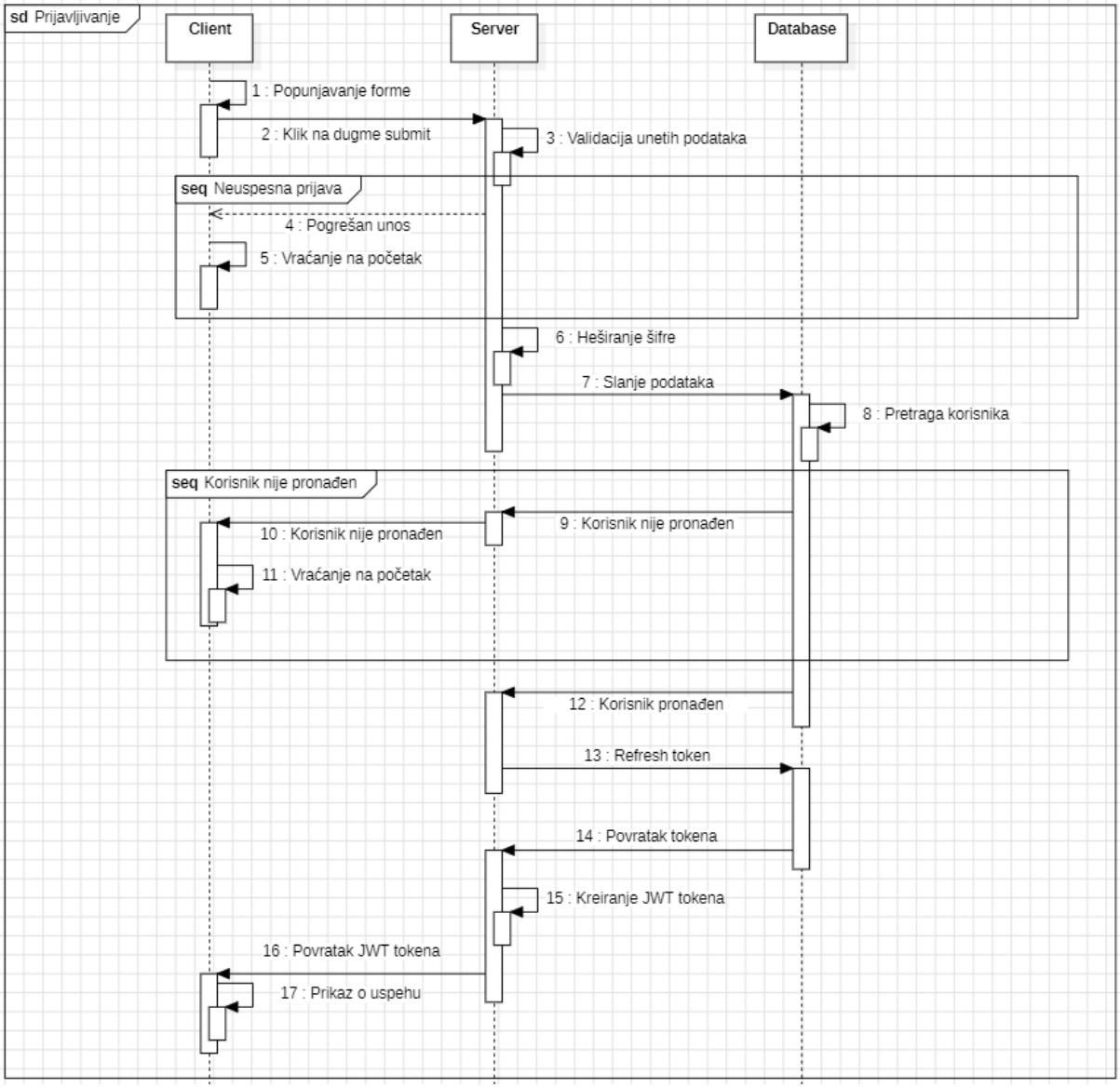
Registracija

Registracija korisnika se vrši popunjavanjem forme za registraciju. Nakon popunjavanja korisnink šalje zahtev serveru gde se vrši validacija unetih podataka. Server proverava validnost parametara pre samog poziva baze. Ukoliko je neki od parametara pogrešnog formata server vraća grešku klijentu. Ukoliko je sve okej server vrši pretragu nad bazom. Ukoliko postoji korisnik sa unetim username-om ili mailom vraća se greška korisniku. U slučaju da korisnik ne postoji, server hešira šifru i heširanu šifru šalje bazi uz sve ostale podatke. Baza skladišti podatke i korisniku se vraća poruka o uspehu.



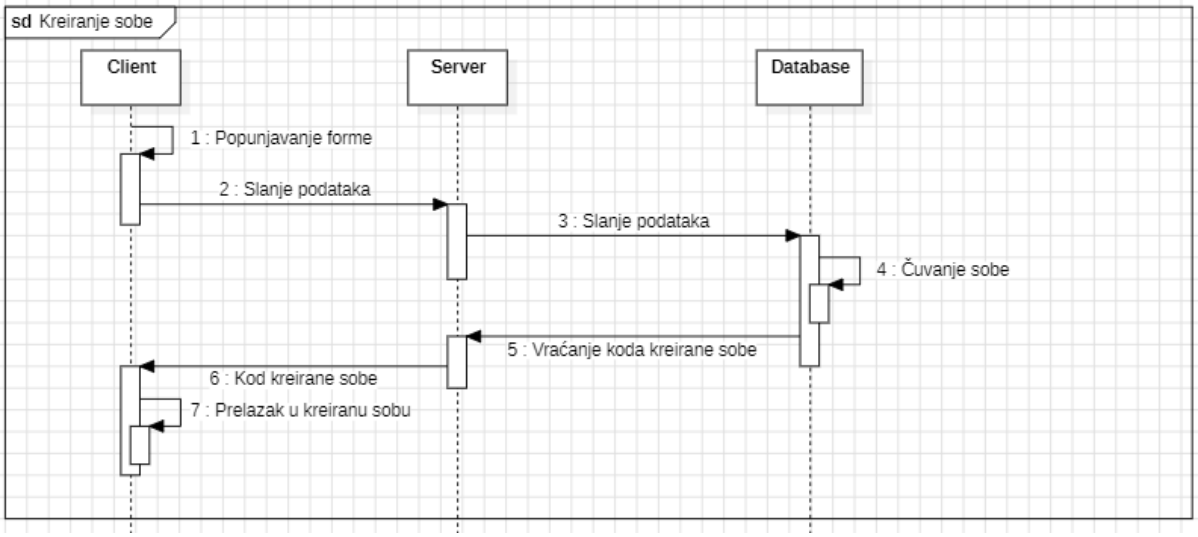
Prijavljivanje

Prijavljivanje se vrši popunjavanjem forme. Nakon popunjene forme se serveru šalju uneti podaci gde se proverava validnost unetih parametara. Ukoliko parametri nisu dobri korisniku se vrati informacija o lošem unosu. Ukoliko su parametri dobri korisnik se pretražuje u bazi pomoću heširane unete šifre i korisničkog imena. Ukoliko korisnik nije pronađen vraća se greška. U slučaju kada je pronađen korisnik, kreira se i čuva u bazi refresh token koji će se koristiti za sledeću prijavu i server kreira JWT token preko kojeg se vrši validacija sesije prijaveljnog korisnika. Korisniku se vraća token koji on koristi za autentikaciju.



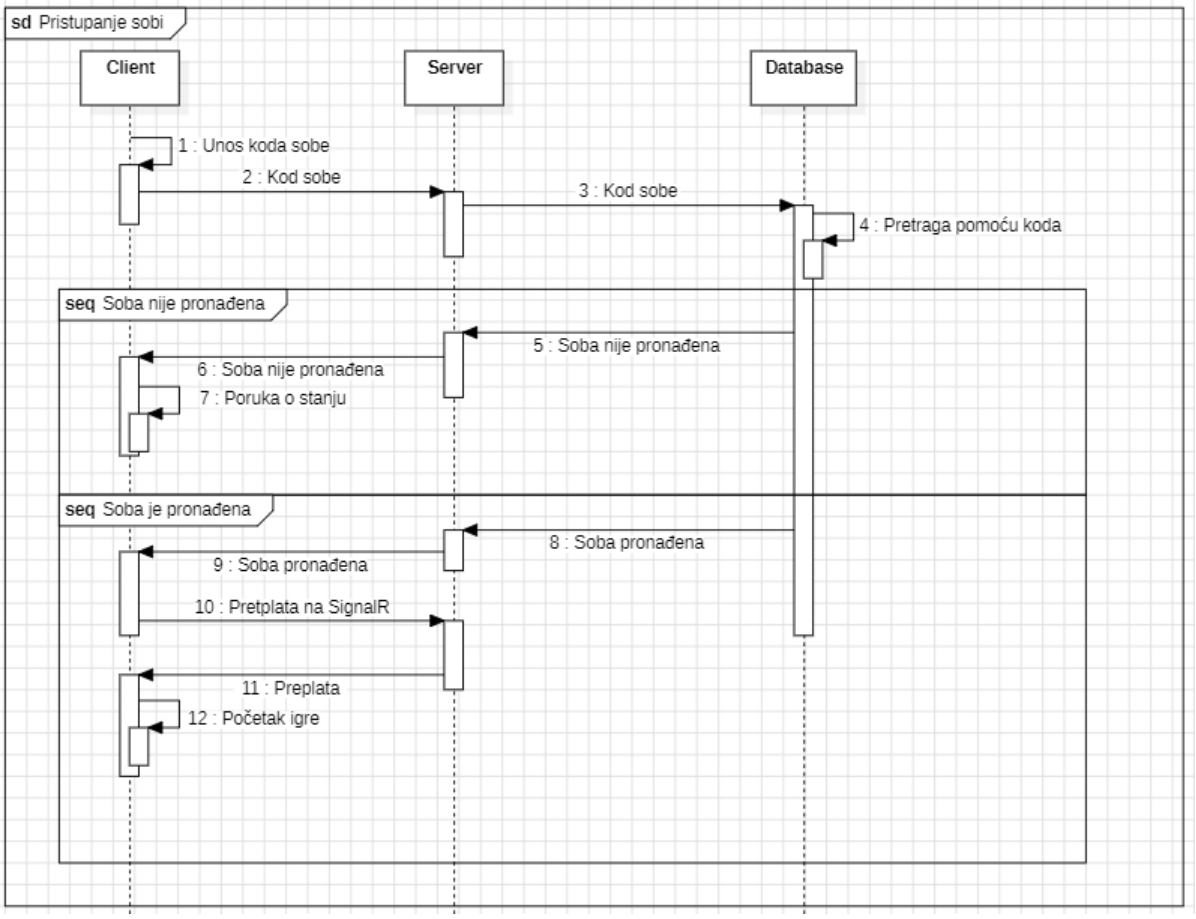
Kreiranje sobe

Igrač serveru šalje zahtev za kreiranje sobe. Server čuva novu sobu u bazi i korisniku vraća kod koji drugi igrači mogu da koriste za pristup samoj sobi.



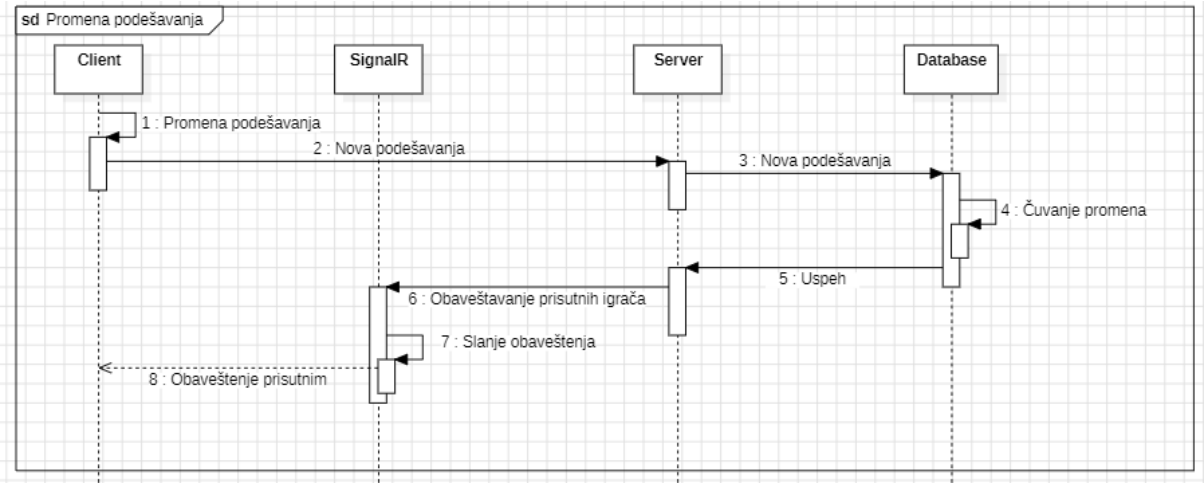
Pristupanje sobi

Igrači prisutpaju sobi korišćenjem koda sobe.



Promene podešavanja

Prilikom promene podešavanja sobe, potrebno je obavestiti sve prisutne korisnike o tome.



Aplikaconi okviri

Biblioteke koje će se koristiti za realizaciju aplikacije su:

- **React.JS** – React će se koristiti kao View sloj same aplikacije. Koristiće se JavaScript verzija react-a.
- **.NET API** – Za serverski deo će se koristiti .NET API koji će koristiti JWT za autorizaciju korisnika. Koristiće Entity Core za mapiranje sa bazom podataka.
- **SignalR** – Koristi se za komunikaciju u realnom vremenu između klijenta i servera.
- **SQL Server** – Koristi se za skladištenje podataka o igračima i o samim sobama koje oni kreiraju.