TrešKviz

Arhitekturni projekat

Aleksa Stančev 17434
Lazar Ignjatović 17124

Sadržaj

1. Kontekst i cilj projekta

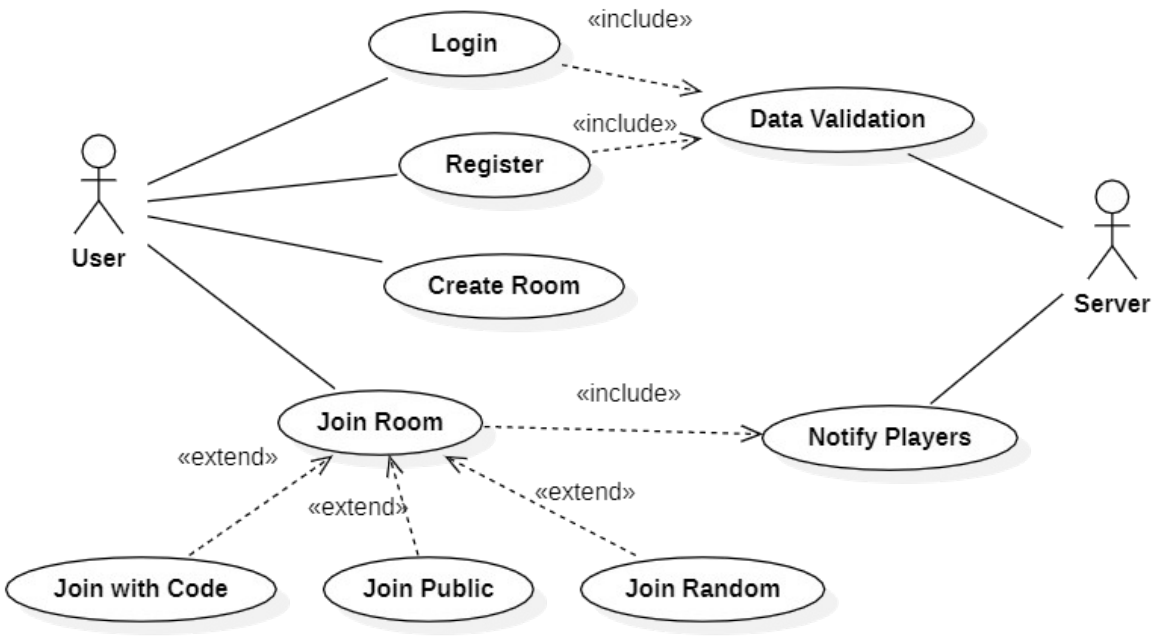
Treš kviz je Web aplikacija namenjena za online multiplayer igranje kviza u kome se boduje znanje popularne kulture. Aplikacija obezbeđuje igranje kviza u realnom vremenu. Igrači imaju mogućnost kreiranja sobe ili pridruživanja postojećoj sobi. Mogu se pridružiti odabranoj javnoj sobi, nasumičnoj javnoj sobi, ili privatnoj sobi putem koda sobe. Igrač koji kreira sobu je Host i on ima administratorske mogućnosti nad sobom. Host može odabrati igre koje se igraju, pokrenuti partiju, promeniti tip sobe (privatna/javna) i može ukloniti igrača iz sobe. Po pokretanju partije igrači ulaze u prvu igru koju je odabrao Host, nakon koje se igra druga igra i tako do poslednje. Svaka igra se igra i boduje u skladu sa svojim pravilima (igranje po potezu ili najbrži odgovor). Nakon završene poslednje igre proglašava se pobednik na osnovu osvojenih poena i igra se završava.

2. Arhitekturni zahtevi

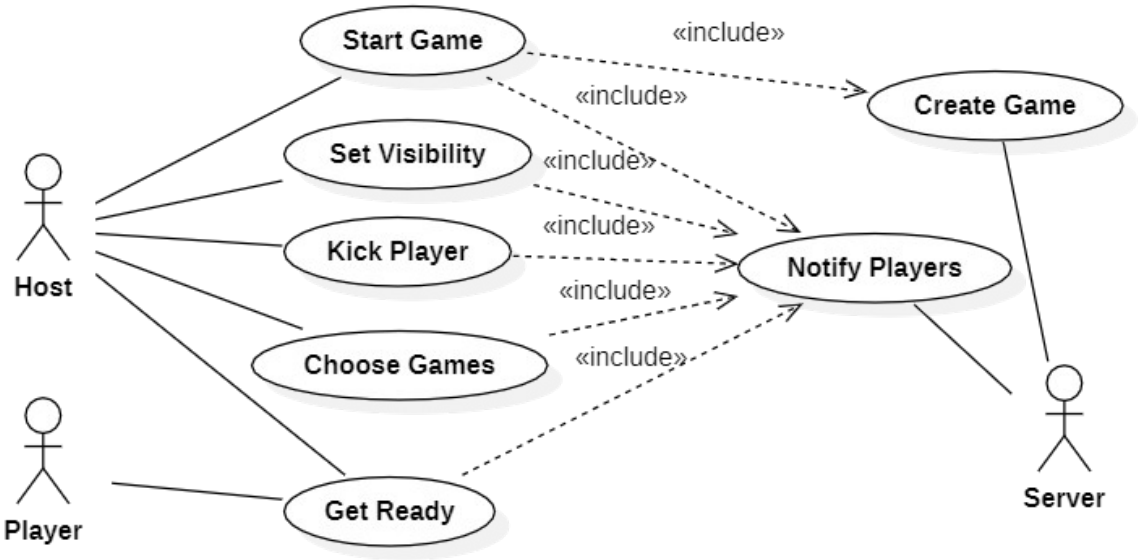
U ovom odeljku su prikazani arhitekturni zahtevi TrešKviz sistema, koji obuhvataju arhitekturno značajne slučajeve korišćenja, nefunkcionalne zahteve i tehnička i poslovna ograničenja.

2.1. Arhitekturno značajni slučajevi korišćenja

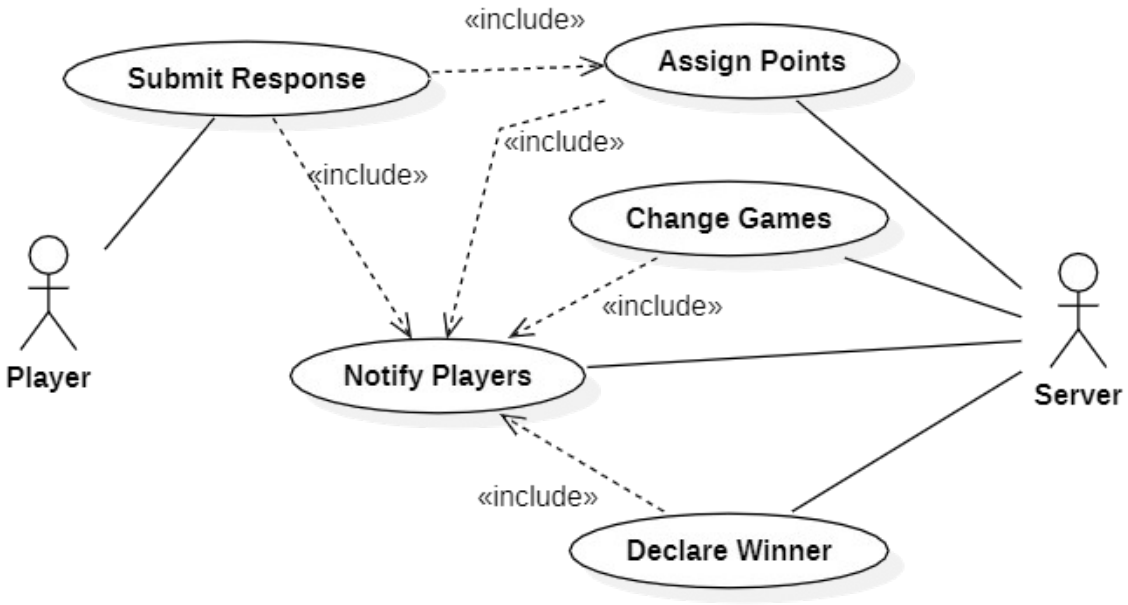
- Registracija i prijava korisnika
- Pridruživanje sobi (javnoj, javnoj nasumično, privatnoj putem koda)
- Kreiranje sobe
- Označavanje spremnosti za početak partije
- Odabir vidljivosti sobe
- Odabir igrara
- Uklanjanje igrača iz sobe
- Pokretanje partije
- Prikaz stanja partije
- Prikaz pitanja
- Slanje odgovora na pitanja
- Prelazak na sledeću igru
- Proglašenje pobednika
- Skladištenje podataka (korisničkih podataka i statistika, kao i pitanja)



Slika 1: Use Case dijagram unutar Lobija



Slika 2: Use Case dijagram unutar Sobe



Slika 3: Use Case dijagram unutar Partije

2.2. Nefunkcionalni zahtevi

- Dostupnost – potrebno je da aplikacija bude dostupna 24/7
- Skalabilnost – potrebno je da aplikacija može da podrži povećanje broja korisnika
- Performanse – aplikacija treba da obezbedi što manje vreme odziva i najbolje performanse u zavisnosti od trenutnog broja korisnika
- Izmenljivost - potrebno je omogućiti relativno laku promenu sistema
- Pouzdanost – sistem treba da omogući perzistenciju na početku/kraju partije
- Sigurnost – Obezbediti autentifikaciju i autorizaciju, kao i enkripciju osetljivih podataka
- Upotrebljivost – potrebno je da aplikacija bude intuitivna i jednostavna za korišćenje

2.3. Tehnička i poslovna ograničenja

• Pristup preko web-a – Neophodno je koristiti web tehnologije koje omogućavaju potrebnu komunikaciju i interakciju između sistema i korisnika

• Komunikacija – Sistem treba da podrži asinhronu komunikaciju između servera i klijenata

• Skrivenost baze podataka – Korisnicima su dostupni samo podaci predviđeni za prikaz, a od njih je sakriven način reprezentacije tih podataka u bazi

Poslovna ograničenja sistema baziraju se na pravilima samih igara i od njih zavisi koje će akcije korisnik moći da obavi u datom trenutku. Takođe, jedno od ograničenja je i budžet samog projekta koji ne dozvoljava iznajmljivanje jačeg hardvera kojim bi se omogućio istovremeni pristup sistemu velikom broju korisnika.

3. Arhitekturni dizajn

U ovom poglavlju arhitekturni dizajn prikazan je kroz arhitekturne obrasce, generalnu strukturu arhitekture, strukturne poglede, bihevioralne poglede i implementaciona pitanja.

3.1. Arhitekturni obrasci

3.1.1. Layered pattern

TrešKviz sistem implementiraće troslojni Layered arhitekturni obrazac. Prvi sloj predstavlja klijentska aplikacija, drugi serverska aplikacija koja sadrži Message Broker-a, a treći sloj je sloj perzistencije podataka. Osim glavnih i serverski i klijentski sloj su dalje organizovani po slojevima. Klijentski deo sadrži slojeve za prezentaciju, logiku i komunikaciju, dok se serverski deo sastoji od slojeva za komunikaciju, logiku i perzistenciju.

3.1.2. Publisher subscriber

Publisher subscriber arhitekturni obrazac će biti implementiran kroz Message Broker koji predstavlja sponu između klijenta i servera. Igrači su automatski subscribe-ovani na odgovarajuće lobby/room/game hub-ove u zavisnosti od toga koji deo aplikacije trenutno koriste. Sva komunikacija se odvija preko Broker-a čime se obezbeđuje vidljivost svih promena u lobiju/sobi/partiji u realnom vremenu.

3.1.3. Repository

TrešKviz sistem sadržaće centralizovanu bazu podataka. Pristup bazi odvijaće se preko Persistence sloja koji će implementirati Repository obrazac i komunicirati sa bazom.

3.1.4. MVVM

Klijentski deo TrešKviz sistema biće realizovan korišćenjem MVVM (Model-View-ViewModel) obrasca, koji Angular framework implicitno realizuje. Komponente komunikacionog i Game Logic sloja klijentske aplikacije (servisi za komunikaciju sa serverom i obradu podataka) predstavljaju modele (Model), jedan deo podkomponenata komponenti prezentacionog sloja (HTML i CSS delovi Angular komponenti) predstavlja poglede (View), dok ostatak (TS skripte) predstavlja sponu između modela i pogleda (ViewModel).

3.1.5. Strategy

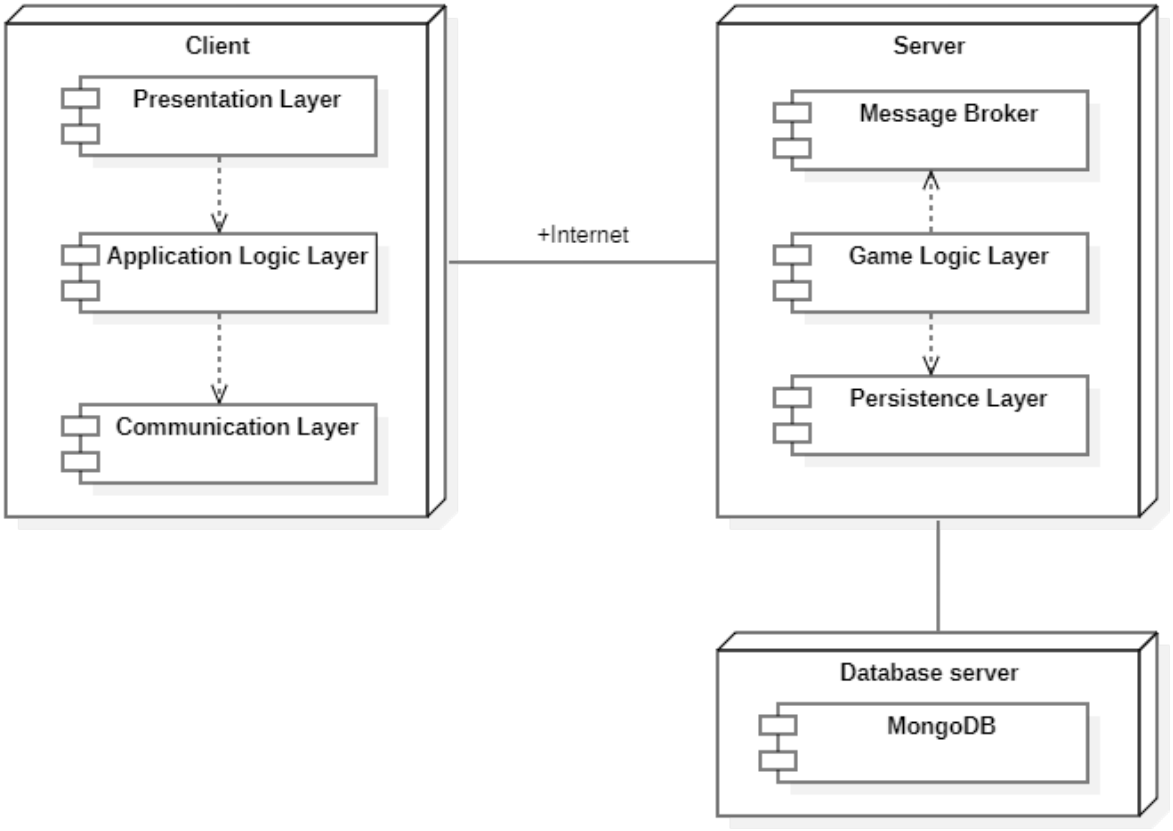
Za obezbeđivanje modularnosti kviza biće iskorišćen Strategy obrazac. Svaka igra ima zajedničke osobine kao što su početak, kraj, slanje odgovora i ažuriranje prikaza stanja igre, ali se takođe odvija po svojim nezavisnim pravilima. Zbog ovih osobina pogodna je i poželjna implementacija Strategy obrasca koji će specifičnosti svake igre izdvojiti u posebne klase.

3.1.6. Singleton

TrešKviz iskoristiće Singleton obrazac za implementaciju RoomMaster i GameMaster klasa. Ove klase sadrže jedinstvene liste soba koje su trenutno aktivne i aktivnih partija respektivno. Radi obezbeđivanja ove jedinstvenosti i dostupnosti potrebno je ove klase realizovati kao Singleton klase.

3.2. Generalna arhitektura

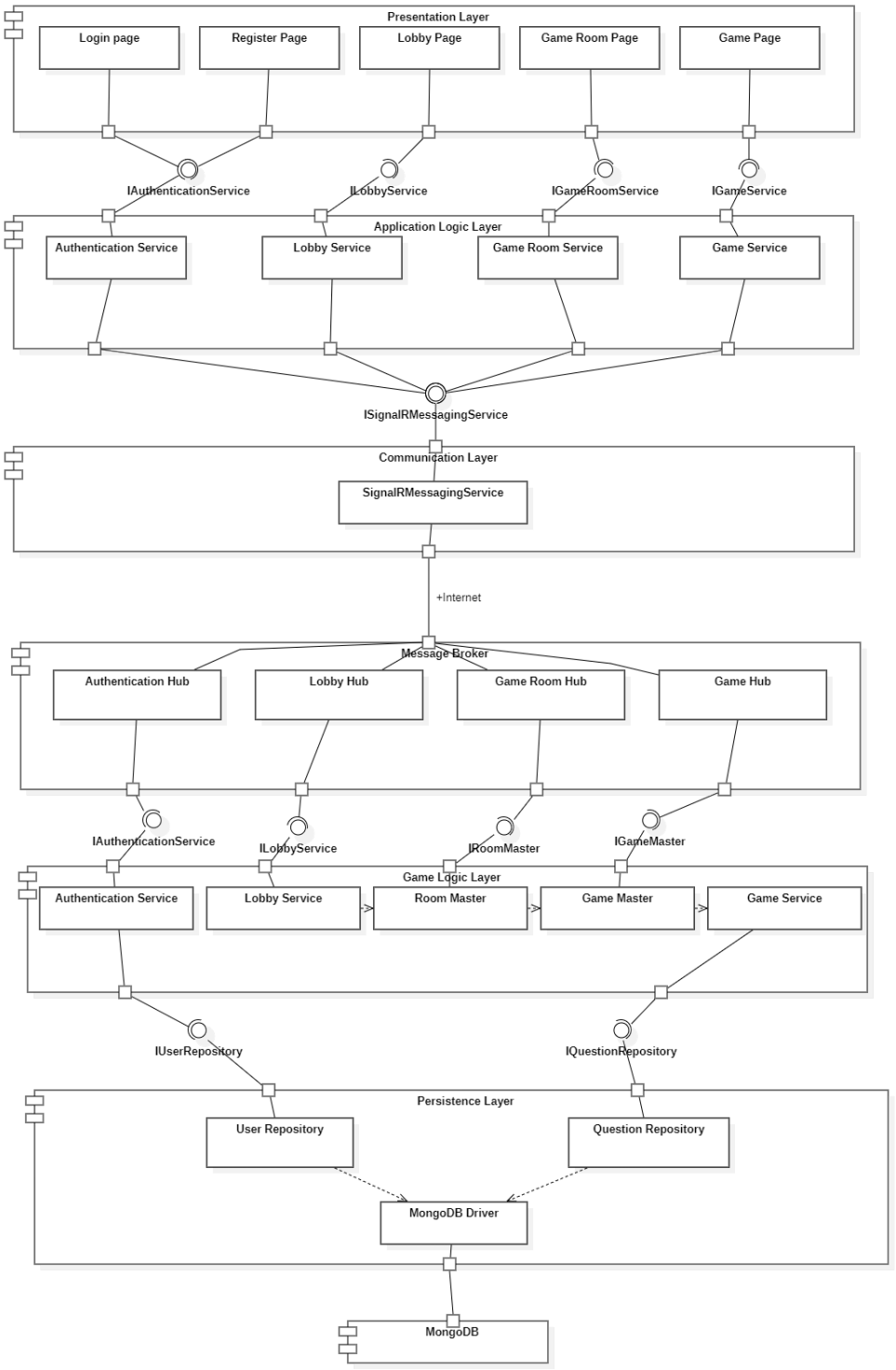
Arhitektura sistema podrazumeva postojanje klijenta, servera i baze podataka u kojoj će se čuvati informacije o korisnicima i njihovim igrama.



Slika 4: Generalna arhitektura sistema

3.3. Strukturni pogledi

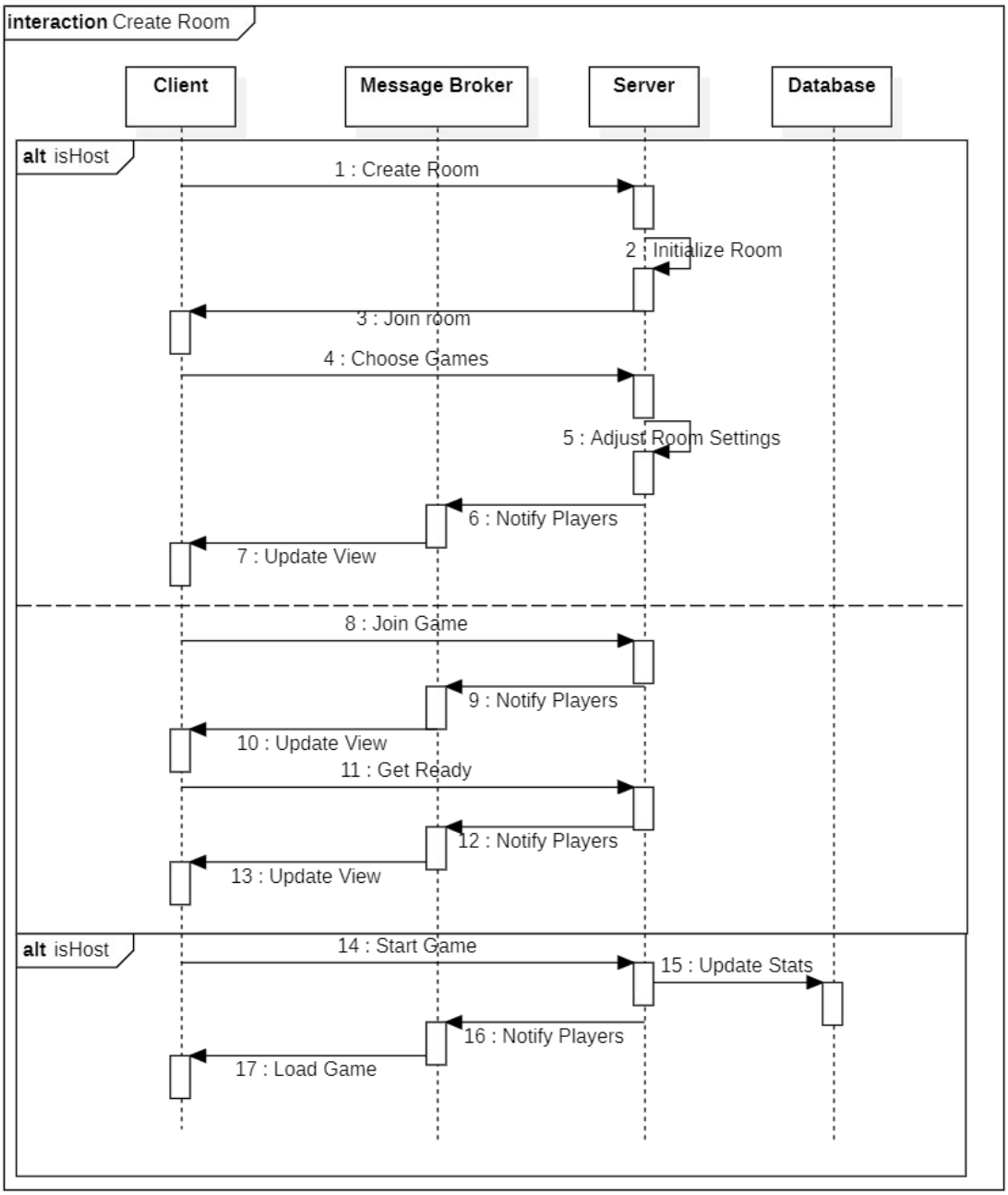
Dijagram koji je prikazan na slici 5 prikazuje strukturni pogled na implementaciju sistema kroz komponente sistema i njihovu povezanost. Struktura klijentskog dela zasnovana je na Model-View-View Model arhitekturnom obrascu koji Angular nameće. Klijentski deo razdvojen je na slojeve koji su dalje razdvojeni na podkomponente koje obrađuju odgovarajuće delove aplikacije. Asonhrona komunikacija između klijenta i servera ostvarena je korišćenjem Message Broker-a. Serverski deo aplikacije je takođe razdvojen na slojeve na osnovu funkcionalnosti. Glavna logika aplikacije sadržana je u Game Logic Layer-u serverskog dela gde se korišćenjem Strategy obrasca obezbeđuje modularnost kviza. U njemu se takođe nalaze i Singleton klase koje vode evidenciju o aktivnim sobama i partijama. Sa bazom se komunicira kroz Persistence Layer serverskog dela aplikacije koji implementira Repository obrazac.



Slika 5: Dijagram strukture sistema

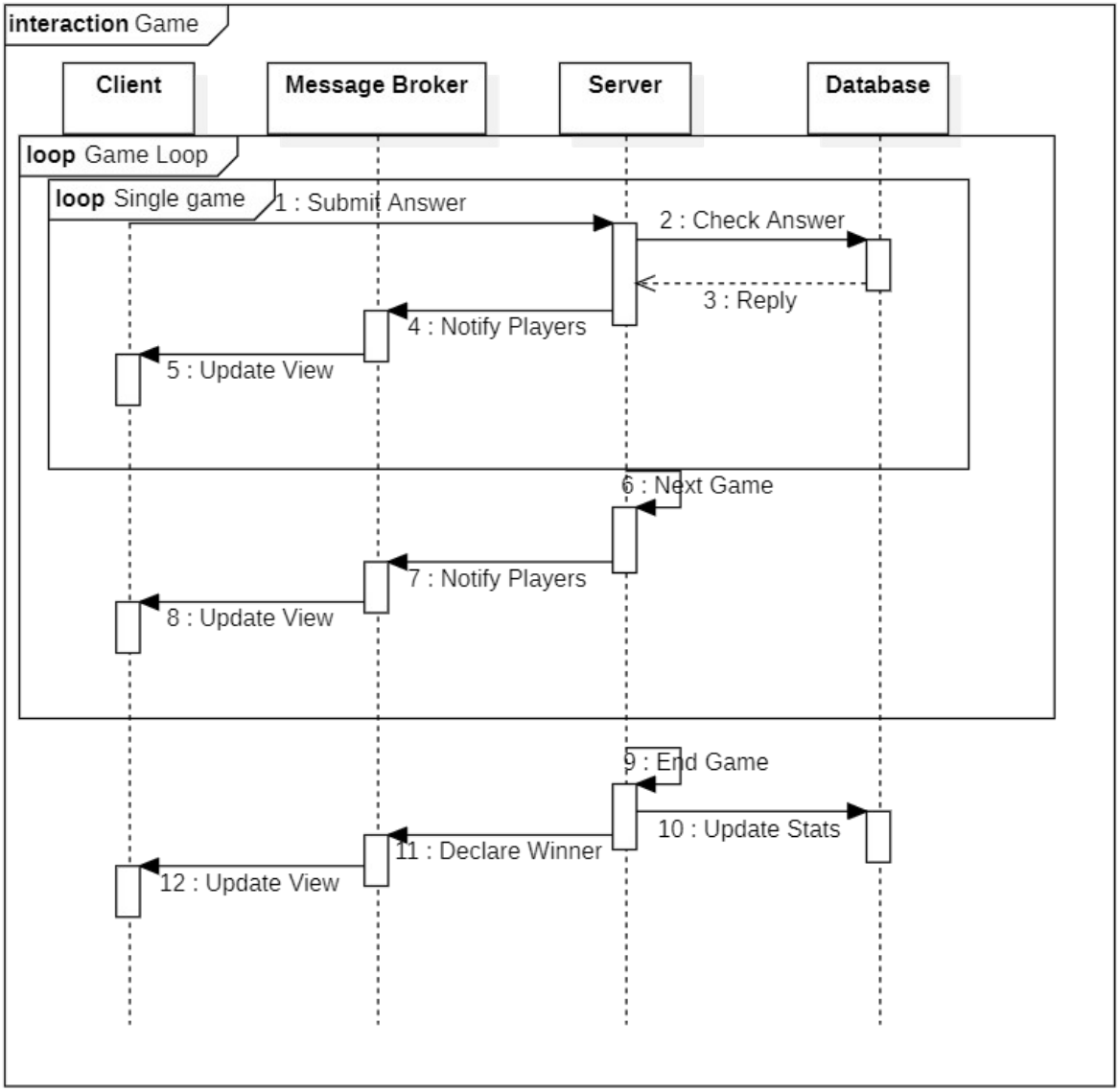
3.4. Biheviоralni pogledi

U narednom odeljku prikazani su dijagrami koji predstavljaju biheivioralne poglede na sistem. Na slici 6 prikazan je sekvencijalni dijagram kreiranja sobe i interakcije u njoj. Nakon kreiranja, host može podešavati parametre sobe, pri čemu su promene vidljive svima u sobi. Klijenti koji nisu host mogu se pridružiti sobi i označiti da su spremni za partiju. Nakon pokretanja partije od strane host-a, u bazi se ažuriraju statistike igrača koji su ušli u novu partiju.



Slika 6: Sekvencijalni dijagram kreiranja igre

Na narednoj slici prikazan je sekvencijalni dijagram koji opisuje sam tok partije. U glavnoj petlji se odigravaju odabrane igre i smenjuju dok se sve igre ne završe. Petlja svake igre menja se blago u zavisnosti od tipa i pravila same igre, ali svaka igra daje igračima mogućnost slanja odgovora, gde su odgovori i poeni igrača vidljivi svima u partiji i ažuriraju se u realnom vremenu. Nakon završetka svih igara proglašava se pobednik i partija se završava, a statistika igrača u bazi se ažurira.



Slika 7: Sekvencijalni dijagram odigravanja partije

3.5. Implementaciona pitanja

Specifikacija biblioteka i programskih okvira:

- Angular – JavaScript frontend framework
- SignalR – Message broker, obezbeđuje API za real-time klijent-server komunikaciju
- .NET Core – Serverska aplikacija
- MongoDB – Baza podataka

4. Analiza arhitekture

4.1. Potencijalni rizici u implementaciji i strategije prevazilaženja

Potencijalni rizik pri implementaciji TrešKviz sistema je problem kapaciteta servera za veliki broj korisnika. Strategija za prevazilaženje ovog rizika može biti testiranje performansi i opterećenja servera. Ovakvom metodom bi se dobili konkretni pokazatelji neophodni za planiranje kapaciteta sistema.