Arhitektura i projektovanje softvera

**Uno – faza I**

Stefan Smiljić, 18379

Nemanja Stanković, 18391

1. **Kontekst i cilj softverskog projekta**:

Cilj ovog softverskog projekta je projektovati i implementirati klon kartaške igre Uno za dva ili više igrača. Ova aplikacija će omogućiti korisnicima da u realnom vremenu, i uz odgovarajući vid komunikacije igraju ovu poznatu kartašku igru.

1. **Arhitekturno specifični zahtevi:**

-Perzistencija podataka – trajno čuvanje podataka igrača, statistike o zarađenim poenima, kao i privremeno čuvanje podataka o trenutnoj sesiji, radi otklanjanja problema vezanih za gubitak veze.

-Dostupnost servera – igrači mogu da zahtevaju početak igre u bilo kom trenutku.

-Skalabilnost – mogućnost dodavanja servera sa podrškom za load-balancing radi podizanja performansi.

-Performanse – smanjivanje odziva aplikacije radi igranja u realnom vremenu.

1. **Funkcionalni zahtevi:**

-Kreiranje naloga i logovanje na servere igre.

-Odabir broja igrača po sobi, i ulazak u sobu za igranje.

-Odigravanje određenih poteza na osnovu pravila igre.

-Korisnici mogu da proveravaju svoje podatke i da vrše potrebne izmene.

1. **Nefunkcionalni zahtevi:**

-Veći broj soba (od 2, 3 ili 4 igrača) istovremeno.

-Provera ispravnosti podataka prilikom logovanja.

-Provera validnosti poteza prilikom igranja.

-Brzo razrešavanje gubitka veze, mogućnost oporavka.

-Minimalna latencija prilikom odigravanja poteza.

1. **Arhitekturni dizajn:**
2. **Arhitekturni obrasci:**

-MVC (Model-View-Controller) – biće korišćen za raspored slojeva same aplikacije, gde će model sadržati logiku aplikacije, pogled (view) će biti rezervisan za frontend tj klijentski deo aplikacije, a kontroler će služiti kao posrednik između modela i pogleda.

-Client-server – klijentska strana podrazumeva igrače koji igraju igru, dok se sva komunikacija između klijenata, logika igre i validacija odvija na serverskom delu.

-Pub/Sub (Publisher/Subscriber) – za svaku sobu posebno, server će da ima SignalR hub koji ima ulogu publishera, dok su svi igrači u toj sobi subscriberi.

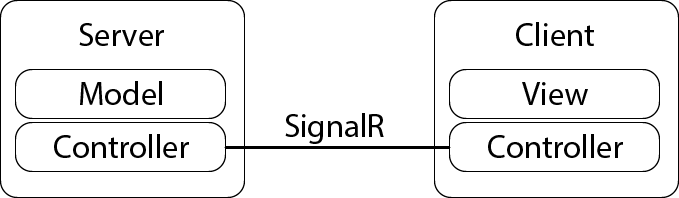
-Repository – podaci će se čuvati na određenoj bazi podataka, kojoj će MVC model da ima pristup.

-Layered – model MVC-a je razložen na više slojeva, i to: servisni, domenski i perzistencioni.

-Broker – biće iskorišćen RabbitMQ za komunikaciju između serverskih delova jer omogućava razdvajanje različitih komponenti uz stabilnu komunikaciju koja garantuje prijem poslatih poruka.

**b) Box-Line dijagrami:**

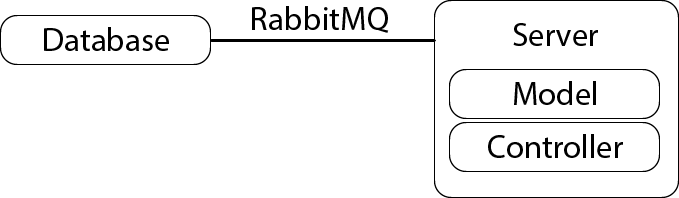
Prikaz MVC modela:

****

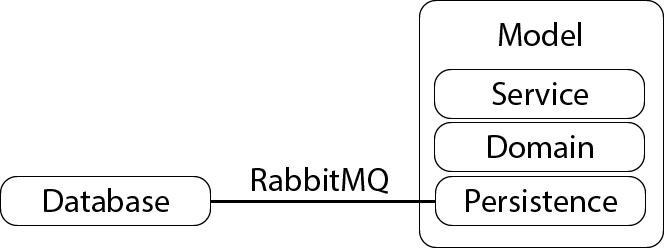
Inicijalno raspoređivanje klijenata ka serverima preko load balancera:



Prikaz povezivanja baze podataka sa serverom:

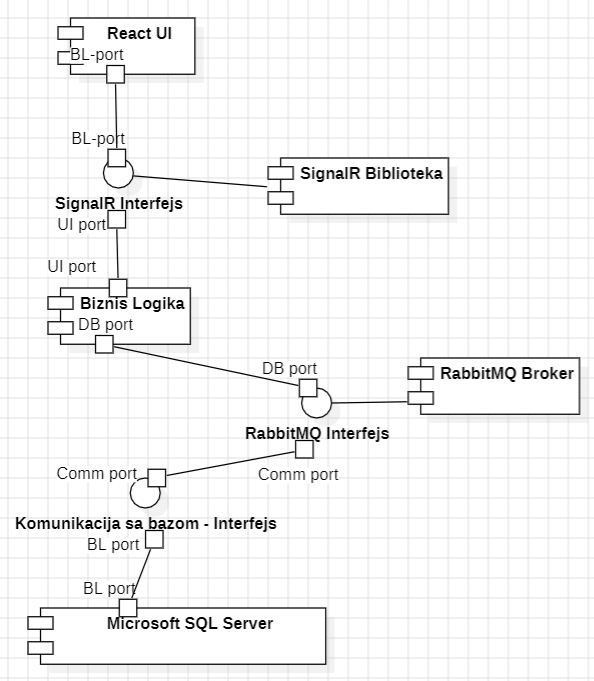


Prikaz layered arhitekture modela I povezivanja modela sa bazom:

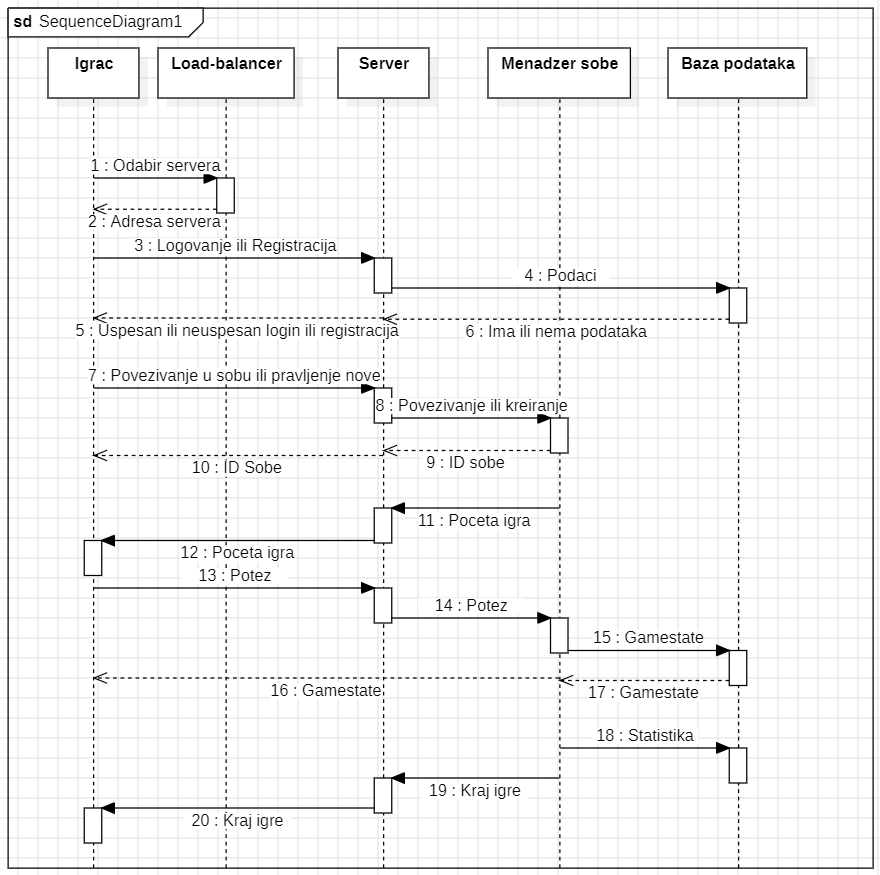


**c) Strukturni i bihevioralni dijagram:**

-Strukturni dijagram – Dijagram komponenti:



-Bihevioralni dijagram – Dijagram sekvence:



1. **Aplikacioni okviri/Biblioteke:**

Serverski deo biće implementiran korišćenjem Micorsoft-ovog .NET frameworka, baza podataka biće realizovana preko Microsoft SQL servera. Interfejs i klijentski deo biće implementirani preko React frameworka. Komunikacija između serverskih delova biće realizovana korišćenjem RabbitMQ brokera, dok će komunikacija između klijenta i servera biti realizovana preko SignalR biblioteke.