I faza

Projekat: igra Uno

Predmet: Arhitektura i projektovanje softvera

Nikola Popović, 15826

Dejan Randjelović, 15843

Sadrzaj

[1. Funkcionalni zahtevi 3](#_Toc531899054)

[2. Arhitekturni zahtevi 3](#_Toc531899055)

[3. Nefunkcionalni zahtevi 3](#_Toc531899056)

[4. Arhitekturni dizajn 3](#_Toc531899057)

[4.1. Strukturni pogled 3](#_Toc531899058)

[4.2. Bihevioralni pogled 4](#_Toc531899059)

[4.3. Alokacioni pogled 5](#_Toc531899060)

[5. Arhitekturni paterni 5](#_Toc531899061)

[6. Specifikacija biblioteka i programskih okvira 7](#_Toc531899062)

# 1. Funkcionalni zahtevi

Osnovna funkcionalnost sistema je mogućnost da više igrača igra igru Uno. Igru mogu da igraju minimum 2, a maksimalno 4 igrača. Špil se sastoji od 108 karata. Karte mogu da se jave u 4 različite boje (crvena, žuta, plava, zelena) i u vidu ,,wild“ karte koja je crne boje. Simboli karata su: 0 (po jedna za svaku boju“), 1-9 (po dve za svaku boju), ,,preskoči-igrača“ (po dve za svaku boju), ,,promeni-smer“ (po dve za svaku boju), ,,kupi-dve“ (po dve za svaku boju), ,,wild-promeni-boju“ (četiri u špilu) i ,,wild-promeni-boju-kupi-četiri“ (četiri u špilu). Na početku igre, jedna karta iz špila se stavlja na stolu i igračima diler deli po 7 karata. Pobednik je igrač koji prvi ostane bez karata.

# 2. Arhitekturni zahtevi

Sistem treba da omogući pamćenje velike količine podataka. Sistem će biti dostupan korisnicima 24/7. Šema baze podataka je sakrivena od korisnika.

# 3. Nefunkcionalni zahtevi

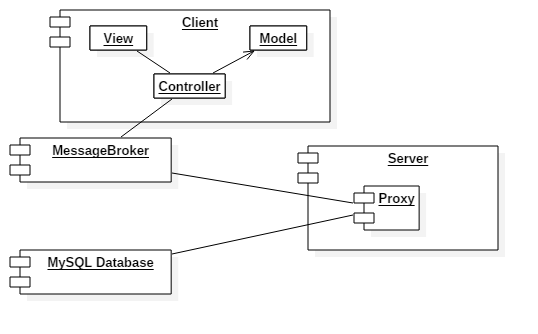
Sistem treba da bude pouzdan, dostupan korisnicima i upotrebljiv. Pored toga, mora obezbediti igru za veći broj igrača-skalabilnost. Sistem treba omogućiti dobre performanse, smanjenjem vremena odziva, i modifikovanje u budućnosti (mogućnost promene)-modifikabilnst.

# 4. Arhitekturni dizajn

## 4.1. Strukturni pogled

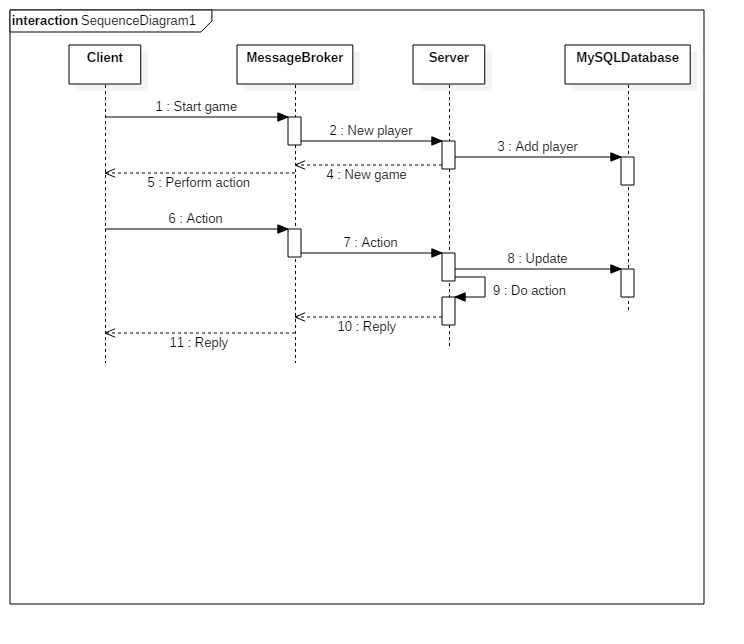
Osnovne komponente sistema su: Klijent (model, view, controller), MessageBroker, Server (Proxy) i Baza podataka.

Klijent je baziran na MVC strukturi. Komunikaciju sa Proxy-jem vrši uz pomoć svog kontrolera preko broker-a, a Proxy vrši pristup MySQL bazi podataka.



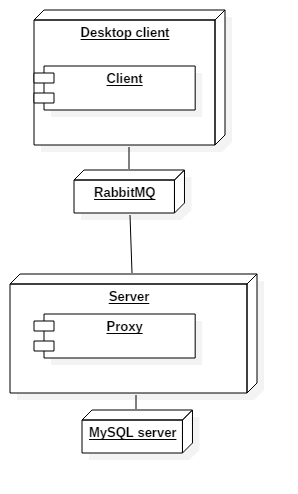
## 4.2. Bihevioralni pogled

Bihevioralni pogled opisuje interakciju izmedju komponenti sistema prilikom povezivanja novog igrača na server. Na početku se povezuju igrači u igru, a zatim se ponavlja odigravanje niza poteza tako što se dodeljuje privilegija poteza naizmenično igračima.



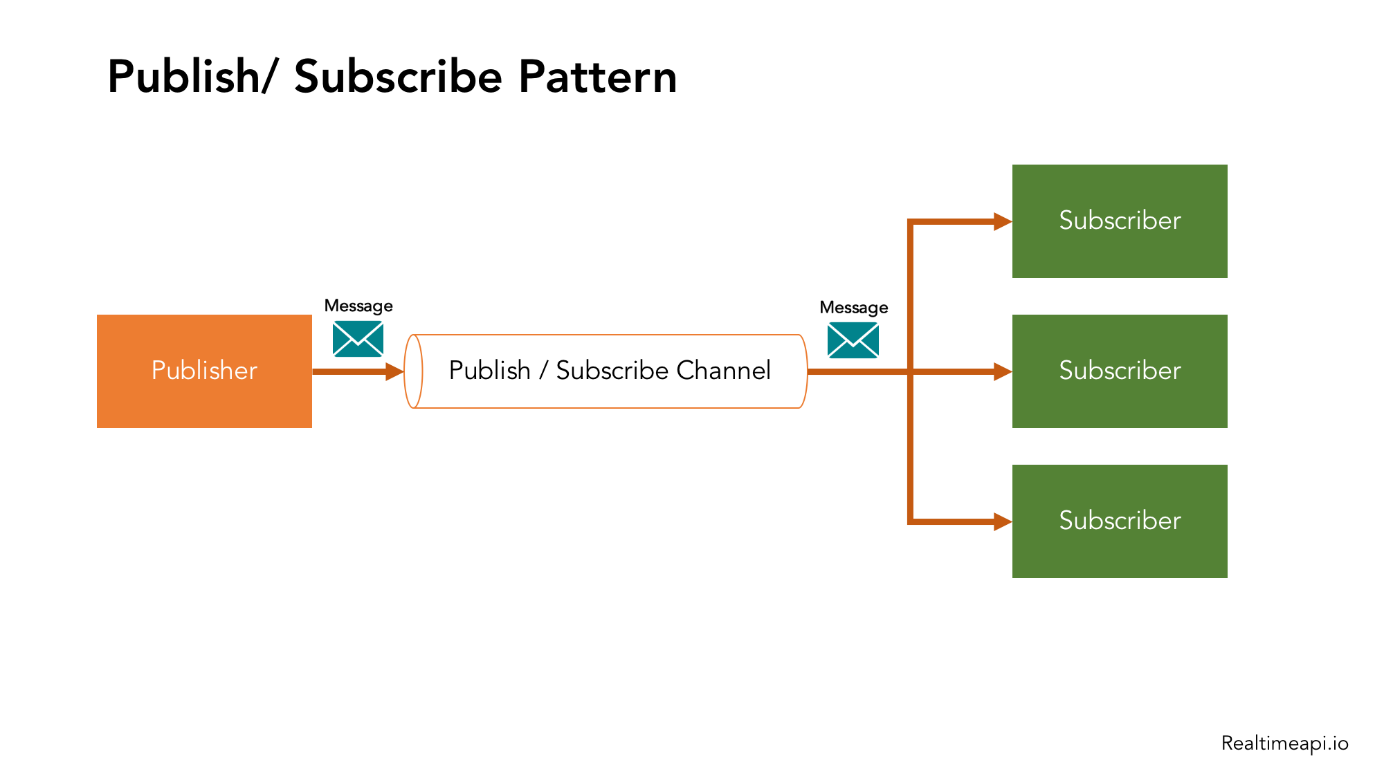
## 4.3. Alokacioni pogled

Alokacioni pogled prikazuje raspored komponenti na stvarnim računarima. Klijent se nalazi na desktop uredjaju. Za komunikaciju sa serverom se koristi RabbitMQ.

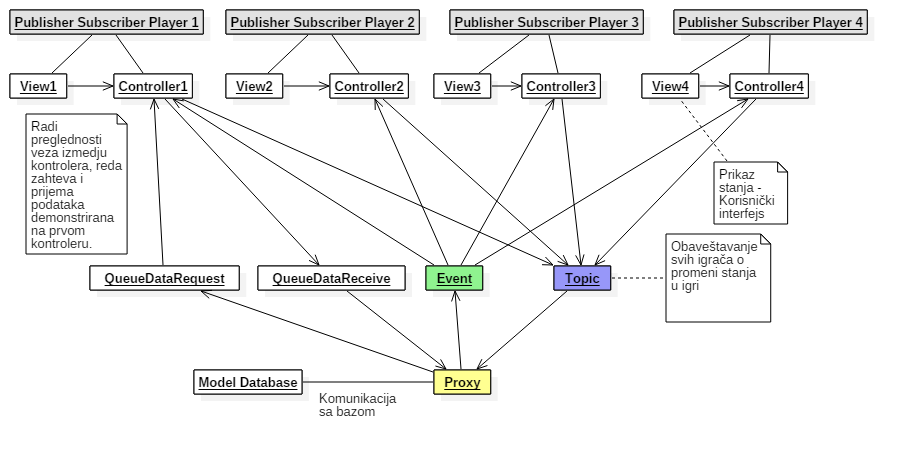


# 5. Arhitekturni paterni

Za organizaciju klijentske aplikacije koristićemo MVC (Model-View-Controller) obrazac. Za realizaciju komunikacionog dela sistema koristićemo Publisher-Subscribe obrazac. Za distribuciju komunikacije izmedju klijenata koristićemo Proxy komponentu.



Slika 1. Publisher-Subscribe obrazac



Slika 2.

Aplikacija će biti organizovana kao MVC. View zavisi od kontrolera, dok kontroler izvršava logiku aplikacije. Kontroler će dakle vršiti komunikaciju sa Proxy komponentom. Proxy će vršiti distribuciju komunikacije izmedju svih igrača. Svaki igrač je publisher i subscriber. Kada igrač izvrši neku akciju (kupovina karte, igranje karte) obaveštavaju se svi ostali igrači o toj akciji. Baza podataka vrši funkciju Model-a. Ona je centralizovana i povezana je sa Proxy-jem. Ukoliko su igraču potrebni podaci iz baze, on šalje zahtev Proxy-ju preko svog kontrolera za te podatke. Proxy na zahtev odgovara pristupom bazi podataka, pri čemu izvlači tražene podatke i vraća ih kontroleru koji je poslao zahtev.

# 6. Specifikacija biblioteka i programskih okvira

Za realizaciju aplikacije koristiće se C# u framework-u .NET. Komunikacija će biti realizovana preko RabbitMQ. Baza podataka će biti napravljena u MySQL-u, dok komunikacija sa bazom preko ORM mapper-a FluentNHibernate.