UNIVERZITET U NIŠU

ELEKTRONSKI FAKULTET



Projekat: I faza

Predmet: Arhitektura i projektovanje softvera

Studenti: Predmetni profesor:

Filip Trajković – 17503 Prof. dr Dragan Stojanović

Danilo Đorović –17100

Niš, decembar 2021.

***Sadržaj***

1. **Kontekst** **i cilj projekta3**
2. Arhitekturni zahtevi3

2.1 Arhitekturno značajni slučajevi korišćenja3

2.2 Nefunkcionalni zahtevi5

2.3 Tehnička i poslovna ograničenja5

1. Arhitekturni dizajn5

3.1 Arhitekturni obrasci5

3.2 Generalna arhitektura8

3.3 Strukturni pogledi8

3.4 Bihevioralni pogledi10

3.5 Implementaciona pitanja12

1. Analiza arhitekture12

4.1 Potencijalni rizici u implementaciji i strategije prevazilaženja12

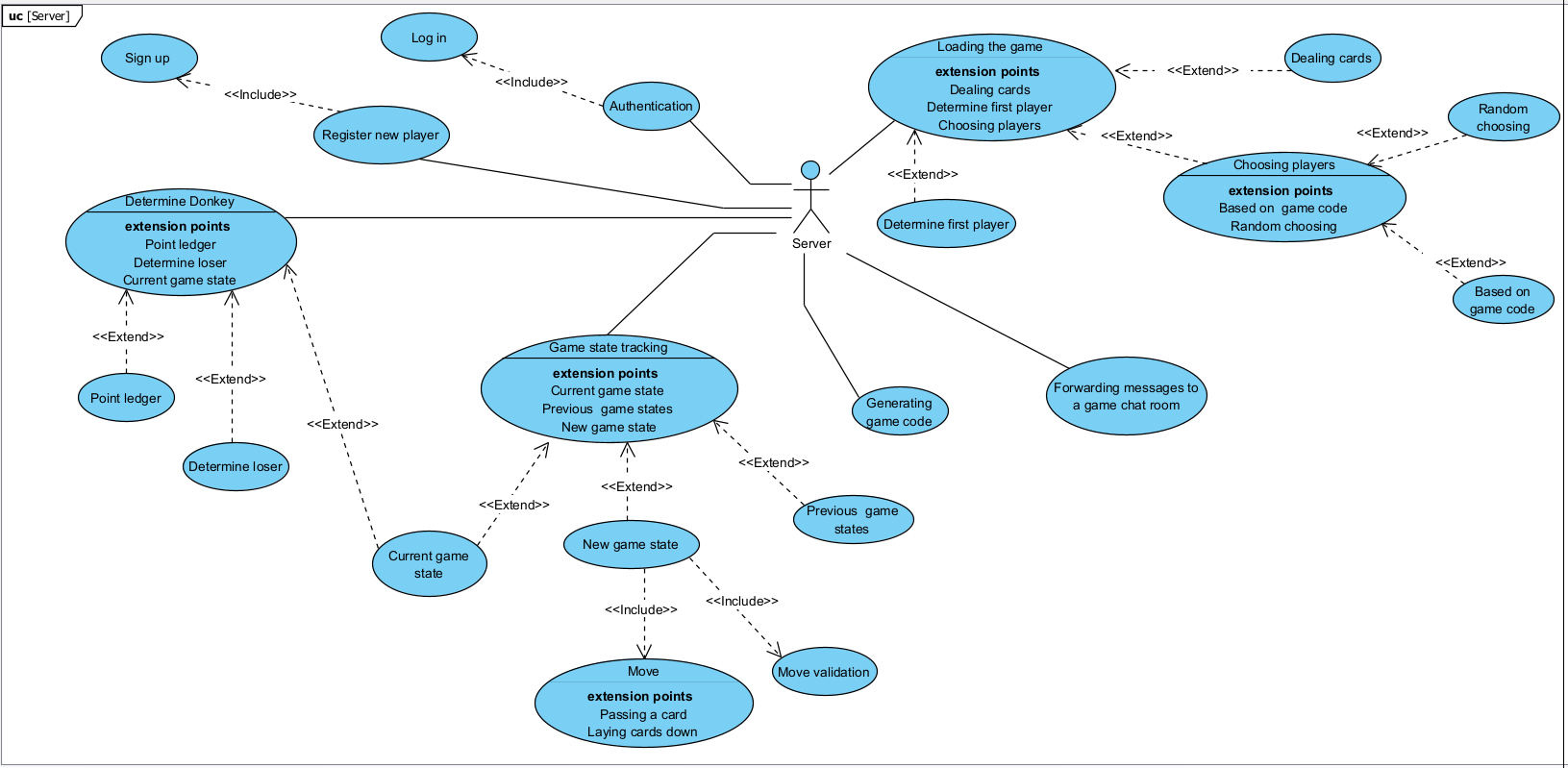
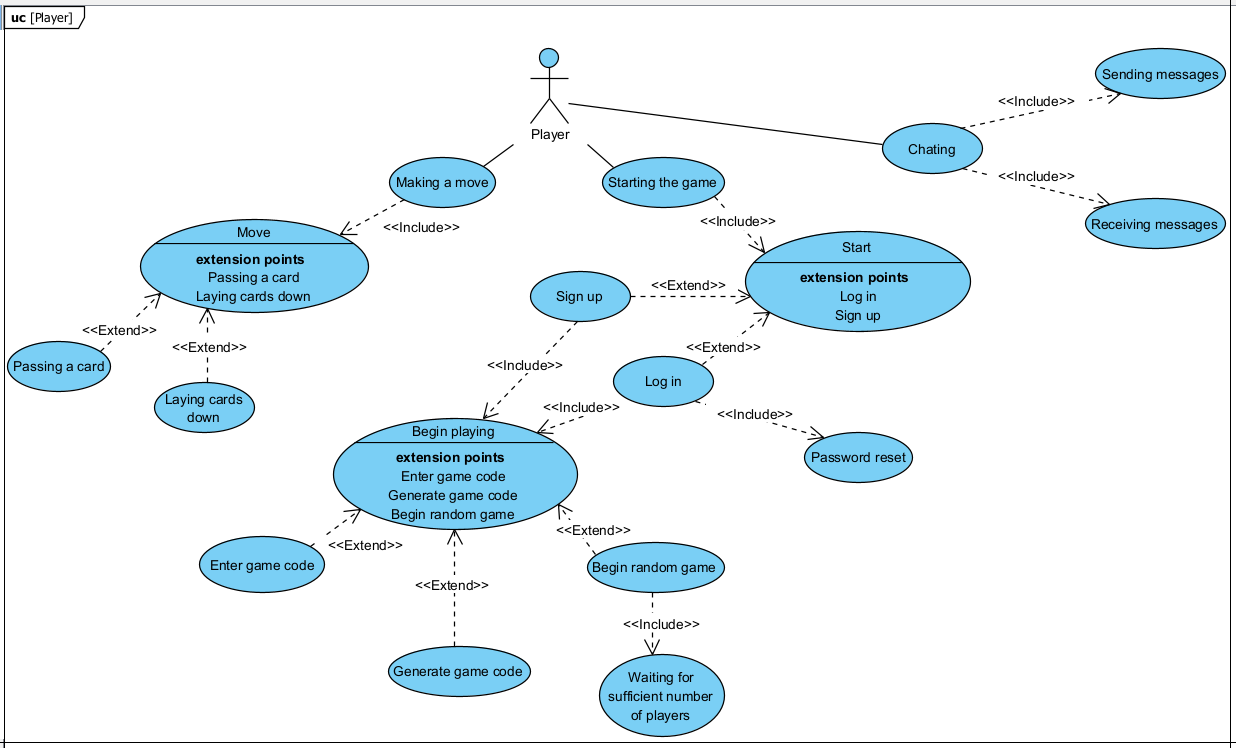
1. ***Kontekst i cilj projekta***

*DonkeyGame* je višekorisnička Web aplikacija koja će omogućiti online mogućnost igranje popularne društvene igre “Magarac“. Aplikacija će funkcionisati po principu višekorisničkog igranja igre u realnom vremenu sa akcentom na medjukorisničku interakciju tokom igre. Startovanjem aplikacije biće ponudjeno kreiranje novog naloga ili logovanje u postojeći. Svaki od novopridošlih igrača će imati mogućnost da pokrene igru što će rezultovati pokretanjem ”matchmaking” mehanizma za pronalazak nezapočete igre spajanjem više igrača koji u tom trenutku čekaju na početak. Igra počinje kada se prikupi određen broj igrača u redu čekanja definisan uslovima igre. Svakom igraču će biti podeljene karte u odnosu na pravila igre. Igra će pratiti unapred definisana i opšte prihvaćena pravila igranja, što uključuje: određivanje igrača koji je na potezu, prosleđivanje karte narednom igraču, spuštanje ruke, proveru pobednika, zabranu spuštanja ruke igraču koji poseduje određenu kartu, određivanje gubitnika, promenu stanja igre nakon svake odigranje runde, proveru validnosti poteza, itd. Ova aplikacije će podržati i međukorisničku interakciju po principu dopisivanja između igrača u toku same igre.

1. ***Arhitekturni zahtevi***

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

* 1. *Arhitekturno značajni slučajevi korišćenja*
* Registracija i prijava korisnika (korisniku je neophodan nalog kako bi učestvovao u igri) uz mogućnost resetovanja šifre prijavljenog korisnika
* Početni meni
* Kreiranje partije
* „Matchmaking“ mehanizam za spajanje igrača tokom kreiranja igre
* Podela karata igračima
* Određivanje igrača koji je prvi na potezu na početku svake runde u partiji
* Prosleđivanje karte narednom igraču
* Spuštanje „ruke“
* Proveru validnosti karata igrača koji je spustio „ruku“
* Određivanje redosleda spuštanja „ruke“ svih ostalih igrača
* Određivanje gubitnika u rundi
* Upis određenog slova u gubitnikovom rezultatu na kraju svake runde
* Provera kraja igre na kraju svake runde
* Međukorisnička interakcija dopisivanjem u toku igre



Use case dijagram igrača/klijenta

Use case dijagram serverske strane aplikacije

* 1. *Nefunkcionalni zahtevi*

• **Pouzdanost** – sistem treba da omogući perzistenciju nakon svakog odigranog poteza igrača, kao i nakon svake odigrane runde

• **Performanse** – aplikacija treba da obezbedi visoku brzinu odziva jer je po pravilima

igre to ključna karakteristika i najbolje performanse u zavisnosti od trenutka poziva određene akcije svakog igrača

• **Dostupnost** – potrebno je da je aplikacija bude dostupna 24/7

• **Skalabilnost** – potrebno je da aplikacija može da podrži povećanje broja korisnika

• **Upotrebljivost** – potrebno je da aplikacija bude intuitivna i prilagođena korisnicima svih uzrasnih doba

• **Jednostavnost** – neophodno je da korisnički interfejs bude razumljiv i funkcionalan, bez gubitka intuitivnosti

• **Sigurnost** – sistem aplikacije mora biti siguran tokom celog trajanja jedne partije, takođe treba obezbediti sigurno prenošenje i čuvanje korisničkih podataka, pre svega lozinki i drugih osetljivih informacija

* 1. *Tehnička i poslovna ograničenja*

Tehnička 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 dva različita tipa komunikacije (sinhronu komunikaciju između klijentskog i serverskog dela sistema i asinhronu komunikaciju prilikom propagacije izmena od jednog klijenta ka ostalima)

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

• **Broj igrača po partiji** – Svaka partija će imati ograničen broj igrača definisan pravilima igre

• **Brzina komunikacije** – Zbog prirode igre, neophodna je brza komunikacija između klijentskog i serverskog dela aplikacije jer je neophodna provera vremena pristizanja akcija od svakog korisnika partije

Poslovna ograničenja sistema baziraju se na ostalim pravilima same igre i od njih zavisi koje će akcije korisnik moći da obavi u datom trenutku.

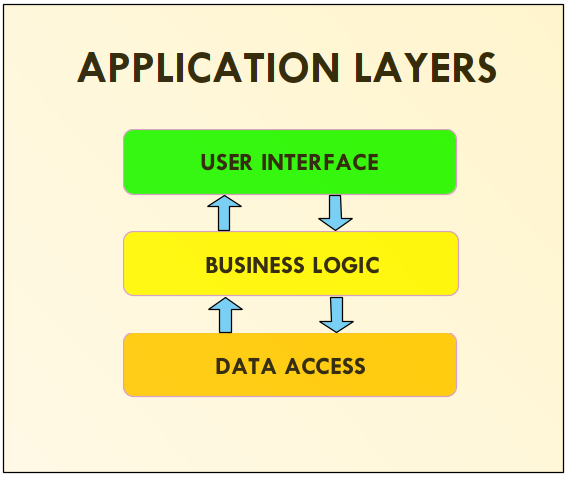
1. ***Arhitekturni dizajn***

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

* 1. *Arhitekturni obrasci*

*Layered*

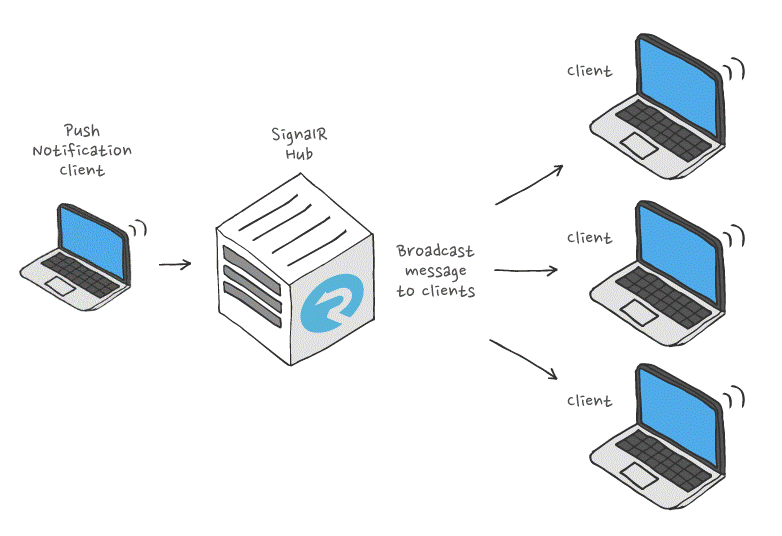
*DonkeyGame* sistem će implementirati troslojni *Layered* (Slojeviti) arhitekturni obrazac. Prvi sloj predstavlja klijentska aplikacija, drugi sloj se sastoji od SignalR i serverske aplikacije koji treba da obezbede aplikacionu logiku, dok treći sloj treba da obezbedi logiku za perzistenciju i komunikaciju sa bazom podataka – implementira *Entity Framework* kao alat za perzistenciju podataka iz relacione baze podataka. Između ovih slojeva postoje još dva međusloja: signalR između klijenta i servera i ORM alat između servera i baze podataka. Serverska aplikacija ostvaruje sinhronu komunikaciju sa klijentskom aplikacijom korišćenjem REST-ful API-ja, dok message broker služi za asinhronu komunikaciju sa klijentom. Serverska aplikacija ostvaruje komunikaciju sa bazom koristeći ORM alat za mapiranje domenskih entiteta.



Troslojna slojevita (Layered) arhitektura

*Publish-subscribe*

*DonkeyGame* sistem će implementirati *Publish-subscribe* arhitekturni obrazac (ovaj obrazac će biti implementiran kroz *SignalR* koji predstavlja sponu između klijenta i servera). Svaka igra ima određeni broj igrača. Kada dođe do bilo koje promene u igri(odigran potez ili poslata poruka), svi igrači vide da je došlo do promene. Igrači su automatski subcribe-ovani na promene u igri u kojoj učestvuju, pa će svaku promenu videti u realnom vremenu.

**

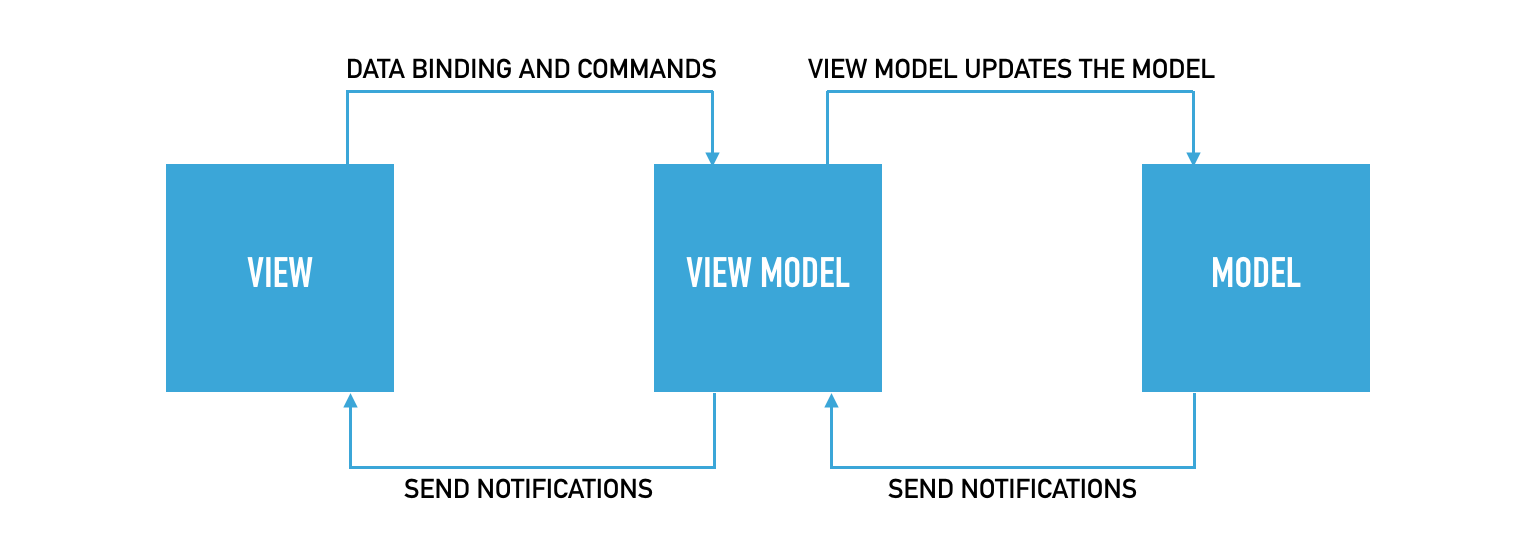
*Dijagram SignalR komunikacionog kanala*

*DOVDE SAM STIGO*

*--------------------------------------------------------------------------------------------------------------------------*

*MVVM (Model – View – View Model)*

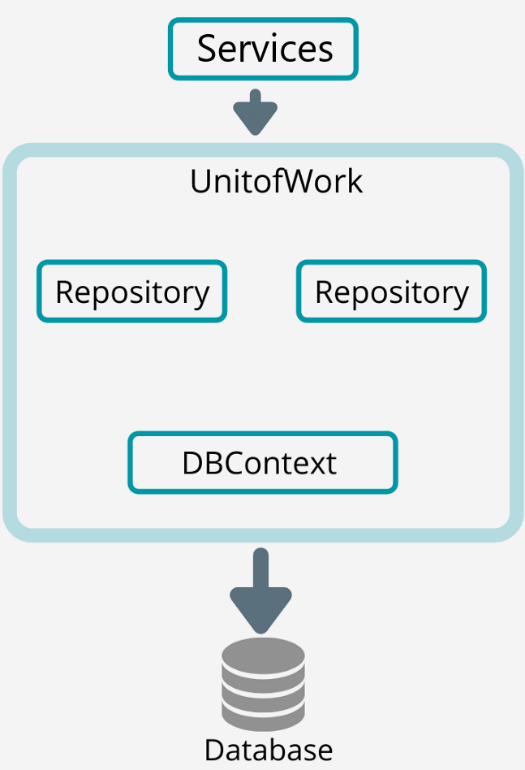
*DonkeyGame* sistem će implementirati *MVVM* obrazac na strani klijentske aplikacije. *MVVM* obrazac je modifikacija standardnog *MVC* obrasca. *MVVM* razdvaja aplikaciju na više komponenti tako da svaka komponenta ima svoje specifične odgovornosti.

**

Model – View – View Model dijagram

*Repository + UnitOfWork*

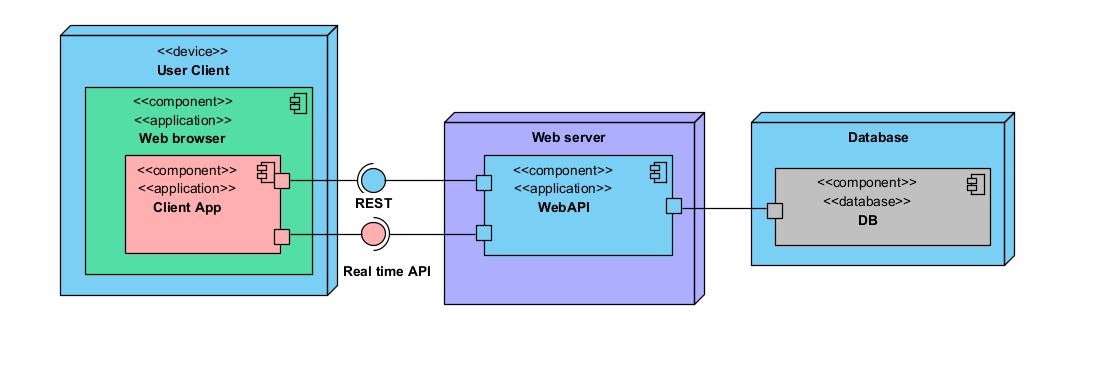
*DonkeyGame* sistem će sadržati centralizovanu bazu podataka. Na sloju podataka koristićemo *Entity Framework Core* kao ORM alat za MS Azure SQL relacionu bazu. Ovaj framework implementira UnitOfWork obrazac u obliku DbContext klase, pri čemu se on sastoji od više DbSet-ova koji čine *Repository* obrazac.

**

Repository i UnitOfWork dijagram

* 1. *Generalna arhitektura*

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

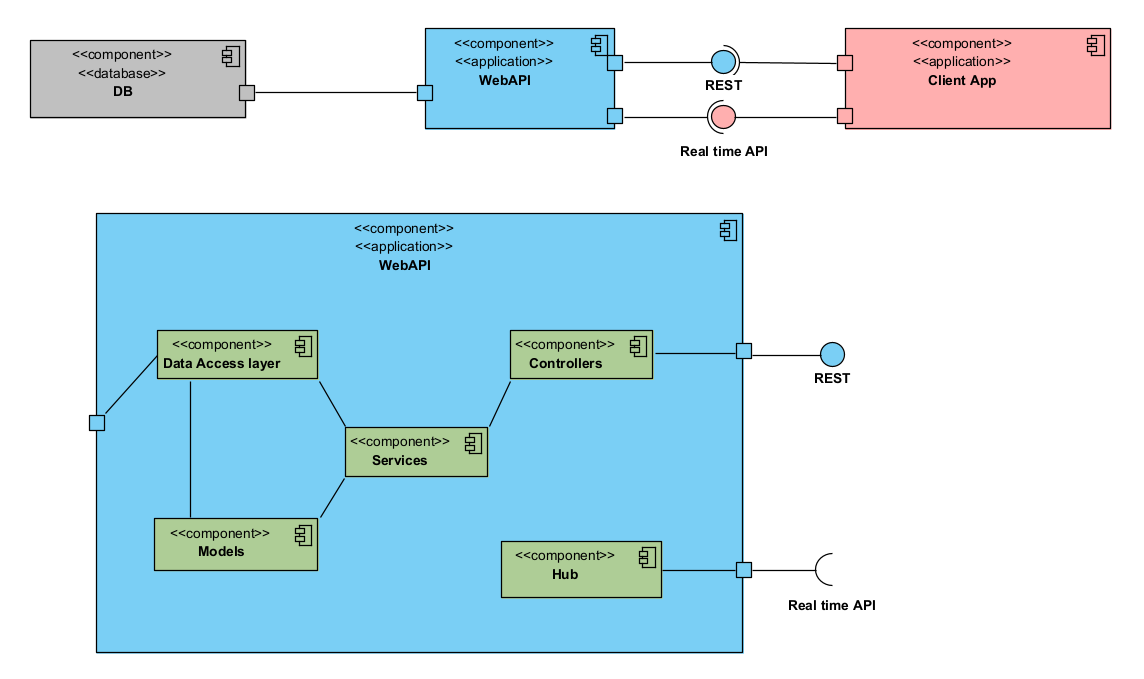


Generalna arhitektura DonkeyGame aplikacije

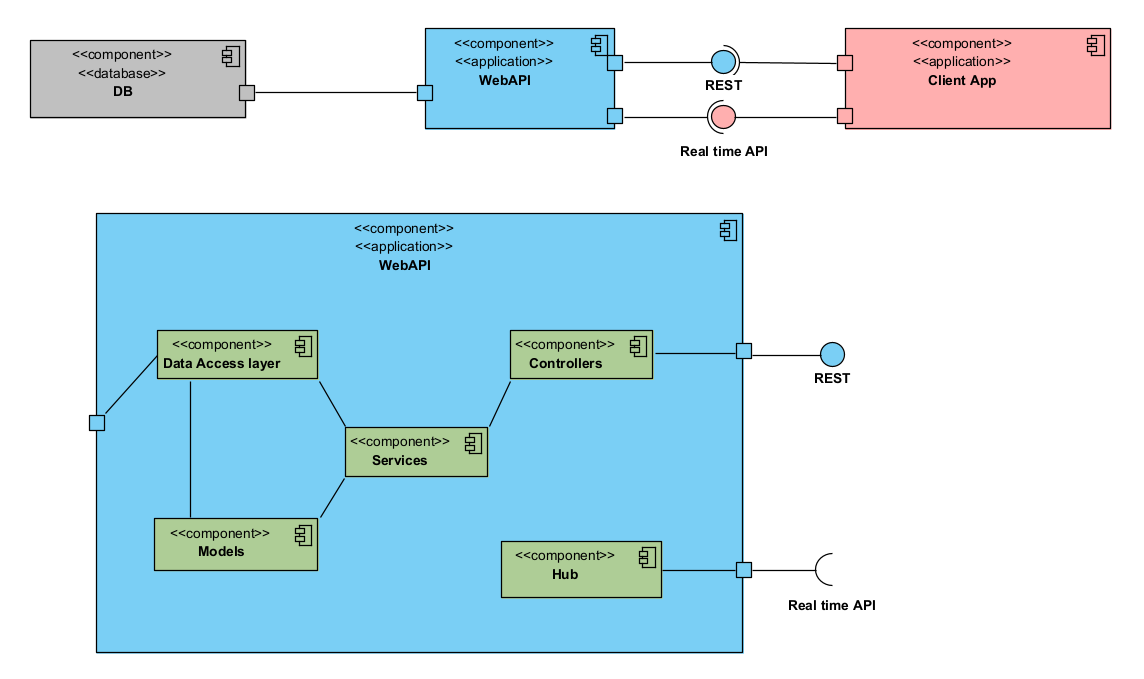
* 1. *Strukturni pogledi*

Sledeći dijagram ilustruje strukturu sistema navodeći komponente sistema kao i njihovu međusobnu povezanost. Struktura klijentske aplikacije zasnovana je na *Model-View-View Model* arhitekturnom obrascu. Klijentska i serverska aplikacija ostvaruju sinhronu komunikaciju korišćenjem REST-ful API-ja. Asinhrona komunikacija između klijentske i serverske aplikacije ostvarena je korišćenjem *Message Broker* arhitekturnog obrasca. Serverska aplikacija je zadužena za komunikaciju sa bazom podataka.

Klijentska aplikacija je implementirana kao *Web aplikacija* i predviđena je za računare svih operativnih sistema. Serverska aplikacija realizuje se kao *ASP.NET Core Web API*, dok se kao server baze podataka koristi *Azure DBMS.* Za asinhronu komunikaciju između klijenta i servera, koristi se *SignalR* message broker.



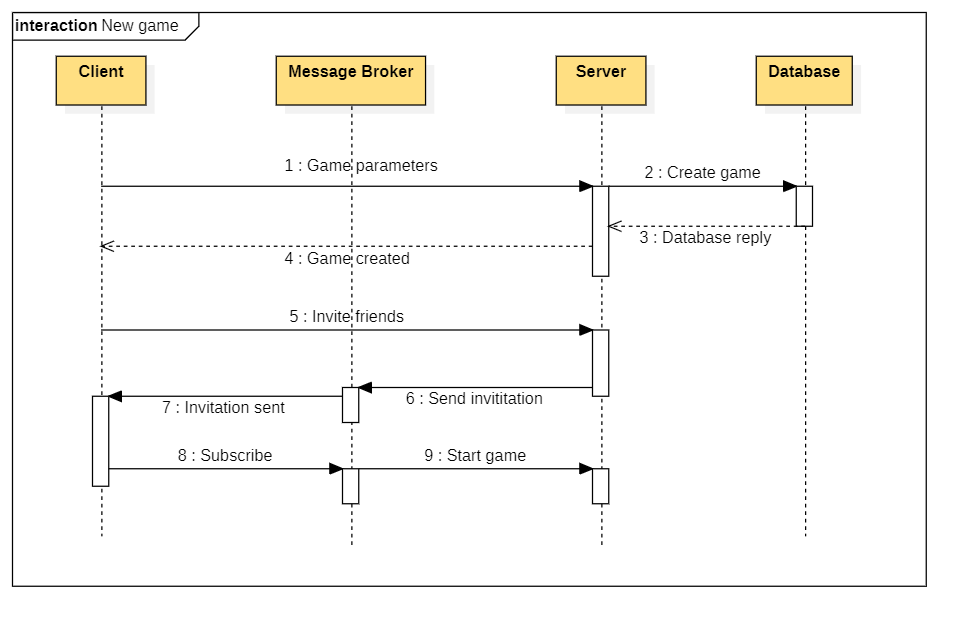
*Dijagram osnovnog strukturnog pogleda*



*Dijagram strukturnog pogleda WebAPI komponente sistema*

* 1. *Bihevioralni pogledi*

Bihevioralni pogled opisuje interakciju između komponenti sistema prilikom nastanka događaja. Sledeći dijagram predstavlja bihevioralni pogled sistema na primeru kreiranje nove igre. Klijent šalje sve parametre igre koje je odabrao serveru, koji nakon toga kreira igru i čuva je u bazi. Zatim, klijent bira koje će saigrače pozvati da se pridruže kreiranoj igri. Svi pridruženi igrači se subscribe-uju na obaveštenja o promenama u datoj igri, nakon čega igra počinje.

**

*Sekvencijalni dijagram za kreiranje nove igre*

Naredni dijagram predstavlja bihevioralni pogled sistema na primeru toka jednog poteza. Server proziva igrača koji je na redu, nakon čega se igrač priprema za napad regrupisanjem svojih armija. U toku svog poteza igrač može da napada proizvoljan broj puta. Nakon svakog napada server proverava stanje igre i po potrebi komunicira sa bazom kako bi se podaci o igri ažurirali. Zatim, server ostvaruje komunikaciju sa message broker-om putem “publish” operacije. Svi igrači koji su u datoj igri (“subscribers“) bivaju obavešteni ako je došlo do izmena.

**

*Sekvencijalni dijagram za odigravanje poteza*

* 1. *Implementaciona pitanja*

Specifikacija biblioteka i programskihokvira**:**

• **React JS** – JavaScript framework za pisanje FrontEnd Web aplikacija

• **SignalR** – obezbeđuje API za real-time klijent-server komunikaciju

• **ASP.NET Core Web API** – Serverska aplikacija

• **Entity Framework** – Objektno-relacioni maper (ORM framework)

• **Microsoft Azure SQL DBMS** – Baza podataka

1. ***Analiza arhitekture***
   1. *Potencijalni rizici u implementaciji i strategije prevazilaženja*

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