***HexaKingdom***

*Arhitektura projekta*

*Veljko Turudic 18022*

*Dimitrije Filipovic 18028*

# Kontekst i cilj softverskog projekta

HexaKingdom je online Web aplikacija, multiplayer, turn based, hexagon-map conquer igrica. Cilj igre je eliminisati protivnike osvajanjem svih heksagon polja. Registrovani korisnici mogu da kreiraju sobu za igranje igrice a ostali mu se pridruzuju. Minimalni broj igraca je 2. Na pocetku igre svaki igrac dobije odredjen broj polja sa istom jacinom resursa. Igraci svake runde dobiju nove resurse za osvajanje neutralnih polja i napadanje polja neprijateljskih igraca. Aplikacija obezbedjuje real-time igru.

# Arhitekturno specificni zahtevi

U ovom odeljku su prikazani arhitekturni zahtevi *HexaKingdom* sistema, koji obuhvataju glavne funkcionalne zahteve, nefunkcionalne zahteve i tehnicka i poslovna ogranicenja.

## Glavni funkcionalni zahtevi

* Registracija korisnika
* Prijavljivanje korisnika
* Kreiranje igre
* Pridruzivanje igri
* Inicijalna raspodela teritorija igračima
* Omogućavanje napadanja i osvajanja teritorija
* Eliminacija protivnika
* Kraj igre
* Razmena poruka izmedju igraca

## Atributi kvaliteta (Nefunkcionalni zahtevi)

**• Pouzdanost** – sistem treba da omoguci perzistenciju na pocetku/kraju poteza igraca

**• Performanse** – aplikacija treba da obezbedi što manje vreme odziva i najbolje performanse u zavisnosti od trenutnog broja korisnika i zauzetih heksagon polja.

**• Dostupnost** – potrebno je da je aplikacija bude dostupna 24/7/365 (99.99%)

**• Modifikabilnost** – potrebno je omoguciti relativno laku promenu sistema.

**• Skalabilnost** – potrebno je da aplikacija moze da podrži povećanje broja korisnika i dodavanje novih mapa.

**• Upotrebljivost** – potrebno je da aplikacija bude intuitivna i jednostavna za koriscenje.

**• Sigurnost** – Obezbediti autentifikaciju i autorizaciju, kao i enkripciju osetljivih podataka. Klasicni SQL Injection ne sme probiti zastitu servera.

## Tehnicka i poslovna ogranicenja (Nefunkcionalni zahtevi)

Tehnicka ogranicenja:

* Maximalni broj igraca proporcionalno zavisi od velicine mape heksagon polja.

Poslovna ogranicenja:

* Samo igracima iz Evrope se moze garantovati dobra latencija igre jer postoji samo 1 cvor servera u Evropi. Baza nije distribuirana na ostale kontinente.

## Arhitekturni dizajn

U nastavku su dati arhitekturni obrasci koji će biti iskorisceni za realizaciju sistema.

## 3.1Arhitekturni dizajn

* **Layered**
* **MVC**
* **Publish-Subscribe**
* **Repository**

## 3.1.1 Layered

Aplikacija će imati tri sloja – prezentacioni sloj klijenta, serverski sloj, kao i sloj perzistencije, koji obuhvata skladište podataka.

Prezentacioni sloj se izvršava na klijentu i ima ulogu posrednika između aplikacije i klijenta u vidu interakcije sa sistemom. Prezentacioni sloj će komunicirati sa slojem ispod sebe, a to je serverski sloj.

Serverski sloj će se izvršavati na serveru i implementirati sve bitne funkcionalnosti koje aplikacija treba da pruži. U okviru serverskog sloja, implementiraće se poslovna logika sistema, odnosno metode potrebne za odigravanje igre, ali i metode potrebne za perzistenciju podataka koje generiše aplikacija.

Serverski sloj komunicira sa prezentacionim slojem, ali i slojem perzistencije, a komunikacija sa klijentima se odigrava direktno preko implementiranog RESTful API-ja, ili posredno, korišćenjem third-party message broker-a. Sloj perzistencije predstavlja samu bazu podataka, koja će skladištiti podatke relevantne za aplikaciju.

## 3.1.2 MVC

Serverska strana će klijentskoj pružati usluge RESTful API-ja, uređenog po MVC obrascu. API će sadržati klase controller-a kao tačke pristupa uslugama API-ja, model će se kao izdvojena komponenta baviti domenskim klasama i njihovom perzistencijom, ali sama view komponenta ovog obrasca neće postojati u standardnom obliku, imajući u vidu da serverska strana neće direktno kreirati delove korisničkog interfejsa.

Umesto toga, API će klijentskoj strani vraćati odgovor u vidu važnih podataka predstavljenih u JSON formatu.

## 3.1.3 Publish-Subscribe

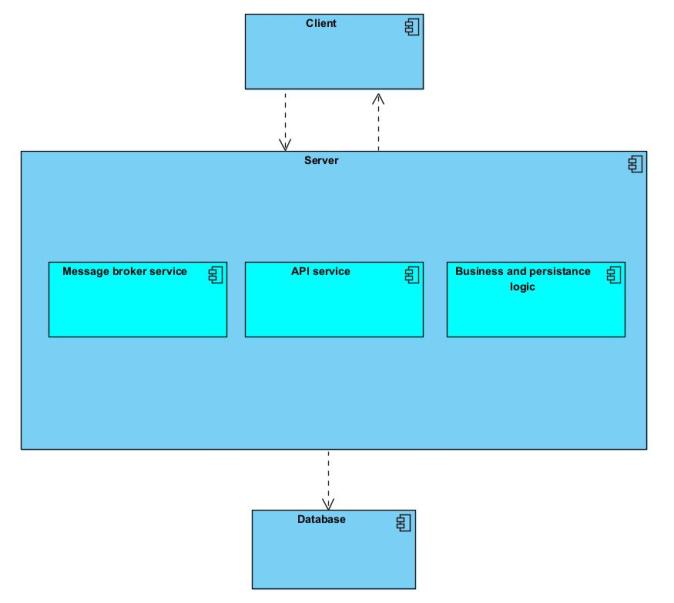
Arhitekturni obrazac *Publish/Subscribe* će biti implementiran preko message broker-a koji će biti zadužen za asinhronu komunikaciju između klijenta i servera.

Svi igrači jedne partije su automatski *subscribe*-ovani za prijem informacija o izmenama stanja igre (odigrani potezi protivnika), kao i za prijem poruka od strane ostalih igrača u partiji. Rezultat ovoga biće vidljiva promena prikaza igre u realnom vremenu.

## 3.1.4 Repository

Za implementaciju perzistencije koristiće se centralizovano skladište pasivnog tipa. Komponente sloja servera komuniciraju sa skladištem pozivom odgovarajućih metoda. Stanje sistema ogleda se u stanju partija igara koje su pokrenute u datom trenutku, i pošto ih ima više, potrebno je obezbediti konkurentan pristup skladištu.

## 3.2 Generalna arhitektura

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

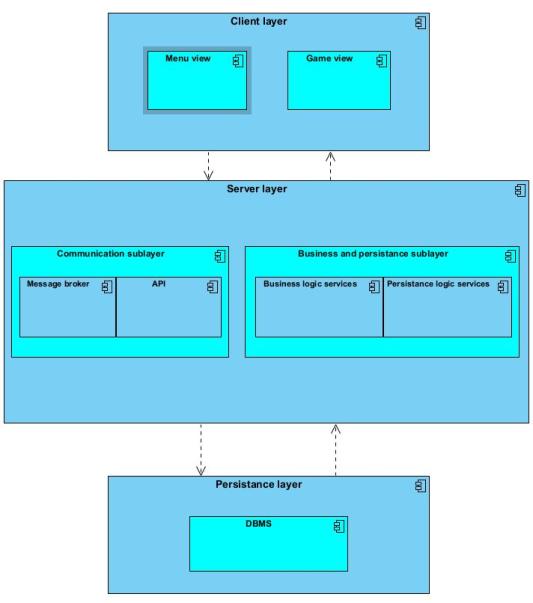
## 3.3 Strukturni pogledi

Dijagram prikazuje strukturu komponenti sistema i njihovu povezanost.

Klijentski sloj, koji predstavlja deo aplikacije sa kojim korisnici interaguju, se sastoji od skripti, od kojih su neke zadužene za prikaz, dok neke ostvaruju komunikaciju sa serverom. Serverski sloj se sastoji od komunikacionog podsloja i podsloja za logiku igre tj. biznis logiku sa podslojem perzistencije.

Komunikacioni podsloj obuhvata RESTful Web API za sinhronu komunikaciju i Message Broker za asinhronu komunikaciju sa klijentom.

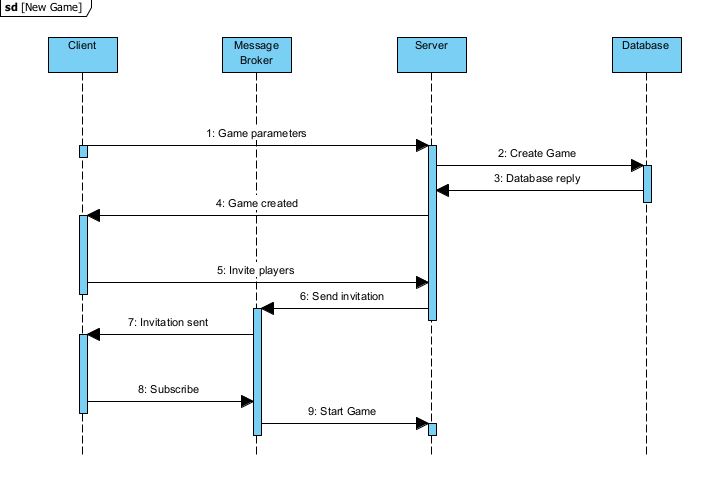
Na sloju perzistencije se nalazi DBMS kao konekcija sa bazom podataka.



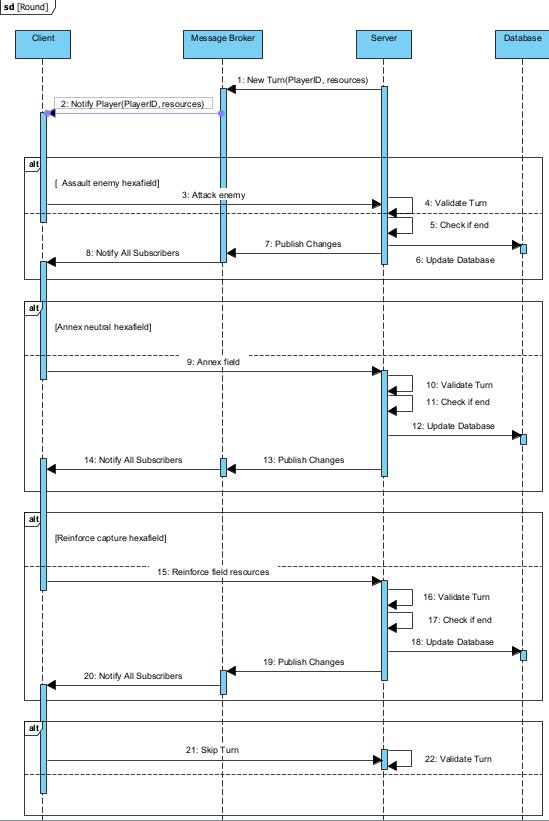
## 3.4 Bihevioralni pogledi

## 3.4.1 Nova igra

Sekvencijalni dijagram koji pokazuje proces kreiranje igre.



## 3.4.2 Odigravanje igre



## Implementacija

* **Front-end: React.js + Typescript, HTML, CSS,**
* **Back-end: Express.js + Typescript**
* **ORM: Mongoose**
* **DBMS: MongoDB**
* **Message-Broker: RabbitMQ**