**Road to Glory**

Arhitekturni dokument

**Sadržaj**

[1. Kontekst i cilj projekta 3](#_Toc153139589)

[2. Arhitekturni zahtevi 3](#_Toc153139590)

[Ne-funkcionalni zahtevi 3](#_Toc153139591)

[Tehnicka i poslovna ograničenja 4](#_Toc153139592)

[3. Arhitekturni dizajn 4](#_Toc153139593)

[Arhitekturni obrasci 4](#_Toc153139594)

[Generalna arhitektura 6](#_Toc153139595)

[Strukturni pogled 6](#_Toc153139596)

[Bihevioralni pogled 7](#_Toc153139597)

[4. Implementacija 10](#_Toc153139598)

# Kontekst i cilj projekta

Road to Glory je strateška igra za 2 igrača koja je realizovana kao veb aplikacija. Cilj igre je osvajanje neprijateljske države. Na početku oba igrača dobijaju jednaku količinu resursa i teritorije da bi mogli valjano da razvijaju svoju državu i vojsku. Igrač razvija državu tako što gradi nove vojne, naučne i objekte za prikupljanje resursa. Igrači igraju naizmenično i u toku protivnikovog poteza igrač može samo da obzervira šta se desava, ali ne i da učestvuje. Da bi igrač osvojio državu potrebno je samo da zauzme grlavni grad te države, a ne i da uništi svu vojsku protivnika.

# Arhitekturni zahtevi

Ovde ćemo definisati arhitekturno značajne slučajeve korišćenja, glavne funkcionalne i ne-funkcionalne zahteve (atributi kvaliteta) i tehnička i poslovna ograničenja vezana za realizaciju projekta Road to Glory.

Funkcionalni zahtevi

* Registracija korisnika
* Prijavljivanje korisnika
* Kreiranje igre
* Pridruživanje igri
* Inicijalna dodela resursa i tertorije svakom od igrača
* Omogućiti igračima da igraju poteze naizmenično
* Omogućiti igračima da grade vojsku
* Omogućiti igračima da grade objekte
* Omogućiti igračima da prikupljaju resurse potrebne za izgradnju
* Omogućiti igračima da unaprede neku od svojih jedinica
* Kraj igre onog trenutka kada neki od igrača zauzme glavni grad protivniku
* Razmena poruka izmedju igrača

## Ne-funkcionalni zahtevi

* **Pouzdanost** **i dostupnost**: Potrebno je omogućiti da aplikacija bude dovoljno pouzdana kako ne bi došlo do gubitka podataka u slučaju gubitka konekcije sa serverom.
* **Skalabilnost**: omogućiti da aplikacija skalira kako bi podržala rastući broj korisnika ili povećane zahteve za resursima. Elastičnost u resursima kako bi se osiguralo glatko funkcioniranje čak i u razdobljima visokog opterećenja.
* **Lakoća** **korišćenja**: Korisnički interfejs treba da bude dovoljno intuitivan za korisnike sistema, kako bi se povećala korisnost aplikacije
* **Performanse**: Aplikacija mora pružati odgovarajuću brzinu odziva za korisničke zahteve. Vreme učitavanja stranica treba biti minimalno, osiguravajući brz i efikasan rad.
* **Sigurnost**: Zaštita korisničkih podataka putem enkripcije i drugih sigurnosnih mehanizama. Implementacija autentikacije i autorizacije korisnika radi zaštite povjerljivih informacija. Takodje, treba osigurati da ni jedan SQL injection ne sme da probije zaštitu servera.

## Tehnicka i poslovna ograničenja

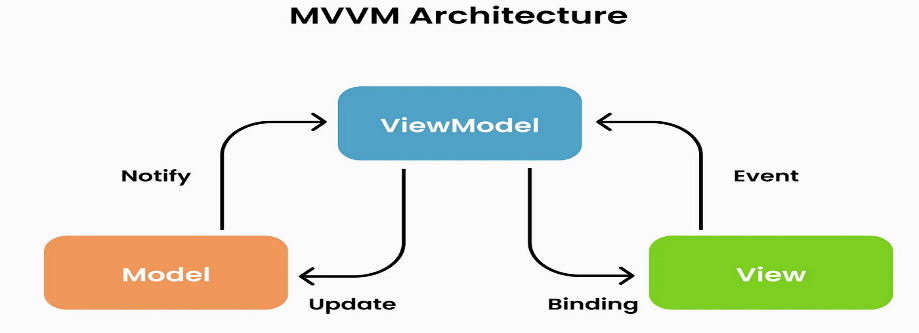
* **Optimizacija za web čitače** – Google Chrome, Mozilla Firefox, Opera
* **Apstrakcija podataka** – važno je da interna organizacija baze podataka(šema) bude sakrivena od strane API-ja
* **Mrežna latencija**: Ovisno o geografskim lokacijama igrača, mrežna latencija može biti problem. Treba minimizirati kašnjenje između akcije igrača i njezine provedbe u igri.

# Arhitekturni dizajn

Dati su arhitekturni obrasci korišćeni u projektovanju arhitekture aplikacije Road to Glory, generalna arhitektura i strukturni i bihevioralni pogled na aplikaciju. Pored ovih obrazaca, moguće je da se dodaj još neki obrasci ukoliko se za njima ukaže potreba.

## Arhitekturni obrasci

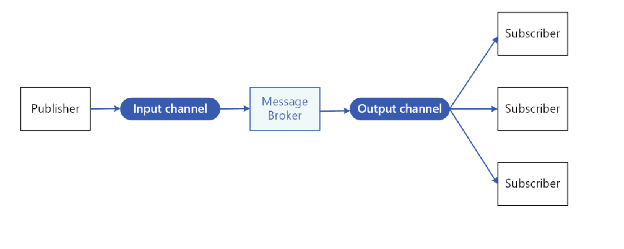
* **Layered Arhitektura**: Arhitekturni obrazac koji forsira podelu strukture aplikacije u slojeve. Svaki sloj koristi usluge sloja neposredno ispod, a pruža usluge sloju neposredno iznad. Kod projektovanja aplikacije Road to Glory biće korišćena troslojna arhitktura, standard za projektovanje web aplikacija. Tri sloja su :
* Klijentski sloj: lokalni interfejs koji se koristi za kreiranje, modeliranje, analizu, predstavljanje, izveštavanje i distribuciju različitog sadržaja. Ovaj sloj predstavljaće sloj prikazan korisnicima u Web browseru preko kog oni mogu da interaguju sa ostalim slojevima aplikacije.
* Serverski sloj: ovaj sloj omogućuje različite oblike komunikacije zavisno o zahtevim klijenta. Zahtevi primljeni putem RESTful API-ja omogućuju trenutnu i direktnu interakciju s aplikacijom, dok se kroz message broker olakšava asinhrona komunikacija, pružajući sistemu sposobnost obrade poruka bez direktne veze s klijentom u stvarnom vremenu.
* Sloj baze podataka: Ovaj sloj je zadužen za pristup podacima, bilo da se radi o bazi podataka, datotekama ili eksternim servisima. Komponente u ovom sloju obavljaju operacije čitanja, pisanja i ažuriranja podataka. Ovaj sloj će omogućiti trajnu perzistenciju ovako nastalih podataka.
* **MVVM** (Model-View-ViewModel): Ovaj arhitekturni obrazac služi kao posrednik između pogleda i modela, takođe služi za prikaz podataka vezanih za određeni pogled. Model-View-ViewModel ćemo koristiti radi razdvajanja korisničkog interfejsa i poslvne logike aplikacija i na taj način pruža mogućnost lakšeg održavanja aplikacije, boljeg testiranja same aplikacije kao i njenu fleksibilnost.



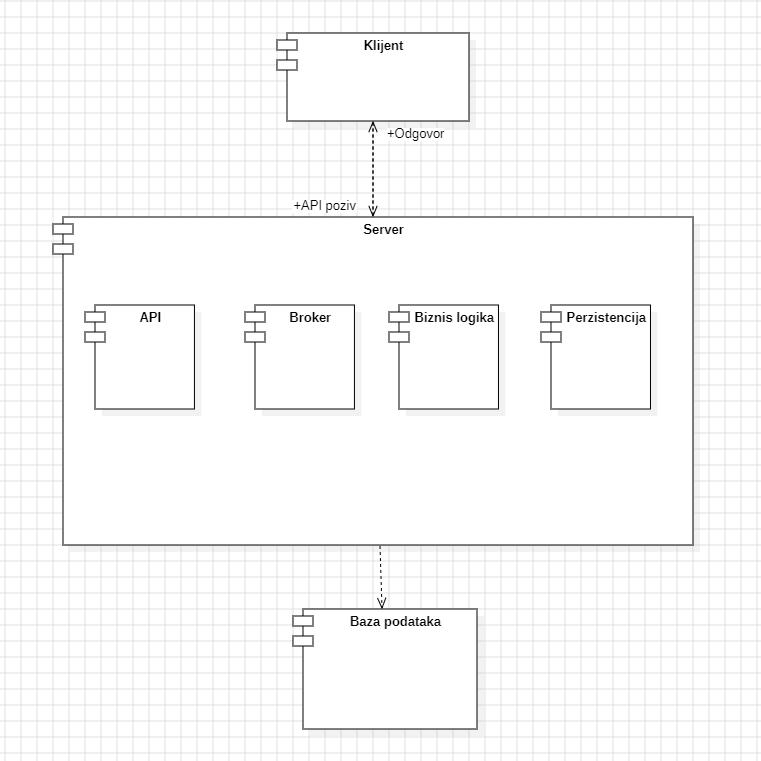
* **Repository**: Arhitekturni obrazac dizajna softvera koji se koristi za razdvajanje poslovne logike aplikacije od logike pristupa podacima. Ovaj obrazac omogućuje klijentima da komuniciraju s podacima kroz jednostavan API, a da pritom ne moraju biti svesni detalja o tome kako se podaci probavljaju iz baze podataka. Oni centralizuju zajedničku funkcionalnost pristupa podacima, obezbeđujući bolju mogućnost održavanja i razdvajajući infrastrukturu ili tehnologiju koja se koristi za pristup bazama podataka od sloja modela domena.

Ovaj obrazac biće korišćen radi postizanja postojanja interfejsa između centralizovanog skladišta (baza podataka) i mapiranja tih podataka(za koji će biti korišćen TypeORM). Klijent će graditi upite koje šalje u skladište za odgovore. Repozitorijumi će enkaplsulirati skupove objekata iz baze podataka i operacije koje se mogu izvršiti nad njima, obezbeđujući put koji je bliži sloju perzistencije

* **Publish/Subscribe**: Arhitekturni obrazac koji obezbeđuje okvir za razmenu poruka izmedju onoga ko ih izdaje (publisher) i pretplatnika (subscriber). Message broker je posrednički softver koji omogućuje komunikaciju između izdavača (publisher) i pretplatnika (subscriber) tako što prima, upravlja i distribuira poruke među različitim dijelovima sustava.



## Generalna arhitektura



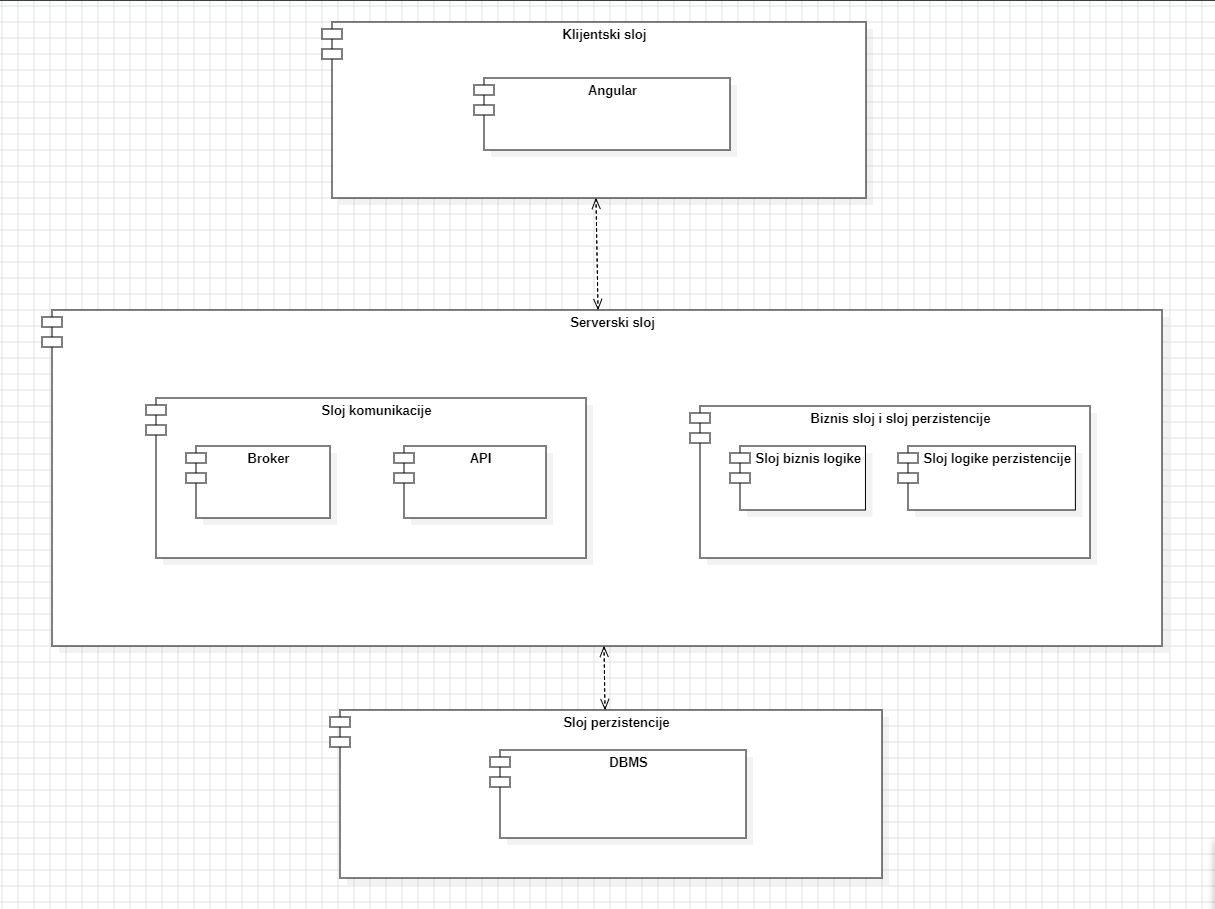
## Strukturni pogled

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.

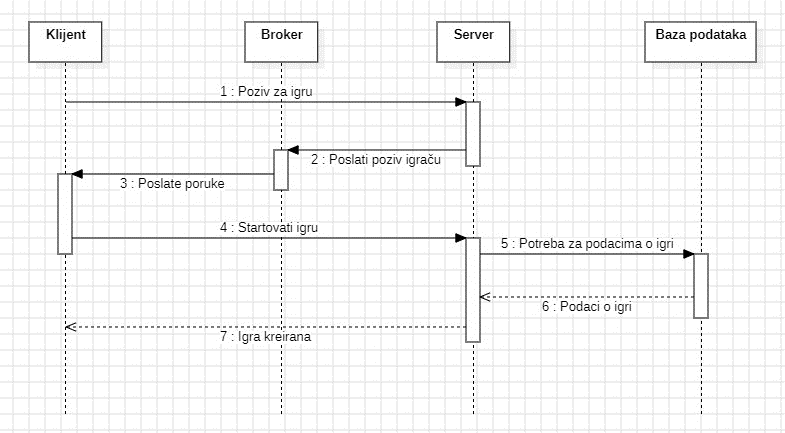
Komunikacioni podsloj obuhvata RESTful Web API za sinhronu komunikaciju i Message Broker za asinhronu komunikaciju sa klijentom.

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

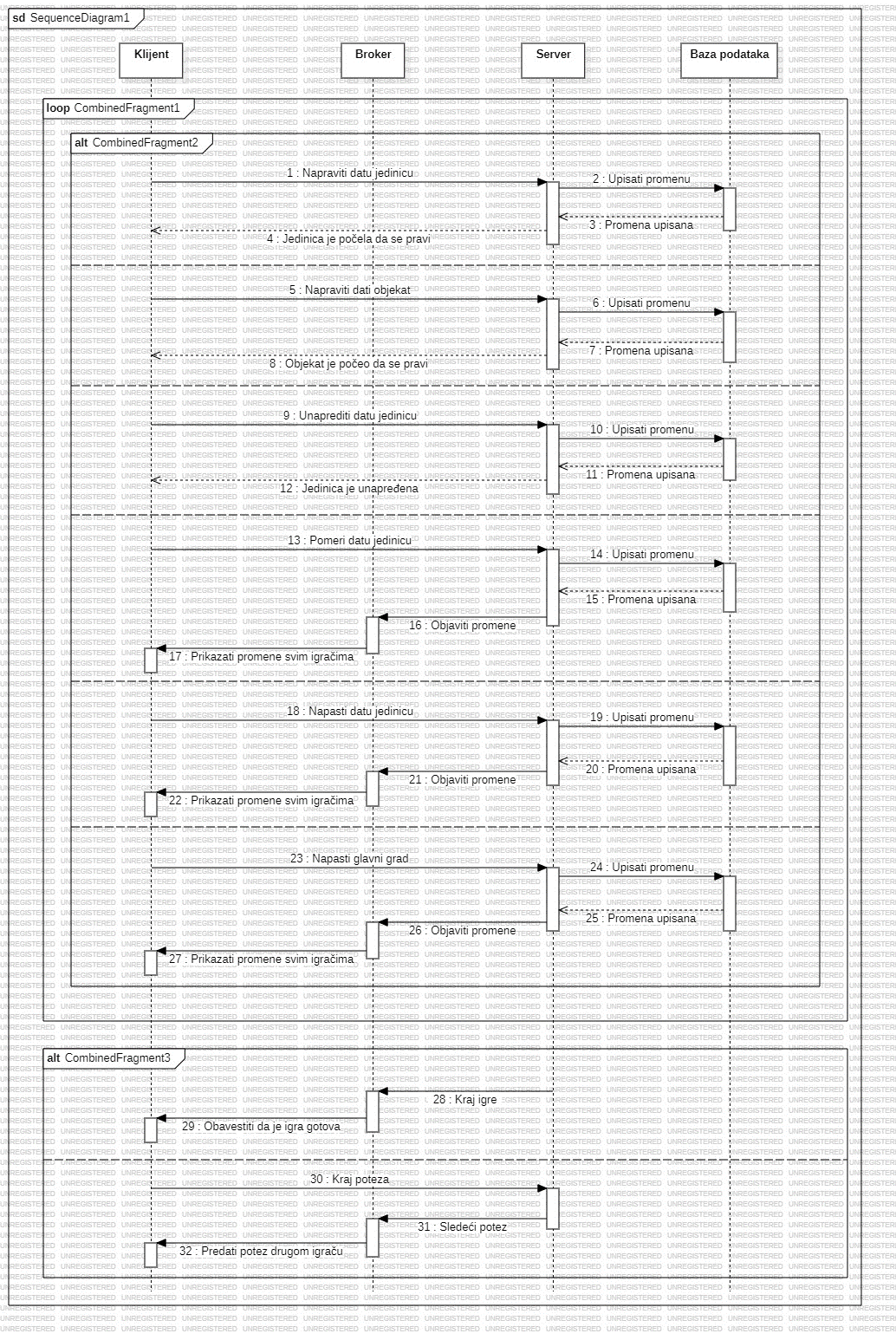


## Bihevioralni pogled

* **Nova Igra**



* **Odigravanje poteza**



# Implementacija

* **Angular** – front-end
* Robustan JavaScript okvir za izradu web aplikacija koji omogućava razvoj visoko performantnih i skalabilnih projekata kroz modularnu komponentnu strukturu. Koristeći TypeScript, pruža snažnu tipizaciju i olakšava otkrivanje grešaka, čime olakšava proces razvoja modernih aplikacija
* **Nest.JS** – back-end
* Moderni Node.js okvir koji kombinuje TypeScript sa strukturom inspirisanom Angularom, olakšavajući razvoj skalabilnih server-side aplikacija kroz modularnost i podršku za mikroservise. Ovaj okvir omogućava jednostavno održavanje i izgradnju visokoperformantnih API-ja kroz svoju korisnički orijentisanu arhitekturu.
* **PostgreSQL** – objetno-relacioni sistem za upravljanje podacima
* **Signal R** - biblioteka koja omogućava dinamičko ažuriranje podataka u stvarnom vremenu
* **TypeORM** – TypeScript kompatibilan alat za objektno-relaciono mapiranje