Arhitektura I Projektovanje softvera FAZA1 –
Arhitekturni projekat

Ena

Nikola Đorđević - 18153

Predrag Tošić – 18454

Kontekst i cilj projekta

Veb aplikacija simulira igru UNO, popularnu kartašku igru, pružajući korisnicima mogućnost da je igraju online sa prijateljima. Aplikacija je dostupna na različitim uređajima i omogućava korisnicima da uživaju bez potrebe za fizičkim kartama.

Korisnici mogu da kreiraju nalog. Registrovani korisnici mogu da kreiraju sobu za igru, a ostali mogu da im se pridruže. Minimalni broj igraca je 2, a maksimalni je 8. Na početku igre svaki korisnik dobija 7 karata. Igrači redom dobijaju potez, tada mogu da iz svog špila stave validnu kartu na zajednički špil, ukoliko nemaju validnu kartu mogu da "kupe" I ukoliko je validna mogu da je stave na zajednički špil.

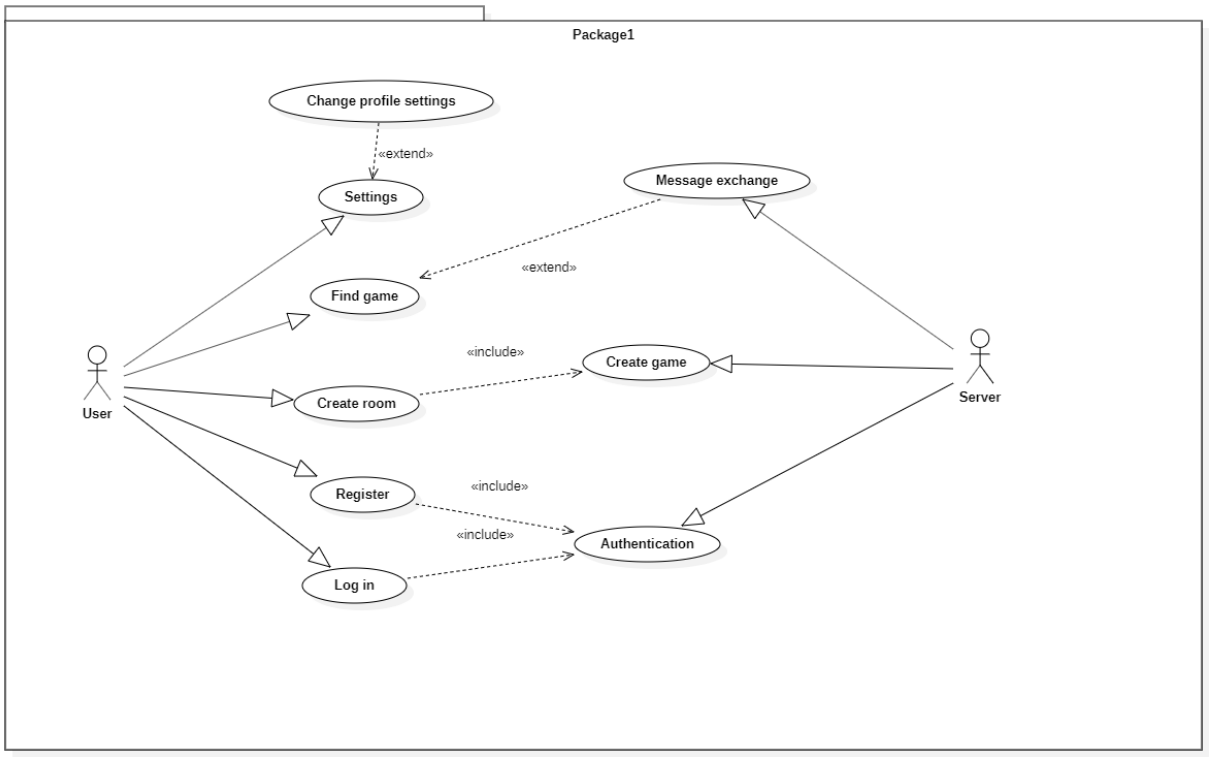
Arhitekturni zahtevi

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

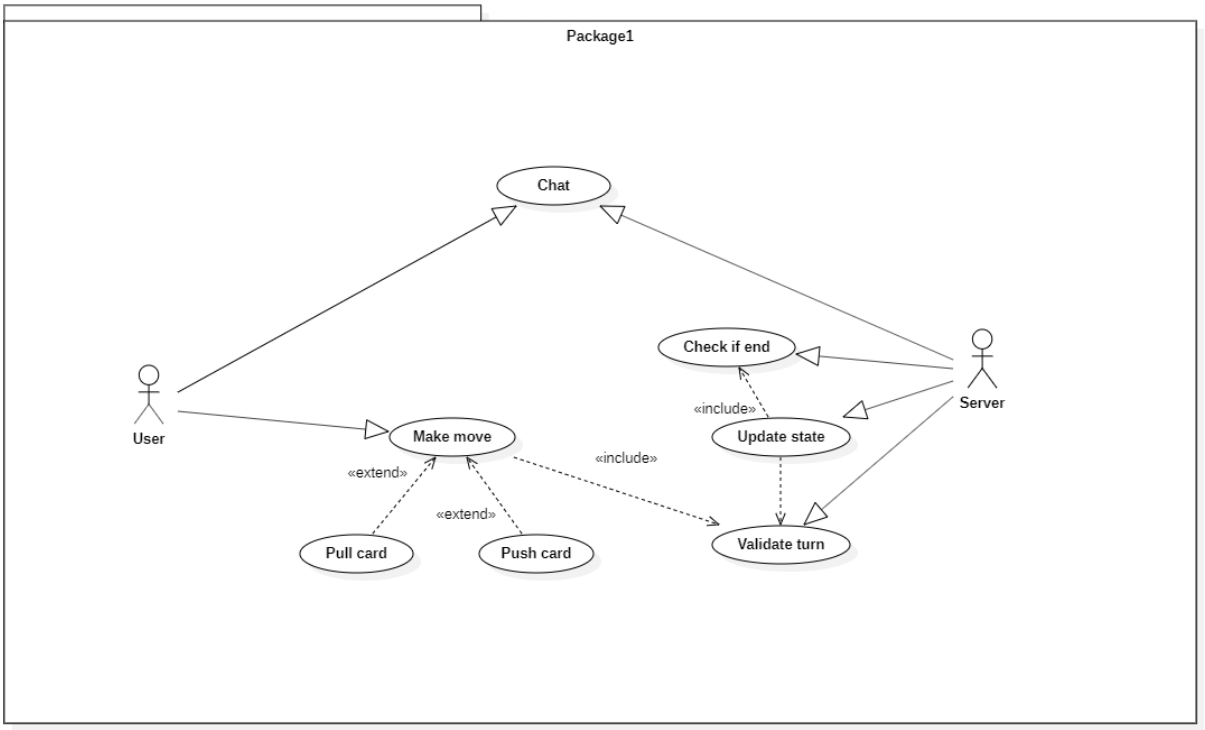
Arhitekturni značajni slučajevi korišćenja

Funkcionalni zahtevi web aplikacije Ena:

- Registracija korisnika
- Prijavljivanje korisnika
- Dodavanje prijatelja
- Prihvatanje zahteva za prijateljstvo
- Kreiranje igre
- Pridruživanje igri
- Inicijalna raspodela karata igračima
- Omogućavanje odigravanja poteza
- Mešanje zajedničkog špila
- Kraj igre
- Razmena poruka izmedju igrača



Log in, register, create and join game use case diagram



Play use case diagram

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. Treba omogućiti da aplikacija bude dostupna korisnicima 24h dnevno
- **Skalabilnost**: Potrebno je obezbediti adekvatan i efikasan rad centralizovane baze podataka sa povećanjem količine podataka koju skladišti, koja nastaje usled povećanja broja korisnika
- **Lakoća korišćenja**: Korisnički interfejs treba da bude dovoljno intuitivan za korisnike sistema, kako bi se povećala korisnost aplikacije
- **Performanse**: aplikacija treba da obezbedi što manje vreme odziva i najbolje performanse u zavisnosti od trenutnog broja korisnika
- **Sigurnost**: Obezbediti autentifikaciju i autorizaciju, kao i enkripciju osetljivih podataka. Klasicni SQL Injection ne sme probiti zastitu servera.

Tehnicka i poslovna ograničenja

- **Optimizacija za web čitače** – Google Chrome, Mozilla Firefox, Opera
- **Notifikacija događaja** – obaveštenje kada korisnik preduzme specifičnu akciju, npr. kad korisnik promeni smer
- **Apstrakcija podataka** – važno je da interna organizacija baze podataka(šema) bude sakrivena od strane API-ja

Arhitekturni dizajn

Dati su arhitekturni obrasci korišćeni u projektovanju arhitekture aplikacije Ena, generalna arhitektura i strukturni i bihevioralni pogled na aplikaciju.

Arhitekturni obrasci

Za projetovanje arhitekture TheScientist aplikacije biće korišeni arhitekturni obrasci: Layered, MVC, Repository i Publish/Subscribe.

- Layered Arhitektura: Arhitekturni obrazac koji forsira podelu strukture aplikacije u slojeve. Svaki sloj u arhitekturi formira apstrakciju oko posla koji treba da se uradi da bi se zadovoljio određeni poslovni zahtev. Kod projektovanja aplikacije TheScientist biće korišćena troslojna arhitktura, standard za projektovanje web aplikacija. Tri sloja su : klijentski, serverski i sloj podataka(sloj baze podataka).
  - 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: Zahtevi primljeni sa klijentskog sloja se prihvataju i šalju dalje odgovarajućem agentu. Ovaj sloj će klijentu omogućiti sinhronu komunikaciju preko RESTful API poziva i asinhronu komunikaciju pomoću message brokera.
  - Sloj baze podataka: Tokom rada ovako projektovane aplikacije (komunikacija klijenta i srvera) nastaće određeni podaci koje će biti neophodno očuvati radi omogućavanja normalne funkcije programa. Ovaj sloj će omogućiti trajnu perzistenciju ovako nastalih podataka.

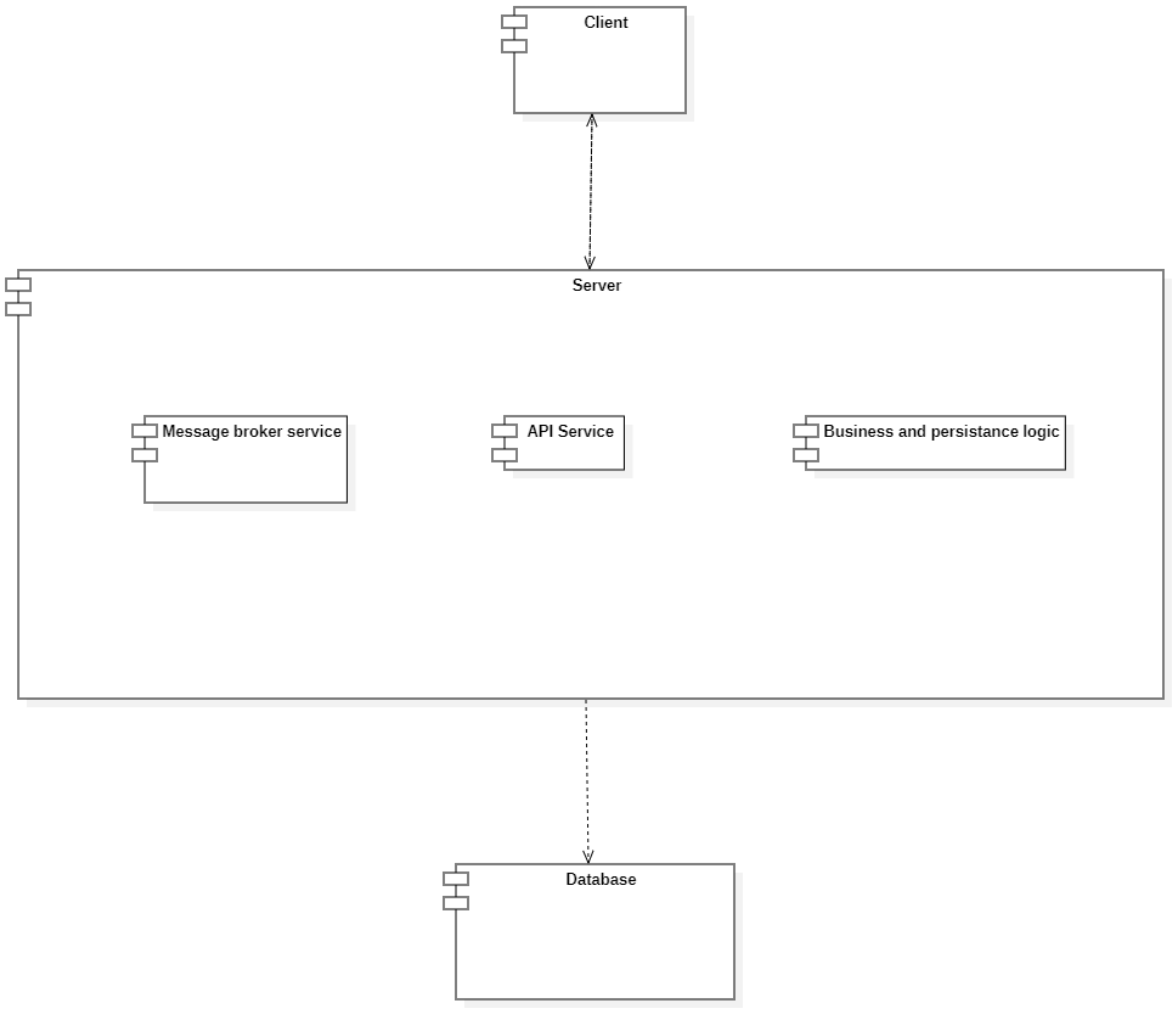
- MVC (Model-View-Controller): Arhitekturni obrazac koji odvaja aplikaciju na tri glavne logičke komponente: model, pogled i kontroler. Svaka od ovih komponenti rukuje specifičnim razvojnim aspektima aplikacije. Ovaj obrazac u aplikaciji Ena će biti upotrebljen za projektovanje RESTful API-ja (sinhrona komunikacija sa klijentom).
  - Model: Centralna komponenta obrasca, dinamička struktura podataka aplikacije, nezavisna od korisničkog interfejsa. Ova komponenta se bavi perzistencijom domenskih klasa.
  - View: Bilo koji grafički prikaz podataka koji su projektovani u modelu. Ova komponenta u aplikaciji Ena neće biti implementirana tako da serverska strana kreira i vraća delove interfejsa, već će se interfejs implementirati korišćenjem JavaScript frameworka React, koji će od servera dobijati podatke u JSON formatu.
  - Controller: Dobijeni input pretvara u komande koje se koriste za model ili view. Ova komponenta obezbeđuje korišćenje usluga samog RESTful API-ja, čije funkcije poziva klijent, funkcija zatim vrši obradu ulaznih podataka i podataka iz modela i tako kreira izlaz u vidu JSON objekata koje klijent lako može da parsira i koristi za svoje potrebe.

- Repository: Arhitekturni obrazac koji aplikaciji omogućava podelu na repozitorijume, klase ili komponente koji obuhvataju logiku potrebnu za pristup izvorima 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 Entity Framework ORM). 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 proizvodi (publisher) i pretplatnika (subscriber). Publisher i Subscriber se oslanjaju na message brokera, koji poruke prenosi od proizvođača do pretplatnika.
  Ovaj obrazac biće korišćen radi predavanja asinhronih poruka klijentu od strane servera, putem message broker komponente.

Generalna arhitektura

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



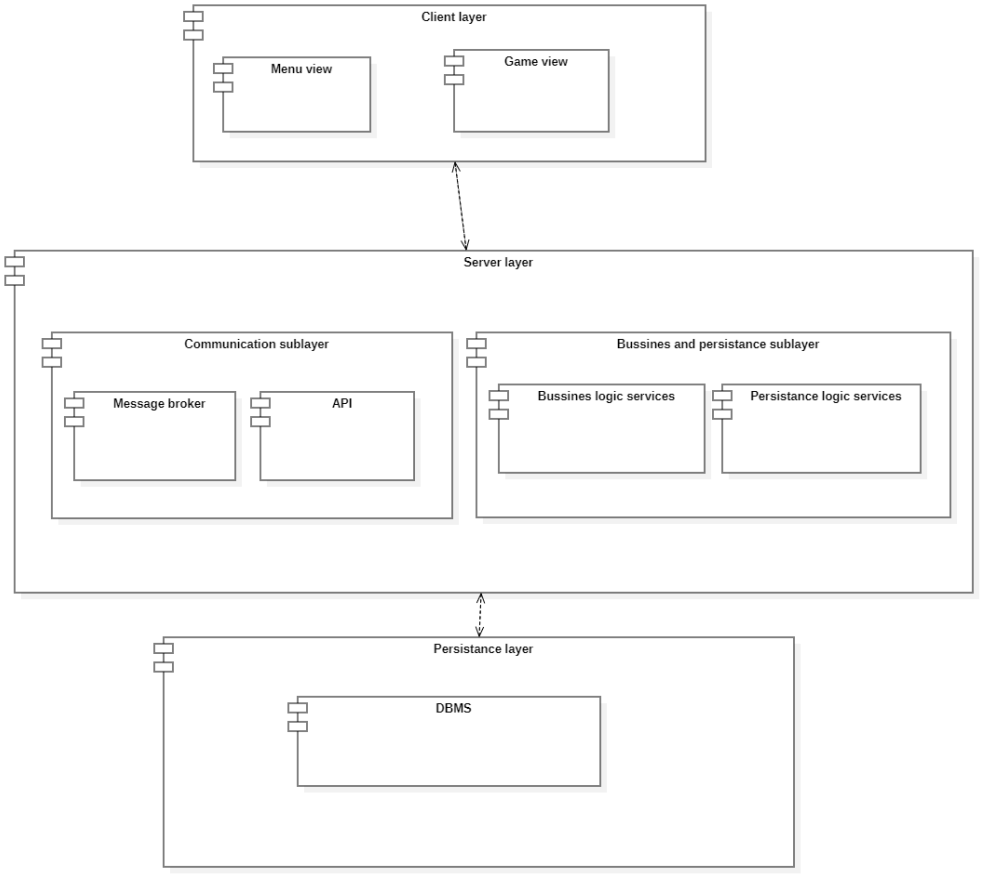
Strukturni pogled

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.

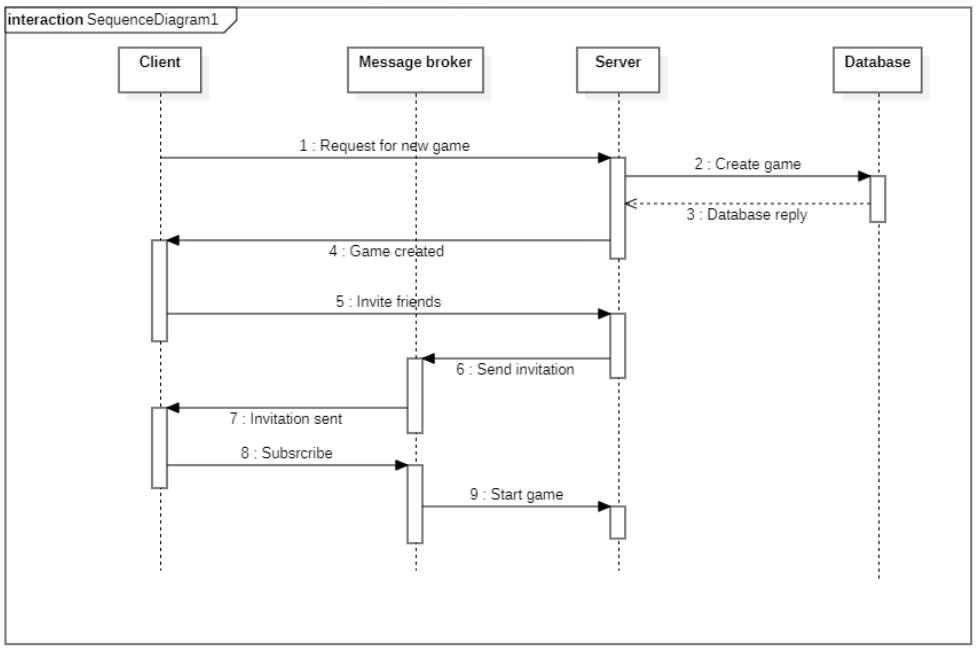
Komunikacioni podsloj obuhvata RESTful Web API za sinhronu komunikaciju i Message Broker za asinhronu komunikaciju sa klijentom.

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

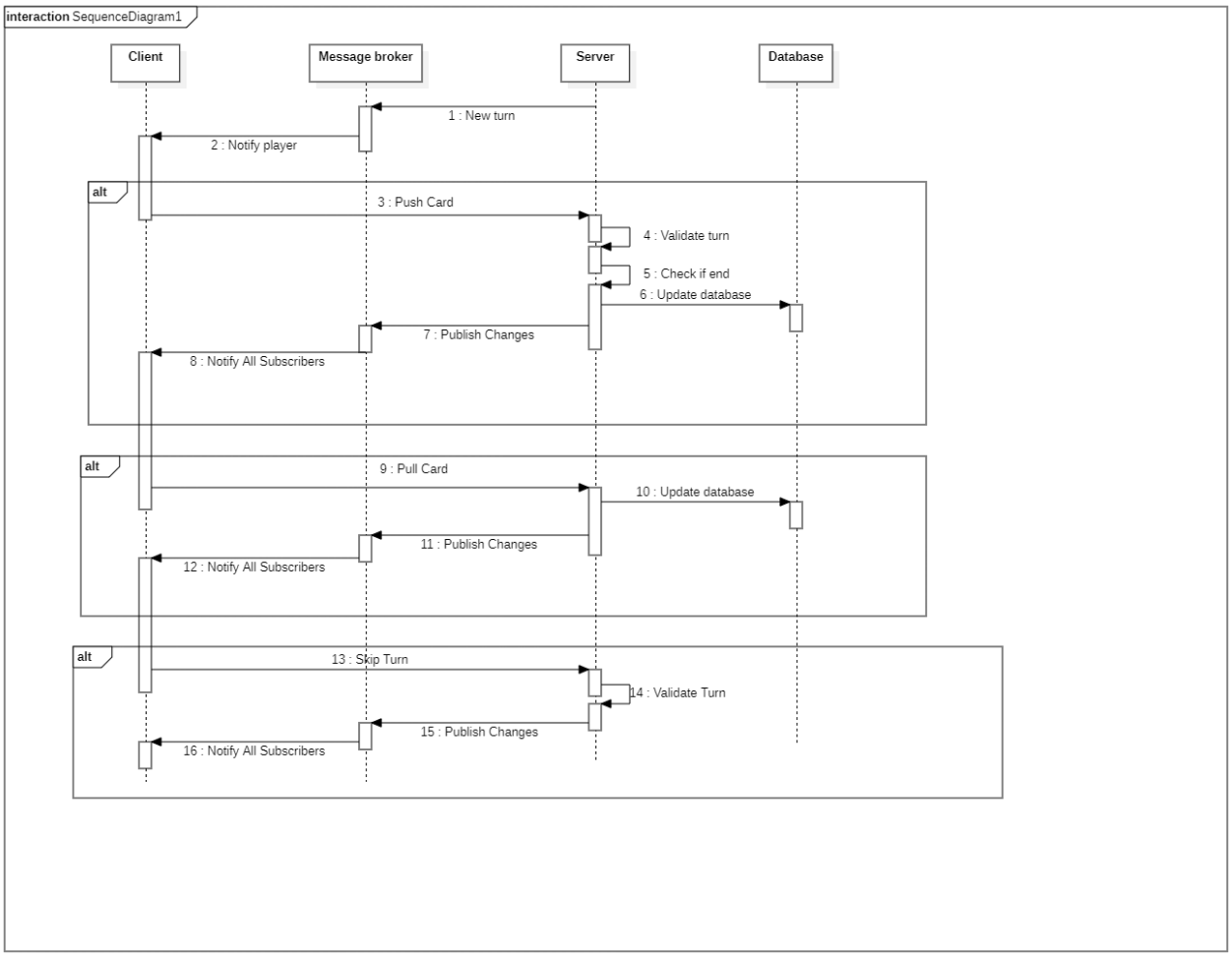


Bihevioralni pogled

- **Nova Igra**



- **Odigravanje poteza**



Implementacija

- **React** - klijentska strana aplikacije
  Razvojna platforma, izgrađena na Typescript-u, uključuje kolekciju dobro integrisanih biblioteka koje pokrivaju širok spektar funkcija, uključujući rutiranje, upravljanje obrascima, komunikaciju klijent-server itd.
- **ASP.NET Core** - serverska strana aplikacije
  Open-source okvir za kreiranje modernih aplikacija. Dizajniran je tako da obezbedi optimizovani razvojni okvir za aplikacije koje se postavljaju u oblak ili se pokreću lokalno.
- **MS SQL Server** - relaciona baza podataka
- **Signal R** - biblioteka za uspostavljanje real-time komunikacije
- **Entity Framework** - okvir za objektno-relaciono mapiranje