Planet Wars

Članovi tima:

Đorđe Jovanović, 17150
Nevena Tufegdžić, 17506

Naziv tima:

Negatim

Datum: 16.01.2022.

SADRŽAJ

1 MODEL KOMUNIKACIJE

1.1 SIGNALR IMPLEMENTACIJA

Za ostvarivanje komunikacije u realnom vremenu, kao i propagaciju odigranog poteza u igri, korišćen je *Microsoft SignalR message-broker*. Kao impementacija *SignalR*-a u okviru sistema, kreirane su dve klase: *MessageHub* i *GameService*.

1.1.1 MessageHub klasa

Ova klasa služi kao ekstenzija *SignalR Hub* klase. Sadrži sve metode koje su potrebne za realizaciju *real-time* komunikacije, koje se koriste pri pružanju *chat* usluge i odigravanja igre.

1.1.2 GameService klasa

Ova klasa implementira *IHubService* interfejs koji pruža *SignalR*, a njena svrha jeste da se iskoriste *SignalR* pogodnosti radi slanja novog stanja igre svim učesnicima, kao i propratnih akcija koje su uključene u odigravanje igre.

1.1.3 CommunicationParam klasa

Svi objekti koji se šalju kroz *Hub* imaju osnovni tip *CommunicationParam*. Ideja ovakvog pristupa jeste da svaka poruka koja se šalje od klijenta ka serveru mora da ima ID klijenta, kao i ime *handler* metode na klijentu koja se poziva kako bi klijenti primili poruku. Pored toga, objekat može imati dodatne parametre, koji su definisani kroz klasu koja je nasleđuje.

Ovakvim pristupom, osigurava se da ukoliko dođe do promene u broju parametara koji se traže na serveru, ne dolazi do „pucanja“ klijenta iz razloga što je dovoljno samo dodati jedan atribut Param klase. Ako klijent ne pošalje neki parametar, smatra se da je on postavljen na null. Sa druge strane, ovaj pristup takođe omogućava bolju čistljivost koda i lakšu skalabilnost. Potencijalno poboljšanje ogledalo bi se u zameni nasleđivanja kompozicijom.

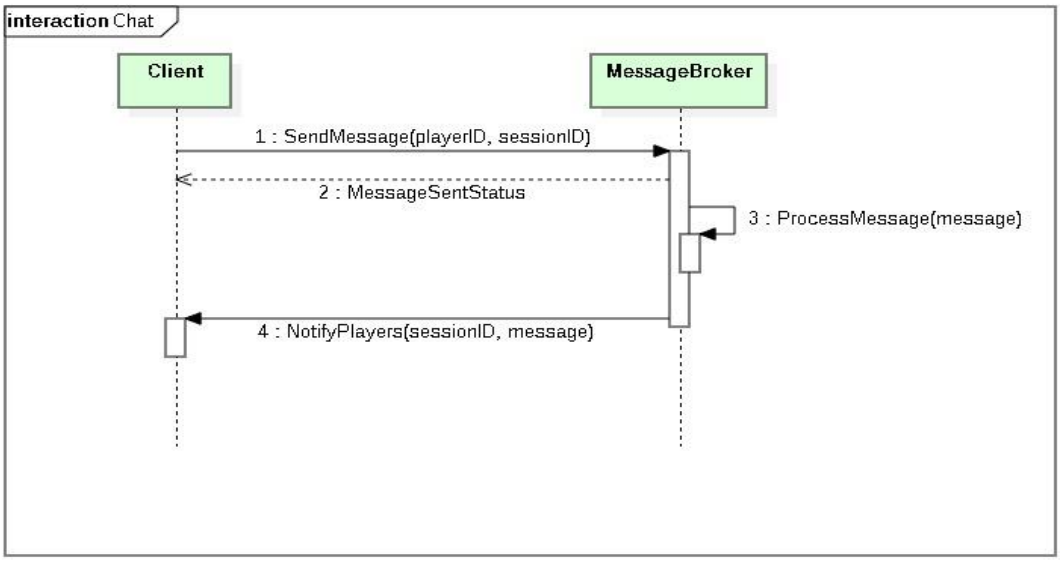
2 SERVISI

2.1 CHAT

Chat-u se pristupa samim pridruživanjem igri. Svaka igra sadrži *chat* grupu, koju modeluje *SignalR* grupa, a pomoću koje se šalju i primaju poruke. *SignalR* metode obezbeđuju da poslatu poruku primaju samo zainteresovane strane, odnosno korisnici u okviru date grupe. Ukoliko korisnik napusti grupu ili je igra završena, nema mogućnost slanja i primanja poruka u okviru te grupe. Korisnik napušta grupu zajedno sa napuštanjem same igre.

Objekti koji se šalju kroz *chat* jesu *MessageDto* tipa, koji nasleđuje *CommunicationParam* klasu. Pored ID-ja korisnika, koji je potreban kako bi se korisnik locirao u grupi, šalje se i ID sesije u okviru koje korisnik šalje poruku. Na taj način, grupa je jedinstveno određena identifikatorom sesije, a korisnik u njoj sopstvenim identifikatorom.

Korisnik šalje poruku grupi tako što kreira konekciju ka *chat endpoint*-u, nakon čega definiše lokalni *handler* koji treba da se iskoristi pri prijemu poruke. Najzad, klijent startuje konekciju i može pozvati neku od metoda iz *MessageHub*-a funkcijom *invoke()*, kojoj prosleđuje potrebne parametre.



2.2 PROPAGACIJA STANJA IGRE

Kako bi svi korisnici u jednoj sesiji pravovremeno dobili informacije o stanju igre od servera, implementiran je *GameService* koji sadrži sve potrebne metode za slanje informacije o potezima, kao što su:

- ko je na potezu i koliko armija je dobio
- ko je odigrao potez, i kakav potez
    - da li je neki igrač pobeđen
    - da li je neka planeta osvojena, itd.

Ova klasa implementirana je kao servis koji se može *inject*-ovati u *SessionController*. Kada igrač odigra potez, potez biva obrađen i potrebno je pozvati metodu servisa koja će obavestiti ostale igrače u sesiji o posledicama. Zatim, potrebno je sračunati koji igrač igra sledeći i koliko armija dobija, pa je zatim potrebno poslati te informacije svim igračima kako bi znali šta da očekuju od konkretnog igrača na potezu.