Model komunikacije

TravelPlan

Članovi tima:

Petar Trifunović 16955
Maša Nešić 16774

Naziv tima:

Muffin Time

Sadržaj:

ASP.NET Core SignalR

SignalR je Microsoft-ova biblioteka koja se koristi radi ostvarivanja komunikacije u realnom vremenu u web aplikacijama. U *TravelPlan* projektu koristi se verzija SignalR biblioteke prilagođena ASP.NET Core framework-u.

Pomoću *RPC* (*remote procedure call*) principa koji SignalR implementira, serverska i klijentska strana ostvaruju komunikaciju, tako da jedna strana ima mogućnost poziva odgovarajućih metoda druge. Protokol koji se koristi radi otvaranja dvosmernog kanala između klijenata i servera jeste *websocket* protokol. Ovaj protokol predstavlja svojevrsnu nadogradnju HTTP protokola, a njegova posebna efikasnost se ogleda u činjenici da ne zahteva otvaranje velikog broja TCP konekcija, s obzirom na to da jedna konkretna websocket konekcija, zahteva samo jednu TCP konekciju. Ukoliko ipak neka strana komunikacije iz bilo kog razloga ne podržava websocket protokol, SignalR će automatski preći na korišćenje nekog od manje naprednih protokola. Konkretno, prvo će pokušati sa korišćenjem *Server-Sent Events* tehnologije, a, u krajnjem slučaju, koristiće se *Long Polling*.

SignalR biblioteka serverske strane između ostalog sadrži i klasu *Hub*. Ova klasa predstavlja API visokog nivoa, koji olakšava implementaciju SignalR komunikacije, pružajući svoje metode i propertije na korišćenje programerima, čime se sakrivaju svi detalji komunikacije na nižem nivou. Zarad korišćenja ovog API-ja, potrebno je na serveru kreirati novu klasu koja će naslediti *Hub*. U novoj klasi definišu se metode koje se mogu pozivati sa udaljene mašine, odnosno iz klijentskog koda, korišćenjem klijentskog SignalR API-ja, navođenjem imena metode koju je potrebno pozvati, i prosleđivanjem odgovarajućeg broja parametara. Unutar metoda klase koja nasleđuje *Hub*, moguće je pozvati metode udaljenih klijenata, i takođe proslediti odgovarajuće parametre. Kao i serverske, i klijentske metode definisane su svojim imenom. Konkretno u javascript-u, imena klijentskih „metoda“ se mogu smatrati imenima događaja (*events*) koje klijent osluškuje i na njih reaguje.

API serverske strane nudi nekoliko načina za izbor klijenata čije će se metode pozivati sa servera. Što se tiče *TravelPlan* projekta, od interesa je pozivanje metoda određene grupe klijenata. Svaki poziv serverske metode od strane klijenta identifikovan je posebnim konekcionim ID-jem. SignalR nudi mogućnost kreiranja imenovanih grupa, i učlanjivanja kiljenata u njih, dodavanjem konekcionog ID-a u grupu. SignalR će, na način nevidljiv programerima, voditi evidenciju o tome koji konekcioni ID pripada kojoj grupi. Jedan ID se može naći u više grupa. Korišćenjem odgovarajućih metoda SingalR biblioteke, moguće je pozvati udaljene metode samo onih klijenata koji pripadaju odgovarajućoj grupi.

Pozivanje udaljenih metoda klijenata nije neophodno vršiti iz serverske klase koja nasleđuje *Hub*. Koristeći ASP.NET Core Dependency Injection, moguće je u bilo koju serversku klasu ubaciti instancu klase koja implementira IHubContext interfejs. Ovo je generički interfejs SignalR biblioteke, i potrebno ga je parametrizovati klasom koja nasleđuje *Hub*. Na ovaj način moguće je iz bilo kog dela serverskog koda pozivati metode klijenata, kao posledice izvršenja određenih akcija na serveru.

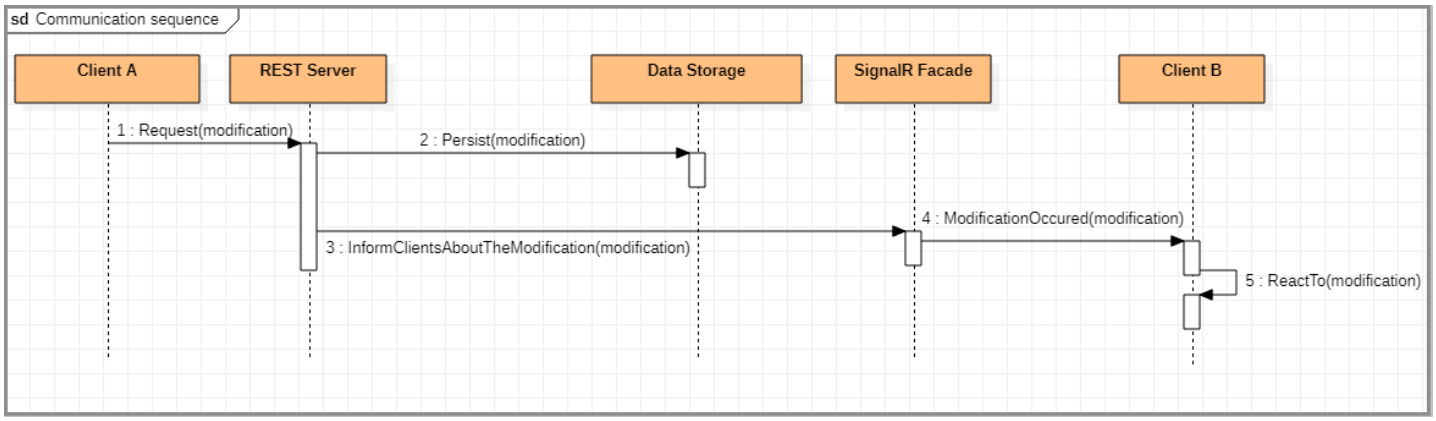
Da bi klijentu bio vidljiv *Hub* preko koga komunicira sa serverom, u konfiguraciji servera potrebno je definisati naziv endpoint-a preko koga će klijent moći da pribavi i koristi konekciju na odgovarajući *Hub*.

Komunikacija u realnom vremenu u aplikaciji

Kako bi se ostvarila komunikacija u realnom vremenu, klijentska aplikacija u odgovarajućem trenutku mora biti učlanjena u odgovarajuću SignalR grupu, kako bi korisnici dobijali informacije o onim dešavanjima u aplikaciji koji su im od interesa. Konkretno, u *TravelPlan* aplikaciji postoje tri vrste grupa:

1. **Grupe namenjene konkretnom putovanju –** Ove grupe namenjene su putnicima određenog putovanja. Svaki put kada korisnik otvori stranicu za pregled nekog putovanja, poziva se odgovarajuća metoda klase koja nasleđuje *Hub* klasu. Unutar ove metode, ID konekcije preko koje je metoda pozvana ubacuje se u grupu tom konkretnom putovanju. Imena ovih grupa kreiraju se po šablonu *„Trip" + ID_putovanja*. Klijent napušta ovu grupu čim se napusti stranica putovanja, i to takođe pozivom metode koja iz grupe izbacuje konekcioni ID.
2. **Grupe namenjene konkretnom korisniku –** Svaki korisnik postaje član po jedne ovakve grupe čim se prijavi na svoj nalog. Za svako putovanje mogu se dodavati stvari koje treba poneti na put, i svaka od njih se može dodeliti nekom putniku kao odgovornost. Kako bi putnici automatski o ovome bili obavešteni, svaki od njih će imati po jednu ličnu grupu, preko kojih će im stizati notifikacije o tome da im je nova stvar dodeljena kao zaduženje, kao i notifikacije o tome da prestaju da za neku stvar budu odgovorni, jer je prebačena drugom korisniku kao zaduženje. ID korisničke konekcije se, pri prijavljivanju na nalog, ubacuje u grupu čije ime poštuje šablon *„User" + ID_korisnika*. Iščlanjivanje iz grupe vrši se pri odjavljivanju sa naloga.
3. **Grupe namenjene konkretnom timu –** Korisnik postaje član timske grupe, za sve timove čiji je deo, za svaki period kada mu je u interesu da vidi promene načinjene nad njegovim timovima real-time. Ti periodi odnose se na vreme kada korisnik pregleda i koristi svoje timove. ID korisničke konekcije se, na početku svakog tog perioda, ubacuje u grupe čije ime poštuje šablon *„Team" + ID_tima* za svaki od timova čiji je deo. Iščlanjivanje iz grupe vrši se na kraju perioda potrebe informisanja u real-time.

Kada se konekcije klijenata ubace u grupe, klijenti počinju sa osluškivanjem događaja i reagovanjem na njih. Svi događaji koje klijenti osluškuju nastaju kao posledice korisničkih akcija koje, pre slanja poruke preko SignalR-a, menjaju stanje aplikacije, odnosno vrše neku perzistentnu promenu u bazi podataka. Na sledećem dijagramu prikazan je uopšten tok dešavanja u aplikaciji, koji nastaje pri nekoj korisničkoj akciji koja zahteva i učešće SignalR-a i komunikacije u realnom vremenu.



Korisnička akcija preko klijentske aplikacije dovodi do slanja zahteva za izmenom serveru, putem RESTful API-ja. Izmena stanja aplikacije čuva se u centralnoj bazi podataka, nakon čega se pozivaju metode fasadne klase za komunikaciju. Ova klasa od ostatka aplikacije skriva pozive SignalR metoda. Unutar ove klase, preko SignalR biblioteke, pozivaju se metode na udaljenim klijentima.

Potpisi funkcija fasadne klase kao parametar koji se prosleđuje klijentu primaju objekat tipa *object*. U zavisnosti od događaja koji je nastupio, u pozive funckija fasadne klase prosleđuju se objekti specijalizovanijih klasa.

Komunikacija pri akcijama koje ne zahtevaju obaveštenja u realnom vremenu

Određene korisničke akcije ne zahtevaju nikakav vid obaveštenja u realnom vremenu, pa ni učešće SignalR-a. Pri ovakvim akcijama, koristi se standardni RESTful princip. Klijentska strana upućuje API pozive ka serveru, čeka odgovor, i nakon njegovog pristizanja ažurira trenutno stanje aplikcaije na klijentu, što uključuje i ažuriranje korisničkog interfejsa.