PPKS projekt: Dokumentacija

Marcel Macukić
Fakultet elektrotehnike i računarstva
Sveučilište u Zagrebu

Svibanj 2025.

Sadržaj

1 Uvod

Ova dokumentacija opisuje razvoj web aplikacije za praćenje narudžbi u stvarnom vremenu. Aplikacija omogućuje korisnicima kreiranje i praćenje statusa svojih pošiljki, dok administratori mogu ažurirati statuse narudžbi. Komunikacija se odvija u stvarnom vremenu pomoću biblioteke SignalR koja koristi WebSocket protokol kada je dostupan.

2 Tehnologije

- ASP.NET Core (.NET 8)
- Entity Framework Core (EF Core)
- SignalR
- Auth0 (autentifikacija)
- SQLite (ili drugi relacijski DB)
- Razor View engine (bez SPA frameworka)

3 SignalR

SignalR je biblioteka koja omogućuje dvosmjernu komunikaciju između klijenta i poslužitelja u stvarnom vremenu. U ovom projektu koristi se za obavještavanje korisnika o promjenama statusa njihovih pošiljki odmah nakon što administrator izvrši promjenu.

3.1 Kako SignalR radi

SignalR automatski odabire najbolji dostupni protokol za komunikaciju. Najpoželjniji protokol je **WebSocket**, ali ako nije podržan, koristi se fallback na Server-Sent Events (SSE) ili Long Polling.

3.2 WebSocket protokol

WebSocket omogućuje trajnu dvosmjernu vezu između preglednika i poslužitelja. Za razliku od HTTP zahtjeva koji su jednokratni i stateless, WebSocket veza ostaje otvorena te omogućuje trenutnu razmjenu podataka bez potrebe za ponovnim slanjem zahtjeva.

3.3 Arhitektura SignalR-a i Hub model

SignalR se temelji na konceptu *Hub* objekata koji predstavljaju krajnje točke preko kojih klijenti i poslužitelj razmjenjuju poruke. Klijentska biblioteka se povezuje na url npr. `/orderHub` te poziva metode na Hubu. Na poslužitelju, Hub metode mogu pozivati metode na klijentu koristeći mehanizam proxyja.

- **Hub protokoli**: Podržani su JSON i binary MessagePack. Klijent u `negotiate` zahtjevu navodi koji protokol preferira.

- **Grupe**: SignalR omogućuje logičko grupiranje konekcija (npr. grupa po ID-u narudžbe) pa server može slati poruke samo relevantnim klijentima.
- **Skaliranje**: Za skaliranje na više instanci koristi se backplane (Redis, Azure SignalR Service) koji replicira poruke između čvorova.

3.4 Slijed uspostave WebSocket veze sa SignalR-om

1. **Negotiation**: Klijent šalje HTTP POST na `/orderHub/negotiate`. Server vraća listu podržanih transporta i token za autorizaciju.
2. **WebSocket handshake**: Klijent otvara HTTP GET zahtjev s headerima `Upgrade: websocket` i `Connection: Upgrade`. Server odgovara statusom 101 (Switching Protocols).
3. **Establish Hub protocol**: Nakon uspješnog handshaka, klijent šalje initial payload s odabranim Hub protokolom (JSON/MessagePack).
4. **Dvosmjerna razmjena**: Poruke se enkapsuliraju u SignalR frame-ove (length-prefixed) i šalju kroz WebSocket kanal u oba smjera.

Ako WebSocket nije dostupan (firewall, preglednik, TLS termination), SignalR automatski prelazi na SSE ili Long Polling bez potrebe za dodatnom konfiguracijom.

3.5 Upravljanje vezom i pouzdanost

- **Keep-alive i ping**: Klijent i server periodično šalju ping okvire radi otkrivanja prekinutih veza.
- **Reconnection**: Ako veza pukne, SignalR pokušava ponovo uspostaviti WebSocket; u međuvremenu razmjena poruka je pauzirana.
- **Back-pressure**: Unutar Huba moguće je limitirati brzinu slanja poruka kako bi se spriječio memory pressure.

3.6 Sigurnost

Komunikacija se odvija preko HTTPS-a (TLS) na portu 443, a autentifikacija se provodi putem Auth0 JWT tokena koji se šalju kao Bearer header tijekom `negotiate` faze te svakog reconnection pokušaja.

3.7 Primjena u projektu

Pri svakoj promjeni statusa narudžbe, server poziva metodu `Clients.Group(orderId).SendAsync("St` Klijentski JavaScript registrira handler za `StatusChanged` i ažurira UI bez refreshanja stranice.

4 Izvorni kod

Na GitHub repozitoriju moguće je pronaći sve izvorne datoteke, konfiguracije te upute za pokretanje aplikacije.

Izvorni kod projekta dostupan je na sljedećoj poveznici:

- `https://github.com/Mmacukic/PPKS.git`

5 Instalacija aplikacije

5.1 Preduvjeti

- .NET SDK 8.0 ili noviji
- SQLite (ako se koristi lokalno)
- Visual Studio / Rider / VS Code

5.2 Koraci

1. Klonirati repozitorij:

```
git clone https://github.com/Mmacukic/PPKS.git
cd ppks-projekt
```

2. Pokrenuti migracije baze:

```
dotnet ef database update
```

3. Pokrenuti aplikaciju:

```
dotnet run
```

4. Otvoriti aplikaciju u pregledniku:
   `https://localhost:5001`

6 Primjer korištenja SignalR-a u projektu

Kod koji instancira SignalR hub i šalje promjene statusa izgleda otprilike ovako:

```
await _hubContext.Clients
    .Group(userEmail)
    .SendAsync("StatusUpdated", shipmentId, newStatus);
```

Na klijentskoj strani koristi se JavaScript koji sluša promjene:

```
connection.on("StatusUpdated", (shipmentId, newStatus) => {
    // A uriraj UI
});
```

7 Zaključak

SignalR je ključna komponenta ove aplikacije jer omogućuje dvosmjernu komunikaciju i obavještavanje korisnika u stvarnom vremenu o statusima pošiljki. Kombinacija .NET Core, EF Core i SignalR pruža robusno i proširivo rješenje za ovaj tip problema.