Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.42:004.738.12
DOI: https://doi.org/10.24867/28BE44Milanko

RAZVOJ VEB APLIKACIJE U REALNOM VREMENU NA PRIMERU CHAT-a

DEVELOPMENT OF A WEB APPLICATION IN REAL TIME USING THE EXAMPLE OF CHAT

Alisa Milanko, *Fakultet tehničkih nauka, Novi Sad*

Oblast – RAČUNARSTVO I AUTOMATIKA

Kratak sadržaj – *Veb aplikacija Chat predstavlja aplikaciju za razmenu poruka između dva korisnika u realnom vremenu. Pored osnovne funkcionalnosti omogućava registrovanje i prijavljivanje korisnika, kao i kreiranje i izmenu ličnog profila. Za implementaciju korišćena je biblioteka SignalR.*

Ključne reči: *Veb u realnom vremenu, SignalR, ASP.NET Core*

Abstract – *The web application Chat is an application for exchanging messages between two users in real time. In addition to basic functionality, it enables user registration and login, as well as creation and modification of a personal profile. The SignalR library was used for implementation.*

Keywords: *Realtime web, SignalR, ASP.NET Core*

1. UVOD

Za razliku od desktop aplikacija koje se instaliraju na nekom lokalnom računaru, veb aplikacije su aplikacije koje se izvršavaju na nekom udaljenom računaru. Takvoj aplikaciji pristupa korisnik, odnosno klijent, sa svog računara korišćenjem interneta i pretraživača. Veb server predstavlja računar na kome se izvršava aplikacija. U ovom radu govorićemo o veb aplikacijama koje rade u realnom vremenu, odnosno o komunikaciji u realnom vremenu. Veb je u svojoj ranoj fazi postojanja bio vrlo različit od onoga što danas poznajemo. Jedan od nedostataka bio je taj da je veb stranicu bilo potrebno ponovo učitati kako bi se dobili novi podaci.

Mogućnost dobijanja podataka bez ponovnog učitavanja stranice drastično je promenila način na koji danas koristimo veb. Pod komunikacijom u realnom vremenu smatra se komunikacija dva ili više korisnika računara u kojoj su za vreme komunikacije svi korisnici na mreži.

Danas jedan od glavnih načina pomoću kojih komuniciramo jesu društvene platforme poput Facebook-a, Twitter-a i Google+-a. Ove platforme su izgrađene na veb tehnologijama u realnom vremenu.

Chat je, nesumnjivo, jedan od najpopularnijih, ali i najspecifičnijih vidova komunikacije putem interneta. Reč je o novom obliku komuniciranja, različitom od standardnog govornog i pisanog jezika.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Željko Vuković, docent.

Dok je jezik Email-a mnogo bliži jeziku pisanih žanrova i u nekim svojim karakteristikama podseća na pisanje klasičnih pisama, jezik Chat-a najviše je sličan živom govoru, iako se i on javlja u pisanoj formi.

2. VEB KOMUNIKACIJA U REALNOM VREMENU

Realno vreme (eng. real time), ponekad prevedeno i kao potrebno vreme, može se objasniti kao najduže vreme koje je potrebno za izvršavanje nekog zadatka. Ovaj pojam često se, donekle pogrešno, koristi za vreme u kojem zadatak mora biti izvršen kako bi se ljudima činilo da je izvršen odmah, uglavnom zbog velike količine aplikacija kojima danas to i jeste realno vreme. Realno vreme može biti bilo koje trajanje u zavisnosti od potreba aplikacije. Potrebe veb aplikacija su se znatno izmenile u prvoj deceniji XXI veka. Korišćenje interneta i veb pretraživača (eng. browser) postalo je sinonim, jer se cela paradigma odvija kroz veb pretraživač. Ljudi ih koriste kao primarnu platformu kako bi odradili sve zahteve koje potražuju na internetu. Veb je podeljen na dva različita tipa: statički i dinamički. Statički veb prikazuje isti sadržaj za svakog korisnika. Oni su obično sa fiksnim brojem stranica koje imaju određeni izgled. Kada se stranica pokreće u pregledaču, sadržaj je statičan i ne menja se kao odgovor na radnje korisnika.

Oni jesu ograničeni, ali ih je vrlo jednostavno napraviti. Sa druge strane, dinamički veb, može da prikaže različit sadržaj i pruža korisničku interakciju korišćenjem naprednog programiranja i baze podataka. Popularne socijalne mreže su izgrađene na vrhu web browser platformi i njih danas koristi više od milijardu korisnika u svetu. Glavni razlog popularnosti socijalnih mreža jeste pružanje informacija i poruka od drugih korisnika u realnom vremenu. Sve ovo je dovelo do praktičnog prestanka korišćenja statičkog veba i dovelo do početka razvoja dinamičkih veb aplikacija koje operišu u realnom vremenu.

2.1. HTTP tehnike

HTTP (Hypertext Transfer Protocol) je mrežni protokol koji se koristi za virtuelno dostavljanje podataka (resursa) na vebu. Ovi podaci obično predstavljaju HTML fajlove, slike, rezultate upita ili nešto drugo. Web browser je HTTP klijent koji šalje zahtev (eng. request) prema HTTP serveru koji potom vraća odgovor (eng. response) prema klijentu koji sadrži resurse. Standardni port na kojem HTTP server osluškuje request-e je -80, ali se mogu koristiti i ostali portovi. HTTP se temelji na stalno otvorenoj vezi između klijenta i servera za vreme razmene poruka. Klijent šalje zahtev serveru koji ga obrađuje i

šalje odgovor. Nakon što klijent primi odgovor, veza između klijenta i servera se prekida. Za svaku sledeću razmenu poruke, odnosno da bi se načinile izmene na veb stranici, potrebno je opet uspostaviti vezu. Ovo dovodi do statičkih veb stranica koje zahtevaju da se browser osveži (eng. refresh) kako bi nove promene bile prikazane. Ova tehnika se zove „serverski pull"(klijent pull-uje podatke sa servera svaki put kada pošalje novi HTTP request.

2.1.1. AJAX

U klasičnoj veb aplikaciji kada korisnik inicira određenu akciju na primer klikom na hyperlink šanje se HTTP zahtev, koji se šalje od strane veb pretraživača na procesuiranje prema serveru. Veb server procesuira request i šalje nazad response, koji sadrži celu veb stranicu za prikazivanje u korisnikovom web browser-u. Dok se ovo događa, korisnik čeka odgovor i njegova interakcija sa aplikacijom je blokirana. Osvežavanje cele veb stranice nije uvek potrebno, na primer prilikom validacije podataka, a rešenje za to nudi AJAX. Ovo rešenje je bazirano na modelu koji dopušta da se slanje zahteva i primanje odgovora dešava u pozadini. Dakle, rešenje koje zahteva real-time vreme izvršavanja se odvija u pozadini, a aktivnosti koje zahtevaju ogromnu količinu podataka (kao što je čekanje odgovora koji sadrži kompletnu veb stranicu) odvijaju se normalno, sa osvežavanjem stranice [1].

2.1.2. Polling

Glavno oružje za izgradnju veb aplikacija, koje sve više zamenjuju desktop aplikacije, jeste taj što AJAX omogućava učitavanje samo delova stranica bez osveživanja čitavog njenog sadržaja. AJAX konekcija prema serveru nije konstantno otvorena, tako da ukoliko aplikacija zahteva konstantnu promenu podataka, AJAX skripta nema način da sazna da li je došlo do promene podataka. Iz tog razloga potrebno je konstantno slati AJAX upite serveru da li je došlo do promene podataka.

Polling predstavlja najjednostavniju tehniku. Reč polling se može prevesti kao anketiranje, jer klijent kontinuirano anketira server [2].

2.1.3. Long Polling

Long polling tehnika se zasniva na rešenju koje vodi poreklo od desktop aplikacija. Reč je o otvaranju live konekcije između servera i klijenta. Ovom tehnikom klijent šalje zahtev prema serveru ali za razliku od klasične polling tehnike, server ne odgovara odmah. Tačnije, ostvaruje se live konekcija između klijenta koji je poslao zahtev, i servera, sve dok se ne dogodi događaj koji inicira da server vrati odgovor klijentu da je potrebno da se ažurira. Nakon primanja odgovora, klijent se ažurira i ponovo šalje zahtev serveru, i server čeka na novi događaj kako bi odgovorio klijentu. Potrebno je uvideti da se ovom tehnikom postiže manji broj zahteva od strane klijenta prema serveru. Ne postoje odgovori koji ne sadrže nikakvu informaciju, kao u tehnici klasičnog polling-a [3].

2.1.4. Forever Frame

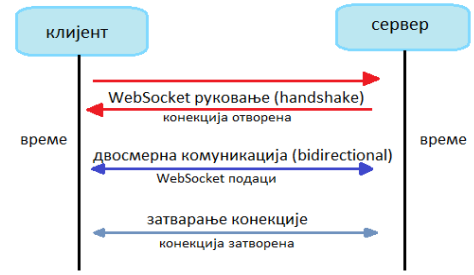
Forever Frame (Hidden iFrame) predstavlja najjednostavniju tehniku za izgradnju dinamičkih veb aplikacija pomoću http streaming-a. Ideja je da se ugradi skriveni HTML element – iFrame. On se postavi da bude nevidljiv, unutrašnji prozor unutar glavnog prozora koji dozvoljava da se uveže jedan HTML dokument unutar drugog HTML dokumenta. iFrame se šalje prvi put kada klijent pošalje zahtev za veb stranicu serveru. Beskonačan je pa otuda i naziv forever frame. Kada se događaj koji zahteva klijent dogodi, server šalje script tagove u skriveni iFrame i ovaj iFrame se polako napuni tagovima. Ti script tagovi predstavljaju JavaScript naredbe koje pozivaju određene funkcije i unutar kojeg je ugrađen skriveni iFrame u kome se nalaze script tagovi. Svaki script tag se izvršava čim bude poslat od strane servera, a primljen od strane klijenta kroz konekciju koja je i dalje otvorena.

2.1.5. Server Sent Events (SSE)

Server Sent Events je još jedna od tehnika za serverski push podataka prema klijentu. Predstavlja jednostavan i efikasan način za slanje ažuriranih podataka u realnom vremenu od servera prema klijentu, pomoću HTTP veze. On je deo HTML5 specifikacije i podržavaju ga svi moderni veb pretraživači. Zasniva se na jednosmernom toku podataka, gde server šalje poruku klijentu, ali klijent ne može da šalje poruku nazad serveru [4].

2.2. Web Socket

Web socket-i predstavljaju HTML5 specifikaciju koja pruža potpunu dvosmernu (eng. duplex) konekciju između klijenta i servera koja razmenjuje podatke pomoću jedinstvenog socket-a preko veba. Moguće je slati neograničen broj zahteva tako dugo dok je veza otvorena i to sa vrlo malim troškom u resursima aplikacije. Veza koja se otvara WebSocket protokolom je takve prirode da dopušta klijentu i serveru da, nezavisni jedan od drugog, šalju poruke jedan drugom, čak i u isto vreme. Takođe, zbog malog trošenja resursa poput radne memorije, server može imati u isto vreme otvoren veliki broj veza prema više klijenata, ako očekujemo da aplikaciju koristi veliki broj korisnika istovremeno.



Slika 1. – *WebSocket komunikacija*

2.2.1. WebSocket Handshake

Kako bi se uspostavila WebSocket konekcija između klijenta i servera, i klijent i server se nadograđuju sa HTTP protokola na WebSocket protokol i ovo se naziva inicijalno rukovanje klijenta i servera (eng. handshake) [5].

3. ASP.NET SIGNALR

Danas se od veb aplikacija mnogo očekuje. Prvo, ne samo da pravilno funkcionišu, već to moraju da rade sa odličnim korisničkim iskustvom. Pored toga, zahteva se da budu veoma brze, te da se informacije isporučuju u realnom vremenu.

Prilikom kreiranja veb aplikacije treba uzeti u obzir nekoliko stvari. Aplikacija ima tendenciju da postane veoma poznata kada korisniku omogući interakciju sa drugim korisnicima ili sa ugrađenom bazom podataka. Preduzeće ima tendenciju da poboljša svoju korisničku bazu ako može da razvije aplikaciju koja će pomoći klijentima da komunicira sa njima.

Dobro dizajnirana veb aplikacija takođe može pomoći da se minimiziraju troškovi i vreme. Na primer, kompanija koja se bavi online prodajom može da napravi aplikaciju koja ažurira svoje cene na mreži i šalje poruku zainteresovanim kupcima o promenama cena. Ova vrsta aplikacije će uštedeti ogromnu količinu vremena. Tako veb aplikacija može pomoći preduzeću da redovno bude u kontaktu sa svojim klijentima.

Aplikacija bi trebalo da radi besprekorno na svim platformama. Da bi se to postiglo aplikacija mora da bude kreirana koristeći neke napredne platforme kao što je SignalR.

ASP.NET SignalR je biblioteka otvorenog koda koja omogućava programerima da pojednostave dodavanje veb funkcionalnosti aplikacijama, u realnom vremenu. Takođe, omogućava kodu na strani servera da trenutno prosleđuje sadržaj klijentima.

Dobri kandidati za upotrebu su aplikacije koje zahtevaju visokofrekventna ažuriranja sa servera, na primer: igre, društvene mreže, aukcije, mape i GPS aplikacije. Zatim, kontrolne table i aplikacije za praćenje, kolaborativne aplikacije i aplikacije koje zahtevaju obaveštenja poput društvenih mreža, e-mail-a, apliakcije za ćaskanje itd.

ASP.NET Core SignalR održava mnogo istih osnovnih koncepata i mogućnosti kao ASP.NET SignalR. Glavne razlike između njih su [6]:

- Automatsko ponovno povezivanje (eng. Automatic Reconnects) – klijentska aplikacija mora eksplicitno da pokrene novu vezu kada je to potrebno.
- Razlike kod klijenta – ASP.NET Core SignalR klijent je napisan u TypeScript-u. Može se koristiti JavaScript ili TypeScript klijentska biblioteka bez obaveznog upućivanja na jQuery.
- Skaliranje (eng. Scaleout) – Sada postoji mala razlika u skaliranju od kada je Azure SignalR Service počela da podržava ASP.NET.
- Podrška protokolu (eng. Protocol Support) – ASP.NET Core SignalR podržava JSON protokol, koji nije kompatibilan sa ranijim verzijama, i novi binarni protkol zasnovan na MessagePack-u. Takođe, omogućava kreiranje novih protokola.
- Ubrizgavanje zavisnosti (eng. Dependency Injection) – ASP.NET nije imao ugrađen Dependency Injection, a ASP.NET Core SignalR ima Depencency Injection koji je ugrađen u okvir (eng. framework) i tako se isporučuje. On se jednostavno može koristiti preko konstruktora, bez dodatne konfiguracije.

3.1. Transports

ASP.NET SignalR pojednostavljuje proces implementacije aplikacija u realnom vremenu. Iako je započeto sa ASP.NET veb aplikacijom, može je koristiti i druga .NET klijentska aplikacija. Jedna od ključnik komponenti SignalR-a jeste transportni medijum, pomoću kojeg komunicira.

Transportni mediji dostupni za SignalR su *WebSocket, Server-Sent Events (SSE), Forever Frame i Long Polling*.

SignalR je obezbedio omotač oko ovih transporta niskog nivoa koji omogućava programerima da se foukusiraju samo na poslovne zahteve umesto da brinu o tome da će se osnovni transport koristiti za slanje i primanje poruka između klijenta i servera. Na osnovu pretraživača i verzije pretraživača koju koristi veb aplikacija u realnom vremenu, SignalR u ASP.NET Core-u, koji je korišćen u ovom zadatku, automatski odlučuje o osnovnom transportu koji će se koristiti. Rezervni (eng. fallback) mehanizam se koristi za odlučivanje o transportnom sloju koji će koristiti aplikacija.

3.2. Hubs

SignalR koristi čvorove za komunikaciju između klijenta i servera. Čvor (eng. hub) je komponenta koja se nalazi u aplikaciji. On šalje i prima poruke od klijenta koristeći poziv udaljene procedure (eng. remote procedure call) koristeći osnovni transport. U suštini hub je centralna tačka u ASP.NET Core-u i on je odgovoran za rutiranje svih komunikacionih poruka u aplikaciji. Klasa koja nasleđuje klasu Microsoft.AspNetCore.SignalR.Hub i sadrži metode koje klijenti mogu da pozovu. Ove metode su poznate kao ,,hubs method'', koje se mogu pozvati iz koda na strani klijenta za slanje podataka serveru ili primanja podataka sa servera. Hubs takođe pružaju način na koji server šalje podatke određenim klijentima [7].

4. APLIKACIJA ZA DOPISIVANJE

4.1 Korišćene tehnologije i alati

Chat aplikacija predstavlja veb aplikaciju koja se sastoji od serverskog i klijentskog dela. Tehnologija koja je korišćena za serverski deo jeste ASP.NET Core, a od alata Microsoft Visual Studio. Na klijentskom delu aplikacije korišćeni su Flutter i Microsoft Visual Studio Code. Takođe, za bazu podataka korišćen je Microsoft SQL Server Management Studio.

4.1.1. ASP.NET Core

ASP.NET Core je okvir za veb razvoj otvorenog koda koji je dizajniran za izgradnju modernih aplikacija. Nudi brojne prednosti u odnosu na originalni ASP.NET okvir, uključujući poboljšane performanse, podršku za više platformi i pojednostavljen razvoj.

4.1.2. Microsoft Visual Studio

Visual Studio je okruženje (IDE – Integrated Development Environment) kompanije Microsoft za razvoj softvera: računarskih programa, veb stranica, veb aplikacija, veb servisa i mobilnih aplikacija. Kao alat Visual Studio je daleko više od jednostavnog editora namenjenog pisanju izvornog koda. Pored editora i dibagera uključuje kompajlere, alate za dovršavanje koda, grafičke dizajnere i mnoge druge funkcije.

4.1.3. Flutter

Flutter je UI (User Interface) softver otvorenog koda koji je kreirao Google. Omogućava da se iz jedne baze koda izgrade izvorno kompajlirane, multiplatformske aplikacije, za bilo koji veb pretraživač Android, Fuchsia, iOS, Linux, macOS i Windows. Programski jezik koji koristi je

Dart koji je optimizovan za brze aplikacije na bilo kojoj platformi.

4.1.4. Microsoft Visual Studio Code

Visual Studio Code je program za uređivanje teksta kompanije Microsoft. Predstavlja integrisano razvojno okruženje (eng. Integrated Development Environment - IDE) koji korisnicima nudi mogućnost uređivanja različitih vrsta teksta i koda.

4.1.5. Microsoft SQL Server Management Studio

Microsoft SQL Server Management Studio je softverska aplikacija koju je razvio Microsoft i koja se koristi za konfigurisanje, upravljanje i administraciju svih komponenti u okviru Microsoft SQL Server-a.

4.2. Arhitektura aplikacije

Arhitektura koja se koristila pri izradi ove aplikacije jeste troslojna arhitektura.

Troslojna arhitektura nastala je evolucijom dvoslojne arhitekture kako bi se prevazišli nedostaci koje je imala dvoslojna.

Aplikacije su podeljene u tri nivoa koji su međusobno labavo povezani i koji međusobno komuniciraju putem interfejsa koje poseduju prezentacioni sloj, sloj poslovne logike i sloj podataka.

4.3. Razmenjivanje poruka pomoću SignalR biblioteke

Ova aplikacija se sastoji od klijenta i servera, tako da je SignalR biblioteku potrebno implementirati na obe strane.

Ono što je prvo potrebno jeste instaliranje SignalR paketa. To se može postići pomoću Nuget menadžera (eng. Package Manager) ili konzole (eng. Package Manager Console). Sledeća stvar koju je potrebno uraditi jeste prijavljivanje servisa na serverskoj strani. Potrebno je kreirati SignalRHub klasu. SignalRHub nasleđuje klasu Hub koja upravlja vezama, grupama i porukama. Ona sadrži samo jednu funkciju SendMessage koja prima parametar SignalRMessage. Parametar predstavlja klasu kreiranu za poruke koje će se razmenjivati među korisnicima. Zatim mapiranje krajnje tačke (eng. endpoint-a) na SignalRHub klasu i prosleđivanje URL-a kako bi korisnici mogli da se konektuju na Hub.

Prelazimo na klijentsku stranu. Kada želimo da rukujemo svim stanjima koja se mogu dogoditi u aplikaciji, možemo koristiti Flutter Bloc. Stanje (eng. state) je stanje aplikacije u bilo kom trenutku. Na primer: kada korisnik dodirne dugme koje ga vodi na drugu stranicu, aplikacija je promenila svoje stanje. Unošenje teksta u prazno polje za tekst takođe dovodi do promene stanja.

Bloc je najbolji način za upravljanje stanjima. Predstavlja šablon dizajna (eng. Design Pattern) koji je kreirao Google da bi pomogao da se poslovna logika odvoji od ostalih slojeva aplikacije.

Cubit predstavlja minimalnu verziju ili podskup Bloc Design Pattern-a koji pojednostavljuje način na koji upravljamo stanjem aplikacije. On je sličan Bloc-u, ali ne zna ništa o događajima i oslanja se na metode za emitovanje novih stanja [8]. U ovom zadatku koristio se Cubit.

Tokom trajanja komunikacije između korisnika, Cubit služi da bi sve vreme slušao stanje aplikacije i tako prikazivao određene poruke. Veza je sve vreme otvorena. Promenom stanja pozivaju se određene metode jedna za drugom, da bi konačno došlo do uspešnog stanja, kad se iz njega mogu izvući određene informacije koje su nam potrebne za prikaz poruka.

5. ZAKLJUČAK

Zadatak ovog rada bio je da se prikaže komunikacija u realnom vremenu, korišćenjem SignalR biblioteke. Na početku prošli smo malo kroz istoriju veba, tehnika za komunikaciju, zatim objasnili neke pojmove koji su bili bitni prilikom izrade, videli kakva je arhitektura aplikacije, a onda i prikazali to na konkretnom primeru jedne aplikacije za ćaskanje.

Kao zaključak možemo reći da je WebSocket tehnika za komunikaciju najefikasnija i najprimenljivija tehnika komunikacije u realnom vremenu. Omogućava jedinstvenu konekciju između klijenta i servera koja može da dostavlja podatke u oba smera, što ostale opisane tehnike ne podržavaju. Lakoća implementacije i upravljanje događajima su takođe neke od prednosti ove tehnike. Veb se razvija iz dana u dan, a društvene mreže jesu jedan od primera njegovog rasta i razvoja koje su već neko vreme naša svakodnevnica. Stigli smo do kraja ovog rada, nadam se da sam postigla svoj cilj i približila razumevanje veb aplikacije u realnom vremenu na baš takvom primeru aplikacije za ćaskanje.

6. LITERATURA

[1] Jesse James Garrent – *Ajax: A New Approach to Web Applications*

[2] https://www.geeksforgeeks.org/what-is-polling-in-ajax/

[3] https://www.pubnub.com/blog/http-long-polling/

[4] https://levelup.gitconnected.com/real-time-updates-made-simple-a-introduction-to-server-sent-events-sse-db78ce3adf23

[5] https://learn.microsoft.com/en-us/aspnet/core/fundamentals/websockets?view=aspnetcore-7.0

[6] https://redwerk.com/blog/asp-net-core-signalr-introduction/

[7] https://learn.microsoft.com/en-us/aspnet/core/signalr/hubs?view=aspnetcore-5.0

[8] Dmitrii Slepnev – State Management Approached In Flutter

Kratka biografija:

Alisa Milanko rođena je u Kikindi 17.05.1995. Završava osnovnu školu „Đura Jakšić", a nakon toga srednju Ekonomsko-trgovinsku školu smer Ekonomski tehničar, takođe u Kikindi, završava 2014. godine. Iste godine upisuje Fakultet tehničkih nauka, smer Primenjeno softversko inženjerstvo. Nakon toga 2019. godine upisuje master studije na istom fakultetu, smer Računarstvo i automatika, podsmer Elektronsko poslovanje.