Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Развој на веб-апликација за комуникација во реално време со .Net и Angular

Дипломска работа

Ментор:
Проф. Д-р Петре Ламески

Кандидат:
Ангела Аврамовска 153166

Членови на комисија:
Проф. Д-р Владимир Трајковиќ

Проф. Д-р Димитар Трајанов

Скопје, Јануари 2025

Содржина

Апстракт

Оваа дипломска работа се фокусира на развој на веб-апликација за комуникација во реално време која користи .NET на серверската страна и Angular на клиентската страна. Целта на оваа дипломска работа беше да се создаде решение за комуникација помеѓу корисниците кои се наоѓаат во ист виртуелен простор (соба). За комуникацијата да биде брза и во реално време, имплементирана е SignalR библиотеката која овозможува поврзување на клиентот со серверот и испраќање на пораки, односно кога се испраќа порака, таа пристигнува инстантно до сите клиенти во реално време без потреба од освежување на страницата.

Апликацијата е дизајнирана да биде лесна за употреба и да овозможи размена на пораки во реално време помеѓу повеќе корисници. Главниот предизвик беше да се обезбеди стабилна комуникација помеѓу корисниците, со лесен кориснички интерфејс за користење на клиентската страна и ефикасен серверски код на .NET со SignalR.

Клучни точки:

- Развој на функционална веб-апликација за комуникација во реално време.
- Креирање и поставување на проектите.
- Користење на .NET рамка за креирање на серверскиот дел од апликацијата.
- Употреба на SignalR за двонасочна комуникација помеѓу клиентот и серверот.
- Применување на Angular како фронтенд рамка за креирање на кориснички интерфејс.
- Управување со пораки во реално време, вклучувајќи и испраќање и примање.
- Ефикасен и оптимизиран модел за размена на податоци помеѓу корисникот и серверот.

1. Вовед

Во современото општество, комуникацијата претставува основа за секојдневниот живот, активностите во образованието и професионалната соработка. Со напредокот на технологијата, расте и потребата за интуитивни и брзи методи на комуникација кои ќе бидат достапни на сите корисници во реално време.
Социјалните мрежи и разните комуникациски апликации го менуваат начинот на кој луѓето разменуваат информации, овозможувајќи беспрекорна интеракција помеѓу корисниците и системите преку ефикасни и оптимизирани модели за размена на податоци, кои служат како основа за современи комуникациски решенија.
Со оваа апликација се прикажува баш тоа, комуникација помеѓу корисници кои се поврзани во ист виртуелен простор, односно споделуваат заедничка „соба“.
Оваа апликација овозможува брза комуникација помеѓу корисниците кои можат да се наоѓаат во исто виртуелно опкружување (соба) без да се соочуваат со задоцнување на пораките.

Со употреба на .NET, Angular и библиотеката SignalR , ова решение нуди механизам за двонасочна комуникација помеѓу клиентската и серверската страна, овозможувајќи корисничко искуство кое е едноставно, интуитивно и брзо. Ова е корисно како за социјални активности, така и за работни задачи каде што моменталната размена на информации е неопходна.

Целта на оваа веб-апликација е да овозможи комуникација помеѓу повеќе корисници кои се во ист виртуелен простор (соба) во реално време. Исто така апликацијата обезбедува интерфејс кој е лесен за употреба и овозможува едноставно внесување на корисничко име и влегување во соба за разговор. Дополнително, апликацијата го решава предизвикот поврзан со задоцнување на пораки при традиционалните методи на комуникација.

Со завршување на оваа дипломска работа, се добива техничко решение кое се фокусира на едноставни потреби и овозможува брза комуникација помеѓу корисниците.

2. Преглед на користени технологии

Во овој дел се опишани технологиите кои се користени во развојот на апликацијата и обезбедуваат комуникација во реално време. Секоја од нив е одбрана за да ја поддржи функционалноста на апликацијата и да ги исполни нејзините барања.

2.1 .NET

Што е .NET?

.Net е рамка за софтвер која е развиена од Microsoft. Со помош на оваа рамка може да се развиваат многу типови на апликации како што се веб, десктоп и мобилни апликации.

Со платформата која комбинира функционалност, библиотеки и алатки, процесот на развој е доста поедноставен.

Една од карактеристиките е тоа што овозможува на програмерите да развиваат апликации кои се компатибилни на Windows, Linux и macOS. Тие можат да пишуваат код на повеќе програмски јазици како што се C#, VB.NET, F# и многу други.
ASP.NET е популарна рамка во .NET која се користи за развивање веб-апликации и сервиси. Таа поддржува веб API, динамични веб-страници и комуникација во реално време која е подобрена со SignalR. Понатаму, .NET поддржува Common Language Runtime (CLR) кој обезбедува безбедно извршување на кодот преку управување со извршувањето на .NET програмите. Компилацијата го претвора кодот во машински инструкции кои потоа се извршуваат на процесорот на компјутерот. [1] [11]

2.2 SignalR

Што е SignalR?

SignalR е библиотека на Microsoft која овозможува двонасочна комуникација помеѓу клиентската и серверската страна во веб-апликациите. [2]

За разлика од стандардниот HTTP протокол, кој овозможува само еднонасочна комуникација (од клиент до сервер), SignalR користи WebSocket протокол за да создаде постојано отворени канали за комуникација помеѓу клиентите и серверот. [3]

SignalR автоматски го избира најоптималниот транспортен протокол во зависност од карактеристиките на мрежата, како што е **WebSockets** кој нуди најбрза и најефикасна комуникација. Кога WebSocket не е достапен, **Server-Sent Events (SSE)** овозможува двонасочна комуникација, додека **Long Polling** се користи во ситуации кога ниту WebSockets ниту SSE не можат да се користат. [2] [9]

Како работи SignalR?

SignalR користи објект наречен **Hub** за поврзување помеѓу клиентите и серверот. Клиентот се поврзува со серверот преку овој Hub, а серверот може да испраќа пораки до одреден клиент или до сите клиенти поврзани на истиот канал.

Комуникацијата се одвива на следниот начин:
Клиентот се поврзува со серверот преку SignalR Hub. Откако успешно ќе се поврзе, серверот слуша настани и испраќа пораки на клиентите кога е потребно. Соодветно клиентот прима пораки и ги обработува според својата логика. [2] [14]

Најголемата предност на SignalR е тоа што поддржува **двонасочна комуникација** со што клиентите можат да комуницираат со серверот и обратно во реално време.
Исто така има поддршка за различни транспортни протоколи и автоматски го избира оптималниот протокол. Бидејќи е развиен од Microsoft, SignalR најдобро се интегрира со .NET апликации. [2]

Типови на комуникации во веб-апликации

1. Асинхрона комуникација
   Асинхроната комуникација се користи кога податоците се испраќаат од клиентот до серверот или обратно. Клиентот може да продолжи со извршување на други задачи додека чека одговор од серверот. Ова е корисно во сценарија каде интеракцијата во реално време не е неопходна и позадинските процеси се поважни.
   Асинхроната комуникација ни овозможува и да ја поделиме нашата логика на задачи што се чекаат меѓусебно со цел да не се блокира извршувањето на нашата апликација.

2. Синхрона комуникација
   Синхроната комуникација се одвива на начин на кој кога ќе се испрати порака или барање, клиентот чека серверот да одговори пред да продолжи. Овој метод може да предизвика бавна комуникација ако на серверот му треба време да го обработи барањето.
   Се употребува во едноставни операции CRUD операции, веб форми и основни барања на податоци.
   Тука се користат стандардните барања за HTTP (GET, POST, PUT, DELETE). [4]

3. Комуникација во реално време
   Комуникацијата во реално време овозможува инстантна размена на податоци помеѓу клиентите и серверите. Податоците се испраќаат и примаат постојано и може динамично да се ажурираат во реално време.
   Се употребува во апликации за разговор, известувања во живо, игри и друго. [5]

2.3 Angular

Што е Angular?

Angular е популарна фронтенд рамка за развој на динамични, едностарни веб апликации (SPA). Оваа рамка користи TypeScript и нуди обемен сет на алатки за развивање на веб апликации со висока функционалност.

Angular обезбедува поддршка за компоненти, директиви, сервиси, RxJS и многу други концепти кои го прават развојот на веб апликации поефикасен.
Еден од најважните аспекти на Angular е неговата архитектура која се базира на компоненти. Секој дел од интерфејсот е претставен како компонента која има свој шаблон (HTML), стилови (CSS) и логика (TypeScript).

Овие компоненти можат да се комбинираат, создавајќи комплексни интерфејси.

Главните карактеристики на Angular се двослојната архитектура која со помош на модули, компоненти и сервиси, овозможува разделување на апликацијата во независни блокови. Потоа RxJS за реактивно програмирање, библиотека која овозможува обработка на асинхрони настани, како што се поврзувања со сервери и обработка на податоци.
Angular се базира на TypeScript, кој е подобрена верзија на JavaScript, што го прави кодот поразбирлив, и дополнително користи сервиси за да ја изврши логиката на апликацијата и да овозможи повторна употреба на кодот помеѓу компонентите.
Angular апликациите се креираат врз база на компоненти, во кои секоја има засебна логика, шаблон и стилови. Тие се одговорни за интерфејсот и интеракцијата со корисникот. Исто така поддржува two-way data binding со што се овозможува синхронизација помеѓу самиот кориснички интерфејс и моделот. Ова подразбира дека промените што ќе направат на интерфејсот, во истиот момент се рефлектираат и во моделот на податоци и обратно. [6] [13] [12]

Пример за употреба на SignalR (серверска страна) и Angular сервис за поврзување со SignalR:

```
using Microsoft.AspNetCore.SignalR;
using SignalChat.Server.Models;

namespace SignalChat.Server.HubConfig
{
    public class ChatHub : Hub
    {

        public override Task OnConnectedAsync()
        {
            Clients.All.SendAsync("OnConnected", "OnConnected is working");
            return base.OnConnectedAsync();
        }

        public async Task SendMessage(Message message)
        {
            if (_connection.TryGetValue(Context.ConnectionId, out
UserRoomConnection? userRoomConnection))
            {
                message.User = userRoomConnection.User;
                message.MessageTime = DateTime.Now.ToString();

                await Clients.Group(userRoomConnection.Room!)
                    .SendAsync("ReceiveMessage", message);
            }
        }
    }
}
```

Во ChatHub класата се дефинирани методите што се повикуваат од клиентската страна, како на пример SendMessage и воспоставувањето на SignalR конекција со OnConnectedAsync.

```
var builder = WebApplication.CreateBuilder(args);

builder.Services.AddSignalR();

var app = builder.Build();

app.UseRouting();
app.MapHub<ChatHub>("/chat");

app.Run();
```

Во Program.cs фајлот се прави конфигурацијата на SignalR.фајлот се прави конфигурацијата на SignalR.

```
import { Injectable } from '@angular/core';
import { HubConnection } from '@microsoft/signalr';
import * as signalR from '@microsoft/signalr';
import { BehaviorSubject } from 'rxjs';
import { iMessage } from '../interfaces/message';

@Injectable({
  providedIn: 'root'
})
export class ChatService {

  public messages$ = new BehaviorSubject<any>([]);
  public messages: any[] = [];

  public connection: HubConnection = new signalR.HubConnectionBuilder()
    .withUrl('https://localhost:7166/chat')
    .configureLogging(signalR.LogLevel.Information)
    .build();

  constructor() {

    this.connection.on('ReceiveMessage', (message: iMessage) => {
      this.messages = [...this.messages, message]
      this.messages$.next(this.messages)
    });

    this.connection.on('OnConnected', (message: string) => {
      console.log(message);
    })
  }

  public async start() {
    try {
      await this.connection.start();
    } catch (error) {
      console.log(error)
    }
  }

  public async sendMessage(message: iMessage) {
    return this.connection.invoke('SendMessage', message)
  }
}
```

HubConnection е генериран со помош на SignalR HubConnectionBuilder кој го поврзува хабот на https://localhost:7166/chat.

Во овој пример се прикажани два настани кои сервисот ги слуша од серверот:

ReceiveMessage – кој додава во листа пораки што пристигнуваат.

OnConnected – запишува порака за успешна конекција.

start() функцијата ја започнува конекцијата.
sendMessage() функцијата ја повикува sendMessage методата на серверот со пораката како параметар.

2.4 WebSocket протокол

SignalR користи WebSocket како преферирaн транспортен протокол за комуникација помеѓу клиентот и серверот.
SignalR користи WebSockets бидејќи тоа е најбрзиот и најповолниот начин за комуникација во реално време. Со употреба на WebSockets, комуникацијата помеѓу клиентот и серверот е постојана, така што нема потреба за повторно отворање на нови конекции. [7]

3. Архитектура на решението

Во оваа точка ќе биде објаснета архитектурата на целокупното решение. Ќе бидат опишани архитектонските компоненти на системот, начинот на комуникација помеѓу клиентската и серверската страна, како и применетите дизајн-пристапи и употребените шаблони за дизајн.

3.1 Преглед на архитектурата

Архитектурата на оваа апликација се состои од две главни компоненти:

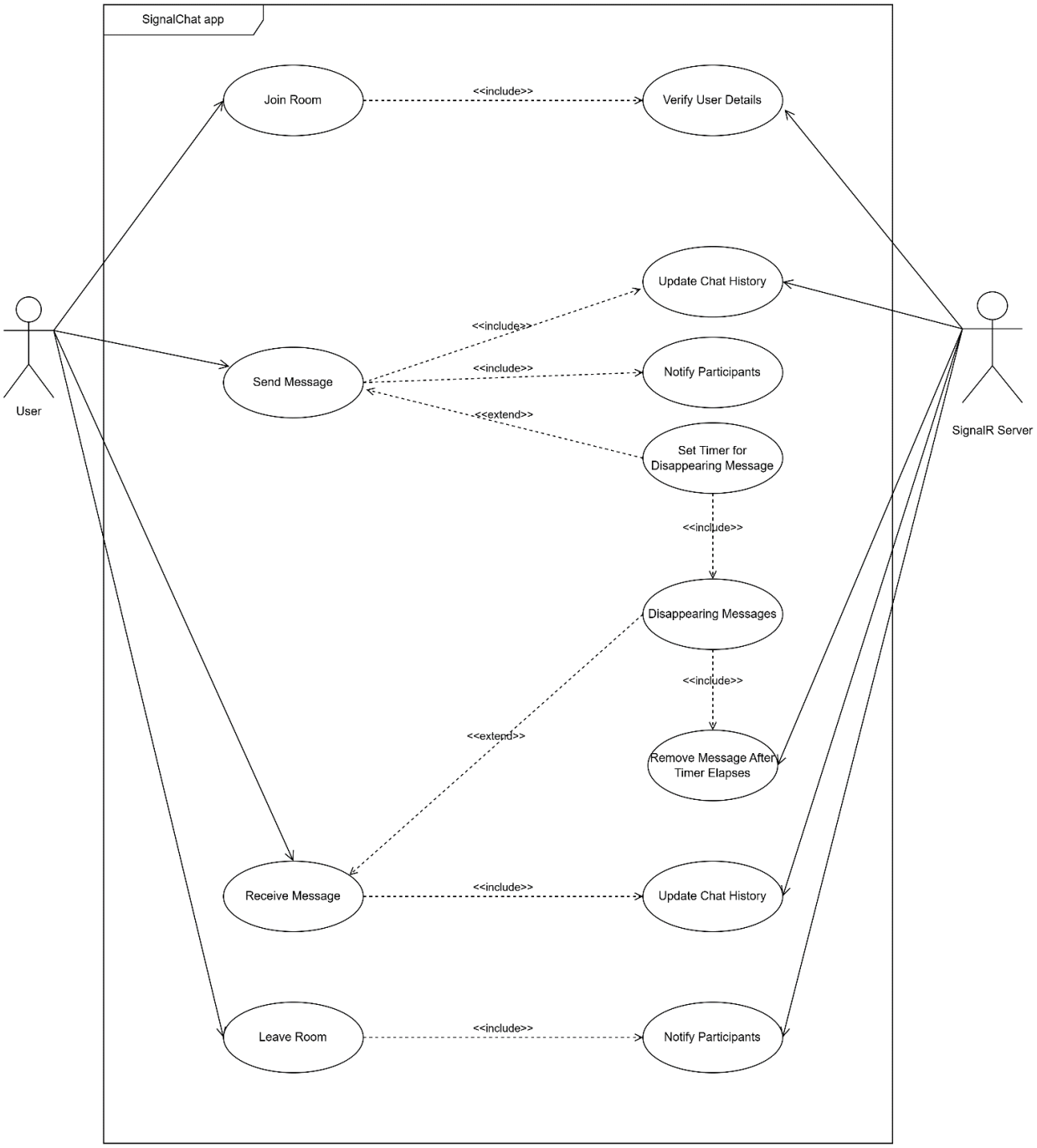
Клиентска страна (Frontend - Angular):

Клиентската страна ја претставува интеракцијата на корисникот со апликацијата. Оваа страна е одговорна за прикажување на корисничкиот интерфејс, испраќање пораки до серверот и примање пораки преку SignalR.

Серверска страна (Backend - .NET SignalR):

Серверската страна ја овозможува комуникацијата помеѓу клиентите со помош на SignalR. Оваа страна ги обработува настаните, управува со комуникацијата помеѓу сите клиенти и испраќа и прима пораки од клиентите.

3.2 UML дијаграм



3.3 Користени дизајн-шаблони

Во текот на имплементацијата на апликацијата, беа применети неколку важни дизајн-шаблони со цел да се обезбеди повторна употреба на кодот, лесно одржување и едноставна комуникација помеѓу компонентите:

Observer Шаблон
Observer (набљудувач) шаблонот се користи за да овозможи еден објект (набљудувач) да биде известен за промените во состојбата на друг објект (набљудуван).
SignalR и RxJS го користат Observer шаблонот за обработка на настани во реално време.

SignalR хабовите ги известуваат сите поврзани клиенти за нови пораки.
RxJS во Angular следи промени на податоците и автоматски ги ажурира компонентите.
Пример:
Кога корисникот испраќа порака, таа се испраќа на серверот. Серверот ги информира сите поврзани клиенти (набљудувачи) дека има нова порака.

Придобивки:
Овозможува реактивно програмирање.
Го олеснува управувањето со асинхрони настани.

Singleton Шаблон
Singleton шаблонот обезбедува дека одредена класа ќе има само една инстанца, која ќе биде достапна глобално.
SignalR хабот на серверот е имплементиран како Singleton. Ова гарантира дека сите клиенти комуницираат со истата инстанца на хабот, што е клучно за синхронизација на пораки.

Придобивки:
Централизирана точка за управување со комуникацијата.
Намалување на ресурсите со споделување на една инстанца.

Dependency Injection (DI)
DI е шаблон кој овозможува зависностите (сервиси, податоци) да се инјектираат во класи, наместо тие сами да ги креираат.
Angular автоматски имплементира DI за сервисите. На пример, ChatService се инјектира во компонентите кои комуницираат со серверот.

Придобивки:
Полесно тестирање и заменување на сервиси.
Намалување на зависноста помеѓу компонентите.

Користењето на овие дизајн-шаблони нуди јасна структура, флексибилност и лесна одржливост на апликацијата. Секој шаблон има своја специфична улога, од организирање на кодот до подобрување на перформансите во реално време. [8]

3.4 Комуникација помеѓу клиент и сервер

Комуникацијата помеѓу клиентската и серверската страна се базира на SignalR. SignalR користи WebSocket протокол за да овозможи двонасочна и високо ефикасна комуникација помеѓу сите поврзани клиенти и серверот.

Клиентот испраќа пораки со помош на сервис, а серверот ги прима тие пораки преку Hub и потоа ги пренесува до сите клиенти кои се поврзани на истиот канал.

3.5 Реактивно програмирање со RxJS

RxJS е библиотека во Angular со која асинхроно се управува со настаните и обработката на податоци. Дозволува програмерите да работат со податоци кои подлежат на промени како на пример API одговори и ажурирања во реално време. [6] [15]

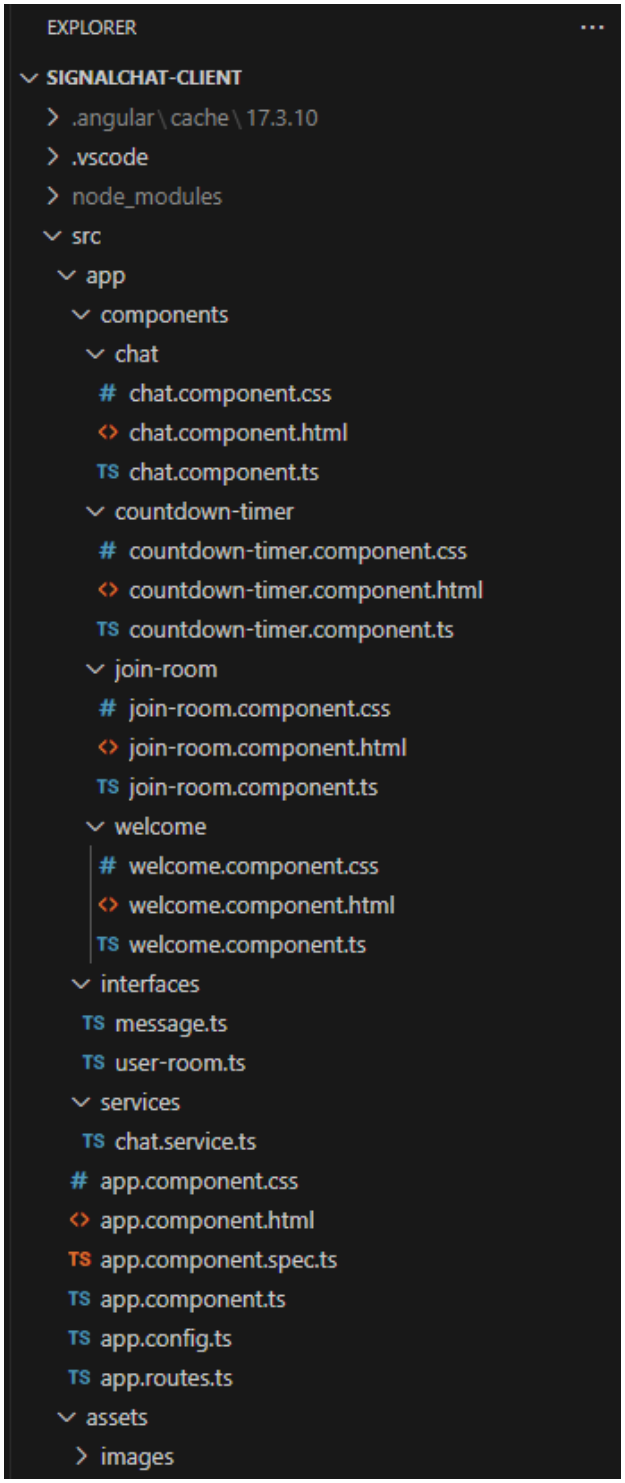
4. Имплементација

Ова поглавје го опфаќа практичниот дел од развојот на апликацијата за комуникација во реално време. Детално ќе биде опишан процесот на имплементација на фронтенд и бекенд компонентите, комуникацијата помеѓу клиентот и серверот, како и начинот на кој се изработуваат главните функционалности.

4.1 Фронтенд имплементација

Во оваа апликација, го користам пакетот @microsoft/signalr за интегрирање на функционалноста на SignalR. Овој пакет ги обезбедува сите алатки што ми се потребни за воспоставување врска помеѓу клиентот (Angular) и серверот (.NET) за комуникација во реално време.

4.1.1 Организација на проектот



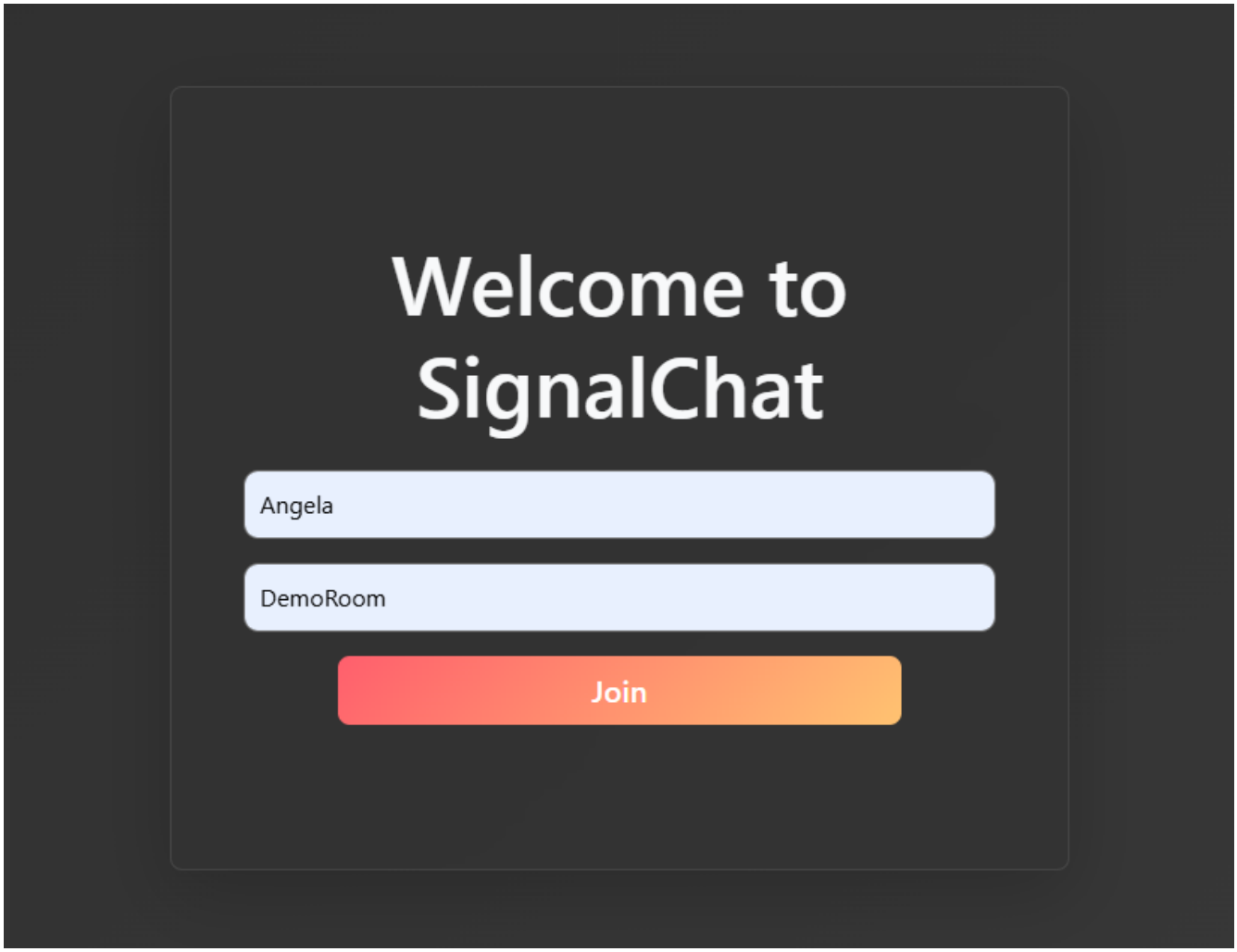
Слика бр. 1 - Структура на Angular апликацијата

Проектот е организиран во неколку делови:

Првиот дел е сместен во **app/components.** Тука се сместени компонентите за корисничкиот интерфејс, како што се welcome-page, join-room и chat. Потоа, во **app/services** е сместен сервисот ChatService кој е одговорен за комуникацијата со серверот преку SignalR. Интерфејсите како Message и UserRoom кои ја структурираат апликацијата се сместени во **app/interfaces.** [13]

4.1.2 Главни компоненти

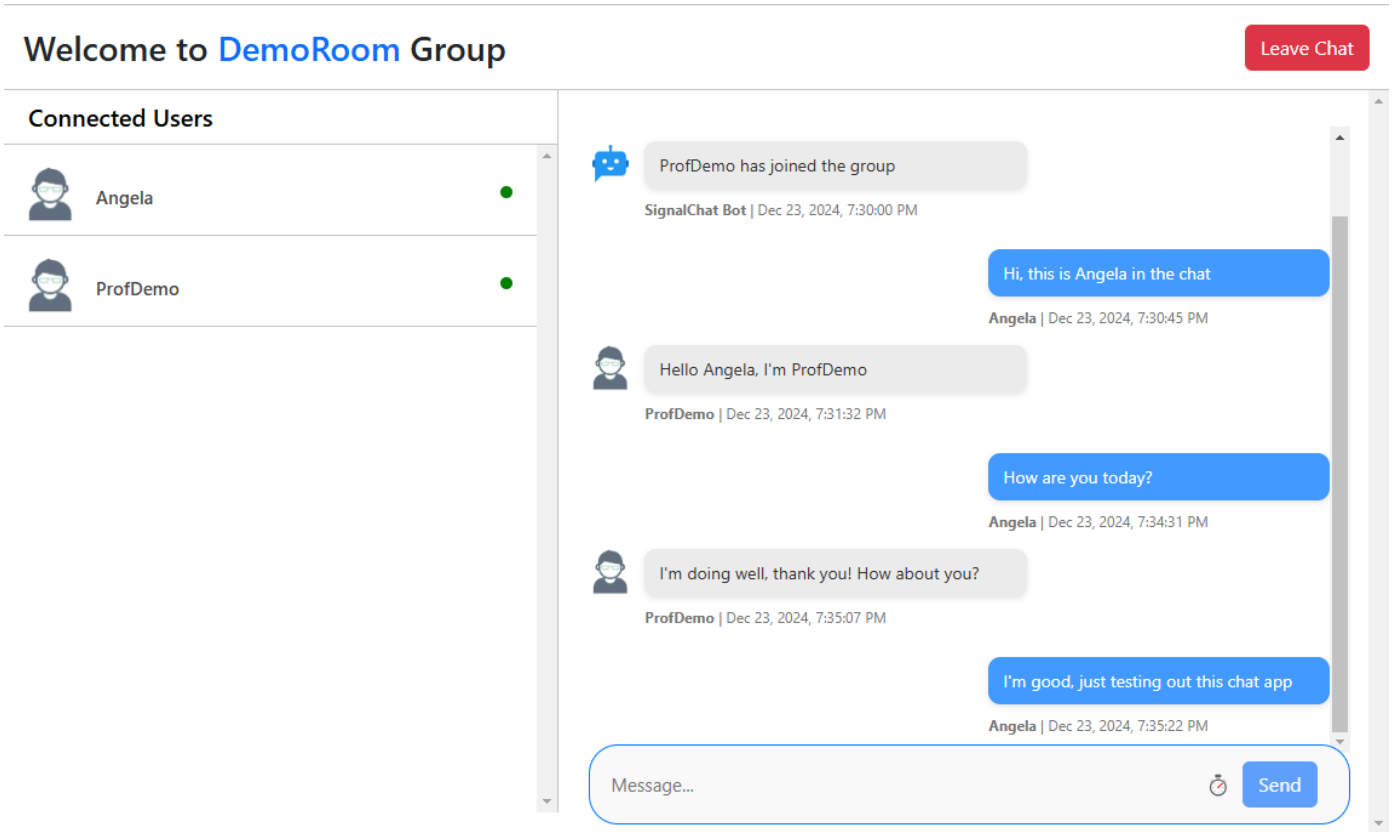
- **Join Room Page:**
  - Откако корисниците ќе кликнат на копчето "Join Room Now", кое ги пренесува на компонентата Join Room, тие тука внесуваат корисничко име и име на соба. Кодот за оваа компонента обезбедува валидација на внесените информации и го носи корисникот до chat-room.



Слика бр.2 - Екран за внесување на корисничко име и име на соба

Клучни функции се валидација на на формата (празни полиња) и чување на внесените податоци во локални променливи.

- **Chat Room:**
  - Главната компонента во која се прикажуваат испратените пораки и листа на активни корисници.



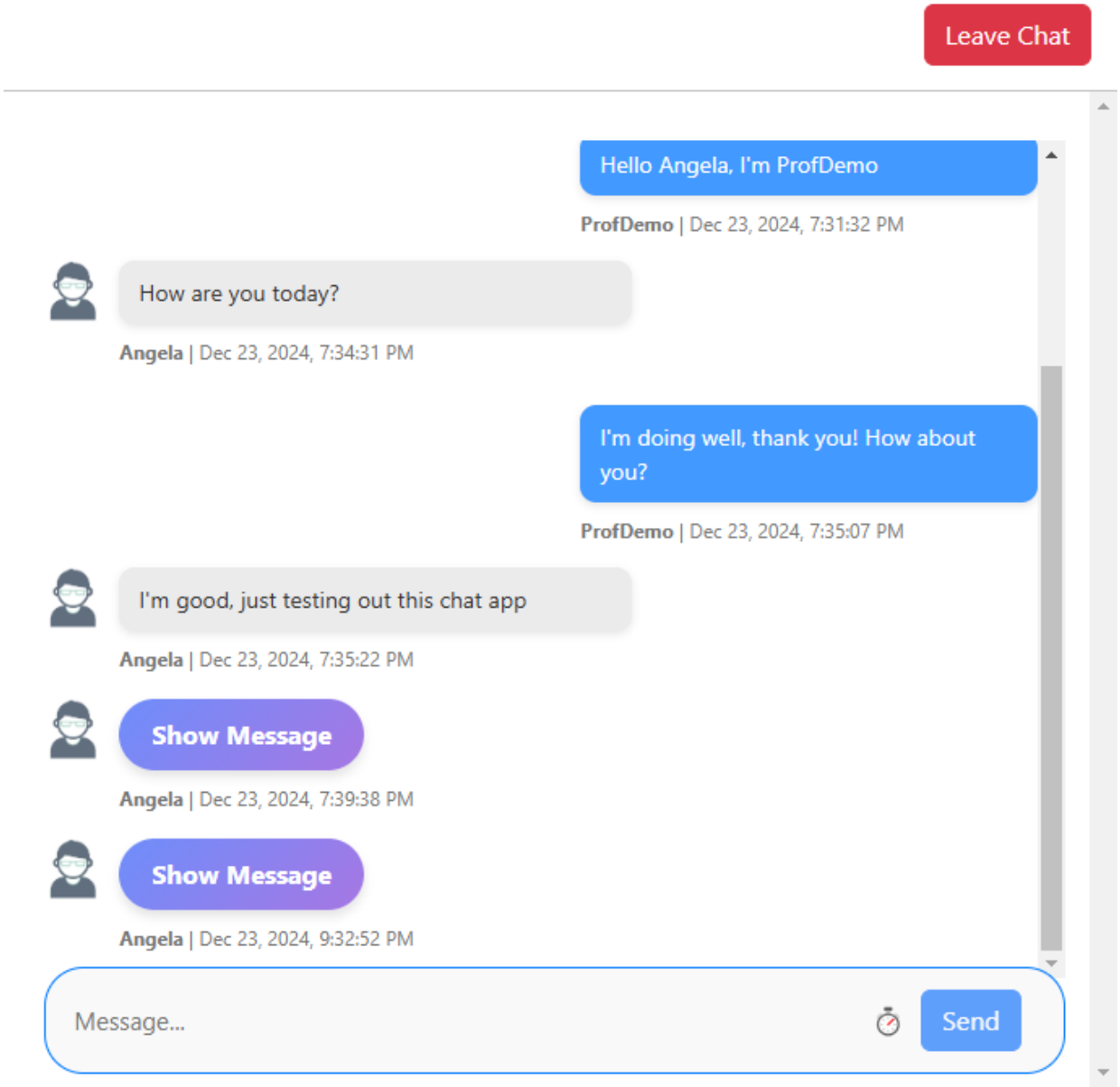
Слика бр.3 - Екран од Chat Room каде се прикажуваат испратените пораки и листата на активни корисници

Едни од клучните функции се динамичното прикажување на пораките користејќи *ngFor директива како и обработката на настани при испраќање пораки.

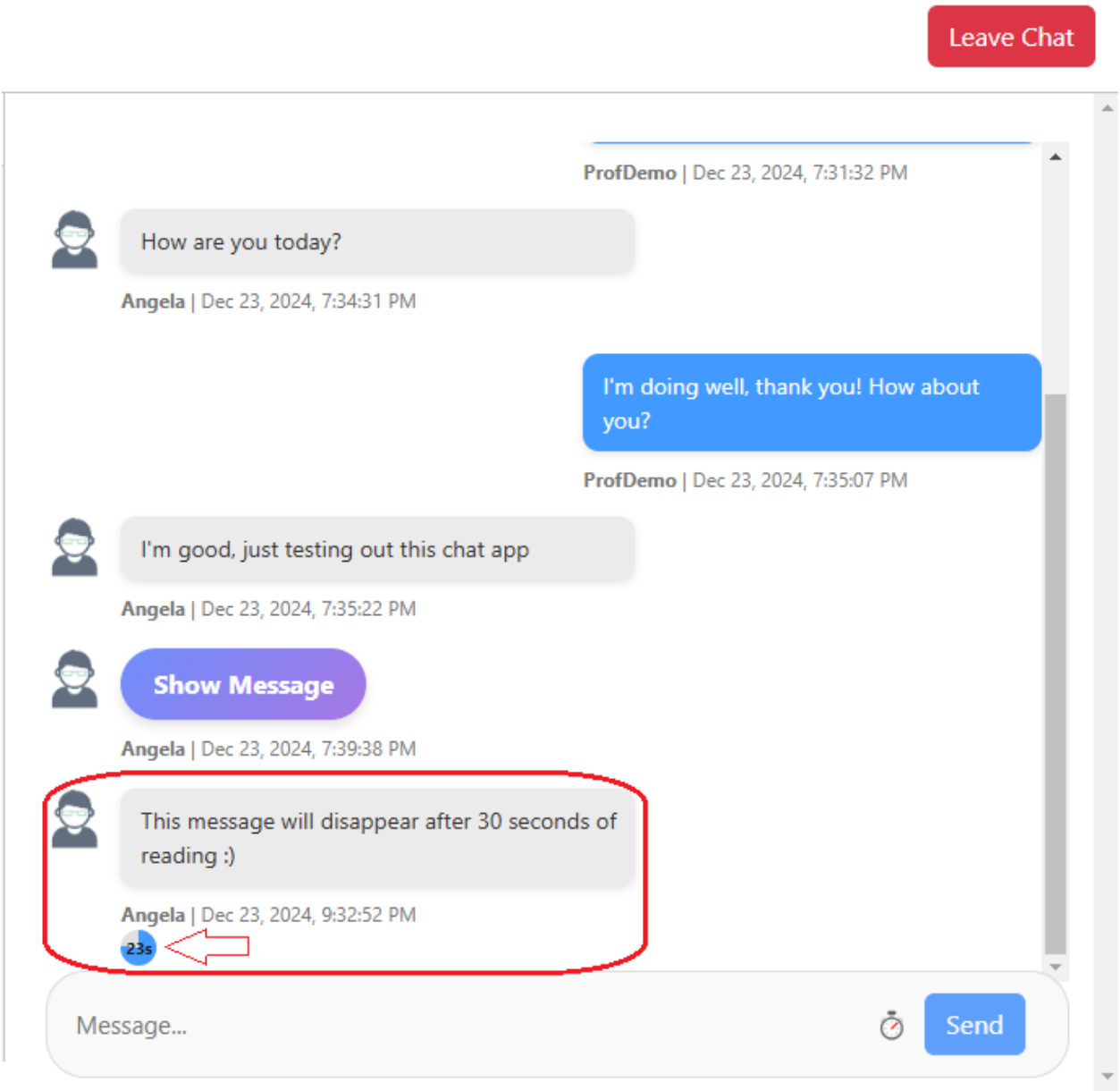
Пораки што исчезнуваат по читањето

Пораките што исчезнуваат по читањето се дополнителна функционалност кои им овозможуваат на корисниците да испраќаат пораки што автоматски се бришат по одредено време (на пример 10, 30 или 60 секунди) откако ќе се прочитаат.
Оваа функција ја подобрува приватноста и гарантира дека чувствителните информации не се зачувуваат.

Функционира на следниов начин:
Корисниците имаат опција да избираат временски интервал кога испраќаат порака. Потоа, кај корисниците што ја примаат пораката се прикажува копче, на кое треба да се притисне за да се види пораката. По притискање на копчето, кај секој корисник почнува одбројување во времетраење на избраниот временски интервал, а на корисничкиот интерфејс се прикажува тајмер покрај пораката. На крајот, кога ќе истече тајмерот, пораката се брише во реално време само кај корисникот што ја отворил.



Слика бр.4 - Екран од Chat Room каде што се пристигнати пораки кои исченуваат по читањето



Слика бр.5 - Екран од Chat Room по притискање на копчето, прикажувајќи тајмер кој одбројува до бришењето на пораката

4.1.3 Комуникација со SignalR

Комуникацијата со серверот е овозможена преку сервисот ChatService.

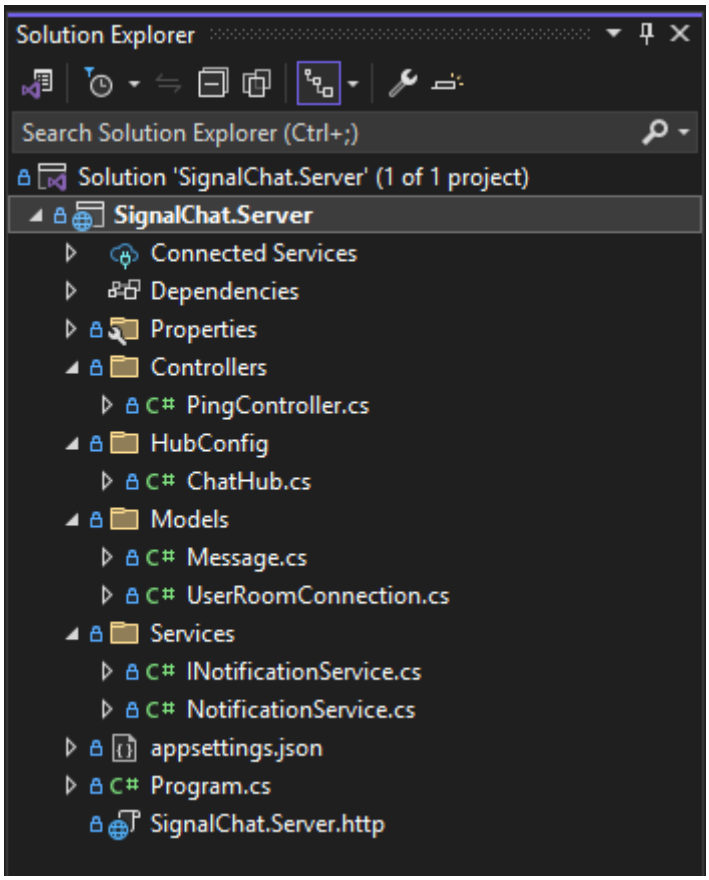
Главните чекори се воспоставување на SignalR конекција со серверот, регистрација на методите за слушање пораки (ReceiveMessage) и испраќање пораки од клиентот до серверот (sendMessage).

Пример:

```
public async sendMessage(message: iMessage) {
  return this.connection.invoke('SendMessage', message)
}
```

Слика бр.6 - Метод за испраќање пораки до серверот преку SignalR

4.2 Бекенд имплементација



Слика бр. 7 - Структура на .Net апликацијата

Бекендот е имплементиран со ASP.NET Core и SignalR. Овој дел од апликацијата овозможува обработка на податоци, менаџирање на собите и испраќање пораки во реално време.

4.2.1 SignalR Hub

ChatHub е главниот дел на бекендот. Сите корисници комуницираат преку овој хаб.

Главните функции се сместени во два дела од кои, **JoinRoom -** метод кој го поврзува корисникот со одредена соба и **SendMessage -** обработува и испраќа пораки до сите корисници во собата.

ConnectionId

Со употребата на SignalR, секоја клиентска конекција добива единствен ConnectionId од серверот, кој служи за идентификација на поврзаниот клиент.
Во хабот, Context.ConnectionId го содржи ConnectionId на моменталниот клиент.
Со ова се овозможува серверот да испраќа пораки на одредени клиенти. Како што е случај во оваа апликација, корисниците што влегуваат во една група на разговор разменуваат пораки меѓусебно без мешање и губење на пораките.

Пример код:

```
public async Task SendMessage(Message message)
{
    if (_connection.TryGetValue(Context.ConnectionId, out UserRoomConnection? userRoomConnection))
    {
        message.User = userRoomConnection.User;
        message.MessageTime = DateTime.Now.ToString();

        await Clients.Group(userRoomConnection.Room!)
            .SendAsync("ReceiveMessage", message);
    }
}
```

Слика бр. 8 - SendMessage метод кој ја обработува и испраќа пораката до сите корисници во соодветната соба

4.2.2 Управување со собите

SignalR автоматски менаџира групи (соби). Методи како AddToGroupAsync и RemoveFromGroupAsync се користат за поврзување и исклучување на корисниците.

4.2.3 Автентикација на корисници

Автентикацијата во оваа апликација е едноставна со што од корисниците се бара само да внесат корисничко име и соба за приклучување. Овој пристап е одбран бидејќи примарниот

фокус е на комуникациските функции во реално време, а исто така и намалување на сложеноста со цел фокусот да биде на развојот и демонстрација на основните функционалности.

4.3 Интеграција помеѓу фронтенд и бекенд

4.3.1 WebSocket

SignalR автоматски користи WebSocket за оптимизирана комуникација. [10]

4.3.2 Проток на податоци

Најпрво, корисникот испраќа порака преку Angular. Потоа пораката се пренесува на SignalR хабот. Следно, хабот ја испраќа пораката до сите корисници во собата и финално Angular интерфејсот ја прикажува новата порака.

4.4 Тестирање на функционалности

4.4.1 Тестирање на фронтендот

Се тестираат прикажувањето на пораките и навигацијата помеѓу компонентите.

4.4.2 Тестирање на бекендот

Се проверува функционирањето на SignalR хабот како и испраќањето и примањето на пораки.

4.4.3 Интеграциско тестирање

Се тестираат реални сценарија како што е пристап во исто време од страна на повеќе корисници.

Овој процес на имплементација дава сигурност дека апликацијата е функционална и оптимизирана за комуникација во реално време.

5. Заклучок

Изработката на апликацијата за комуникација во реално време претставуваше уникатна можност за примена на современи технологии и архитектурни концепти. Оваа дипломска работа покажува како Angular и .NET SignalR можат да се користат заедно за да се создаде едноставна, но ефикасна апликација која ги исполнува основните потреби за комуникација во реално време.

5.1 Постигнати цели

Преку оваа дипломска работа се реализирани неколку важни аспекти како што е функционалност на апликацијата, односно развиена е веб-апликација која им овозможува на корисниците да се поврзат во заеднички соби за разговор и да разменуваат пораки во реално време.
Добиена е теоретска основа преку детално разгледување и објаснување на технологиите што се користени (Angular, .NET SignalR, WebSockets).
Исто така, апликацијата е изградена според принципите на модуларен дизајн, со лесно одржување и можност за проширување.

5.2 Ограничувања и предизвици

Иако апликацијата функционира правилно, постојат одредени ограничувања и предизвици што беа согледани за време на изработката.
Најпрво, апликацијата е наменета за основна употреба и не ги содржи напредните функции како што се криптирање на пораките, интеграција со надворешни платформи, или сложено управување со корисници.
Дополнително, не се користи база на податоци за зачувување на корисниците и пораките.
Потоа, иако WebSockets овозможуваат брза и ефикасна комуникација, можат да се појават проблеми при голем број на истовремени корисници, што би барало дополнителна оптимизација.
Механизмите за обработка на грешки и известување би можеле да бидат подобрени за покомплексни сценарија.

5.3 Идни можности за надградба

Во иднина, апликацијата би можела да се прошири со дополнителни функционалности, како што се кориснички интерфејси односно додавање функционалност за регистрирање и автентикација на корисници, а за подобрување на приватноста и безбедноста би можело да се имплементира енкрипција на пораките.
Потоа, додавање можност за споделување слики, видеа или датотеки, како и административен панел со кој ќе може да се управува со соби, корисници и преглед на активност.

Работата на оваа дипломска работа ми овозможи практично да ги применам теоретските знаења стекнати за време на студиите. Исто така, го зајакнав разбирањето за принципите на комуникација во реално време и работата со современи технологии како што се Angular и .NET.

Апликацијата е функционална основа за понатамошен развој и претставува добар пример за тоа како може да се изгради решение со користење на модерни технологии.

1. Референци

[1] **Вовед во .NET Core**
https://learn.microsoft.com/en-gb/dotnet/core/introduction?WT.mc_id=dotnet-35129-website
Пристапено на: 09.12.2024

[2] **ASP.NET Core SignalR**
https://learn.microsoft.com/en-us/aspnet/core/signalr/
Пристапено на: 09.12.2024

[3] **ASP.NET Core WebSockets**
https://learn.microsoft.com/en-us/aspnet/core/fundamentals/websockets?view=aspnetcore-8.0
Пристапено на: 10.12.2024

[4] **Синхрона и Асинхрона комуникација**
https://www.techtarget.com/whatis/definition/synchronous-asynchronous-API
Пристапено на: 10.12.2024

[5] **Комуникација во реално време**
https://www.tutorialspoint.com/real-time-communications-rtc
Пристапено на: 10.12.2024

[6] **Angular Документација**
https://v17.angular.io/docs
Пристапено на: 11.12.2024

[7] **WebSockets API**
https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API
Пристапено на: 11.12.2024

[8] **ASP.NET Core Roadmap**
https://roadmap.sh/aspnet-core
Пристапено на: 15.12.2024

[9] **Протоколи за комуникација**
https://getstream.io/blog/communication-protocols/
Пристапено на: 10.12.2024

[10] **Апликаци во реално време во .NET со SignalR**
https://medium.com/@kova98/real-time-apps-in-net-with-signalr-f4e0381771ab
Пристапено на: 15.12.2024

[11] **Што е ASP.NET**
https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet
Пристапено на: 10.12.2024

[12] **Real-Time Application Development со SignalR и Angular**

https://medium.com/@gabrielbastosdeveloper/real-time-application-development-with-signalr-and-angular-74c7d869afaf
Пристапено на: 09.12.2024
[13] **Angular преглед**
https://angular.dev/overview
Пристапено на: 14.12.2024

[14] **ASP.NET Core SignalR упатство**
https://learn.microsoft.com/en-us/aspnet/core/tutorials/signalr?view=aspnetcore-8.0&tabs=visual-studio
Пристапено на: 10.12.2024

[15] **Angular Roadmap**
https://roadmap.sh/angular
Пристапено на: 14.12.2024