Составитель: Садовский Р.В.

Лабораторная работа №28
Обмен данными

1 Цель работы

1.1 Изучить процесс разработки сетевых приложений с использованием SignalR на C#;

2 Литература

2.1 Основы SignalR // Metanit.com – URL: https://metanit.com/sharp/signalr/1.12.php. – Режим доступа: свободный. – Текст : электронный.

3 Подготовка к работе

3.1 Повторить теоретический материал (см. п.2).

3.2 Изучить описание лабораторной работы.

4 Основное оборудование

4.1 Персональный компьютер.

5 Задание

5.1 Создание сервера для чата на SignalR

5.1.1 Создать новый проект ASP.Net minimal api.

5.1.2 Подключить в файле Program пакет Microsoft.AspNetCore.SignalR

5.1.3 Зарегистрировать в сервисах SignalR

```
builder.Services.AddSignalR();
```

5.1.4 Создать класс ChatHub – наследник класса Hub, в котором реализовать асинхронный метод для ретрансляции сообщений, полученных от пользователя другим пользователям с помощью вызова удаленного callback-метода.

```
await Clients.All.SendAsync(имя_callback, отправляемые_данные);
```

5.1.5 Создать endpoint для ChatHub при помощи

```
app.MapHub<ChatHub>("/адрес");
```

5.2 Создание клиента для чата на SignalR

5.2.1 Создать новый проект консольного приложения

5.2.2 Установить в проекте пакет Microsoft.AspNetCore.SignalR.Client

5.2.3 Установить подключение к серверу SignalR при помощи

```
var connection = new HubConnectionBuilder()
    .WithUrl("адрес hub")
    .Build();
```

5.2.4 Настроить callback для получения сообщения от сервера исходя из данных, отправляемых в п.5.1.4

```
connection.On<типы_получаемых_данных>(имя_callback,
(получаемые_данные) =>
{
    //Код для вывода полученного сообщения
```

```
});
```

5.2.5 Запустить соединение к серверу

```
await connection.StartAsync();
```

5.2.6 Далее необходимо бесконечно запрашивать у пользователя сообщения и отправлять их на сервер при помощи

```
await connection.InvokeAsync(имя_метода_обработки_сообщений_hub, отправляемые_данные);
```

5.3 Реализация комнат-чатов

5.3.1 Добавить в класс ChatHub два поля типа ConcurrentDictionary<string, string> для хранения данных о именах пользователей и подключениях к комнатам.

5.3.2 Реализовать метод JoinRoom(string roomName, string userName), в котором необходимо сохранить данные пользователя в соответствующие словари

```
// Получаем соединение
var connectionId = Context.ConnectionId;

// Привязать соединение к комнате и пользователю
ConnectionToRoom[connectionId] = roomName;
ConnectionToUser[connectionId] = userName;
// Добавляем соединение в группу по имени комнаты
await Groups.AddToGroupAsync(connectionId, roomName);
```

5.3.3 Далее при обработке сообщений на сервере можно использовать

```
// Проверка наличия имени пользователя и комнаты в ConcurrentDictionary
if (ConnectionToRoom.TryGetValue(connectionId, out var room) &&
    ConnectionToUser.TryGetValue(connectionId, out var user))
{
// Отправка данных клиентам в определенной группе
    await Clients.Group(room).SendAsync("ReceiveMessage", user, message);
}
```

5.3.4 Необходимо обработать отключение пользователя от группы при разрыве соединения с клиентом, для этого необходимо переопределить стандартный метод OnDisconnectedAsync и использовать в нем

```
await Groups.RemoveFromGroupAsync(connectionId, room);
```

6 Порядок выполнения работы

6.1 Запустить MS Visual Studio и создать оконное приложение C#.

6.2 Выполнить все задания из п.5 в одном решении.

6.3 Ответить на контрольные вопросы.

7 Содержание отчета

7.1 Титульный лист

7.2 Цель работы

7.3 Ответы на контрольные вопросы

7.4 Вывод

8 Контрольные вопросы

8.1 Что такое SignalR?

8.2 Каким образом осуществляется обмен данными между клиентом и сервером SignalR?