

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Кафедра Інформатики

Звіт з лабораторної роботи № 3
з дисципліни: «Обмін даними у Web-застосунках»
по темі: «gRPC»

Виконала:
ст. гр. ІТІНФ-21-2
Краснянська В.В.

Перевірив:
Шелест В.А.

Харків 2024

3 gRPC - виклик віддаленої процедури

3.1 Мета роботи: Вивчення взаємодії комунікації за допомогою технології відкладених процедур gRPC

3.2 Завдання для лабораторної роботи

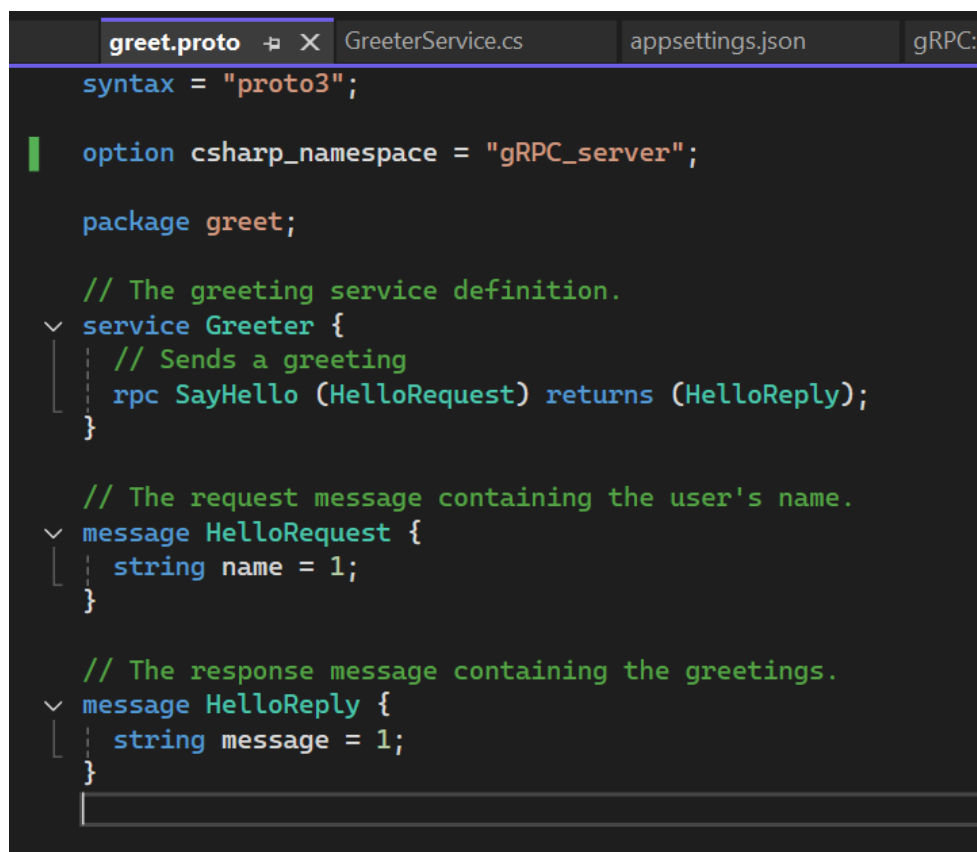
- Реалізувати сервер, який буде приймати запити за технологією gRPC
- Реалізувати клієнт, який має надсилати запити до серверу. Може бути реалізовано сервер-сервер за потреби

GitHub: <https://github.com/kras2v/ODUW>

Хід роботи

1. Розробка Protocol Buffers

Файл .proto описує структуру даних і визначає, які методи сервер підтримує, щоб клієнт міг викликати ці методи.



```
greet.proto GreeterService.cs appsettings.json gRPC:
syntax = "proto3";

option csharp_namespace = "gRPC_server";

package greet;

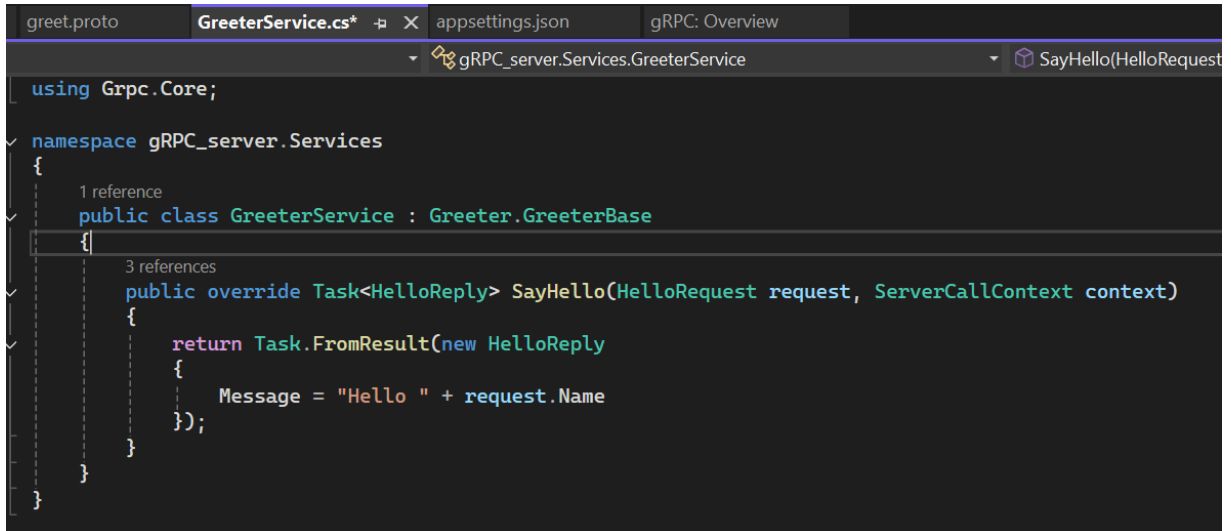
// The greeting service definition.
service Greeter {
  // Sends a greeting
  rpc SayHello (HelloRequest) returns (HelloReply);
}

// The request message containing the user's name.
message HelloRequest {
  string name = 1;
}

// The response message containing the greetings.
message HelloReply {
  string message = 1;
}
```

2. Створення gRPC-сервера

Сервер буде отримувати та обробляти запити від клієнта. Під час серіалізації було визначено що наш сервер буде виконувати функцію SayHello, яку було реалізовано нижче, функція буде отримувати HelloRequest, парсити з нього ім'я користувача та надавати HelloReply:

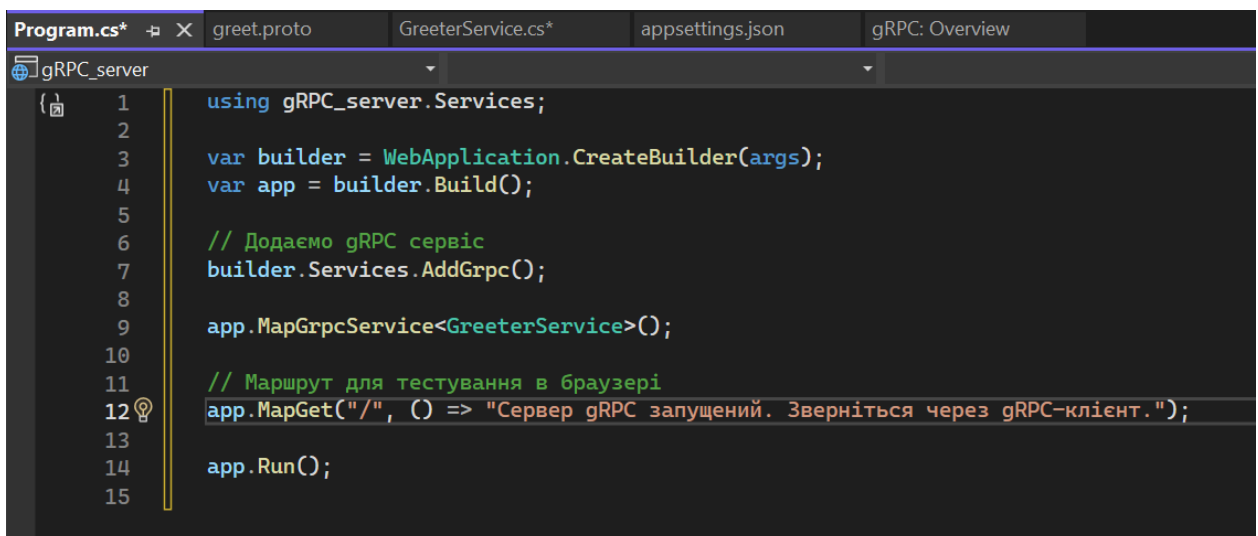


```
greet.proto GreeterService.cs* x appsettings.json gRPC: Overview
gRPC_server.Services.GreeterService SayHello(HelloRequest request, ServerCallContext context)

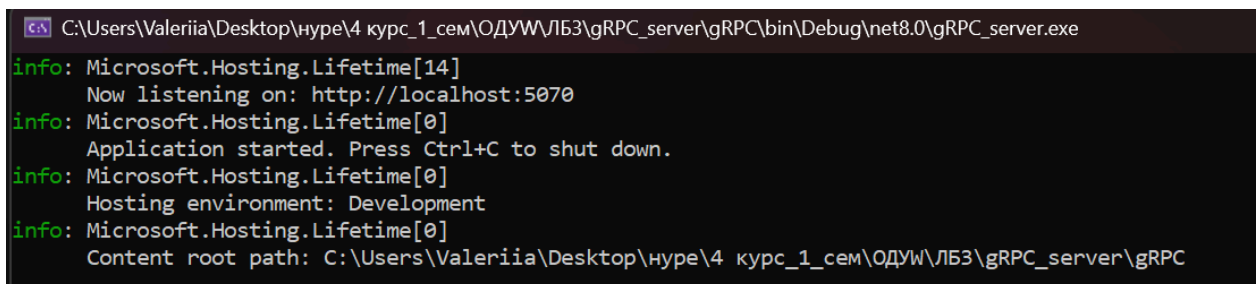
using Grpc.Core;

namespace gRPC_server.Services
{
    1 reference
    public class GreeterService : Greeter.GreeterBase
    {
        3 references
        public override Task<HelloReply> SayHello(HelloRequest request, ServerCallContext context)
        {
            return Task.FromResult(new HelloReply
            {
                Message = "Hello " + request.Name
            });
        }
    }
}
```

3. Запуск та налаштування сервера



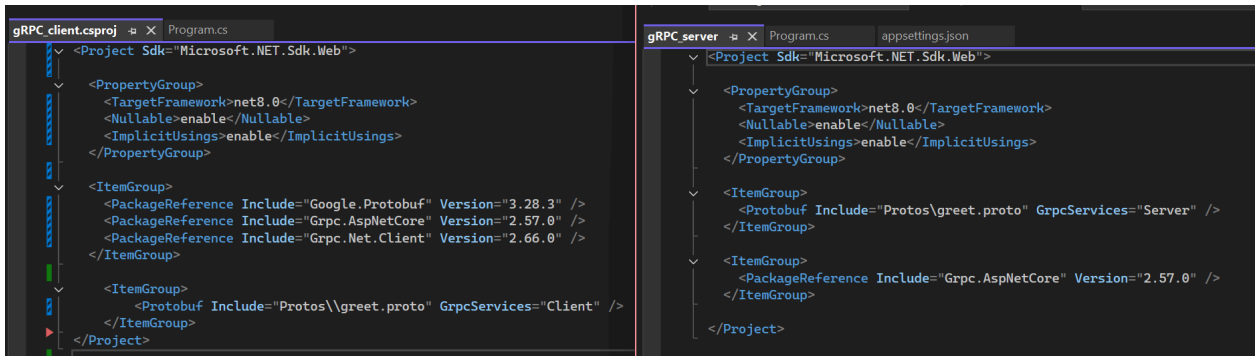
```
Program.cs* x greet.proto GreeterService.cs* appsettings.json gRPC: Overview
gRPC_server
{
    1 using gRPC_server.Services;
    2
    3 var builder = WebApplication.CreateBuilder(args);
    4 var app = builder.Build();
    5
    6 // Додаємо gRPC сервіс
    7 builder.Services.AddGrpc();
    8
    9 app.MapGrpcService<GreeterService>();
    10
    11 // Маршрут для тестування в браузері
    12 app.MapGet("/", () => "Сервер gRPC запущений. Зверніться через gRPC-клієнт.");
    13
    14 app.Run();
    15
}
```



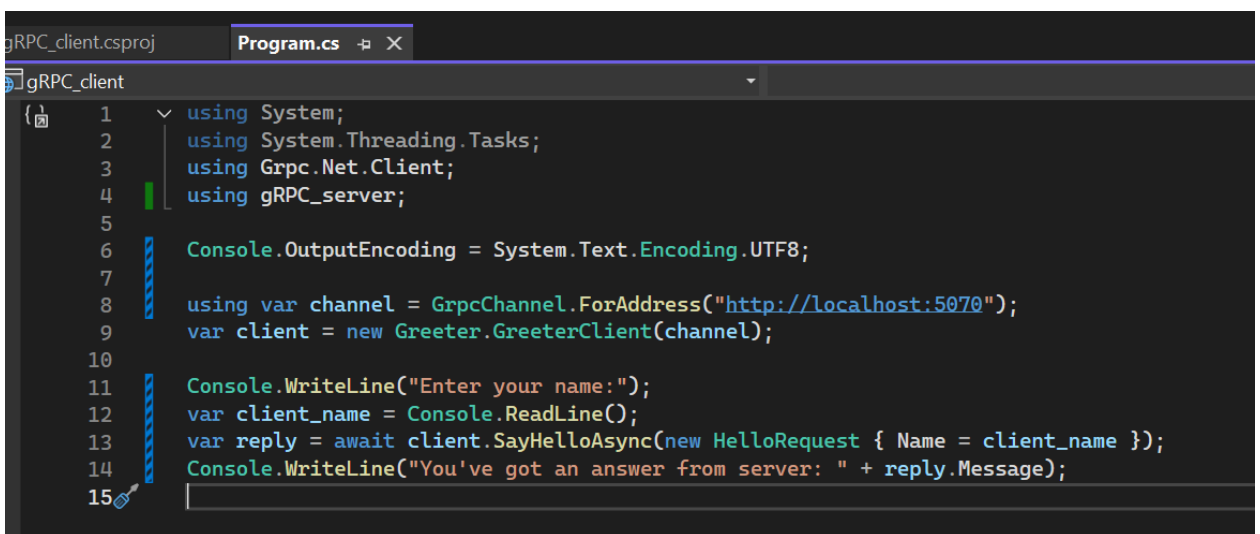
```
C:\Users\Valeriia\Desktop\нурє\4 курс_1_сем\ОДУ\ЛБ3\gRPC_server\gRPC\bin\Debug\net8.0\gRPC_server.exe
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5070
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Users\Valeriia\Desktop\нурє\4 курс_1_сем\ОДУ\ЛБ3\gRPC_server\gRPC
```

4. Конфігурація Protocol Buffers

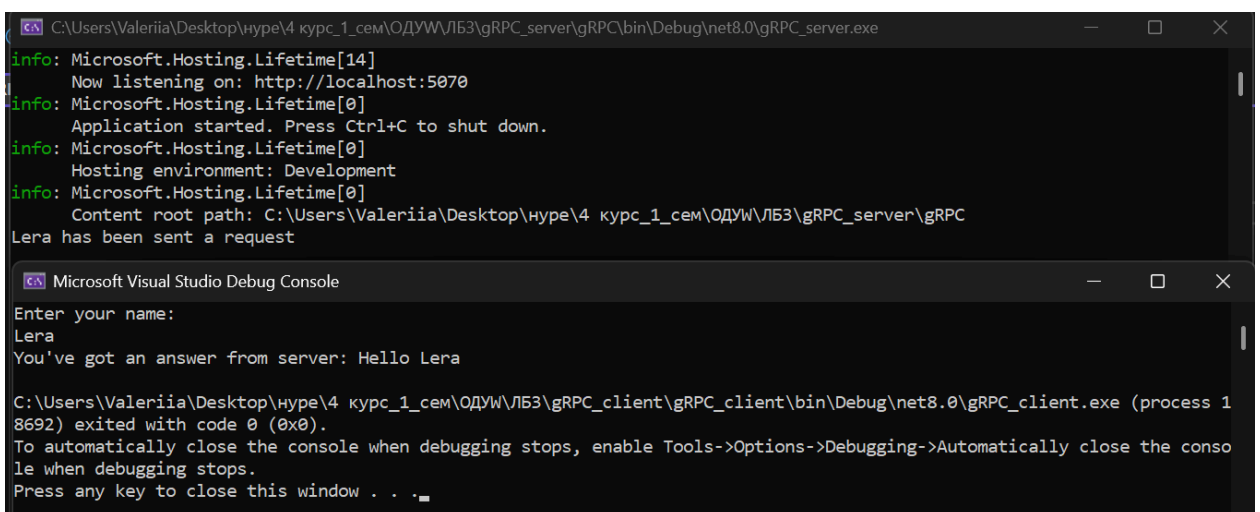
Також для роботи як сервера, так і клієнта proto файли мають додаватись до файлу конфігурацій проєкту (csproj файл) як protobuf із позначкою, що це файл серверу до генерації відповідного коду.



5. Розробка клієнта



Результат роботи:



Висновок: під час виконання лабораторної роботи, було засвоєно основні навички використання gRPC