Tutorial

gRPC for .NET Developers

Zahra Bayat

Ali Bayat

gRPC for

.NET Developers

https://www.linkedin.com/in/zahrabayat/

https://www.linkedin.com/in/alibayatgh/

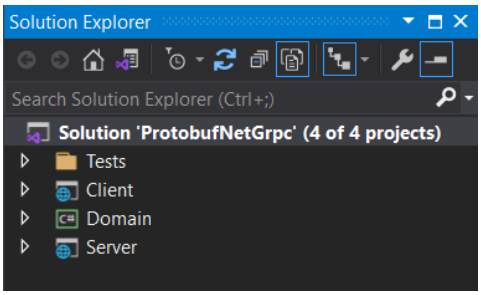
Contents

Welcome to the gRPC tutorial on Asp.Net Core

In this tutorial I want to explain how to use ***gRPC*** using the ***protobuf-net.Grpc*** library in ***ASP.NET Core Web API***.

Let's go.

Project structure

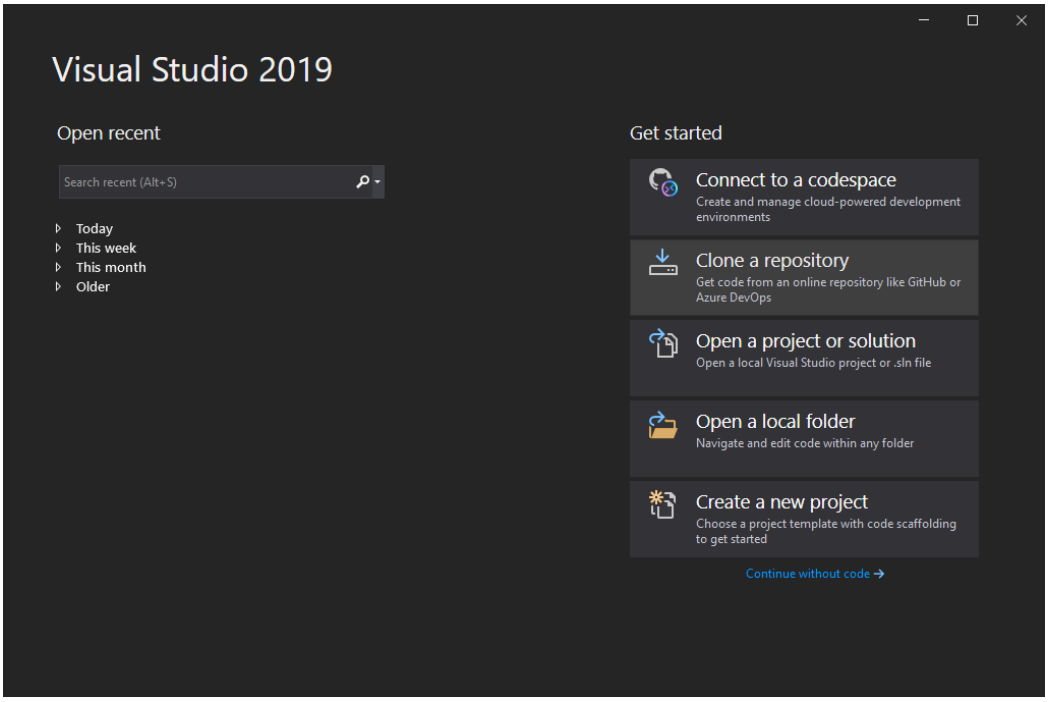


There are three projects in this solution

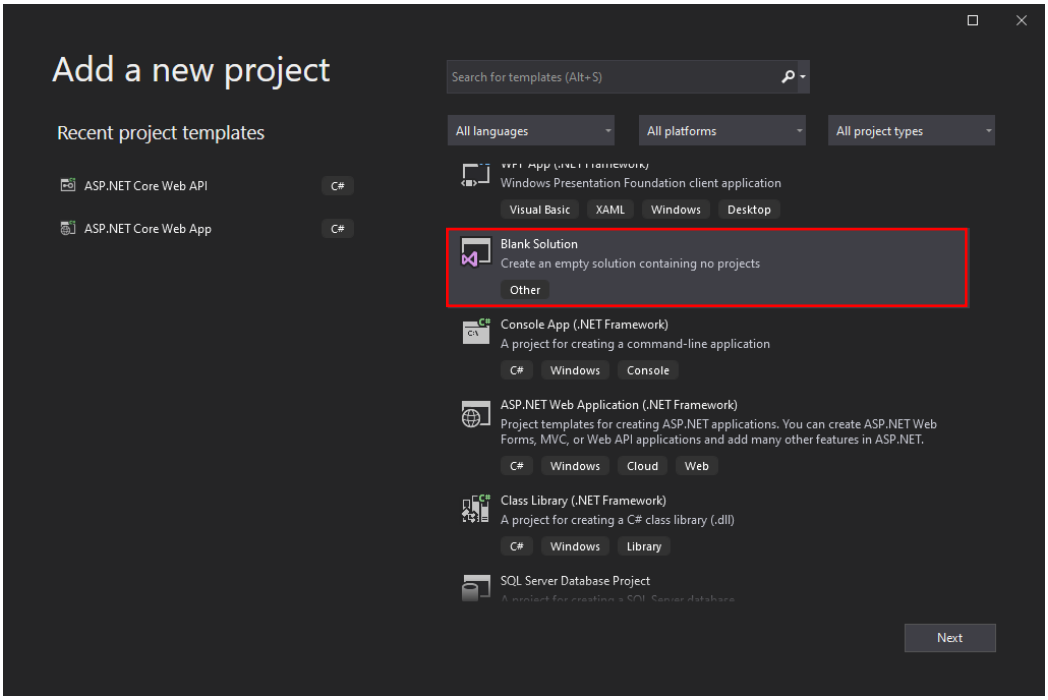
- **Domain:** This project includes input and output models of ***gRPC*** services methods and interfaces that these services implement.
- **Server:** This project contains the services that we want to connect to with ***gRPC***.
- **Client:** This project contains ***APIs*** that call the ***Server*** project services.

Step 1: Create a blank solution

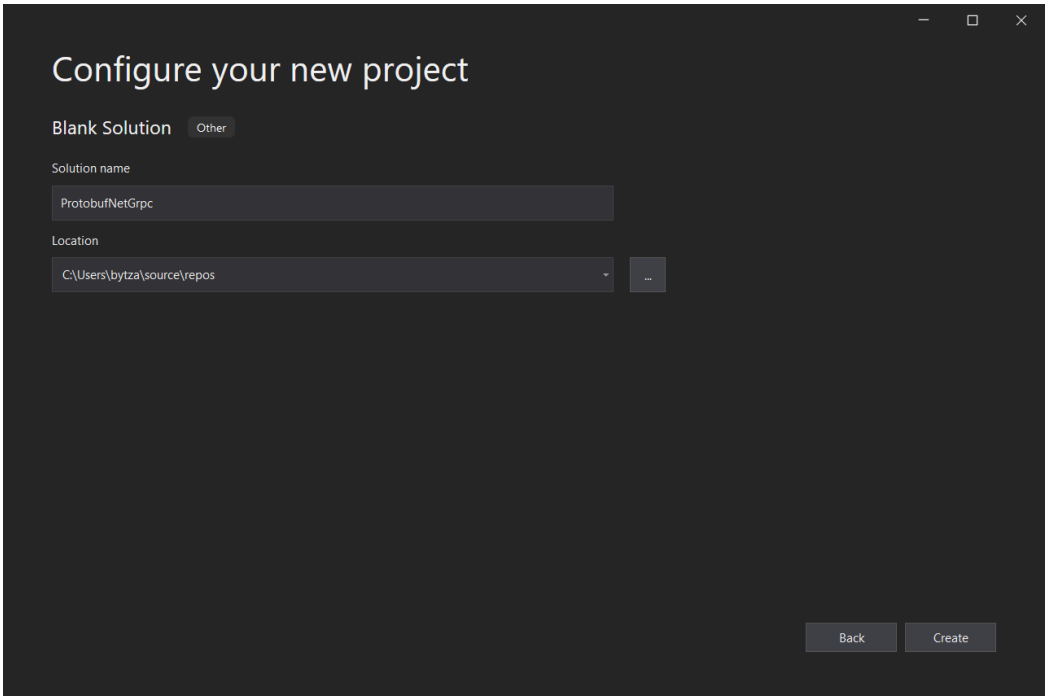
Open ***Visual Studio 2019*** and click on ***Create a new project***.



Now select the ***Blank Solution*** option and click ***Next***.

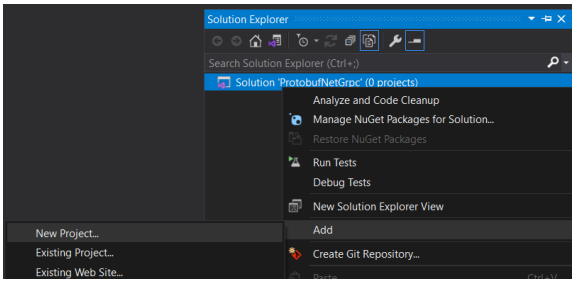


Now name this solution ***ProtobufNetGrpc*** and click ***Create***.

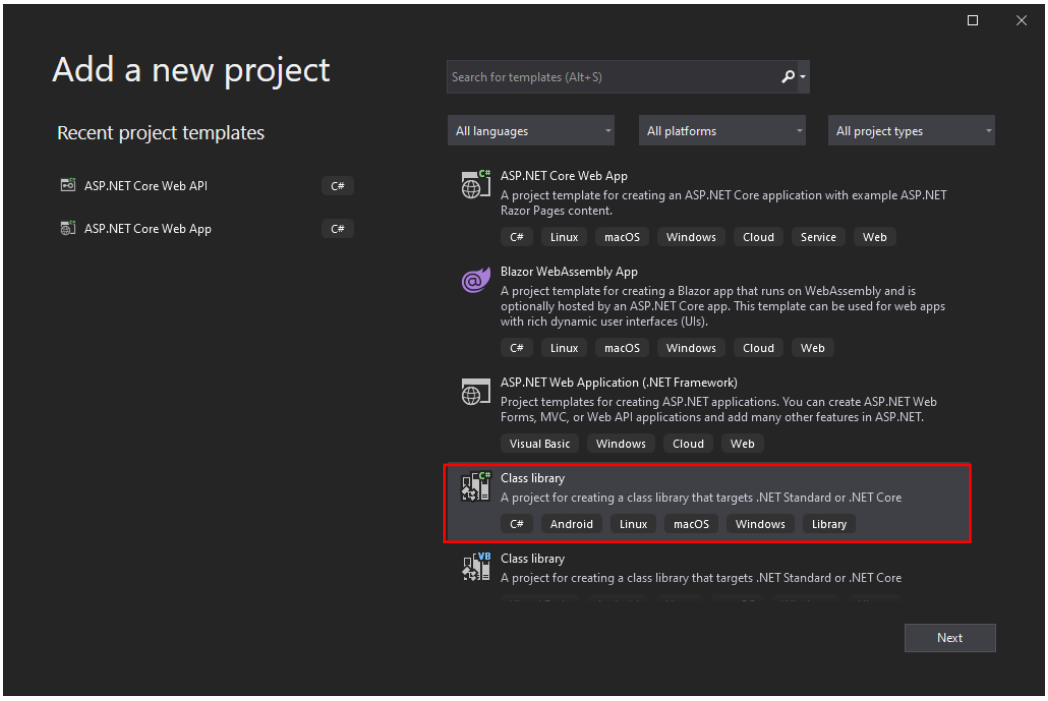


Step 2: Add the Domain project to the solution

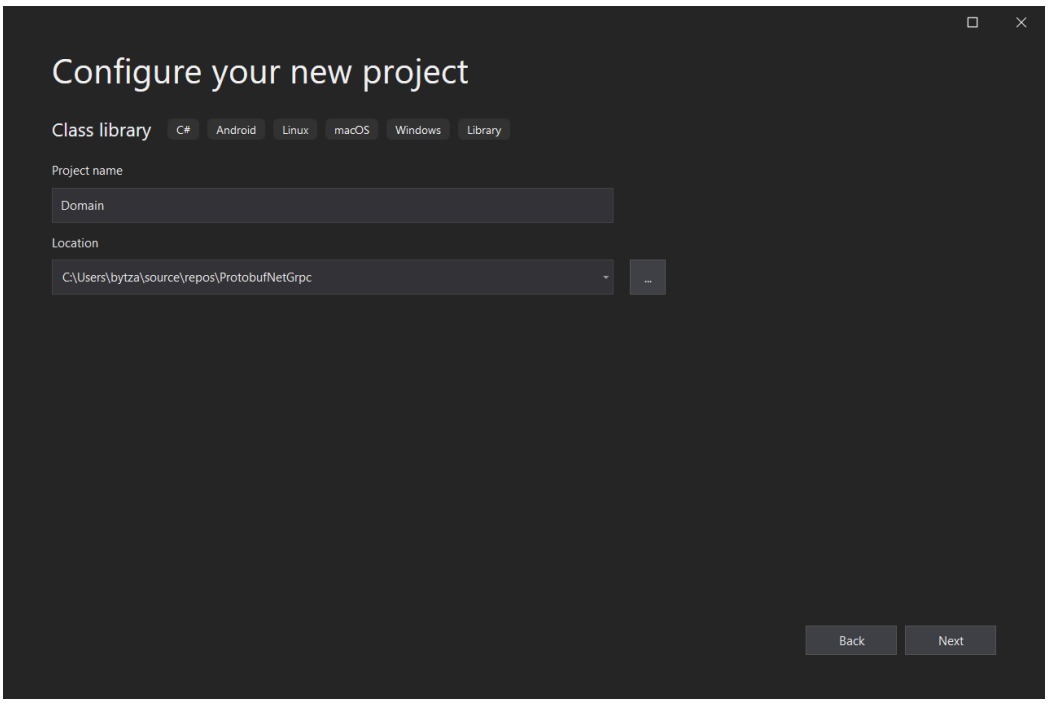
Right-click on the solution and select ***Add → New project***.



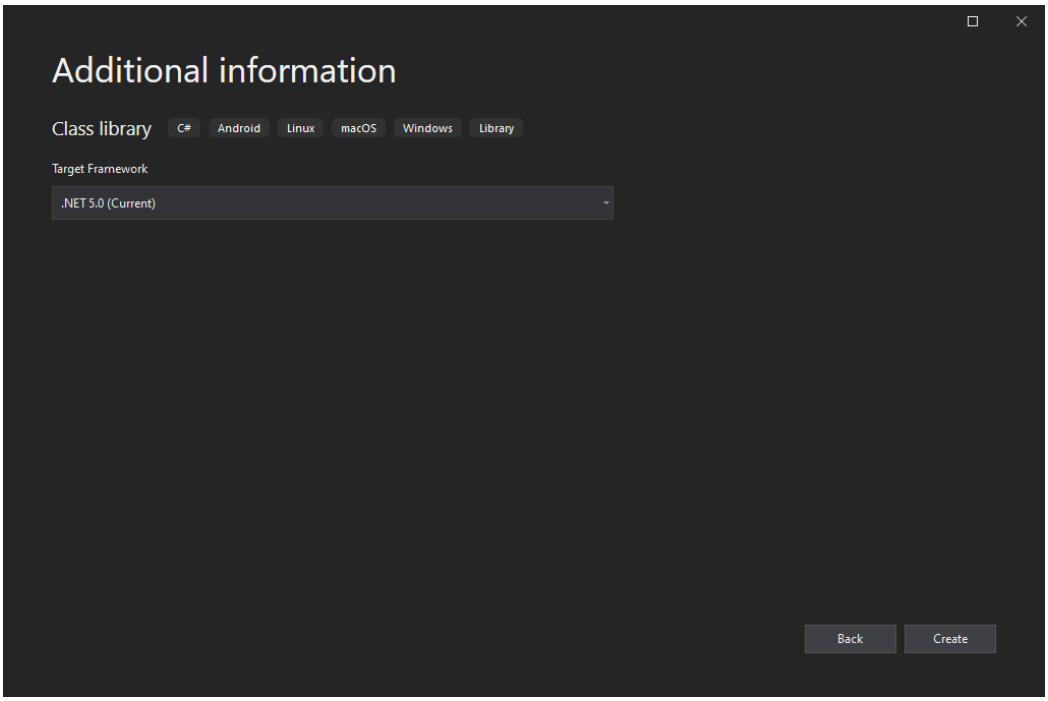
Select ***Class Library*** and click ***Next***.



In the next box, name the project ***Domain*** and click ***Next***.



Now select ***.NET 5.0*** and click ***Create***



Step 1-1:

Before doing anything, we must add the required packages to the project. So, right-click on the project and select ***Edit Project File***.

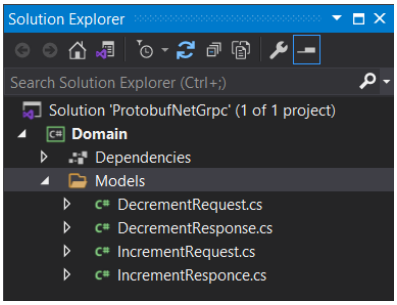
Now, change this file.

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>net5.0</TargetFramework>
  </PropertyGroup>

  <ItemGroup>
      <PackageReference Include="System.ServiceModel.Primitives" Version="4.8.1" />
      <PackageReference Include="protobuf-net.Grpc.AspNetCore" Version="1.0.152" />
       <PackageReference Include="protobuf-net.Grpc.AspNetCore.Reflection" Version="1.0.152"
       />
      <PackageReference Include="protobuf-net.Grpc" Version="1.0.152" />
       <PackageReference Include="Grpc.Net.Client" Version="2.37.0" />
  </ItemGroup>

</Project>
```

Note: Do not forget to add all four packages and after that save this file to add packages to the project.

Step 1-2:

You must have defined input and output models for every services method. These classes must be serialized. So, create a ***Models*** folder and add the following classes to it.



IncrementRequest class:

```
using System.Runtime.Serialization;

namespace Domain.Models
{
    [DataContract]
    public class IncrementRequest
    {
        [DataMember(Order = 1)]
        public int Inc { get; set; }
    }
}
```

IncrementResponce class:

```
using System.Runtime.Serialization;

namespace Domain.Models
{
    [DataContract]
    public class IncrementResponce
    {
        [DataMember(Order = 1)]
        public int Result { get; set; }
    }
}
```

If you want input or output models to have a parameter constructor, you must also define a parameter less constructor in the class.

DecrementRequest class:

```
using System;
using System.Runtime.Serialization;

namespace Domain.Models
{
    [DataContract]
    public class DecrementRequest
    {
        [Obsolete("Not allowed to use this for instantiation")]
        public DecrementRequest()
        {
```

```
        }
        public DecrementRequest(int dec)
        {
            Dec = dec;
        }

        [DataMember(Order = 1)]
        public int Dec { get; set; }
    }
}
```

DecrementResponse class:

```
using System;
using System.Runtime.Serialization;

namespace Domain.Models
{
    [DataContract]
    public class DecrementResponse
    {
        [Obsolete("Not allowed to use this for instantiation")]
        public DecrementResponse()
        {

        }
        public DecrementResponse(int result)
        {
            Result = result;
        }

        [DataMember(Order = 1)]
        public int Result { get; set; }
    }
}
```

Note: Be sure to leave the order numbers in a row.

Step 1-3:

In this step we need to define an ***ICounterService*** interface so that ***CounterService*** can implement it. This service is located in the Server project. In fact, ***ICounterService*** is the interface between the project server and the client.

```
using Domain.Models;
using ProtoBuf.Grpc.Configuration;
using System.ServiceModel;

namespace Domain.IServices
{
    [ServiceContract(Name = "CounterService")]
    public interface ICounterService
    {
        [SimpleRpcExceptions]
```

```
        IncrementResponce Increment(IncrementRequest request);

        [SimpleRpcExceptions]
        DecrementResponse Decrement(DecrementRequest request);
    }
}
```

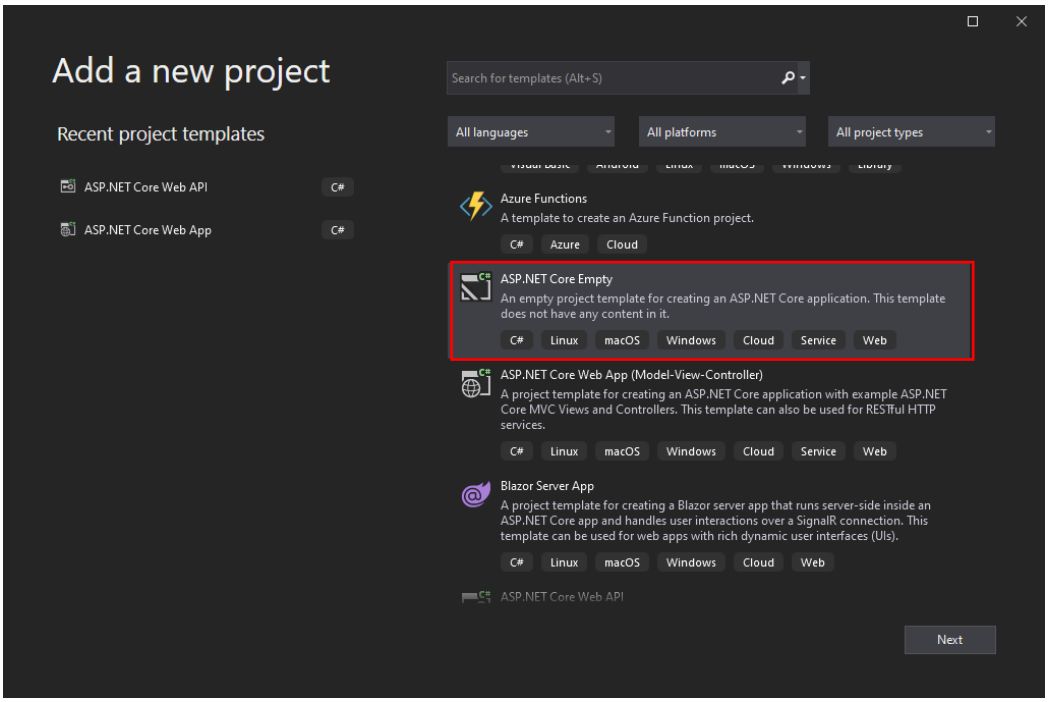
Note: that you must specify the name property in the attribute ***[ServiceContract (Name = "CounterService")].***

Note: The ***[SimpleRpcExceptions]*** attribute is used to manage exceptions. If you have the possibility of an exception in your method, you must place this attribute above your method.

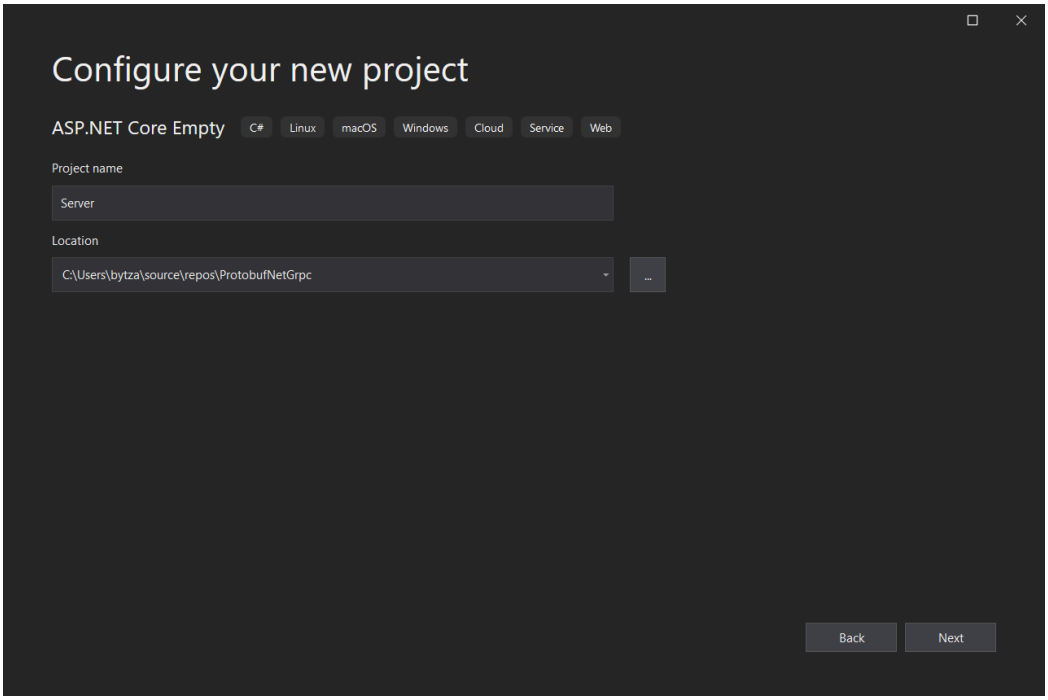
Step 3: Add the Server project to the solution

Right-click on the solution and select ***Add → New project***.

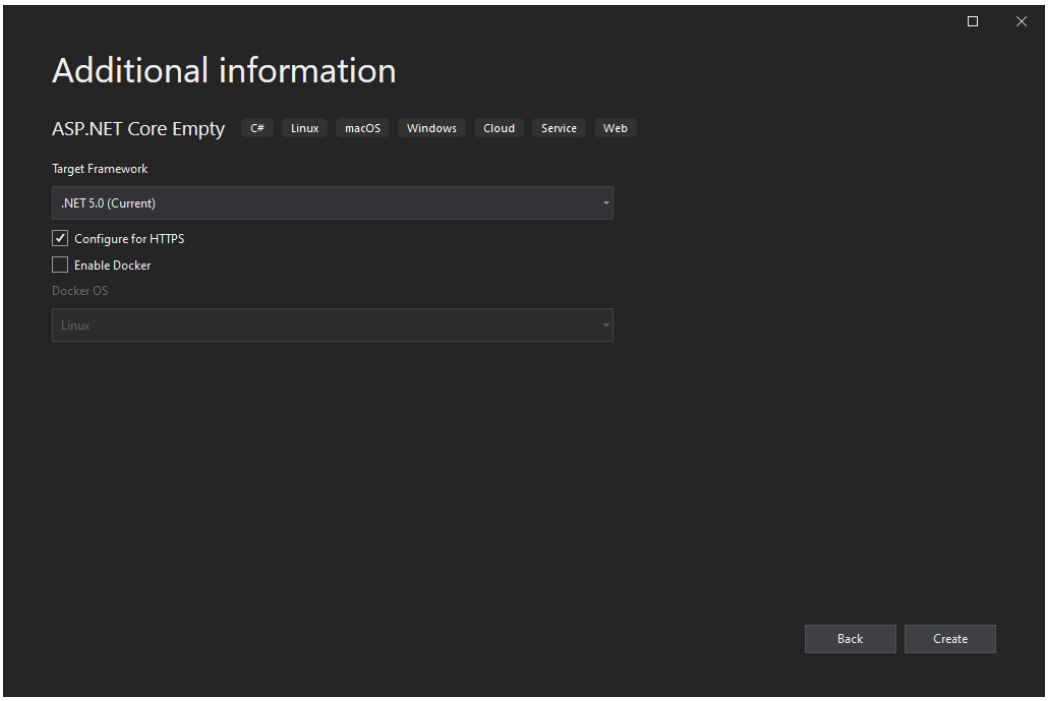
Now, select ***ASP.NET Core Empty*** and click ***Next***.



In the next box, name the project ***Server*** and click the ***Next***.



Now, select ***.NET 5.0*** and click ***Create***.



Step 3-1:

Before doing anything, we need to add a reference from the ***Domain*** project to the current project. So, right-click on the project and select ***Edit Project File***.

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net5.0</TargetFramework>
  </PropertyGroup>
```

```
<ItemGroup>
  <ProjectReference Include="..\Domain\Domain.csproj" />
</ItemGroup>

</Project>
```

Step 3-2:

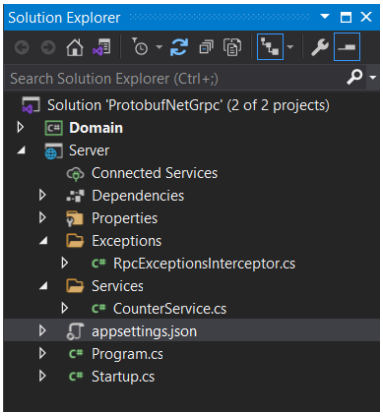
Now we need to create the ***CounterService*** service. This service must implement the ***ICounterService*** interface.

```
using Domain.IServices;
using Domain.Models;
using System;

namespace Server.Services
{
    public class CounterService : ICounterService
    {
        private int counter = 0;

        public IncrementResponce Increment(IncrementRequest request)
        {
            try
            {
                if (request.Inc == 0)
                    throw new NullReferenceException("IncrementRequest model is null");

                counter += request.Inc;

                var result = new IncrementResponce { Result = counter };
                return result;
            }
            catch (Exception)
            {
                throw;
            }
        }
        public DecrementResponse Decrement(DecrementRequest request)
        {
            try
            {
                if (request.Dec == 0)
                    throw new NullReferenceException("DecrementRequest model is null");

                counter -= request.Dec;

                var result = new DecrementResponse(counter);
                return result;
            }
            catch (Exception)
            {
                throw;
            }
```

```
        }
    }
}
```

Step 3-3:

Because exceptions in this library do not cross layers, you must define an Interceptor class.




```
using Grpc.Core;
using ProtoBuf.Grpc.Configuration;
using System;

namespace Server.Exceptions
{
    public class RpcExceptionsInterceptor : ServerExceptionsInterceptorBase
    {
        private RpcExceptionsInterceptor() { }
        private static RpcExceptionsInterceptor? _sInstance;

        public static RpcExceptionsInterceptor Instance => _sInstance ??= new
        RpcExceptionsInterceptor();

        private static bool ShouldWrap(Exception exception, out Status status)
        {
             status = new Status(
             StatusCode.Internal
             , exception.Message, exception);

            return true;
        }

        protected override bool OnException(Exception exception, out Status status)
            => ShouldWrap(exception, out status);

    }
}
```

Step 3-4:

Now you need to make the following changes in the ***Startup*** class:

- Register the ***AddCodeFirstGrpc*** and ***AddCodeFirstGrpcReflection*** services in the ***ConfigureServices*** method.

- Register ***RpcExceptionsInterceptor*** and add ***AddCodeFirstGrpc*** service to ***Interceptors***.
- Register the ***CounterService*** service.
- In the ***Configure*** method, you must also specify ***ICounterService*** and ***MapCodeFirstGrpcReflectionService*** as endpoints.

Modify Startup class:

```
using Domain.IServices;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.DependencyInjection.Extensions;
using Microsoft.Extensions.Hosting;
using ProtoBuf.Grpc.Server;
using Server.Exceptions;
using Server.Services;
using System.IO.Compression;

namespace Server
{
    public class Startup
    {
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddCodeFirstGrpc(config =>
            {
                config.Interceptors.Add(typeof(RpcExceptionsInterceptor));
                 config.ResponseCompressionLevel = CompressionLevel.Optimal;
            });

            services.TryAddSingleton(RpcExceptionsInterceptor.Instance);
            services.AddCodeFirstGrpcReflection();

            services.AddScoped<ICounterService, CounterService>();

        }

        public const string HTTP_SERVER_IS_RUNNING_MESSAGE = "Http server is running";
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }

            app.UseRouting();

            app.UseEndpoints(endpoints =>
            {
                endpoints.MapGrpcService<ICounterService>();
                endpoints.MapCodeFirstGrpcReflectionService();
                endpoints.MapGet("/", async context =>
```

```
                {
                    await context.Response.WriteAsync(z);
                });
            });
        }
    }
}
```

Step 3-5:

At this point we need to configure ***Kestrel***.

```
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Server.Kestrel.Core;
using Microsoft.Extensions.Hosting;

namespace Server
{
    public class Program
    {
        public static void Main(string[] args)
        {
            CreateHostBuilder(args).Build().Run();
        }

        public static IHostBuilder CreateHostBuilder(string[] args) =>
             Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.ConfigureKestrel(options =>
                    {
                        // Set properties and call methods on options
                        options.Limits.Http2.MaxStreamsPerConnection = 100;
                        options.Limits.Http2.HeaderTableSize = 4096;
                        options.ListenLocalhost(14001);
                        options.ConfigureEndpointDefaults(p => p.Protocols =
                        HttpProtocols.Http2);

                    }).UseStartup<Startup>();
                });
    }
}
```

Step 3-6:

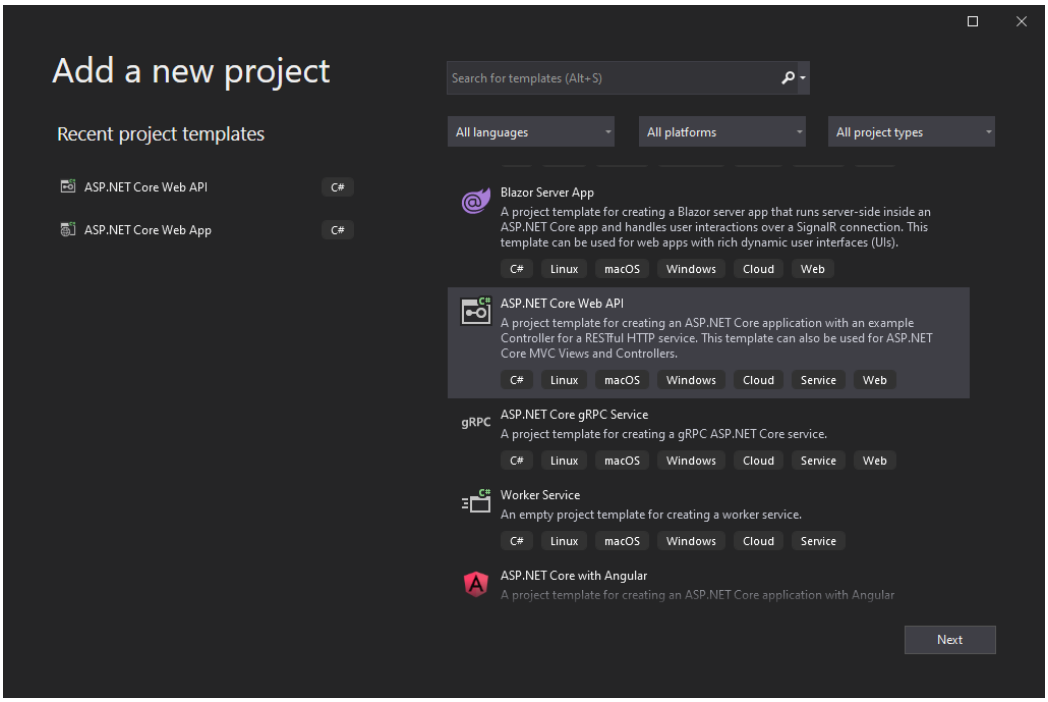
Finally, you need to change the ***appsettings.json*** file to configure Kestrel.

```
{

  "Kestrel": {
    "EndpointDefaults": {
      "Protocols": "Http2"
    }
  }
}
```

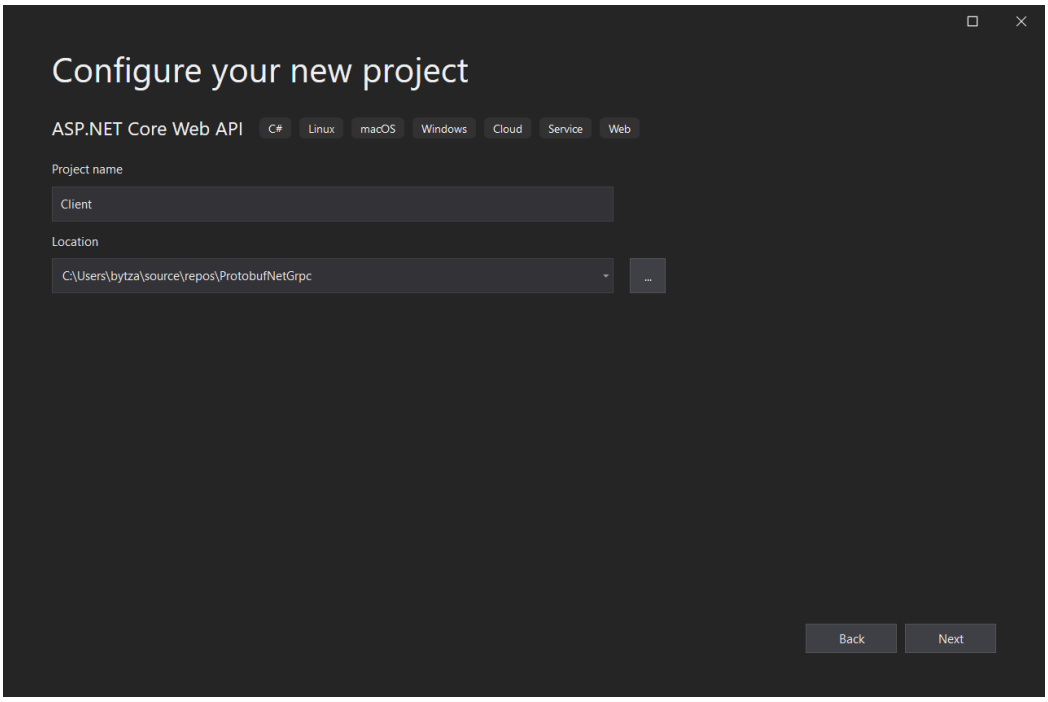
Step 4: Add the Client project to the solution

Right-click on the solution and select **Add-> New project**.

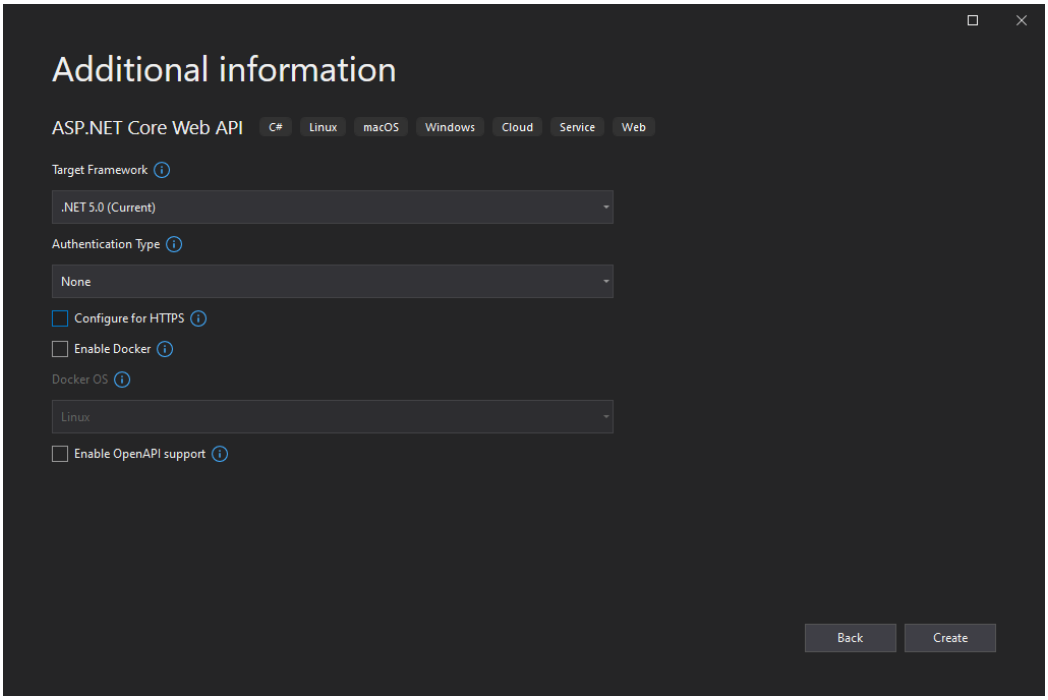
Now, select ***ASP.NET Core Web API*** and click ***Next***.



In the next box, name the project ***Client*** and click the ***Create***.



Now select ***.NET 5.0*** and click ***Create***



Step 4-1:

Before doing anything, we need to add a reference from the ***Domain*** project to the current project. So, right-click on the project and select ***Edit Project File***.

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net5.0</TargetFramework>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Swashbuckle.AspNetCore" Version="5.6.3" />
  </ItemGroup>

  <ItemGroup>
    <ProjectReference Include="..\Domain\Domain.csproj" />
  </ItemGroup>

</Project>
```

Now, we must define in the ***appsettings.json*** file the port we specified in the ***Program*** class in the ***Server*** project.

```
{
  "GrpcServerUrl": "http://localhost:14001"
}
```

Step 4-2:

Now, we need to define a class called ***ServerGrpcConfig***. This class is for communicating with the ***Server*** project.

```
using Domain.IServices;
using Grpc.Net.Client;
```

```
using ProtoBuf.Grpc.Client;

namespace Client
{
    public class ServerGrpcConfig
    {
        private readonly GrpcChannel _channel;

        public ServerGrpcConfig(string arzshomarOnUrl)
        {
            GrpcClientFactory.AllowUnencryptedHttp2 = true;
            _channel = GrpcChannel.ForAddress(arzshomarOnUrl);

        }
        public ICounterService CreateCounterServiceGrpc()
        {
            return _channel.CreateGrpcService<ICounterService>();
        }
    }
}
```

Step 4-3:

In this step, we need to fetch the value of the ***GrpcServerUrl*** key from the ***appsetting*** file and pass it to the ***ServerGrpcConfig*** class constructor. To do this we need to register ***ServerGrpcConfig*** in the startup.

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

namespace Client
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        public void ConfigureServices(IServiceCollection services)
        {
            services.AddSingleton(new
            ServerGrpcConfig(Configuration.GetValue<string>("GrpcServerUrl")));
            services.AddControllers();
        }

        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
```

```
            app.UseHttpsRedirection();

            app.UseRouting();

            app.UseAuthorization();

            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllers();
            });
        }
    }
}
```

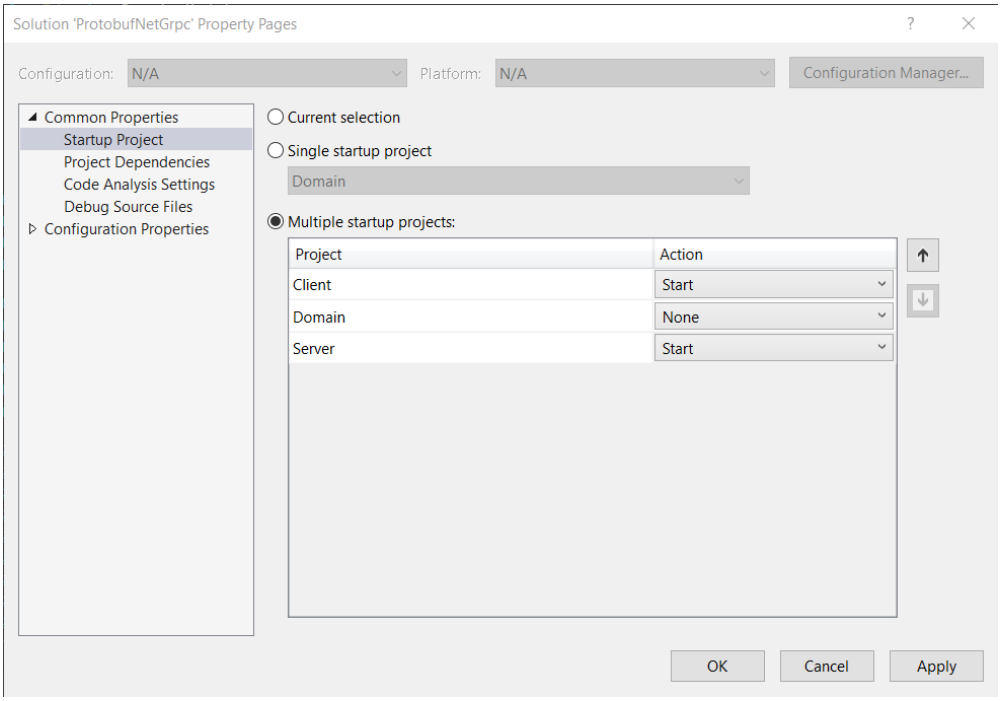
Step 4-4:

Finally, you need to create a controller called ***HomeController*** in the controller folder.
This controller has two action methods called ***Increment*** and ***Decrement***.
These action methods call the Increment and Decrement methods of the ***ICounterService*** service.

```
using Domain.IServices;
using Domain.Models;
using Grpc.Core;
using Microsoft.AspNetCore.Mvc;

namespace Client.Grpc.Controllers
{
    [ApiController]
    [Route("[controller]")]
    public class HomeController : ControllerBase
    {
        private ICounterService _serverGrpcConfig;

        public HomeController(ServerGrpcConfig serverGrpcConfig)
        {
            _serverGrpcConfig = serverGrpcConfig.CreateCounterServiceGrpc();
        }

        [Route("Increment/{inc:int}")]
        public IActionResult Increment(int inc)
        {
            try
            {
                var result = _serverGrpcConfig.Increment(new IncrementRequest {Inc = inc });

                return Ok(result);
            }
            catch (RpcException ex)
            {
                return BadRequest(ex.Message);
            }
        }
```

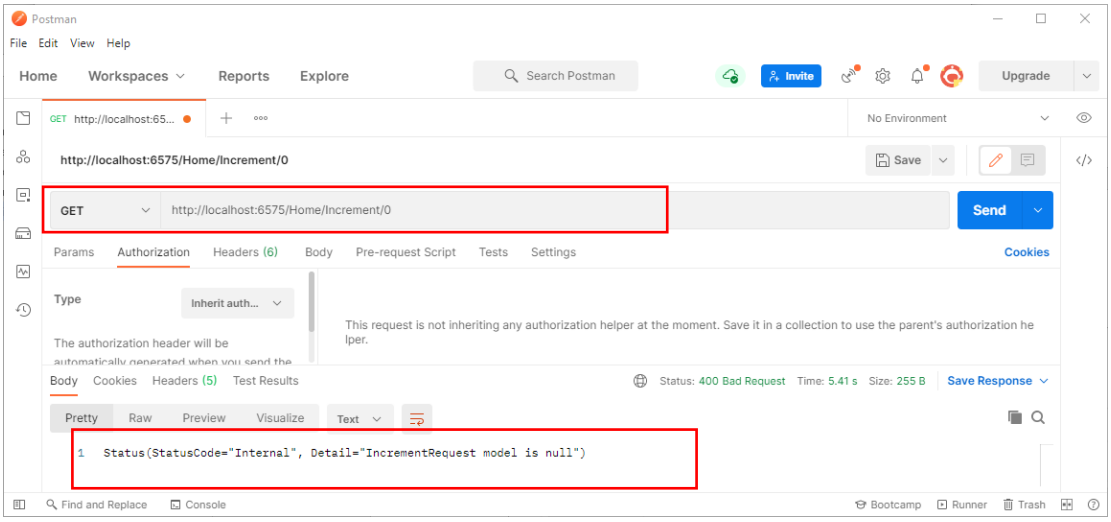
```
        [Route("Decrement/{dec:int}")]
        public IActionResult Decrement(int dec)
        {
            try
            {
                var result = _serverGrpcConfig.Decrement(new DecrementRequest(dec));

                return Ok(result);
            }
            catch (RpcException ex)
            {
                return BadRequest(ex.Message);
            }
        }
    }
}
```

You must run the ***Server*** and ***Client*** projects together to see the result. To do this, you can right-click on the solution and select ***Properties***. Now, like the picture, put these two projects in start mode and click ***OK.***



Then send a request to the ***Client*** with the ***Postman*** as shown below.

http://localhost:6575/Home/Increment/0



As you can see, because the ***API*** input data is a zero number, an exception was thrown that we were able to pass the error message through ***gRPC***.