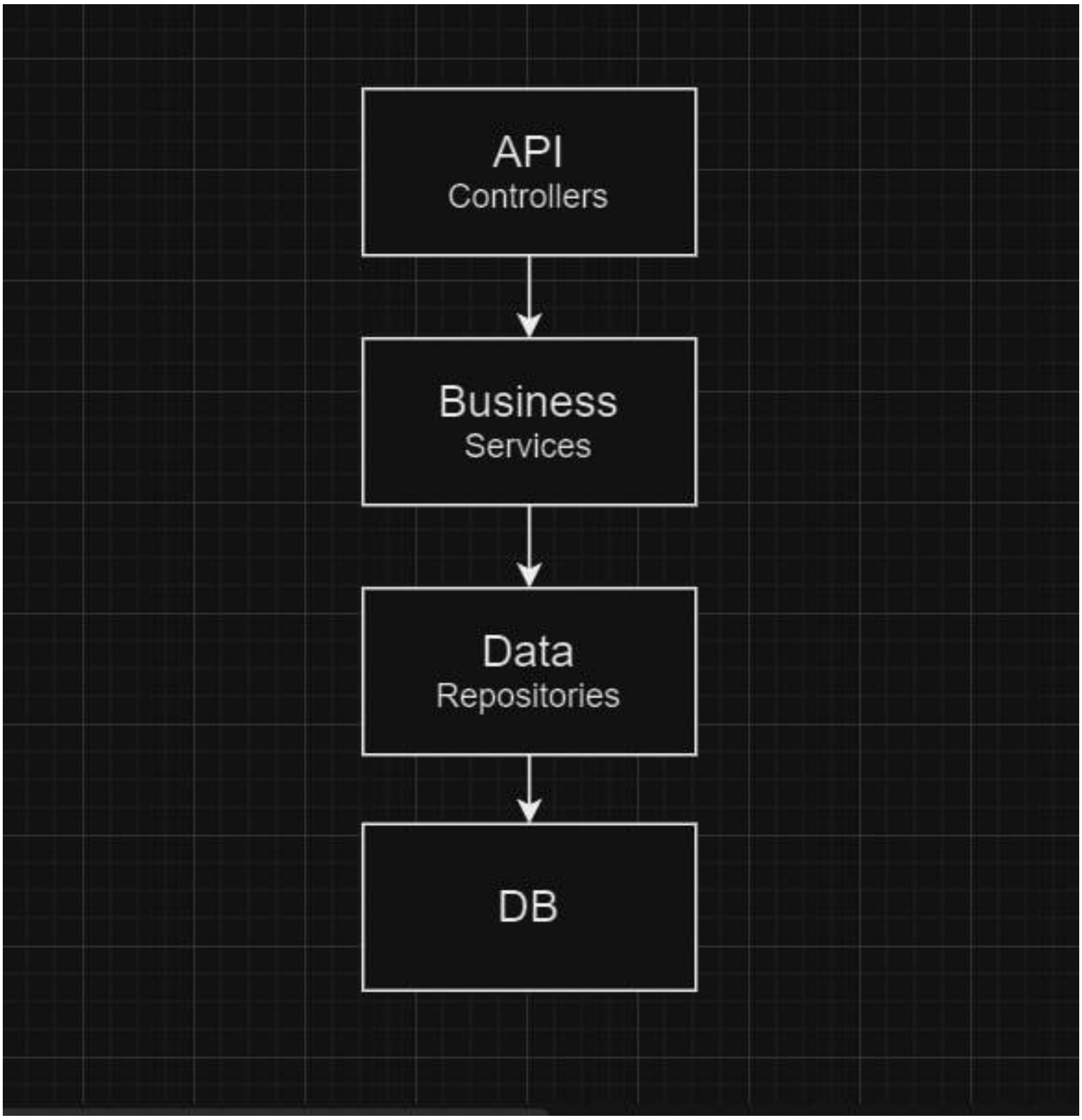
## Підхід до побудови архітектури проекту:

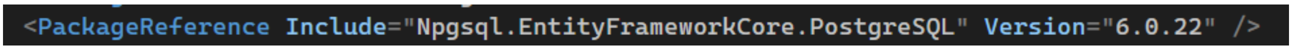
Layered architecture



## База даних:

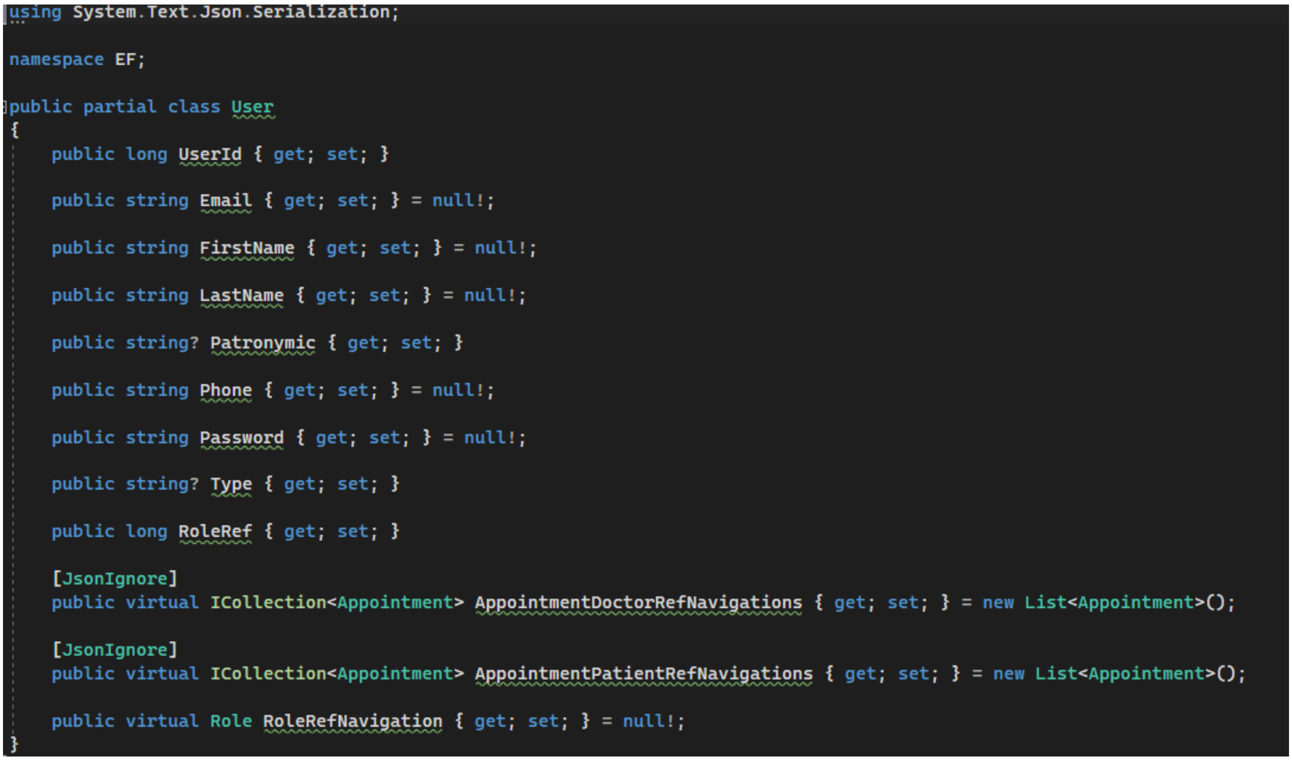
PostgreSQL (у ролі сервера використовуємо Neon)

## Entity Framework:



```
<PackageReference Include="Npgsql.EntityFrameworkCore.PostgreSQL" Version="6.0.22" />
```

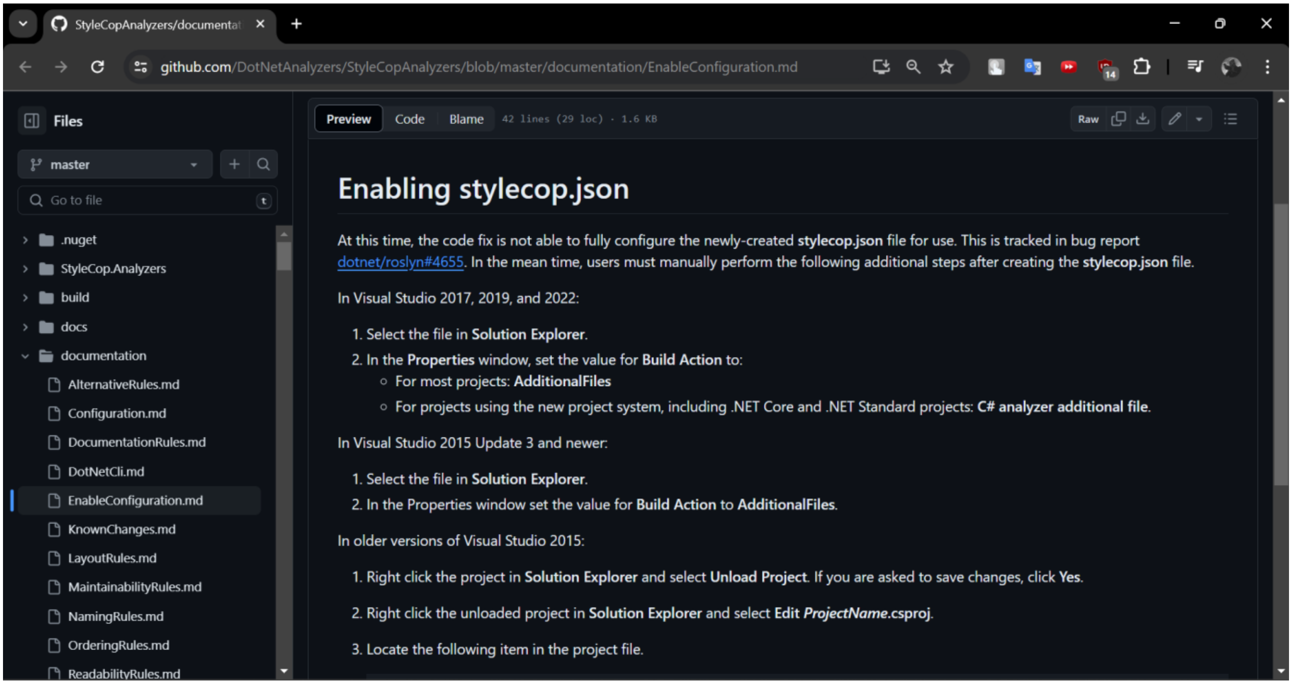
Приклад entity:



```
using System.Text.Json.Serialization;

namespace EF;

public partial class User
{
    public long UserId { get; set; }

    public string Email { get; set; } = null!;

    public string FirstName { get; set; } = null!;

    public string LastName { get; set; } = null!;

    public string? Patronymic { get; set; }

    public string Phone { get; set; } = null!;

    public string Password { get; set; } = null!;

    public string? Type { get; set; }

    public long RoleRef { get; set; }

    [JsonIgnore]
    public virtual ICollection<Appointment> AppointmentDoctorRefNavigations { get; set; } = new List<Appointment>();

    [JsonIgnore]
    public virtual ICollection<Appointment> AppointmentPatientRefNavigations { get; set; } = new List<Appointment>();

    public virtual Role RoleRefNavigation { get; set; } = null!;
}
```

## Stylecop:

Налаштування Stylecop відповідно до документації:



### Enabling stylecop.json

At this time, the code fix is not able to fully configure the newly-created **stylecop.json** file for use. This is tracked in bug report dotnet/roslyn#4655. In the mean time, users must manually perform the following additional steps after creating the **stylecop.json** file.

In Visual Studio 2017, 2019, and 2022:

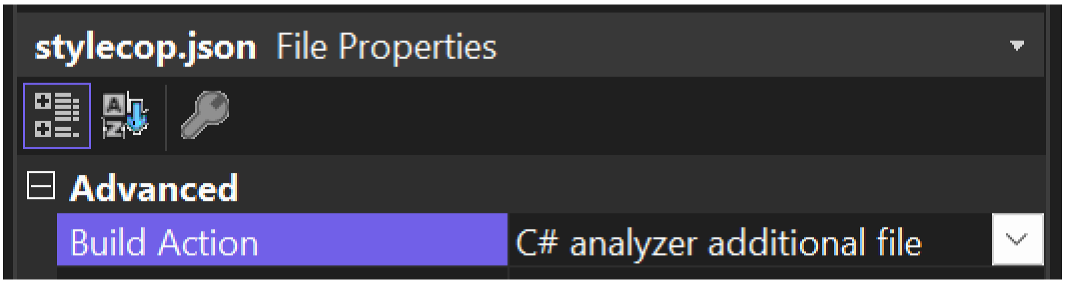
1. Select the file in **Solution Explorer**.
2. In the **Properties** window, set the value for **Build Action** to:
    - For most projects: **AdditionalFiles**
    - For projects using the new project system, including .NET Core and .NET Standard projects: **C# analyzer additional file**.

In Visual Studio 2015 Update 3 and newer:

1. Select the file in **Solution Explorer**.
2. In the Properties window set the value for **Build Action** to **AdditionalFiles**.

In older versions of Visual Studio 2015:

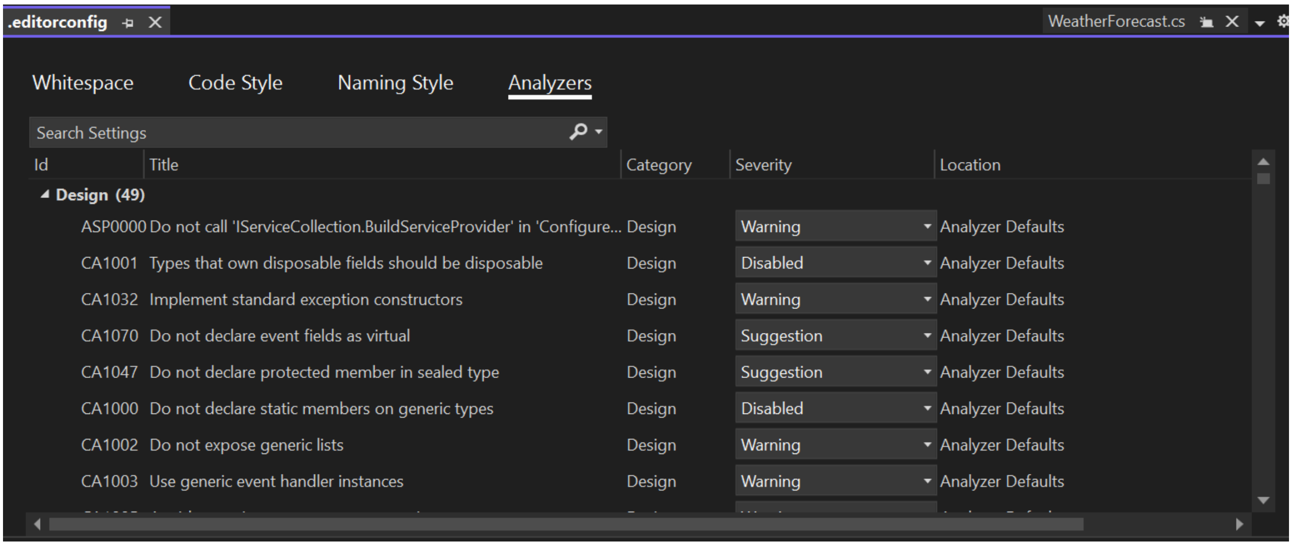
1. Right click the project in **Solution Explorer** and select **Unload Project**. If you are asked to save changes, click **Yes**.
2. Right click the unloaded project in **Solution Explorer** and select **Edit *ProjectName*.csproj**.
3. Locate the following item in the project file.



Advanced

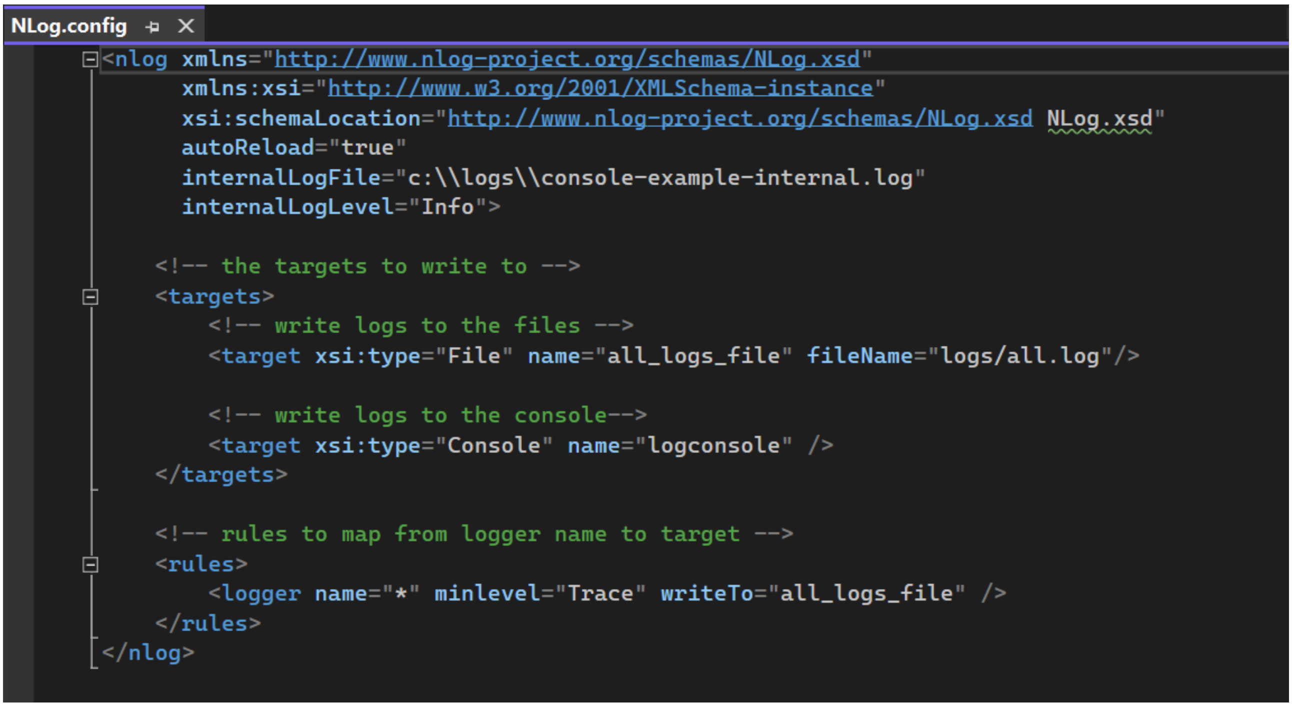
Build Action: C# analyzer additional file

Для налаштування Stylecop правил використали файл .editconfig:

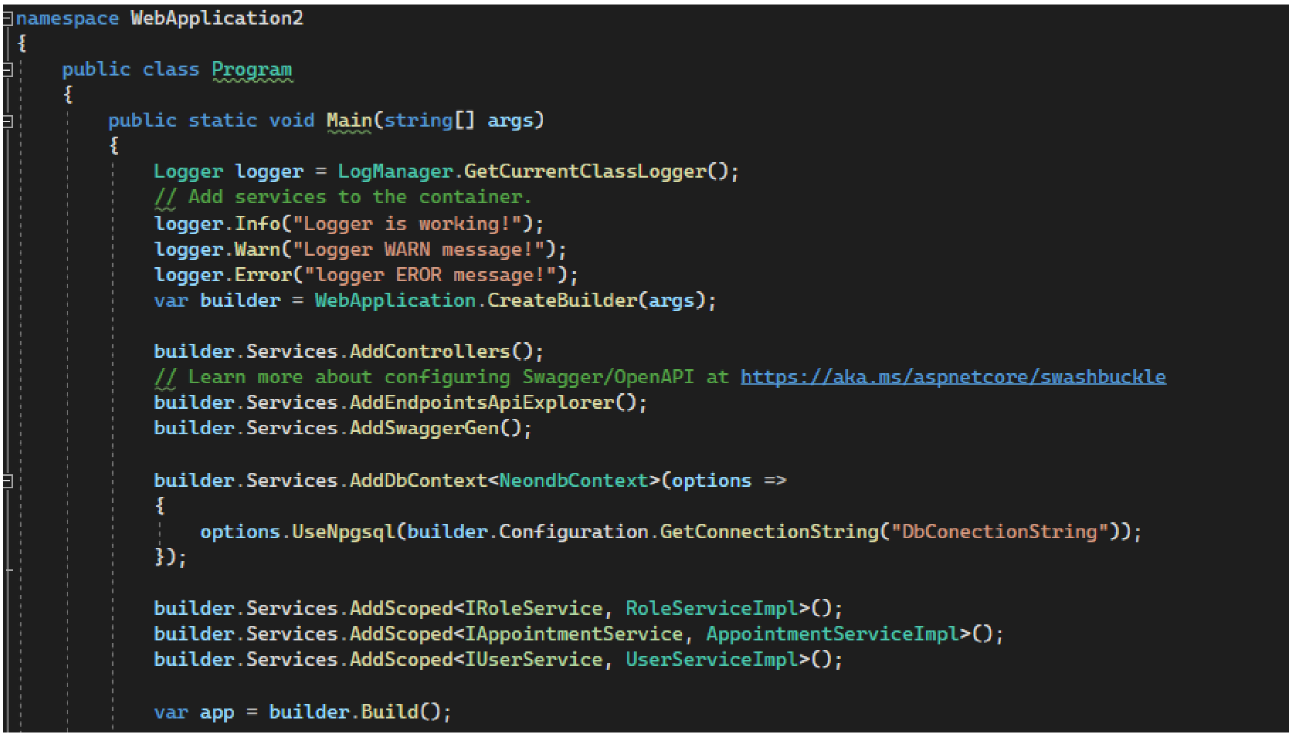


| Id | Title | Category | Severity | Location |
|---|---|---|---|---|
| Design (49) | | | | |
| ASP0000 | Do not call 'IServiceCollection.BuildServiceProvider' in 'Configure... | Design | Warning | Analyzer Defaults |
| CA1001 | Types that own disposable fields should be disposable | Design | Disabled | Analyzer Defaults |
| CA1032 | Implement standard exception constructors | Design | Warning | Analyzer Defaults |
| CA1070 | Do not declare event fields as virtual | Design | Suggestion | Analyzer Defaults |
| CA1047 | Do not declare protected member in sealed type | Design | Suggestion | Analyzer Defaults |
| CA1000 | Do not declare static members on generic types | Design | Disabled | Analyzer Defaults |
| CA1002 | Do not expose generic lists | Design | Warning | Analyzer Defaults |
| CA1003 | Use generic event handler instances | Design | Warning | Analyzer Defaults |

## NLog:



```xml
<nlog xmlns="http://www.nlog-project.org/schemas/NLog.xsd"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.nlog-project.org/schemas/NLog.xsd NLog.xsd"
      autoReload="true"
      internalLogFile="c:\\logs\\console-example-internal.log"
      internalLogLevel="Info">

    <!-- the targets to write to -->
    <targets>
        <!-- write logs to the files -->
        <target xsi:type="File" name="all_logs_file" fileName="logs/all.log"/>

        <!-- write logs to the console-->
        <target xsi:type="Console" name="logconsole" />
    </targets>

    <!-- rules to map from logger name to target -->
    <rules>
        <logger name="*" minlevel="Trace" writeTo="all_logs_file" />
    </rules>
</nlog>
```



```
namespace WebApplication2
{
    public class Program
    {
        public static void Main(string[] args)
        {
            Logger logger = LogManager.GetCurrentClassLogger();
            // Add services to the container.
            logger.Info("Logger is working!");
            logger.Warn("Logger WARN message!");
            logger.Error("logger EROR message!");
            var builder = WebApplication.CreateBuilder(args);

            builder.Services.AddControllers();
            // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
            builder.Services.AddEndpointsApiExplorer();
            builder.Services.AddSwaggerGen();

            builder.Services.AddDbContext<NeondbContext>(options =>
            {
                options.UseNpgsql(builder.Configuration.GetConnectionString("DbConectionString"));
            });

            builder.Services.AddScoped<IRoleService, RoleServiceImpl>();
            builder.Services.AddScoped<IAppointmentService, AppointmentServiceImpl>();
            builder.Services.AddScoped<IUserService, UserServiceImpl>();

            var app = builder.Build();
```