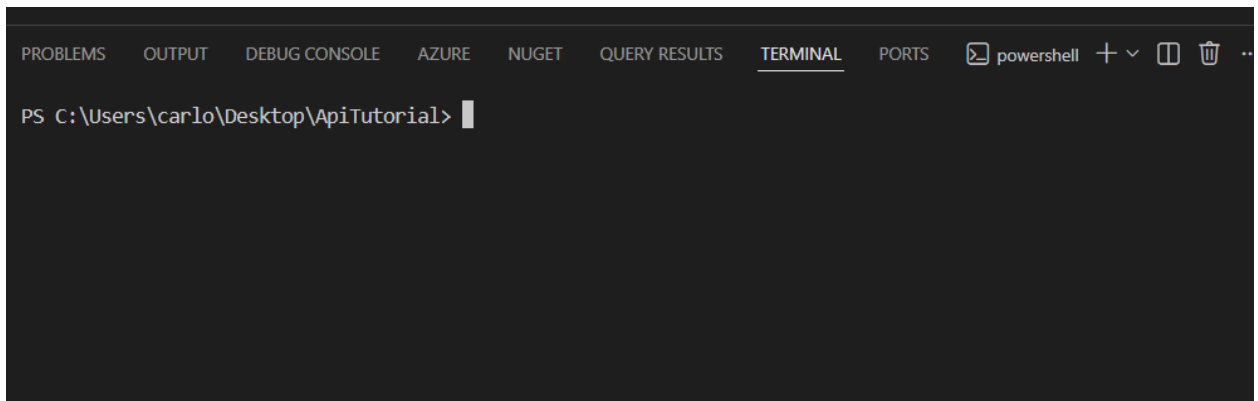
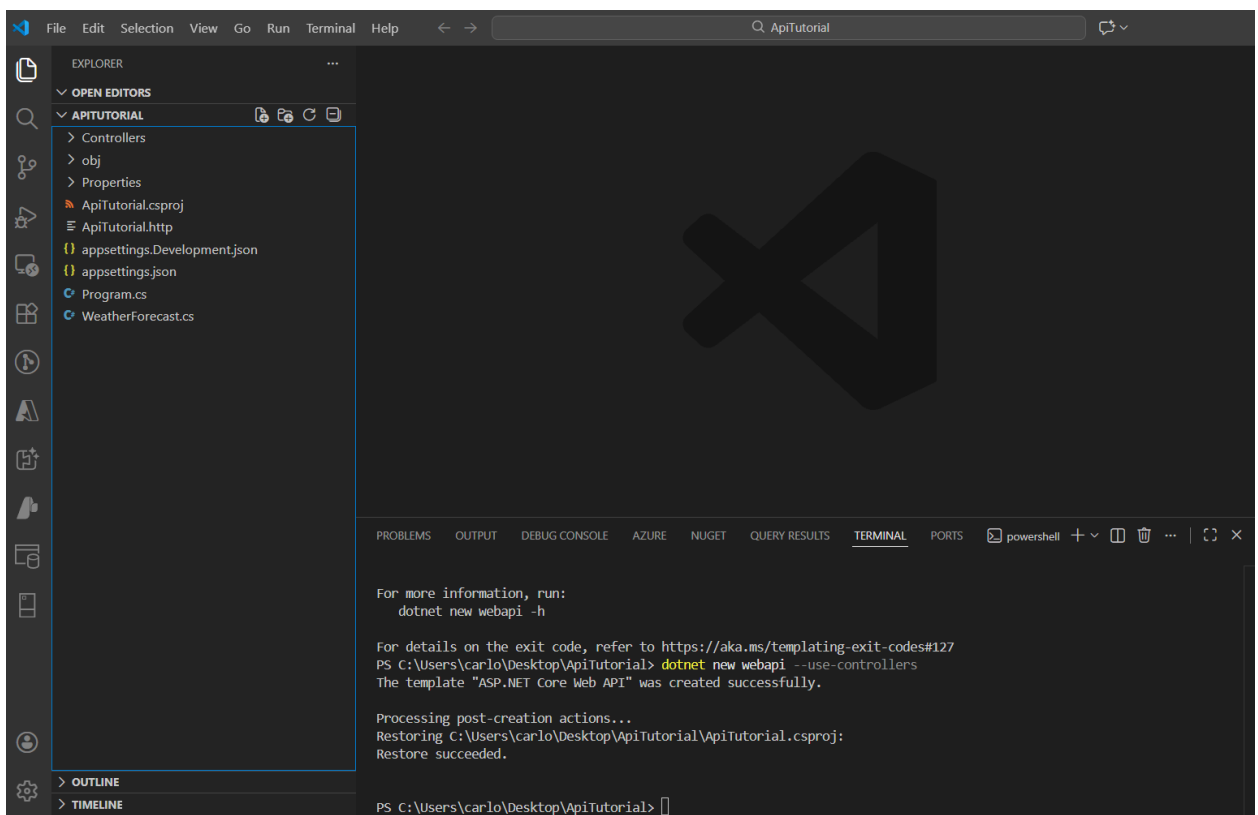


1. Create a new folder and open it with Visual Studios Code
2. Step two open the terminal with control + J or cmnd + J if on mac
3. From there make sure youre in the terminal



A screenshot of a PowerShell terminal window within Visual Studio Code. The terminal title bar shows 'powershell' and standard window controls. The command prompt displays the current directory: `PS C:\Users\carlo\Desktop\ApiTutorial>` with a cursor at the end.

4. From there enter “dotnet new webapi --use-controllers” your screen should look like below



A screenshot of the Visual Studio Code interface. The Explorer panel on the left shows the file structure of the 'APITUTORIAL' project, including folders like 'Controllers' and 'obj', and files like 'ApiTutorial.csproj', 'ApiTutorial.http', 'appsettings.Development.json', 'appsettings.json', 'Program.cs', and 'WeatherForecast.cs'. The main editor area is empty, showing the Visual Studio Code logo. The Terminal panel at the bottom shows the output of the command `dotnet new webapi --use-controllers`. The output text is as follows:

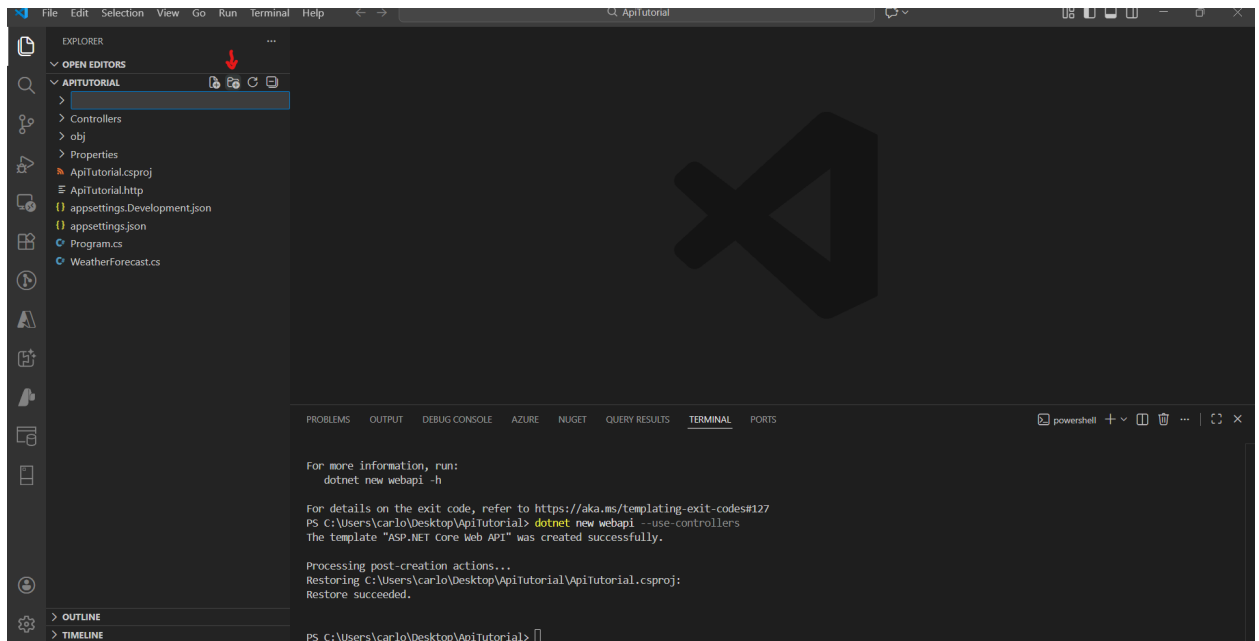
```
For more information, run:
  dotnet new webapi -h

For details on the exit code, refer to https://aka.ms/templating-exit-codes#127
PS C:\Users\carlo\Desktop\ApiTutorial> dotnet new webapi --use-controllers
The template "ASP.NET Core Web API" was created successfully.

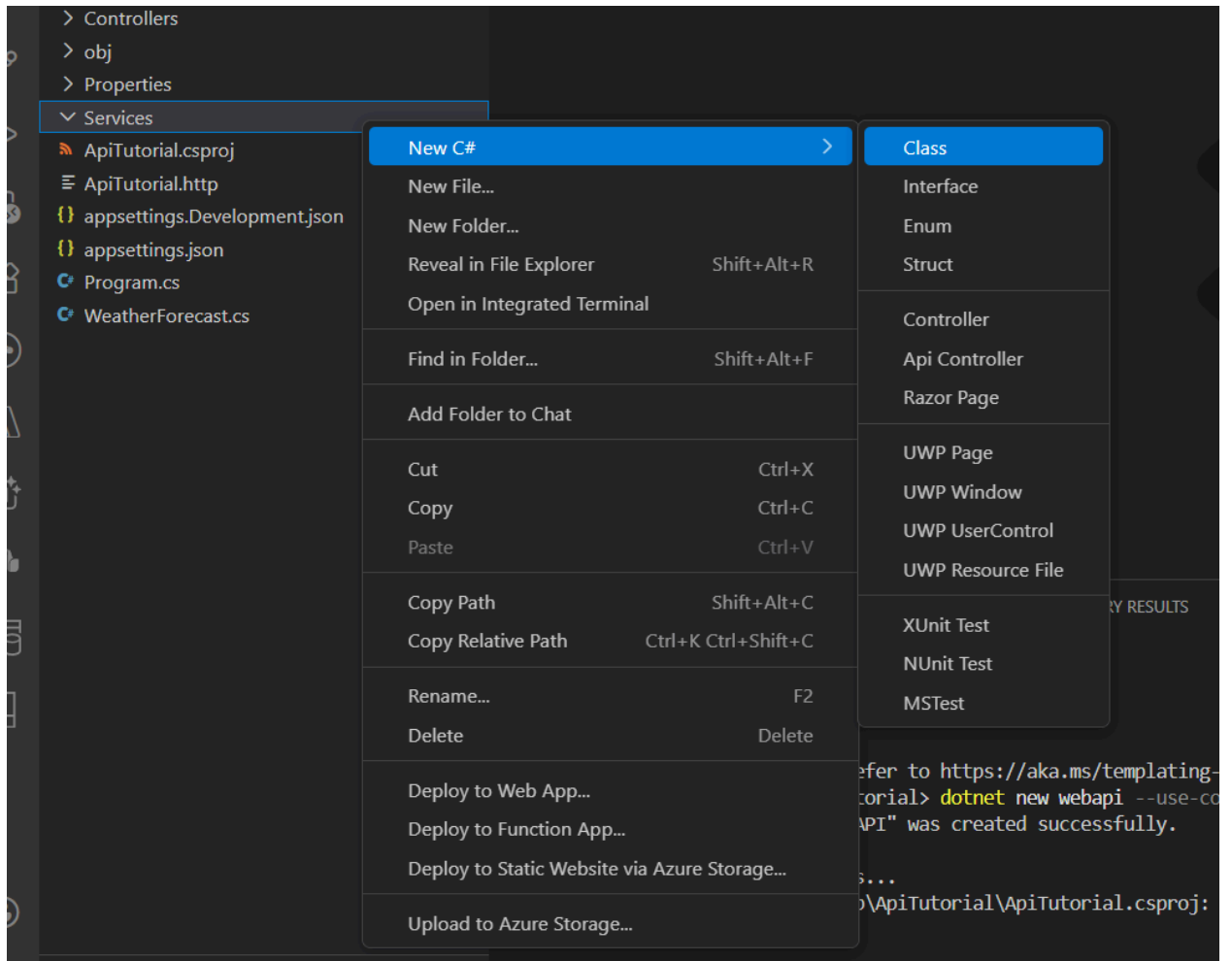
Processing post-creation actions...
Restoring C:\Users\carlo\Desktop\ApiTutorial\ApiTutorial.csproj:
Restore succeeded.

PS C:\Users\carlo\Desktop\ApiTutorial>
```

- Click the blank space below WeatherForecast then click on the new folder icon make a new folder and name it "Services"



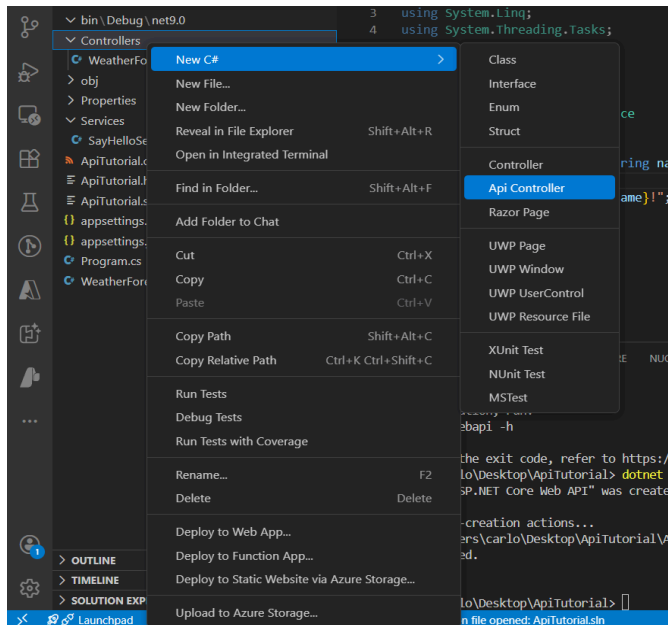
6. From there right click the Services Folder and make a new Class Name it "SayHelloService"



7. Once in the service file our logic should look like this

```
public string Hello(string name)
{
    return $"Hello, {name}!";
}
```

8. After our logic we're going to go to our controllers folder and right clicking the folder then hovering new C# and making an "Api Controller" then name it "SayHelloController"



9. In here we'll make a readonly for our variable and set the value

```
10.     private readonly SayHelloService _sayHelloService;
11. public SayHelloController(SayHelloService sayHelloService)
12.     {
13.         _sayHelloService = sayHelloService;
14.     }
```

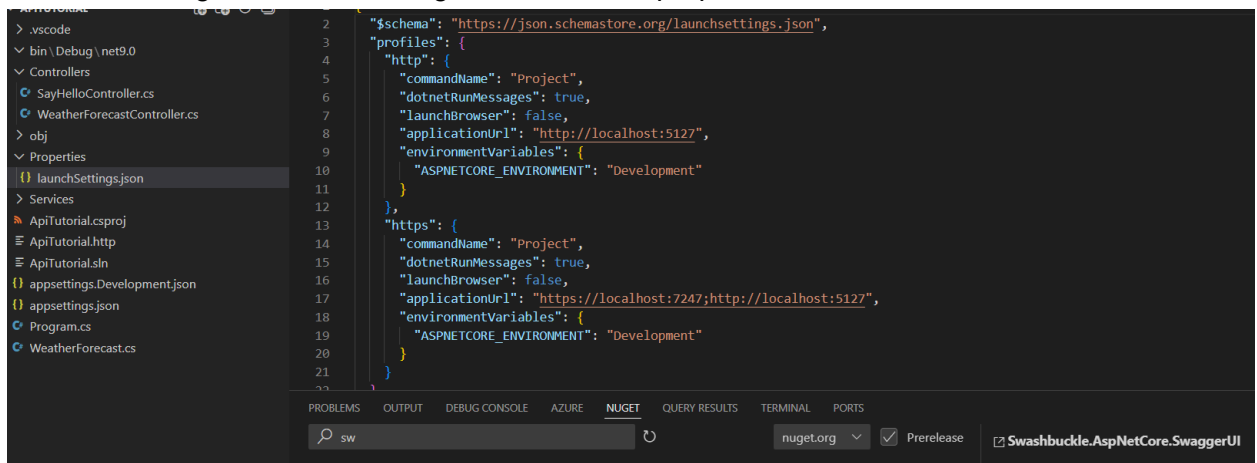
15. Then our get and calling our function

```
[HttpGet("{name}")]
public ActionResult<string> Get(string name)
{
    return _sayHelloService.Hello(name);
}
```

16. Our logic and routing is now done in order to see on swagger we need to install a few thing open the terminal and paste this in

```
dotnet add package Swashbuckle.AspNetCore.SwaggerGen --version 9.0.0
dotnet add package Swashbuckle.AspNetCore.Swagger --version 9.0.0
dotnet add package Swashbuckle.AspNetCore.SwaggerUI --version 9.0.0
dotnet add package Swashbuckle.AspNetCore --version 9.0.0
```

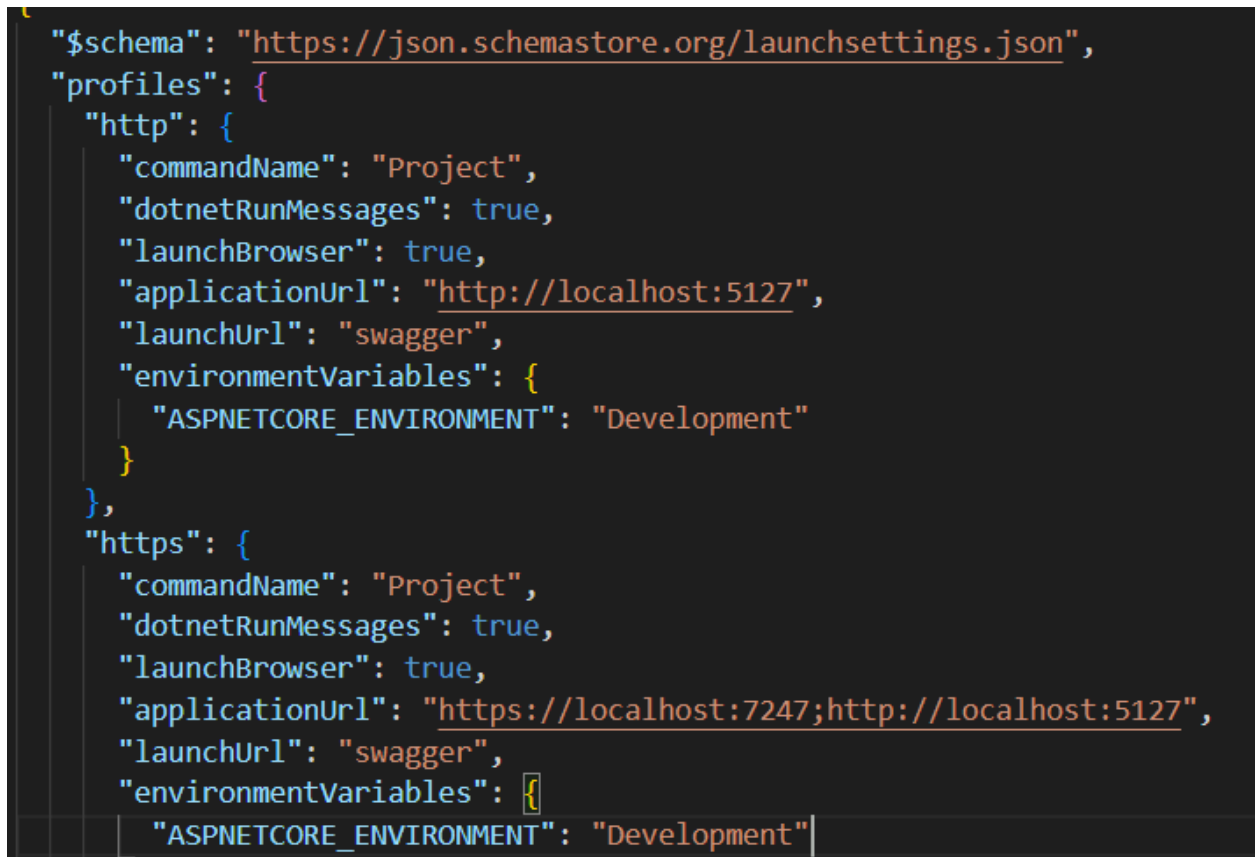
17. For there we'll go to launch settings located in our properties



The screenshot shows the Visual Studio Code interface. On the left, the 'Properties' folder is expanded, showing 'launchSettings.json'. The main editor displays the content of this file, which is a JSON configuration for launch settings. It defines two profiles: 'http' and 'https'. The 'http' profile has 'commandName': 'Project', 'dotnetRunMessages': true, 'launchBrowser': false, 'applicationUrl': 'http://localhost:5127', and 'environmentVariables' with 'ASPNETCORE_ENVIRONMENT': 'Development'. The 'https' profile has 'commandName': 'Project', 'dotnetRunMessages': true, 'launchBrowser': false, 'applicationUrl': 'https://localhost:7247;http://localhost:5127', and 'environmentVariables' with 'ASPNETCORE_ENVIRONMENT': 'Development'.

```
2  "$schema": "https://json.schemastore.org/launchsettings.json",
3  "profiles": {
4    "http": {
5      "commandName": "Project",
6      "dotnetRunMessages": true,
7      "launchBrowser": false,
8      "applicationUrl": "http://localhost:5127",
9      "environmentVariables": {
10       "ASPNETCORE_ENVIRONMENT": "Development"
11     }
12   },
13   "https": {
14     "commandName": "Project",
15     "dotnetRunMessages": true,
16     "launchBrowser": false,
17     "applicationUrl": "https://localhost:7247;http://localhost:5127",
18     "environmentVariables": {
19       "ASPNETCORE_ENVIRONMENT": "Development"
20     }
21   }
22 }
```

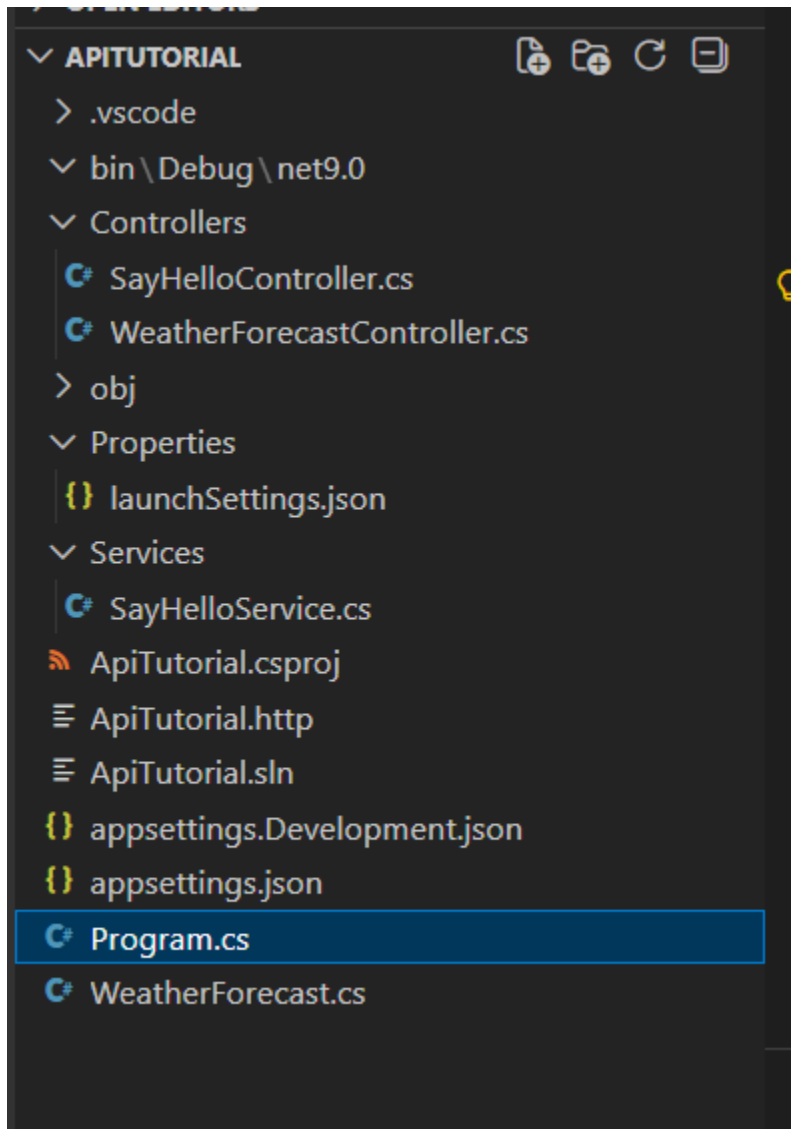
18. Under applicationUrl we're going to add "launchUrl": "swagger" for instances



The screenshot shows the Visual Studio Code interface with the 'launchSettings.json' file open. The 'http' profile's 'applicationUrl' is now 'http://localhost:5127', and the 'https' profile's 'applicationUrl' is 'https://localhost:7247;http://localhost:5127'. Both profiles now have a 'launchUrl' property set to 'swagger'. The 'environmentVariables' section remains unchanged.

```
"$schema": "https://json.schemastore.org/launchsettings.json",
"profiles": {
  "http": {
    "commandName": "Project",
    "dotnetRunMessages": true,
    "launchBrowser": true,
    "applicationUrl": "http://localhost:5127",
    "launchUrl": "swagger",
    "environmentVariables": {
      "ASPNETCORE_ENVIRONMENT": "Development"
    }
  },
  "https": {
    "commandName": "Project",
    "dotnetRunMessages": true,
    "launchBrowser": true,
    "applicationUrl": "https://localhost:7247;http://localhost:5127",
    "launchUrl": "swagger",
    "environmentVariables": {
      "ASPNETCORE_ENVIRONMENT": "Development"
    }
  }
}
```

19. From there we are going to add a couple things to our [program.cs](#)



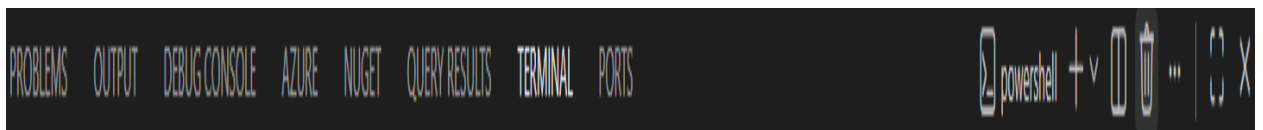
20. In here you're going to add your service

```
builder.Services.AddScoped<ApiTutorial.Services.SayHelloService>();
```

21. And your `app.UseSwagger();` and `app.UseSwaggerUI();` make sure they're in the same spot as mine

```
Program.cs
1  var builder = WebApplication.CreateBuilder(args);
2
3  // Add services to the container.
4
5  builder.Services.AddControllers();
6  builder.Services.AddScoped<ApiTutorial.Services.SayHelloService>();
7  // Learn more about configuring OpenAPI at https://aka.ms/aspnet/openapi
8  builder.Services.AddOpenApi();
9
10 builder.Services.AddEndpointsApiExplorer();
11 builder.Services.AddSwaggerGen();
12
13 var app = builder.Build();
14
15 // Configure the HTTP request pipeline.
16 if (app.Environment.IsDevelopment())
17 {
18     app.UseSwagger();
19     app.UseSwaggerUI();
20 }
21
22 app.UseHttpsRedirection();
23
24 app.UseAuthorization();
25
26 app.MapControllers();
27
28 app.Run();
29
```

22. From there you're going to open the terminal and click the trash icon



23. Then run the terminal again with `ctrl j` or `cmd j` then type in `dotnet watch run`