**NSwag vs Swashbuckle**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**What is Swagger:**

1. **Swagger API is more familiar to everyone than Open API because the Swagger API is came first than OPEN API.**
2. **Swagger API is an open source to Address every common issue with REST full services**
3. **Before Swagger API there is no specific standard way to work with REST full services to maintain their documentation.**
4. **Swagger API is initiated in 2011**
5. **Open API is initiated in 2015.**
6. **In 2016, Swagger API 2.0 is renamed to OpenAPI2.0**
7. **In 2017, Open API3.0 is updated with more industry standard formats to describe how to work with REST API's.**

This document is only considering .NET API’s and preparing yaml and swagger UI

**NSWAG:**

<https://github.com/RicoSuter/NSwag>

NSwag is a Swagger/OpenAPI 2.0 and 3.0 toolchain for .NET, .NET Core, Web API, ASP.NET Core. The NSwag project provides tools to generate OpenAPI specifications from existing ASP.NET Web API controllers and client code from these OpenAPI specifications.

The project combines the functionality of Swashbuckle (OpenAPI/Swagger generation) and AutoRest (client generation) in one toolchain (these two libs are not needed). This way a lot of incompatibilites can be avoided and features which are not well described by the OpenAPI specification or JSON Schema are better supported (e.g. inheritance, enum and reference handling).

The NSwag project heavily uses **NJsonSchema** for .NET for JSON Schema handling and C#/TypeScript class/interface generation.

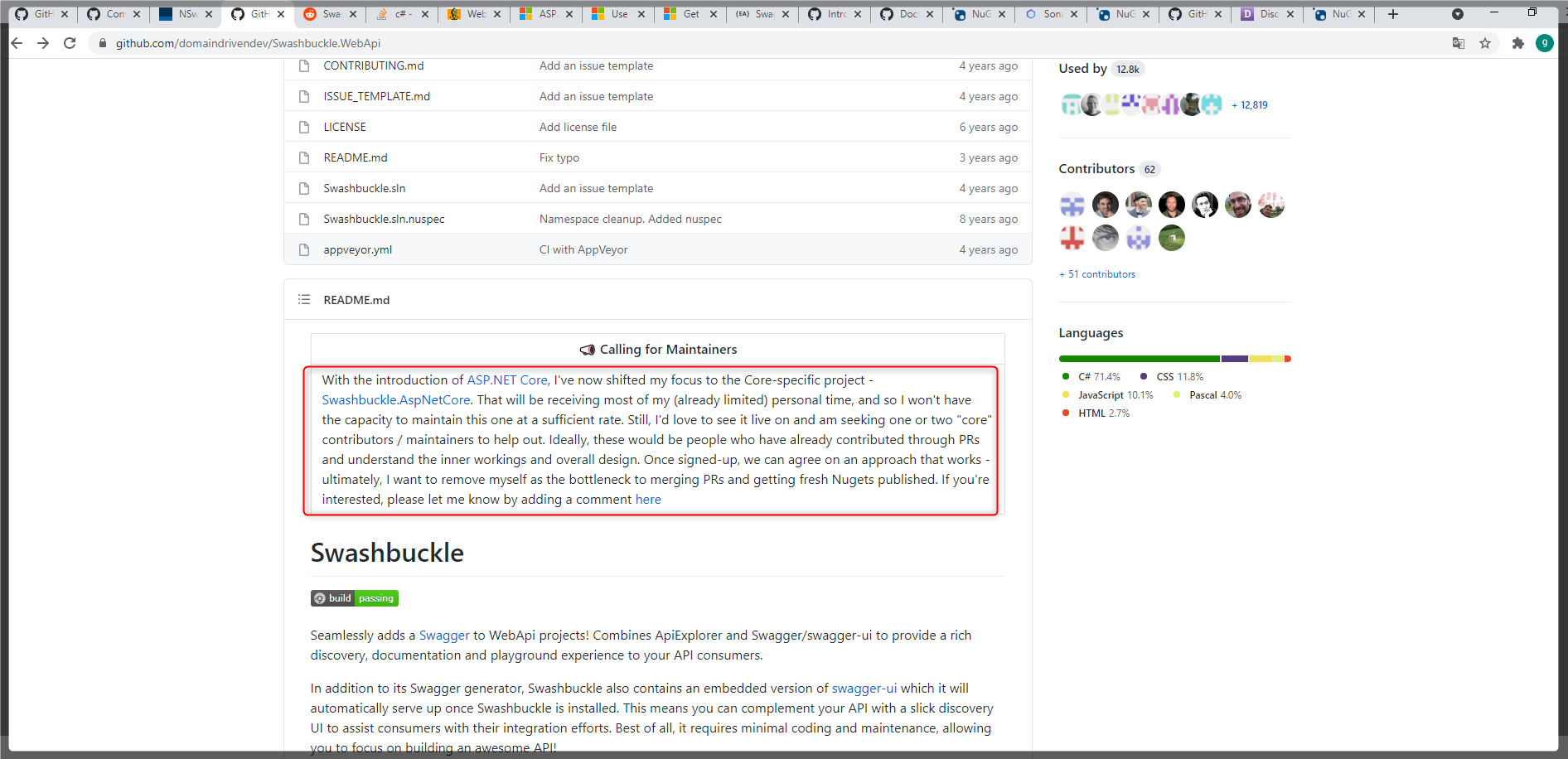
**Swashbuckle:**

<https://github.com/domaindrivendev/Swashbuckle.WebApi>

Seamlessly adds a [Swagger](http://swagger.io/) to WebApi projects! Combines ApiExplorer and Swagger/swagger-ui to provide a rich discovery, documentation and playground experience to your API consumers.

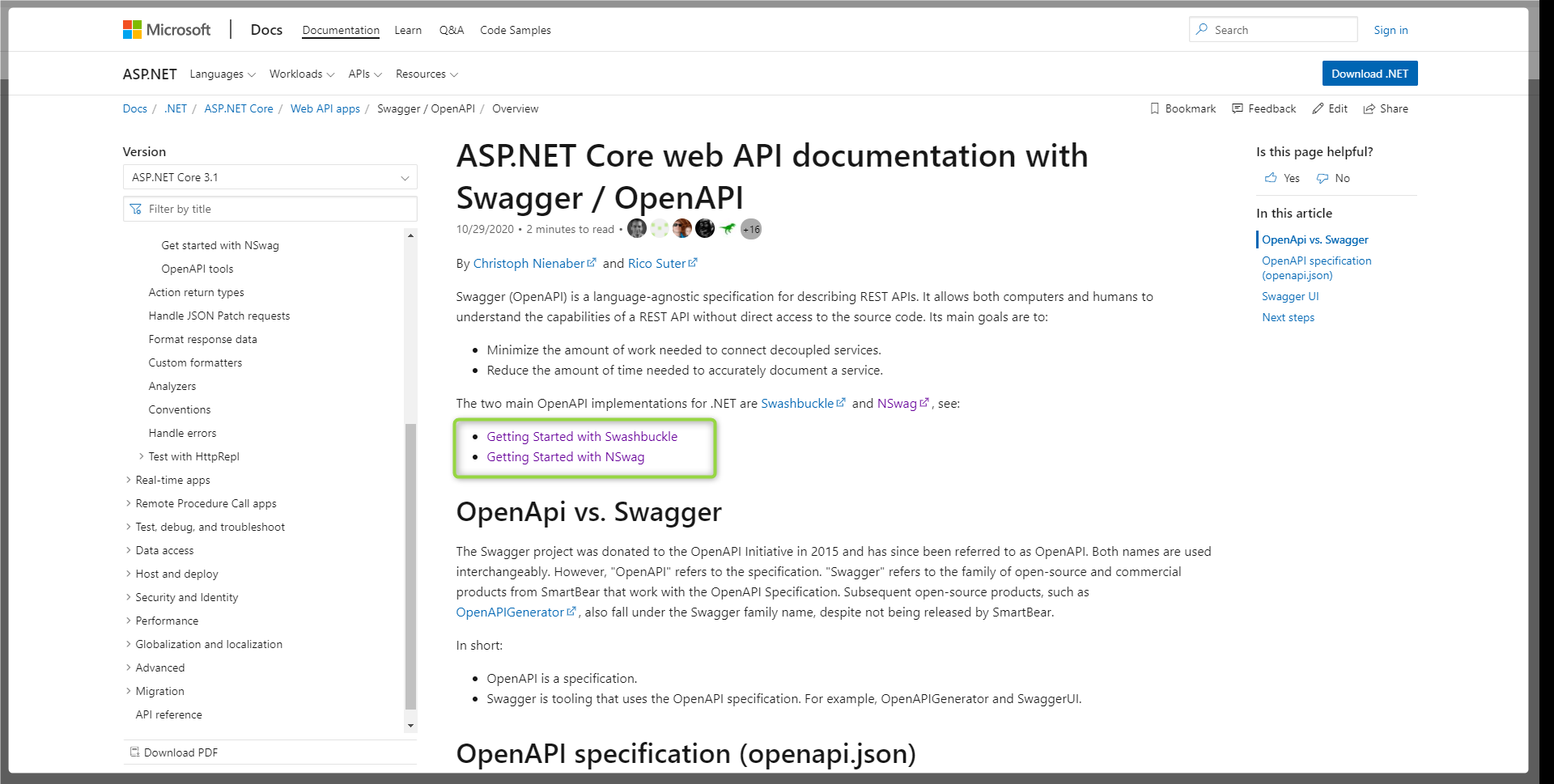
In addition to its Swagger generator, Swashbuckle also contains an embedded version of [swagger-ui](https://github.com/swagger-api/swagger-ui) which it will automatically serve up once Swashbuckle is installed. This means you can complement your API with a slick discovery UI to assist consumers with their integration efforts. Best of all, it requires minimal coding and maintenance, allowing you to focus on building an awesome API!

**Officially said there will be no maintenance in future**

****

**Recommends:** Microsoft Recommends both

<https://docs.microsoft.com/en-us/aspnet/core/tutorials/web-api-help-pages-using-swagger?view=aspnetcore-3.1>



**Remarks of Swashbuckle over NSwag.**

**Ref:** [**https://www.codeproject.com/Articles/5259717/WebApiClientGen-vs-Swashbuckle-plus-NSwag**](https://www.codeproject.com/Articles/5259717/WebApiClientGen-vs-Swashbuckle-plus-NSwag) **published on** 8 May 2020

* Swashbuckle translates server side struct System.Drawing.Point to client-side class Point.
* Open API and NSwag supports inheritance, however Swashbuckle's support for inheritance is poor, as of Swashbuckle.AspNetCore 5.0.
* Open API and NSwag provide limited supports for enum, however, Swashbuckle supports even less.
* NSwag does support namespace and enum, however, not working well with the Swagger definition file generated by Swashbuckle.AspNet Core 5.0.

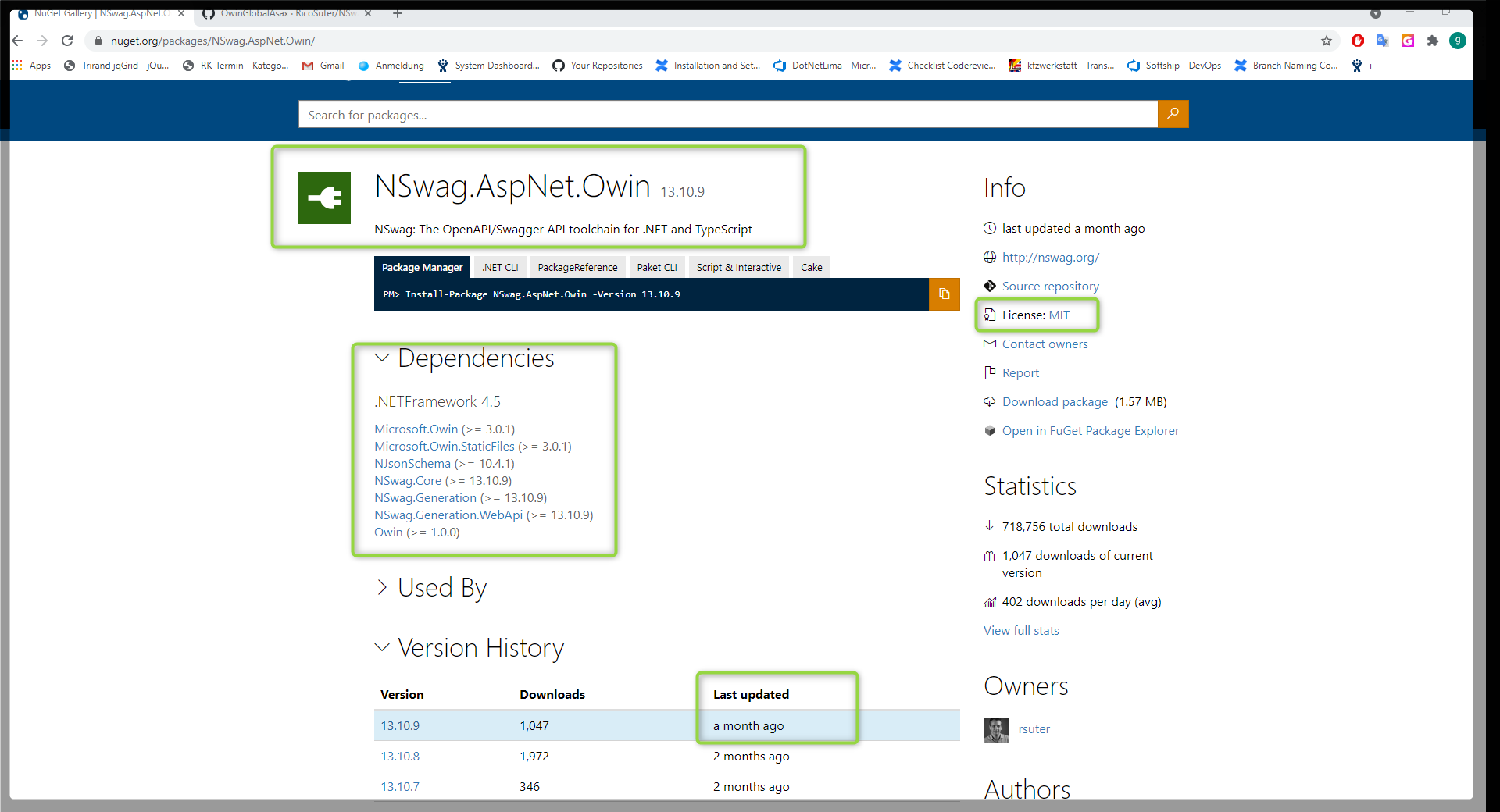
**NOTE:** My understanding NSwag is good when we are generating Typescript code in client side using .NET libraries.

**Licenses:**

1. Swashbuckle has BSD-3-Clause license.
2. NSwag has MIT license.

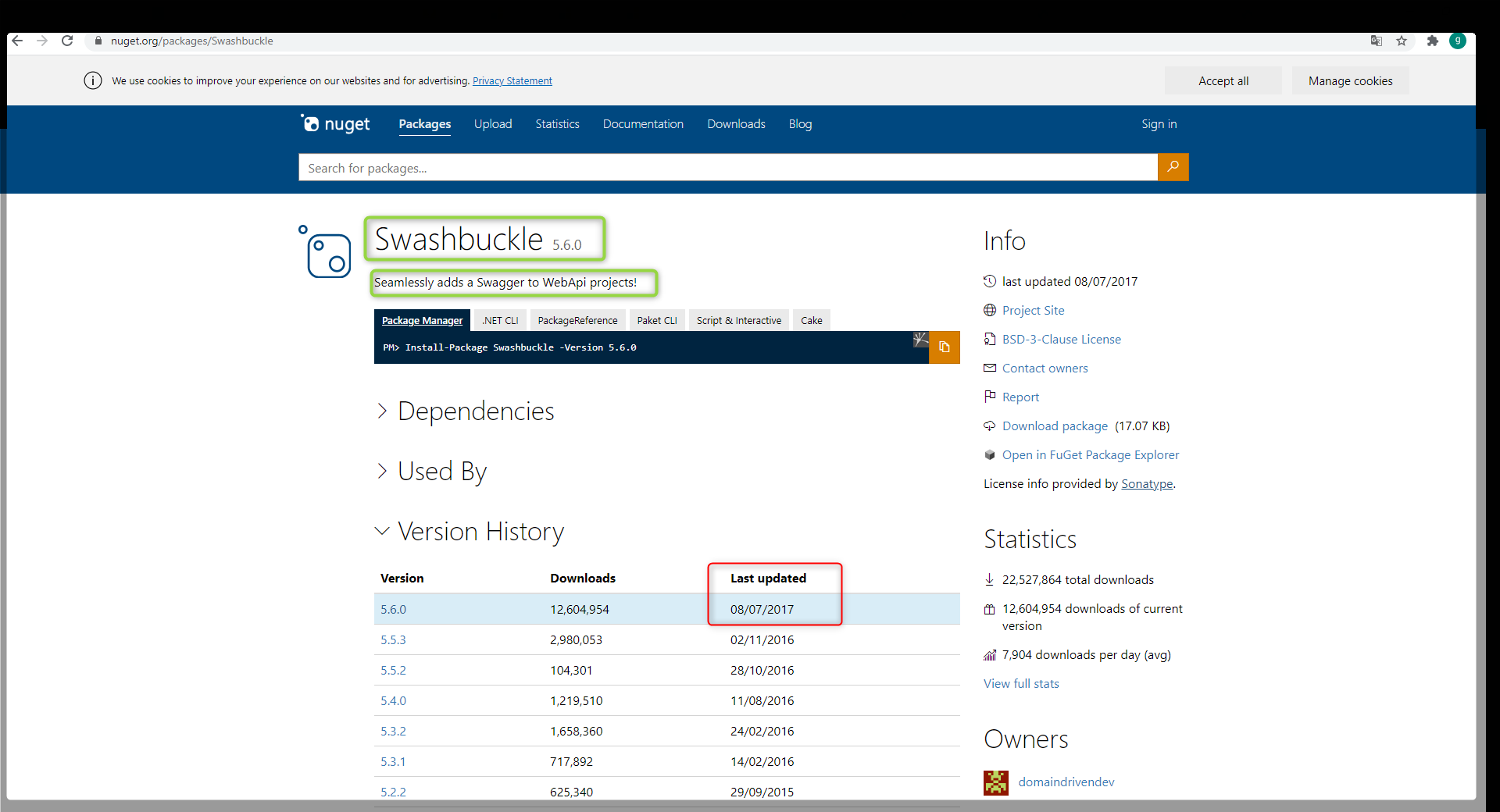
[**https://www.nuget.org/packages/NSwag.Core/**](https://www.nuget.org/packages/NSwag.Core/)

Nswag has very frequent updates



[**https://www.nuget.org/packages/Swashbuckle**](https://www.nuget.org/packages/Swashbuckle)

No Updates since few years especially on .net Swashbuckle

****

**NSwag Integration to .NET Application.**

**Note: Controllers must not be inherited from MVC**

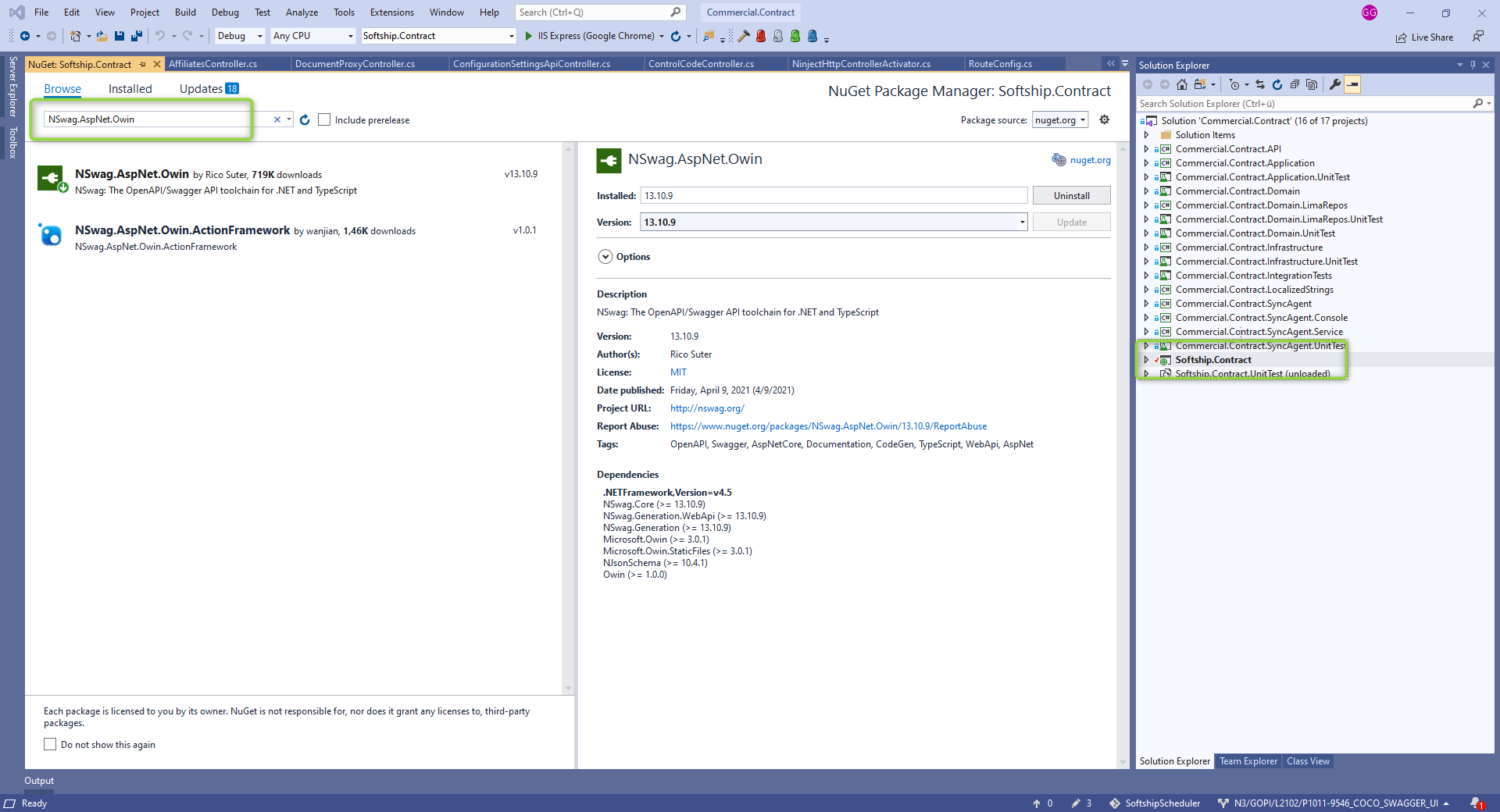
[**https://github.com/RicoSuter/NSwag/blob/54d85d3f2ce254c31b61f56abcbeb4560c05e4b0/src/NSwag.Generation.WebApi/WebApiOpenApiDocumentGenerator.cs#L36**](https://github.com/RicoSuter/NSwag/blob/54d85d3f2ce254c31b61f56abcbeb4560c05e4b0/src/NSwag.Generation.WebApi/WebApiOpenApiDocumentGenerator.cs#L36)

****

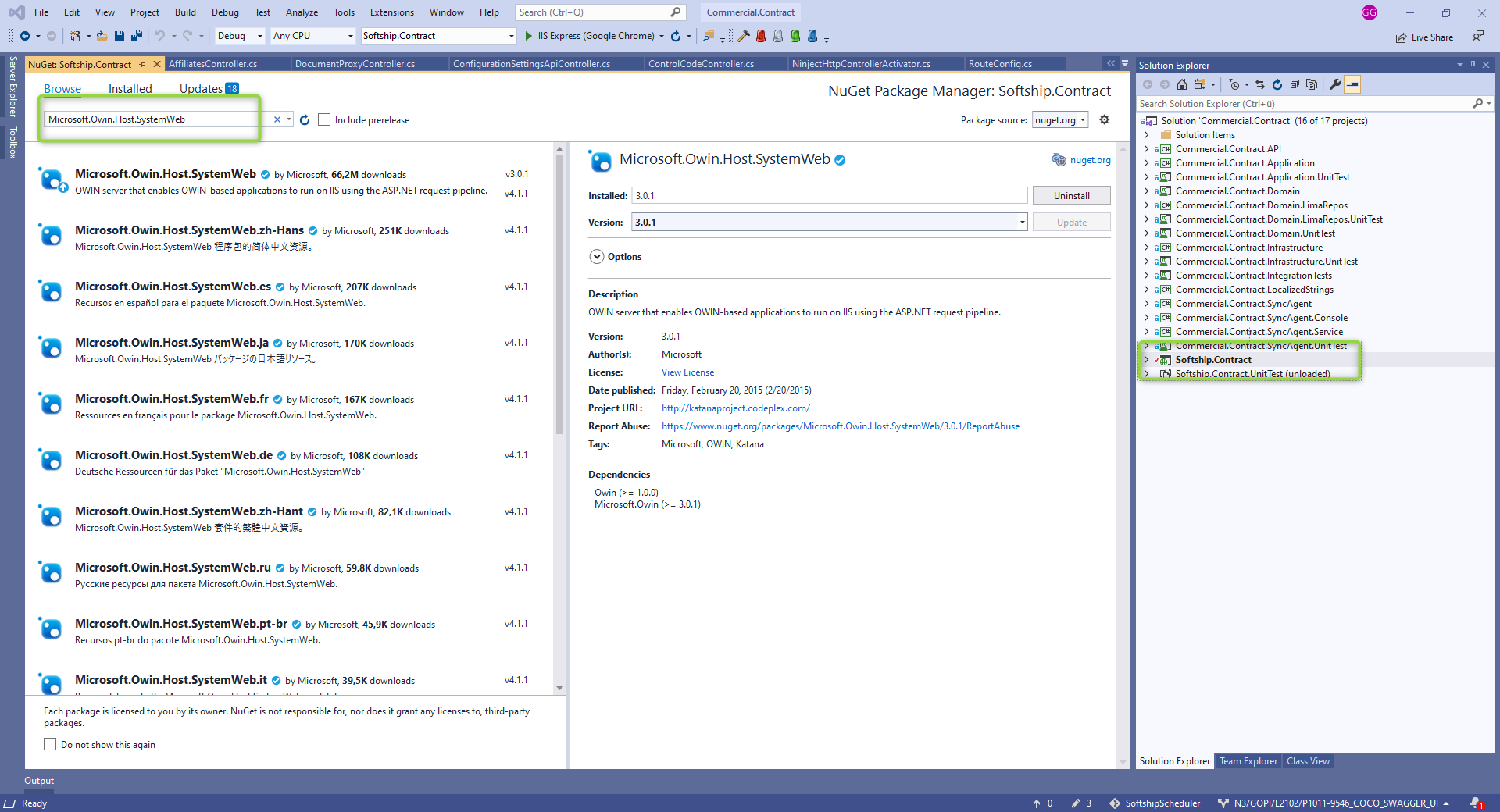
Ref: <https://github.com/RicoSuter/NSwag/wiki/OwinGlobalAsax#integration>

**Step 1:**  Install NuGet packages

* NSwag.AspNet.Owin



* Microsoft.Owin.Host.SystemWeb



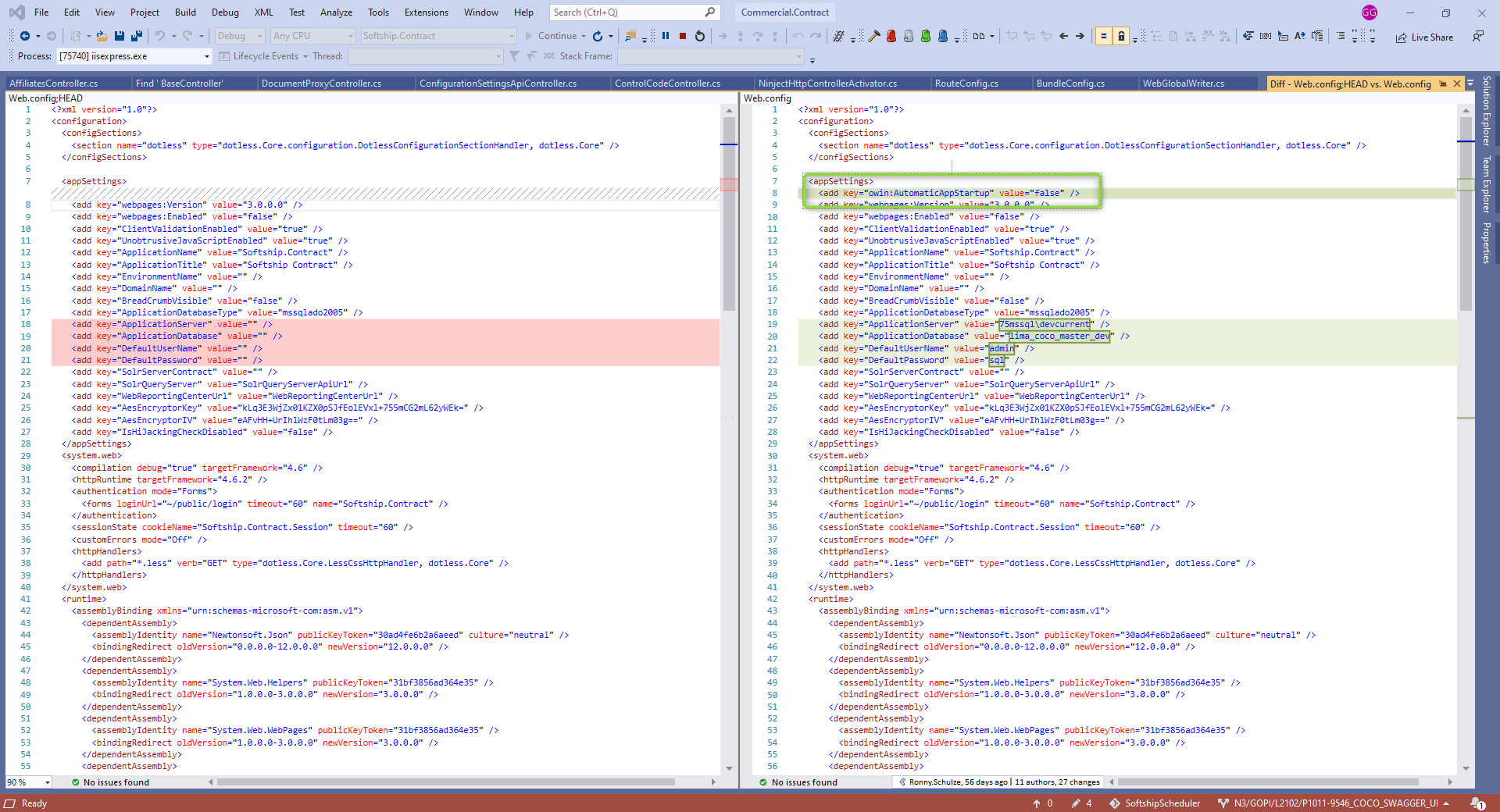
**Step 2:**  Edit web.config

* 1. open your Web.config and add the following app setting:

<appSettings>

<add key="owin:AutomaticAppStartup" value="false" />

</appSettings>



* 1. **Pipe all request to the .NET pipeline**

In the system.webServer tag, set runAllManagedModulesForAllRequests to true so that all requests are piped to ASP.NET:

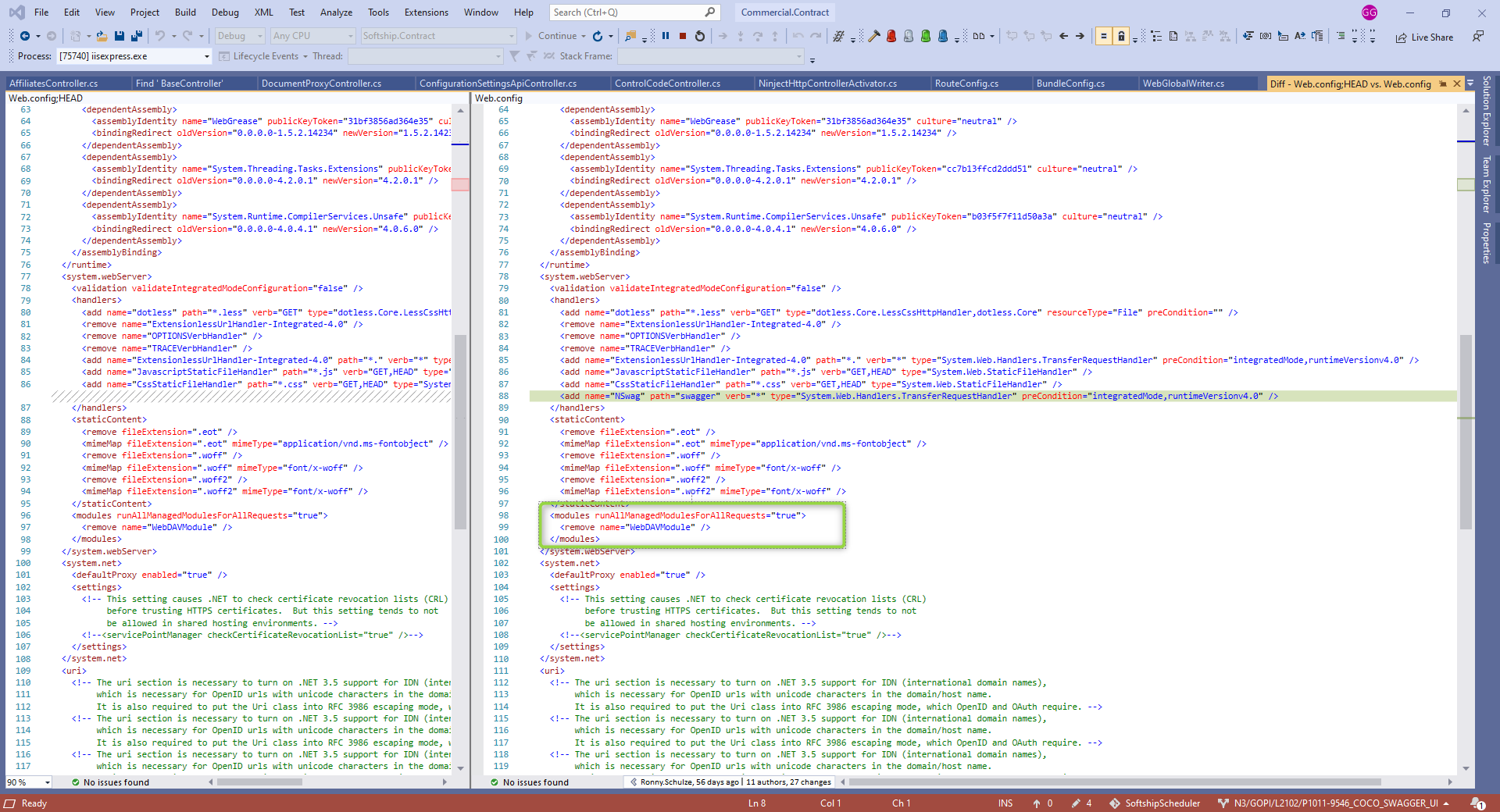
<system.webServer>

...

<modules runAllManagedModulesForAllRequests="true" />

...

</system.webServer>



* 1. **Pipe only the Swagger request to the specific middlewares**

Important: The routes defined in the web.config and the UseSwagger/UseSwaggerUi methods must be the same:

<system.webServer>

...

<handlers>

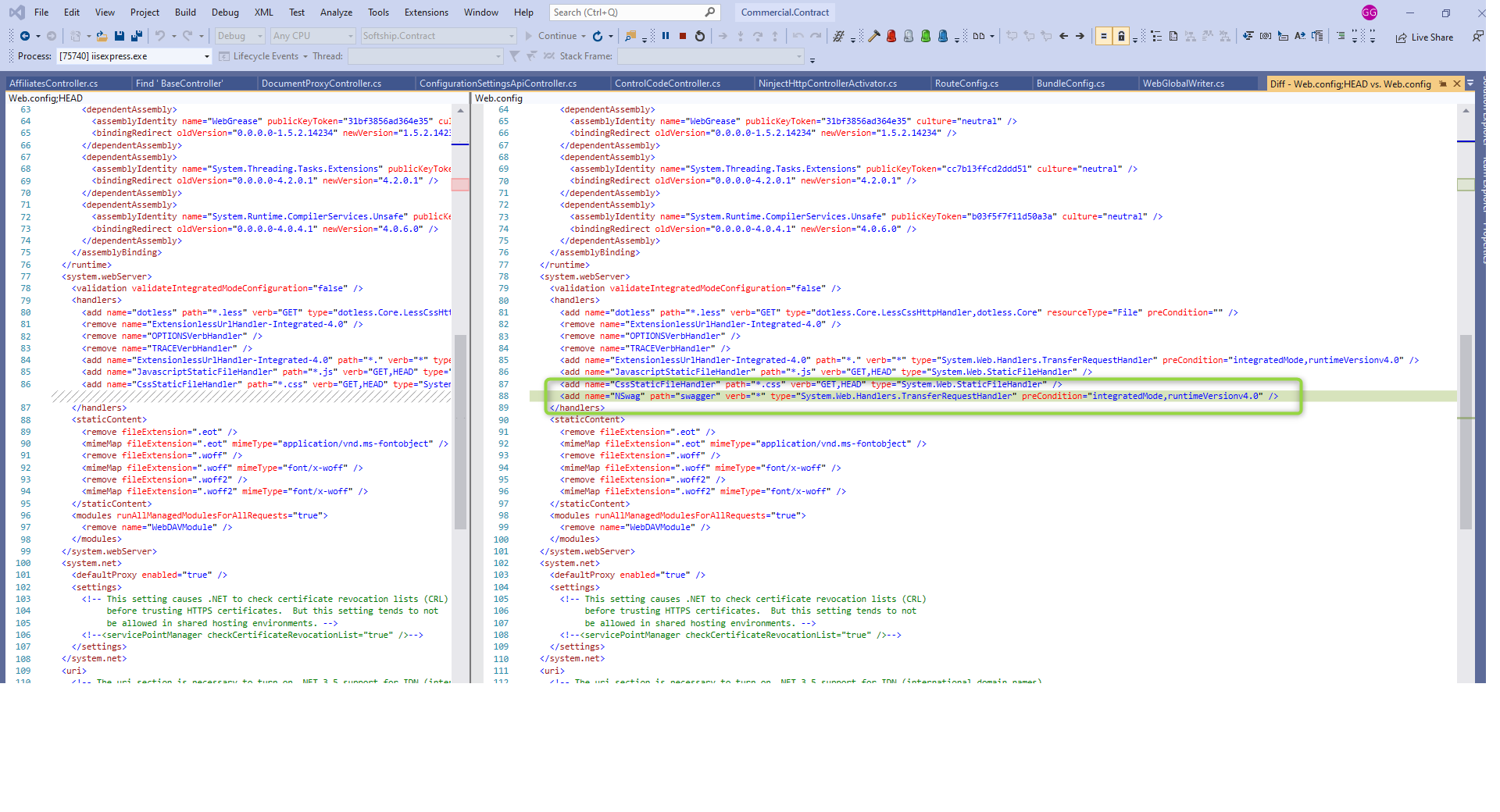
...

<add name="NSwag" path="swagger" verb="\*" type="System.Web.Handlers.TransferRequestHandler" preCondition="integratedMode,runtimeVersionv4.0" />

...

...

</system.webServer>



**Step 3:**  Edit Global.asax.cs

Now, open the Global.asax.cs and add the following call at the beginning of the Application\_Start method:

RouteTable.Routes.MapOwinPath("swagger", app =>

{

app.UseSwaggerUi3(typeof(WebApiApplication).Assembly, settings =>

{

settings.MiddlewareBasePath = "/swagger";

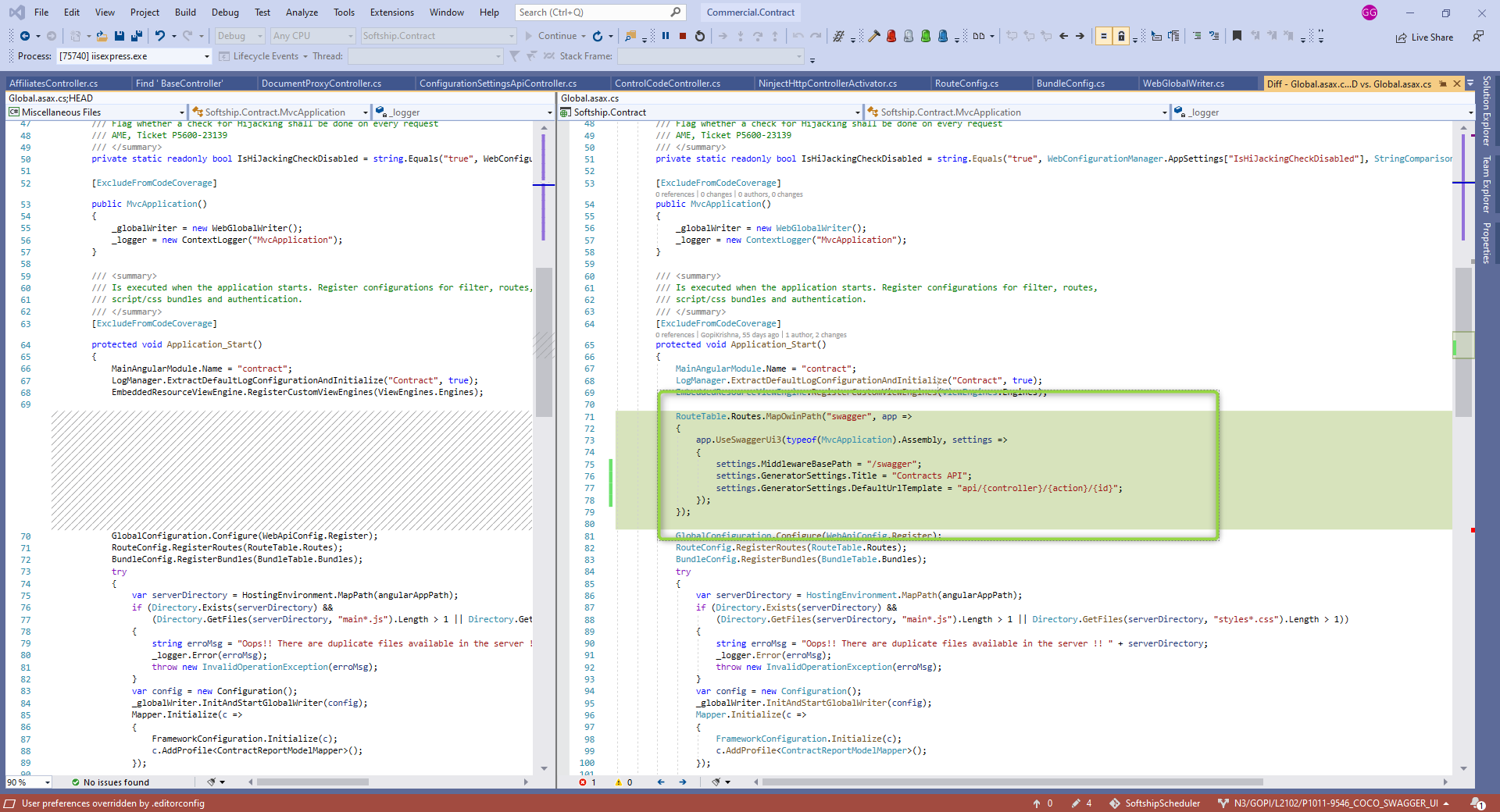
//settings.GeneratorSettings.DefaultUrlTemplate = "api/{controller}/{id}"; //this is the default one

settings.GeneratorSettings.DefaultUrlTemplate = "api/{controller}/{action}/{id}";

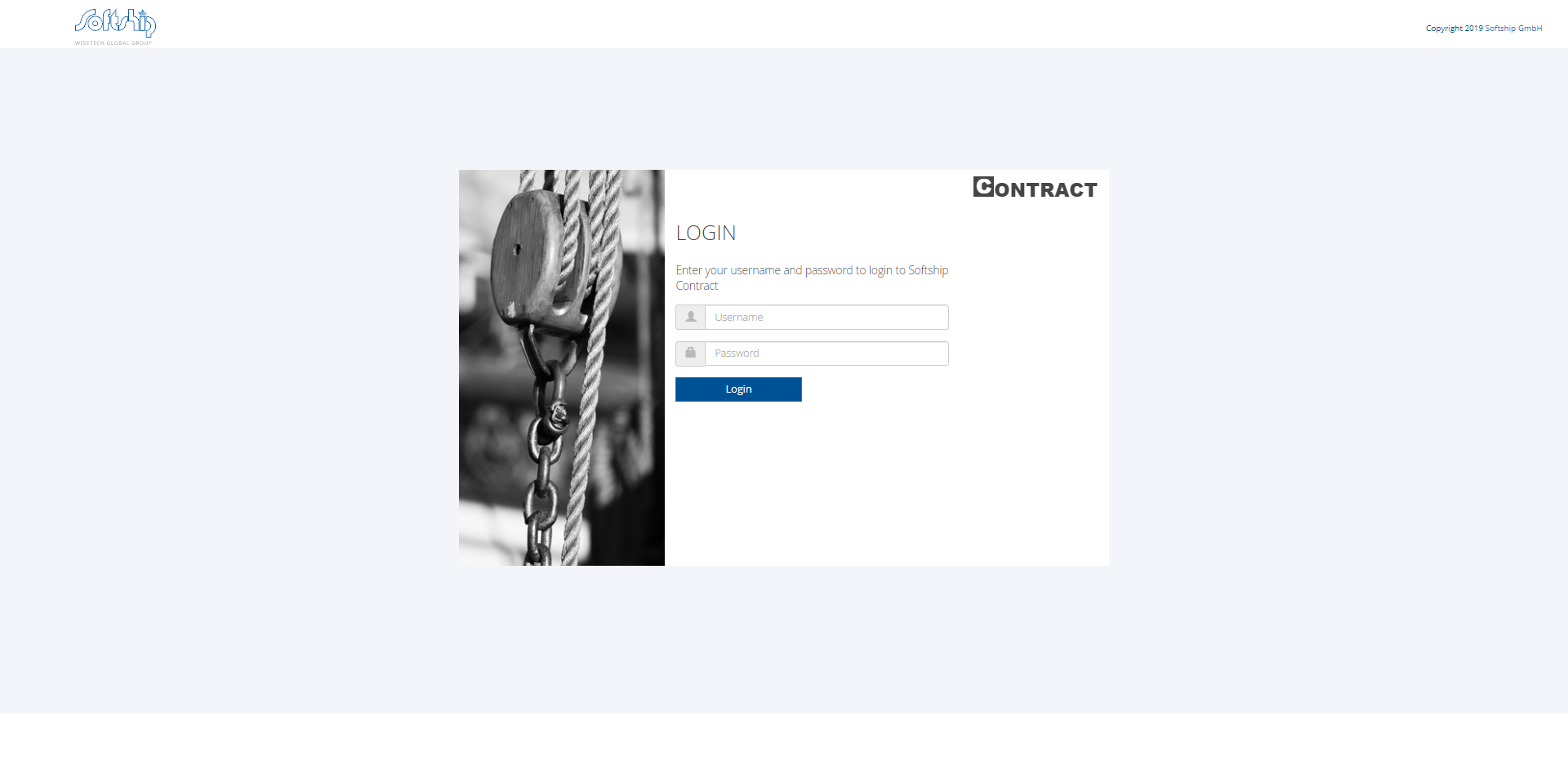
});

});

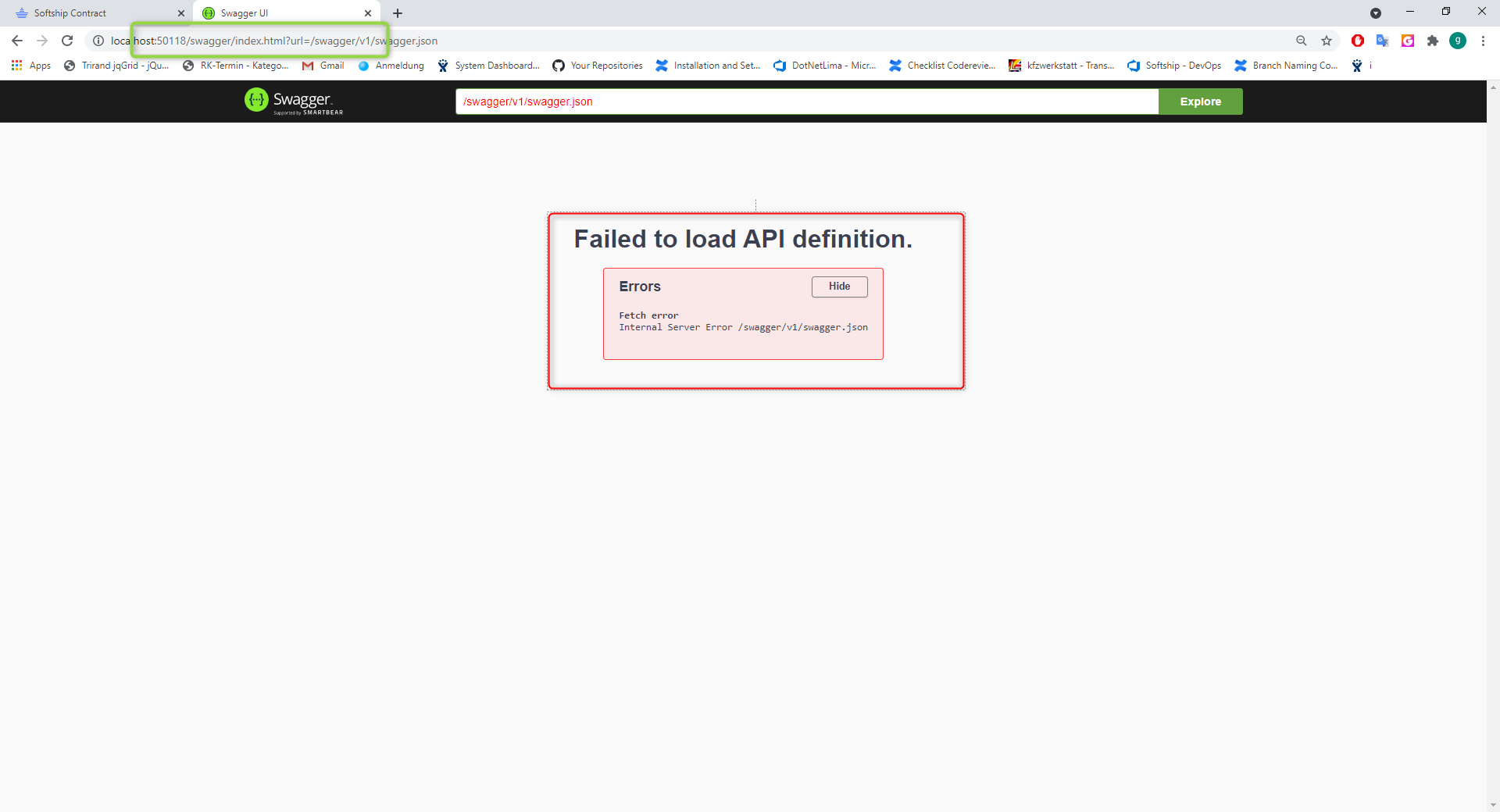
...



Run the application

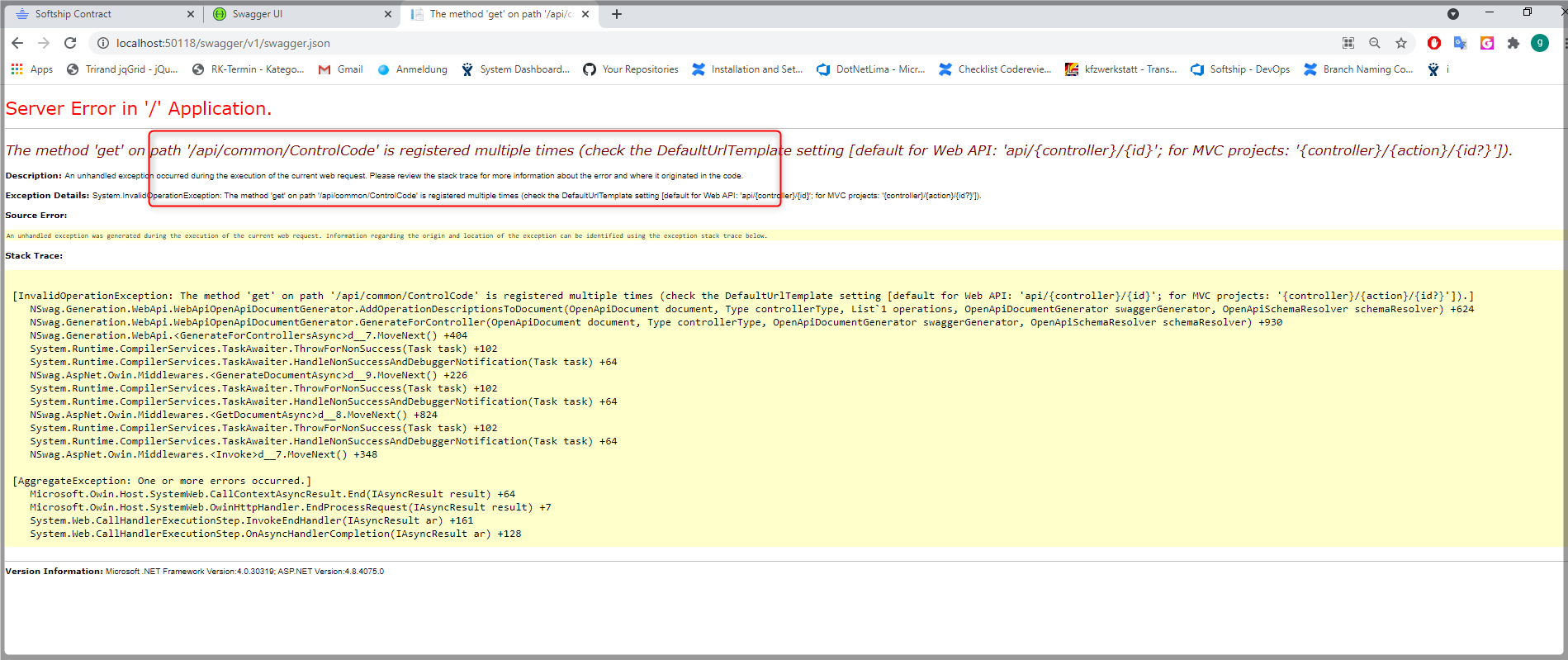


Open another tab and enter swagger url => <http://localhost:50118/swagger/>

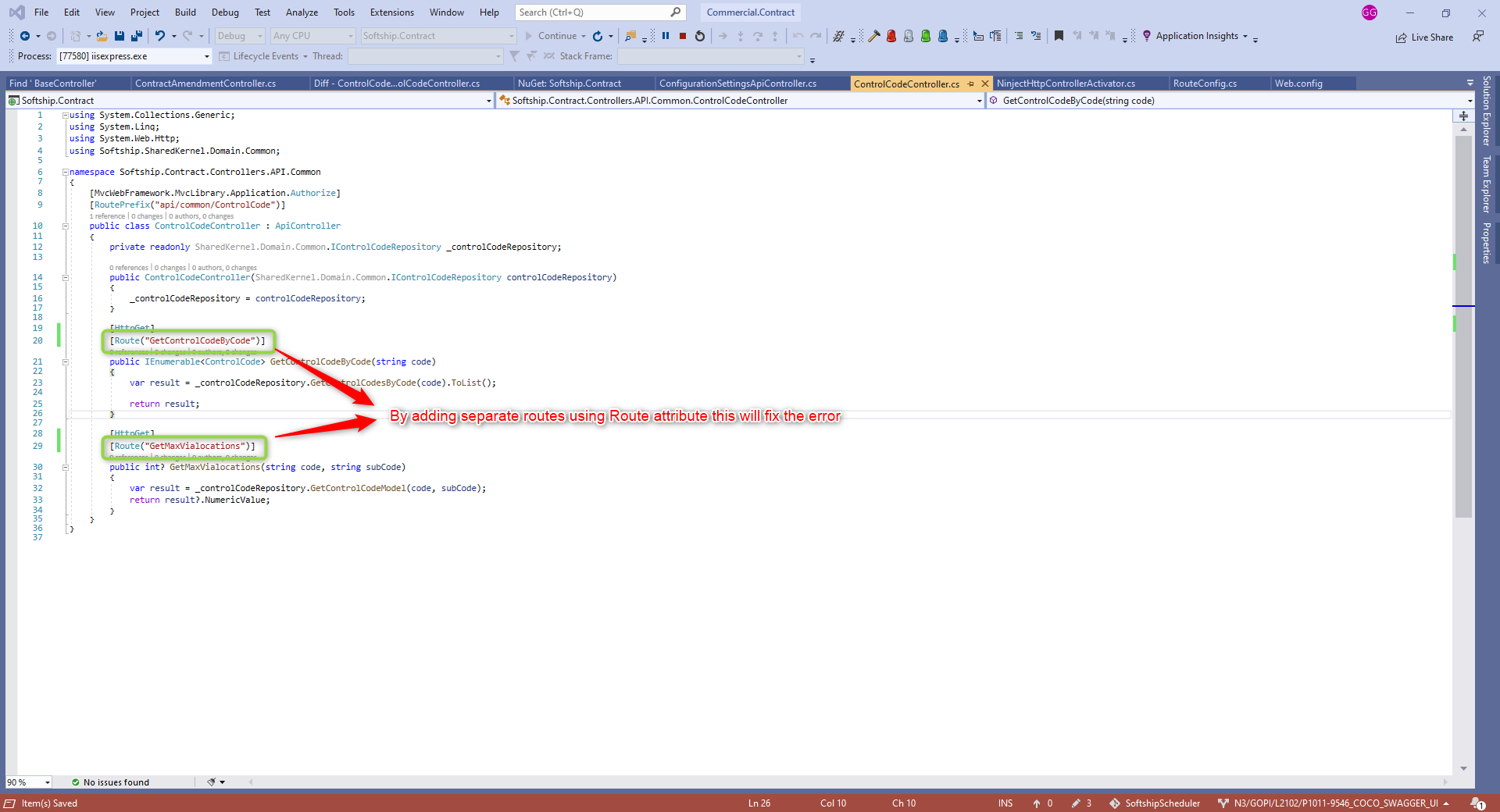


To find the exact error open another tab and enter url (i.e swagger.json) =>

<http://localhost:50118/swagger/v1/swagger.json>

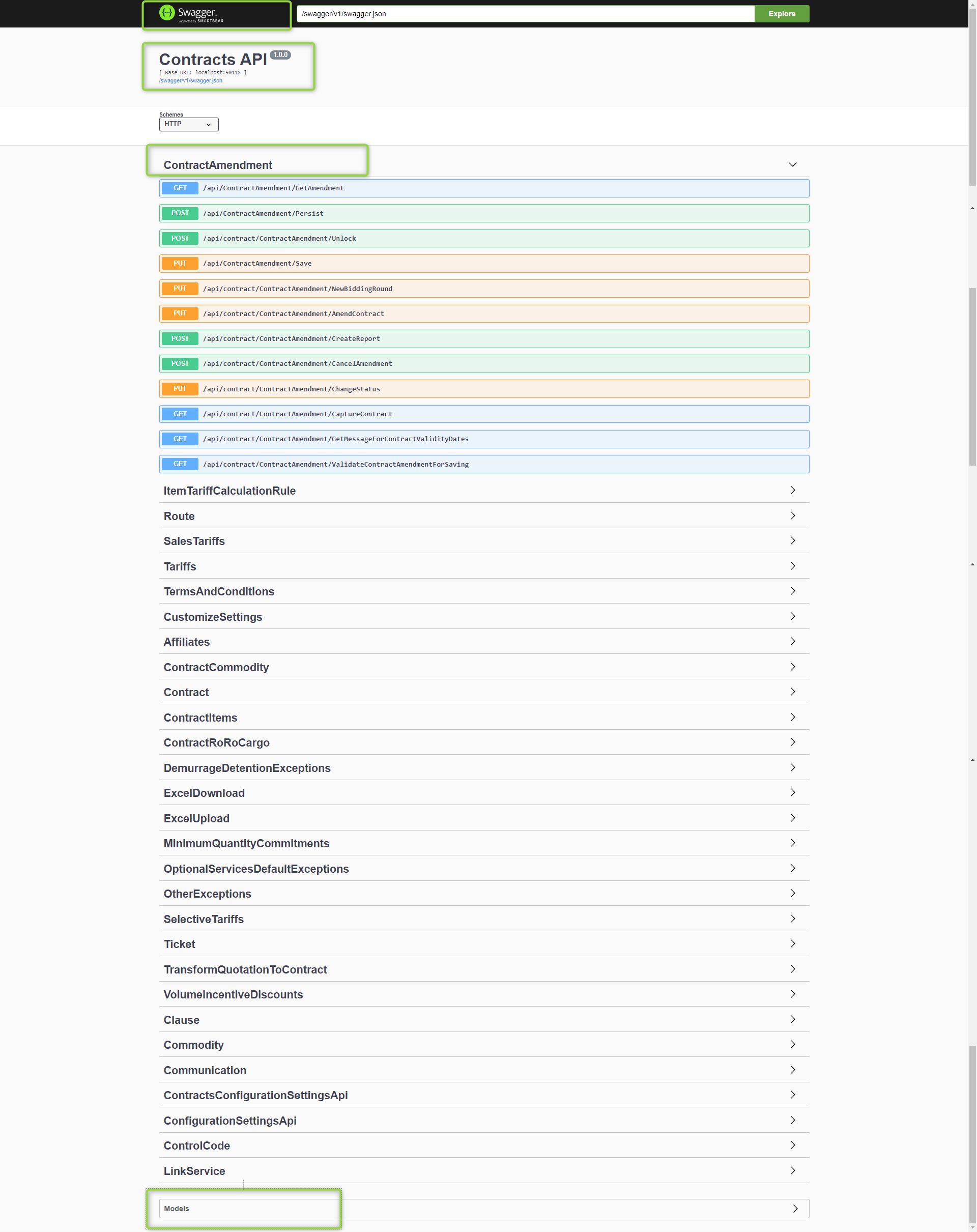


Go to corresponding controller and add Route attributes to the ActionMethods



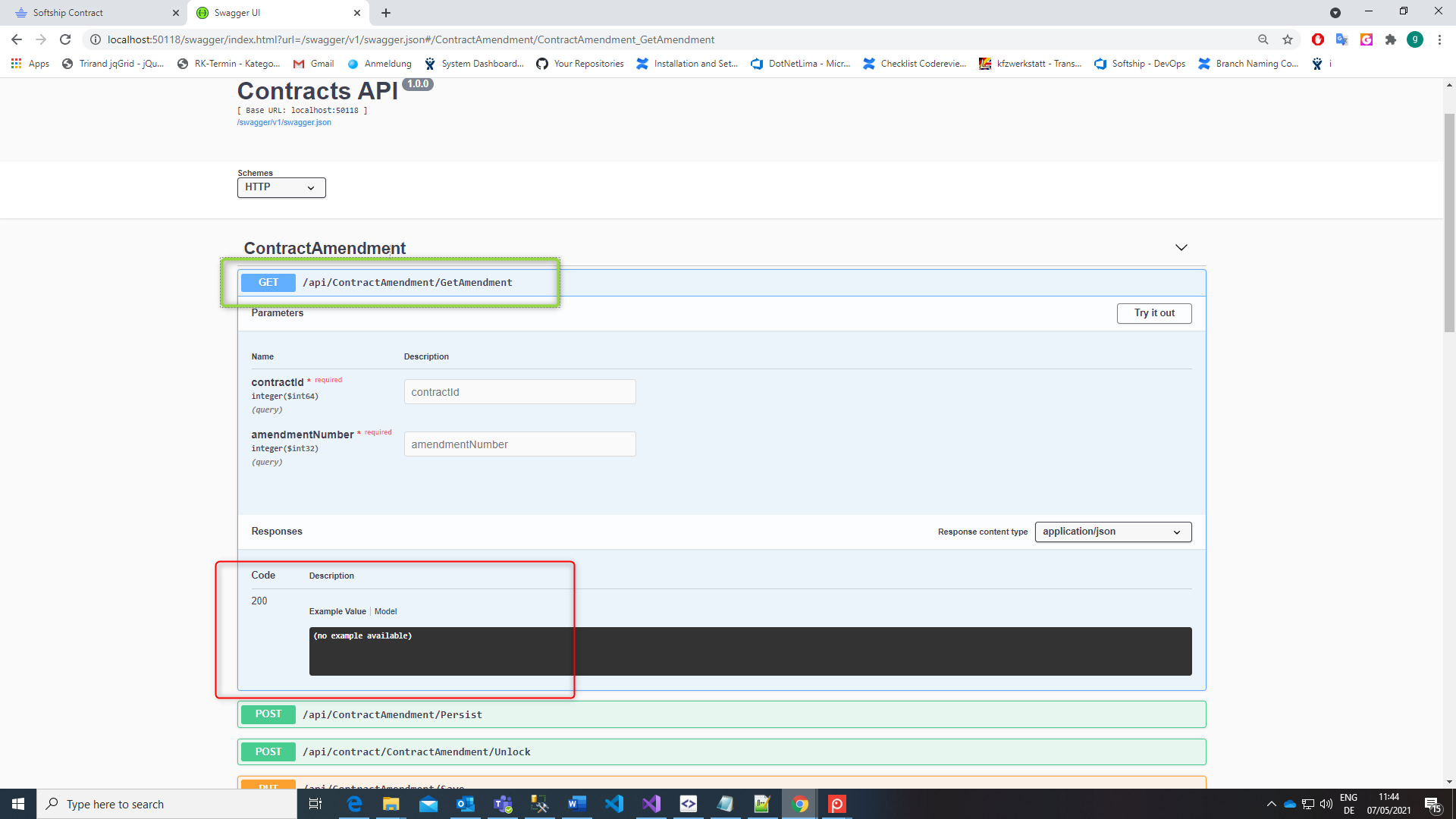
After this code fix Run the application again and redirect to swagger url.

Now successfully you can able to see the swagger UI

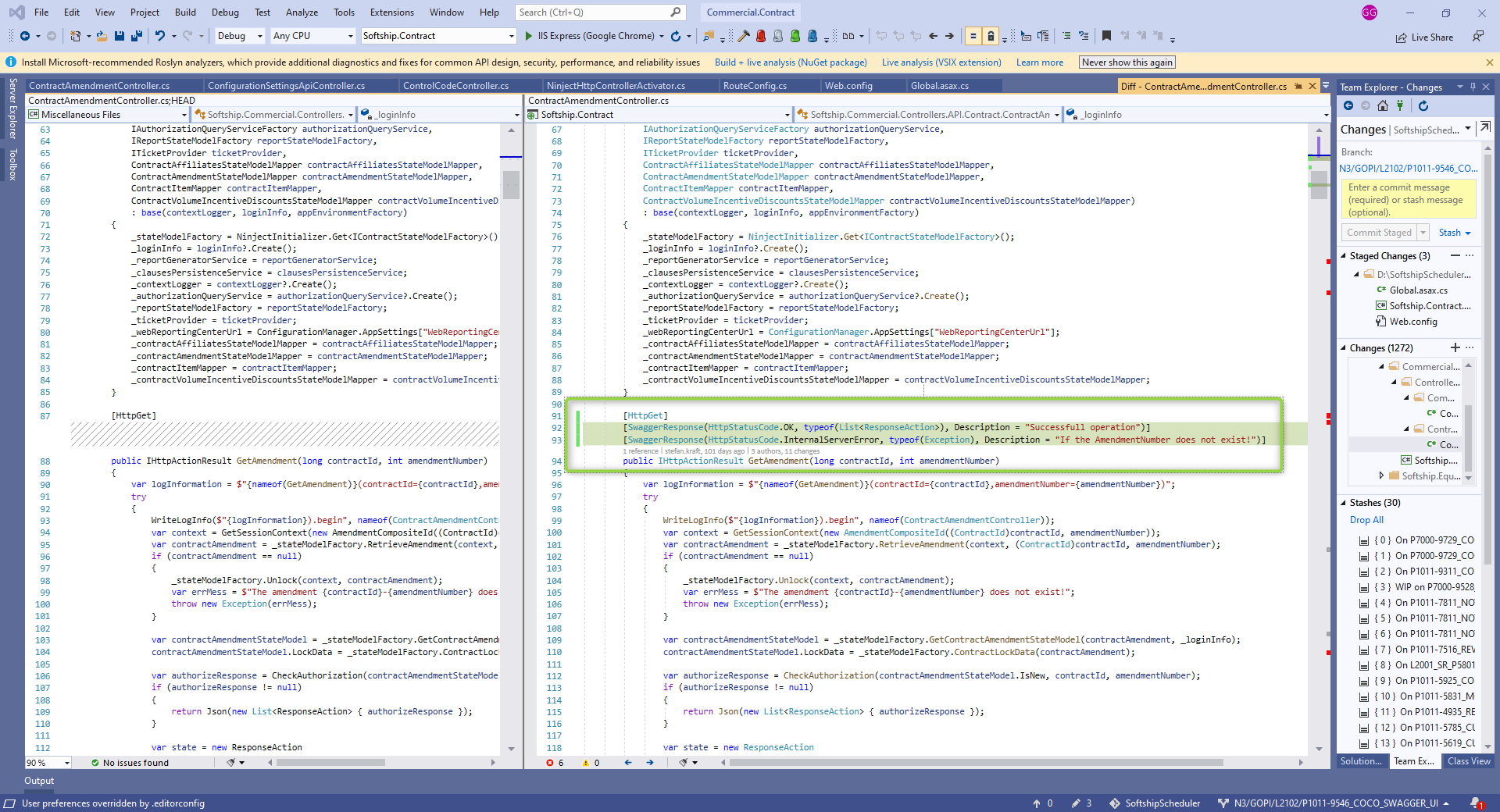


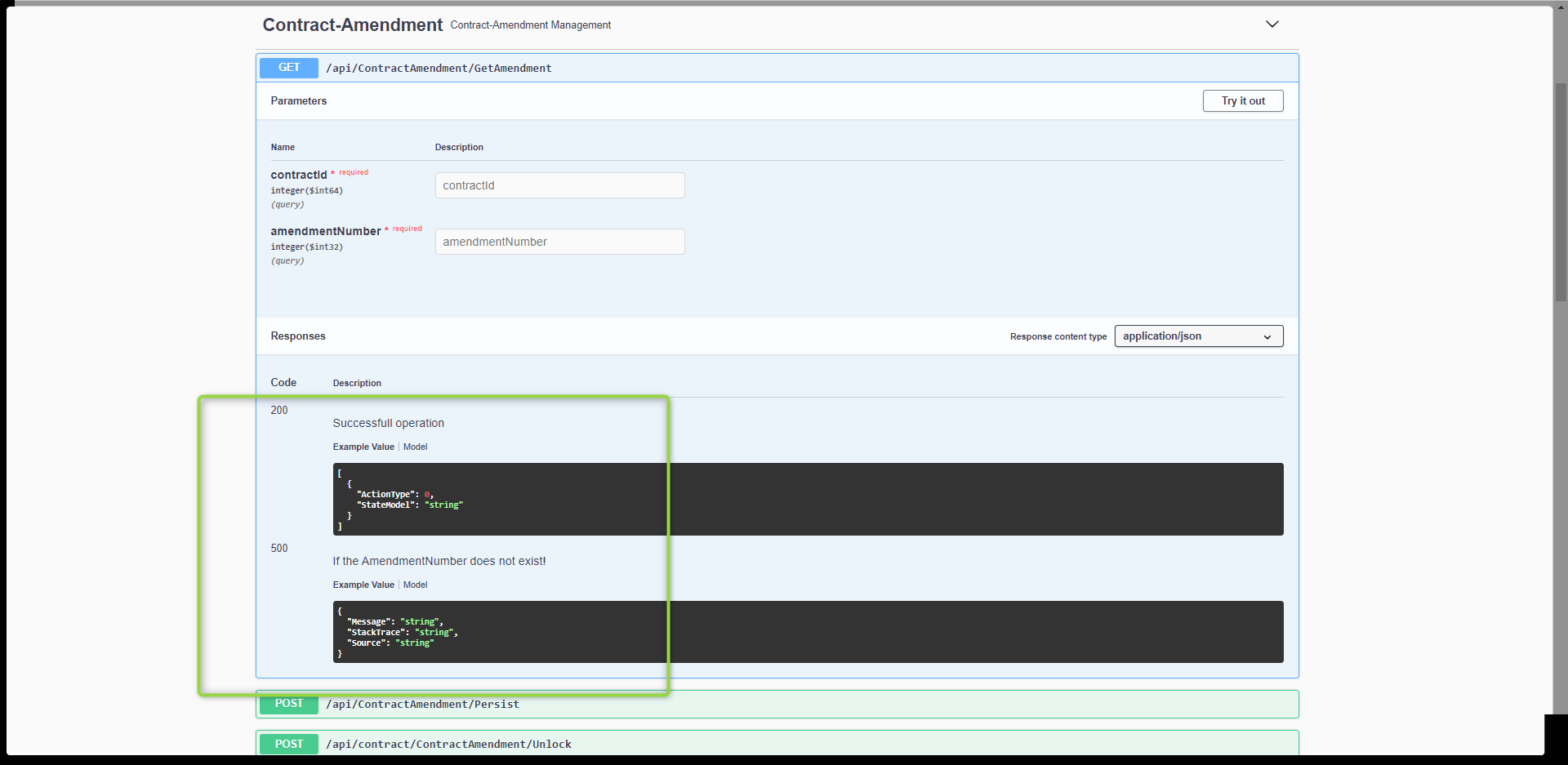
Before code changes

ContractAmendment => GetAmendment doesn’t have any 200 response type.



After adding “SwaggerResponse” attributes





**Useful references:**

<https://github.com/RicoSuter/NSwag/issues/1746>

<https://github.com/RicoSuter/NSwag/wiki/WebApiOpenApiDocumentGenerator>

<https://stackoverflow.com/questions/54343217/nswag-for-asp-net-web-api-2-show-security-in-swagger-gui>

<https://github.com/RicoSuter/NSwag/wiki/AspNetCore-Middleware#enable-api-key-authorization>

<https://github.com/RicoSuter/NSwag/blob/master/src/NSwag.Generation/Processors/Security/SecurityDefinitionAppender.cs>