

Mariusz Wieczorek

mariusz.wieczorek@kabat.pl

Streszczenie

Swagger

WEB API

Dokumentacja

**Dokumentacja Aplikacji**



Musimy dodać NuGet: **Swashbuckle.AspNetCore.** W .Net5 już jest w szablonie startowym.  
Zgodnie z dokumentacją wprowadzić zmiany do Startup.cs

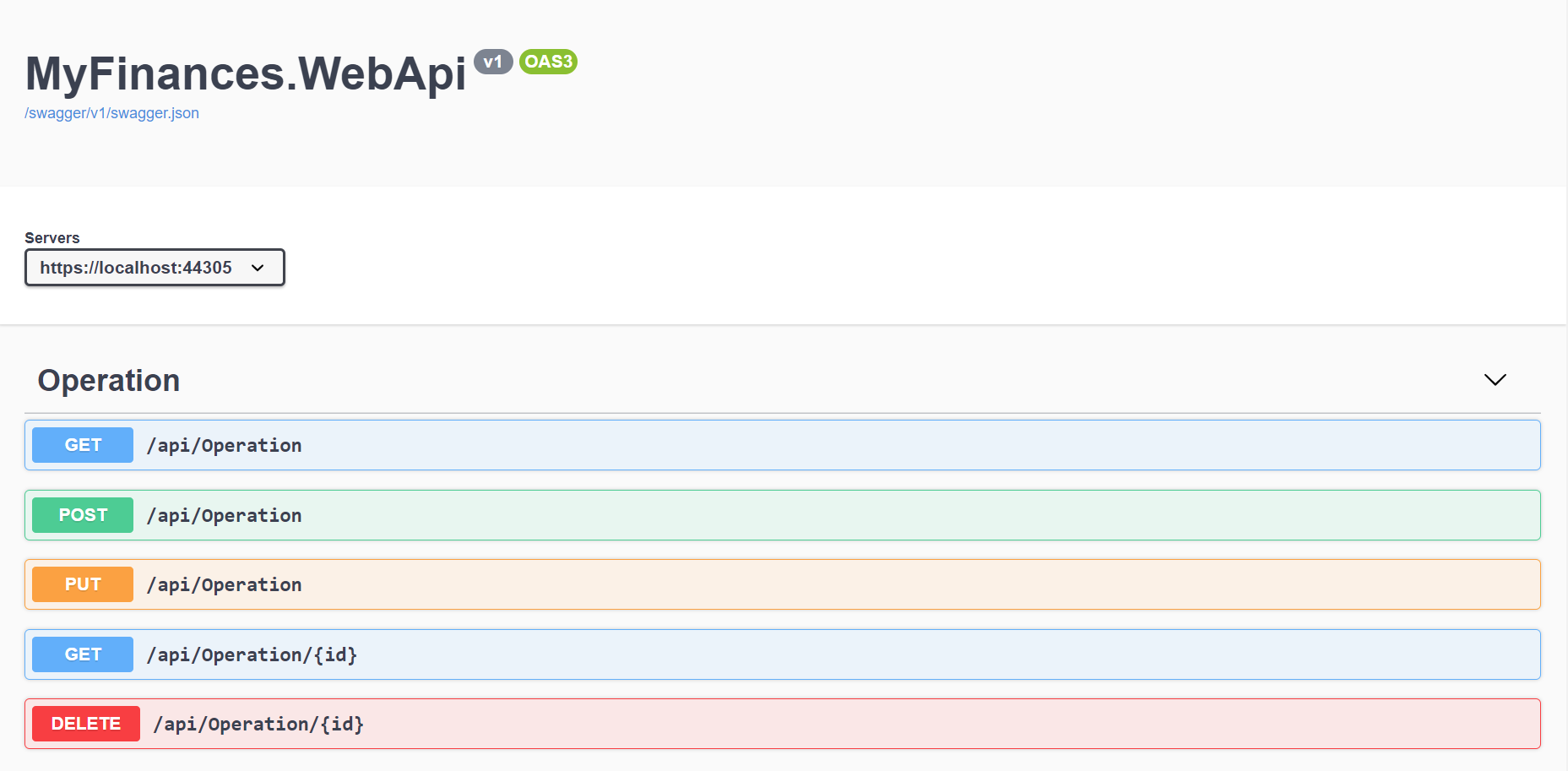
<https://docs.microsoft.com/pl-pl/aspnet/core/tutorials/getting-started-with-swashbuckle?view=aspnetcore-5.0&tabs=visual-studio>

|  |
| --- |
| public void ConfigureServices(IServiceCollection services)  {  services.AddScoped<UnitOfWork, UnitOfWork>();  services.AddDbContext<MyFinancesContext>(options =>  options.UseSqlServer(Configuration.GetConnectionString("MyFinancesContext")));  services.AddControllers();  services.AddSwaggerGen();  services.AddSwaggerGen(c =>  {  c.SwaggerDoc("v1", new OpenApiInfo { Title = "MyFinances.WebApi", Version = "v1" });  }); |

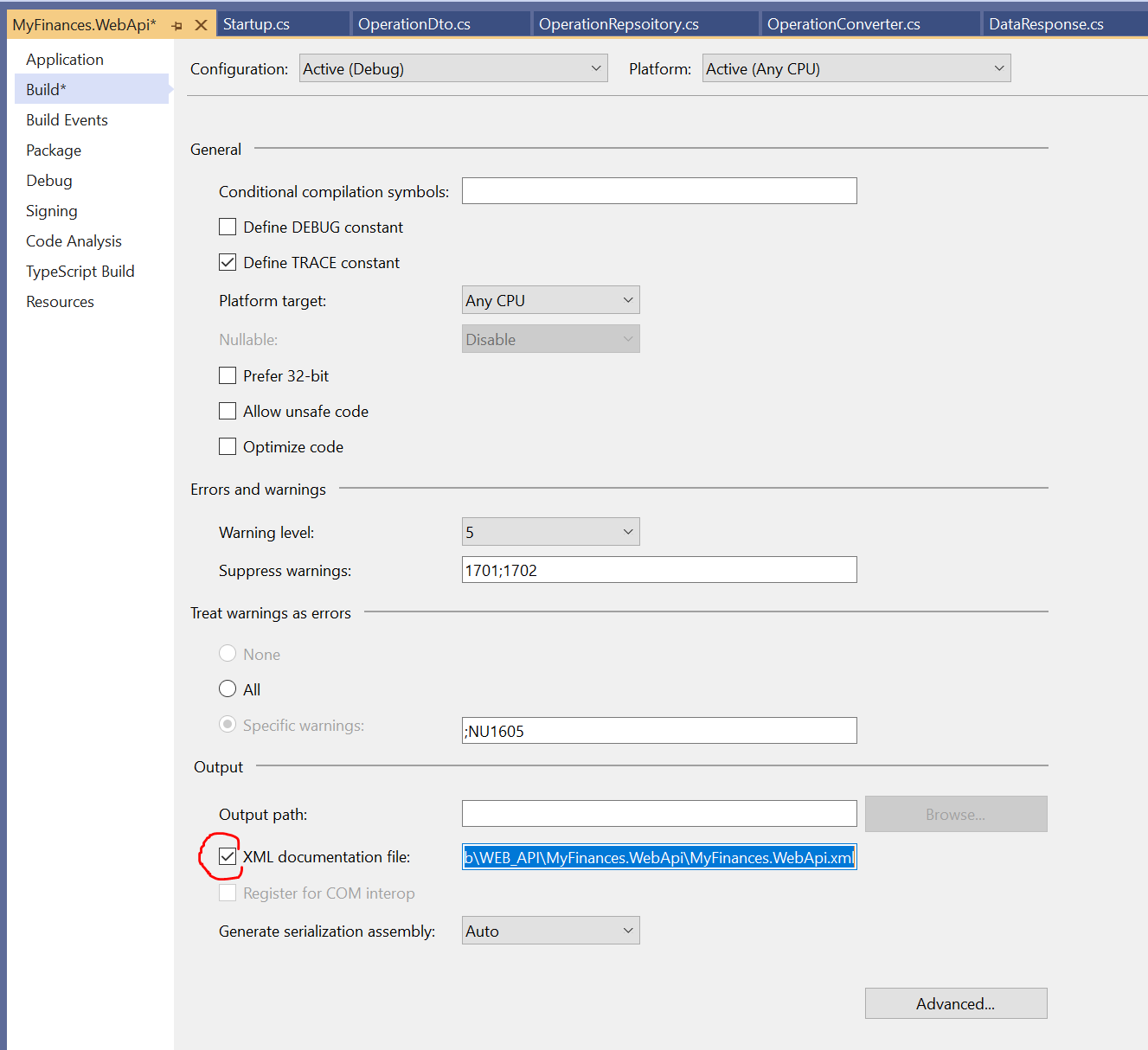
|  |
| --- |
| public void Configure(IApplicationBuilder app, IWebHostEnvironment env)  {  if (env.IsDevelopment())  {  app.UseDeveloperExceptionPage();  app.UseSwagger();  app.UseSwaggerUI(c =>   c.SwaggerEndpoint("/swagger/v1/swagger.json", "MyFinances.WebApi v1"));  } |

Po uruchomieniu

Widzimy spis naszych akcji z kontrolera Operations



Brakuje nam dokumentacji.  
Na projekcie we właściwościach wchodzimy na zakładkę build, odhaczamy opcję



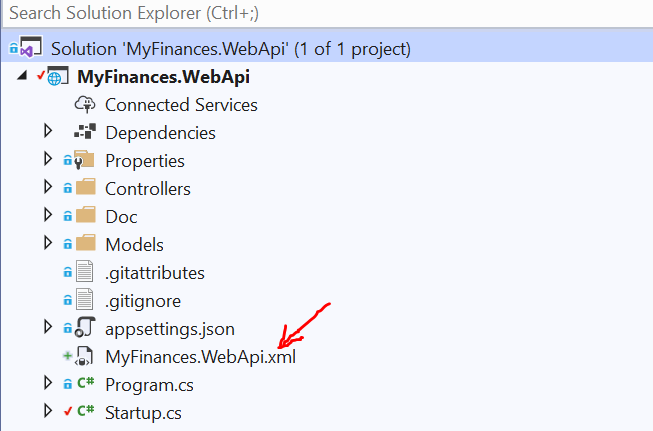
Po tym musimy w naszym kontrolerze uzupełnić dokumentację dodając komentarze do akcji za pomocą trzech znaków ///

|  |
| --- |
| /// <summary>  /// Get Operation by Id  /// </summary>  /// <param name="id"> Operation Id</param>  /// <returns>Single operation</returns>  [HttpGet("{id}")]  public DataResponse<OperationDto> Get(int id)  {  var responce = new DataResponse<OperationDto>();  try  {  responce.Data = \_unitOfWork.Operation.Get(id)?.ToDto();  }  catch (Exception ex)  {  // logowanie do pliku ...  responce.Errors.Add(new Error(ex.Source, ex.Message));  }  return responce;  } |

Musimy jeszcze dokonać zmian w Startup.cs

|  |
| --- |
| services.AddSwaggerGen(); |
| services.AddSwaggerGen(c =>  {  var xmlFile = $"{Assembly.GetExecutingAssembly().GetName().Name}.xml";  var xmlPath = Path.Combine(AppContext.BaseDirectory, xmlFile);  c.IncludeXmlComments(xmlPath);  }); |

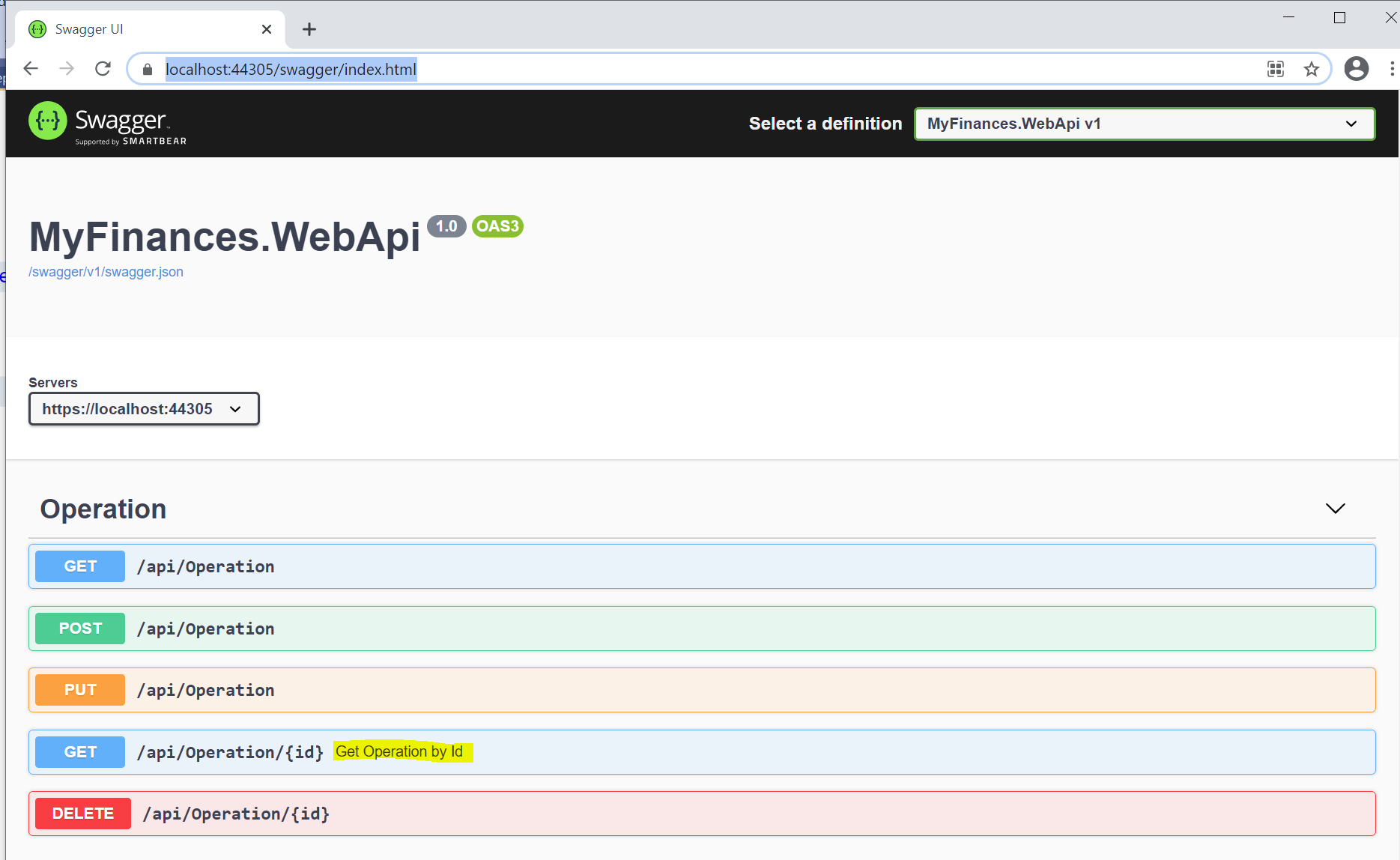
W projekcie powstał nowy plik, jest to plik naszej dokumentacji



|  |
| --- |
| <?xml version="1.0"?>  <doc>  <assembly>  <name>MyFinances.WebApi</name>  </assembly>  <members>  <member name="M:MyFinances.WebApi.Controllers.OperationController.Get(System.Int32)">  <summary>  Get Operation by Id  </summary>  <param name="id"> Operation Id</param>  <returns>Single operation</returns>  </member>  </members>  </doc> |

|  |
| --- |
| **Porada !**  Od tego momentu wszystkie metody, które nie mają komentarza będą podkreślane na zielono. Aby tego uniknąć można przegrać plik z ustawieniami w inne miejsce np. w folder Documentation a później w ustawieniach ponownie wyłączyć XML documentation ( włączenie stworzyło tylko ten plik ) a w Startup.cs ustawić nową ścieżkę do tego XML’a . |

W Swaggerze będą widoczne komentarze zapisane przy akcjach.



Swagger to też narzędzie do testowania aplikacji. Możemy ją tutaj przetestować bardzo podobnie jak w Insomni.