УДК 004.9

Документирование API веб-сервиса с помощью OpenAPI спецификации

Николаев Сергей Валерьевич
Приамурский государственный университет им. Шолом-Алейхема
магистрант

Пасюков Александр Андреевич
Приамурский государственный университет им. Шолом-Алейхема
магистрант

Якимов Антон Сергеевич
Приамурский государственный университет им. Шолом-Алейхема
магистрант

Баженов Руслан Иванович
Приамурский государственный университет им. Шолом-Алейхема
к. п. н., доцент, зав. кафедрой информационных систем, математики и методик обучения

Аннотация
Необходимость документации тех или иных аспектов своего продукта всегда присутствовала для разработчиков информационных систем. С развитием всемирной информационной паутины, роста популярности REST и интеграции различных сервисов между собой важность документации только повысилась. Отдельной темой стало документирования API своего продукта. Хорошо задокументированный API облегчает его использования, упрощает интеграцию с текущим продуктом, упрощает тестирование и дальнейшую разработку внутри команды. В данной статье будет продемонстрировано решение с использованием OpenAPI спецификации, также известной, как Swagger спецификация. В качестве примера будет использован .NET Framework и язык программирования C#.
Ключевые слова: Информационные технологии, языки программирования, документация, API, C#, .NET, OpenAPI, Swagger.

Documenting the web service API using the OpenAPI specification

Nikolaev Sergey Valerievich
Sholom-Aleichem Priamursky State University
Master student

Pasyukov Alexandr Andreevich
Sholom-Aleichem Priamursky State University
Master student

Yakimov Anton Sergeevich
Sholom-Aleichem Priamursky State University
Master student

Bazhenov Ruslan Ivanovich
Sholom-Aleichem Priamursky State University
Candidate of pedagogical sciences, associate professor, Head of the Department of Information Systems, Mathematics and teaching methods

Abstract
The need for documentation of various aspects of its product has always been present for developers of information systems. With the development of the worldwide information web and the integration of various services among themselves, the evaluation of documentation has only increased. A separate topic was the documentation of the API of its product. A well-documented API makes it easier to use, simplifies integration with the current product, simplifies testing and further development within the team. This article will show a solution using the OpenAPI specification. As an example, the .NET Framework and the C # programming language will be used.
Keywords: Information technologies, programming languages, documentation, API, REST, C #, .NET, OpenAPI, Swagger.

Одной из популярных архитектур приложений на сегодняшний момент является REST архитектура. О данной архитектуре пишет в своей диссертации R.T.Fielding [1]. Данная архитектура подразумевает стиль архитектуры программного обеспечения для распределенных систем, который, как правило, используется для построения веб-служб. Системы, поддерживающие REST, называются RESTful-системами.

Благодаря подобной архитектуре приложение хорошо масштабируется и легко расширяется. Так же веб-службы можно использовать в других приложениях, тем самым интегрировав два приложения между собой. Тем самым документация той или иной веб-службы является важным аспектом, благодаря которому упрощается использование, интеграция, дальнейшая разработка и тестирование.

В мире существует несколько решений, которые могут быть использованы в качестве готового решения. RAML[2], API Blueprint [3] и OpenAPI, так же известный как Swagger [4]. Каждое из решений базируется на своей спецификации, благодаря которой и происходит процесс документирования API. Пример автоматической документации веб-службы описан в статье А.В. Плотникова и С.В. Золотарева [6].

В качестве примера будет использоваться Swagger спецификация. Данный framework один из популярнейших на данный момент. С его помощью возможно: генерация программного кода по спецификации (генерация возможна на нескольких языках программирования), генерация спецификации из кода, построение UI интерфейса по имеющейся

спецификации и др. инструменты. Более подробно с ними можно ознакомиться на официальном сайте [4].

Сформулируем задачу. Необходимо задокументировать веб-сервис путем автоматической генерации swagger спецификации и по данной спецификации построить UI интерфейс с возможностью просматривать формат запроса и формат ответа, а также с возможностью тестовой отправки запроса.

Для демонстрации необходимо создать проект в Microsoft Visual Studio типа Web API. Созданного по умолчанию проекта вполне достаточно для поставленной цели. В нем имеется класс контроллер, его содержимое можно посмотреть на рисунке 1.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Net;
5  using System.Net.Http;
6  using System.Web.Http;
7
8  namespace webAPI.Controllers
9  {
       ссылок: 0 | 0 запросов
10     public class ValuesController : ApiController
11     {
12         // GET api/values
           ссылок: 0 | 0 запросов | 0 исключения
13         public IEnumerable<string> Get()
14         {
15             return new string[] { "value1", "value2" };
16         }
17
18         // GET api/values/5
           ссылок: 0 | 0 запросов | 0 исключения
19         public string Get(int id)
20         {
21             return $"value{id}";
22         }
23
24         // POST api/values
           ссылок: 0 | 0 запросов | 0 исключения
25         public void Post([FromBody]string value)
26         {
27         }
28
29         // PUT api/values/5
           ссылок: 0 | 0 запросов | 0 исключения
30         public void Put(int id, [FromBody]string value)
31         {
32         }
33
34         // DELETE api/values/5
           ссылок: 0 | 0 запросов | 0 исключения
35         public void Delete(int id)
36         {
37         }
38     }
39 }
40
```

Рисунок 1 - Содержимое класса контроллера

В комментариях (строки: 12, 18, 24 и т.д.) указан тип http запроса (метод) и URL(адрес) ресурса, так же известного как конечная точка.

Отметим, что в данном случае, комментарии так же являются видом документации, но доступной в локальном участке кода и только для разработчиков, у которых есть доступ к исходному коду приложения. Для достижений поставленных целей, подобной документации мало.

Запустим проект в режиме отладки (рисунок 2). В верхней части html страницы, в меню навигации можно заметить вкладку «API». Перейдя по ней, увидим список имеющихся ресурсов (рисунок 3). Если данная вкладка отсутствует, то необходимо добавить NuGet пакет “Microsoft.AspNet.WebApi.HelpPage” в данный проект.

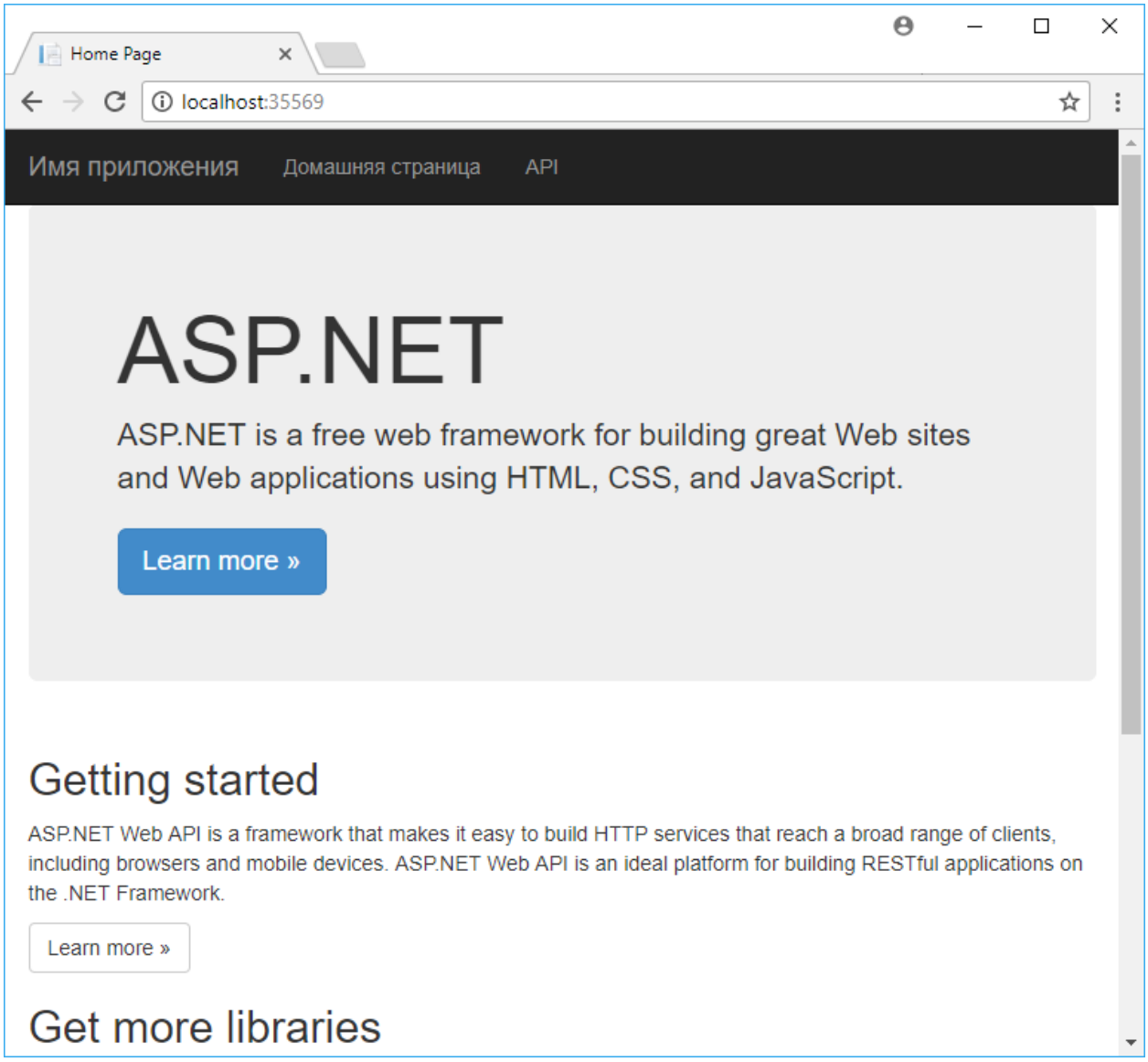

Рисунок 2 - Запущенный проект в режиме отладки

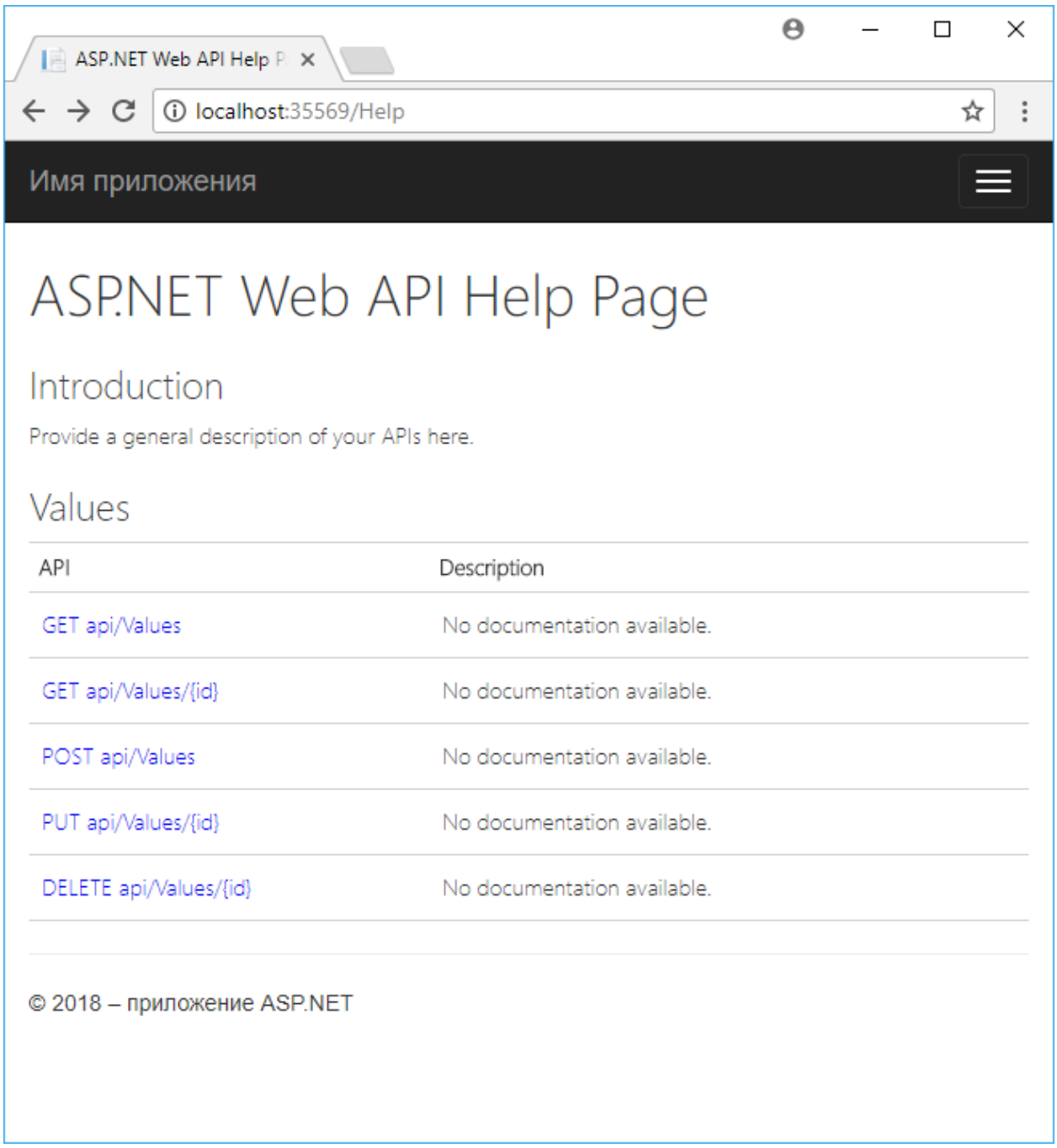

Рисунок 3 - Список конечных точек проекта

Из имеющегося списка выберем любую конечную точку, например, первую. Информация об этом ресурсе будет выведена на экран (рисунок 4). В данную информацию включены описание параметров запроса и форма ответа в json и в xml форматах. Данная документация, может быть использована, для интеграции с другими сервисами. Генерируется она автоматически, при сборке. То есть если сейчас в проект добавить еще один контроллер, информация о нем так же будет отображена.

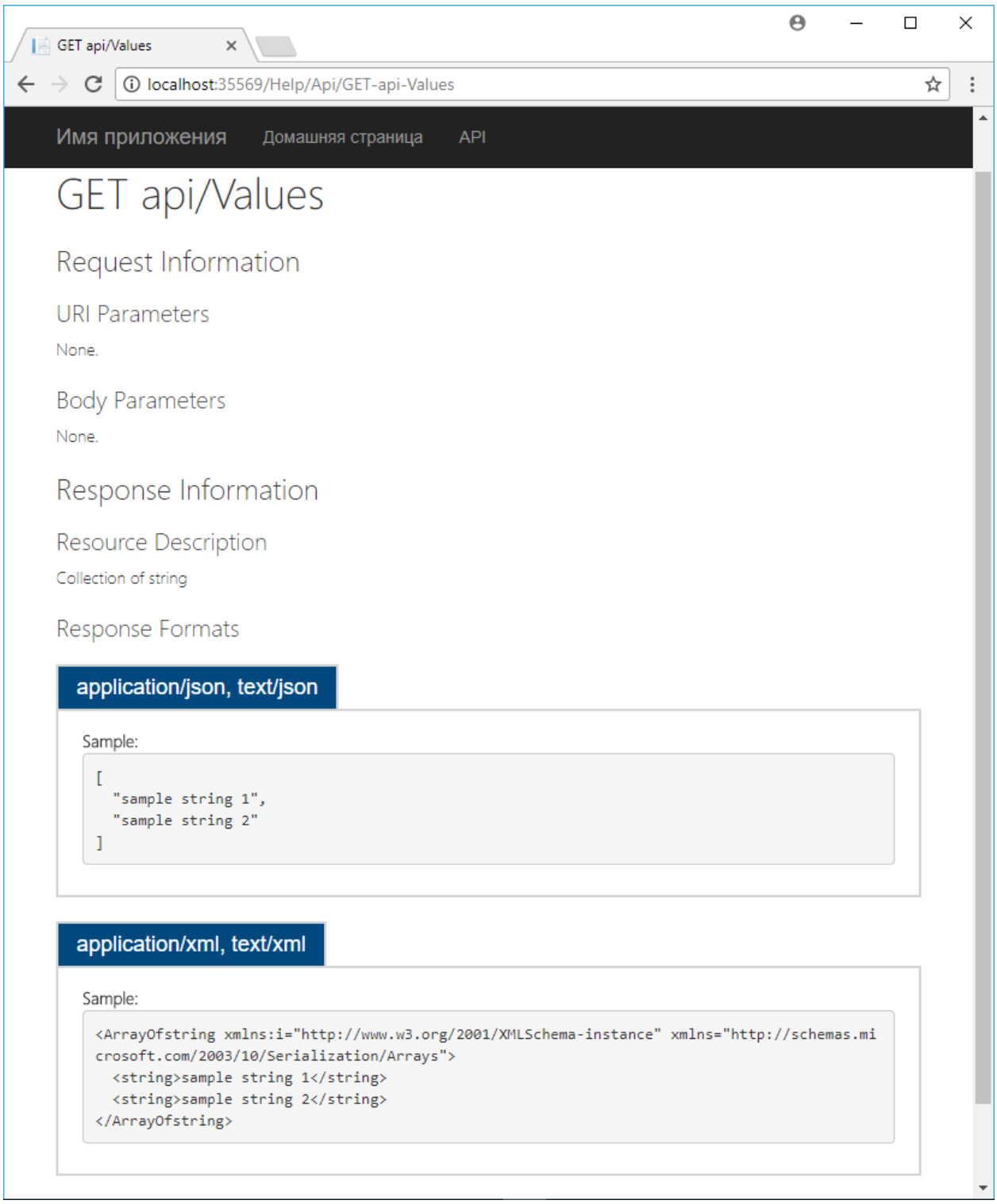

Рисунок 4 - Информация о конечной точке «api/values»

Недостатком такого способа является неудобство в тестировании. Если с GET запросом нам достаточно набрать URL конечной точки и включить при необходимости нужные параметры в адресную строку. То, например, с методом POST, нам в теле запроса, необходимо передавать описание объекта, для чего придется привлекать дополнительные инструменты. Например, инструмент «Postman», подробнее о котором можно узнать на официальном сайте [5]. Или заняться доработкой текущего функционала и внедрить возможность тестирования, что займет неопределенное время.

Более простым решением является использование, framework-a Swagger, с помощью библиотеки swashbuckler для проектов .NET web API. Данная библиотека включает в себя генерацию спецификации по имеющемуся коду, а также формирования UI интерфейса из данной спецификации. Для использования библиотеки необходимо добавить NuGet пакет Swashbuckle в свой проект (рисунок 5).

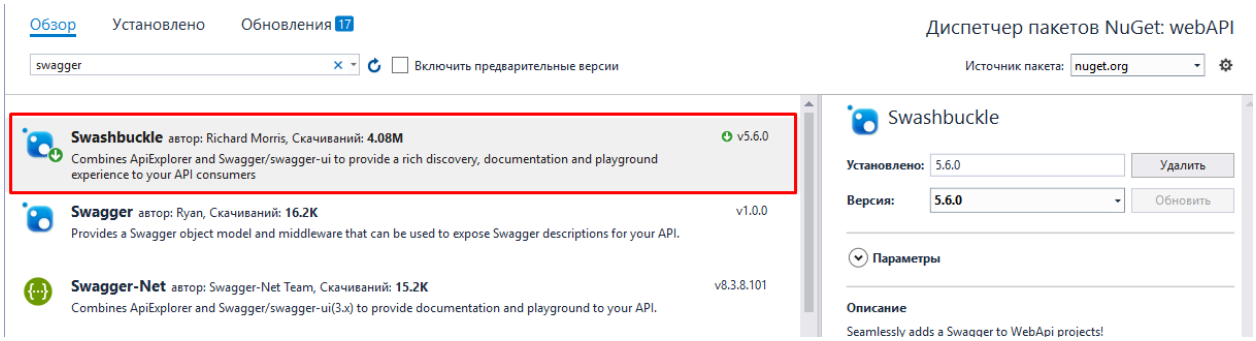

Рисунок 5 - Установленный NuGet пакет в демонстрационном проекте

После установки пакета, можно запустить проект в режиме отладки. В адресной строке будет адрес вида: http://localhost:35569. Скорее всего будет отличаться номер порта через двоеточие. Необходимо к этому адресу добавить «/swagger». Запрос будет перенаправлен к специальному интерфейсу, вид которого изображен на рисунке 6.

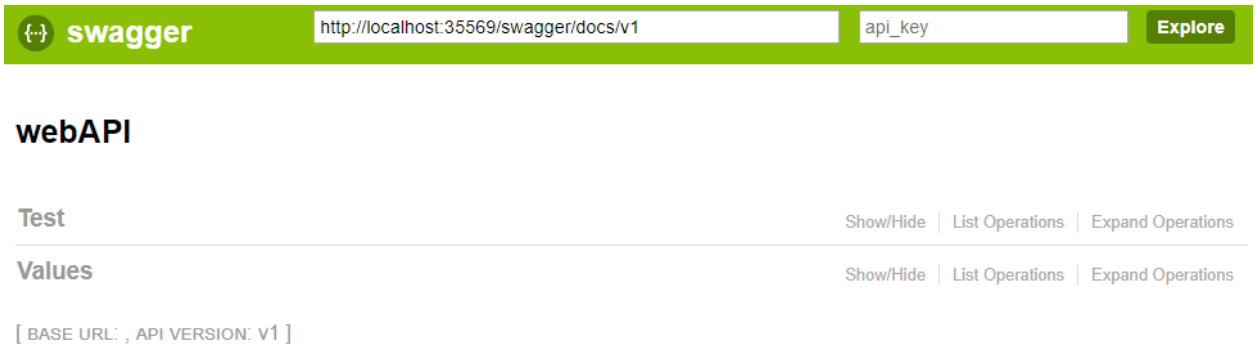

Рисунок 6 - Интерфейс документация API созданный с помощью библиотеки Swashbuckle и framework-a Swagger

Так как в проекте присутствует уже два контроллера: Test и Values, то их мы и видим на рисунке. Большая часть текста - это ссылки, нажав на которые можно получить более подробную информацию об объекте. На рисунке 7 представлена информация об одной конечной точке. К этому состоянию пришли путем клика по ссылкам: «Values», затем «GET /api/Values/{id}».

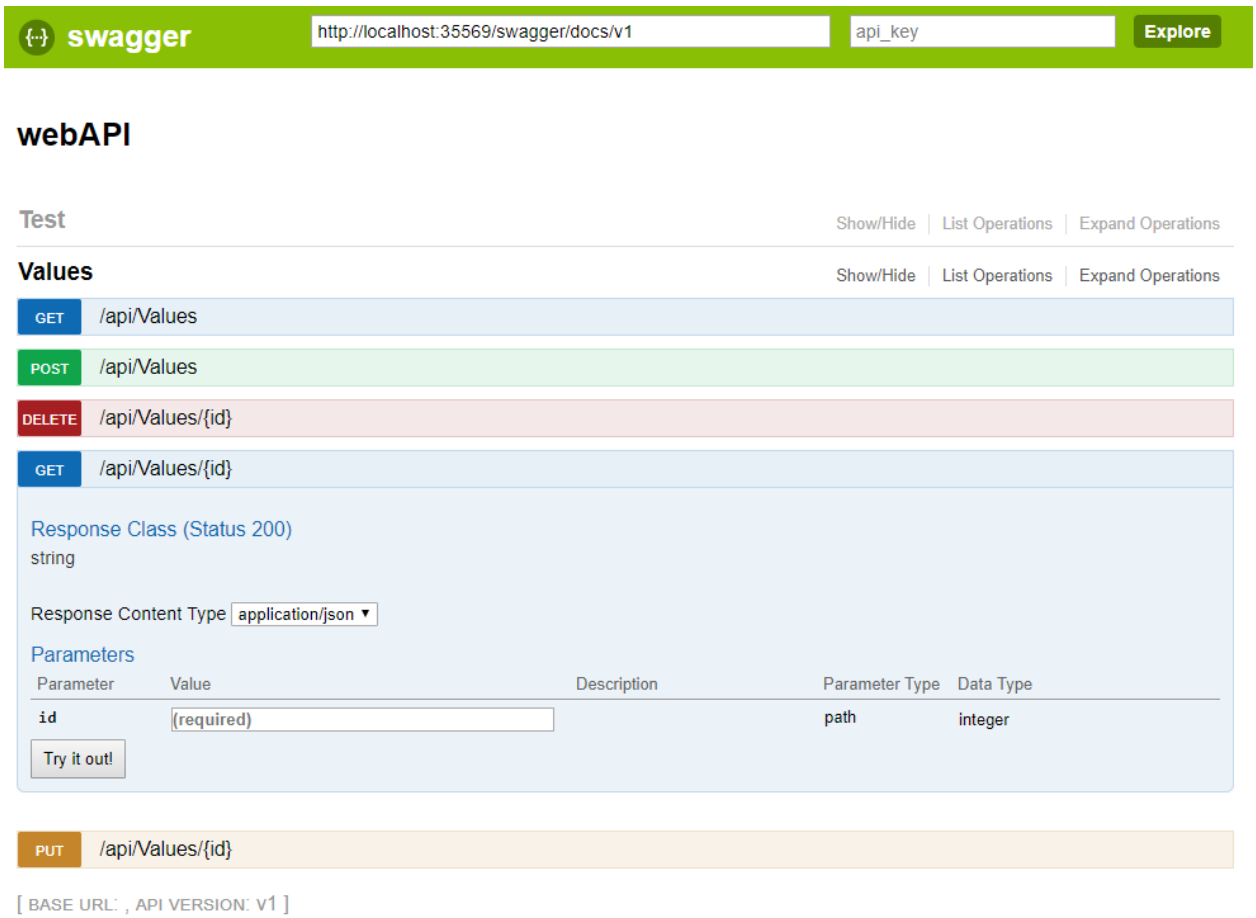

Рисунок 7 - Информация об одной из конечных точек

В описании написано, что при выполнении запроса с результатом в виде кода 200 (запрос выполнен успешно) то формат ответа будет в виде string (строка.). Ниже можно выбрать в каком формате получить ответ при тестовом выполнении запроса. На рисунке 7 указано значение: «application/json». Далее указан список параметров, в данном случае он всего один – это параметр с именем id. Для того, чтобы отправить тестовый запрос, на необходимо ввести значение параметра в графу value. Запись (required), говорит о том, что параметр обязателен и отправка запроса без него невозможна. Справа указана информация о параметрах. Данный параметр передаются в адресной строке и од должен быть целым числом, об этом говорят тип параметра «path» и тип данных integer, соответственно. На рисунке 8 изображен результат отправки тестового запроса.

Здесь присутствует команда для использования утилиты curl. curl – это консольная утилита для передачи данных используя URL-синтаксис, подробную информацию можно найти в интернете.

Ниже url, на который отправлен запрос, в данном случае это «http://localhost:35569/api/Values/13». Далее текст ответа, код ответа и заголовки ответа. Тоже самое можно проделать с другими конечными точками, отправив тестовый запрос со всеми необходимыми параметрами.

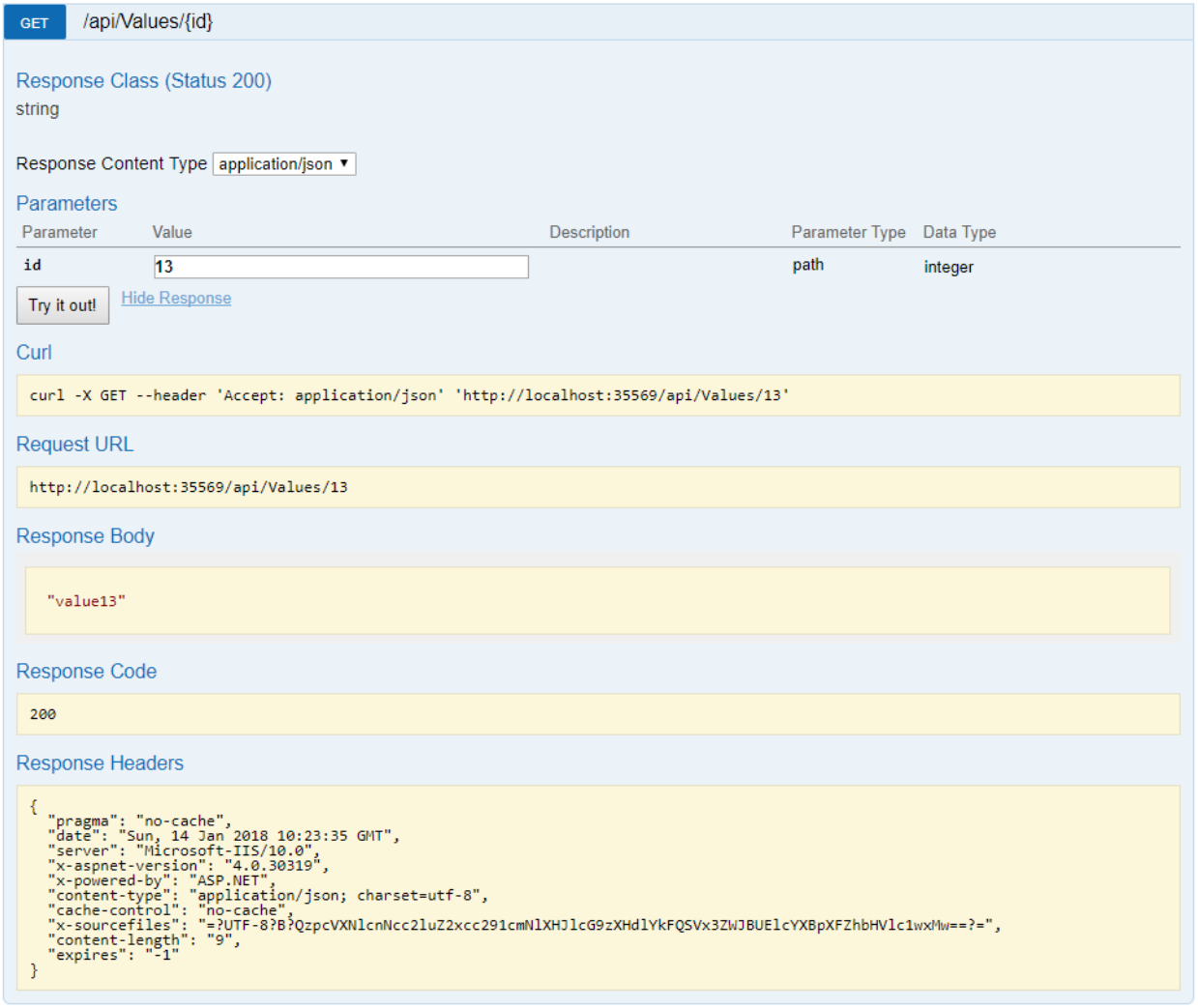

Рисунок 8 - Результаты тестового запроса

Так как данная документация формируется автоматически, то при добавлении новых контроллеров или конечных точек, они так же автоматически будут задокументированы. Данный факт несомненно облегчает труд разработчика.

Поставленная задача выполнены. Имеется задокументированный веб-сервис, документация генерируется автоматически, имеет свой визуальный интерфейс с возможностью отправлять тестовые запросы.

Но как говорилось ранее, фреймворк Swagger имеет свою спецификацию, но в процессе работы, мы не только не ознакомились с ней, но и ни разу не видели ее. На рисунке 7 вверху, в текстовом поле мы можем увидеть адрес. Если скопировать его и вставить в адресную строку браузера, то получим описание нашего сервиса в виде Swagger спецификации. Выглядит данный текст не совсем читабельно. Скопируем его полностью из окна браузера. Для удобного просмотра и изучения Swagger спецификации рекомендуется использовать инструмент, расположенный по электронному адресу: https://editor.swagger.io/ (Рисунок 9). На данном сайте уже имеется пример задокументированного API.

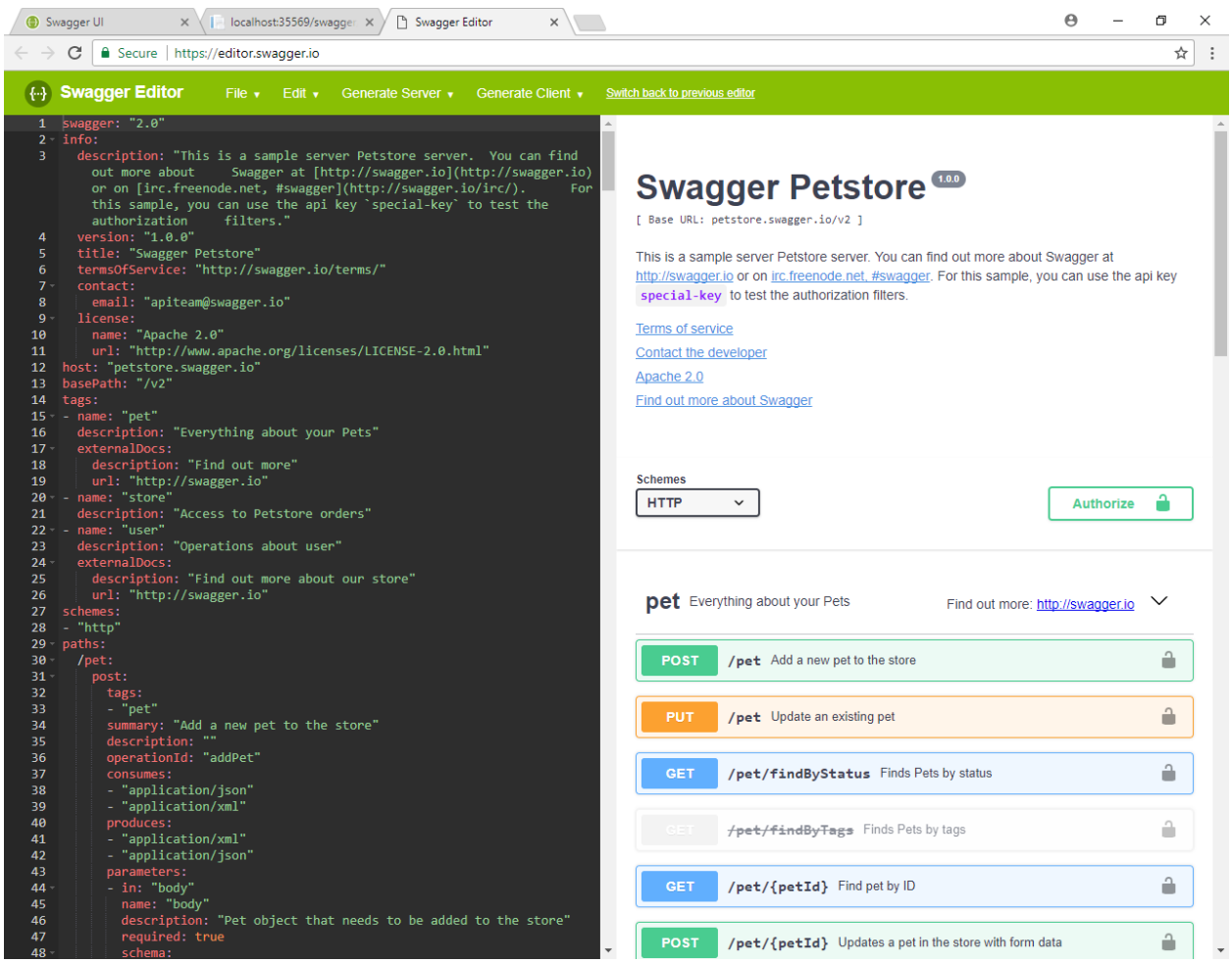

Рисунок 9 - Изображение инструмента Swagger Editor

Слева находится текст документации, справа визуальный интерфейс, который в режиме реального времени строится по тексту спецификации слева. Очистив левую часть и вставив туда текст спецификации демонстрационного веб-сервиса, скопированный ранее из браузера мы увидим справа построенный визуальный интерфейс для нашего API, который по функционалу такой же, как и интерфейс на рисунках 7-8. С помощью данного инструмента можно изучить, как будет меняться интерфейс документации от изменения спецификации. Можно добавлять конечные точки, параметры, поэкспериментировать с авторизацией и описанием.

Вверху также есть две интересные вкладки «Generate Server» и «Generate Client». Они служат для того, чтобы генерировать клиентскую или серверную часть по имеющейся спецификации. Стоит также отметить, что необязательно использовать библиотеку Swashbuckle. Можно написать собственную программу, которая будет генерировать данный текст и уже с ним работать в инструментах Swagger framework-a. Так же Swagger не привязан к языку программирования, например, генерация серверной и клиентской части можно производить не только на языке C#, но и на большинстве популярных языках программирования.

Библиографический список

1. Roy Thomas Fielding. Architectural Styles and the Design of Network-based Software Architectures. // https://www.ics.uci.edu URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf (дата обращения: 12.01.2018).
2. Официальный сайт RAML URL: https://raml.org/ (дата обращения: 13.01.2018).
3. Официальный сайт API Blueprint. URL: https://apiblueprint.org/ (дата обращения: 13.01.2018).
4. Официальный сайт OpenAPI (Swagger). URL: https://swagger.io/ (дата обращения: 13.01.2018).
5. Официальный сайт проекта Postman. URL: https://www.getpostman.com/ (дата обращения: 13.01.2018).
6. Плотников А.В., Золотарев С.В. Автоматическое документирование интерфейса веб-службы, описание rest api службы // Материалы XVII Международной научно-методической конференции «Информатика: проблемы, методология, технологии», 2017. С 134-138.