МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

МАТЕРИАЛЫ

VII Международной молодежной научной конференции

«МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ИНФОРМАЦИОННЫХ, ТЕХНИЧЕСКИХ И ЭКОНОМИЧЕСКИХ СИСТЕМ»

Томск, 23–25 мая 2019 г.

Под общей редакцией
кандидата технических наук И.С. Шмырина

Томск
Издательский Дом Томского государственного университета
2019

4. *Томская К.М.* Определение метода шкалирования для идентификации нот с помощью частот основного тона // Российская наука в современном мире: сборник статей XVI международной научно-практической конференции. – 2018. – С. 88–91.

5. *Якимук А.Ю.* Генерация фильтров для одновременной маскировки // Электронные средства и системы управления. – 2018. – № 1–2. – С. 29–31.

6. *Leydon C., Bauer J.J., Larson C.R.* The role of auditory feedback in sustaining vocal vibrato // Acoustical Society of America. – 2003. – V. 114(3). – P. 1575–1581.

7. *Ferguson S., Moere A.V., Cabrera D.* Seeing sound: real-time sound visualisation in visual feedback loops used for training musicians // Ninth International Conference on Information Visualisation (IV'05). London, UK. – 2005. – P. 97–102.

СОЗДАНИЕ API-КЛИЕНТОВ С ПОМОЩЬЮ КОД-ГЕНЕРАТОРОВ

А.Д. Никифоров, С.И. Самохина
Томский государственный университет

Введение

В современном web-программировании отдельное место заняла разработка API. API (application programming interface) – это язык, регламентированный способ общения одной компьютерной программы с другой, для исполнения какой-нибудь общей задачи, когда одна программа выполняет запросы другой [1]. Если говорить о web-программировании, то также к аббревиатуре API необходимо добавить аббревиатуру REST(Representational State Transfer) – это архитектурное решение взаимодействия компонентов информационной системы расположенных в сети интернет. REST определяет ряд правил, которые должны соблюдаться при проектировании программного продукта [2].

1. Обозначение проблемы

В процессе разработке новых интернет приложений программисту очень часто приходится обращаться к чужим API, использовать их интерфейс и внедрять в свою программу. Существует несколько ситуаций, в которых создание API будет целесообразным:

1. Мобильное приложение. Множество мобильных приложений для различных сервисов работают при использовании API этих самых сервисов. Если описано API, сделано мобильное приложение, то клиент со смартфоном будет получать информацию в свое устройство именно через API.

2. Открытое программное обеспечение. Создаётся API, при помощи которого пользователи при желании смогут создать новые клиенты для существующего приложения или новые сервисы на его основе.

3. Максимальное разделение frontend и backend. Например, при использовании frontend -фреймворков.

При интеграции с платформой, предлагающей API REST, разработчик может использовать сгенерированную или написанную человеком специальную библиотеку – SDK, также у программиста есть возможность самому писать HTTP-запросы для взаимодействия с чужим API.

Выбирая из этих двух вариантов, с уверенностью можно сказать, что первый – наиболее удобный, так как необходимо только обратиться к уже существующим методам и классам, чтобы вызвать функцию определенного API. Второй вариант тоже имеет место быть при отсутствии уже сделанных библиотек или же при незнании популярных языков программирования, на которых обычно и делаются API-клиенты [3].

Теперь необходимо разобраться в том, как сделать библиотеку или SDK, которую можно будет подключить в любой проект.

Существует несколько путей решения этой проблемы. Первый способ – вручную. Вся специфика подключения к API и генерация всех классов будет происходить с помощью программиста-разработчика. Этот подход имеет ряд преимуществ, одно из ко-

торых то, что человек может учесть все нюансы уже созданного API и реализовать их в SDK, которое будет распространятся для разработки новых сервисов. Но в таком подходе есть и большой минус – время, затрачиваемое на разработку таких пакетов разработчиком. Другой путь – это генераторы, которые на основе документации к API могут автоматически создать код на необходимом языке программирования.

Таким образом, имея в арсенале код-генератор и правильно написанную документацию API, есть возможность создать SDK, которое можно свободно использовать другим разработчикам. Однако, возникает несколько вопросов. Во-первых, где взять документацию API. Во-вторых, как сформулировать общий регламент спецификации? И в-третьих, как сделать этот регламент доступным для большинства код-генераторов? На помощь в этой ситуации пришла спецификация OpenAPI.

2. Спецификация OpenAPI

Спецификация OpenAPI – это формат описания API или язык определения API. Файл спецификации OpenAPI позволяет описывать API, включая (среди прочего): общую информацию об API, доступные пути (/ресурсы), доступные операции по каждому пути (get / resources), вход / выход для каждой операции.

Спецификацию OpenAPI Specification можно найти в репозитории github OpenAPI Initiative. В этом документе описываются все аспекты спецификации OpenAPI. Использование языка определения API, такого как спецификация OpenAPI, позволяет легко и быстро описать API. Это особенно полезно, когда идёт процесс разработки. Будучи простым текстовым файлом, файл спецификации OpenAPI можно совместно использовать и управлять в любом VCS. После написания файла спецификации его можно использовать в нескольких случаях: исходный материал для документации; спецификация для разработчиков; частичное или полное генерирование кода [4]. Файл OpenAPI может быть написан либо в JSON, либо в YAML. Пример спецификации на рис. 1.

```
{
    "swagger": "2.0",
    "info": {
        "version": "1.0.0",
        "title": "Simple API",
        "description": "A simple API to learn how to write OpenAPI Specification"
    },
    "schemes": [
        "https"
    ],
    "host": "simple.api",
    "basePath": "/openapi101",
    "paths": {
        "/persons": {
            "get": {
                "summary": "Gets some persons",
                "description": "Returns a list containing all persons.",
                "responses": {
                    "200": {
                        "description": "A list of Person",
                        "schema": {
                            "type": "array",
                            "items": {
                                "properties": {
                                    "firstName": {
                                        "type": "string"
                                    },
                                    "lastName": {
                                        "type": "string"
                                    },
                                    "username": {
                                        "type": "string"
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

Рис. 1. Пример описания API по спецификации OpenApi в формате json

3. Создание документации OpenAPI

Для того, чтобы создать документацию OpenAPI для WebAPI проекта необходим nuget пакет – Swashbuckle. Swashbuckle знает, как создать спецификацию OpenAPI для WebAPI.

Чтобы добавить Swashbuckle к проекту, необходимо ввести в командную строку следующую команду: dotnet add package Swashbuckle.

Теперь необходимо подключить Swashbuckle к проекту и создать документацию OpenAPI.

Базовая конфигурация для Swashbuckle довольно проста. Во-первых, нужно добавить следующую процедуру в ConfigureServices – метод Startup класса, чтобы зарегистрировать генератор Swagger и определить документ OpenAPI (рис. 2).

```
services.AddSwaggerGen(o => {
    o.SwaggerDoc(
        "v1",
        new Info
        {
           Title = "swaggerPlayground API",
           Version = "v1"
        });
});
```

Рис. 2. Регистрация swagger в ConfigureServices

Во-вторых, добавить промежуточное программное обеспечение, в котором сгенерированный документ Swagger будет сервером. Для этого необходимо добавить следующие строки в Configure метод (рис. 3).

```
app.UseSwagger();
app.UseSwaggerUI(o =>
{
    o.SwaggerEndpoint("/swagger/v1/swagger.json", "API");
});
```

Рис. 3. Добавление swagger в приложение, как middlware сервис

На этом этапе можно запустить приложение и перейти к: http://localhost:5000/swagger/v1/swagger.json. Должен получиться JSON документ, как на рис. 4. Это спецификация OpenAPI – на основе неё, с помощью код генератора, можно создать клиентскую библиотеку.

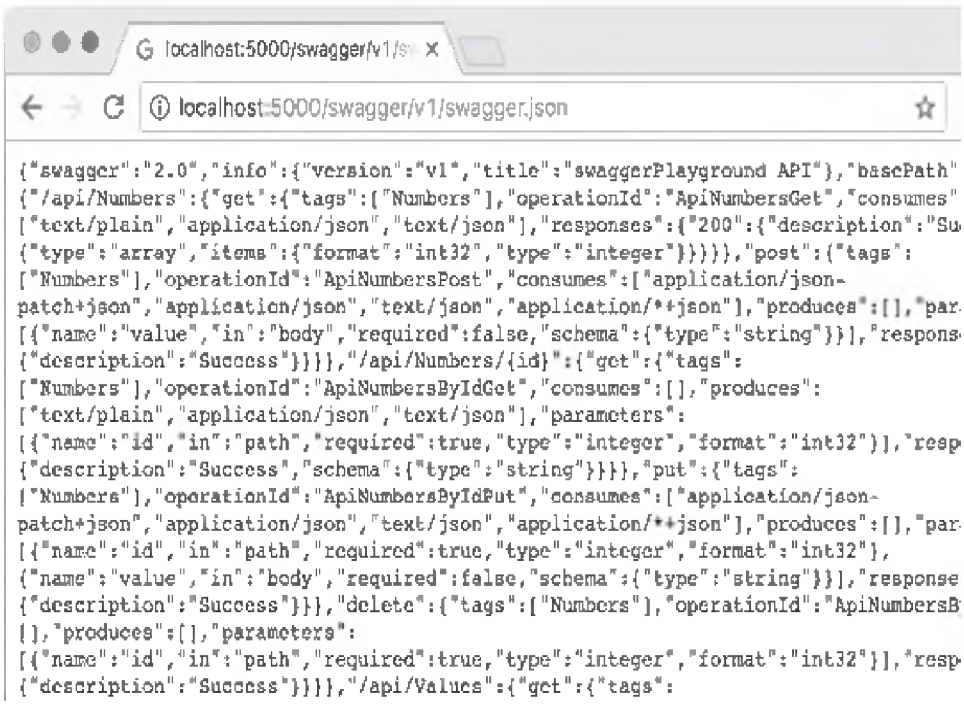

Рис. 4. Спецификация OpenApi

4. Генерация SDK

После того, как создана спецификация OpenAPI, из неё можно генерировать SDK, которое потом будет использоваться разработчиками для создания новых приложений. Вот несколько примеров программ, разработанных специально для автоматического создания SDK: APIMATIC, Restlet Studio, Swagger Codegen, REST United, AutoRest. У каждой из этих программ есть свои особенности предоставления инструментов для генерации кода, но суть у всех одна и та же. Для примера рассмотрим программу AutoRest. AutoRest – это инструмент для создания клиентских библиотек с использованием файла спецификации, который описывает REST API. Используя autorest, есть возможность создавать клиентские библиотеки для следующих языков: C#, NodeJS, Python, Java, Ruby, Go. Теперь необходимо взять OpenAPI файл и создать клиентскую библиотеку. Для этого необходимо предоставить AutoRest входной файл, выходную папку и язык, на котором необходимо создать клиентскую библиотеку. Например, на рис. 5 показано, как из командной строки Windows можно начать генерацию на языке C#.

```
autorest --input-file=swagger.json --output-folder=generated_csharp
--csharp
```

Рис. 5. Команда Autorest для создания кода

Аналогичным образом, по необходимости, можно генерировать клиентскую библиотеку для любого другого языка, заменяя «--csharp», например, на «python», «go» или другим поддерживаемым языком. Если служба защищена, и, например, требуется токен OAuth, необходимо также добавить команду «--add-credentials». Это создаст несколько разные конструкторы для принятия клиентом учетных данных.

После проделанных манипуляций AutoRest создаст файлы с кодом на выбранном языке, в которых будут обращения к API в соответствии с описанной документацией

OpenAPI. Когда есть сгенерированный код, необходимо создать пользовательскую библиотеку, которую можно подключить к стороннему проекту.

В данный момент AutoRest не создаёт файл csproj вместе с сгенерированным кодом, поэтому нужно вручную создать проект в Microsoft Visual Studio типа Class Library с использованием технологии .NET Framework. Далее необходимо добавить сгенерированные классы в созданный проект, после чего собрать его. Теперь в папке bin проекта есть библиотека, которую можно добавить в nuget пакет и выложить его на хостинг nuget.org. После этих действий сгенерированную библиотеку можно добавлять в любой проект.

Заключение

Автоматическая генерация SDK – сложная перспектива; есть много вещей, которые необходимо учитывать при попытке автоматически генерировать код. Отсутствующие данные, такие как определения моделей, коды ошибок и информация аутентификации, могут привести к тому, что автоматически сгенерированный код будет иметь очень низкое качество, а в некоторых случаях код не будет работать. Дизайн API также является основным фактором при попытке автоматически генерировать код. Нет единого, широко принятого, стандартизованного в отрасли стандарта проектирования для API. Не говоря уже о том, что некоторые API разработаны плохо, возвращая некорректно отформатированный JSON и производя неожиданные результаты.

ЛИТЕРАТУРА

1. Tilda API. Автоматическая интеграция проекта на Тильде с собственным сайтом [Электронный ресурс]. – URL: http://help-ru.tilda.ws/api (дата обращения 11.03.2019)

2. *Masse M.* REST API Design Rulebook. – O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2012. – 94 с.

3. API — Википедия [Электронный ресурс]. – URL: https://ru.wikipedia.org/wiki/API (дата обращения 11.03.2019)

4. About Swagger Specification [Электронный ресурс]. – URL: https://swagger.io/docs/specification/about/ (дата обращения: 11.03.2019).