# Генерация сетевого слоя на основе OpenAPI (Swagger) спецификации

Теория + практика

Антон Давыдов @dydus0×14 Swoo & Кошелёк

# Дисклеймер:

При использовании на практике рассмотренных подходов из вашей Core/инфраструктурной команды могут быть уволены разработчики. Используйте на свой страх и риск. Podlodka и спикер ответственности не несут.

# Как разговаривают клиент и сервер?

#### API

- REST (JSON/XML)
- RPC (gRPC/Thrift)
- SOAP
- GraphQL
- •

# В высокой долей вероятности у вас REST API с JSON

# Типичные проблемы и сложности на практике при разработке/поддержке API

- Нет документации
- Версионирование
- Много бойлерплейт кода (в сериализации)
- Нужно ковыряться в нюансах работы АРІ

# Способы декларации АРІ и документирования

- Из уст в уста
- В базе знаний список эндпойнтов
- Примеры curl, postman запросов
- IDL (RPC)
- OpenAPI/Swagger (REST)

Хорошая документация - это

Когда не надо изучать исходники или снифать траффик, чтобы понять как работает API

# Что такое OpenAPI и Swagger?





https://www.openapis.org

https://swagger.io

# OpenAPI Specification (OAS)

The OpenAPI Specification allows the description of a remote API accessible through HTTP or HTTP-like protocols.

https://oai.github.io/Documentation/specification.html

# Пример. Twitter OAS #1

```
"openapi" : "3.0.0",
"info" : {
  "description" : "API Reference - Labs v2",
  "version" : "2.13",
  "title" : "Tweets and Users",
  "termsOfService": "https://developer.twitter.com/en/developer-terms/agreement-and-policy.html",
  "contact" : {
    "name" : "Twitter Developers",
    "url" : "https://developer.twitter.com/"
  "license" : {
    "name" : "Twitter Developer Agreement and Policy",
    "url": "https://developer.twitter.com/en/developer-terms/agreement-and-policy.html"
},
"servers" : [ {
  "description" : "Twitter API",
  "url" : "https://api.twitter.com"
"tags" : [ {
  "name" : "General",
  "description" : "Miscellaneous endpoints for general API functionality",
  "externalDocs" : {
    "description" : "Find out more",
    "url" : "https://developer.twitter.com/en/docs/labs"
}, {
  "name" : "Tweets",
  "description" : "APIs related to Tweets",
  "externalDocs" : {
    "description" : "Find out more",
    "url" : "https://developer.twitter.com/en/docs/labs/tweets-and-users/overview"
}, {
  "name" : "Users",
  "description" : "APIs related to Users",
  "externalDocs" : {
    "description" : "Find out more",
    "url" : "https://developer.twitter.com/en/docs/labs/tweets-and-users/overview"
```

https://api.twitter.com/labs/2/openapi.json

# Пример. Twitter OAS #2

```
"/labs/2/users/{id}" : {
  "get" : {
    "tags" : [ "Users" ],
   "summary" : "Return details for the specified users",
    "description": "This endpoint returns information about a user. Specify user by ID.",
    "operationId" : "findUserById",
    "parameters" : [ {
     "name" : "id",
     "in" : "path",
     "description" : "Required. A User ID.",
     "required" : true,
     "schema" : {
       "$ref": "#/components/schemas/UserID"
   }, {
     "$ref": "#/components/parameters/UserExpansionsParameter"
     "$ref" : "#/components/parameters/TweetFieldsParameter"
   }, {
     "$ref" : "#/components/parameters/UserFieldsParameter"
   }, {
     "$ref": "#/components/parameters/MediaFieldsParameter"
     "$ref" : "#/components/parameters/PlaceFieldsParameter"
   }, {
     "$ref": "#/components/parameters/PollFieldsParameter"
   } ],
    "responses" : {
     "200" : {
        "description" : "The request was successful",
        "content" : {
          "application/json" : {
            "schema" : {
              "$ref": "#/components/schemas/SingleUserLookupResponse"
     "default" : {
        "$ref": "#/components/responses/HttpErrorResponse
```

https://api.twitter.com/labs/2/openapi.json

#### OAS

- Описывает REST API с помощью JSON, YAML
- Machine-readable
- Human-readable

# **OpenAPI и Swagger**

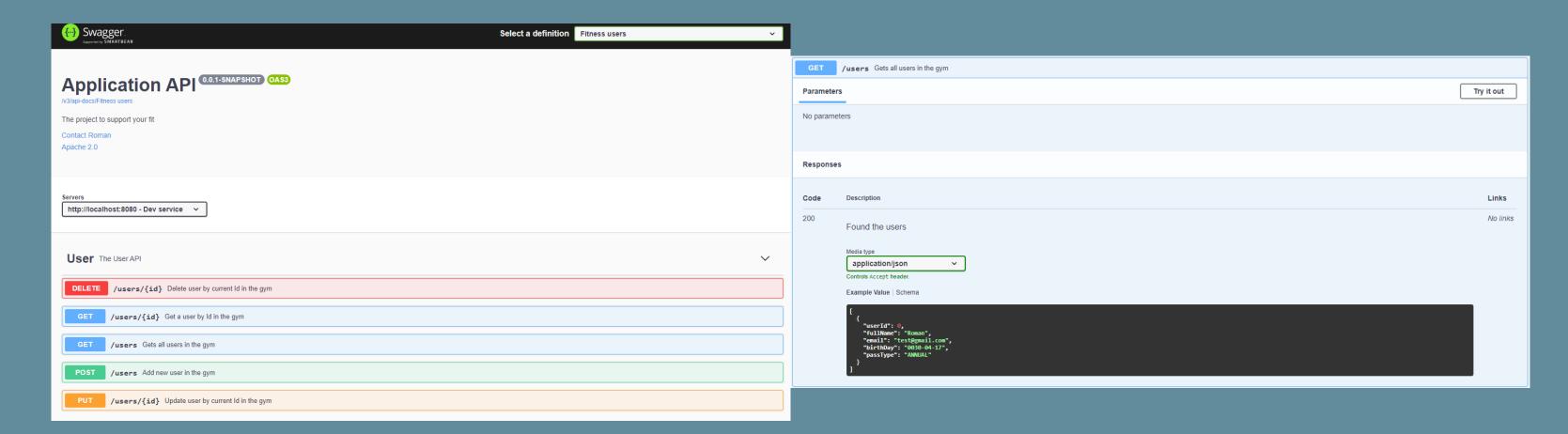
OpenAPI refers to the industry-standard specification for RESTful API design. Swagger refers to a set of SmartBear tools.

https://nordicapis.com/whats-the-difference-between-swagger-and-openapi/

## Поддержка Swagger со стороны бекенда

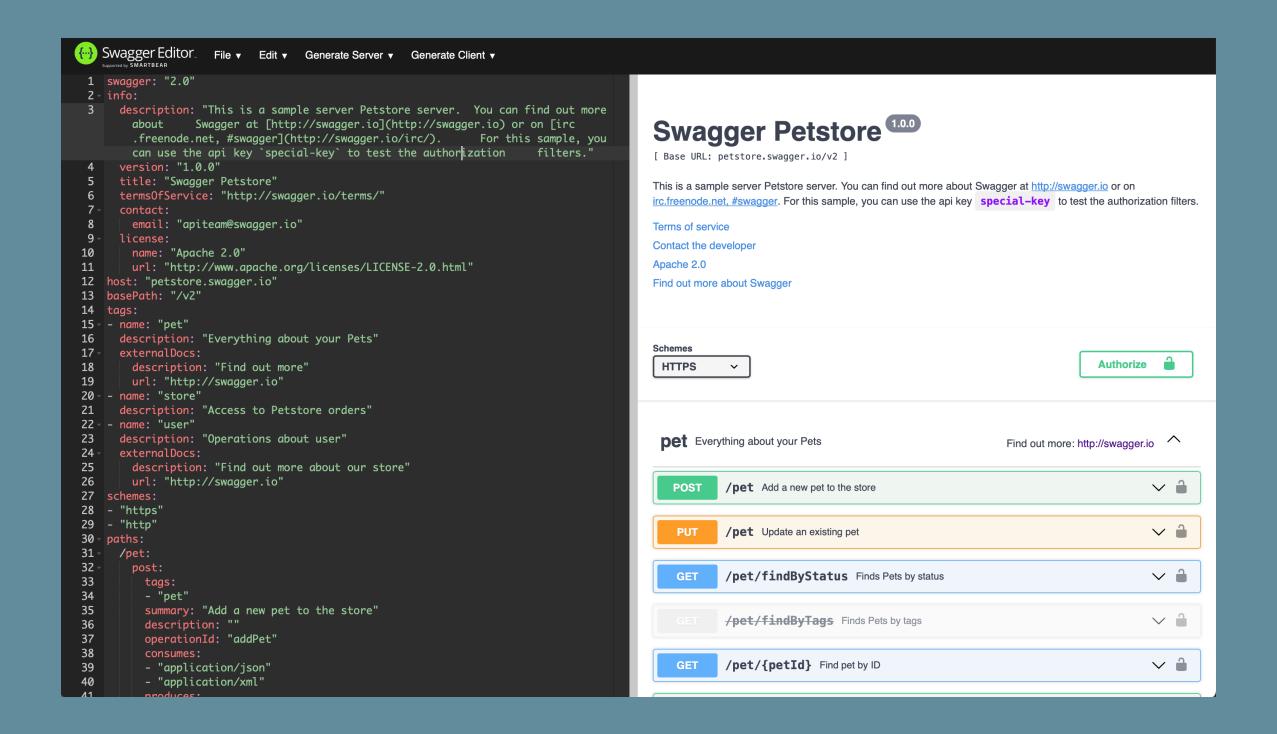
- Спецификация на основании кода (Java Annotations)
- Спецификация отдельно от кода (Swagger Editor или руками)

# Пример Swagger Ul



https://habr.com/ru/post/541592/

# Пример Swagger Editor



# Типичные проблемы и сложности на практике при разработке/поддержке API

- 🕜 Нет документации
- Версионирование
- Много бойлерплейт кода (в сериализации)
- Нужно ковыряться в нюансах работы АРІ

# Хотелка при создании сетевого слоя

Скрыть факт выполнения кода на удаленной машине

# Кодогенераторы по ОАЅ

# Правильный кодогенератор

- Прячет семантическую сложность
- Сгенерированный код не нужно трогать и саппортить
- В целом не обязательно пушить в репозиторий
- Убирает сложность работы с рантайме

# Неправильный кодогенератор

- Не прячет "кишки" реализации
- Необходимо вникать в детали работы
- Может генерировать странный/невалидный код

## Принцип работы генератора

- Парсинг входного файла во внутреннее представление
- Генерация конечного кода на базе шаблонов

# Кодогенерация REST клиента для iOS в 2014

```
"struct": "GoogleLocation",
"typedef" : {
      "lat" : "double",
      "lng" : "double"
"struct": "GoogleGeometry",
"typedef" : {
        "location" : "GoogleLocation"
```

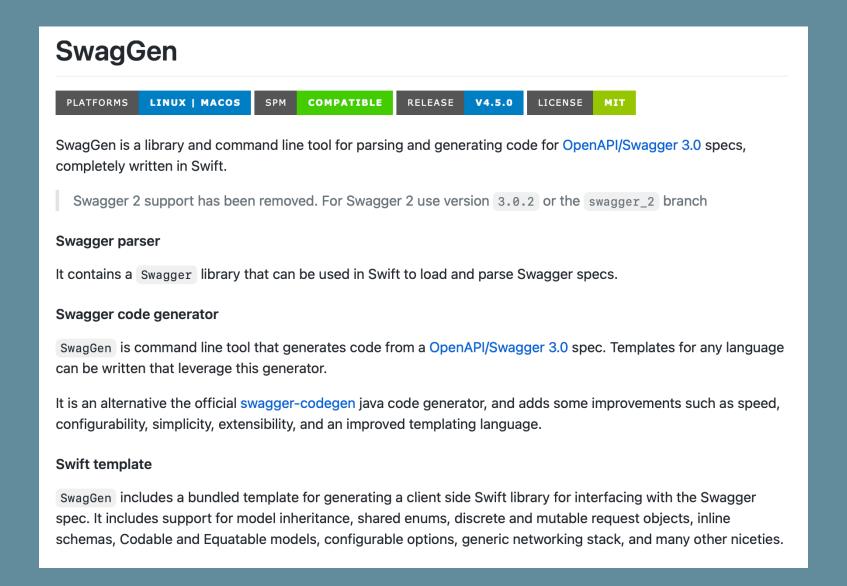
ifacegen https://habr.com/ru/company/ifree/blog/213697/

# openapi-generator

|                                    | Languages/Frameworks  |
|------------------------------------|---|
| API clients                        | ActionScript, Ada, Apex, Bash, C, C# (.net 2.0, 3.5 or later, .NET Standard 1.3 - 2.0, .NET Core 2.0, .NET 5.0. Libraries: RestSharp, HttpClient), C++ (Arduino, cpp-restsdk, Qt5, Tizen, Unreal Engine 4), Clojure, Crystal, Dart, Elixir, Elm, Eiffel, Erlang, Go, Groovy, Haskell (http-client, Servant), Java (Jersey1.x, Jersey2.x, OkHttp, Retrofit1.x, Retrofit2.x, Feign, RestTemplate, RESTEasy, Vertx, Google API Client Library for Java, Rest-assured, Spring 5 Web Client, MicroProfile Rest Client), k6, Kotlin, Lua, Nim, Node.js/JavaScript (ES5, ES6, AngularJS with Google Closure Compiler annotations, Flow types, Apollo GraphQL DataStore), Objective-C, OCaml, Perl, PHP, PowerShell, Python, R, Ruby, Rust (hyper, reqwest, rust-server), Scala (akka, http4s, scalaz, sttp, swagger-async-httpclient), Swift (2.x, 3.x, 4.x, 5.x), Typescript (AngularJS, Angular (2.x - 11.x), Aurelia, Axios, Fetch, Inversify, jQuery, Nestjs, Node, redux-query, Rxjs) |
| Server stubs                       | Ada, C# (ASP.NET Core, NancyFx), C++ (Pistache, Restbed, Qt5 QHTTPEngine), Erlang, F# (Giraffe), Go (net/http, Gin, Echo), Haskell (Servant), Java (MSF4J, Spring, Undertow, JAX-RS: CDI, CXF, Inflector, Jersey, RestEasy, Play Framework, PKMST, Vert.x), Kotlin (Spring Boot, Ktor, Vertx), PHP (Laravel, Lumen, Mezzio (fka Zend Expressive), Slim, Silex, Symfony), Python (FastAPI, Flask), NodeJS, Ruby (Sinatra, Rails5), Rust (rust-server), Scala (Akka, Finch, Lagom, Play, Scalatra)  |
| API<br>documentation<br>generators | HTML, Confluence Wiki, Asciidoc, Markdown, PlantUML   |
| Configuration files                | Apache2   |
| Others                             | GraphQL, JMeter, Ktorm, MySQL Schema, Protocol Buffer, WSDL   |

https://github.com/OpenAPITools/openapi-generator

# SwagGen



#### https://github.com/yonaskolb/SwagGen

# Постучим по клавиатуре

Swift генерация по OAS на практике

## Решенные проблемы

- Иет документации
- И Версионирование
- Иного бойлерплейт кода (в сериализации)
- Иужно ковыряться в нюансах работы АРГ
- 🔽 Единый источник правды

#### Доп возможности

# Генерация мок сервера на случай, если настоящий бекенд ещё не готов

## Текущие недостатки кодогенерации по OAS

- Странная архитектура генерируемого кода
- Генерация некомпилируемого кода
- Кодген заточен под один сервис (одну OAS)
- Ошибки в спецификациях

# Х Поддержка нового эндпойнта врукопашную

- Изучить доку (поснифать трафик)
- Задекларировать Codable модельки
- Создать новый метод сетевого слоя
- Решить нерешаемую задачу нейминга
- Написать тесты

# 

- Попросить бекендера обновить Swagger
- Запустить код ген
- PROFIT!

#### Выводы

- Человечество уже решило задачу (де)сериализации JSON
- В перспективе создание и поддержка сетевого слоя задачка без кодинга
- Кодогенераторы по OAS будут в плачевном состоянии, пока их не станут использовать и исправлять
- Пробуйте! 🖺





tg/tw @dydus0×14 tg канал https://t.me/km\_engineering