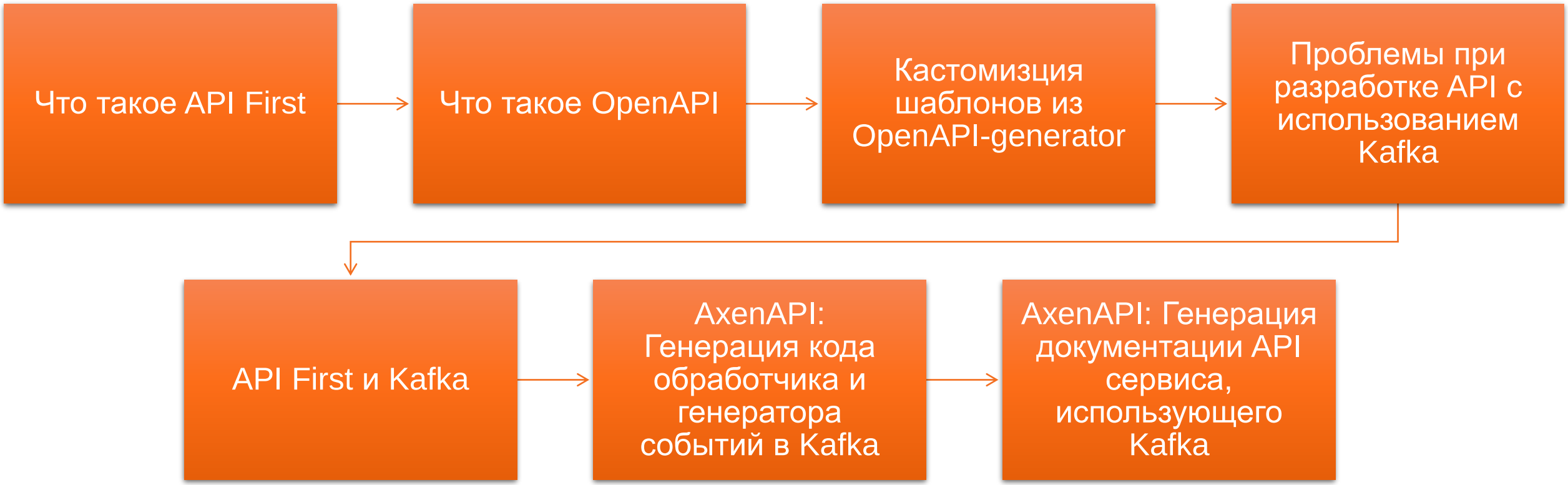
axenix
innovation

OpenAPI и как его можно применить для Kafka

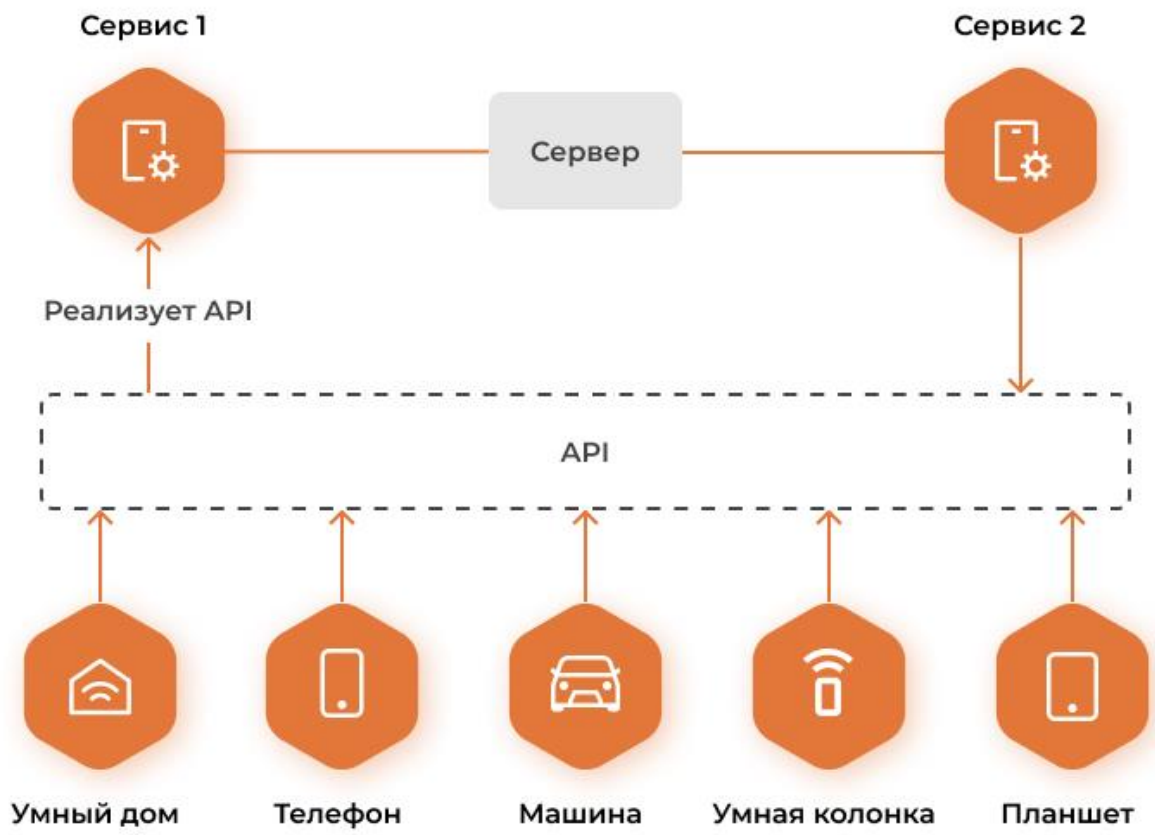
OpenAPI, API First, Kafka и OpenAPI

14/10/2023

План доклада

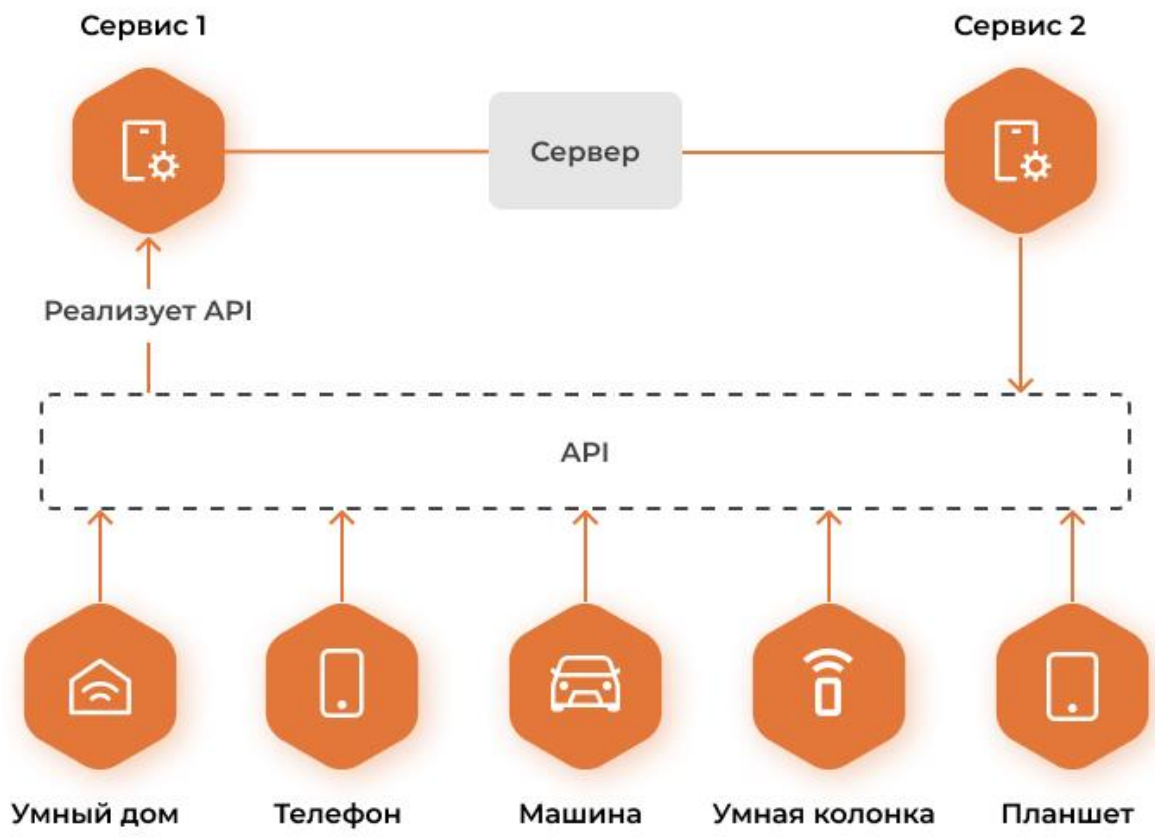


Что такое API First



API – важнейшай часть современного серверного приложения. Вы можете даже не знать количество клиентов вашего приложения.

Что такое API First



API – важнейшай часть современного серверного приложения. Вы можете даже не знать количество клиентов вашего приложения.

API First – подход разработки, при котором API (то, что будет отдано пользователю приложения) является ключевой частью вашего продукта. Сначала создается и утверждается API, и только потом начинается реализация API.

Проблема согласования протоколов

Зачем нужен подход API First

Зачем нужен подход API First

1. Ускоряет процесс разработки: многие процессы могут идти параллельно.

Зачем нужен подход API First

1. Ускоряет процесс разработки: многие процессы могут идти параллельно.
2. Уменьшается риск ошибок, связанных с отличием протоколов.

Зачем нужен подход API First

1. Ускоряет процесс разработки: многие процессы могут идти параллельно.
2. Уменьшается риск ошибок, связанных с отличием протоколов.
3. Улучшается качество проектирования API.

Зачем нужен подход API First

1. Ускоряет процесс разработки: многие процессы могут идти параллельно.
2. Уменьшается риск ошибок, связанных с отличием протоколов.
3. Улучшается качество проектирования API.
4. Появляются возможности генерации кода по спецификации API.

OpenAPI — Что это?

OpenAPI – это спецификация для описания HTTP API.

Какую информацию об API можно добавить в OpenAPI спецификацию:

1. Список доступных серверов.
2. URL, http метод.
3. Информацию о схеме авторизации.
4. Описание headers, path params, request params и их обязательность.
5. Информацию о статусах и ответах при этих статусах (разные body при разных статусах).
6. Описание полей request и response body.

```yaml
servers:
  - url: http://petstore.swagger.io/v1
paths:
  /pets:
    get:
      summary: List all pets
      operationId: listPets
      tags: <1 item>
      security:
        - ApiKeyAuth: []
      parameters:
        - name: limit
          in: query # "path", "query", "cookie", "header"
          description: How many items to return at one time (max 100)
          required: false
          schema:
            type: integer
            format: int32
      responses:
        '200':
          description: A paged array of pets
          headers:
            x-next:
              description: A link to the next page of responses
              schema:
                type: string
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/Pets"
```

OpenAPI — Что это?

OpenAPI – это спецификация для описания HTTP API.

Какую информацию об API можно добавить в OpenAPI спецификацию:

1. Список доступных серверов.
2. URL, http метод.
3. Информацию о схеме авторизации.
4. Описание headers, path params, request params и их обязательность.
5. Информацию о статусах и ответах при этих статусах (разные body при разных статусах).
6. Описание полей request и response body.

```yaml
servers:
  - url: http://petstore.swagger.io/v1
paths:
  /pets:
    get:
      summary: List all pets
      operationId: listPets
      tags: <1 item>
      security:
        - ApiKeyAuth: []
      parameters:
        - name: limit
          in: query # "path", "query", "cookie", "header"
          description: How many items to return at one time (max 100)
          required: false
          schema:
            type: integer
            format: int32
      responses:
        '200':
          description: A paged array of pets
          headers:
            x-next:
              description: A link to the next page of responses
              schema:
                type: string
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/Pets"
```

OpenAPI — Что это?

OpenAPI – это спецификация для описания HTTP API.

Какую информацию об API можно добавить в OpenAPI спецификацию:

1. Список доступных серверов.
2. URL, http метод.
3. Информацию о схеме авторизации.
4. Описание headers, path params, request params и их обязательность.
5. Информацию о статусах и ответах при этих статусах (разные body при разных статусах).
6. Описание полей request и response body.

```yaml
servers:
  - url: http://petstore.swagger.io/v1
paths:
  /pets:
    get:
      summary: List all pets
      operationId: listPets
      tags: <1 item>
      security:
        - ApiKeyAuth: []
      parameters:
        - name: limit
          in: query # "path", "query", "cookie", "header"
          description: How many items to return at one time (max 100)
          required: false
          schema:
            type: integer
            format: int32
      responses:
        '200':
          description: A paged array of pets
          headers:
            x-next:
              description: A link to the next page of responses
              schema:
                type: string
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/Pets"
```

OpenAPI — Что это?

OpenAPI – это спецификация для описания HTTP API.

Какую информацию об API можно добавить в OpenAPI спецификацию:

1. Список доступных серверов.
2. URL, http метод.
3. Информацию о схеме авторизации.
4. Описание headers, path params, request params и их обязательность.
5. Информацию о статусах и ответах при этих статусах (разные body при разных статусах).
6. Описание полей request и response body.

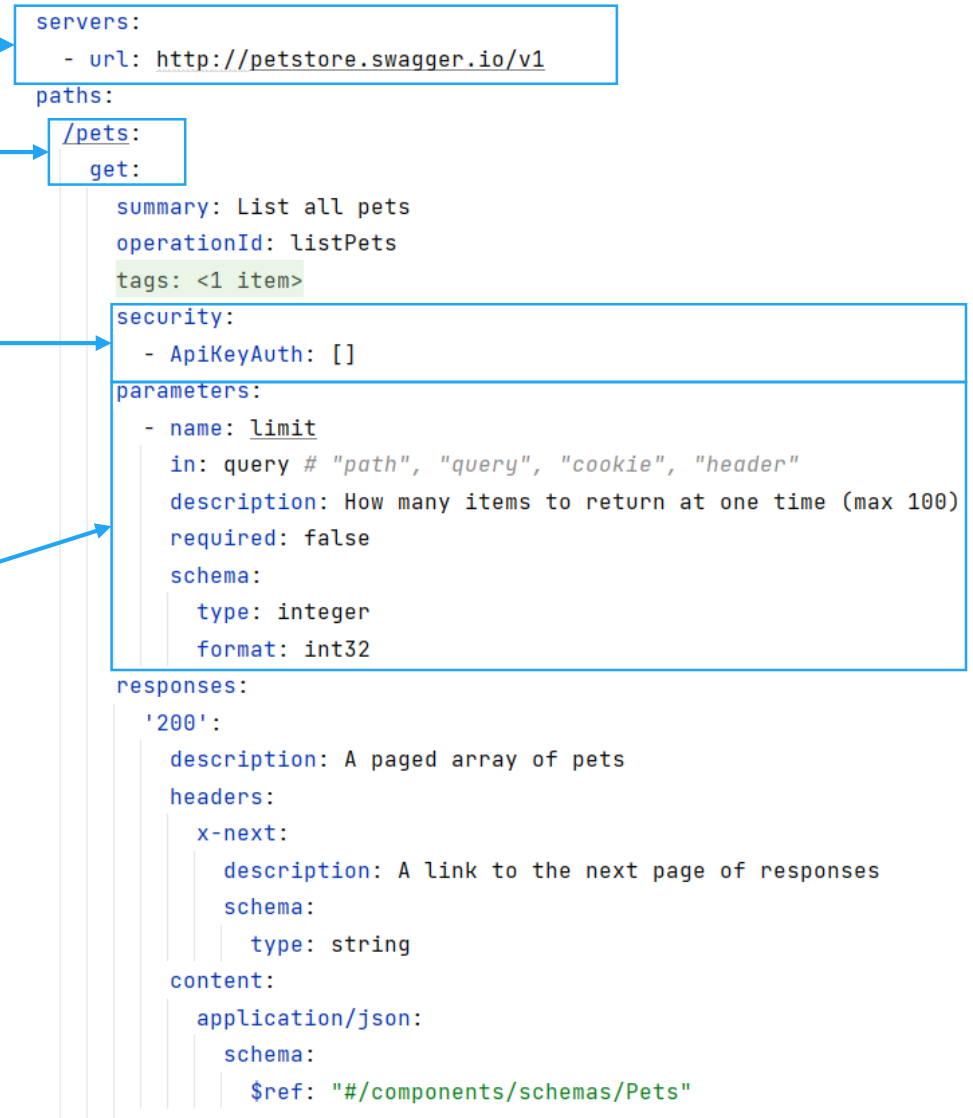
```yaml
servers:
  - url: http://petstore.swagger.io/v1
paths:
  /pets:
    get:
      summary: List all pets
      operationId: listPets
      tags: <1 item>
      security:
        - ApiKeyAuth: []
      parameters:
        - name: limit
          in: query # "path", "query", "cookie", "header"
          description: How many items to return at one time (max 100)
          required: false
          schema:
            type: integer
            format: int32
      responses:
        '200':
          description: A paged array of pets
          headers:
            x-next:
              description: A link to the next page of responses
              schema:
                type: string
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/Pets"
```

OpenAPI — Что это?

OpenAPI – это спецификация для описания HTTP API.

Какую информацию об API можно добавить в OpenAPI спецификацию:

1. Список доступных серверов.
2. URL, http метод.
3. Информацию о схеме авторизации.
4. Описание headers, path params, request params и их обязательность.
5. Информацию о статусах и ответах при этих статусах (разные body при разных статусах).
6. Описание полей request и response body.



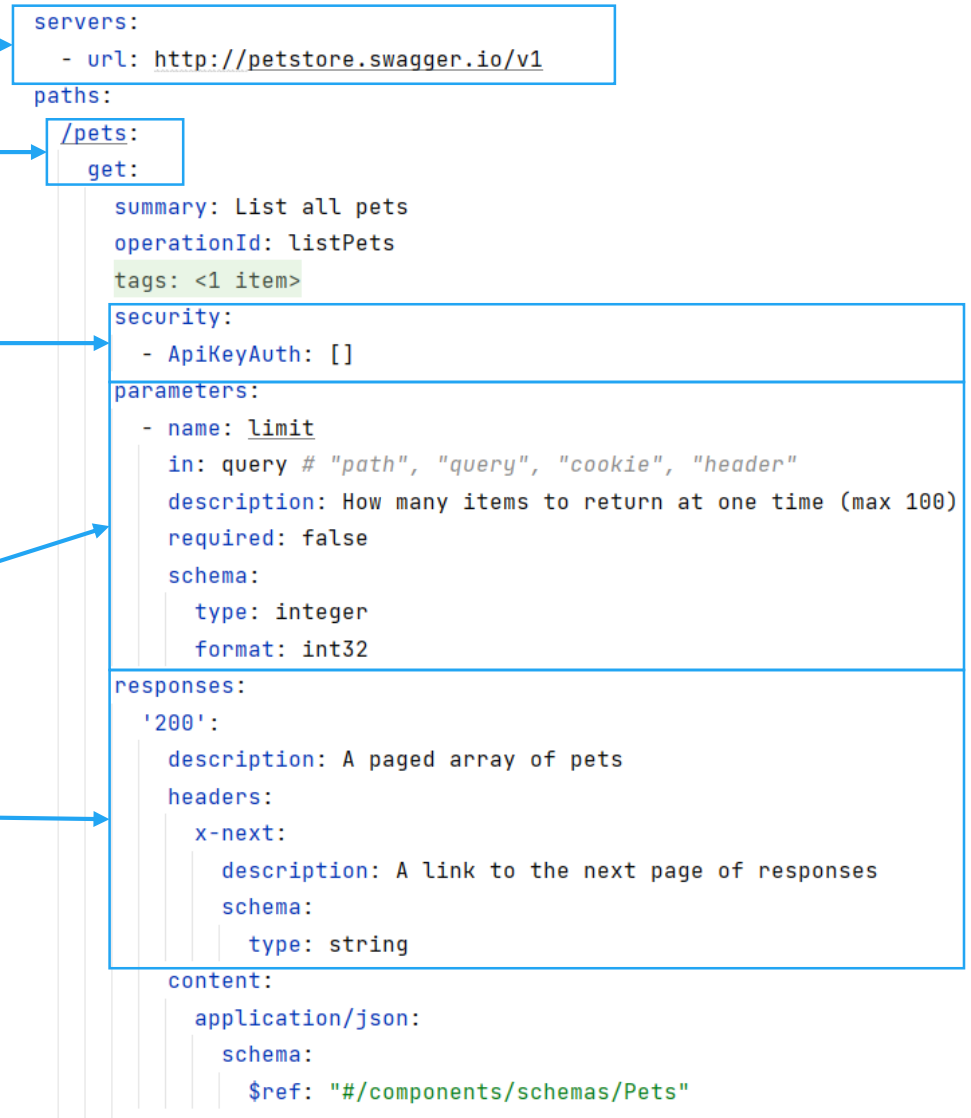
```yaml
servers:
  - url: http://petstore.swagger.io/v1
paths:
  /pets:
    get:
      summary: List all pets
      operationId: listPets
      tags: <1 item>
      security:
        - ApiKeyAuth: []
      parameters:
        - name: limit
          in: query # "path", "query", "cookie", "header"
          description: How many items to return at one time (max 100)
          required: false
          schema:
            type: integer
            format: int32
      responses:
        '200':
          description: A paged array of pets
          headers:
            x-next:
              description: A link to the next page of responses
              schema:
                type: string
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/Pets"
```

OpenAPI — Что это?

OpenAPI – это спецификация для описания HTTP API.

Какую информацию об API можно добавить в OpenAPI спецификацию:

1. Список доступных серверов.
2. URL, http метод.
3. Информацию о схеме авторизации.
4. Описание headers, path params, request params и их обязательность.
5. Информацию о статусах и ответах при этих статусах (разные body при разных статусах).
6. Описание полей request и response body.



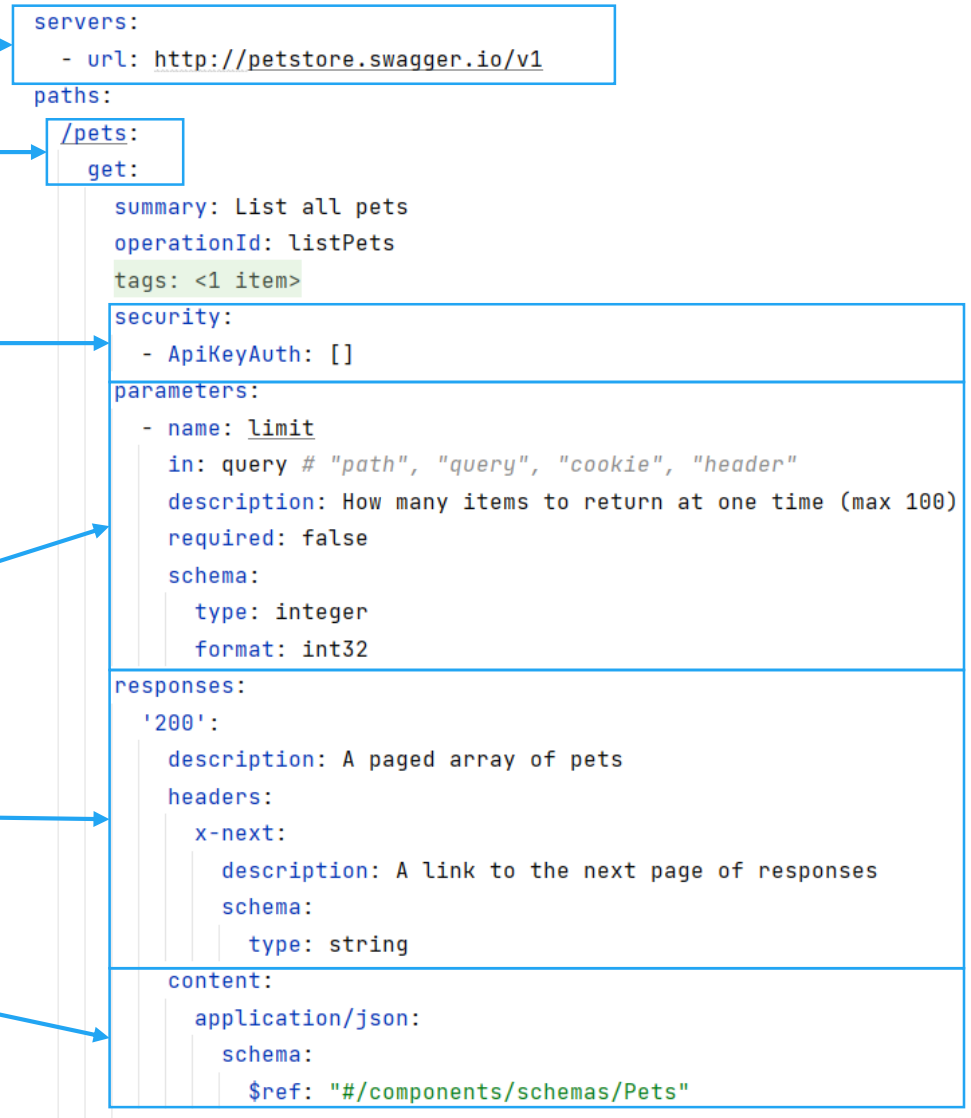
```yaml
servers:
  - url: http://petstore.swagger.io/v1
paths:
  /pets:
    get:
      summary: List all pets
      operationId: listPets
      tags: <1 item>
      security:
        - ApiKeyAuth: []
      parameters:
        - name: limit
          in: query # "path", "query", "cookie", "header"
          description: How many items to return at one time (max 100)
          required: false
          schema:
            type: integer
            format: int32
      responses:
        '200':
          description: A paged array of pets
          headers:
            x-next:
              description: A link to the next page of responses
              schema:
                type: string
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/Pets"
```

OpenAPI — Что это?

OpenAPI – это спецификация для описания HTTP API.

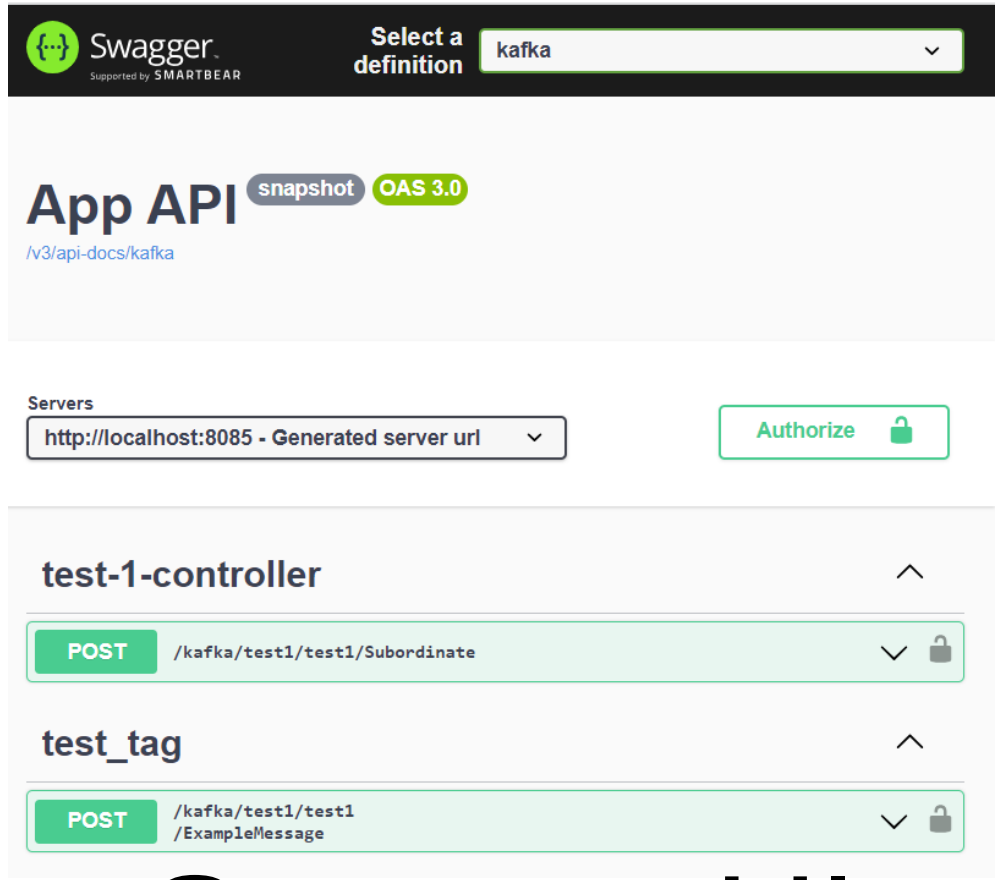
Какую информацию об API можно добавить в OpenAPI спецификацию:

1. Список доступных серверов.
2. URL, http метод.
3. Информацию о схеме авторизации.
4. Описание headers, path params, request params и их обязательность.
5. Информацию о статусах и ответах при этих статусах (разные body при разных статусах).
6. Описание полей request и response body.



OpenAPI — Что это?

OpenAPI



SwaggerUI

Как начать работать по API First подходу

В начале разработки

1. Разработать (или доработать) спецификацию в формате OpenAPI.
2. Генерировать по ней код.

OpenAPI Generator

В процессе разработки

0. Инициировать первую версию спецификации в формате OpenAPI

SpringDoc

OpenAPI 3
+
Spring Boot

Swagger™
Supported by SMARTBEAR

OpenAPI Generator

```yaml
servers:
  - url: http://petstore.swagger.io/v1
paths:
  /pets:
    get:
      summary: List all pets
      operationId: listPets
      tags: <1 item>
      security:
        - ApiKeyAuth: []
      parameters:
        - name: limit
          in: query # "path", "query", "cookie", "header"
          description: How many items to return at one time (max 100)
          required: false
          schema:
            type: integer
            format: int32
      responses:
        '200':
          description: A paged array of pets
          headers:
            x-next:
              description: A link to the next page of responses
              schema:
                type: string
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/Pets"
```

OpenAPI Specification

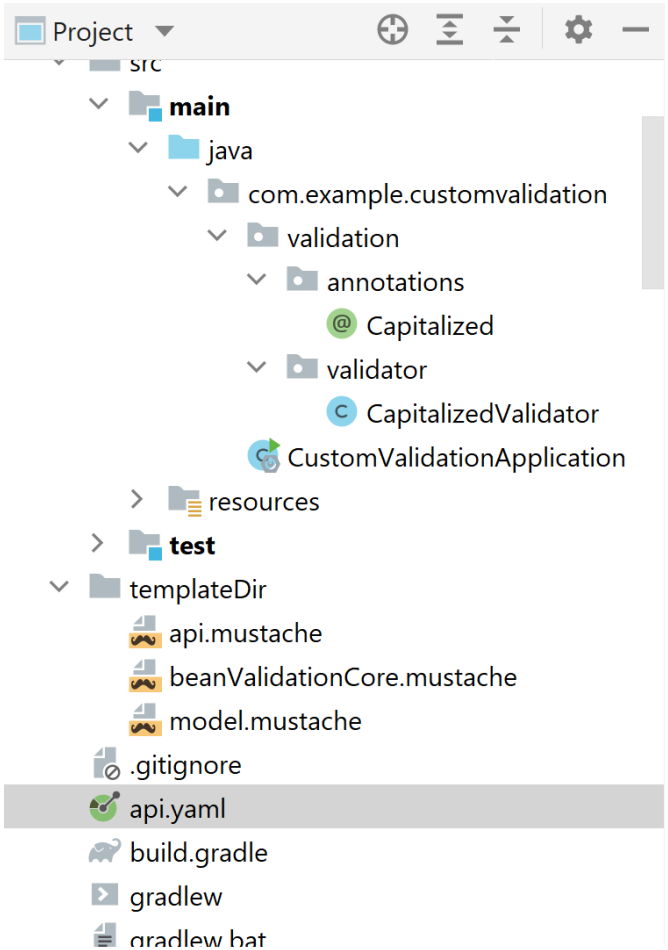
OpenAPI Generator: генерация кода по спецификации API

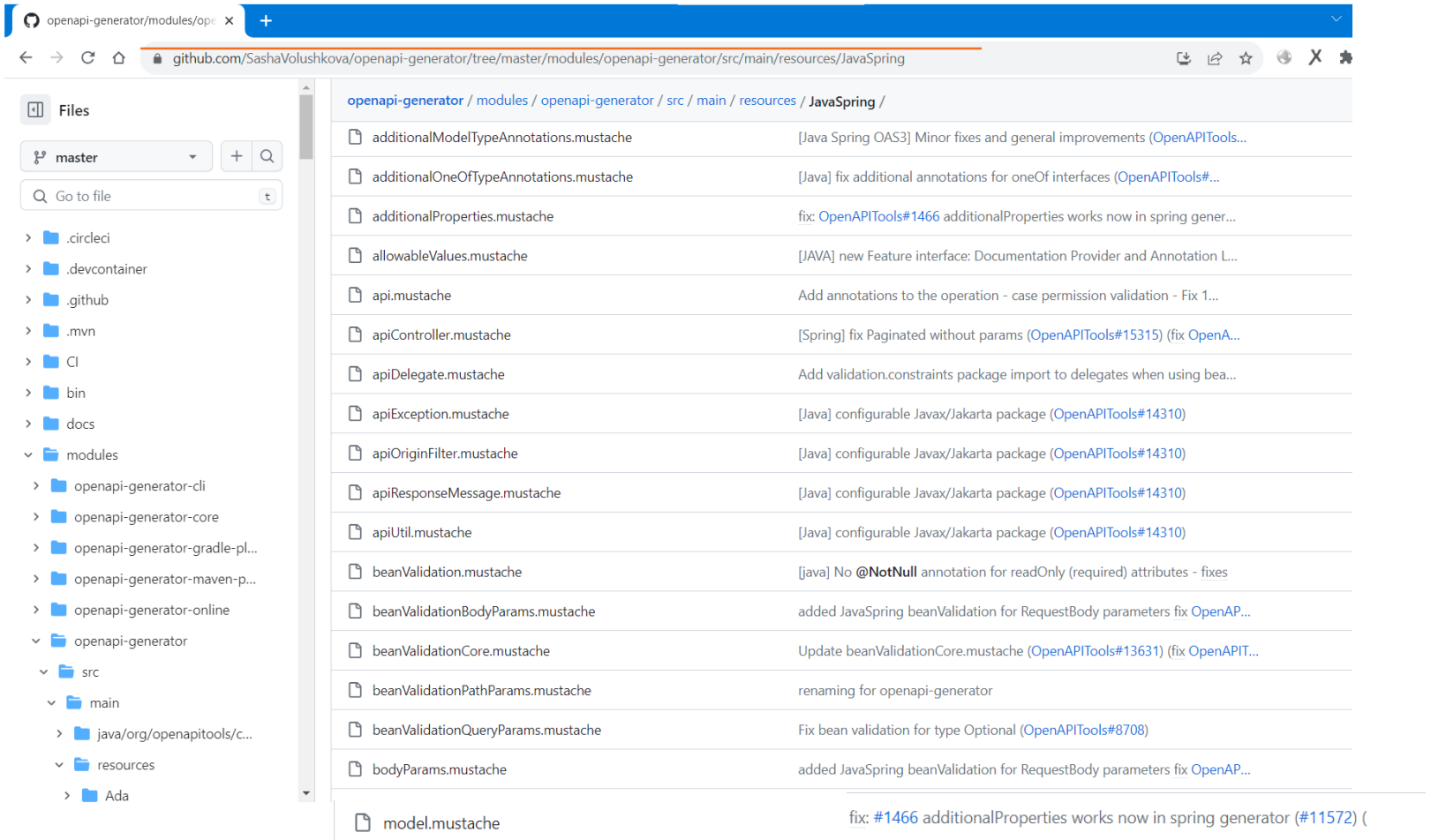
Application code

Подробнее про API First

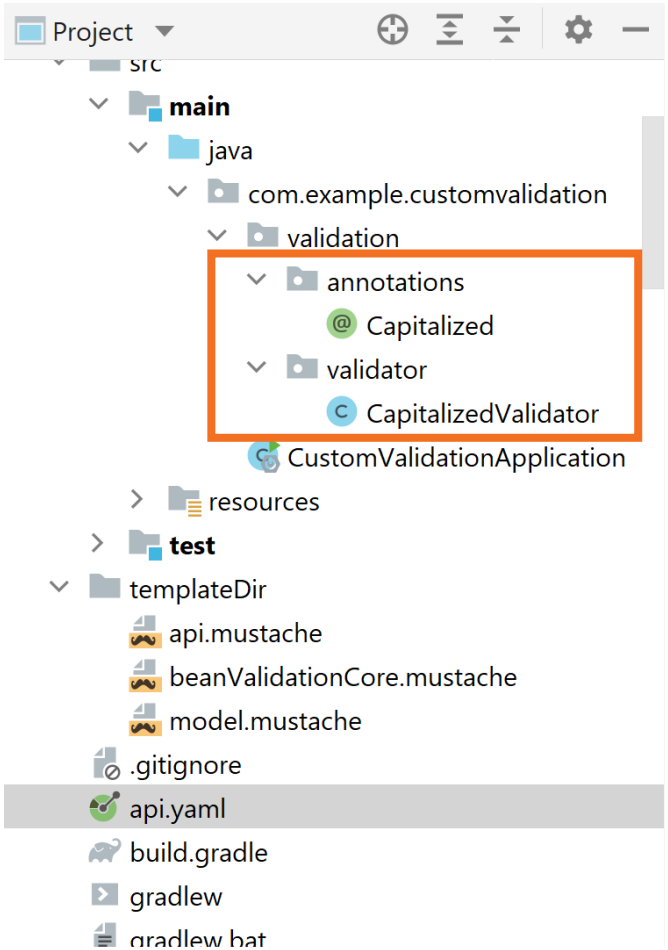
Подробнее про API First рассказано в статье https://habr.com/ru/company/axenix/blog/694340/ .

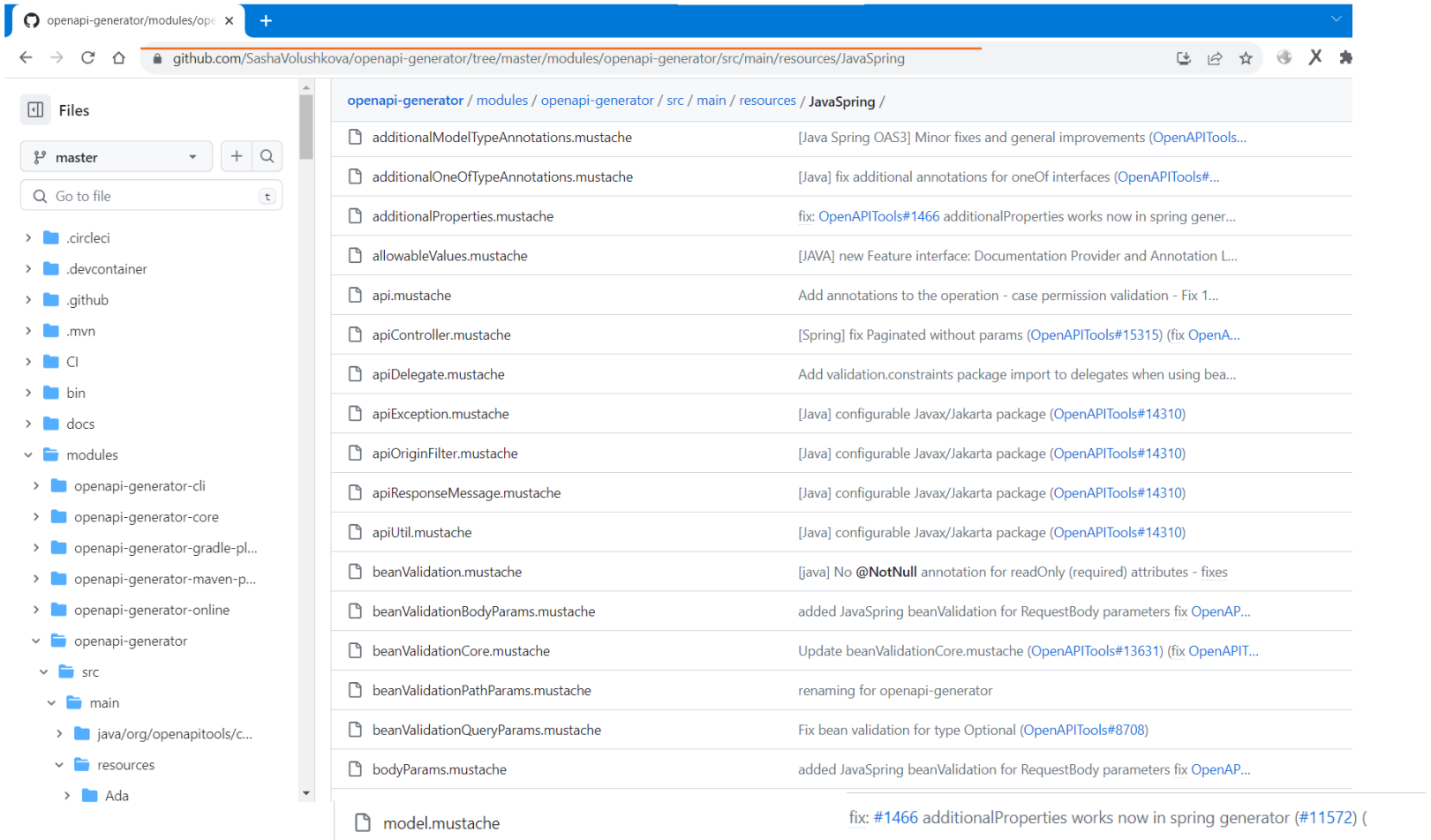
OpenAPI Generator. Кастомизация



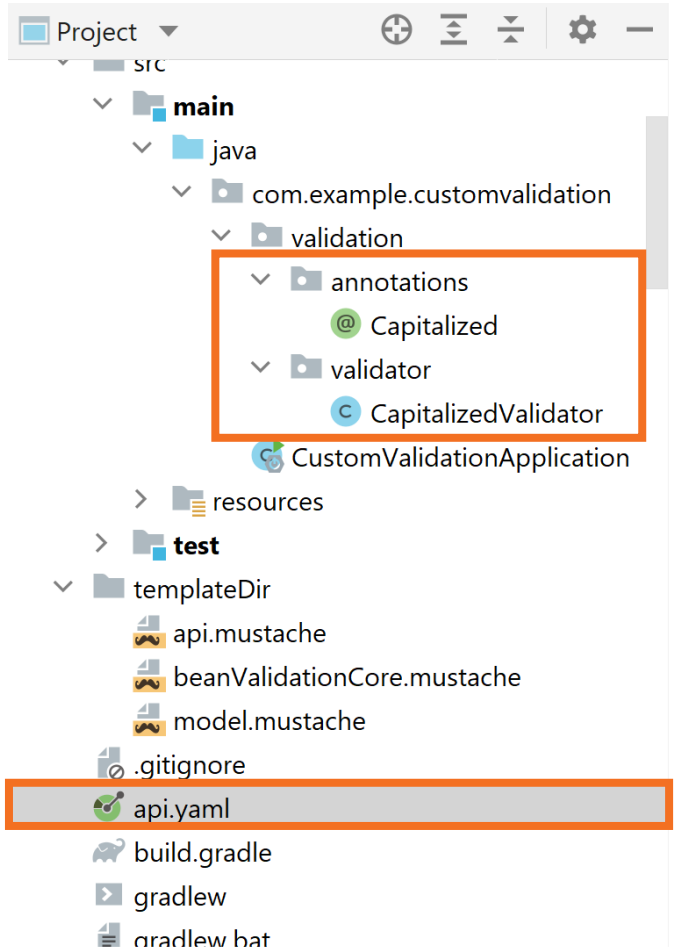


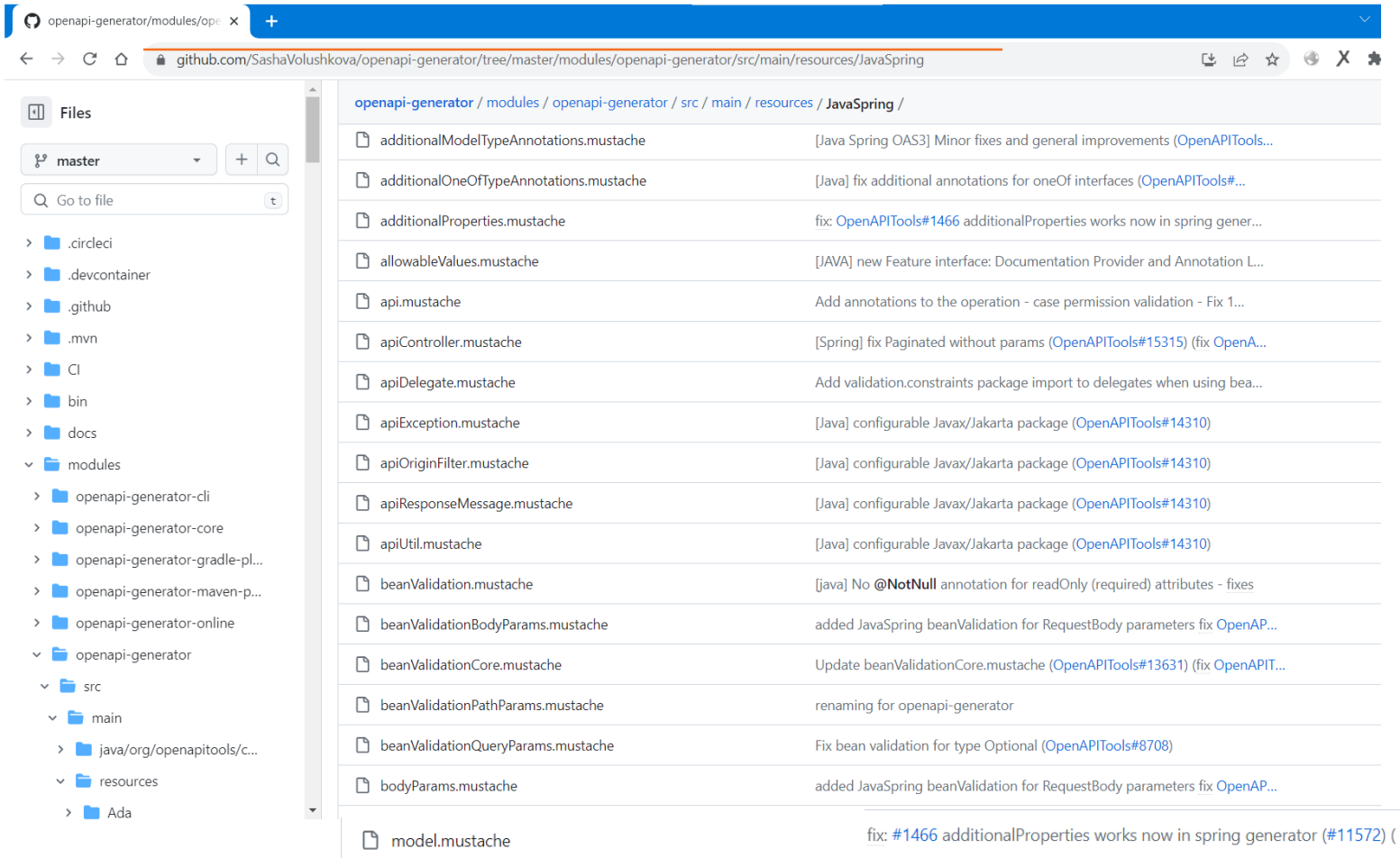
OpenAPI Generator. Кастомизация



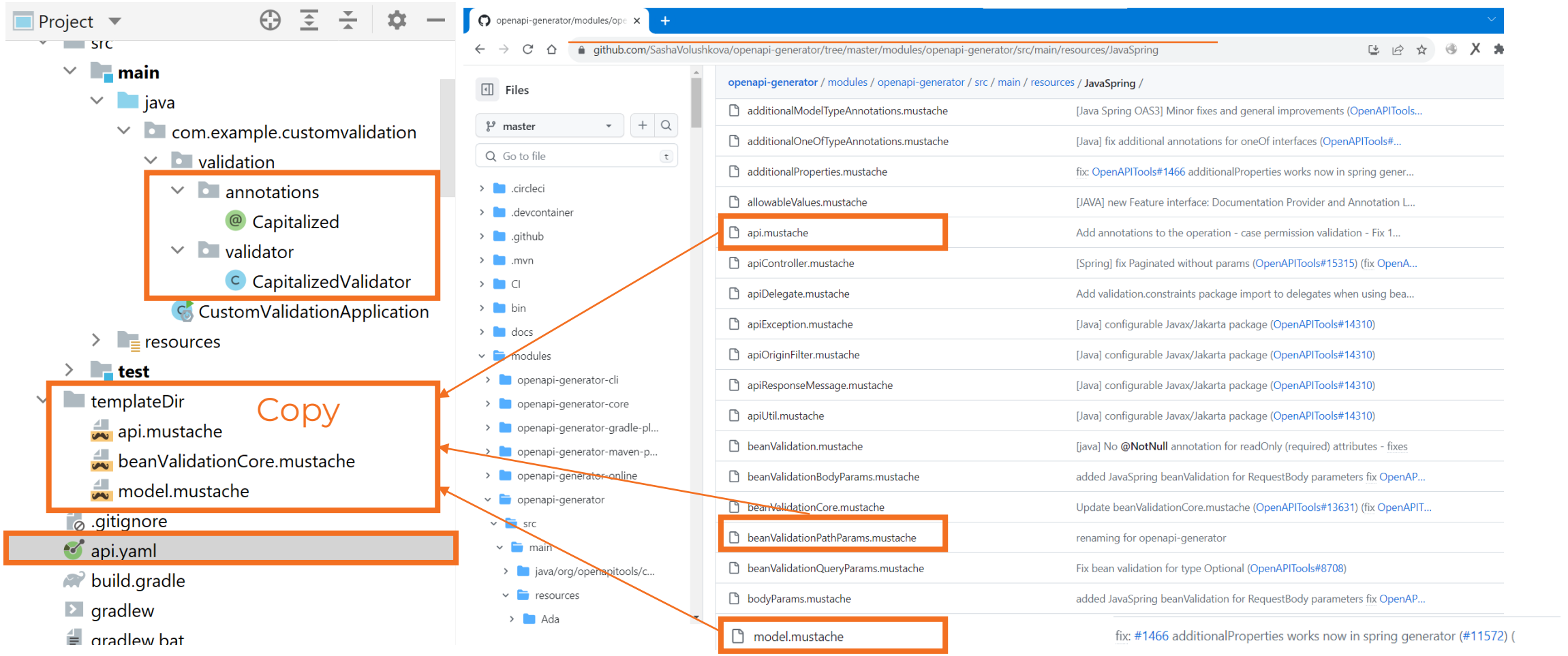


OpenAPI Generator. Кастомизация





OpenAPI Generator. Кастомизация

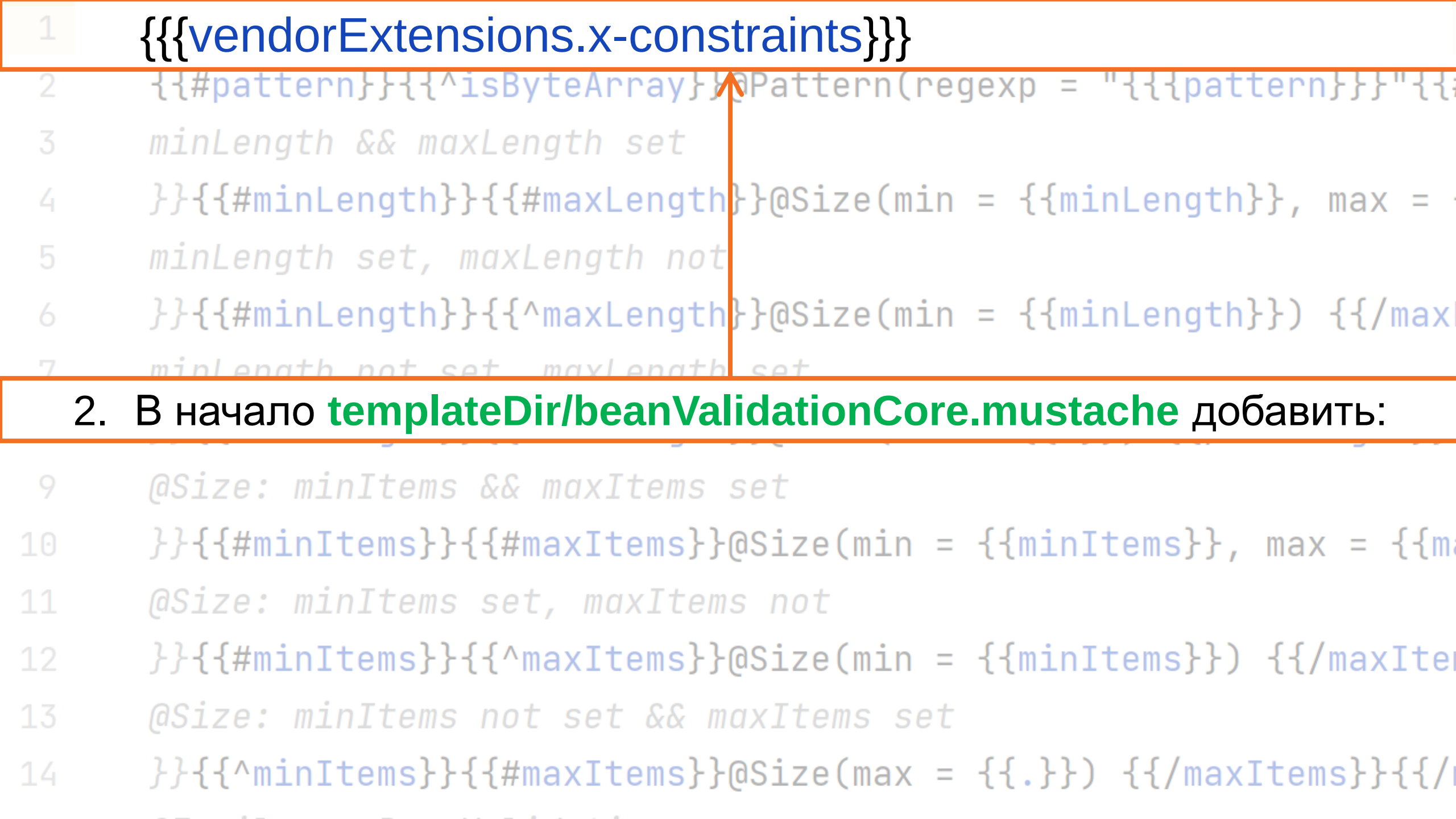


```
components:
  schemas:
    Pet:
      required:
        - id
        - name
      properties:
        id:
          minimum: 5
          maximum: 10
        name:
          type: string
          x-constraints: "@Capitalized(required = true)"
        tag:
          type: string
    Pets:
      type: array
```

1. Добавить в спецификацию (api.yaml) property:

{{{vendorExtensions.x-constraints}}}

2. В начало **templateDir/beanValidationCore.mustache** добавить:



```
/**
 * NOTE: This class is auto generated by OpenAPI Generator (https://openapi-generator.tech) ({{{ger
 * https://openapi-generator.tech
 * Do not edit the class manually.
 */
package {{package}};

{{#imports}}import {{import}};
{{/imports}}
import com.example.customvalidation.validation.annotations.*;
{{#swagger2AnnotationLibrary}}
import io.swagger.v3.oas.annotations.ExternalDocumentation;
```

3. Добавить в файл **templateDir/api.mustache** import к package, где находится ваша аннотация валидации

```
1   package {{package}};
2
3   import java.net.URI;
4   import java.util.Objects;
5   {{#imports}}import {{import}};
6   {{/imports}}
    import com.example.customvalidation.validation.annotations.*;
8   {{#openApiNullable}}
9   import org.openapitools.jackson.nullable.JsonNullable;
10  {{/openApiNullable}}
```

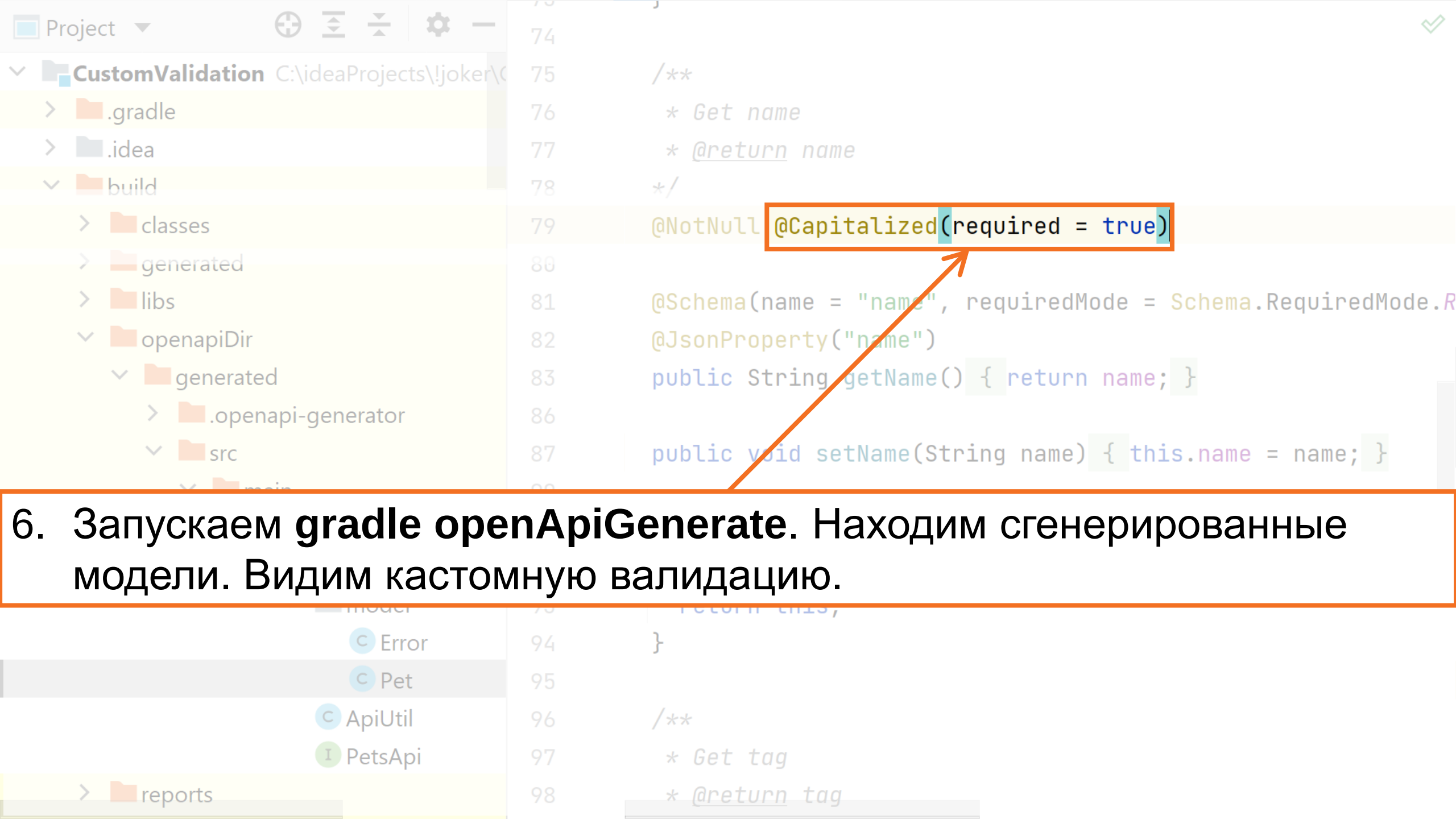
4. Добавить в файл **templateDir/model.mustache** import к package, где находится ваша аннотация валидации

```
openApiGenerate {
    generatorName = "spring"
    outputDir = getBuildDir().absolutePath + "/openapiDir/generated"
    inputSpec = getProjectDir().absolutePath + "/api.yaml"
    globalProperties = [
            apis: "",
            models: "",
            supportingFiles: 'ApiUtil.java'
    ]
```

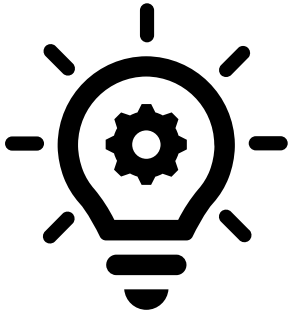
5. Добавить путь к шаблону в этап **openApiGenerate в build.gradle**:

```
    templateDir = getProjectDir().absolutePath + "/templateDir/"
    configOptions = [
            useSpringBoot3: "true",
            dateLibrary: "java8",
            interfaceOnly: "true",
            skipDefaultInterface: "false",
            openApiNullable: "false",
            generateSupportingFiles: "true"
```

openApiGenerate{}

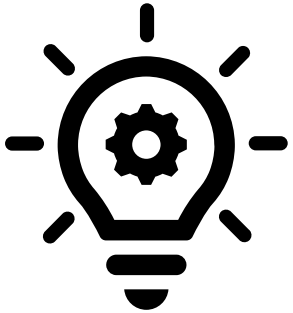

Project
CustomValidation C:\ideaProjects\!joker\C
.gradle
.idea
build
classes
generated
libs
openapiDir
generated
.openapi-generator
src
Error
Pet
ApiUtil
PetsApi
reports
@NotNull @Capitalized(required = true)
@Schema(name = "name", requiredMode = Schema.RequiredMode.R
@JsonProperty("name")
public String getName() { return name; }
public void setName(String name) { this.name = name; }
6. Запускаем **gradle openApiGenerate**. Находим сгенерированные модели. Видим кастомную валидацию.

OpenAPI Generator. Кастомизация. Выводы



1. Наличие своих правил валидации не является препятствием для использования генерации кода.

OpenAPI Generator. Кастомизация. Выводы

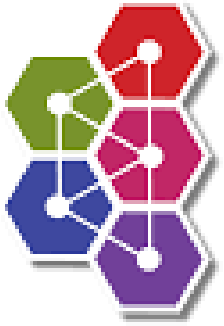


1. Наличие своих правил валидации не является препятствием для использования генерации кода.
2. OpenAPI Generator – гибкий инструмент. Трансляцию из OpenAPI в Java можно сделать в любом удобном для вас виде.

Не HTTP(s) единым...

kafka

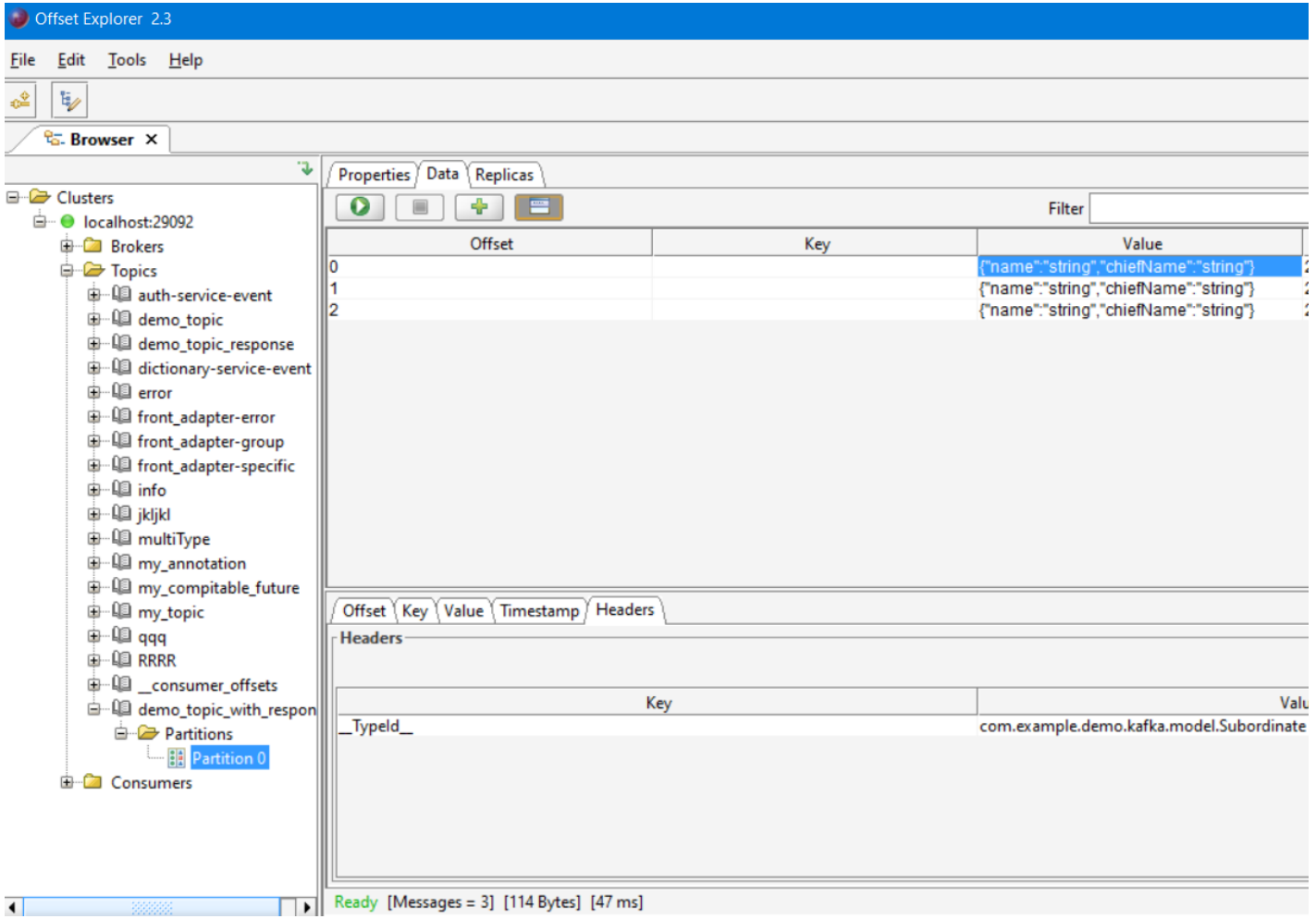
RabbitMQ



APACHE ACTIVEMQ®

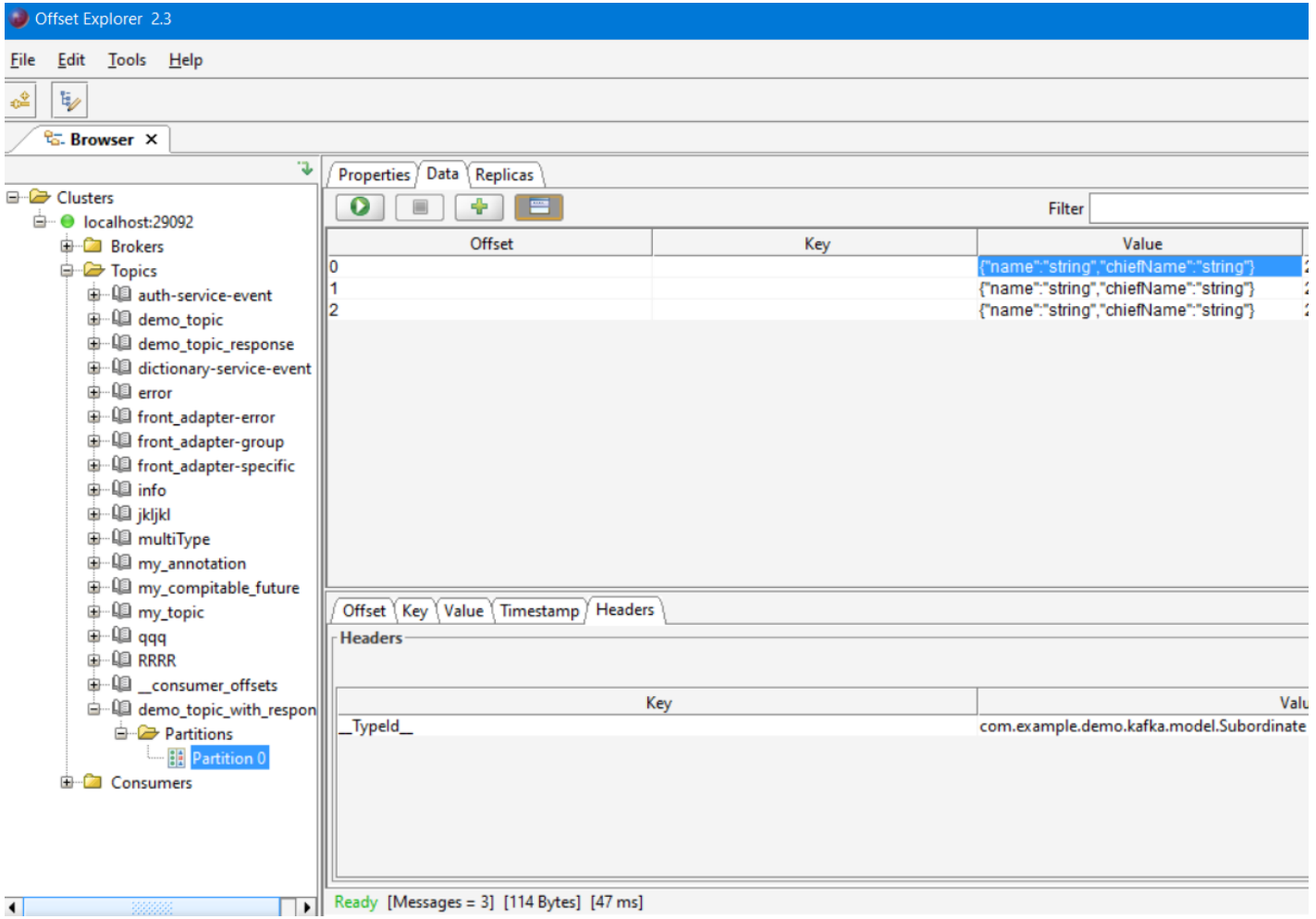
Проблемы при работе с брокерами сообщений

1. Отсутствие единой документации endpoints.



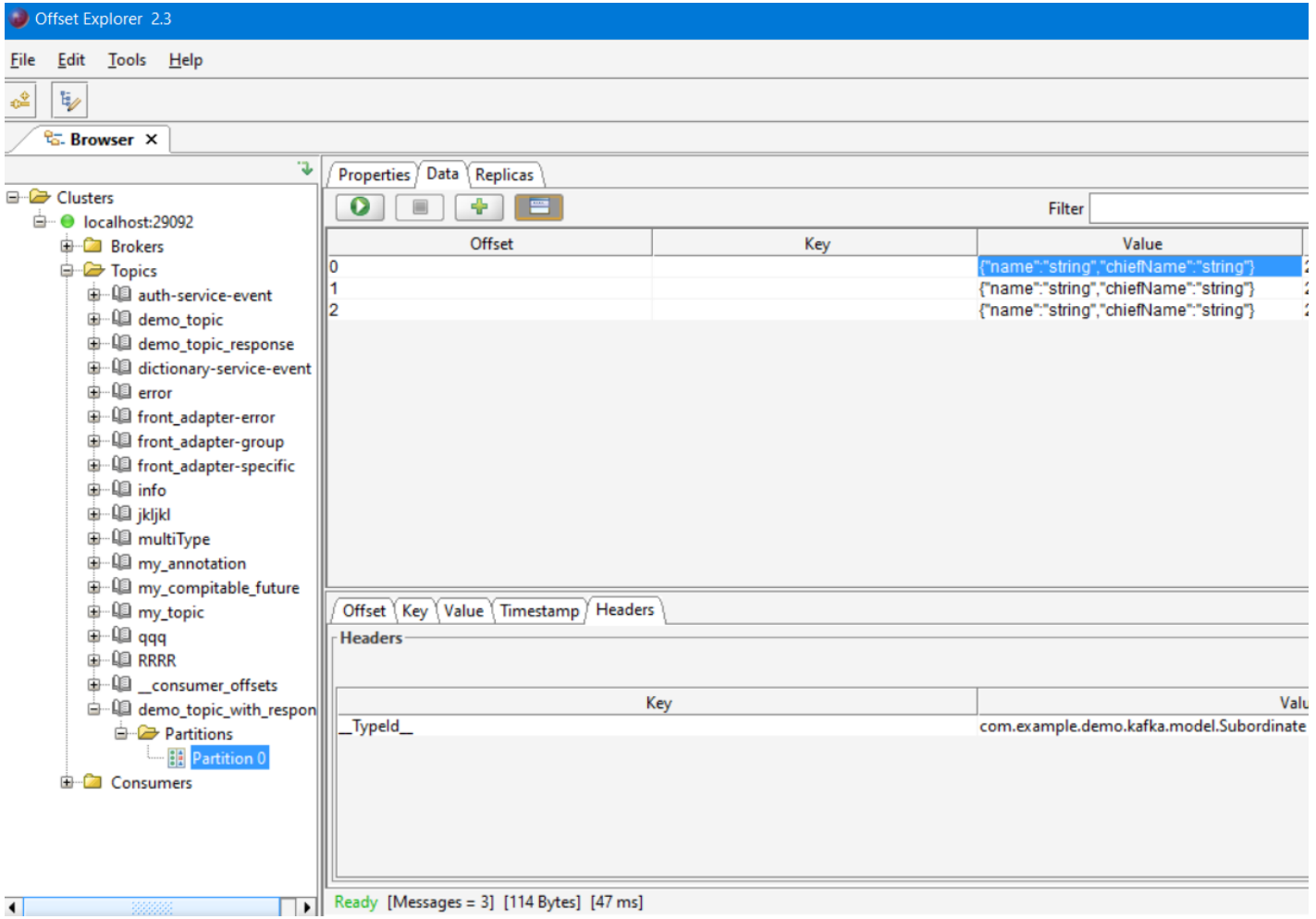
Проблемы при работе с брокерами сообщений

1. Отсутствие единой документации endpoints.
2. Отсутствие версионирования документации.



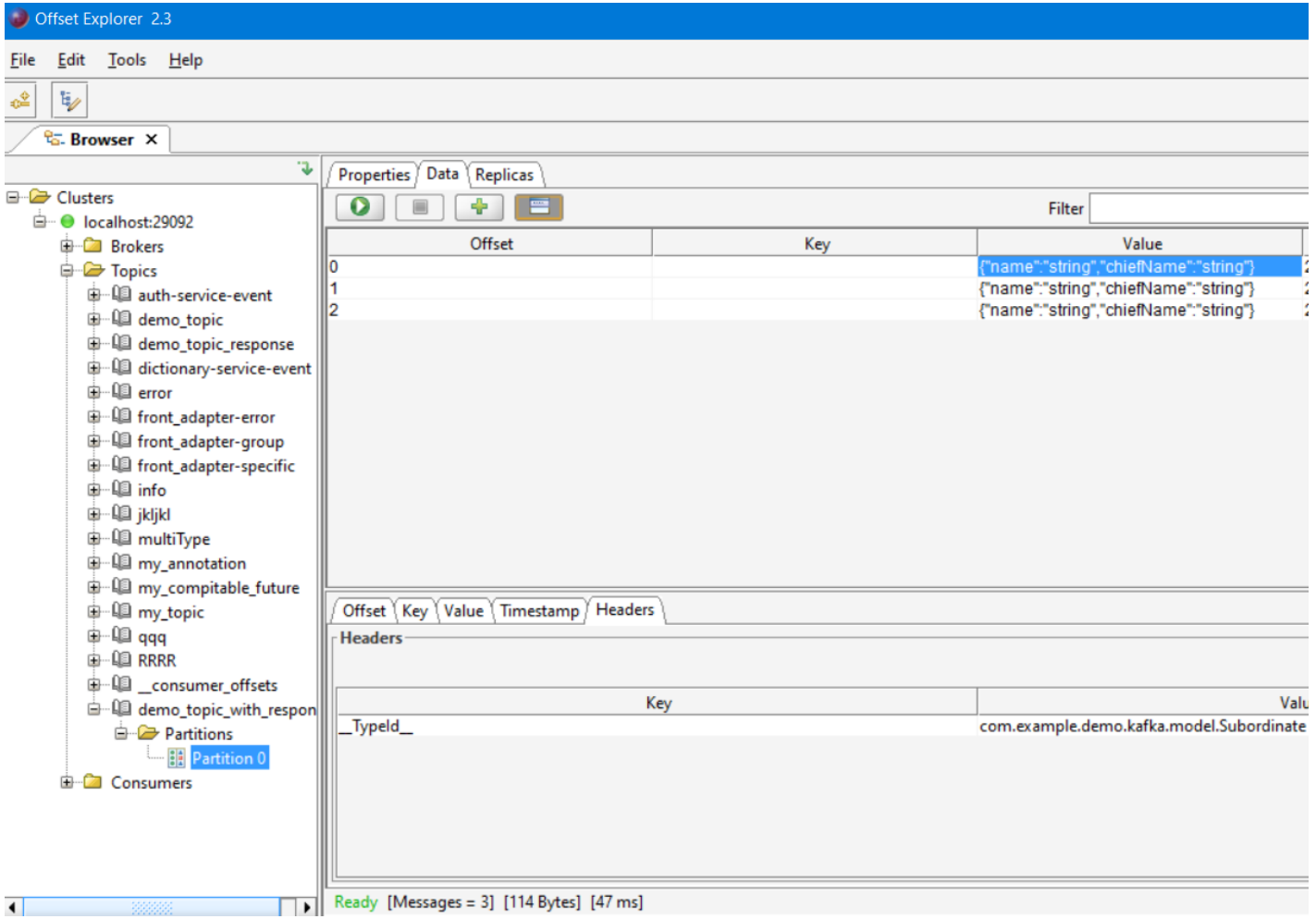
Проблемы при работе с брокерами сообщений

1. Отсутствие единой документации endpoints.
2. Отсутствие версионирования документации.
3. **Большие сложности для использования подхода API First.**



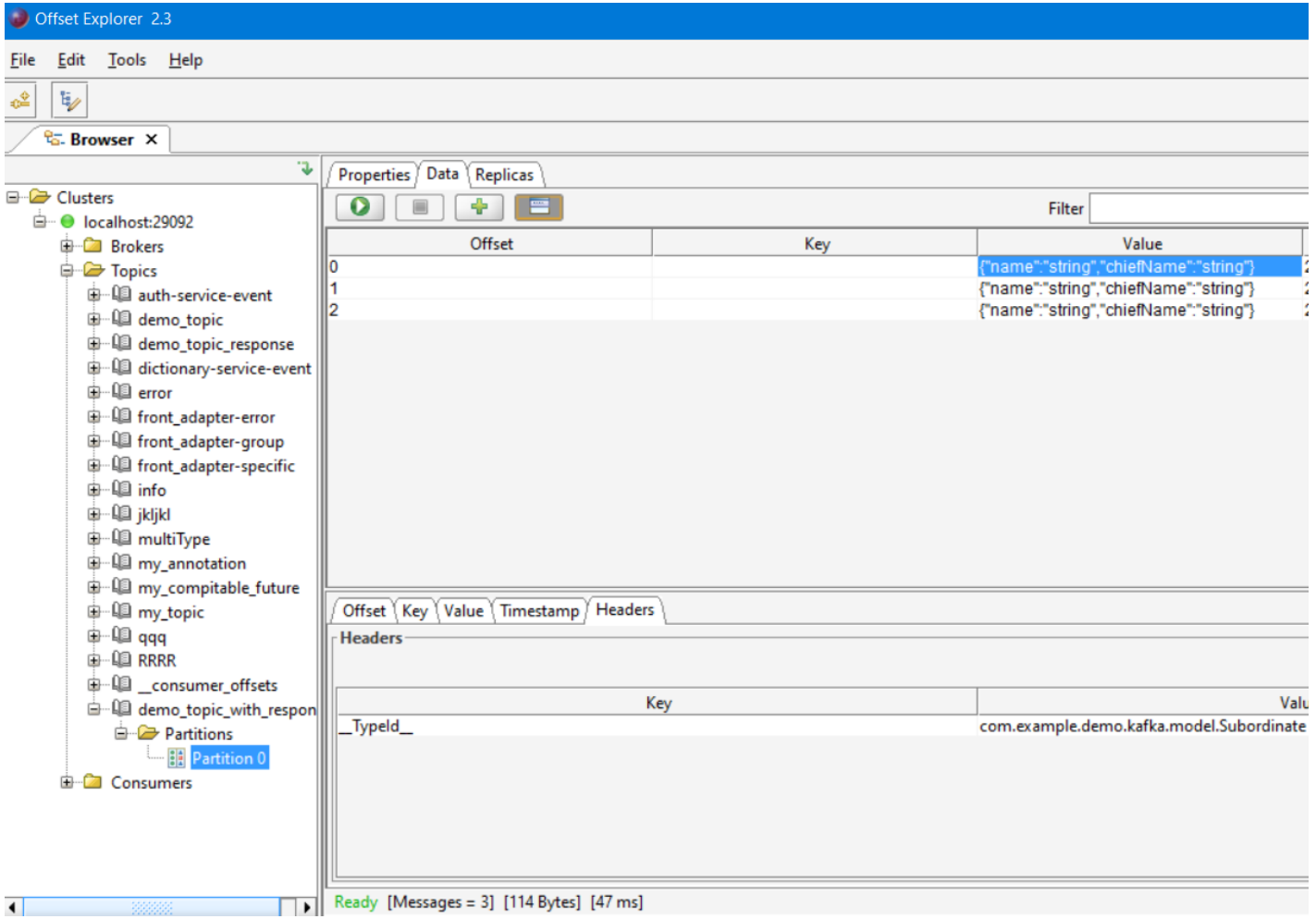
Проблемы при работе с брокерами сообщений

1. Отсутствие единой документации endpoints.
2. Отсутствие версионирования документации.
3. **Большие сложности для использования подхода API First.**
4. Достаточно сложный (и/или платный) UI по отправке сообщений – яркий пример Offset Explorer для Kafka.



Проблемы при работе с брокерами сообщений

1. Отсутствие единой документации endpoints.
2. Отсутствие версионирования документации.
3. **Большие сложности для использования подхода API First.**
4. Достаточно сложный (и/или платный) UI по отправке сообщений – яркий пример Offset Explorer для Kafka.
5. Отсутствие примеров входных данных.



API First и Kafka. Готовые решения (нам не подошли ☹)

В начале разработки

1. Разрабатывать (или дорабатывать) спецификацию в формате OpenAPI.
2. Генерировать по ней код.

Инфраструктура по работе со спецификацией в формате AsyncAPI

AsyncAPI

В процессе разработки

0. Инициировать первую версию спецификации в формате OpenAPI

Генерация документации в формате AsyncAPI, подключение SwaggerUI

SpringWolf

Почему нам не подходит AsyncAPI

AsyncAPI

SpringWolf

Почему нам не подходит AsyncAPI

AsyncAPI

SpringWolf

1. Инфраструктура вокруг AsyncAPI написана на JS.

Почему нам не подходит AsyncAPI

AsyncAPI

SpringWolf

1. Инфраструктура вокруг AsyncAPI написана на JS.
2. Это еще один DSL.

Почему нам не подходит AsyncAPI

AsyncAPI

SpringWolf

1. Инфраструктура вокруг AsyncAPI написана на JS.
2. Это еще один DSL.
3. Генератор создает приложение целиком.

Почему нам не подходит AsyncAPI

AsyncAPI

1. Инфраструктура вокруг AsyncAPI написана на JS.
2. Это еще один DSL.
3. Генератор создает приложение целиком.

SpringWolf

1. Игнорирует валидации.

Почему нам не подходит AsyncAPI

AsyncAPI

1. Инфраструктура вокруг AsyncAPI написана на JS.
2. Это еще один DSL.
3. Генератор создает приложение целиком.

SpringWolf

1. Игнорирует валидации.
2. Сложности при подключении.

Почему нам не подходит AsyncAPI

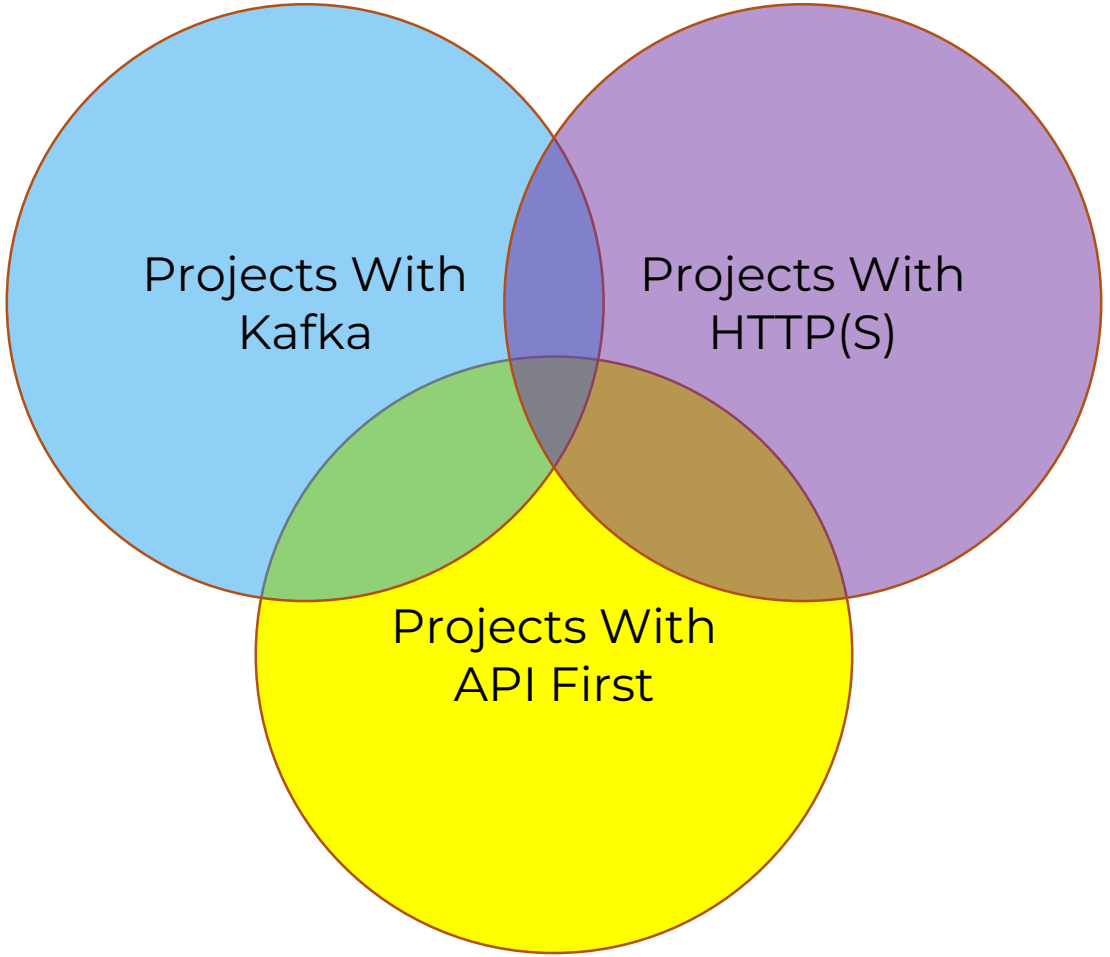
AsyncAPI

1. Инфраструктура вокруг AsyncAPI написана на JS.
2. Это еще один DSL.
3. Генератор создает приложение целиком.

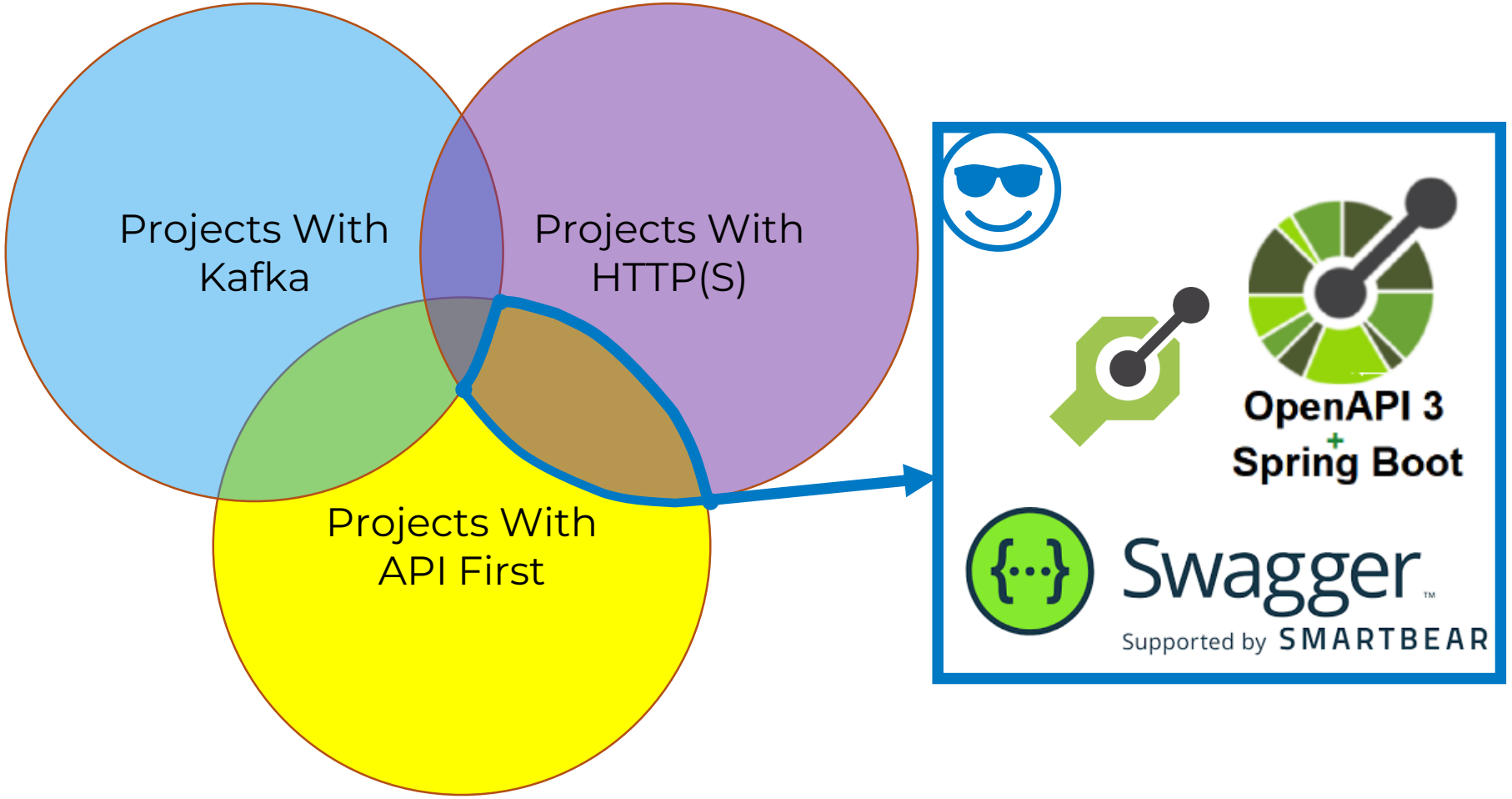
SpringWolf

1. Игнорирует валидации.
2. Сложности при подключении.
3. Игнорирует headers.

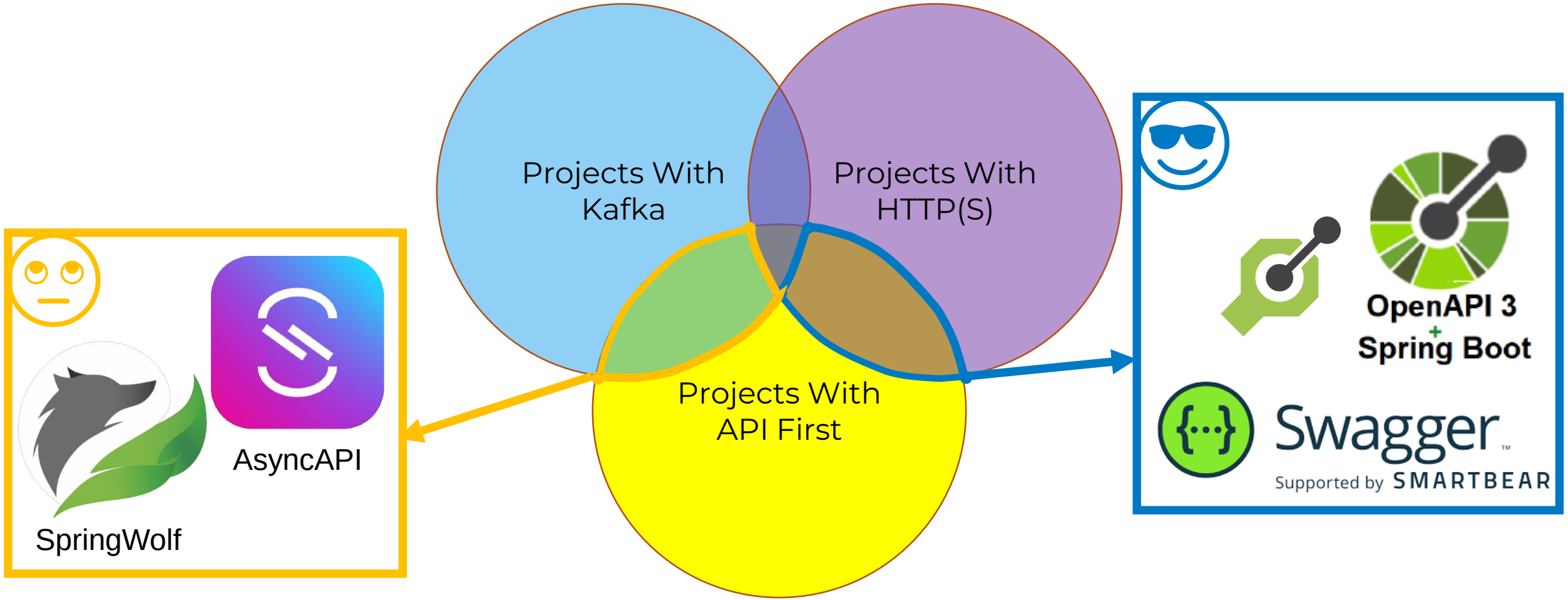
API First с Kafka и HTTP



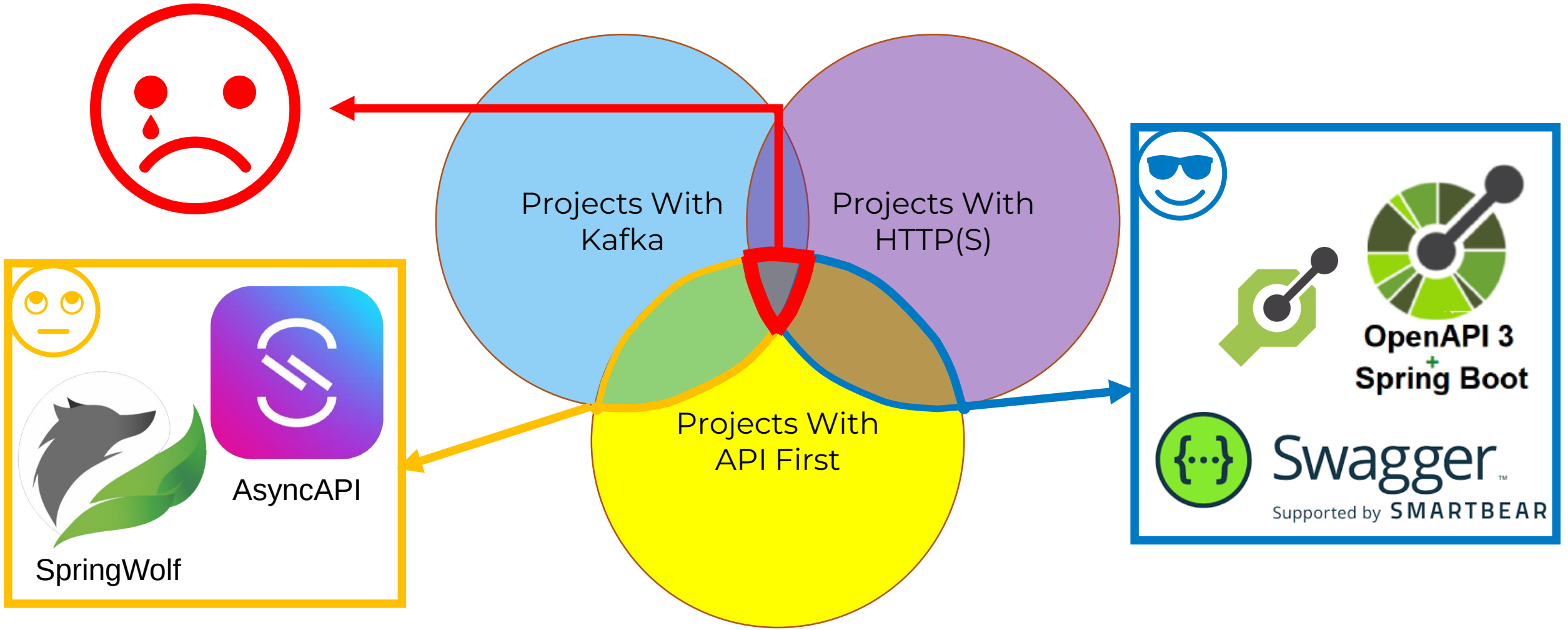
API First с Kafka и HTTP



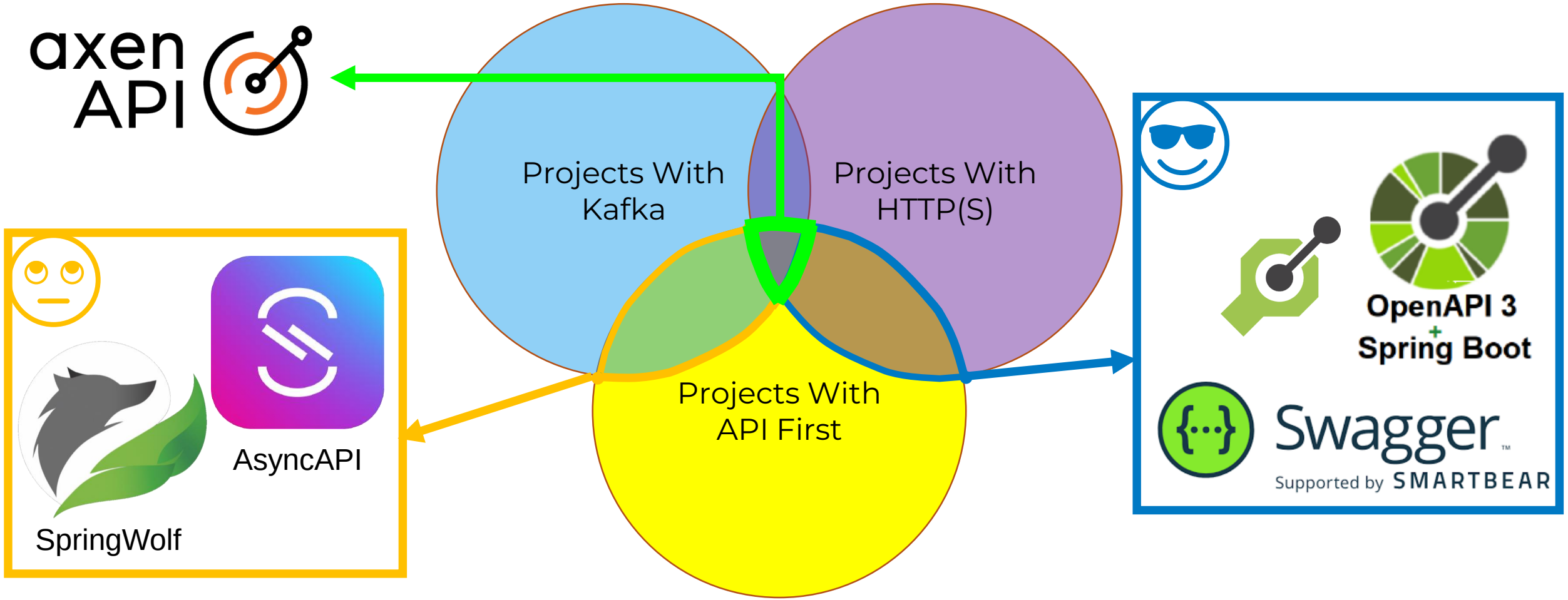
API First с Kafka и HTTP



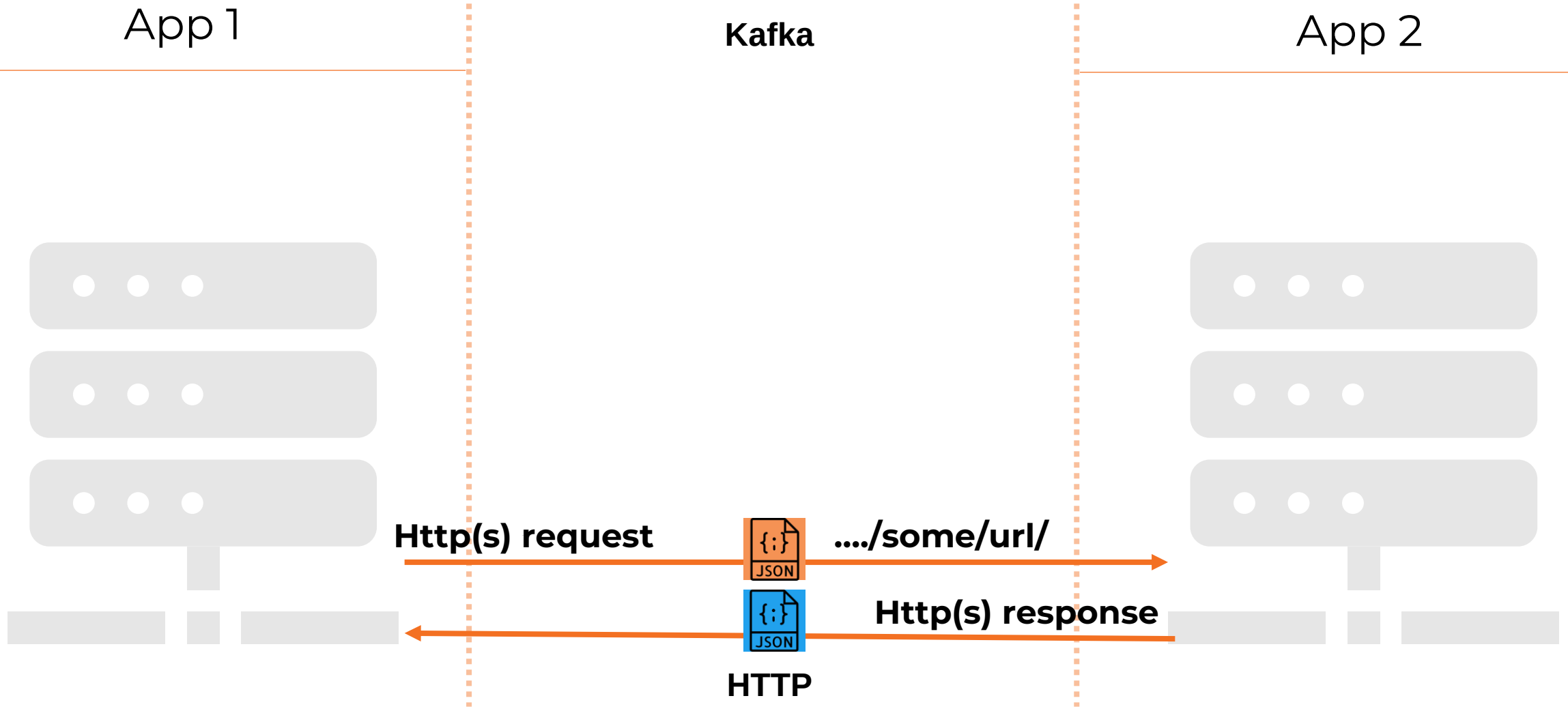
API First с Kafka и HTTP



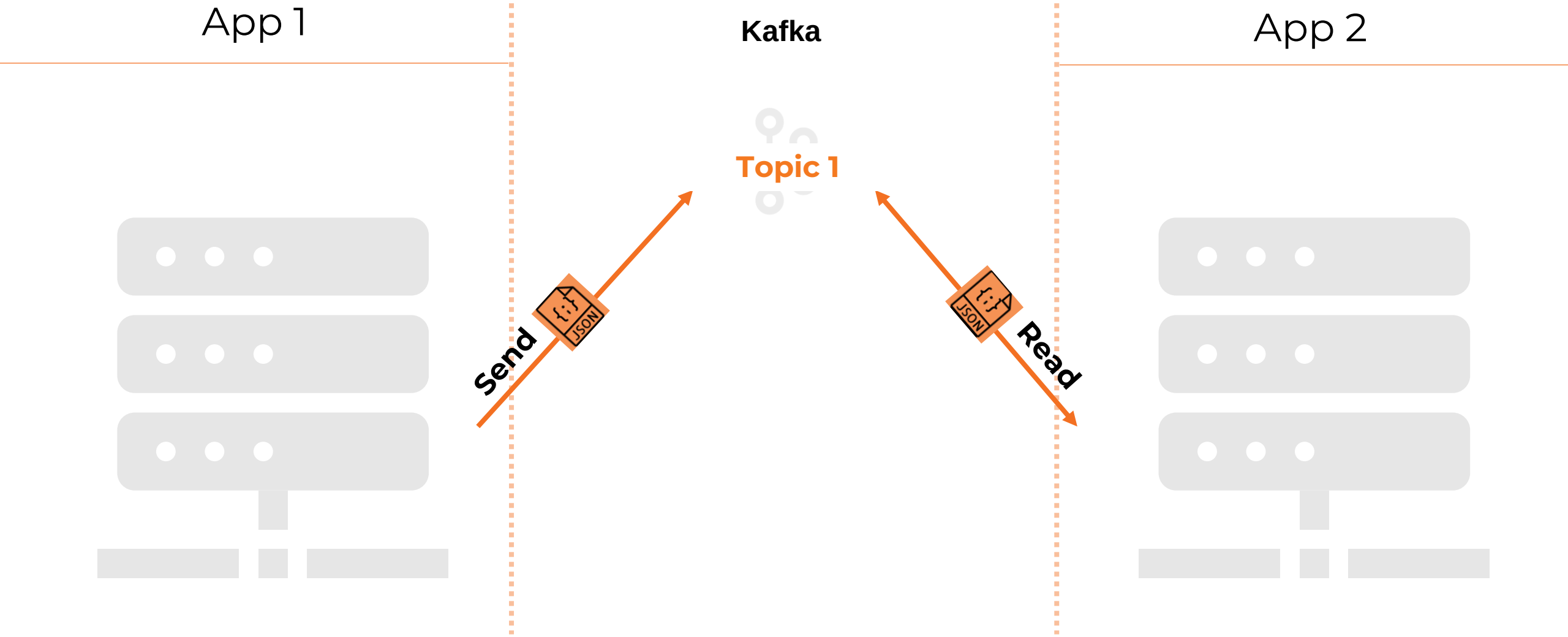
API First с Kafka и HTTP



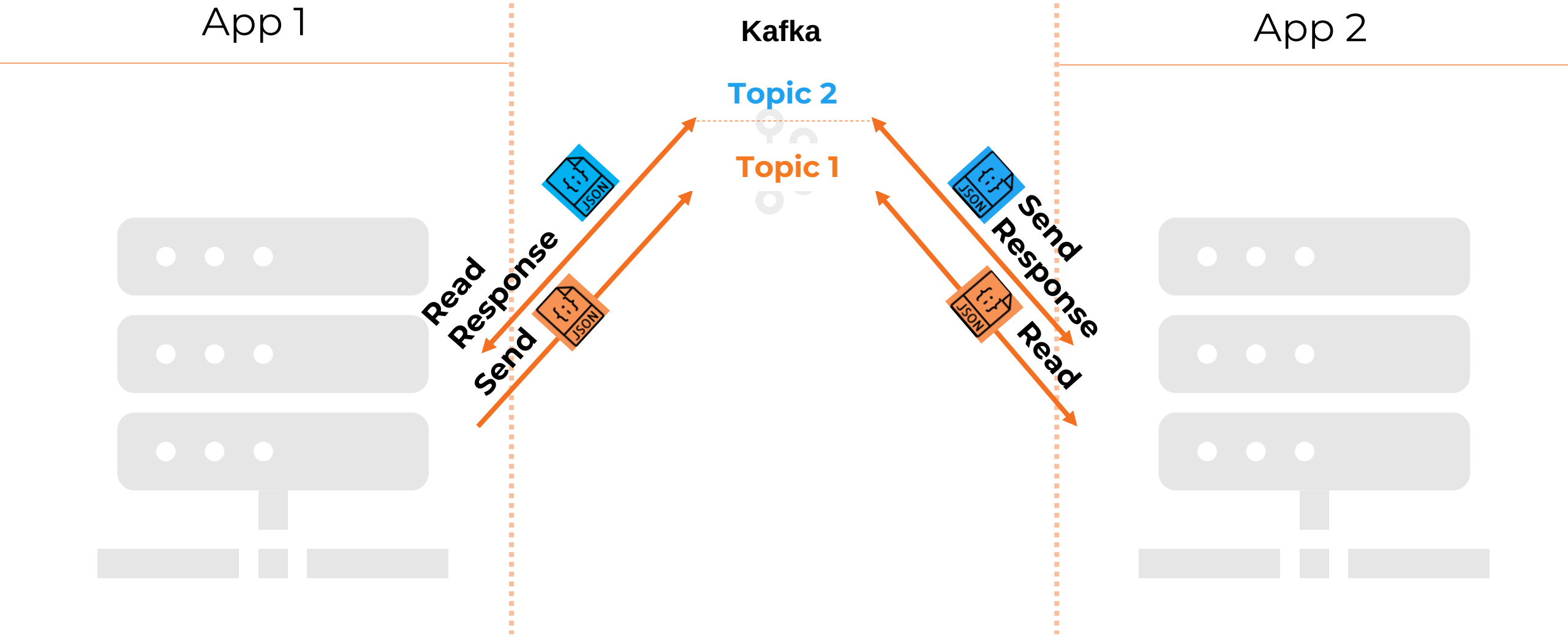
Kafka VS HTTP с точки зрения клиента



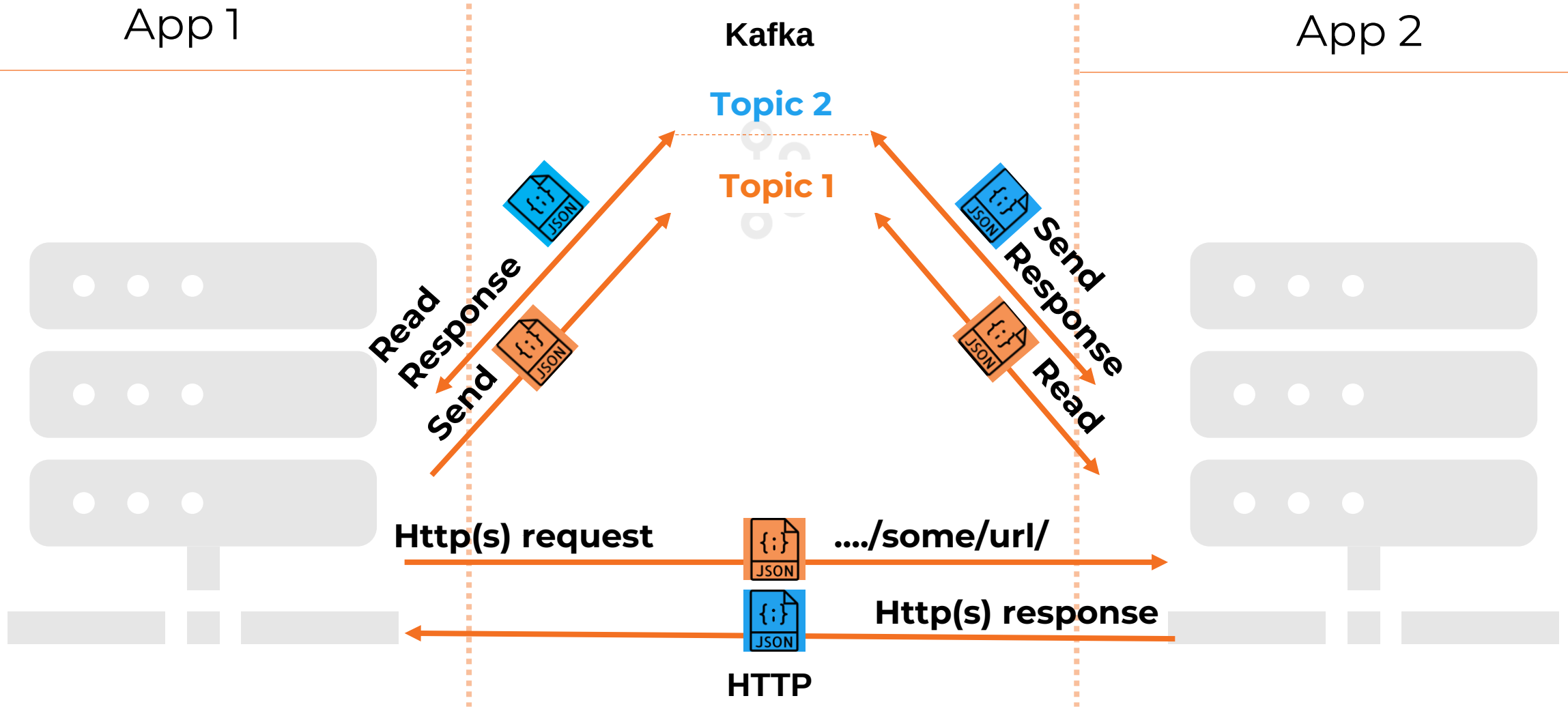
Kafka VS HTTP с точки зрения клиента



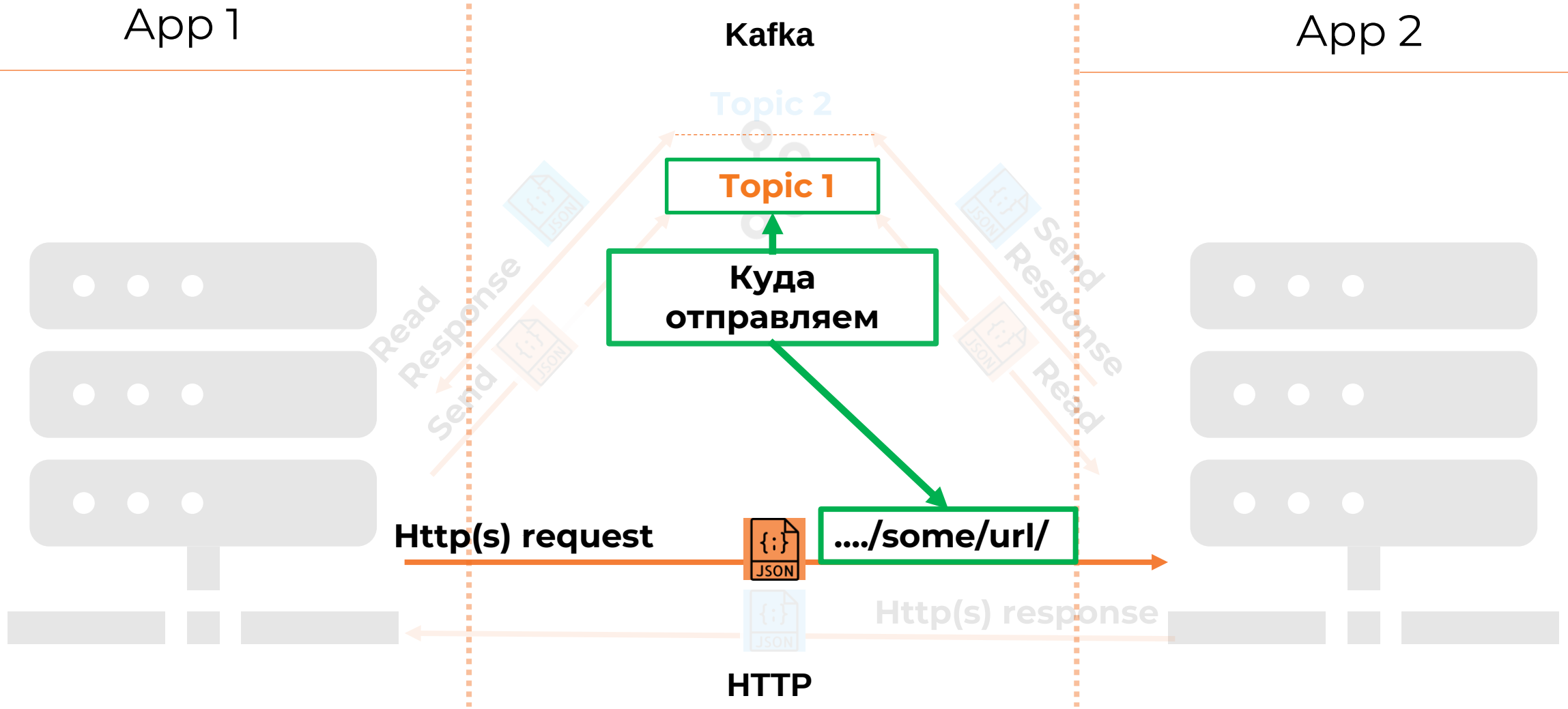
Kafka VS HTTP с точки зрения клиента



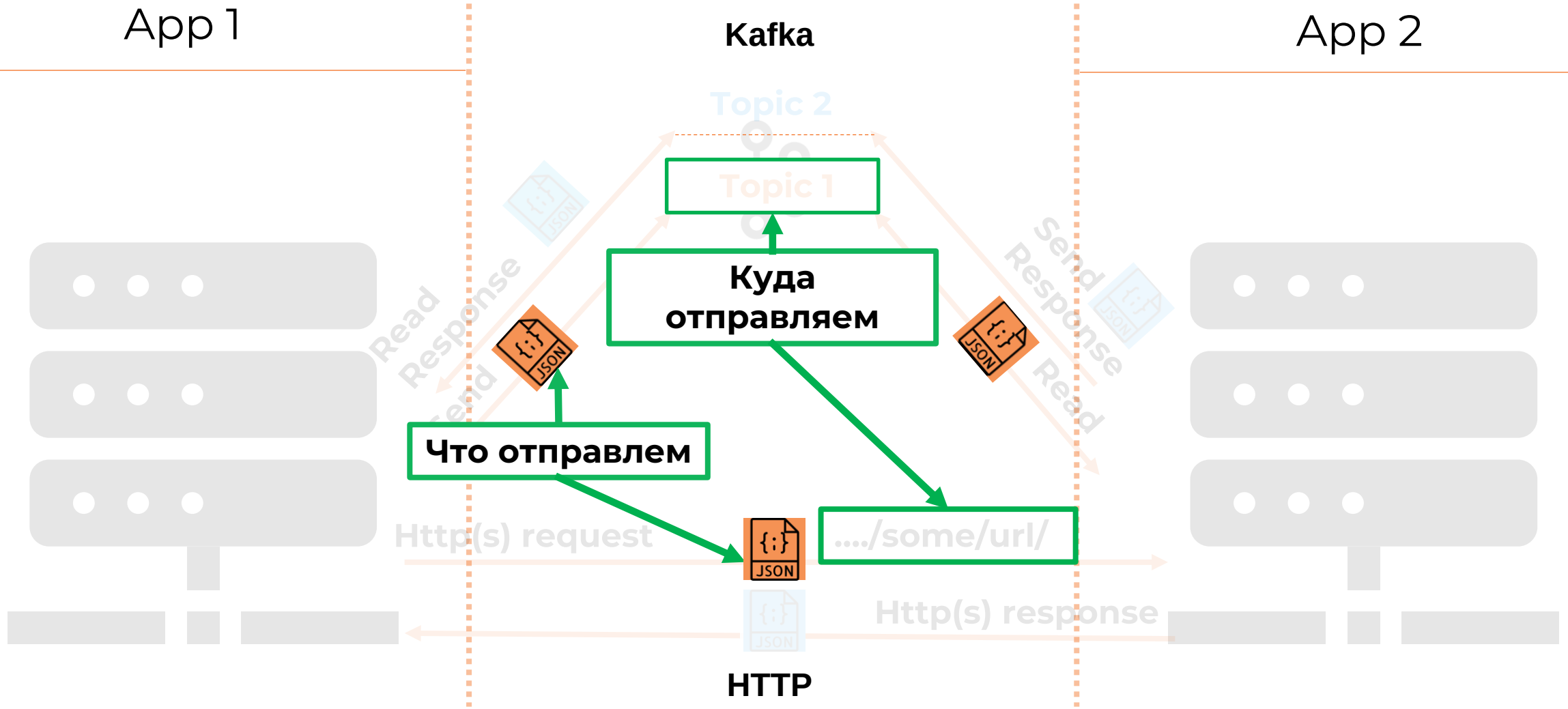
Kafka VS HTTP с точки зрения клиента



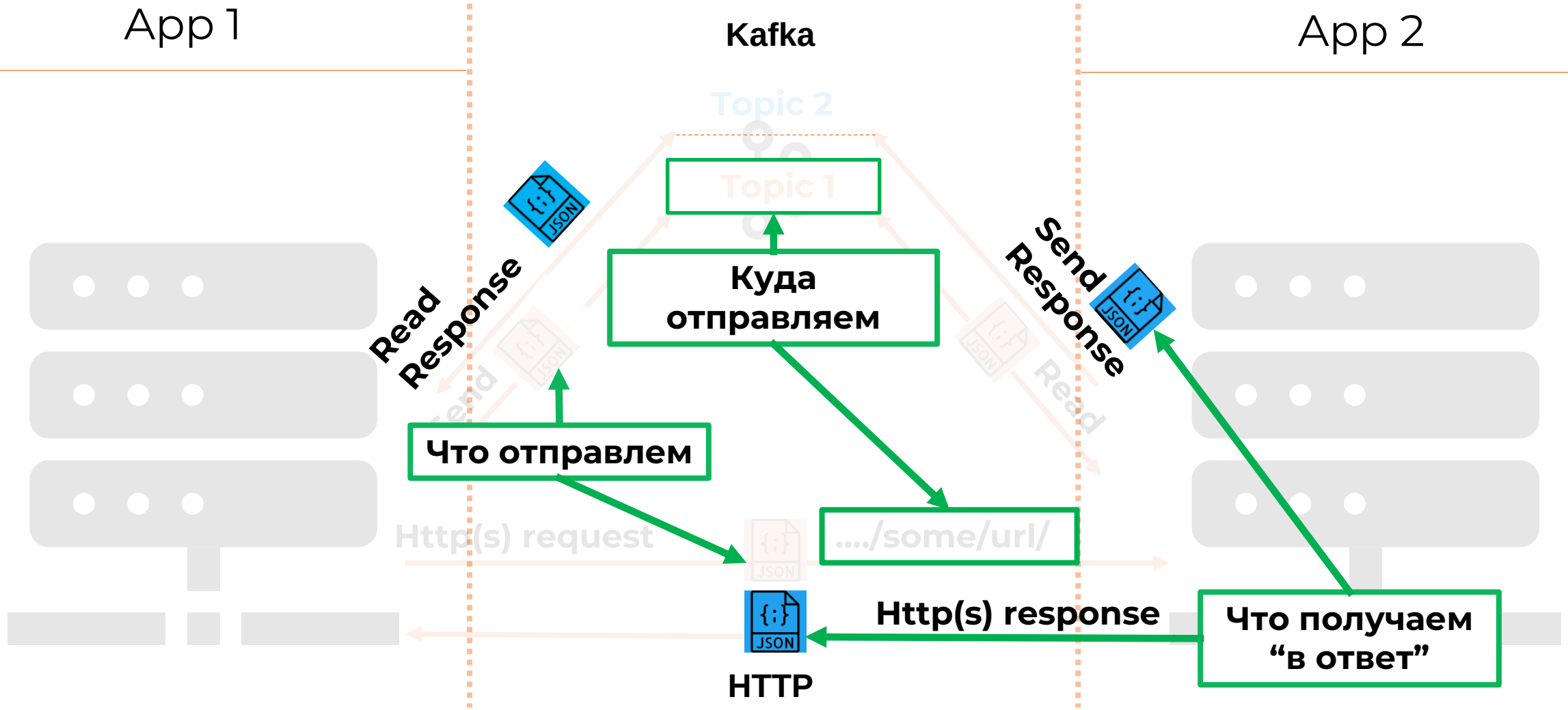
Kafka VS HTTP с точки зрения клиента



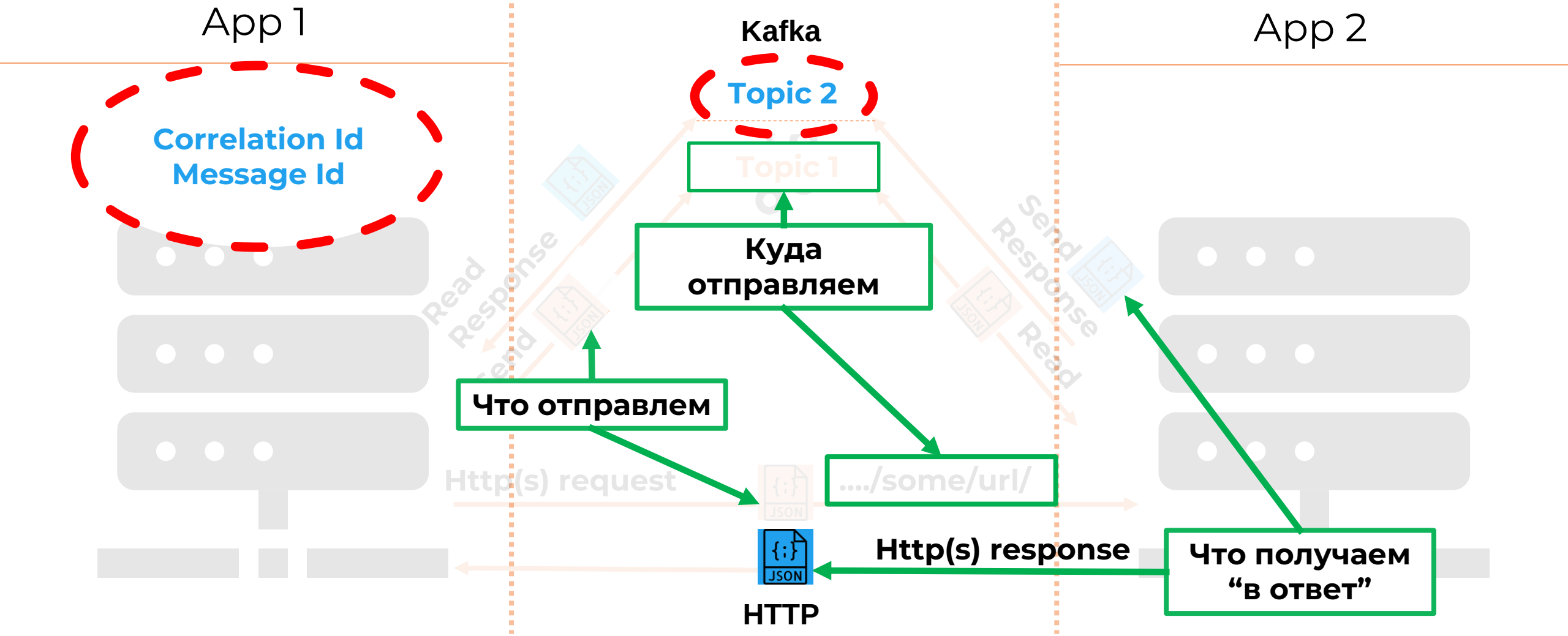
Kafka VS HTTP с точки зрения клиента



Kafka VS HTTP с точки зрения клиента



Kafka VS HTTP с точки зрения клиента



AxenAPI: адаптация OpenAPI под Kafka

В начале разработки

1. Разабатываете (или дорабатываете) спецификацию в формате OpenAPI.
2. Генерируете по ней код.

AxenAPI
Gradle Plugin

В процессе разработки

0. Инициировать первую версию спецификации в формате OpenAPI

AxenAPI
Library

Структура OpenAPI спецификации

```
"paths": {
  "/kafka/group-2/multiType/Subordinate": {
    "post": {
      "tags": [
        "group-2",
        "subordinate"
      ],
      "description": "Message listener with Subordinate payload type",
      "operationId": "executeSubordinate",
      "parameters": [
        {
          "name": "req",
          "in": "query",
          "schema": {
            "type": "string"
          }
        }
      ],
      "requestBody": {···
      },
      "responses": {
        "200": {
          "description": "Возвращает ответ Subordinate в топик, передаваемый через хедер replyTopic. Возвращаемое значение не перехватывается",
          "content": {
            "*/*": {
              "schema": {
                "$ref": "#/components/schemas/Subordinate"
              }
            }
          }
        }
      }
    }
  },
  "/kafka/multiType/Object": {
    "post": {
      "tags": [
        "group-2",
        "default handler"
```

Структура OpenAPI спецификации

Префикс “kafka”

```
"paths": {
  "/kafka/group-2/multiType/Subordinate": {
    "post": {
      "tags": [
        "group-2",
        "subordinate"
      ],
      "description": "Message listener with Subordinate payload type",
      "operationId": "executeSubordinate",
      "parameters": [
        {
          "name": "req",
          "in": "query",
          "schema": {
            "type": "string"
          }
        }
      ],
      "requestBody": {···
      },
      "responses": {
        "200": {
          "description": "Возвращает ответ Subordinate в топик, передаваемый через хедер replyTopic. Возвращаемое значение не перехватывается",
          "content": {
            "*/*": {
              "schema": {
                "$ref": "#/components/schemas/Subordinate"
              }
            }
          }
        }
      }
    }
  },
  "/kafka/multiType/Object": {
    "post": {
      "tags": [
        "group-2",
        "default handler"
```

Структура OpenAPI спецификации

Префикс “kafka”
Post метод

```
"paths": {
  "/kafka/group-2/multiType/Subordinate": {
    "post": {
      "tags": [
        "group-2",
        "subordinate"
      ],
      "description": "Message listener with Subordinate payload type",
      "operationId": "executeSubordinate",
      "parameters": [
        {
          "name": "req",
          "in": "query",
          "schema": {
            "type": "string"
          }
        }
      ],
      "requestBody": {···
      },
      "responses": {
        "200": {
          "description": "Возвращает ответ Subordinate в топик, передаваемый через хедер replyTopic. Возвращаемое значение не перехватывается",
          "content": {
            "*/*": {
              "schema": {
                "$ref": "#/components/schemas/Subordinate"
              }
            }
          }
        }
      }
    }
  },
  "/kafka/multiType/Object": {
    "post": {
      "tags": [
        "group-2",
        "default handler"
```

Структура OpenAPI спецификации

Префикс “kafka”
Post метод
Наименование группы(необязательно)

```
"paths": {
  "/kafka/group-2/multiType/Subordinate": {
    "post": {
      "tags": [
        "group-2",
        "subordinate"
      ],
      "description": "Message listener with Subordinate payload type",
      "operationId": "executeSubordinate",
      "parameters": [
        {
          "name": "req",
          "in": "query",
          "schema": {
            "type": "string"
          }
        }
      ],
      "requestBody": {···
      },
      "responses": {
        "200": {
          "description": "Возвращает ответ Subordinate в топик, передаваемый через хедер replyTopic. Возвращаемое значение не перехватывается",
          "content": {
            "*/*": {
              "schema": {
                "$ref": "#/components/schemas/Subordinate"
              }
            }
          }
        }
      }
    }
  },
  "/kafka/multiType/Object": {
    "post": {
      "tags": [
        "group-2",
        "default handler"
```

Структура OpenAPI спецификации

Префикс “kafka”
Post метод
Наименование группы(необязательно)
Наименование топика

```
"paths": {
  "/kafka/group-2/multiType/Subordinate": {
    "post": {
      "tags": [
        "group-2",
        "subordinate"
      ],
      "description": "Message listener with Subordinate payload type",
      "operationId": "executeSubordinate",
      "parameters": [
        {
          "name": "req",
          "in": "query",
          "schema": {
            "type": "string"
          }
        }
      ],
      "requestBody": {···
      },
      "responses": {
        "200": {
          "description": "Возвращает ответ Subordinate в топик, передаваемый через хедер replyTopic. Возвращаемое значение не перехватывается",
          "content": {
            "*/*": {
              "schema": {
                "$ref": "#/components/schemas/Subordinate"
              }
            }
          }
        }
      }
    }
  },
  "/kafka/multiType/Object": {
    "post": {
      "tags": [
        "group-2",
        "default handler"
```

Структура OpenAPI спецификации

```
"paths": {
  "/kafka/group-2/multiType/Subordinate": {
    "post": {
      "tags": [
        "group-2",
        "subordinate"
      ],
      "description": "Message listener with Subordinate payload type",
      "operationId": "executeSubordinate",
      "parameters": [
        {
          "name": "req",
          "in": "query",
          "schema": {
            "type": "string"
          }
        }
      ],
      "requestBody": {···
      },
      "responses": {
        "200": {
          "description": "Возвращает ответ Subordinate в топик, передаваемый через хедер replyTopic. Возвращаемое значение не перехватывается",
          "content": {
            "*/*": {
              "schema": {
                "$ref": "#/components/schemas/Subordinate"
              }
            }
          }
        }
      }
    }
  },
  "/kafka/multiType/Object": {
    "post": {
      "tags": [
        "group-2",
        "default handler"
```

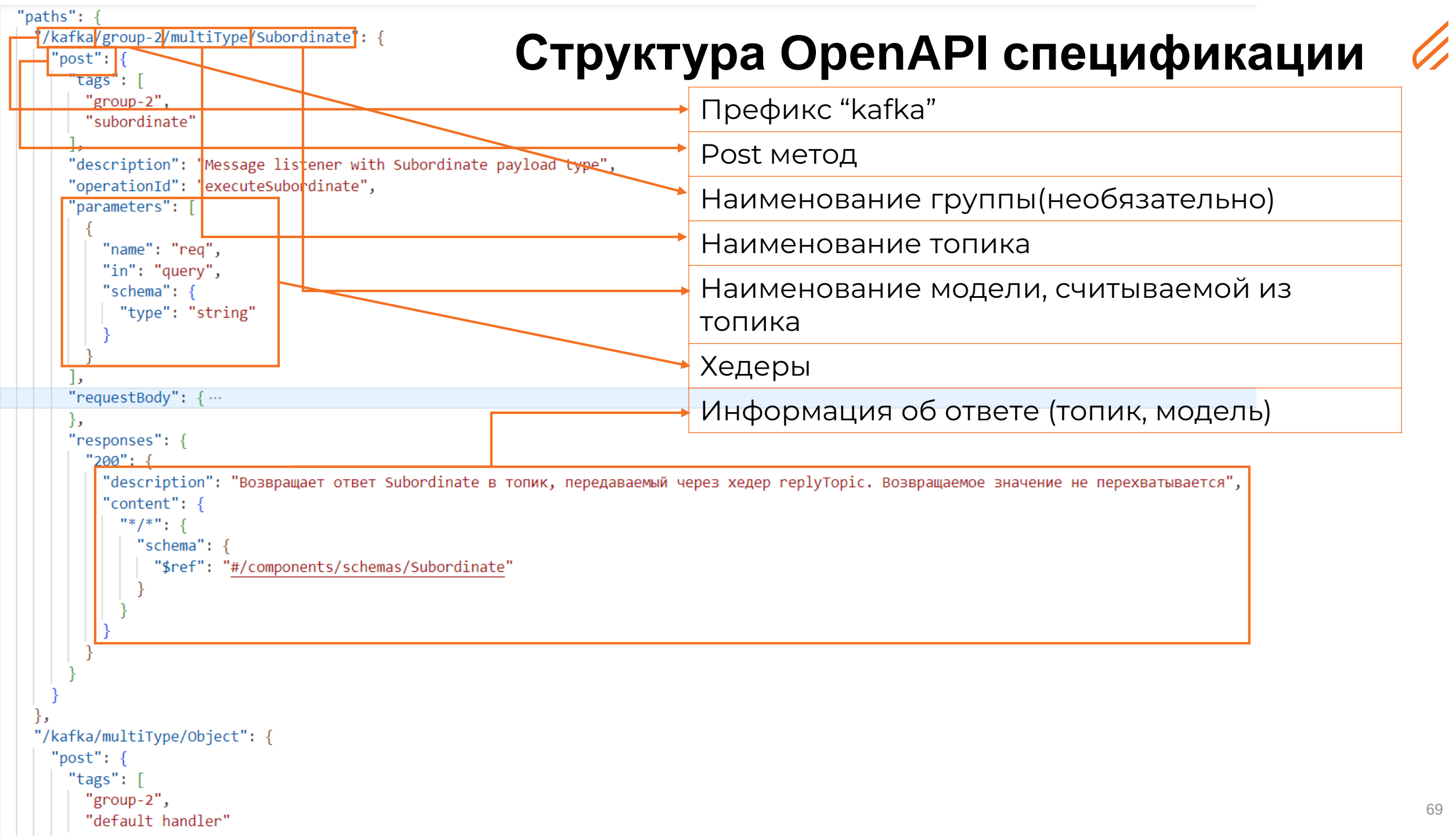
Префикс “kafka”
Post метод
Наименование группы(необязательно)
Наименование топика
Наименование модели, считываемой из топика

Структура OpenAPI спецификации

```
"paths": {
  "/kafka/group-2/multiType/Subordinate": {
    "post": {
      "tags": [
        "group-2",
        "subordinate"
      ],
      "description": "Message listener with Subordinate payload type",
      "operationId": "executeSubordinate",
      "parameters": [
        {
          "name": "req",
          "in": "query",
          "schema": {
            "type": "string"
          }
        }
      ],
      "requestBody": {···
      },
      "responses": {
        "200": {
          "description": "Возвращает ответ Subordinate в топик, передаваемый через хедер replyTopic. Возвращаемое значение не перехватывается",
          "content": {
            "*/*": {
              "schema": {
                "$ref": "#/components/schemas/Subordinate"
              }
            }
          }
        }
      }
    }
  },
  "/kafka/multiType/Object": {
    "post": {
      "tags": [
        "group-2",
        "default handler"
```

Префикс “kafka”
Post метод
Наименование группы(необязательно)
Наименование топика
Наименование модели, считываемой из топика
Хедеры

Структура OpenAPI спецификации



```
"paths": {
  "/kafka/group-2/multiType/Subordinate": {
    "post": {
      "tags": [
        "group-2",
        "subordinate"
      ],
      "description": "Message listener with Subordinate payload type",
      "operationId": "executeSubordinate",
      "parameters": [
        {
          "name": "req",
          "in": "query",
          "schema": {
            "type": "string"
          }
        }
      ],
      "requestBody": {···
      },
      "responses": {
        "200": {
          "description": "Возвращает ответ Subordinate в топик, передаваемый через хедер replyTopic. Возвращаемое значение не перехватывается",
          "content": {
            "*/*": {
              "schema": {
                "$ref": "#/components/schemas/Subordinate"
              }
            }
          }
        }
      }
    }
  },
  "/kafka/multiType/Object": {
    "post": {
      "tags": [
        "group-2",
        "default handler"
```

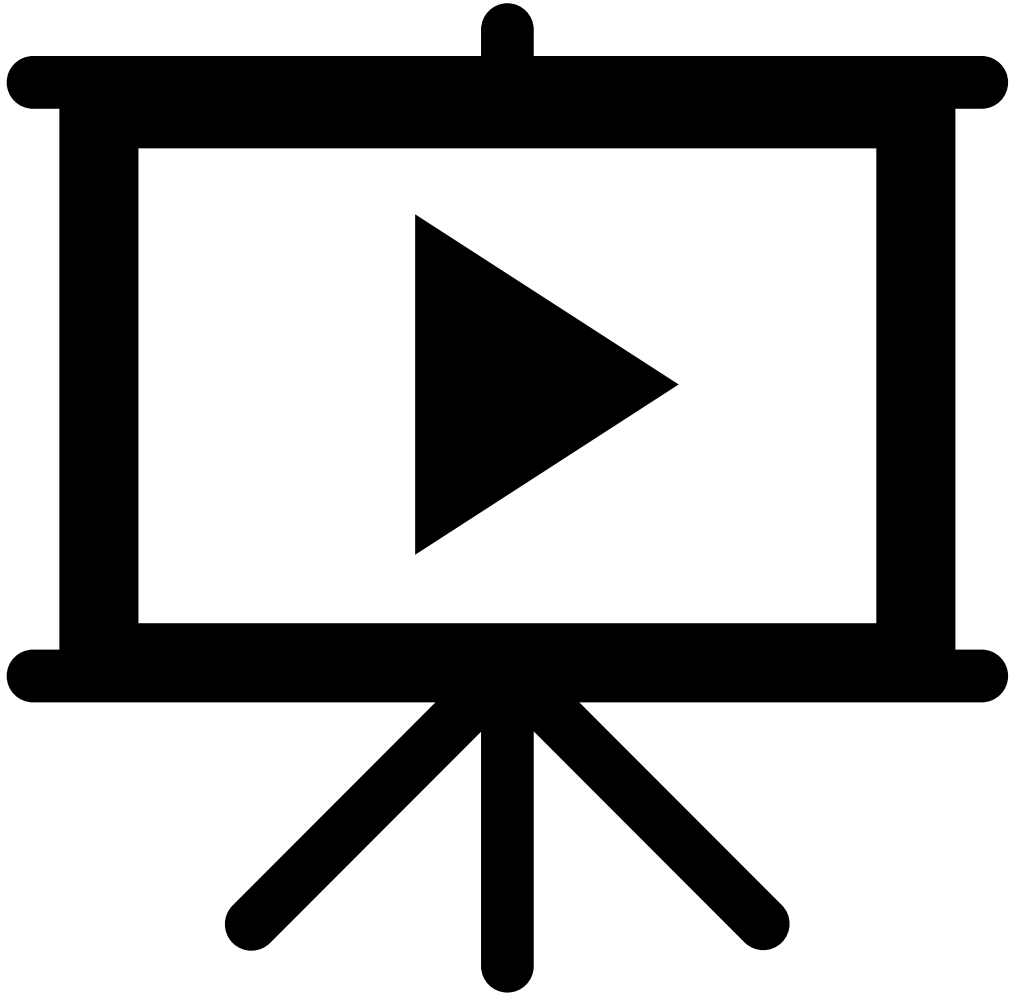
Префикс “kafka”
Post метод
Наименование группы(необязательно)
Наименование топика
Наименование модели, считываемой из топика
Хедеры
Информация об ответе (топик, модель)

Gradle-плагин AxenAPI

Gradle-плагин AxenAPI предназначен для генерации кода на Java с использованием Spring по спецификации API, реализованного с помощью брокеров сообщений.

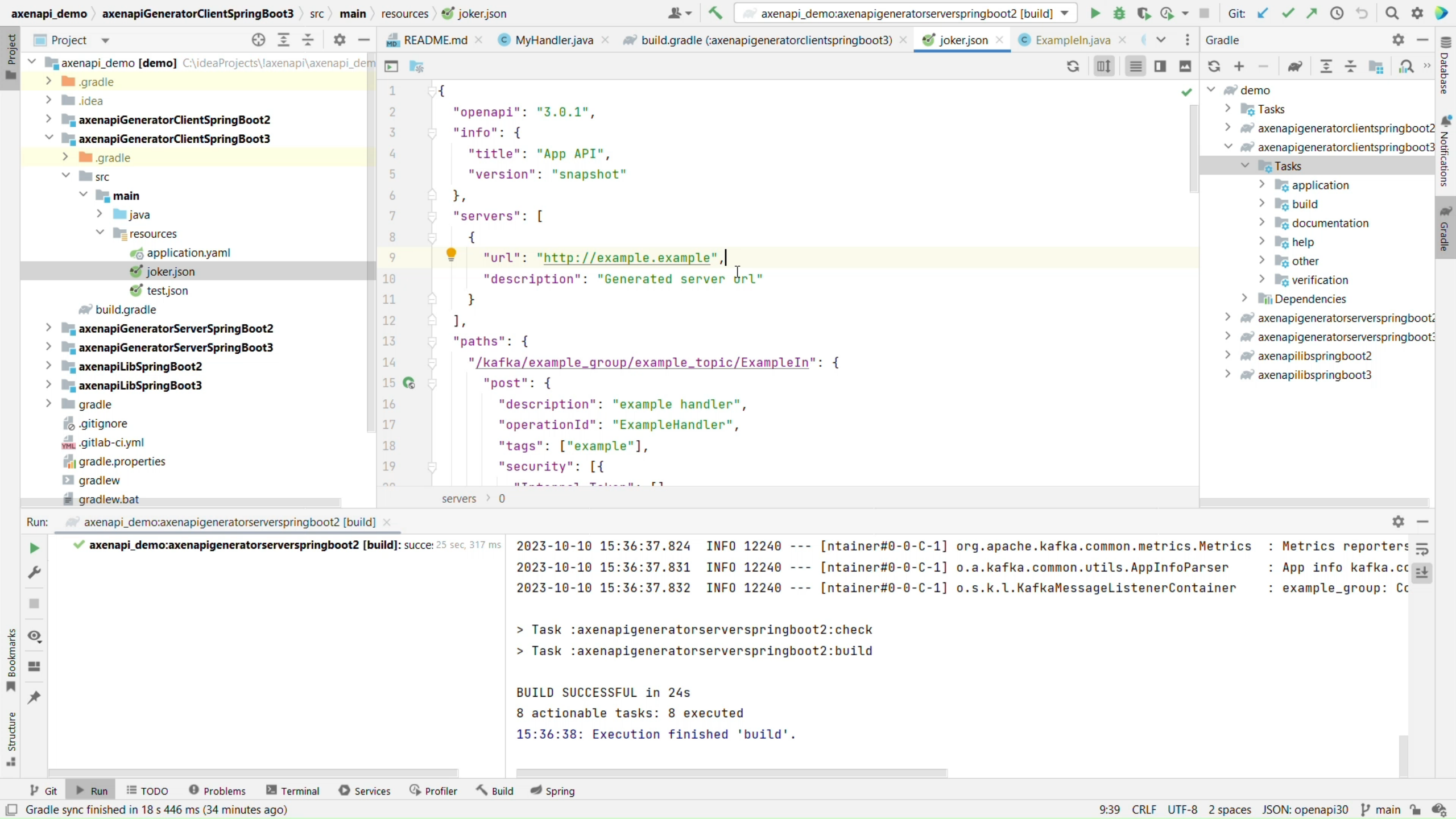
Gradle-плагин AxenAPI. Демо

- Видео 1 – создание спецификации API и генерация по ней кода сервера.
- Видео 2 – генерация кода клиентского приложения по спецификации API.

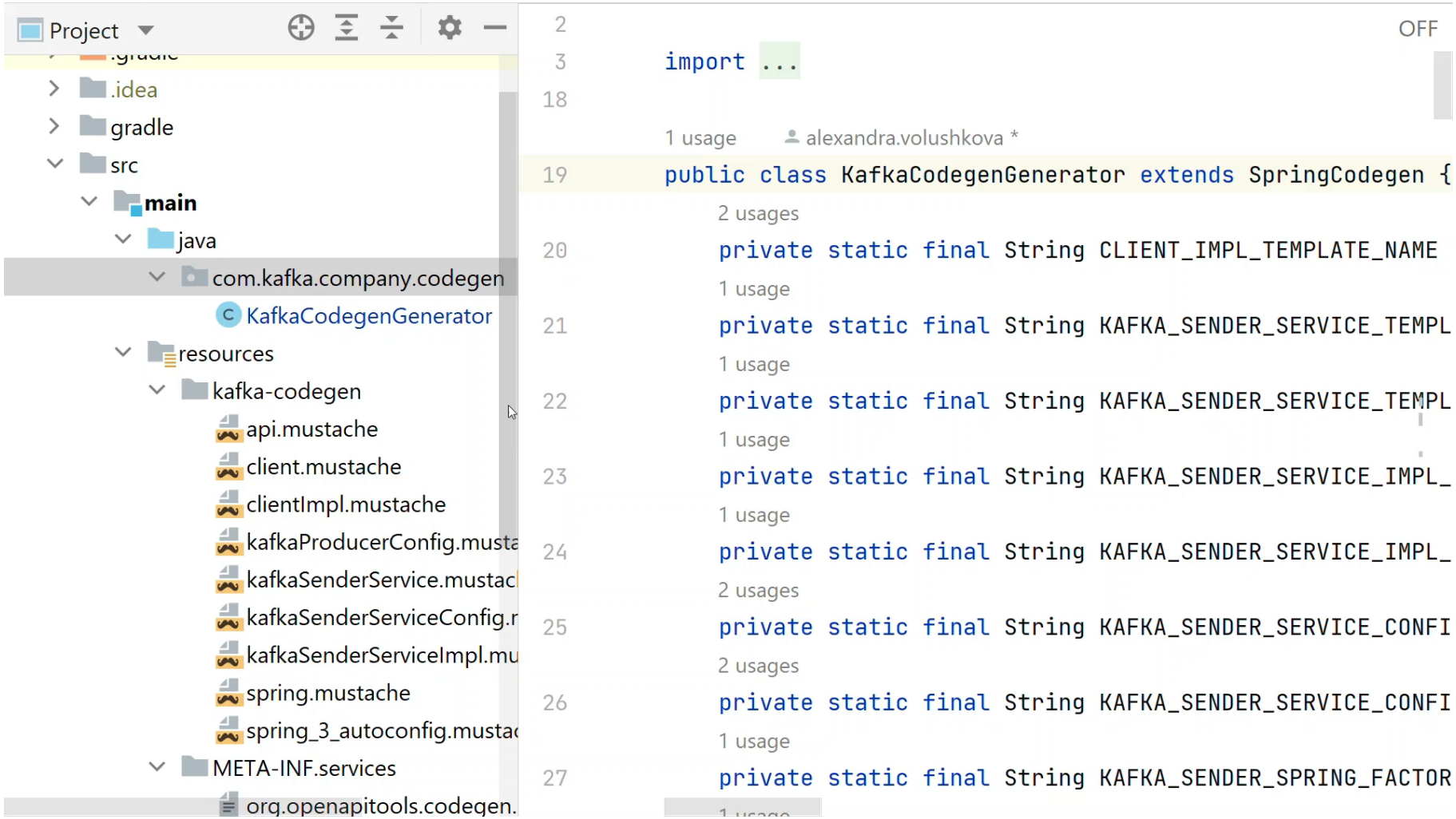


```
    "version": "snapshot"
  },
  "servers": [
    {
      "url":  "http://example.example",
      "description": "Generated server url"
    }
  ],
  "paths": {

  },
  "components": {
    "schemas": {
      "ExampleIn": {"type": "object"...},
      "ExampleOut": {"type": "object"...}
    },
    "securitySchemes": {
      "Internal-Token": {
```

paths


Project
axenapi_demo [demo] C:\ideaProjects\!axenapi\axenapi_dem
.gradle
.idea
axenapiGeneratorClientSpringBoot2
axenapiGeneratorClientSpringBoot3
.gradle
src
main
java
resources
application.yaml
joker.json
test.json
build.gradle
axenapiGeneratorServerSpringBoot2
axenapiGeneratorServerSpringBoot3
axenapiLibSpringBoot2
axenapiLibSpringBoot3
gradle
.gitignore
.gitlab-ci.yml
gradle.properties
gradlew
gradlew.bat
README.md
MyHandler.java
build.gradle (:axenapigeneratorclientspringboot3)
joker.json
ExampleIn.java
{
"openapi": "3.0.1",
"info": {
"title": "App API",
"version": "snapshot"
},
"servers": [
{
"url": "http://example.example",
"description": "Generated server url"
}
],
"paths": {
"/kafka/example_group/example_topic/ExampleIn": {
"post": {
"description": "example handler",
"operationId": "ExampleHandler",
"tags": ["example"],
"security": [{
servers 〉 0
Gradle
demo
Tasks
axenapigeneratorclientspringboot2
axenapigeneratorclientspringboot3
Tasks
application
build
documentation
help
other
verification
Dependencies
axenapigeneratorserverspringboot2
axenapigeneratorserverspringboot3
axenapilibspringboot2
axenapilibspringboot3
Run: axenapi_demo:axenapigeneratorserverspringboot2 [build]
axenapi_demo:axenapigeneratorserverspringboot2 [build]: succe: 25 sec, 317 ms
2023-10-10 15:36:37.824 INFO 12240 --- [ntainer#0-0-C-1] org.apache.kafka.common.metrics.Metrics : Metrics reporters
2023-10-10 15:36:37.831 INFO 12240 --- [ntainer#0-0-C-1] o.a.kafka.common.utils.AppInfoParser : App info kafka.co
2023-10-10 15:36:37.832 INFO 12240 --- [ntainer#0-0-C-1] o.s.k.l.KafkaMessageListenerContainer : example_group: Co
> Task :axenapigeneratorserverspringboot2:check
> Task :axenapigeneratorserverspringboot2:build
BUILD SUCCESSFUL in 24s
8 actionable tasks: 8 executed
15:36:38: Execution finished 'build'.
Git
Run
TODO
Problems
Terminal
Services
Profiler
Build
Spring
Gradle sync finished in 18 s 446 ms (34 minutes ago)
9:39
CRLF
UTF-8
2 spaces
JSON: openapi30
main

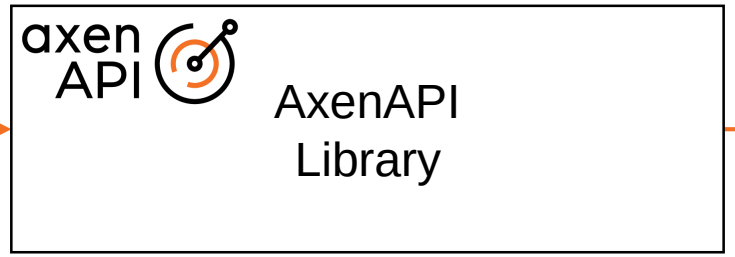
Gradle-плагин AxenAPI. Как мы это сделали?



Библиотека AxenAPI

Библиотека AxenAPI предназначена для генерации OpenAPI документации для различных listeners различных брокеров в сервисе – kafka (ActiveMq, Rabbit).

Application code




```yaml
servers:
  - url: http://petstore.swagger.io/v1
paths:
  /pets:
    get:
      summary: List all pets
      operationId: listPets
      tags: <1 item>
      security:
        - ApiKeyAuth: []
      parameters:
        - name: limit
          in: query # "path", "query", "cookie", "header"
          description: How many items to return at one time (max 100)
          required: false
          schema:
            type: integer
            format: int32
      responses:
        '200':
          description: A paged array of pets
          headers:
            x-next:
              description: A link to the next page of responses
              schema:
                type: string
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/Pets"
```

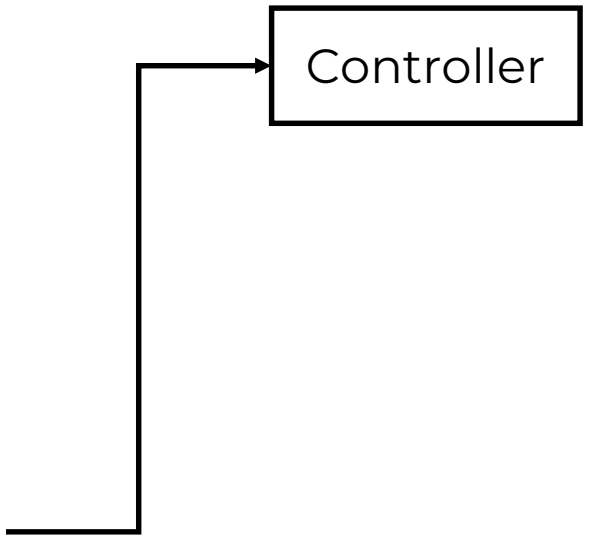
OpenAPI Specification

Библиотека AxenAPI. Принцип работы

```
@KafkaListener(topics = {"demo_topic"}, groupId = "foo")
public class DemoListenerDifferencesTypes {

   @KafkaHandler
   @KafkaHandlerDescription("Принимает  ExampleMessage. Находится в
DemoListener. Ничего не возвращает.")
   @KafkaHandlerHeaders( headers = {
         @KafkaHandlerHeader(header = "header_1", required = true),
         @KafkaHandlerHeader(header = "header_2")
   })
   @KafkaSecured
   @KafkaHandlerTags(tags = {"demo"})
   public void listenGroupFoo(@Payload ExampleMessage message) {
      System.out.println("Received Message in group foo: " + message);
   }

   @KafkaHandler
   @KafkaHandlerDescription("Принимает  Subordinate. Находится в
DemoListener. Ничего не возвращает.")
   @KafkaHandlerTags(tags = {"demo"})
   public void listenGroupFoo(@Payload Subordinate message) {
      System.out.println("Received Message in group foo: " + message);
   }
}
```

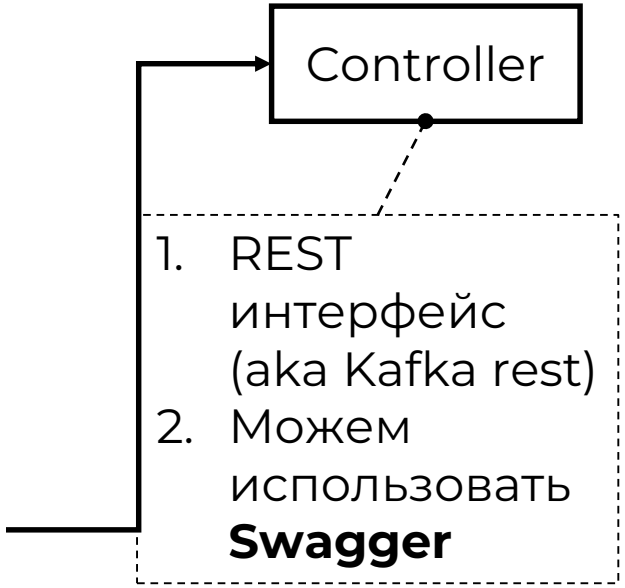
Библиотека AxenAPI. Принцип работы

```
@KafkaListener(topics = {"demo_topic"}, groupId = "foo")
public class DemoListenerDifferencesTypes {

   @KafkaHandler
   @KafkaHandlerDescription("Принимает  ExampleMessage. Находится в
DemoListener. Ничего не возвращает.")
   @KafkaHandlerHeaders( headers = {
         @KafkaHandlerHeader(header = "header_1", required = true),
         @KafkaHandlerHeader(header = "header_2")
   })
   @KafkaSecured
   @KafkaHandlerTags(tags = {"demo"})
   public void listenGroupFoo(@Payload ExampleMessage message) {
      System.out.println("Received Message in group foo: " + message);
   }

   @KafkaHandler
   @KafkaHandlerDescription("Принимает  Subordinate. Находится в
DemoListener. Ничего не возвращает.")
   @KafkaHandlerTags(tags = {"demo"})
   public void listenGroupFoo(@Payload Subordinate message) {
      System.out.println("Received Message in group foo: " + message);
   }
}
```



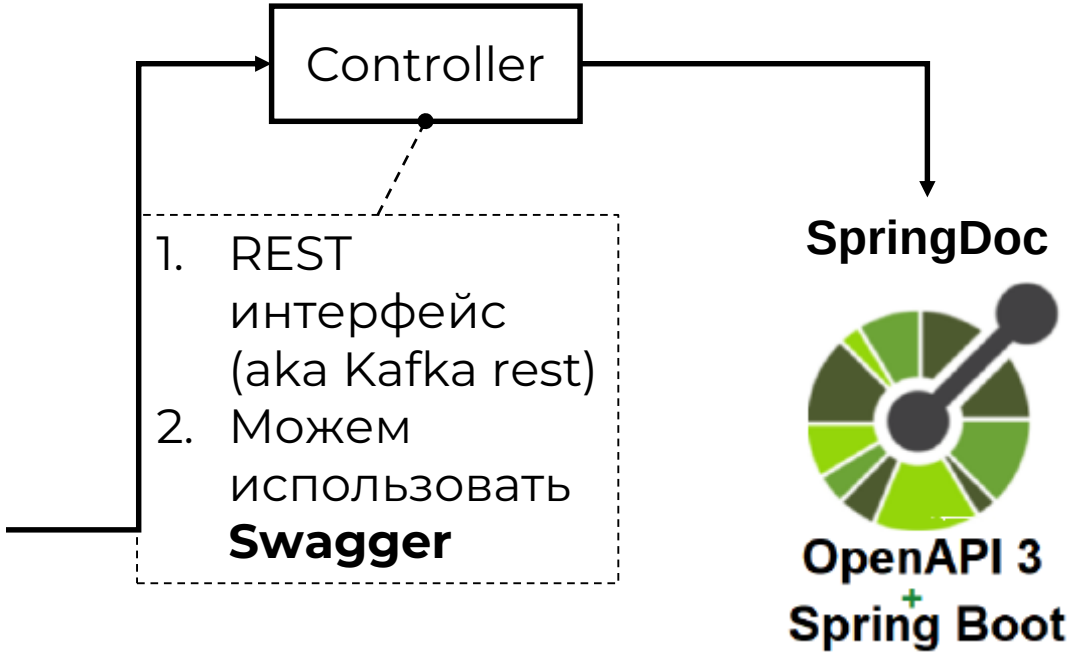
Библиотека AxenAPI. Принцип работы

```
@KafkaListener(topics = {"demo_topic"}, groupId = "foo")
public class DemoListenerDifferencesTypes {

   @KafkaHandler
   @KafkaHandlerDescription("Принимает  ExampleMessage. Находится в
DemoListener. Ничего не возвращает.")
   @KafkaHandlerHeaders( headers = {
         @KafkaHandlerHeader(header = "header_1", required = true),
         @KafkaHandlerHeader(header = "header_2")
   })
   @KafkaSecured
   @KafkaHandlerTags(tags = {"demo"})
   public void listenGroupFoo(@Payload ExampleMessage message) {
      System.out.println("Received Message in group foo: " + message);
   }

   @KafkaHandler
   @KafkaHandlerDescription("Принимает  Subordinate. Находится в
DemoListener. Ничего не возвращает.")
   @KafkaHandlerTags(tags = {"demo"})
   public void listenGroupFoo(@Payload Subordinate message) {
      System.out.println("Received Message in group foo: " + message);
   }
}
```



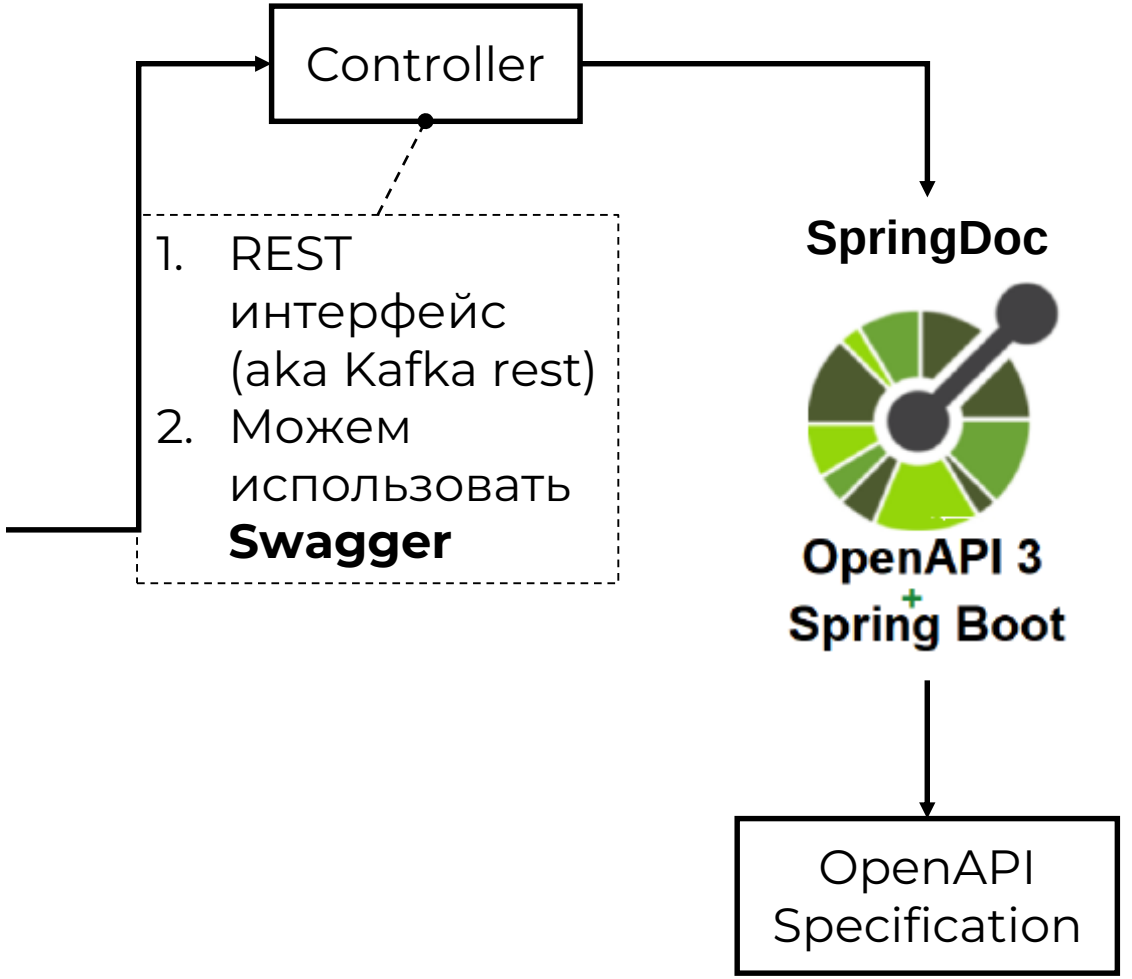
Библиотека AxenAPI. Принцип работы

```
@KafkaListener(topics = {"demo_topic"}, groupId = "foo")
public class DemoListenerDifferencesTypes {

   @KafkaHandler
   @KafkaHandlerDescription("Принимает  ExampleMessage. Находится в
DemoListener. Ничего не возвращает.")
   @KafkaHandlerHeaders( headers = {
         @KafkaHandlerHeader(header = "header_1", required = true),
         @KafkaHandlerHeader(header = "header_2")
   })
   @KafkaSecured
   @KafkaHandlerTags(tags = {"demo"})
   public void listenGroupFoo(@Payload ExampleMessage message) {
      System.out.println("Received Message in group foo: " + message);
   }

   @KafkaHandler
   @KafkaHandlerDescription("Принимает  Subordinate. Находится в
DemoListener. Ничего не возвращает.")
   @KafkaHandlerTags(tags = {"demo"})
   public void listenGroupFoo(@Payload Subordinate message) {
      System.out.println("Received Message in group foo: " + message);
   }
}
```



Библиотека AxenAPI. Принцип работы

```
@KafkaListener(topics = {"demo_topic"}, groupId = "foo")
public class DemoListenerDifferencesTypes {

   @KafkaHandler
   @KafkaHandlerDescription("Принимает  ExampleMessage. Находится в
DemoListener. Ничего не возвращает.")
   @KafkaHandlerHeaders( headers = {
         @KafkaHandlerHeader(header = "header_1", required = true),
         @KafkaHandlerHeader(header = "header_2")
   })
   @KafkaSecured
   @KafkaHandlerTags(tags = {"demo"})
   public void listenGroupFoo(@Payload ExampleMessage message) {
      System.out.println("Received Message in group foo: " + message);
   }

   @KafkaHandler
   @KafkaHandlerDescription("Принимает  Subordinate. Находится в
DemoListener. Ничего не возвращает.")
   @KafkaHandlerTags(tags = {"demo"})
   public void listenGroupFoo(@Payload Subordinate message) {
      System.out.println("Received Message in group foo: " + message);
   }
}
```



Библиотека AxenAPI. Подключение

1. Добавить зависимость

```
// swagger for kafka
annotationProcessor "org.axenix:axenapi:1.0.0-SNAPSHOT"
implementation ("org.axenix:axenapi:1.0.0-SNAPSHOT")
```

Библиотека AxenAPI. Подключение

1. Добавить зависимость

```
// swagger for kafka
annotationProcessor "org.axenix:axenapi:1.0.0-SNAPSHOT"
implementation ("org.axenix:axenapi:1.0.0-SNAPSHOT")
```

2. Добавить аннотации

```
@KafkaListener(topics = {"demo_topic"}, groupId = "foo")
public class DemoListenerDifferencesTypes {

   @KafkaHandler
   @KafkaHandlerDescription("Принимает  ExampleMessage.
Находится в DemoListener. Ничего не возвращает.")

   @KafkaHandlerHeaders( headers = {
         @KafkaHandlerHeader(header = "header_1", required = true),
         @KafkaHandlerHeader(header = "header_2")
   })
   @KafkaSecured
   @KafkaHandlerTags(tags = {"demo"})
   public void listenGroupFoo(@Payload ExampleMessage message) {
      System.out.println("Received Message in group foo: " + message);
   }

   @KafkaHandler
   @KafkaHandlerDescription("Принимает  Subordinate. Находится в
DemoListener. Ничего не возвращает.")
   @KafkaHandlerTags(tags = {"demo"})
   public void listenGroupFoo(@Payload Subordinate message) {
      System.out.println("Received Message in group foo: " + message);
   }
}
```

Библиотека AxenAPI. Подключение

1. Добавить зависимость

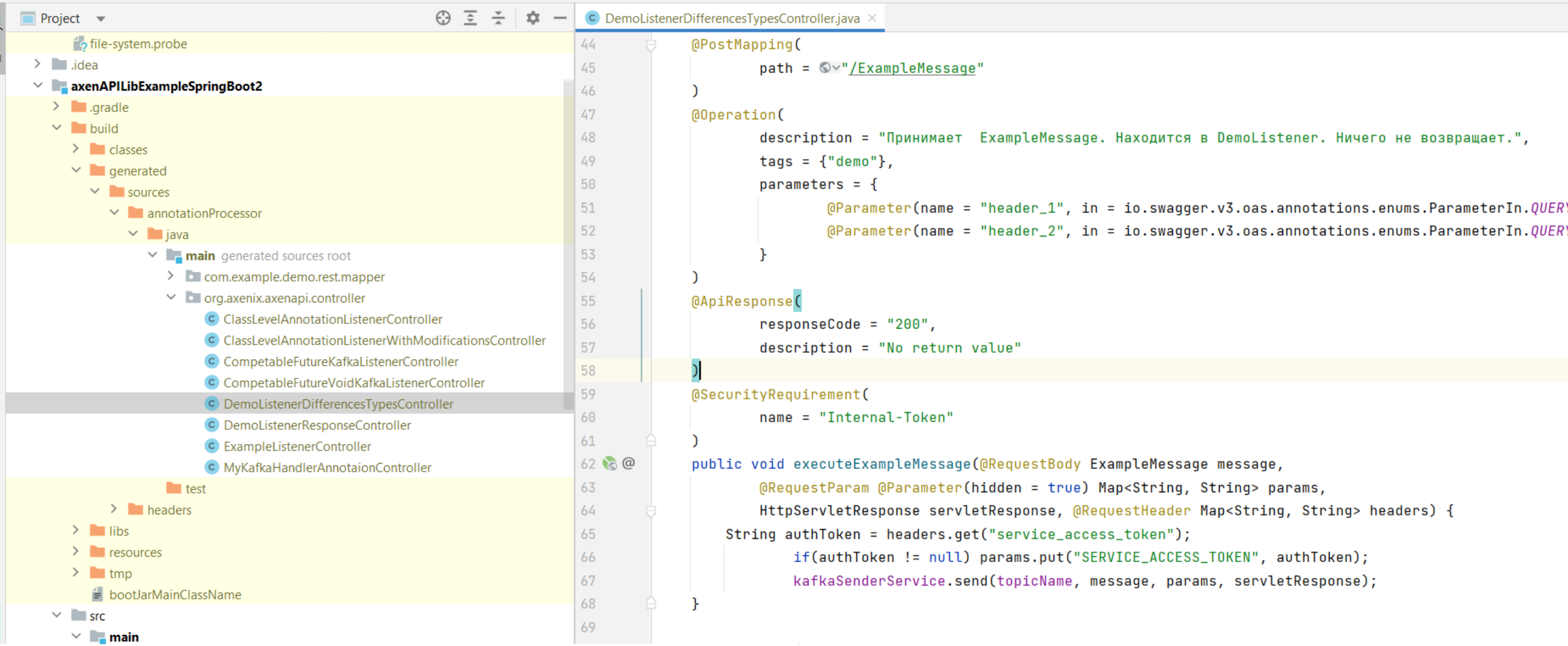
```
// swagger for kafka
annotationProcessor "org.axenix:axenapi:1.0.0-SNAPSHOT"
implementation ("org.axenix:axenapi:1.0.0-SNAPSHOT")
```

2. Добавить аннотации

```
@KafkaListener(topics = {"demo_topic"}, groupId = "foo")
public class DemoListenerDifferencesTypes {

   @KafkaHandler
   @KafkaHandlerDescription("Принимает  ExampleMessage.
Находится в DemoListener. Ничего не возвращает.")

   @KafkaHandlerHeaders( headers = {
         @KafkaHandlerHeader(header = "header_1", required = true),
         @KafkaHandlerHeader(header = "header_2")
   })
   @KafkaSecured
   @KafkaHandlerTags(tags = {"demo"})
   public void listenGroupFoo(@Payload ExampleMessage message) {
      System.out.println("Received Message in group foo: " + message);
   }

   @KafkaHandler
   @KafkaHandlerDescription("Принимает  Subordinate. Находится в
DemoListener. Ничего не возвращает.")
   @KafkaHandlerTags(tags = {"demo"})
   public void listenGroupFoo(@Payload Subordinate message) {
      System.out.println("Received Message in group foo: " + message);
   }
}
```

3. Собрать проект

Библиотека AxenAPI. Подключение. Результат



Библиотека AxenAPI. Подключение. Результат

AxenAPI подключен – генерируются контроллеры.

Подключаем Swagger-UI в проект. Для этого необходимо добавить зависимость:

implementation 'org.springdoc:springdoc-openapi-ui:1.6.13'

Далее дан пример кода, который описывает:

- две схемы авторизации
- две группы API (kafka и rest)

```
@Configuration
public class OpenApiConfiguration {
   @Bean
   public OpenAPI customOpenAPI() {
      return new OpenAPI()
            .components(new Components().addSecuritySchemes("Public-Bearer-Jwt",
                  new SecurityScheme().type(SecurityScheme.Type.HTTP)
                        .scheme("bearer")
                        .bearerFormat("JWT")
                        .in(SecurityScheme.In.HEADER))
                  .addSecuritySchemes("Internal-Token",
                        new SecurityScheme().type(SecurityScheme.Type.APIKEY)
                              .bearerFormat("JWT")
                              .in(SecurityScheme.In.HEADER)
                              .name("SERVICE_ACCESS_TOKEN"))
            ).info(new Info().title("App API").version("snapshot"));
   }
   @Bean
   GroupedOpenApi kafkaApis() {
      return GroupedOpenApi.builder().group("kafka").pathsToMatch("/**/kafka/**").build();
   }
   @Bean
   GroupedOpenApi restApis() {
      return GroupedOpenApi.builder().group("rest").pathsToMatch("/**/users/**").build();
   }
}
```

Библиотека AxenAPI. Подключение. Результат

AxenAPI подключен – генерируются контроллеры.

Подключаем Swagger-UI в проект. Для этого необходимо добавить зависимость:

implementation 'org.springdoc:springdoc-openapi-ui:1.6.13'

Далее дан пример кода, который описывает:

- две схемы авторизации
- две группы API (kafka и rest)

```
@Configuration
public class OpenApiConfiguration {
   @Bean
   public OpenAPI customOpenAPI() {
      return new OpenAPI()
            .components(new Components().addSecuritySchemes("Public-Bearer-Jwt",
                  new SecurityScheme().type(SecurityScheme.Type.HTTP)
                        .scheme("bearer")
                        .bearerFormat("JWT")
                        .in(SecurityScheme.In.HEADER))
                  .addSecuritySchemes("Internal-Token",
                        new SecurityScheme().type(SecurityScheme.Type.APIKEY)
                              .bearerFormat("JWT")
                              .in(SecurityScheme.In.HEADER)
                              .name("SERVICE_ACCESS_TOKEN"))
            ).info(new Info().title("App API").version("snapshot"));
   }
   @Bean
   GroupedOpenApi kafkaApis() {
      return GroupedOpenApi.builder().group("kafka").pathsToMatch("/**/kafka/**").build();
   }
   @Bean
   GroupedOpenApi restApis() {
      return GroupedOpenApi.builder().group("rest").pathsToMatch("/**/users/**").build();
   }
}
```

Библиотека AxenAPI. Подключение. Результат

AxenAPI подключен – генерируются контроллеры.

Подключаем Swagger-UI в проект. Для этого необходимо добавить зависимость:

implementation 'org.springdoc:springdoc-openapi-ui:1.6.13'

Далее дан пример кода, который описывает:

- две схемы авторизации
- две группы API (kafka и rest)

```
@Configuration
public class OpenApiConfiguration {
   @Bean
   public OpenAPI customOpenAPI() {
      return new OpenAPI()
            .components(new Components().addSecuritySchemes("Public-Bearer-Jwt",
                  new SecurityScheme().type(SecurityScheme.Type.HTTP)
                        .scheme("bearer")
                        .bearerFormat("JWT")
                        .in(SecurityScheme.In.HEADER))
                  .addSecuritySchemes("Internal-Token",
                        new SecurityScheme().type(SecurityScheme.Type.APIKEY)
                              .bearerFormat("JWT")
                              .in(SecurityScheme.In.HEADER)
                              .name("SERVICE_ACCESS_TOKEN"))
            ).info(new Info().title("App API").version("snapshot"));
   }
   @Bean
   GroupedOpenApi kafkaApis() {
      return GroupedOpenApi.builder().group("kafka").pathsToMatch("/**/kafka/**").build();
   }
   @Bean
   GroupedOpenApi restApis() {
      return GroupedOpenApi.builder().group("rest").pathsToMatch("/**/users/**").build();
   }
}
```

Библиотека AxenAPI. Подключение. Результат

AxenAPI подключен – генерируются контроллеры.

Подключаем Swagger-UI в проект. Для этого необходимо добавить зависимость:

implementation 'org.springdoc:springdoc-openapi-ui:1.6.13'

Далее дан пример кода, который описывает:

- две схемы авторизации
- две группы API (kafka и rest)

```
@Configuration
public class OpenApiConfiguration {
   @Bean
   public OpenAPI customOpenAPI() {
      return new OpenAPI()
            .components(new Components().addSecuritySchemes("Public-Bearer-Jwt",
                  new SecurityScheme().type(SecurityScheme.Type.HTTP)
                        .scheme("bearer")
                        .bearerFormat("JWT")
                        .in(SecurityScheme.In.HEADER))
                  .addSecuritySchemes("Internal-Token",
                        new SecurityScheme().type(SecurityScheme.Type.APIKEY)
                              .bearerFormat("JWT")
                              .in(SecurityScheme.In.HEADER)
                              .name("SERVICE_ACCESS_TOKEN"))
            ).info(new Info().title("App API").version("snapshot"));
   }
   @Bean
   GroupedOpenApi kafkaApis() {
      return GroupedOpenApi.builder().group("kafka").pathsToMatch("/**/kafka/**").build();
   }
   @Bean
   GroupedOpenApi restApis() {
      return GroupedOpenApi.builder().group("rest").pathsToMatch("/**/users/**").build();
   }
}
```

Библиотека AxenAPI. Подключение. Результат

AxenAPI подключен – генерируются контроллеры.

Подключаем Swagger-UI в проект. Для этого необходимо добавить зависимость:

implementation 'org.springdoc:springdoc-openapi-ui:1.6.13'

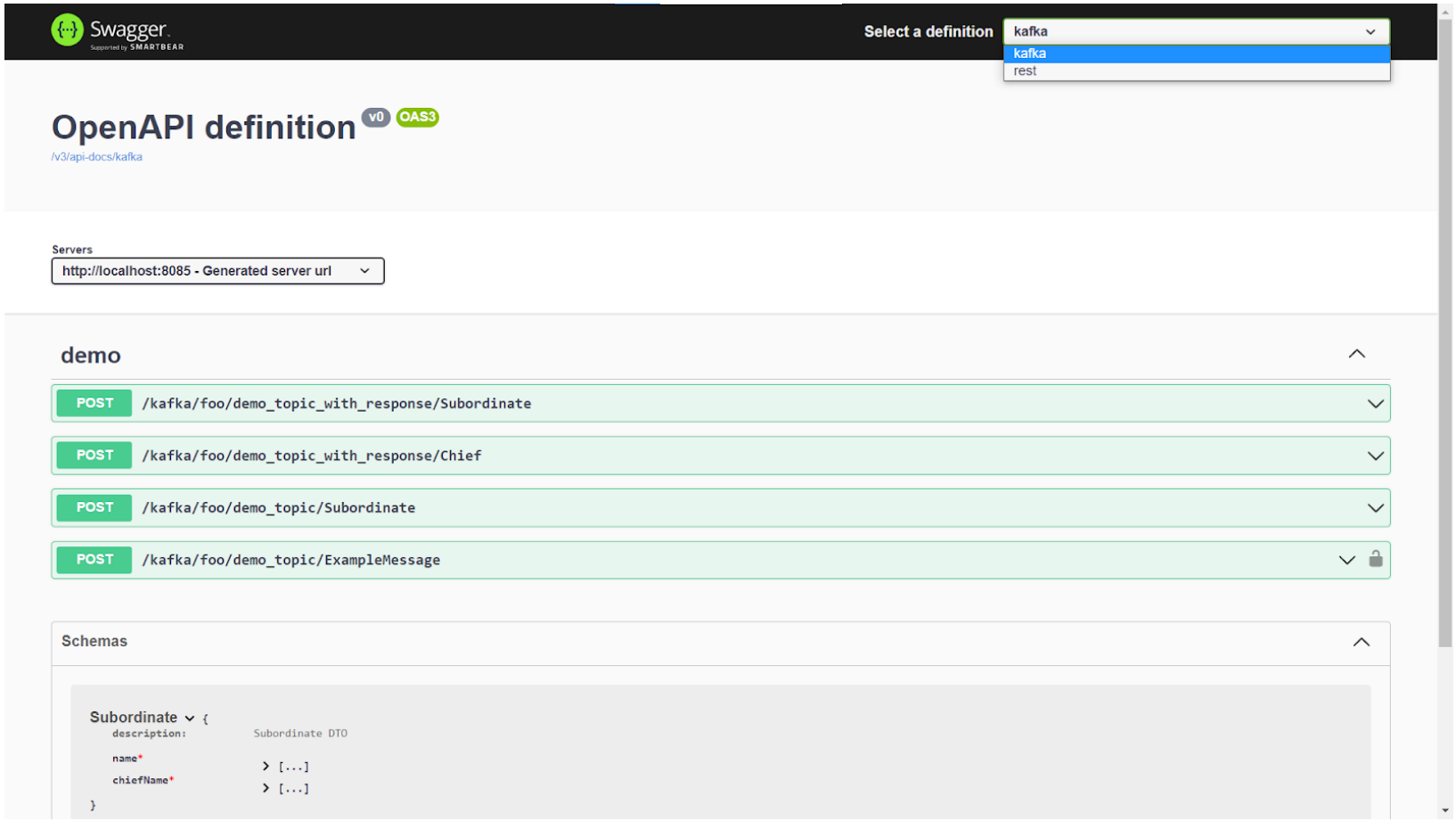
Далее дан пример кода, который описывает:

- две схемы авторизации
- две группы API (kafka и rest)

```
@Configuration
public class OpenApiConfiguration {
   @Bean
   public OpenAPI customOpenAPI() {
      return new OpenAPI()
            .components(new Components().addSecuritySchemes("Public-Bearer-Jwt",
                  new SecurityScheme().type(SecurityScheme.Type.HTTP)
                        .scheme("bearer")
                        .bearerFormat("JWT")
                        .in(SecurityScheme.In.HEADER))
                  .addSecuritySchemes("Internal-Token",
                        new SecurityScheme().type(SecurityScheme.Type.APIKEY)
                              .bearerFormat("JWT")
                              .in(SecurityScheme.In.HEADER)
                              .name("SERVICE_ACCESS_TOKEN"))
            ).info(new Info().title("App API").version("snapshot"));
   }
   @Bean
   GroupedOpenApi kafkaApis() {
      return GroupedOpenApi.builder().group("kafka").pathsToMatch("/**/kafka/**").build();
   }
   @Bean
   GroupedOpenApi restApis() {
      return GroupedOpenApi.builder().group("rest").pathsToMatch("/**/users/**").build();
   }
}
```

Библиотека AxenAPI. Подключение. Результат

Внимательно рассмотрим результат подключения Swagger UI.



Swagger
Supported by SMARTBEAR

Select a definition: kafka
- kafka
- rest

OpenAPI definition v0 OAS3

/v3/api-docs/kafka

Servers
http://localhost:8085 - Generated server url

demo

- POST /kafka/foo/demo_topic_with_response/Subordinate
- POST /kafka/foo/demo_topic_with_response/Chief
- POST /kafka/foo/demo_topic/Subordinate
- POST /kafka/foo/demo_topic/ExampleMessage

Schemas

Subordinate {
description: Subordinate DTO
name* > [...]
chiefName* > [...]
}

Swagger
Supported by SMARTBEAR

Select a definition: kafka
- kafka
- rest

OpenAPI definition v0 OAS3

/v3/api-docs/kafka

Разделение на группы

Servers
http://localhost:8085 - Generated server url

demo

- POST /kafka/foo/demo_topic_with_response/Subordinate
- POST /kafka/foo/demo_topic_with_response/Chief
- POST /kafka/foo/demo_topic/Subordinate
- POST /kafka/foo/demo_topic/ExampleMessage

Schemas

Subordinate {
description: Subordinate DTO
name* > [...]
chiefName* > [...]
}

Swagger
Supported by SMARTBEAR

Select a definition: kafka
- kafka
- rest

OpenAPI definition v0 OAS3

/v3/api-docs/kafka

Разделение на группы

Servers
http://localhost:8085 - Generated server url

Интерфейсы для отправки событий в топик

demo

- POST /kafka/foo/demo_topic_with_response/Subordinate
- POST /kafka/foo/demo_topic_with_response/Chief
- POST /kafka/foo/demo_topic/Subordinate
- POST /kafka/foo/demo_topic/ExampleMessage

Schemas

Subordinate {
description: Subordinate DTO
name* > [...]
chiefName* > [...]
}

Swagger
Supported by SMARTBEAR

Select a definition: kafka
- kafka
- rest

OpenAPI definition v0 OAS3

/v3/api-docs/kafka

Разделение на группы

Servers
http://localhost:8085 - Generated server url

Интерфейсы для отправки событий в топик

demo

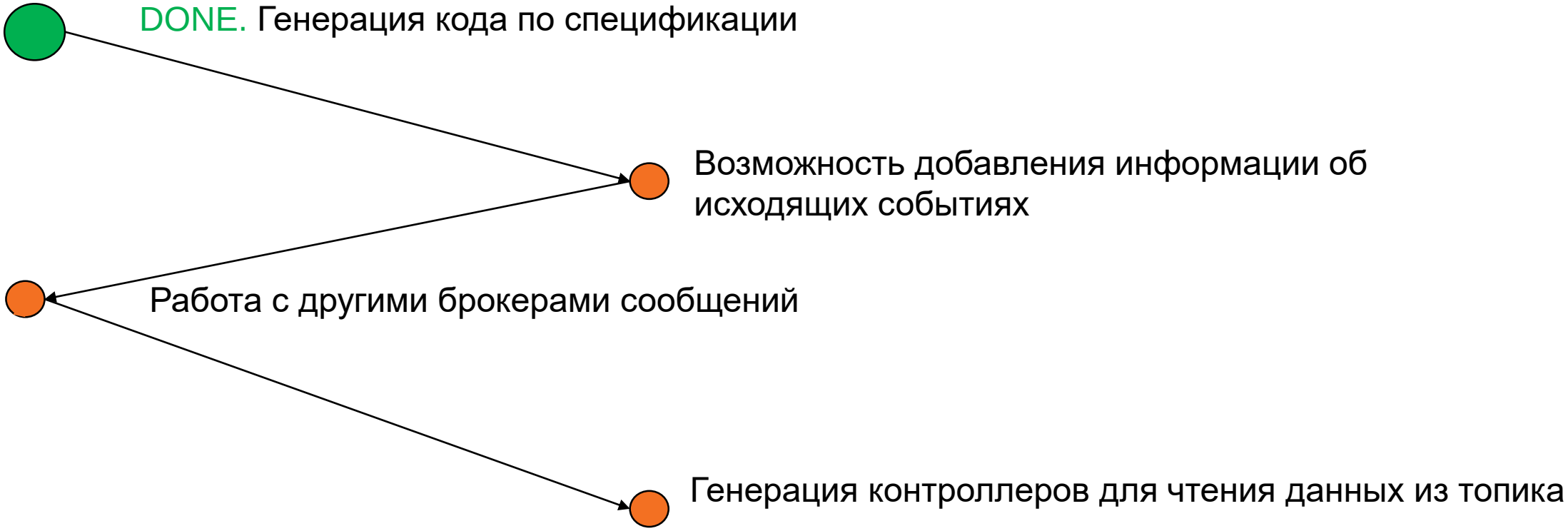
- POST /kafka/foo/demo_topic_with_response/Subordinate
- POST /kafka/foo/demo_topic_with_response/Chief
- POST /kafka/foo/demo_topic/Subordinate
- POST /kafka/foo/demo_topic/ExampleMessage

Schemas

Модели получаемых событий

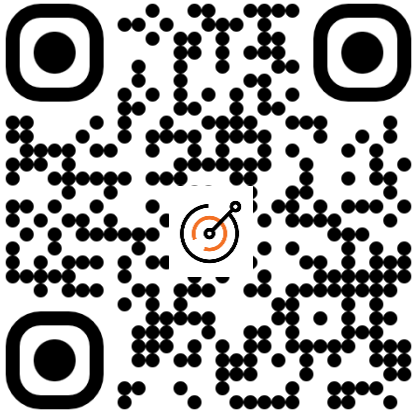
Subordinate {
  description: Subordinate DTO
  name* > [...]
  chiefName* > [...]
}

AxenAPI. Планы развития



Выводы

- Участвуйте в тестировании и разработке AxenAPI ☺
- OpenAPI Generator – гибкий инструмент. Трансляцию из OpenAPI в Java можно сделать в любом удобном для вас виде.
- Генерируйте всё, что можно сгенерировать.



https://github.com/AxenAPI

Axenix Habr Блог

axenix
innovation

Спасибо!

О компании ООО «Аксеникс Инновации» – российская компания, входит в группу **«AXENIX»**, резидент Инновационного центра «Сколково», ведущая научно-технические исследования в области информационных технологий и предоставляющая на базе собственных разработок широкий спектр профессиональных услуг, направленных на цифровизацию бизнеса.

Подробнее на axenix-innovation.pro