



OpenAPI и как его можно применить для Kafka

OpenAPI, API First, Kafka и OpenAPI

14/10/2023





План доклада

Что такое API First

Что такое OpenAPI

Кастомизция шаблонов из OpenAPI-generator

Проблемы при разработке API с использованием Kafka

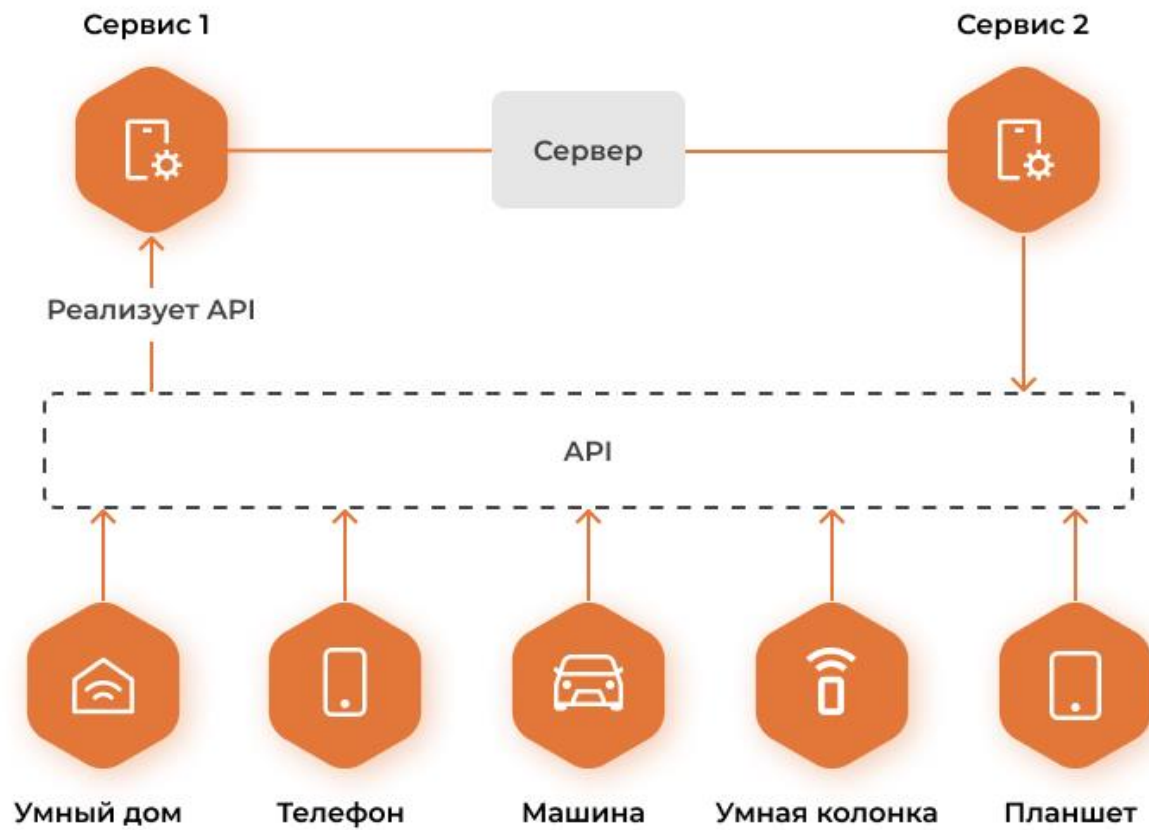
API First и Kafka

АхенAPI:
Генерация кода обработчика и генератора событий в Kafka

АхенAPI: Генерация документации API сервиса, использующего Kafka



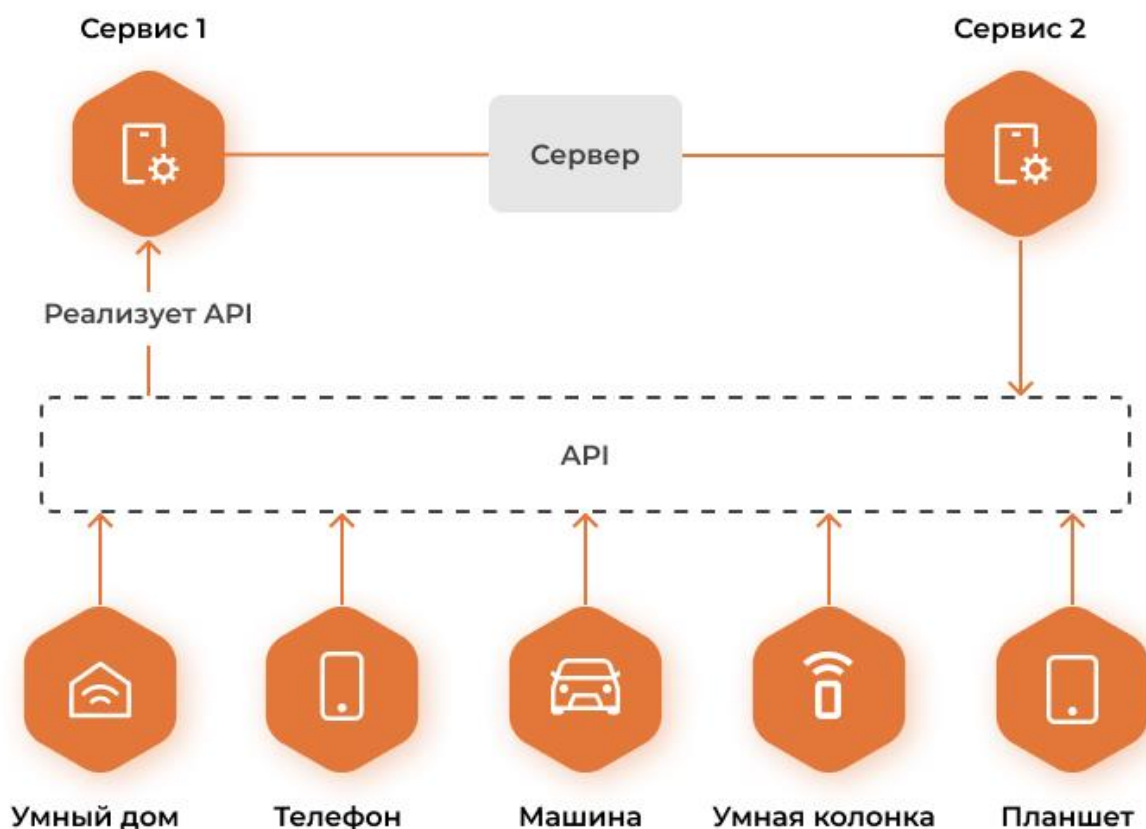
Что такое API First



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



Что такое API First



API – важнейшая 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.

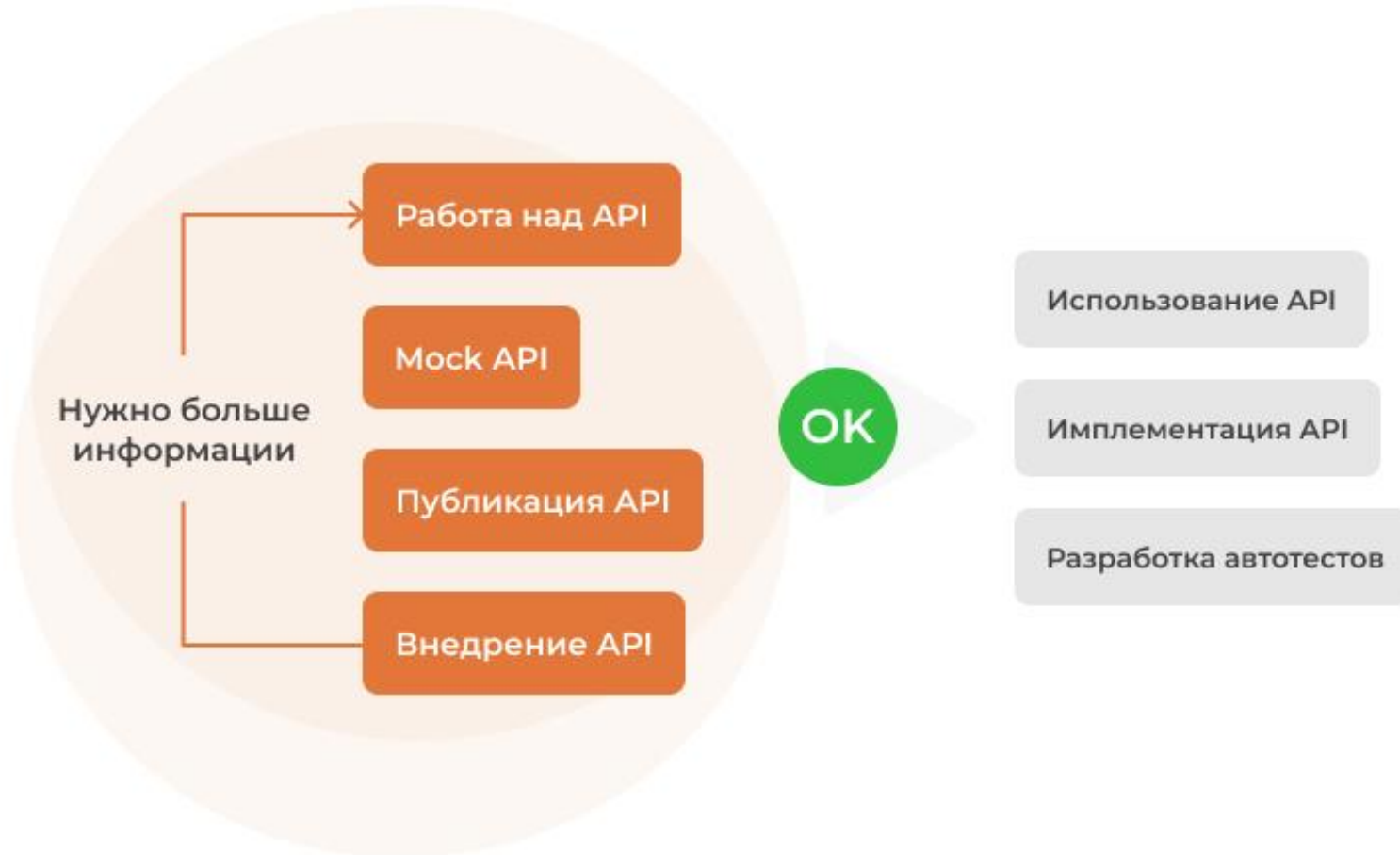


Проблема согласования протоколов



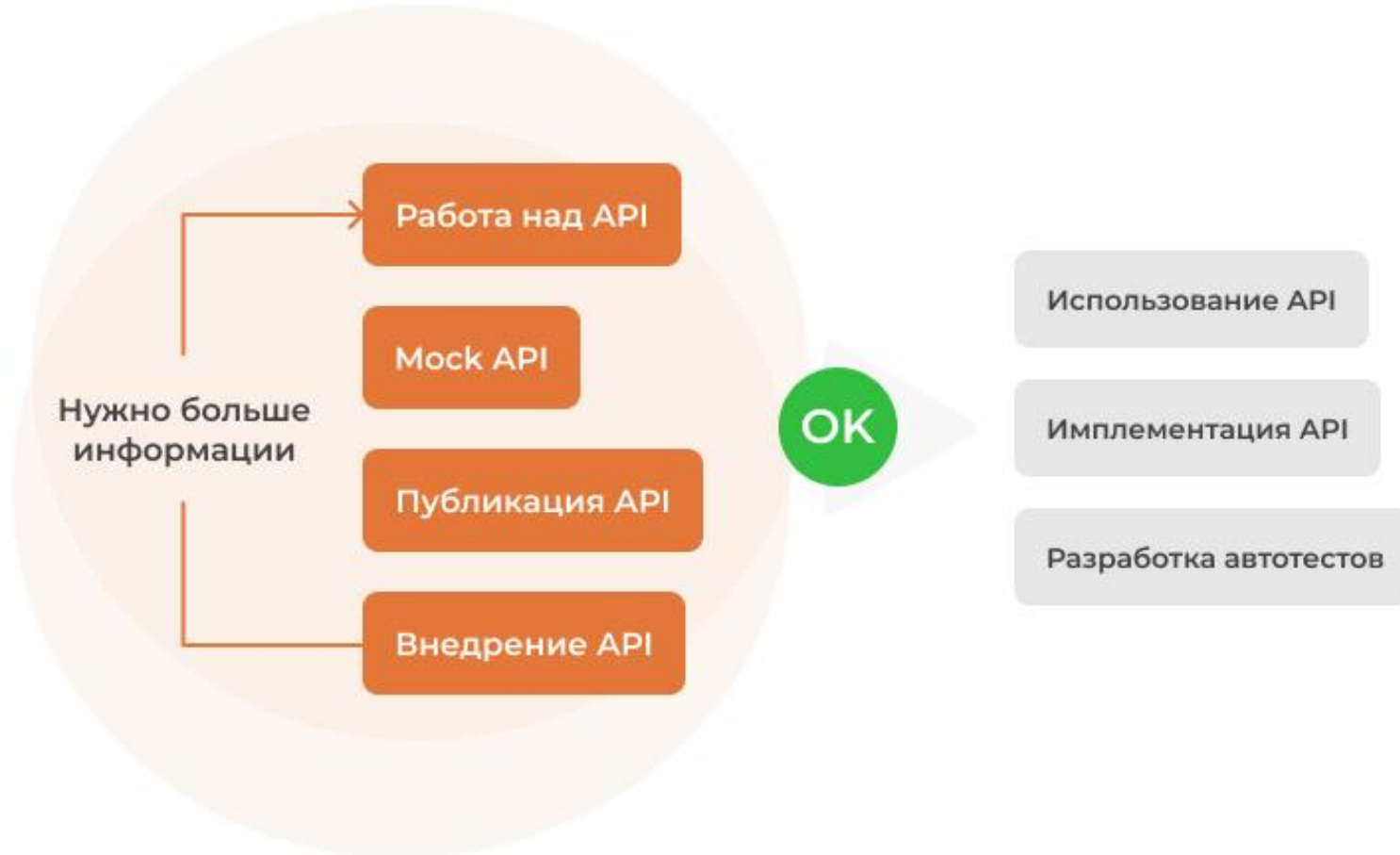


Зачем нужен подход API First





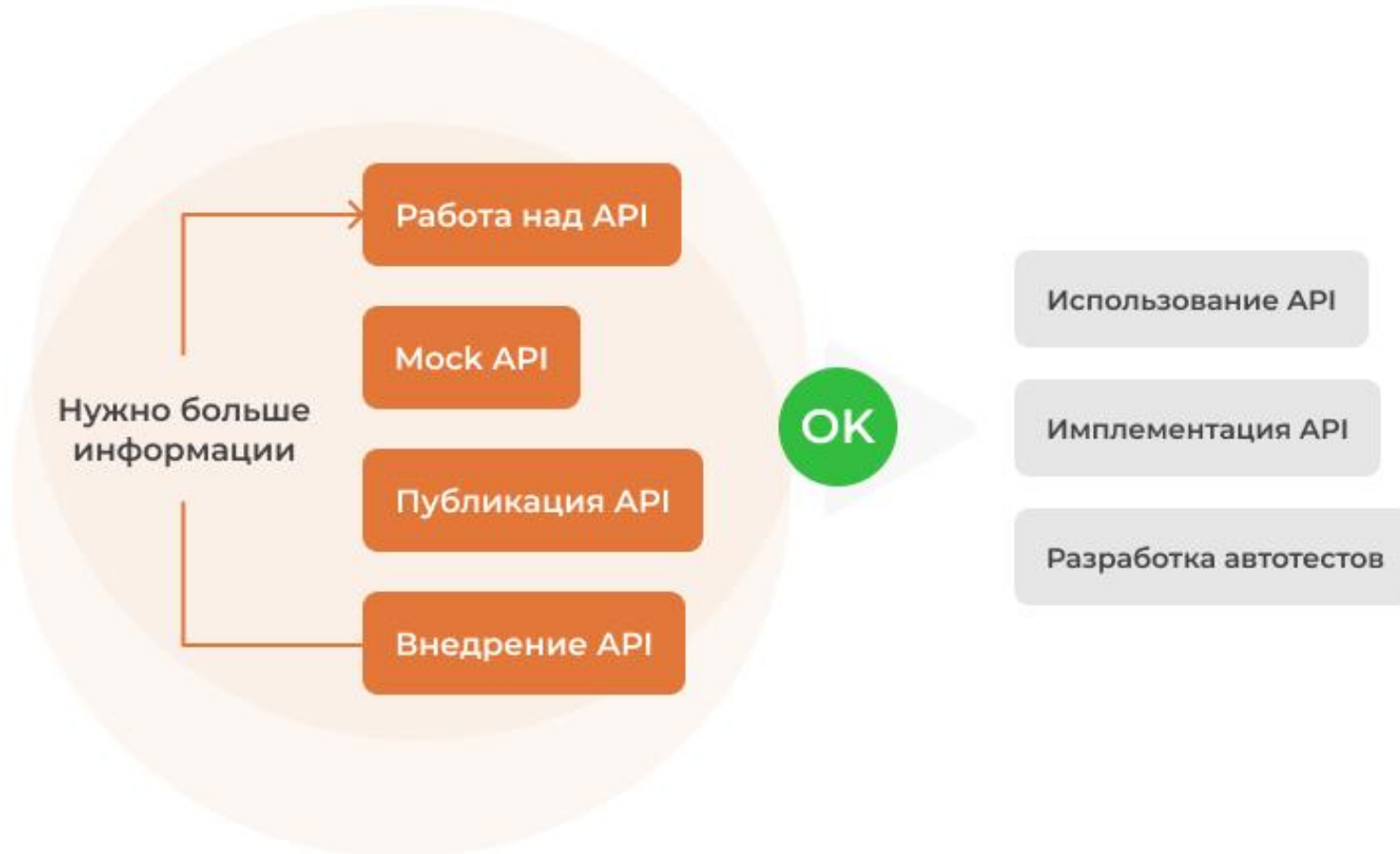
Зачем нужен подход API First



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



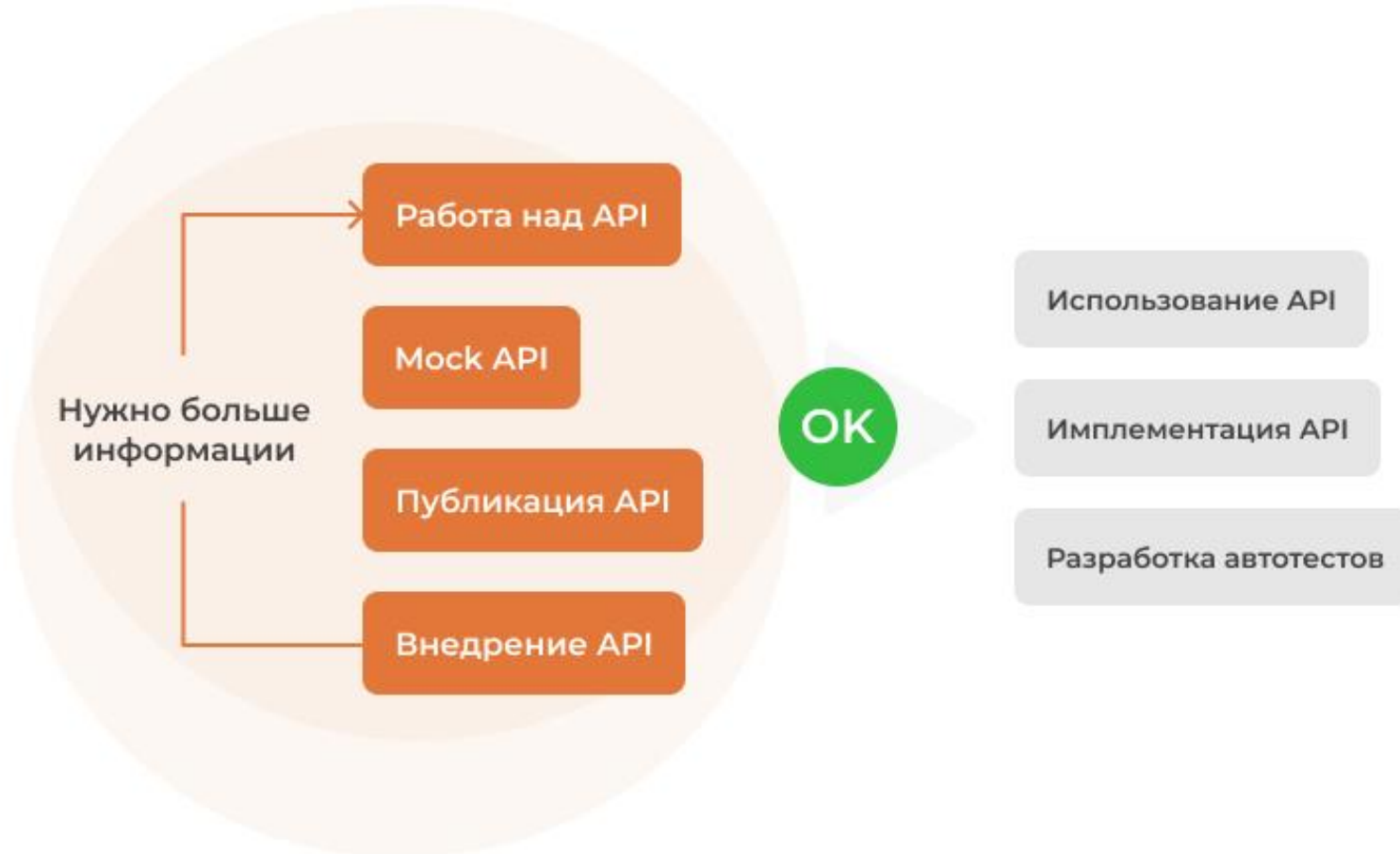
Зачем нужен подход API First



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



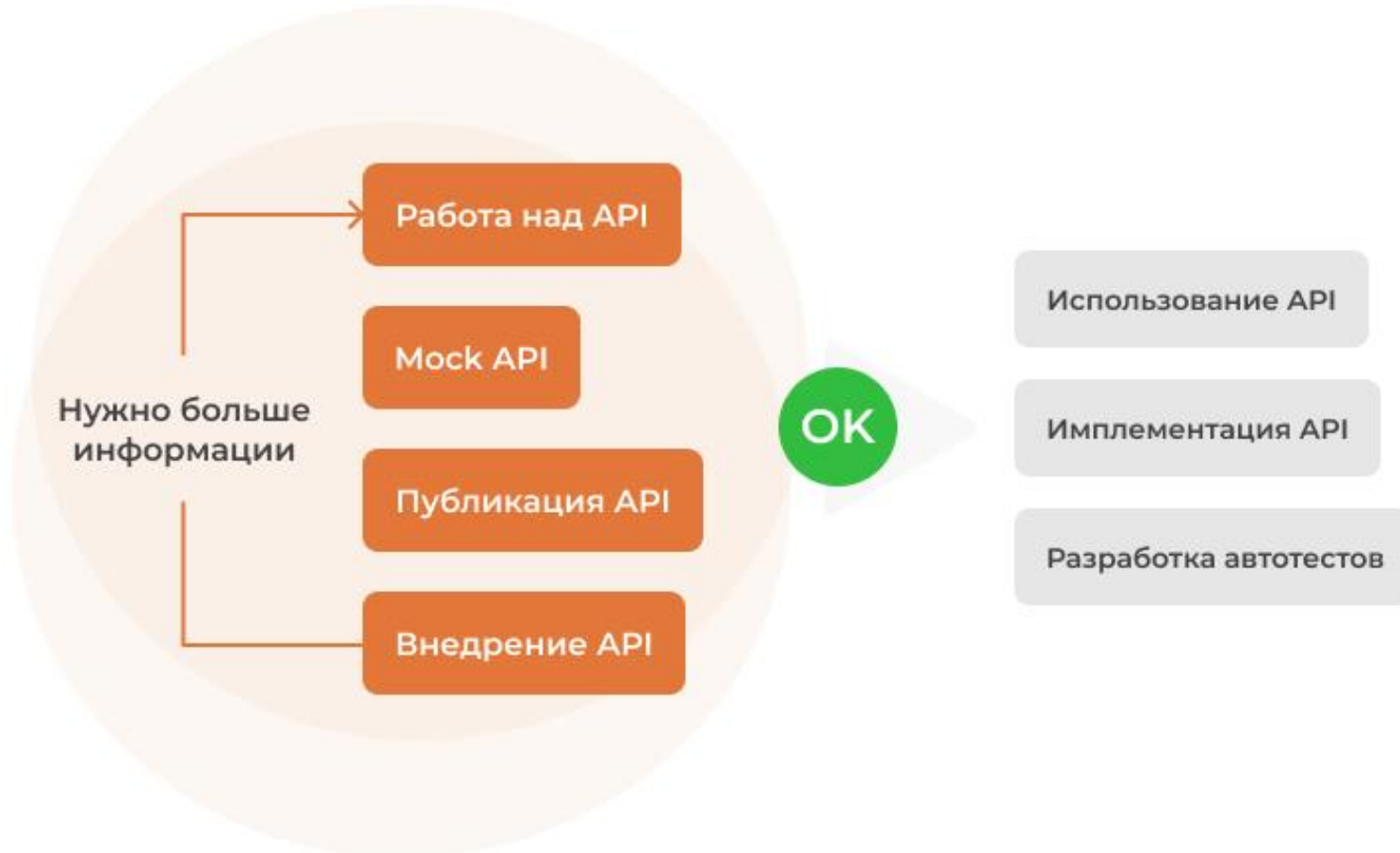
Зачем нужен подход API First



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.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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



The screenshot shows the SwaggerUI interface for an API. At the top, there is a 'Swagger' logo and a 'Select a definition' dropdown menu with 'kafka' selected. Below this, the API is identified as 'App API' with a 'snapshot' tag and 'OAS 3.0' version. The URL is '/v3/api-docs/kafka'. There is a 'Servers' dropdown menu showing 'http://localhost:8085 - Generated server url' and an 'Authorize' button. The interface lists two API endpoints: 'test-1-controller' with a POST method at '/kafka/test1/test1/Subordinate', and 'test_tag' with a POST method at '/kafka/test1/test1/ExampleMessage'. Both endpoints have expand/collapse and lock icons.

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





OpenAPI Generator

```
.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:
генерація 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

The image shows a development environment with an IDE on the left and a web browser on the right. The IDE displays the project structure for 'openapi-generator', highlighting the 'resources' directory. The browser shows the GitHub repository page for 'openapi-generator/src/main/resources/JavaSpring', listing various Mustache templates and their associated commit messages.

Project Structure (IDE):

- src
 - main
 - java
 - com.example.customvalidation
 - validation
 - annotations
 - Capitalized
 - validator
 - CapitalizedValidator
 - CustomValidationApplication
 - resources
 - test
 - templateDir
 - api.mustache
 - beanValidationCore.mustache
 - model.mustache
 - .gitignore
 - api.yaml
 - build.gradle
 - gradlew
 - gradlew.bat



OpenAPI Generator. Кастомизация

The image shows a screenshot of an IDE (IntelliJ IDEA) and a web browser. The IDE window on the left displays the project structure for 'openapi-generator/modules/openapi-generator'. The 'validation' folder is highlighted with an orange box, containing sub-folders 'annotations' and 'validator'. The 'annotations' folder contains '@ Capitalized' and 'CapitalizedValidator'. The 'validator' folder contains 'CapitalizedValidator'. The 'resources' folder contains 'api.yaml', which is also highlighted with an orange box. Other files in the project include 'build.gradle', 'gradlew', and 'gradlew.bat'. The web browser window on the right shows the GitHub repository page for 'openapi-generator/modules/openapi-generator/src/main/resources/JavaSpring'. It displays a list of files with their commit messages, such as 'additionalModelTypeAnnotations.mustache' and 'additionalOneOfTypeAnnotations.mustache'. The commit messages are truncated and include links to GitHub issues.

File	Commit Message
additionalModelTypeAnnotations.mustache	[Java Spring OAS3] Minor fixes and general improvements (OpenAPITools#...)
additionalOneOfTypeAnnotations.mustache	[Java] fix additional annotations for oneOf interfaces (OpenAPITools#...)
additionalProperties.mustache	fix: OpenAPITools#1466 additionalProperties works now in spring gener...
allowableValues.mustache	[JAVA] new Feature interface: Documentation Provider and Annotation L...
api.mustache	Add annotations to the operation - case permission validation - Fix 1...
apiController.mustache	[Spring] fix Paginated without params (OpenAPITools#15315) (fix OpenA...
apiDelegate.mustache	Add validation.constraints package import to delegates when using bea...
apiException.mustache	[Java] configurable Javax/Jakarta package (OpenAPITools#14310)
apiOriginFilter.mustache	[Java] configurable Javax/Jakarta package (OpenAPITools#14310)
apiResponseMessage.mustache	[Java] configurable Javax/Jakarta package (OpenAPITools#14310)
apiUtil.mustache	[Java] configurable Javax/Jakarta package (OpenAPITools#14310)
beanValidation.mustache	[java] No @NotNull annotation for readOnly (required) attributes - fixes
beanValidationBodyParams.mustache	added JavaSpring beanValidation for RequestBody parameters fix OpenAP...
beanValidationCore.mustache	Update beanValidationCore.mustache (OpenAPITools#13631) (fix OpenAPIT...
beanValidationPathParams.mustache	renaming for openapi-generator
beanValidationQueryParams.mustache	Fix bean validation for type Optional (OpenAPITools#8708)
bodyParams.mustache	added JavaSpring beanValidation for RequestBody parameters fix OpenAP...
model.mustache	fix: #1466 additionalProperties works now in spring generator (#11572) (



OpenAPI Generator. Кастомизация

The image shows a development environment with an IDE on the left and a web browser on the right. The IDE displays the project structure for 'openapi-generator', highlighting the 'validation' sub-package under 'com.example.customvalidation'. The web browser shows the GitHub repository page for 'openapi-generator/src/main/resources/JavaSpring', listing various Mustache templates. Orange boxes and arrows highlight the relationship between the IDE files and the browser list.

Project Structure (IDE):

- main
 - java
 - com.example.customvalidation
 - validation
 - annotations
 - Capitalized
 - validator
 - CapitalizedValidator
 - resources
 - test
 - templateDir
 - api.mustache
 - beanValidationCore.mustache
 - model.mustache

Files List (Browser):

File Name	Description
additionalModelTypeAnnotations.mustache	[Java Spring OAS3] Minor fixes and general improvements (OpenAPITools#...)
additionalOneOfTypeAnnotations.mustache	[Java] fix additional annotations for oneOf interfaces (OpenAPITools#...)
additionalProperties.mustache	fix: OpenAPITools#1466 additionalProperties works now in spring gener...
allowableValues.mustache	[JAVA] new Feature interface: Documentation Provider and Annotation L...
api.mustache	Add annotations to the operation - case permission validation - Fix 1...
apiController.mustache	[Spring] fix Paginated without params (OpenAPITools#15315) (fix OpenA...
apiDelegate.mustache	Add validation.constraints package import to delegates when using bea...
apiException.mustache	[Java] configurable Javax/Jakarta package (OpenAPITools#14310)
apiOriginFilter.mustache	[Java] configurable Javax/Jakarta package (OpenAPITools#14310)
apiResponseMessage.mustache	[Java] configurable Javax/Jakarta package (OpenAPITools#14310)
apiUtil.mustache	[Java] configurable Javax/Jakarta package (OpenAPITools#14310)
beanValidation.mustache	[java] No @NotNull annotation for readOnly (required) attributes - fixes
beanValidationBodyParams.mustache	added JavaSpring beanValidation for RequestBody parameters fix OpenAP...
beanValidationCore.mustache	Update beanValidationCore.mustache (OpenAPITools#13631) (fix OpenAPIT...
beanValidationPathParams.mustache	renaming for openapi-generator
beanValidationQueryParams.mustache	Fix bean validation for type Optional (OpenAPITools#8708)
bodyParams.mustache	added JavaSpring beanValidation for RequestBody parameters fix OpenAP...
model.mustache	fix: #1466 additionalProperties works now in spring generator (#11572) (

Annotations:

- The 'validation' folder in the IDE is highlighted with an orange box.
- The 'templateDir' folder in the IDE is highlighted with an orange box, with the word 'Copy' written in orange next to it.
- The 'api.yaml' file in the IDE is highlighted with an orange box.
- The 'api.mustache', 'beanValidationCore.mustache', 'beanValidationPathParams.mustache', and 'model.mustache' files in the browser list are highlighted with orange boxes.
- Orange arrows point from the IDE boxes to the corresponding files in the browser list.

```
84 components:
85   schemas:
86     Pet:
87       required:
88         - id
89         - name
90       properties:
91         id:
```

1. Добавить в спецификацию ([api.yaml](#)) property:

```
94     minimum: 5
95     maximum: 10
96     name:
97     type: string
```

x-constraints: "@Capitalized(required = true)"

```
99     tag:
100     type: string
101   Pets:
102     type: array
```

```
1 {{{vendorExtensions.x-constraints}}}
```

```
2 {{#pattern}}{{^isArray}}@Pattern(regexp = "{{{pattern}}}"{{/
3 minLength && maxLength set
4 }}{{#minLength}}{{#maxLength}}@Size(min = {{minLength}}, max =
5 minLength set, maxLength not
6 }}{{#minLength}}{{^maxLength}}@Size(min = {{minLength}}) {{/max
7 minlength not set, maxlength set
```

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

```
9 @Size: minItems && maxItems set
10 }}{{#minItems}}{{#maxItems}}@Size(min = {{minItems}}, max = {{m
11 @Size: minItems set, maxItems not
12 }}{{#minItems}}{{^maxItems}}@Size(min = {{minItems}}) {{/maxItem
13 @Size: minItems not set && maxItems set
14 }}{{^minItems}}{{#maxItems}}@Size(max = {{.}}) {{/maxItems}}{{/
```

```
1 /**
2  * NOTE: This class is auto generated by OpenAPI Generator (https://openapi-generator.tech) ({{{gen
3  * https://openapi-generator.tech
4  * Do not edit the class manually.
5  */
6 package {{package}};
7
8 {{#imports}}import {{import}};
9 {{/imports}}
```

import com.example.customvalidation.validation.annotations.*;

```
11 {{#swagger2AnnotationLibrary}}
12 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}}  
11 {{#serializableModel}}
```



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

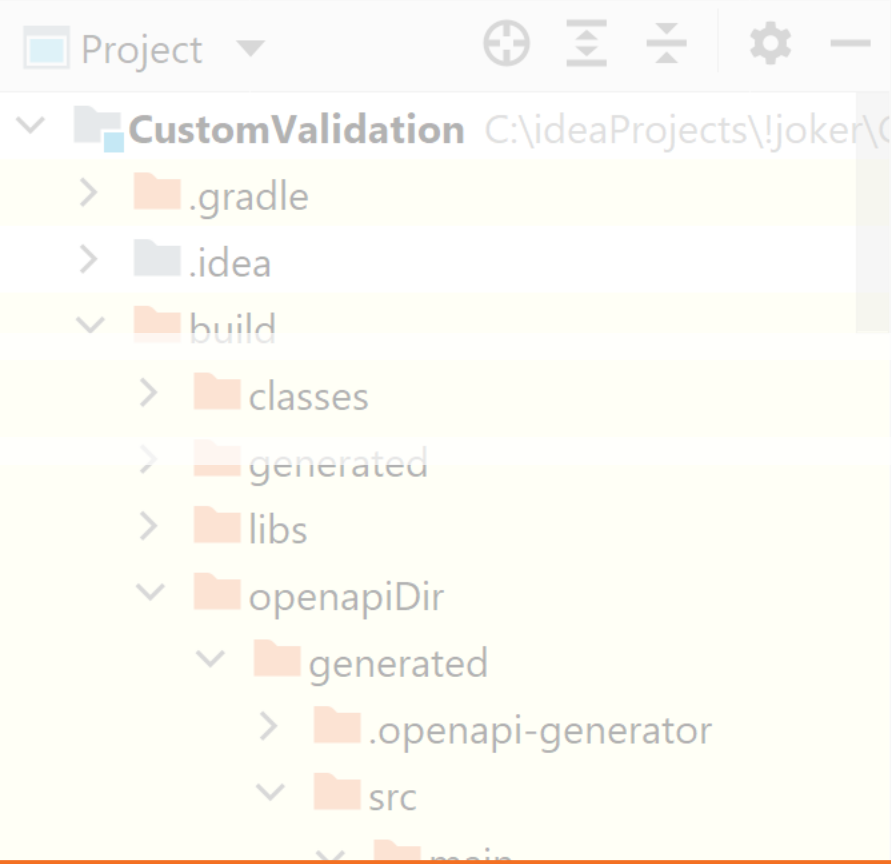
```
41
42 ▶ openApiGenerate {
43     generatorName = "spring"
44     outputDir = getBuildDir().absolutePath + "/openapiDir/generated"
45     inputSpec = getProjectDir().absolutePath + "/api.yaml"
46     globalProperties = [
47         apis: "",
48         models: "",
49         supportingFiles: 'ApiUtil.java'
50     ]

```

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

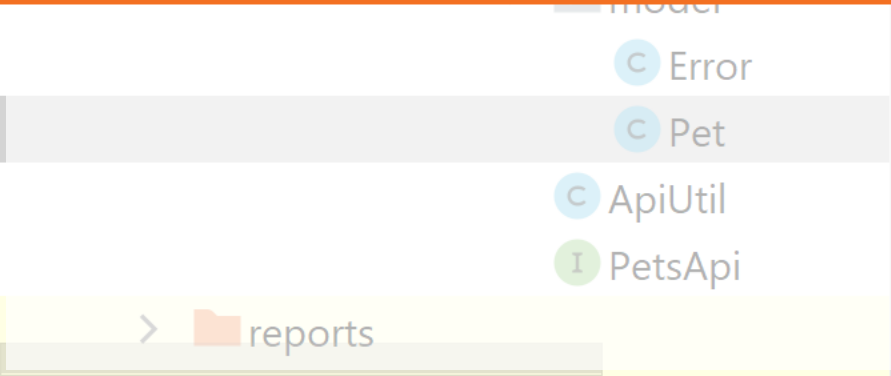
```
53     skipOverwrite = true
54     templateDir = getProjectDir().absolutePath + "/templateDir/"
55     configurations = [
56         useSpringBoot3: "true",
57         dateLibrary: "java8",
58         interfaceOnly: "true",
59         skipDefaultInterface: "false",
60         openApiNullable: "false",
61         generateSupportingFiles: "true"
62     ]
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }
101 }
102 }
103 }
104 }
105 }
106 }
107 }
108 }
109 }
110 }
111 }
112 }
113 }
114 }
115 }
116 }
117 }
118 }
119 }
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
1000 }

```



```
74
75 /**
76  * Get name
77  * @return name
78 */
79 @NotNull @Capitalized(required = true)
80
81 @Schema(name = "name", requiredMode = Schema.RequiredMode.R
82 @JsonProperty("name")
83 public String getName() { return name; }
86
87 public void setName(String name) { this.name = name; }
88
```

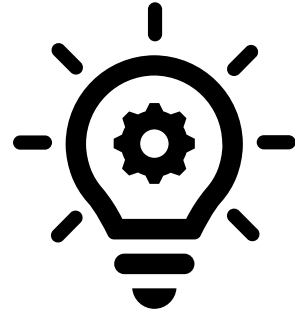
6. Запускаем **gradle openApiGenerate**. Находим сгенерированные модели. Видим кастомную валидацию.



```
94 }
95
96 /**
97  * Get tag
98  * @return tag
```



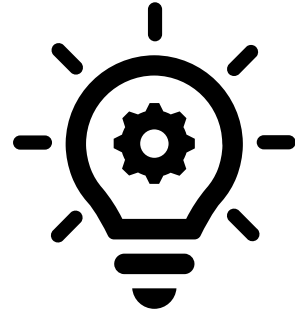
OpenAPI Generator. Кастомизация. Выводы



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.

The screenshot shows the Offset Explorer 2.3 interface. On the left, a tree view displays the cluster structure under 'localhost:29092', including 'Brokers', 'Topics', and 'Partitions'. The 'demo_topic' is expanded to show 'Partition 0'. The main pane shows message details for 'demo_topic' in 'Partition 0'. The 'Data' tab is active, displaying a table of messages with columns 'Offset', 'Key', and 'Value'. The 'Value' column contains JSON strings: `{"name":"string","chiefName":"string"}`. Below the table, the 'Headers' tab is active, showing a table with columns 'Key' and 'Value'. The 'Value' column contains the string `com.example.demo.kafka.model.Subordinate`. The status bar at the bottom indicates 'Ready [Messages = 3] [114 Bytes] [47 ms]'.

Offset	Key	Value
0		<code>{"name":"string","chiefName":"string"}</code>
1		<code>{"name":"string","chiefName":"string"}</code>
2		<code>{"name":"string","chiefName":"string"}</code>

Key	Value
<code>_TypeId_</code>	<code>com.example.demo.kafka.model.Subordinate</code>



Проблемы при работе с брокерами сообщений

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

The screenshot displays the Offset Explorer 2.3 interface. On the left, a tree view shows the cluster structure under 'localhost:29092', including 'Brokers', 'Topics', and 'Partitions'. The 'demo_topic' is expanded to show 'Partition 0'. The main pane shows message details for 'Partition 0' with a table of messages:

Offset	Key	Value
0		{"name":"string","chiefName":"string"}
1		{"name":"string","chiefName":"string"}
2		{"name":"string","chiefName":"string"}

Below the table, the 'Headers' section is visible, showing a key-value pair: '_TypeId_' with value 'com.example.demo.kafka.model.Subordinate'. The status bar at the bottom indicates 'Ready [Messages = 3] [114 Bytes] [47 ms]'.



Проблемы при работе с брокерами сообщений

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

The screenshot shows the Offset Explorer 2.3 interface. On the left, a tree view displays the cluster structure under 'localhost:29092', including 'Brokers', 'Topics', and 'Partitions'. The 'demo_topic' is expanded to show 'Partition 0'. On the right, the 'Data' tab is active, displaying a table of messages with columns for 'Offset', 'Key', and 'Value'. The messages contain JSON data: {"name":"string","chiefName":"string"}. Below the table, the 'Headers' tab is visible, showing a header with 'Key' and 'Value' pairs, including '_TypeId_' and 'com.example.demo.kafka.model.Subordinate'. The status bar at the bottom indicates 'Ready [Messages = 3] [114 Bytes] [47 ms]'.



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.

The screenshot shows the Offset Explorer 2.3 application. The left pane displays a tree view of Kafka clusters and topics. The right pane shows message details for a specific topic and partition, including a table of messages with Offset, Key, and Value columns.

Offset	Key	Value
0		{"name":"string","chiefName":"string"}
1		{"name":"string","chiefName":"string"}
2		{"name":"string","chiefName":"string"}

Below the table, there are tabs for Headers, Key, Value, Timestamp, and Headers. The Headers tab is active, showing a table with columns for Key and Value. The Value column contains the text "com.example.demo.kafka.model.Subordinate".

At the bottom of the application, a status bar indicates "Ready [Messages = 3] [114 Bytes] [47 ms]".



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.

The screenshot shows the Offset Explorer 2.3 application. The left pane displays a tree view of a Kafka cluster on localhost:29092, including Brokers, Topics, and Consumers. The right pane shows the 'Data' tab for a selected topic, displaying a table of messages with columns for Offset, Key, and Value. The Value column contains JSON objects like {"name":"string","chiefName":"string"}. Below the table, there are tabs for Headers, Timestamp, and Value, and a status bar at the bottom indicates 'Ready [Messages = 3] [114 Bytes] [47 ms]'.

Offset	Key	Value
0		{"name":"string","chiefName":"string"}
1		{"name":"string","chiefName":"string"}
2		{"name":"string","chiefName":"string"}



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.



AsyncAPI

Инфраструктура по работе со спецификацией в формате AsyncAPI

В процессе разработки

0. Инициировать первую версию спецификации в формате OpenAPI



SpringWolf

Генерация документации в формате AsyncAPI, подключение SwaggerUI



Почему нам не подходит AsyncAPI



AsyncAPI





Почему нам не подходит AsyncAPI



AsyncAPI



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



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.



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.



1. Игнорирует валидации.
2. Сложности при подключении.



Почему нам не подходит AsyncAPI



AsyncAPI

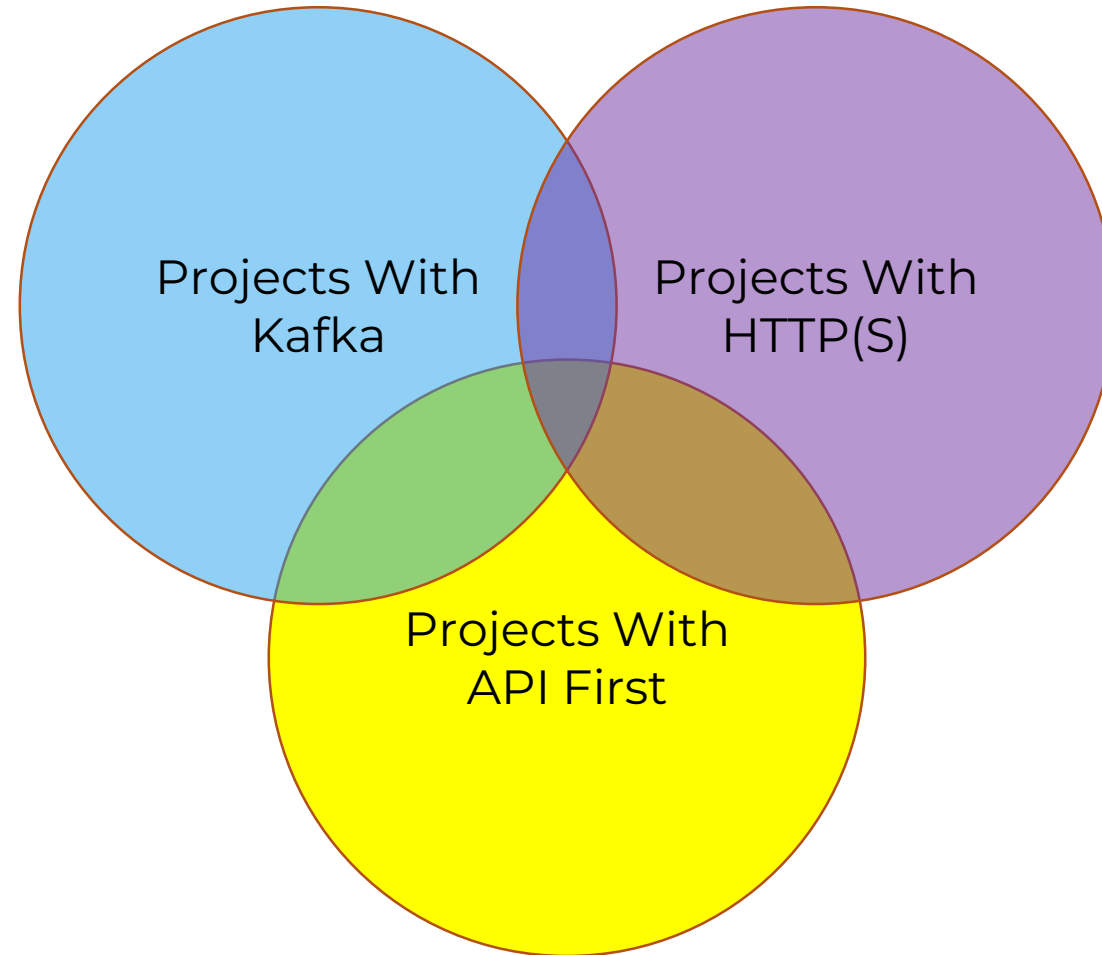
1. Инфраструктура вокруг AsyncAPI написана на JS.
2. Это еще один DSL.
3. Генератор создает приложение целиком.



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

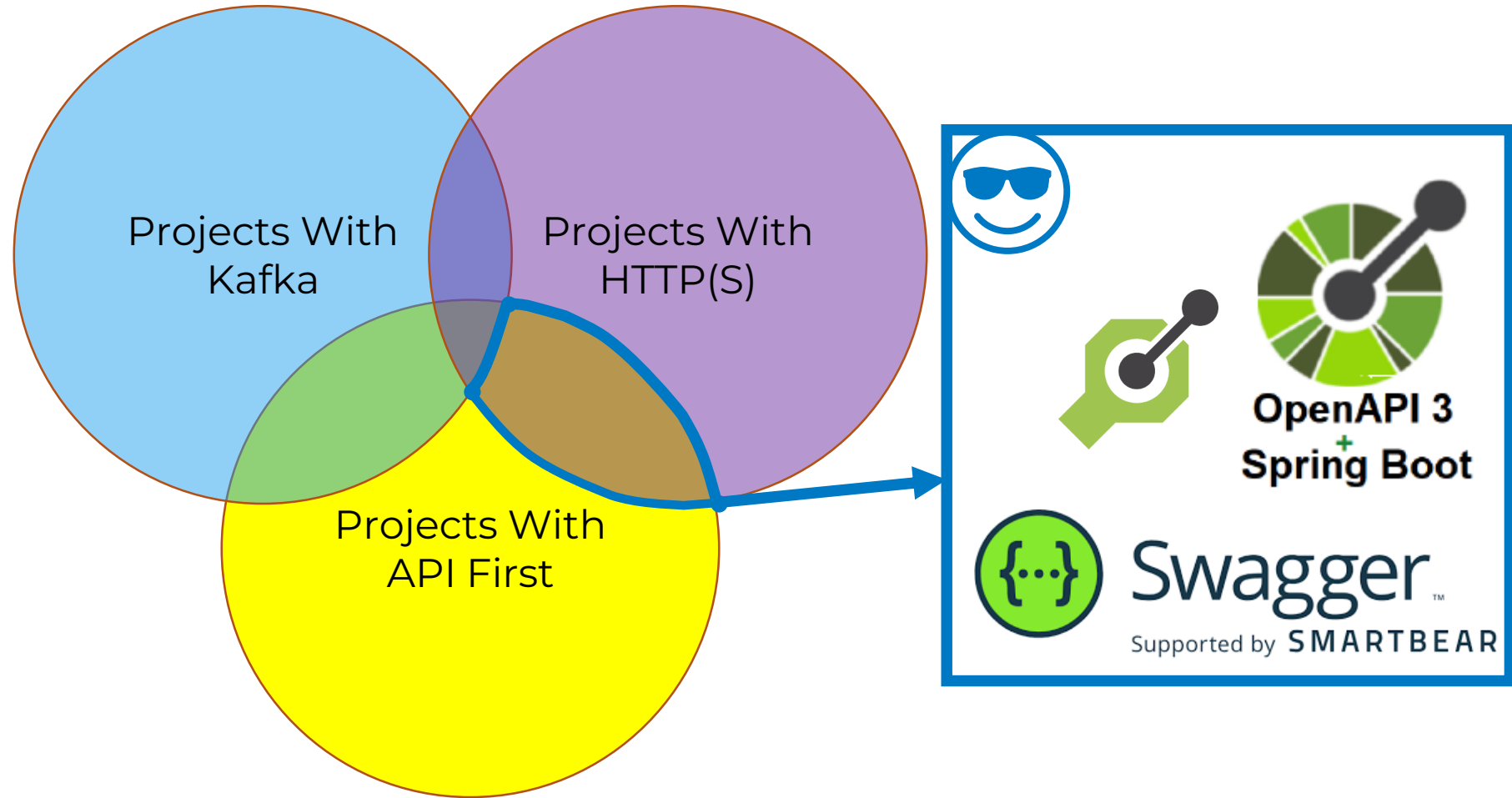


API First с Kafka и HTTP



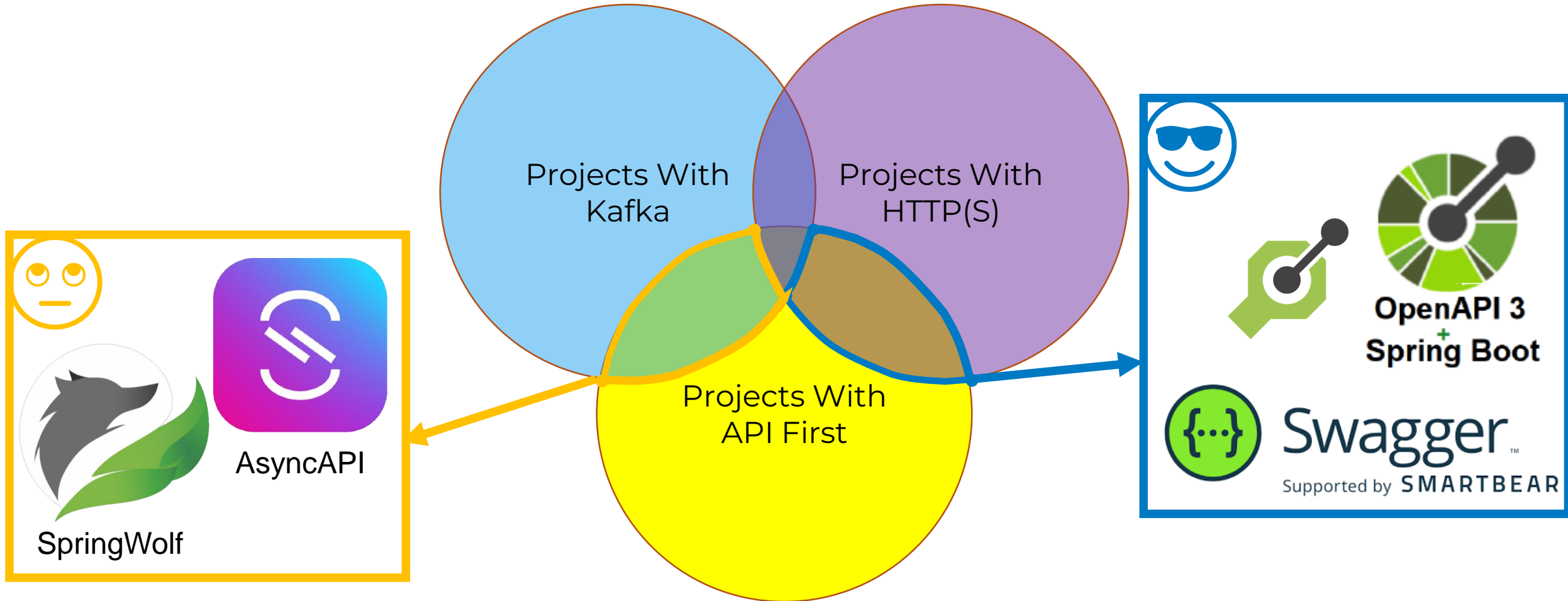


API First с Kafka и HTTP



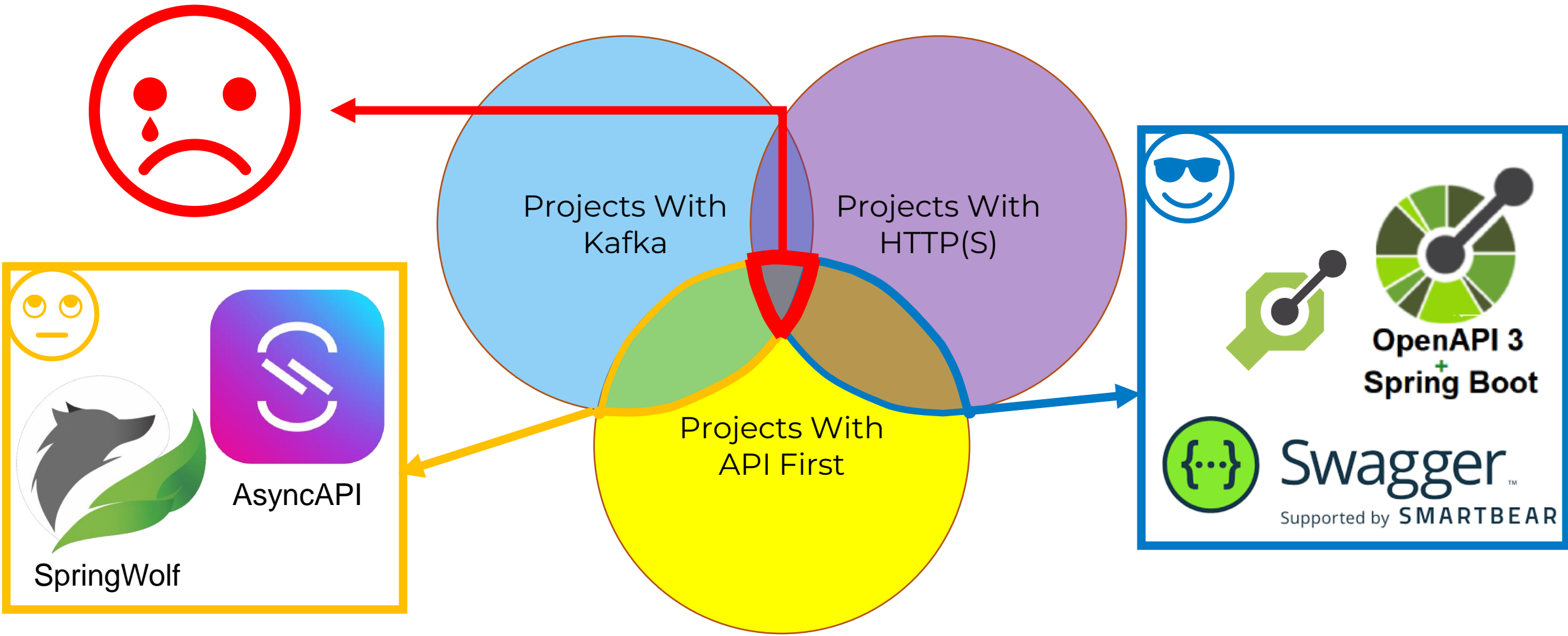


API First с Kafka и HTTP





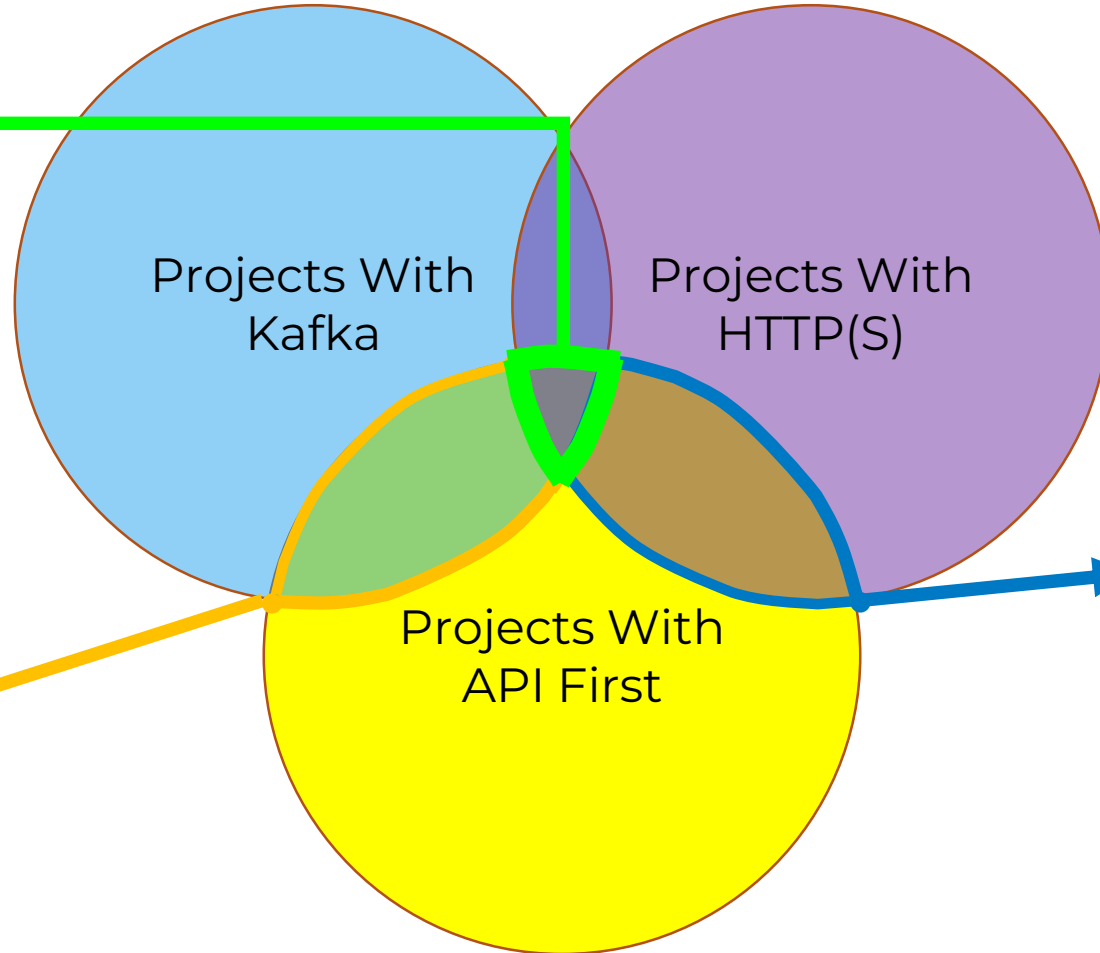
API First с Kafka и HTTP





API First с Kafka и HTTP

axen API



SpringWolf

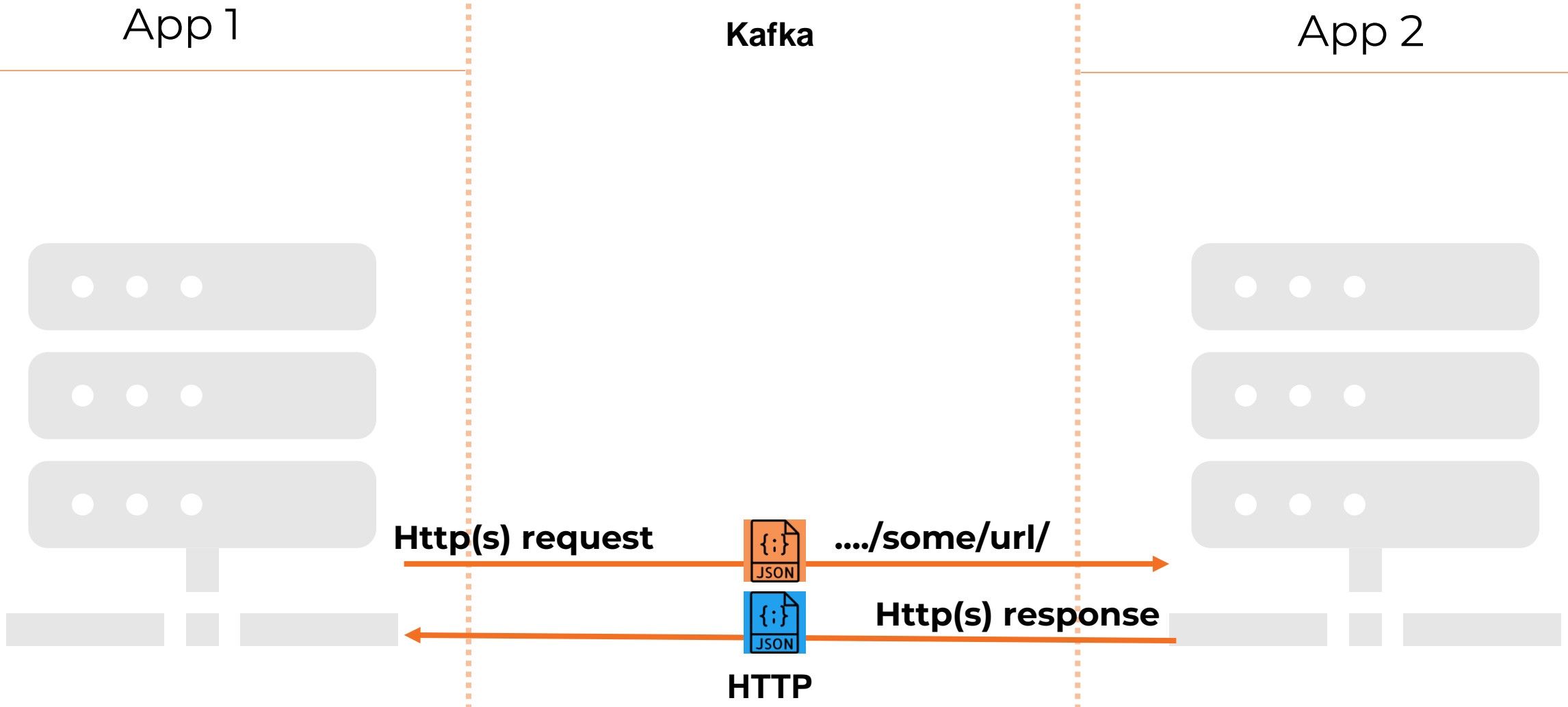
AsyncAPI

OpenAPI 3
Spring Boot

Swagger™
Supported by SMARTBEAR



Kafka VS HTTP с точки зрения клиента



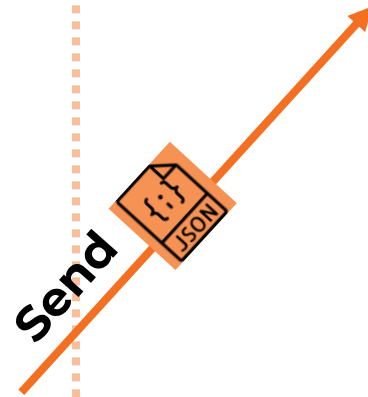
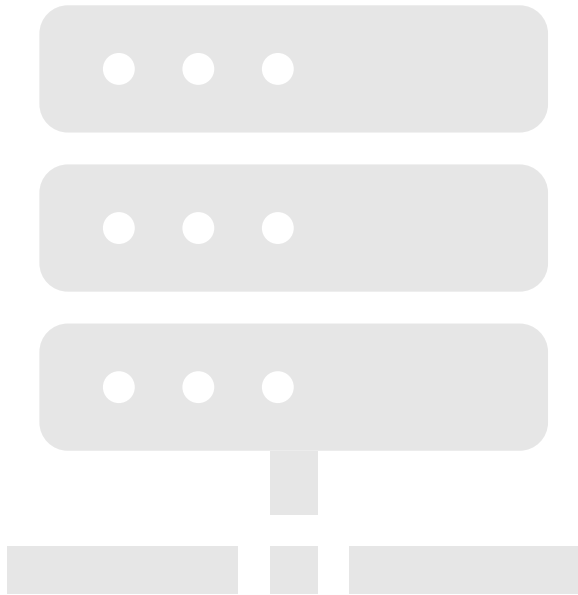


Kafka VS HTTP с точки зрения клиента

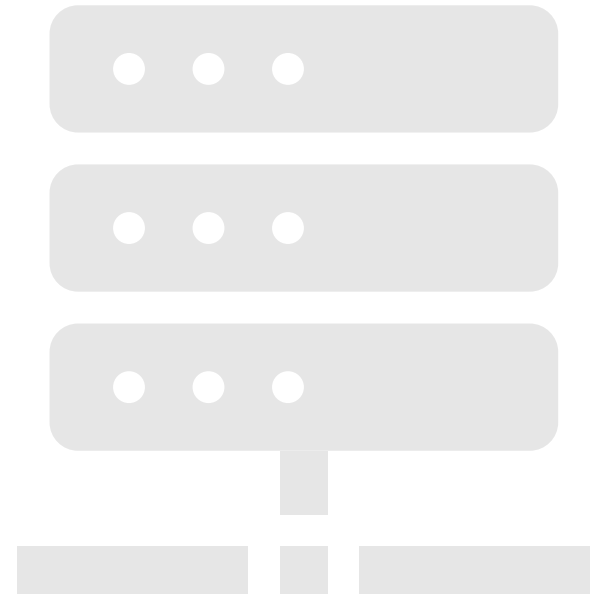
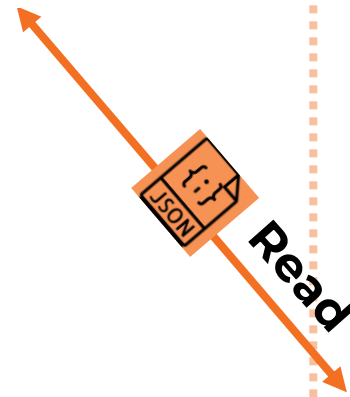
App 1

Kafka

App 2

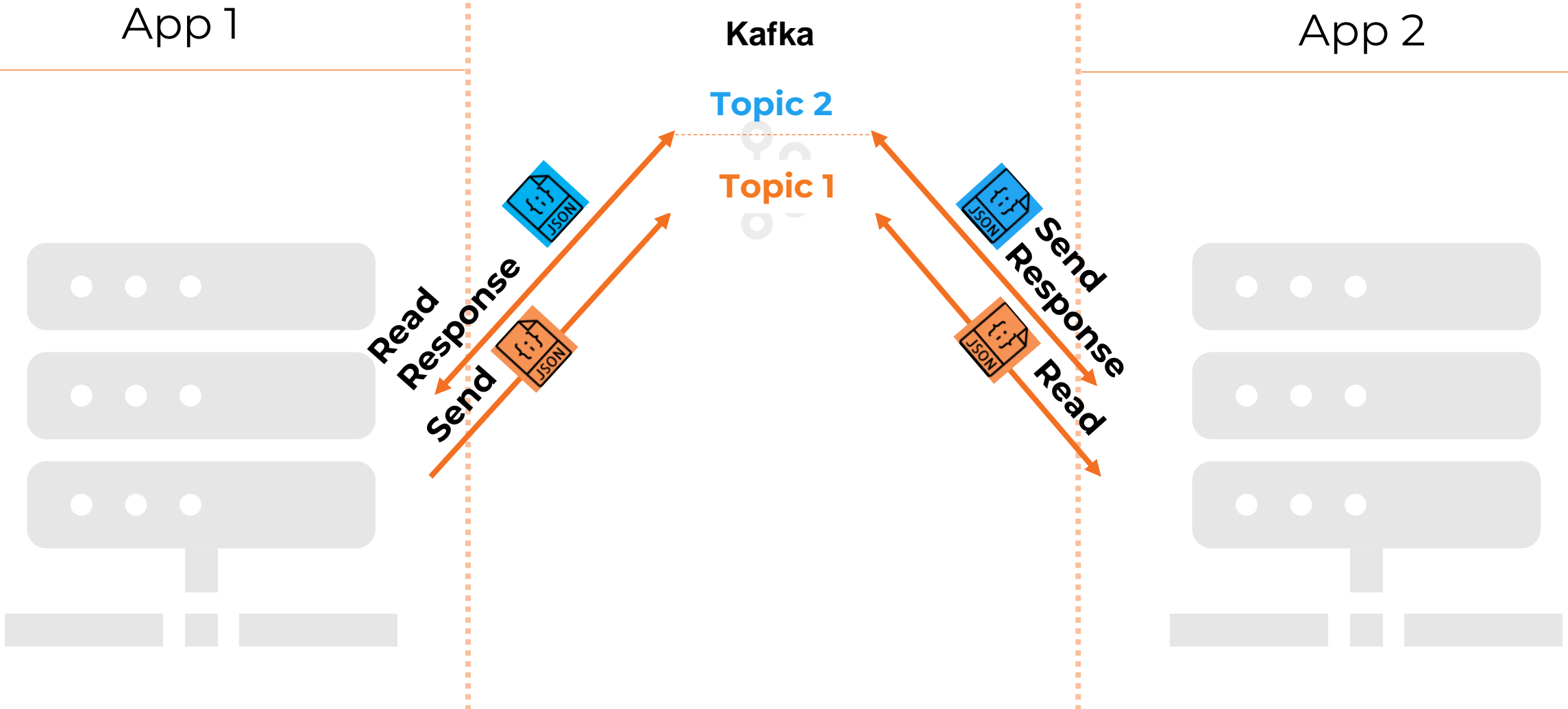


Topic 1



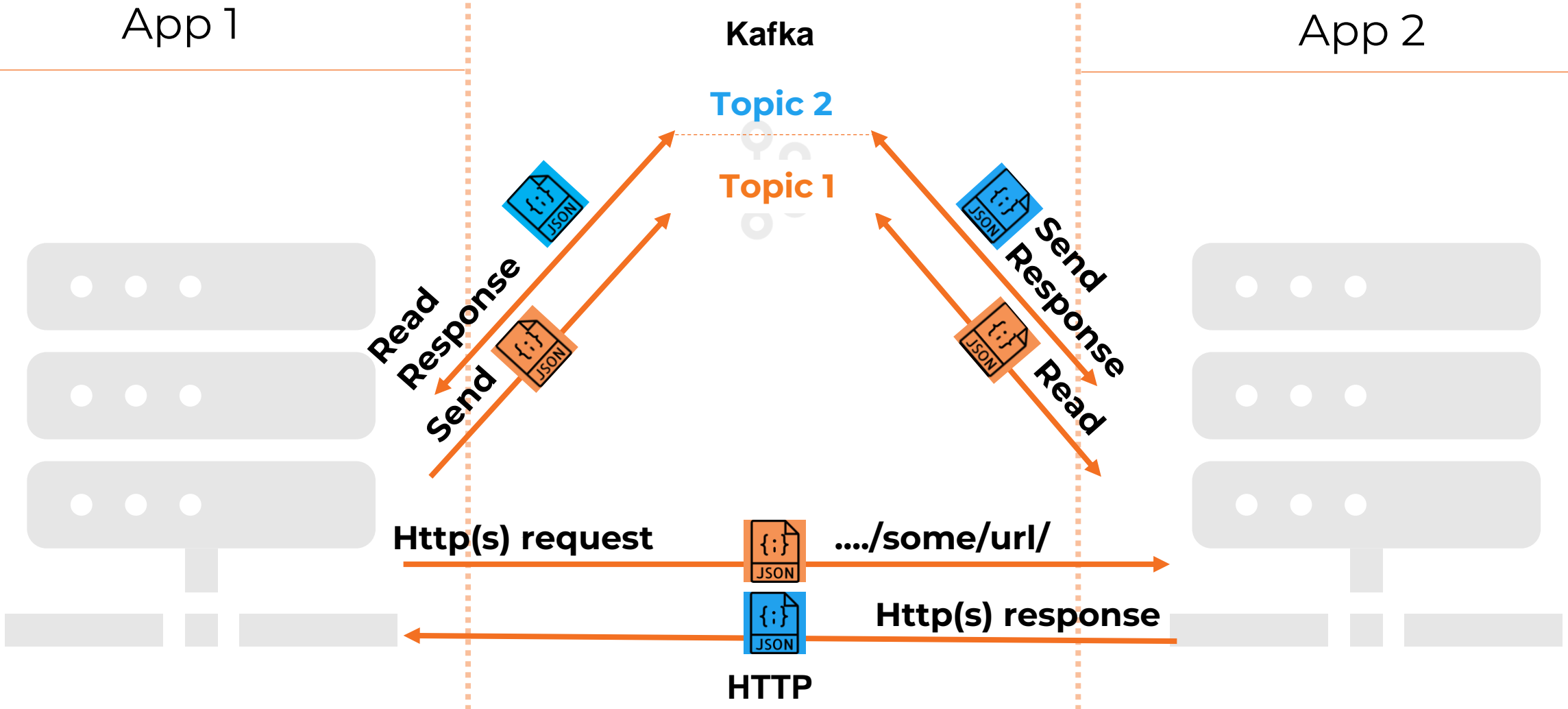


Kafka VS HTTP с точки зрения клиента



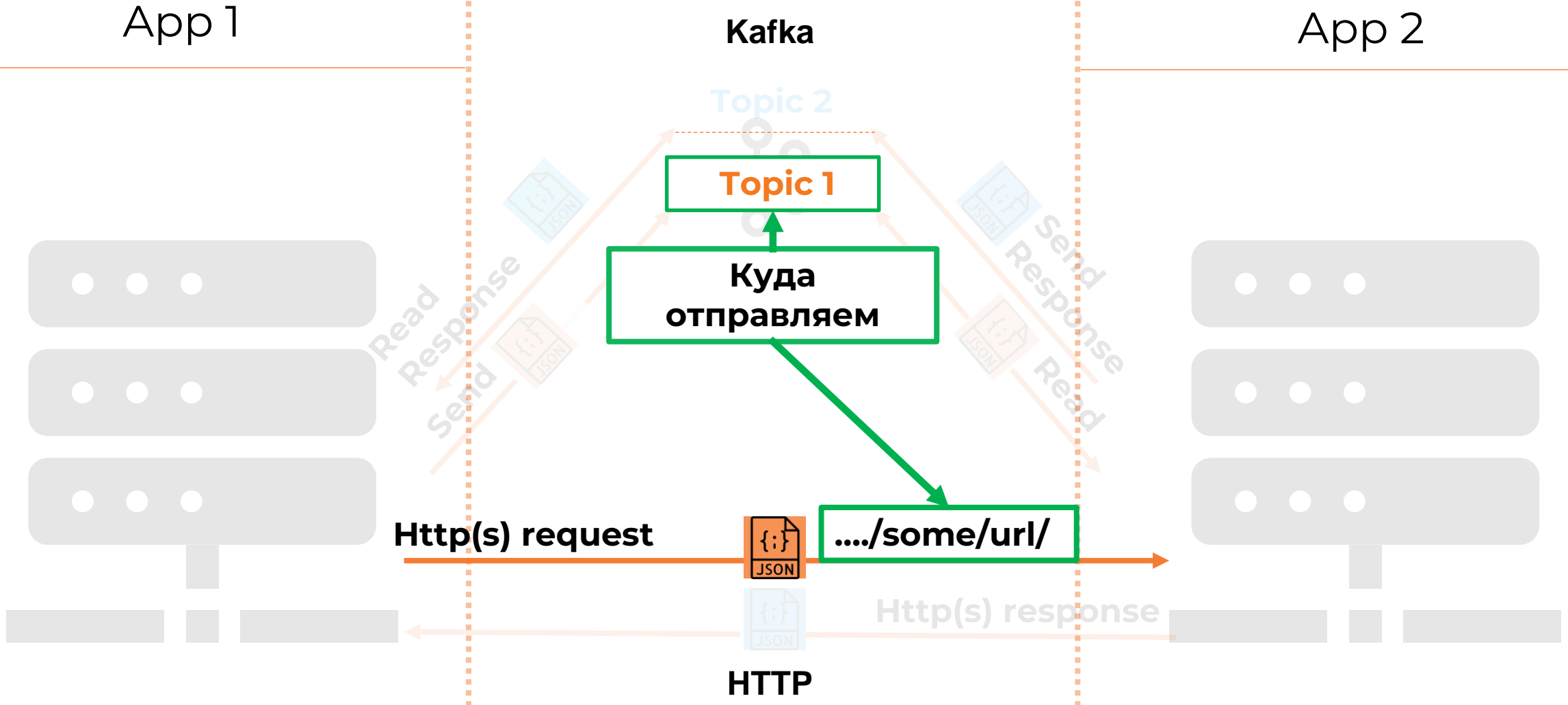


Kafka VS HTTP с точки зрения клиента





Kafka VS HTTP с точки зрения клиента



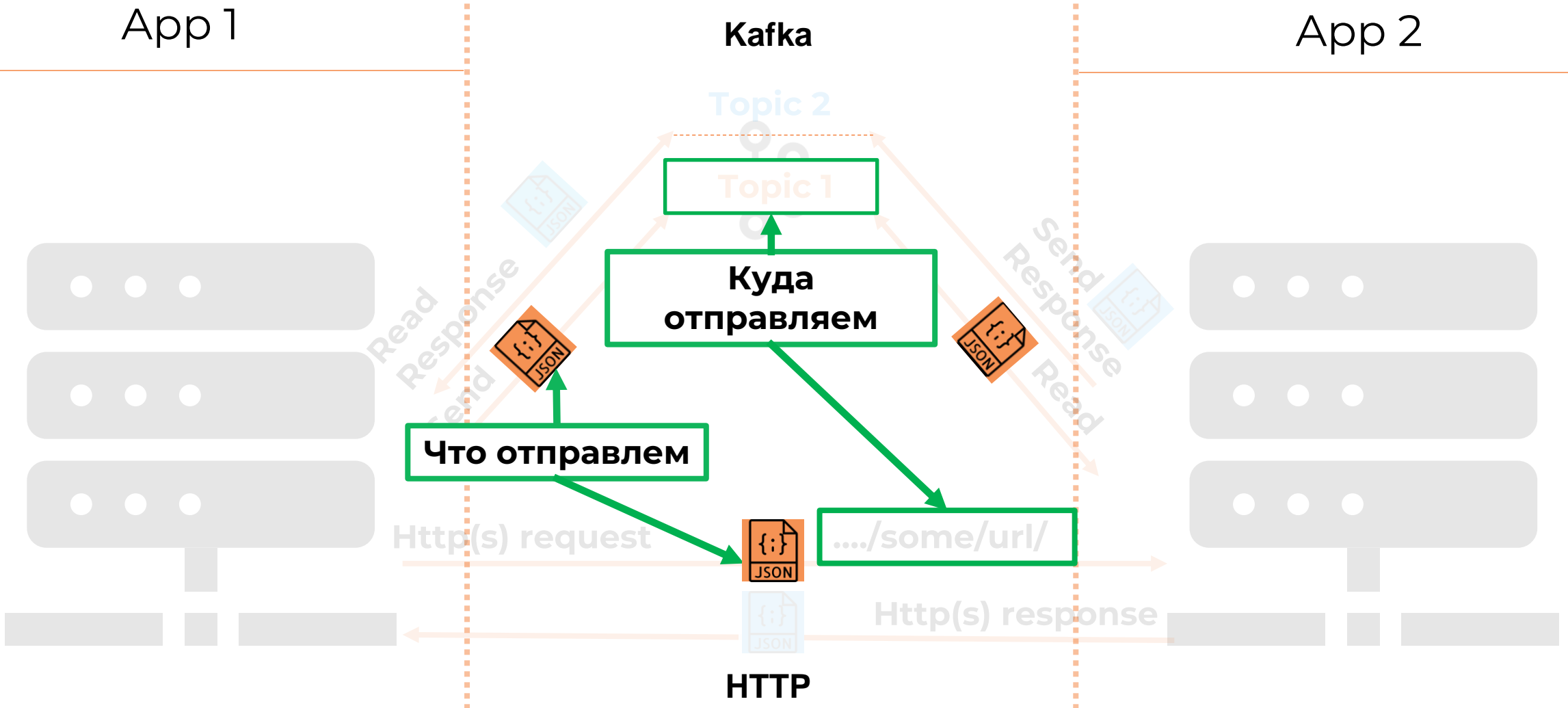


Kafka VS HTTP с точки зрения клиента

App 1

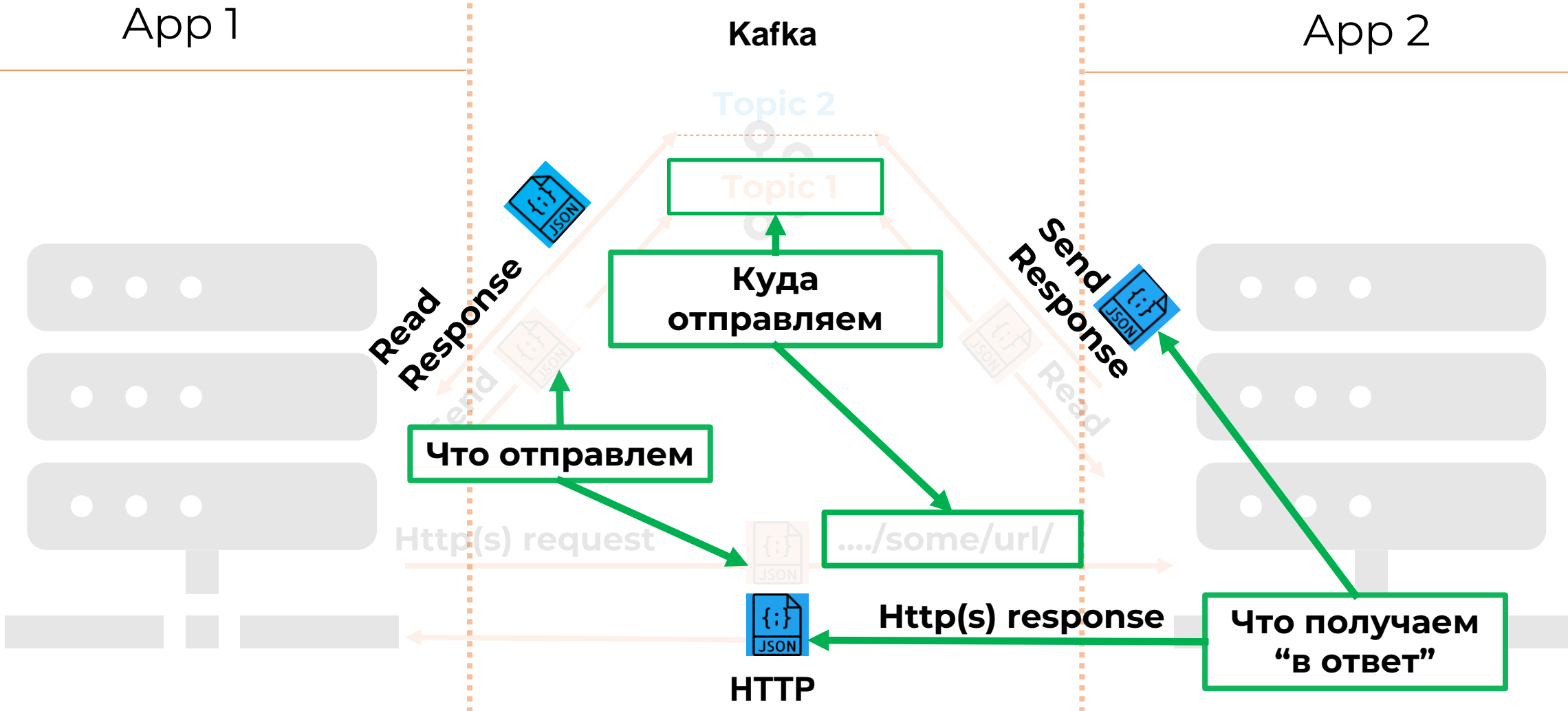
Kafka

App 2



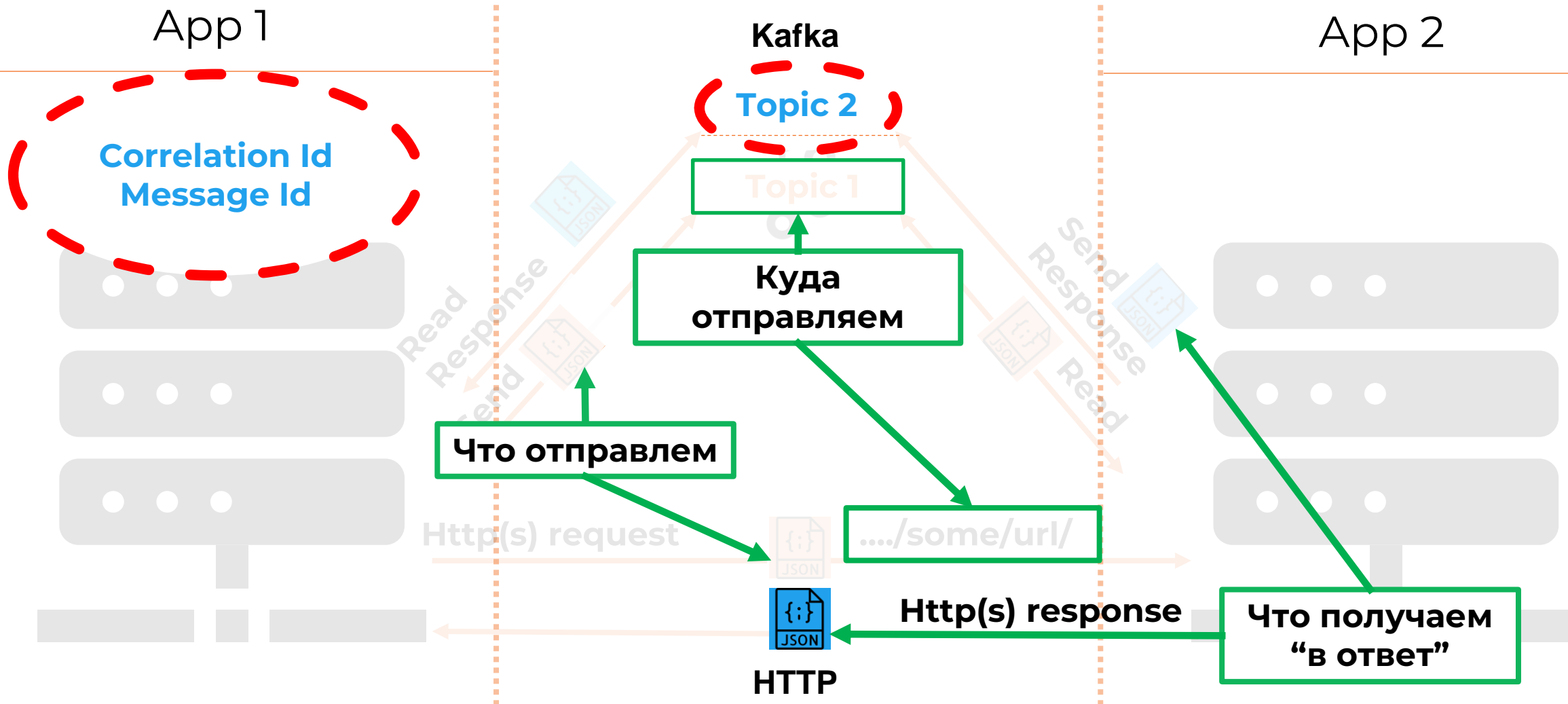


Kafka VS HTTP с точки зрения клиента





Kafka VS HTTP с точки зрения клиента

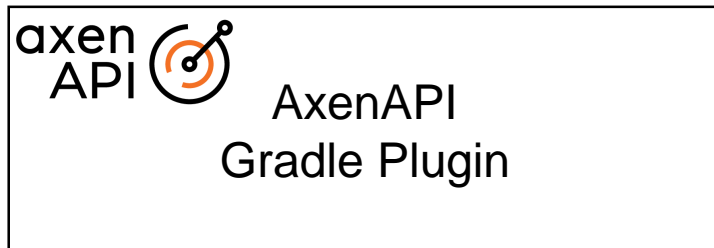




АхенAPI: адаптация OpenAPI под Kafka

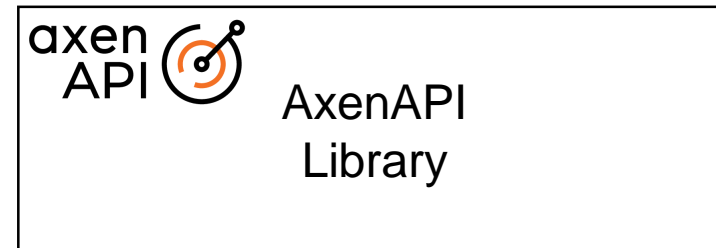
В начале разработки

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



В процессе разработки

0. Инициировать первую версию спецификации в формате OpenAPI



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



```
"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

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)

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



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)

```
"description": "Возвращает ответ Subordinate в топик, передаваемый через хедер replyTopic. Возвращаемое значение не перехватывается",  
"content": {  
  "*/*": {  
    "schema": {  
      "$ref": "#/components/schemas/Subordinate"  
    }  
  }  
}
```



Gradle-плагин AxenAPI

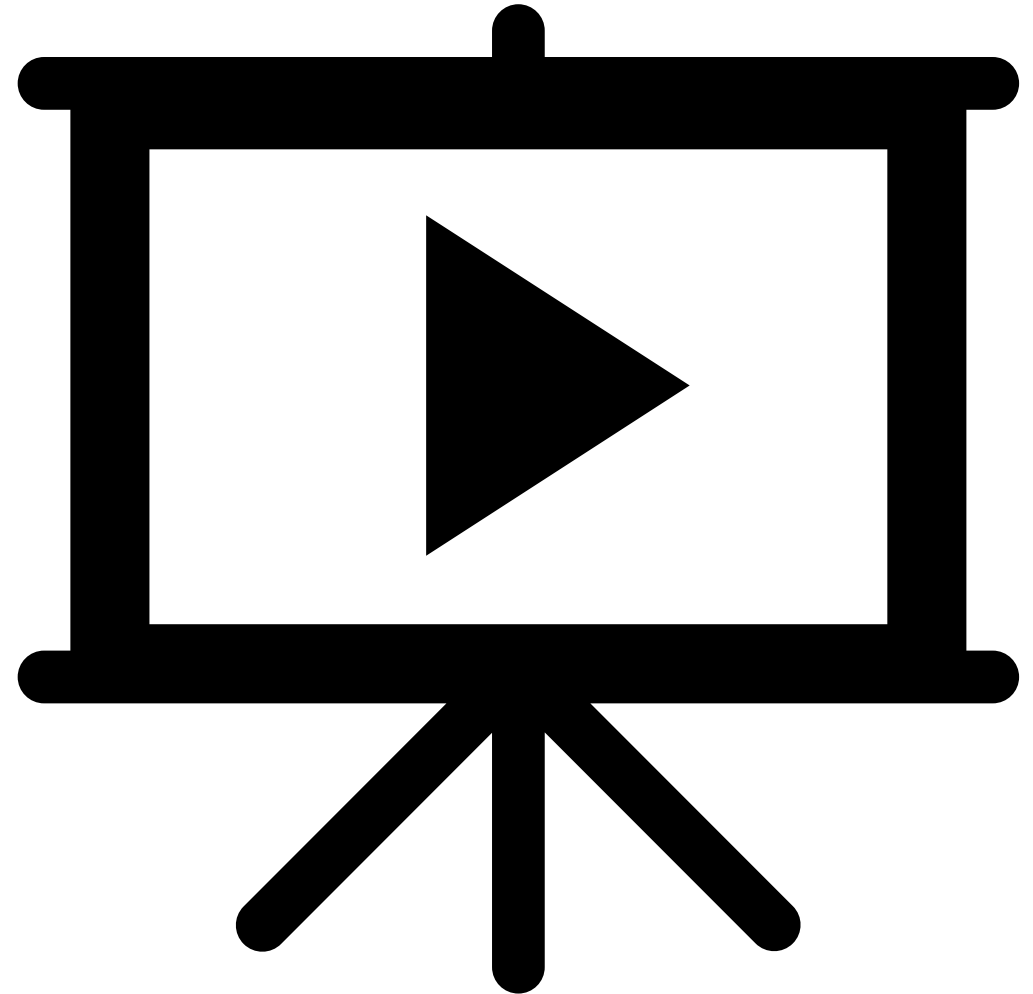
Gradle-плагин AxenAPI предназначен для генерации кода на Java с использованием Spring по спецификации API, реализованного с помощью брокеров сообщений.





Gradle-плагин AxenAPI. Демо

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



```
1  title: "App API",
5  "version": "snapshot"
6  },
7  "servers": [
8    {
9      "url": "http://example.example",
10     "description": "Generated server url"
11   }
12 ],
13 "paths": {
14   |
15
16 },
17 "components": {
18   "schemas": {
19     "ExampleIn": {"type": "object"...},
29     "ExampleOut": {"type": "object"...}
39   },
40   "securitySchemes": {
41     "Internal-Token": {
paths
```


Project

- axenapi_demo [demo] C:\ideaProjects\axenapi\axenapi_demo
 - .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

```
1 {
2   "openapi": "3.0.1",
3   "info": {
4     "title": "App API",
5     "version": "snapshot"
6   },
7   "servers": [
8     {
9     "url": "http://example.example",
10    "description": "Generated server url"
11  }
12 ],
13 "paths": {
14   "/kafka/example_group/example_topic/ExampleIn": {
15     "post": {
16       "description": "example handler",
17       "operationId": "ExampleHandler",
18       "tags": ["example"],
19       "security": [{
20         "ApiKeyAuth": []
21       ]
22     }
23   }
24 }
```

Gradle

- demo
 - Tasks
 - axenapigeneratorclientspringboot2
 - axenapigeneratorclientspringboot3
 - Tasks
 - application
 - build
 - documentation
 - help
 - other
 - verification
 - Dependencies
 - axenapigeneratorserverspringboot2
 - axenapigeneratorserverspringboot3
 - axenapilibspringboot2
 - axenapilibspringboot3

Run: axenapi_demo:axenapigeneratorserverspringboot2 [build] x

axenapi_demo:axenapigeneratorserverspringboot2 [build]: success: 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.cc
2023-10-10 15:36:37.832 INFO 12240 --- [ntainer#0-0-C-1] o.s.k.l.KafkaMessageListenerContainer : example_group: Cc

> Task :axenapigeneratorserverspringboot2:check
> Task :axenapigeneratorserverspringboot2:build

BUILD SUCCESSFUL in 24s
8 actionable tasks: 8 executed
15:36:38: Execution finished 'build'.
```



Gradle-плагин АхенAPI. Как мы это сделали?

```
2
3 import ...
18
19 public class KafkaCodegenGenerator extends SpringCodegen {
20     private static final String CLIENT_IMPL_TEMPLATE_NAME
21     private static final String KAFKA_SENDER_SERVICE_TEMPL
22     private static final String KAFKA_SENDER_SERVICE_TEMPL
23     private static final String KAFKA_SENDER_SERVICE_IMPL_
24     private static final String KAFKA_SENDER_SERVICE_IMPL_
25     private static final String KAFKA_SENDER_SERVICE_CONFI
26     private static final String KAFKA_SENDER_SERVICE_CONFI
27     private static final String KAFKA_SENDER_SPRING_FACTOR
```

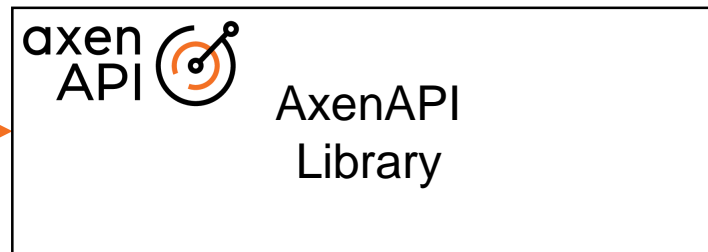


Библиотека AxenAPI

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



Application
code



```
.:vers:  
- 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 AcheAPI. 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);
    }
}
```

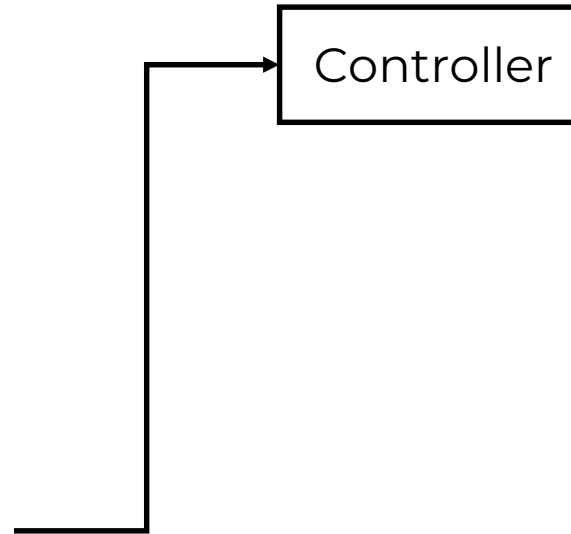


Библиотека AcheAPI. Принцип работы

```
@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);
    }
}
```



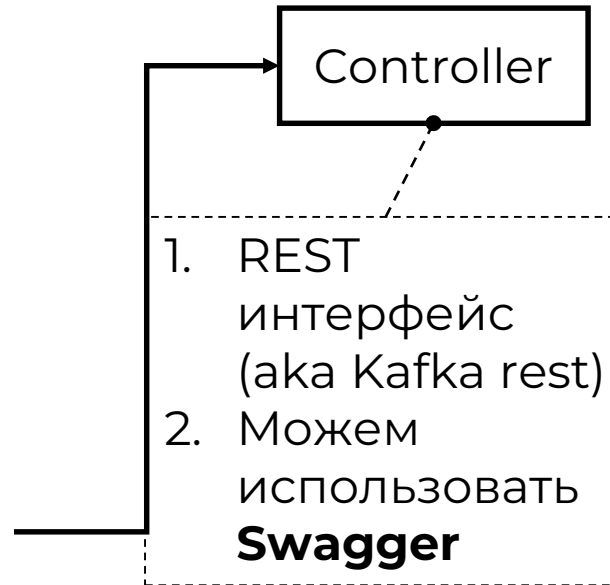


Библиотека AcheAPI. Принцип работы

```
@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);
    }
}
```



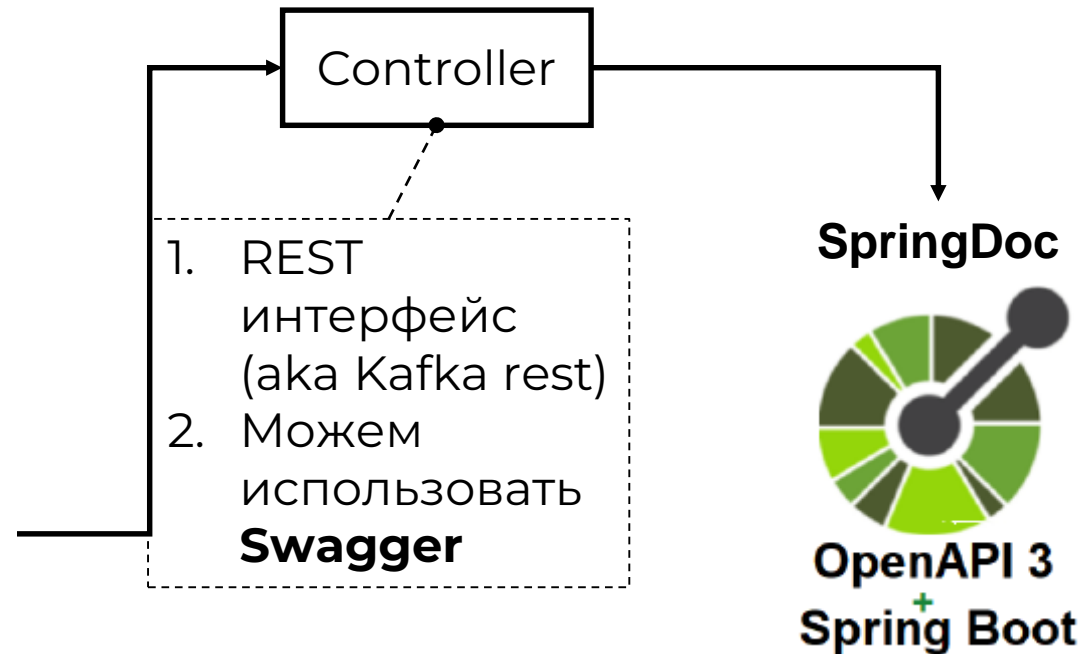


Библиотека AcheAPI. Принцип работы

```
@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);
    }
}
```



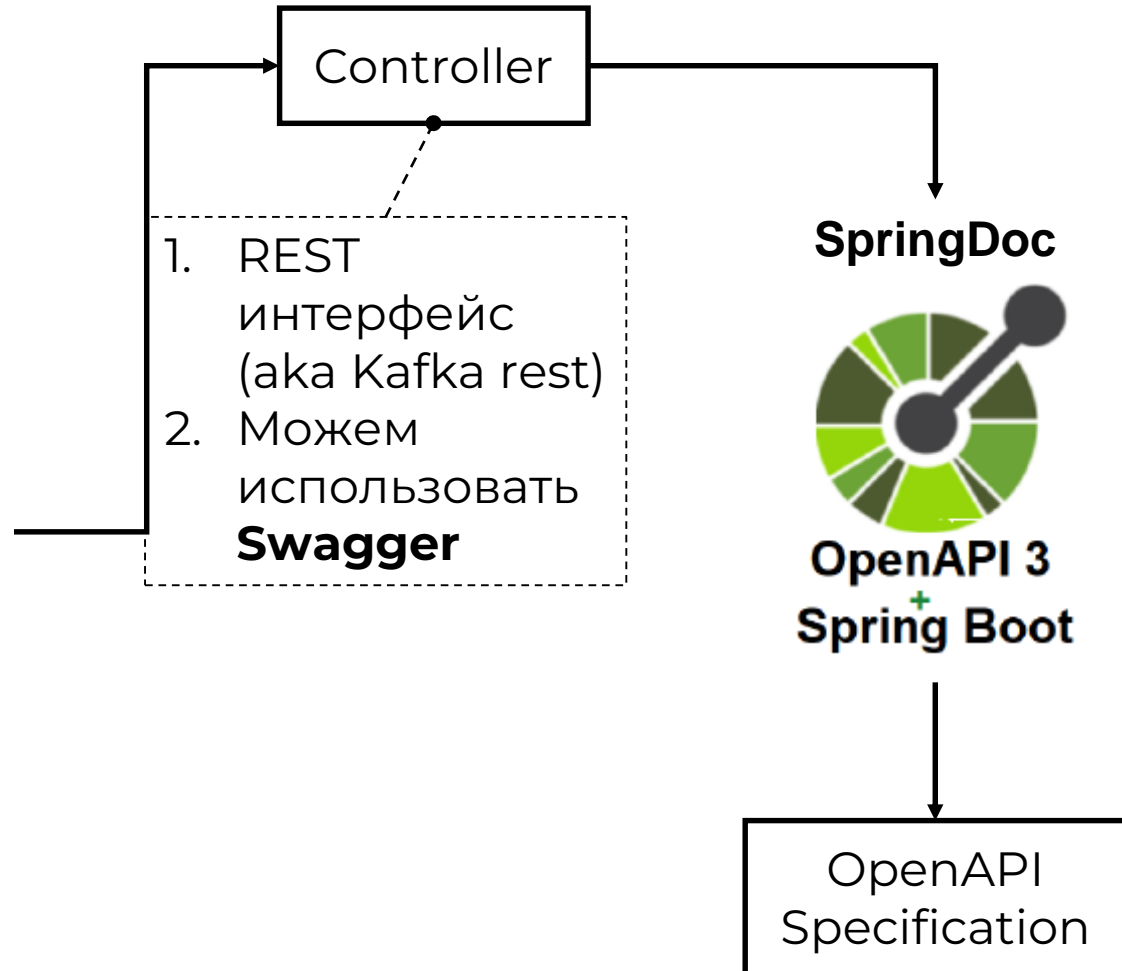


Библиотека AcheAPI. Принцип работы

```
@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 AхенAPI. Подключение

1. Добавить зависимость

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



Библиотека AхенAPI. 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 AхенAPI. Подключение

1. Добавить зависимость

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

3. Собрать проект

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. Результат

```
44 @PostMapping(  
45     path = "/ExampleMessage"  
46 )  
47 @ApiOperation(  
48     description = "Принимает ExampleMessage. Находится в DemoListener. Ничего не возвращает.",  
49     tags = {"demo"},  
50     parameters = {  
51         @Parameter(name = "header_1", in = io.swagger.v3.oas.annotations.enums.ParameterIn.QUERY),  
52         @Parameter(name = "header_2", in = io.swagger.v3.oas.annotations.enums.ParameterIn.QUERY)  
53     }  
54 )  
55 @ApiResponse(  
56     statusCode = "200",  
57     description = "No return value"  
58 )  
59 @SecurityRequirement(  
60     name = "Internal-Token"  
61 )  
62 @  
63 public void executeExampleMessage(@RequestBody ExampleMessage message,  
64     @RequestParam @Parameter(hidden = true) Map<String, String> params,  
65     HttpServletResponse servletResponse, @RequestHeader Map<String, String> headers) {  
66     String authToken = headers.get("service_access_token");  
67     if(authToken != null) params.put("SERVICE_ACCESS_TOKEN", authToken);  
68     kafkaSenderService.send(topicName, message, params, servletResponse);  
69 }
```



Библиотека АхенAPI. Подключение. Результат

АхенAPI 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 АхенAPI. Подключение. Результат

АхенAPI 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 АхенAPI. Подключение. Результат

АхенAPI 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 АхенAPI. Подключение. Результат

АхенAPI 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 АхенAPI. Подключение. Результат

АхенAPI 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 АхенAPI. Подключение. Результат

Внимательно рассмотрим
результат подключения
Swagger UI.

The screenshot displays the Swagger UI interface for an OpenAPI definition. At the top, the Swagger logo is visible, along with the text 'Supported by SMARTBEAR'. A dropdown menu labeled 'Select a definition' is open, showing three options: 'kafka', 'kafka', and 'rest', with 'kafka' selected. Below this, the title 'OpenAPI definition' is shown with version indicators 'v0' and 'OAS3'. The URL '/v3/api-docs/kafka' is displayed. A 'Servers' section contains a dropdown menu with the value 'http://localhost:8085 - Generated server url'. The main content area is titled 'demo' and lists four API endpoints, all with a 'POST' method:

- 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

Below the endpoints, a 'Schemas' section is visible, showing the definition for the 'Subordinate' schema:

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

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* > [...]  
}
```

OpenAPI definition v0 OAS3

/v3/api-docs/kafka

Select a definition

- kafka
- kafka
- rest

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* > [...]
}
```

OpenAPI definition v0 OAS3

/v3/api-docs/kafka

Select a definition

- kafka
- kafka
- rest

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* > [...]
}
```

OpenAPI definition v0 OAS3

/v3/api-docs/kafka

Select a definition

- kafka
- kafka
- rest

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

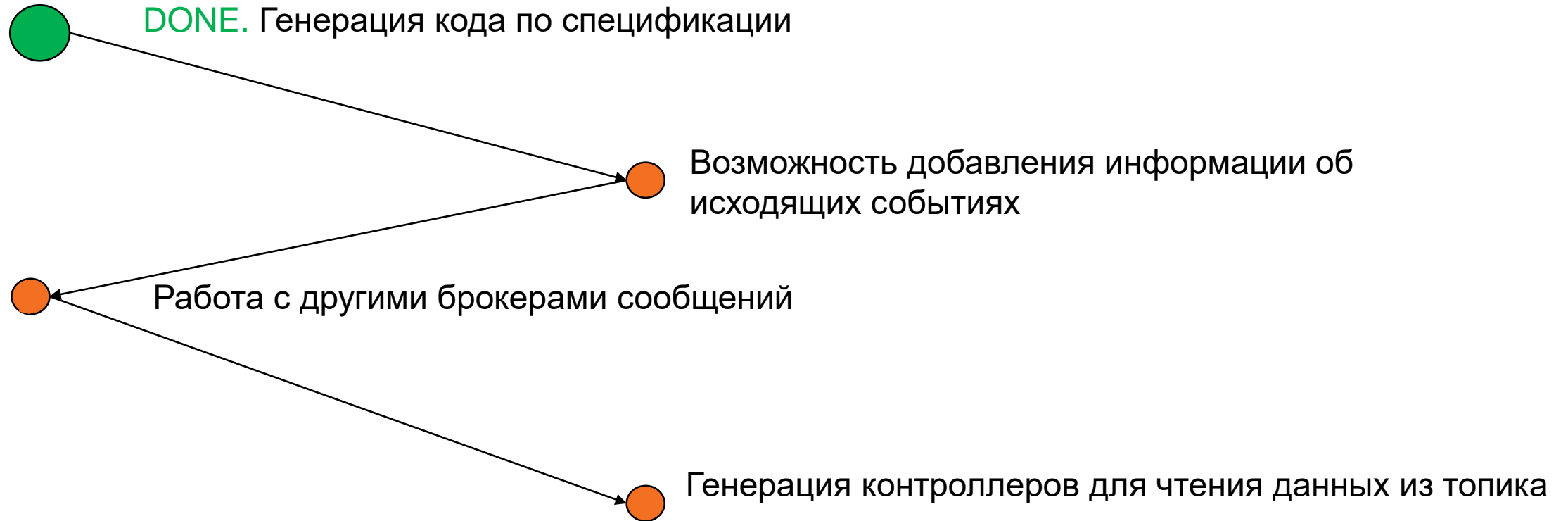
Schemas

Модели получаемых событий

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



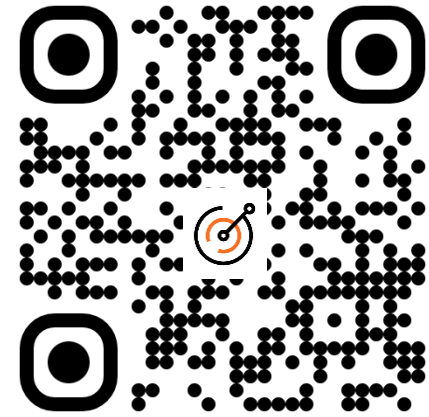
АхепAPI. Планы развития



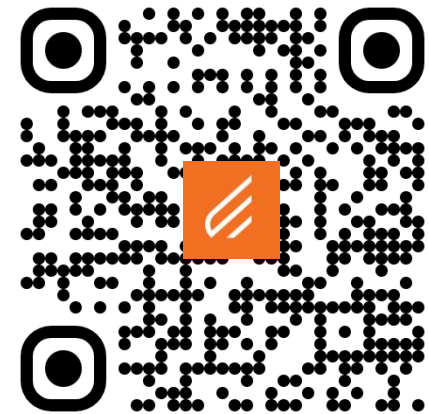


Выводы

- 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](#)



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

