Разработка серверного ПО. Лекция 2. Описание и проектирование API

Спецификация OpenAPI (OAS)

Спецификация OpenAPI (OAS) – это формат описания REST API, не зависящий от языка программирования. Этот формат продвигается OpenAPI Initiative (OAI).

XML

```xml
<Servers>
  <Server>
    <name>Server1</name>
    <owner>Prajwal</owner>
    <status>active</status>
  </Server>
  <Server>
    <name>Server2</name>
    <owner>John</owner>
    <status>inactive</status>
  </Server>
</Servers>
```

JSON

```json
{
  "Servers": [
    {
      "name": "Server1",
      "owner": "Prajwal",
      "status": "active"
    },
    {
      "name": "Server2",
      "owner": "John",
      "status": "inactive"
    }
  ]
}
```

YAML

```yaml
Servers:
  - name: Server1
    owner: Prajwal
    status: active
  - name: Server2
    owner: John
    status: inactive
```

YAML в сравнении с JSON Обратите внимание на следующие моменты:

- имена и значения свойств в YAML не заключены в двойные кавычки ("");
- структурные фигурные скобки ({}) и запятые (,) в JSON заменены символами перевода строки и отступами в YAML;
- скобки массива ([]) и запятые (,) заменены черточками (-) и символами перевода строки;
- в отличие от JSON, YAML допускает комментарии, начинающиеся с символа #.

Преобразовать один из этих форматов в другой можно относительно легко. Однако имейте в виду, что при преобразовании документа из формата YAML в JSON комментарии будут утеряны.

Базовый документ OAS предоставляет общую информацию об API, такую как его имя и версия. Он описывает доступные ресурсы (идентифицированные по путям) и операции каждого ресурса (или действия, которые вы видели в предыдущей главе), идентифицированные HTTP-методами, включая их параметры и ответы.

Существуют и другие форматы описания REST API; наиболее заметными конкурентами OAS являются RAML и Blueprint.

Документ OAS – это простой текстовый файл, который легко можно сохранить в системе управления версиями, такой как Git, как и код. Таким образом, управлять версиями и отслеживать изменения будет просто.

Документ OAS имеет структуру, которая помогает более эффективно описывать программный интерфейс. Вы должны описать ресурсы, операции, параметры и ответы. Вы можете определить повторно используемые компоненты (например, модель данных), избегая болезненного и рискованного искусства копирования и вставки фрагментов описаний API.

Говоря о написании документа OAS, вы можете использовать свой любимый текстовый редактор, но я рекомендуется использовать редактор, который специально предназначен для обработки этого формата. Речь идет об онлайн-редакторе Swagger (https://editor-next.swagger.io/).

Поскольку OAS представляет собой машиночитаемый формат, этот редактор предлагает такие функции, как автозаполнение и проверка документов, а правая панель дает полезную визуализацию отредактирован ного документа. Чтобы увидеть удобное для восприятия представление структур данных, используемых в качестве параметров или ответов, особенно полезны представления Model и Example Value.

Документ OAS можно использовать для создания справочной документации по API, в которой показаны все доступные ресурсы и операции. Для этого можно использовать пользовательский интерфейс Swagger (https://github.com/swagger-api/swagger-ui), который показывает документ OAS как в правой панели редактора Swagger Editor.

Есть и другие инструменты. Например, в качестве альтернативы Swagger UI можно использовать такой инструмент, как ReDoc (https://github.com/Rebilly/ReDoc), также с открытым исходным кодом.

Как только API будет описан с помощью формата описания API, из него можно частично сгенерировать код реализации. Вы получите пустой костяк исходного кода, который также можно использовать для создания рабочего макета. Потребители также могут воспользоваться преимуществами таких машиночитаемых описаний API для генерации кода для использования API. И такой формат также может использоваться инструментами тестирования API или безопасности, а также многими другими инструментами, связанными с API. Например, большинство решений для API-шлюзов (прокси, созданные для предоставления доступа и защите API) можно настроить с помощью файла описания API, такого как документ OAS. (Список инструментов на базе OAS https://openapi.tools/)

Создание документа OAS

Минимальный, но допустимый документ OAS:

```
openapi: «3.0.0» ①
info: ②
   title: Shopping API
   version: «1.0»
paths: {} ③
```

① **Версия OAS.**
② **Общая информация API.**
③ **Пустые пути**

Описание ресурса

```
openapi: «3.0.0»
info:
   title: Shopping API
   version: «1.0»
paths: ①
   /products: ②
      description: The products catalog ③
```

① **Ресурсы API.**
② **Путь к ресурсу.**
③ **Описание ресурса.**

Свойство description не является обязательным, но будет полезно всегда и из него может быть сгенерирована документация.

Описание действия над ресурсом

```
openapi: «3.0.0»
info:
   title: Shopping API
   version: «1.0»
paths:
   /products: ①
      description: The products catalog
      get: ②
         summary: Search for products ③
         description: |  ④
         Search for products in catalog
         using a free query parameter
```

① **Ресурс.**
② **HTTP-метод действия.**
③ **Краткое описание действия.**
④ **Длинное описание действия.**

Свойство summary – это краткое описание действия без подробностей. Есть также свойство description, которое можно взять для предоставления более подробного описания действия.

Описание ответов на действия

```
openapi: «3.0.0»
info:
   title: Shopping API
   version: «1.0»
paths:
   /products:


   description: The products catalog
   get:
      summary: Search for products
      description: | Search for products in catalog
         using a free query parameter
      responses: ①
         "200": ②
            description: | ③
            Products matching free query parameter
```

① **Список ответов.**
② **Код ответа 200 OK.**
③ **Описание ответ.**

Описание еще одного действия

```
openapi: "3.0.0"
info:
   title: Shopping API
   version: «1.0»
paths:
   /products: ①
      description: The products catalog
      get:
         summary: Search for products
         description: |
            Search for products in catalog
            using a free query parameter
         responses:
            "200":
               description: |
               Products matching free query parameter
post: ②
   summary: Add product ③
   description: | ④
         Add product (described in product info
         ➥ parameter) to catalog
responses: ⑤
   "200": ⑥
      description: | ⑦
         Product added to catalog
```

① **Ресурс.**
② **HTTP-метод действия.**
③ **Краткое описание действия.**
④ **Длинное описание действия.**
⑤ **Список ответов.**
⑥ **Ответ 200 OK.**
⑦ **Описание ответа 200 OK.**

Описание параметров

```
openapi: "3.0.0"
info:
   title: Shopping API
   version: "1.0"
paths:
   /products: ①
      get: ②
         summary: Search for products
            description: | Search for products in catalog
               ➥ using a free query parameter
               parameters: ③
               [...]
         responses:
            "200":
               description: | Products matching free query
               ➥ parameter
```

① **Ресурс.**
② **Действие.**
③ **Список параметров действия (за исключением тела).**

```
parameters:
   - name: free-query ①
      description: |  ②
         A product's name, reference, or partial
         ➥ description
         in: query ③
         required: false ④
         schema: ⑤
            type: string ⑥
```

① **Имя параметра.**
② **Описание параметра.**
③ **Расположение параметра.**
④ **Является ли параметр обязательным.**
⑤ **Описание структуры данных параметра.**
⑥ **Тип параметра (строка).**

Параметр находится в (in) запросе (query), но не является обязательным, и что его структура данных описана в схеме (schema). Данная схема просто указывает на то, что тип этого параметра – строка.

Описание базового товара с помощью JSON Schema

```
type: object ①
properties: ②
   reference: ③
      type: string ④
   name: ③
      type: string ④
   price: ③
      type: number ④
```

① **Эта схема описывает объект.**
② **Здесь содержатся свойства.**
③ **Имя свойства.**
④ **Тип свойства.**

Обязательные и необязательные свойства

```
type: object
required: ①
   - reference
   - name
   - price
properties:
   reference: ②
      type: string
   name: ②
      type: string
   price: ②
      type: number
   description: ③
      type: string
```

① **Список обязательных свойств.**
② **Обязательные свойства.**
③ **Необязательное свойство.**

Документирование схемы JSON

```yaml
type: object
description: A product ①
required:
   - reference
   - name
   - price
properties:
   reference:
      type: string
      description: Product's unique identifier ②
      example: ISBN-9781617295102 ③
   name:
      type: string
      example: The Design of Web APIs ③
   price:
      type: number
      example: 44.99 ③
   description:
      type: string
      example: A book about API design ③
```

① **Описание объекта.**
② **Описание свойства.**
③ **Пример значения свойства.**

Описание комплексного свойства с помощью JSON Schema

```
type: object
description: A product
required:
   - reference
   - name
   - price
   - supplier ①
properties:
   reference:
      type: string
      description: Product's unique identifier
      example: ISBN-9781617295102
name:
   type: string
   example: The Design of Web APIs
price:
   type: number
   example: 44.99
description:
   type: string
   example: A book about API design
supplier: ②
   type: object


  description: Product's supplier
  required: ③
     - reference
     - name
  properties: ④
     reference:
        type: string
        description: Supplier's unique identifier
        example: MANPUB
     name:
        type: string
        example: Manning Publications
```

① **Необходимо указать поставщика.**
② **Свойство объекта supplier.**
③ **Необходимые свойства.**
④ **Описание свойств.**

Описание данных ответа

```
openapi: "3.0.0"
info:
   title: Shopping API
   version: "1.0"
paths:
   /products:
      get:
      summary: Search for products
      description: | Search using a free query (query parameter)
      parameters:
         [...]
      responses:
         "200":
            description: Products matching free query
            content: ①
               application/json: ②
                  schema: ③
                     [...]
```

① **Определение тела ответа.**
② **Медиа тип тела ответа.**
③ **JSON-схема тела ответа.**

```
responses:
   "200":
      description: Products matching free query
      content:
         application/json: ①
            schema: ②
               type: array ③
               description: Array of products
               items: ④
                  type: object
                  description: A product
                  required:
                     - reference
                     - name
                     - price
                     - supplier
               properties:
                  reference:
                     description: Unique ID identifying a product
                     type: string
                  name:
                     type: string
                  price:
                     description: Price in USD
                     type: number
                  description:
                     type: string
                  supplier:
                     type: object
                     description: Product's supplier
                     required:
                        - reference
                        - name
                     properties:
                        reference:
                           type: string

                        name:
                           type: string
```

① Медиа тип тела ответа.
② JSON-схема тела ответа.
③ Тип ответа – массив.
④ Схема элементов массива.

Описание параметра тела действия

```
openapi: "3.0.0"
   info:
   title: Shopping API
version: "1.0"
paths:
   /products:
   description: The products catalog
   [...]
   post:
      summary: Add product
      description: Add product to catalog
      requestBody: ①
         description: Product's information ②
         application/json: ③
            schema: ④
               [...]
      responses:
         "200":
            description: Product added to catalog
```

① **Определение параметра тела.**
② **Описание параметра тела.**
③ **Медиа тип параметра тела.**
④ **Схема параметра тела.**

Полное описание параметра тела

```yaml
requestBody:
   description: Product's information
   content:
      application/json:
         schema: ①
            required:
               - name
               - price
               - supplierReference
            properties:
               name:
                  type: string
               price:
                  type: number
               description:
                  type: string
               supplierReference:
                  type: string
```

① **Схема параметра тела.**

Объявление схемы многократного использования

```yaml
openapi: "3.0.0"
[...]
components: ①
   schemas: ②
      product: ③
         type: object ④
         description: A product
         required:
            - reference
            - name
            - price
            - supplier
         properties:
            reference:
               description: |
                  Unique ID identifying
                  a product
               type: string
            name:
               type: string
            price:
               description: Price in USD
               type: number
            description:
               type: string
         supplier:
               type: object
               description: Product's supplier
               required:
                  - reference
                  - name
               properties:
                  reference:
                     type: string
                  name:
                     type: string
```

① **Компоненты многократного использования.**
② **Схемы многократного использования.**
③ **Имя схемы многократного использования.**
④ **Схема JSON.**

Использование предопределенного компонента с ссылкой

```
post:
   summary: Add product
   description: Add product to catalog
   [...]
   responses:
      "200":
         description: Product added to catalog
         content:
            application/json:
               schema: ①
               $ref: "#/components/schemas/product" ②
```

① **Схема ответа.**
② **Ссылка на предопределенную схему.**

Использование предопределенного компонента в массиве

```
get:
   summary: Search for products

   description: |
      Search using a free query (query parameter)
   parameters:
      [...]
   responses:
      "200":
         description: Products matching free query
         content:
            application/json:
               schema: ①
                  type: array ②
                  items: ③
                     $ref: "#/components/schemas/product" ④
```

① **Схема ответа.**
② **Массив.**
③ **Схема элементов массива.**
④ **Ссылка на предопределенную схему.**

Использование предопределенного параметра

```
components:
   parameters:
      productId: ①
```

```
        [...]
paths:
   /products:
   [...]
   /products/{productId}: ②
   delete:
      parameters:
         - $ref: #/components/parameters/productid ③
      [...]
   put:
      parameters:
         - $ref: #/components/parameters/productid ③
      [...]
   patch:
      parameters:
         - $ref: #/components/parameters/productid ③
      [...]
```

① **Определение параметра пути.**
② **Путь ресурса товара с параметром.**
③ **Ссылка на предопределенный параметр.**

Параметры уровня ресурса

```
components:
   parameters:
      productId: ①
         [...]
paths:
   /products:
   [...]
   /products/{productId}:
      parameters: ②
         - $ref: #/components/parameters/productId ③
```

```
    delete: ④
      [...]
    put: ④
      [...]
    patch: ④
      [...]
```

① **Определение параметра пути.**
② **Параметры уровня ресурса.**
③ **Ссылка на предопределенный параметр.**
④ **Определений параметров пути больше нета.**

Проектирование API

Цель проектирования программного обеспечения (или чего-либо еще), которое удовлетворяет потребности пользователей, не является чем-то новым. Это всегда было целью, сколько существует программное обеспе чение. Многочисленные методы существовали и по-прежнему изобре таются, для того чтобы коллекционировать потребности пользователей, глубже и точнее понимать их и наконец создать программное обеспечение, которое более или менее эффективно и точно выполняет свои задачи. Все эти методы могут использоваться для определения целей API.

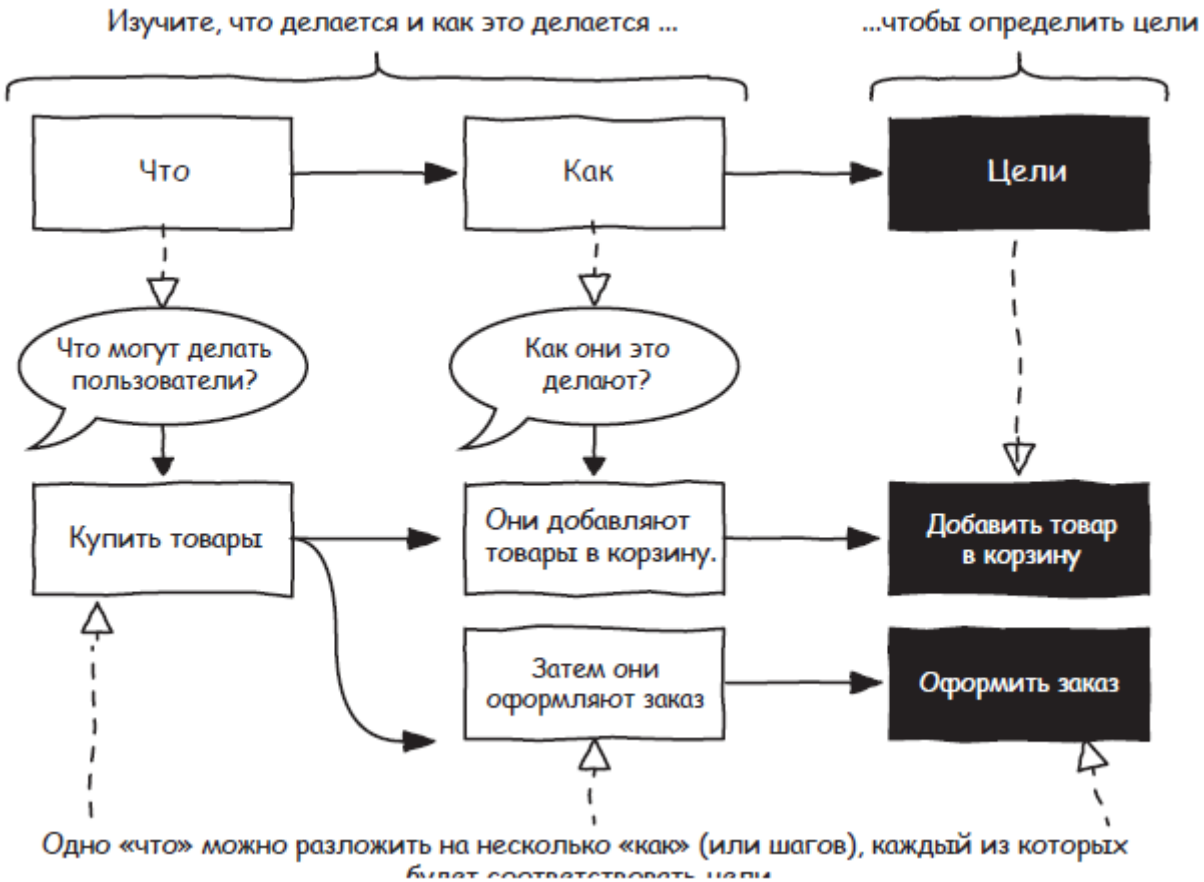
Первый шаг в процессе проектирования API-интерфейса – это установить, чего могут добиться его пользователи, – определить реальные цели API-интерфейса.

При разработке API важно иметь глубокие и точные знания о том,

- кто может использовать API;
- что он может делать;
- как он это делает;
- что ему нужно для этого;
- что он получает взамен.

Эти фрагменты информации являются основой вашего API. Они нужны для проектирования правильного программного интерфейса.

"Что?" и "Как?"



Что люди делают, когда совершают покупки в интернете? Покупают товары. А как они их покупают? Добавляют их в свою корзину, а затем оформляют заказ.

Таким образом, покупка товаров представляет собой процесс, состоящий из двух этапов, который можно представить с помощью следующих целей: добавить товар в корзину и оформить заказ. Без разбиения мы могли бы ошибочно перечислить цель покупки одного товара. Вот почему следует разложить действия на способы их реализации – если мы этого не сделаем, то можем пропустить какие-то цели.

Определение входных и выходных данных

Давайте теперь подробно рассмотрим каждый шаг, чтобы определить его входные и выходные данные.

Начнем с добавления товаров в корзину. Что нужно людям, чтобы это сделать? Очевидно, им нужен товар и корзина. Хорошо, а они получают что-то взамен, когда добавляют товар в свою корзину? Нет.

Нужно ли людям что-нибудь, чтобы оформить заказ, отправленный в корзину? Ясное дело, корзина. Ответ по-прежнему очевиден. А они получают что-то взамен? Да, подтверждение заказа.

Выявление недостающих целей

Как пользователь получает товар, чтобы добавить его в корзину? Это загадка. Кажется, мы упустили одну из целей, а может и больше. Как избежать этого? Универсального средства от этого не существует, но, изучая источники входных данных и использование выходных данных, можно обнаружить пропущенные «что» или «как», а следовательно, пропущенные или неопознанные цели. Итак, как пользователи получают товар, чтобы добавить его в корзину? Вероятно, сначала они ищут его по названию или описанию. Таким образом, мы можем добавить новый шаг «Поиск товаров» в список действий «Покупка товаров». Мы не должны забывать применить свой опрос к этому новому шагу.

Как пользователи управляют своими заказами? Они должны иметь возможность перечислять свои заказы в хронологическом порядке, чтобы проверить их статус.

Во-первых, список заказов: какие входные данные необходимы для составления списка заказов? Никаких. Что возвращается? Список заказов. Откуда поступают входные данные? Нам не нужны никакие входные данные для составления списка заказов, поэтому нам не нужно думать о том, откуда они берутся. Что мы делаем с выходными данными? Пользователи могут выбрать один из заказов в списке, чтобы проверить его статус. На этом все, что касается первого этапа управления заказами.

Второй этап, проверка статуса заказа: какие входные данные необходимы для проверки статуса заказа? Заказ. Что возвращается? Статус заказа. Откуда поступает заказ? Из корзины или списка заказов. Что мы делаем со статусом заказа? Нам нужно предоставить эти данные, чтобы проинформировать пользователей, не более того.

Необходимо расширить опрос:

- Что могут делать пользователи?
- Как они это делают?
- Что им нужно для этого?
- Что они получают взамен?
- Новый вопрос для определения недостающих целей: откуда поступают входные данные?
- Новый вопрос для определения недостающих целей: как используются выходные данные?

Идентификация всех пользователей

Идентификация различных типов пользователей является обязательной при построении исчерпывающего списка целей API. Таким образом, нужно добавить еще один вопрос, чтобы явно

идентифицировать их всех:

- Новый вопрос для идентификации всех пользователей и исключения пропущенных действий: кто такие пользователи?
- Что они могут делать?
- Как они это делают?
- Что им нужно для этого?
- Что они получают взамен?
- Откуда поступают входные данные?
- Как используются выходные данные?

Если сначала мы определим различные типы пользователей нашего API, нам будет проще составить полный список целей. Обратите внимание, что термин пользователь используется здесь в широком смысле; это может быть конечный пользователь, использующий приложение, которое потребляет API, само потребительское приложение или роли или профили конечного пользователя или потребительского приложения.

Использование таблицы целей

- Кто – здесь вы перечисляете пользователей API (или профили);
- Что – здесь вы перечисляете, что могут делать эти пользователи;
- Как – здесь вы разбиваете каждое действие на этапы;
- Входные данные (источник) – здесь вы перечисляете, что необходимо для каждого шага и откуда это берется (чтобы определить недостающих пользователей, действия или способы их реализации);
- Выходные данные (использование) – здесь вы перечисляете, что возвращает каждый этап и как это используется (чтобы определить недостающих пользователей, действия или способы их реализации);
- Цели – здесь вы четко и кратко переформулируете каждый способ реализации + входные данные + выходные данные.

Кто	Что	Как	Входные данные	Выходные данные	Цели
Клиенты	Покупка товаров	Поиск товаров	Каталог (управление каталогом), бесплатный запрос (предоставляется пользователем)	Товары (добавить товар в корзину)	Поиск товаров в каталоге с помощью бесплатного запроса
		Добавить товар в корзину	Товар (поиск товаров) корзина (принадлежит пользователю)		Добавить товар в корзину
Администратор	Управление каталогом	Добавить товар в каталог	Каталог (принадлежит пользователю), товар (предоставляется пользователем)		Добавить товар в каталог

Перечисление целей API – итерационный процесс. Вы должны действовать шаг за шагом – не пытаясь сделать все сразу. И вам также нужно будет уточнить и изменить этот список на основе соображений или ограничений, таких как удобство использования, производительность или безопасность.

Исключение влияние поставщика на API