# Практическая работа по теме "REST API и Swagger"

### Цель практики

* Продемонстрировать навык преобразования бизнес-требований в спецификацию REST API.
* Связать бизнес-сущности с REST-ресурсами и HTTP-методами.
* Показать умение проектировать структуру данных (JSON) для запросов и ответов.
* Подготовить базовую спецификацию в формате OpenAPI (Swagger) для передачи в команду разработки.

Вы работаете бизнес-аналитиком в компании, которая разрабатывает платформу для ведения блогов. Руководство поставило задачу реализовать базовый функционал для управления публикациями через API, чтобы в будущем можно было создать мобильное приложение и интегрироваться с внешними сервисами.

Ваша задача — спроектировать и описать REST API для управления статьями в блоге.

### Условие задания

1. Проанализируйте следующий описанный бизнес-сценарий.
2. На основе этого сценария:
   * Определите основные **ресурсы** API.
   * Спроектируйте **эндпоинты** (URI + HTTP-метод) для всех необходимых CRUD-операций (Create, Read, Update, Delete).
   * Опишите **структуру данных** (модель) для ресурса "Статья" в формате JSON.
   * Создайте фрагмент **спецификации OpenAPI 3.0** в формате YAML для одной операции (на ваш выбор), например, для создания новой статьи.
3. Обозначьте ключевые поля, их типы и обязательность.

### Описание бизнес-сценария

* Любой пользователь (даже неавторизованный) может получить список всех опубликованных статей.
* Любой пользователь может просмотреть одну конкретную статью по её уникальному идентификатору (ID).
* Авторизованный пользователь (Автор) может создать новую статью. При создании он должен указать заголовок и текст статьи. Изначально статья создается в статусе "черновик" (draft).
* Автор может редактировать **только свои** статьи. Он может изменить заголовок и текст.
* Автор может удалить **только свою** статью.
* У каждой статьи есть следующие атрибуты:
  + id (уникальный идентификатор, генерируется сервером)
  + title (заголовок, обязательное поле)
  + content (текст статьи, обязательное поле)
  + authorId (ID автора, присваивается сервером)
  + status (статус: draft или published)
  + createdAt (дата создания, присваивается сервером)
  + updatedAt (дата последнего обновления, присваивается сервером)

### Решение (какие артефакты подготовить)

**1. Ресурсы и Эндпоинты**

* **Ресурс:** Коллекция статей (/articles) и конкретная статья (/articles/{id}).
* **Эндпоинты:**
  + Получение списка статей: GET /articles
  + Создание новой статьи: POST /articles
  + Получение конкретной статьи: GET /articles/{id}
  + Обновление статьи: PUT или PATCH /articles/{id}
  + Удаление статьи: DELETE /articles/{id}

**2. Модель данных (JSON Schema)**

Опишите JSON-объект для сущности "Статья", указав типы полей.

**3. Фрагмент спецификации OpenAPI (YAML)**

Напишите код для описания эндпоинта POST /articles, включая:

* summary и description.
* requestBody с описанием тела запроса (ссылка на схему ArticleInput).
* responses с описанием успешного ответа 201 Created и возможных ошибок (400 Bad Request, 401 Unauthorized).

### Правильный ответ (ожидаемый результат)

**1. Список эндпоинтов**

|  |  |  |
| --- | --- | --- |
| **HTTP-метод** | **URI** | **Описание** |
| GET | /articles | Получить список всех опубликованных статей |
| POST | /articles | Создать новую статью (требуется авторизация) |
| GET | /articles/{id} | Получить статью по ID |
| PUT | /articles/{id} | Полностью обновить статью (требуется авторизация) |
| DELETE | /articles/{id} | Удалить статью (требуется авторизация) |

**2. Модель данных "Статья" (Article)**

{

  "id": "c7a7e3d1-3b7c-4a32-8e8a-3e3c5f2b1a6e",

  "title": "Заголовок моей статьи",

  "content": "Полный текст моей статьи...",

  "authorId": "user-12345",

  "status": "published",

  "createdAt": "2024-09-21T12:30:00Z",

  "updatedAt": "2024-09-21T12:30:00Z"

}

**3. Фрагмент спецификации OpenAPI для POST /articles**

paths:

  /articles:

    post:

      summary: Создать новую статью

      description: Создает новую статью в статусе "черновик". Требуется авторизация.

      tags:

        - Статьи

      requestBody:

        description: Данные для новой статьи

        required: true

        content:

          application/json:

            schema:

              $ref: '#/components/schemas/ArticleInput'

      responses:

        '201':

          description: Статья успешно создана

          content:

            application/json:

              schema:

                $ref: '#/components/schemas/Article'

        '400':

          description: Неверный запрос (например, не указан заголовок)

          content:

            application/json:

              schema:

                $ref: '#/components/schemas/Error'

        '401':

          description: Пользователь не авторизован

components:

  schemas:

    Article:

      type: object

      properties:

        id:

          type: string

          format: uuid

          description: Уникальный идентификатор статьи

        title:

          type: string

          description: Заголовок статьи

        content:

          type: string

          description: Содержимое статьи

        authorId:

          type: string

          description: Идентификатор автора

        status:

          type: string

          enum: [draft, published]

          description: Статус статьи

        createdAt:

          type: string

          format: date-time

        updatedAt:

          type: string

          format: date-time

    ArticleInput:

      type: object

      properties:

        title:

          type: string

          description: Заголовок статьи

        content:

          type: string

          description: Содержимое статьи

      required:

        - title

        - content

    Error:

      type: object

      properties:

        code:

          type: integer

        message:

          type: string