Шпаргалка по REST API

Краткая шпаргалка по RESTful веб-сервисам

Для новичков, изучающих REST API

Содержание

1 Что такое REST API?

REST (Representational State Transfer) — архитектурный стиль для веб-сервисов, где ресурсы доступны через HTTP-запросы.

Для кого: Новички (основы программирования) и мидлы (лучшие практики). **Требуется**: Знание HTTP, JSON, основ Python.

2 Основы REST

2.1 Принципы

- **Клиент-сервер**: Разделение интерфейса и сервера.
- **Stateless**: Запросы независимы, содержат все данные.
- **Кэширование**: Ответы кэшируются (Cache-Control).
- **Единообразие**: Стандарты URI, методов, заголовков.
- **Слои**: Промежуточные серверы (прокси).
- **HATEOAS**: Гипермедиа-ссылки в ответах (`rel`, `href`).

2.2 Термины

Ресурс Сущность (например, пользователь).

URI Адрес ресурса (`https://api.example.com/v1/users/1`).

HTTP-метод Действие (GET, POST).

Статус-код Результат (200 OK).

Заголовок Метаданные (Content-Type).

3 Компоненты REST API

3.1 HTTP-методы

- `GET`: Получить ресурс.
- `POST`: Создать ресурс.
- `PUT`: Обновить полностью.
- `PATCH`: Обновить частично.
- `DELETE`: Удалить ресурс.

3.2 Статус-коды

- `200 OK`: Успех.
- `201 Created`: Создан.
- `204 No Content`: Успех без данных.
- `400 Bad Request`: Неверный запрос.

- `401 Unauthorized`: Нет аутентификации.
- `403 Forbidden`: Доступ запрещён.
- `404 Not Found`: Не найден.
- `429 Too Many Requests`: Лимит запросов.
- `500 Internal Server Error`: Ошибка сервера.

3.3 URI-дизайн

- Существительные: `https://api.example.com/v1/users`.
- Версионирование: `https://api.example.com/v1`.
- Фильтрация: `https://api.example.com/v1/users?role=admin`.

3.4 Заголовки

- `Content-Type`: `application/json`.
- `Authorization`: `Bearer <token>`.
- `Accept`: Формат ответа.

3.5 Форматы данных

- **JSON**: Основной формат.
- **XML**: Реже используется.

4 Антипаттерны REST

- Глаголы в URI: `/getUsers`.
- Глубокая вложенность: `/users/1/orders/2/items`.
- Неправильные статус-коды: `200` для ошибок.

5 Безопасность

- **CSRF**: Токены для POST, PUT, PATCH, DELETE.
- **CORS**: Ограничение доменов.
- **HTTPS**: Шифрование TLS.
- **Rate-limiting**: Лимит запросов (429).
- **JWT**: Токены для аутентификации.

6 Обработка ошибок

Используйте RFC 7807 (Problem Details): Поля: `type`, `title`, `status`, `detail`, `instance`. Пример:

```
{
  "type": "https://tools.ietf.org/html/rfc7807#section-3",
  "title": "Not found",
  "status": 404,
  "detail": "User not found",
  "instance": "/v1/users/1"
}
```

7 Масштабируемость и мониторинг

- **Пагинация**: `https://api.example.com/v1/users?limit=10&offset=20`.
- **Логирование**: Запись запросов и ошибок.
- **Метрики**: Время ответа, ошибки (Prometheus).
- **Трейсинг**: OpenTelemetry для отслеживания.

8 Пример FastAPI

```
from fastapi import FastAPI, HTTPException

app = FastAPI()
users = {}

@app.post("/v1/users")
async def create_user(id: int, name: str, email: str):
    if id in users:
        raise HTTPException(status_code=400, detail="User exists")
    users[id] = {"id": id, "name": name, "email": email}
    return users[id]

@app.get("/v1/users/{id}")
async def get_user(id: int):
    if id not in users:
        raise HTTPException(status_code=404, detail="User not found")
    return users[id]
```

Описание: Минимальное API для создания и получения пользователя.