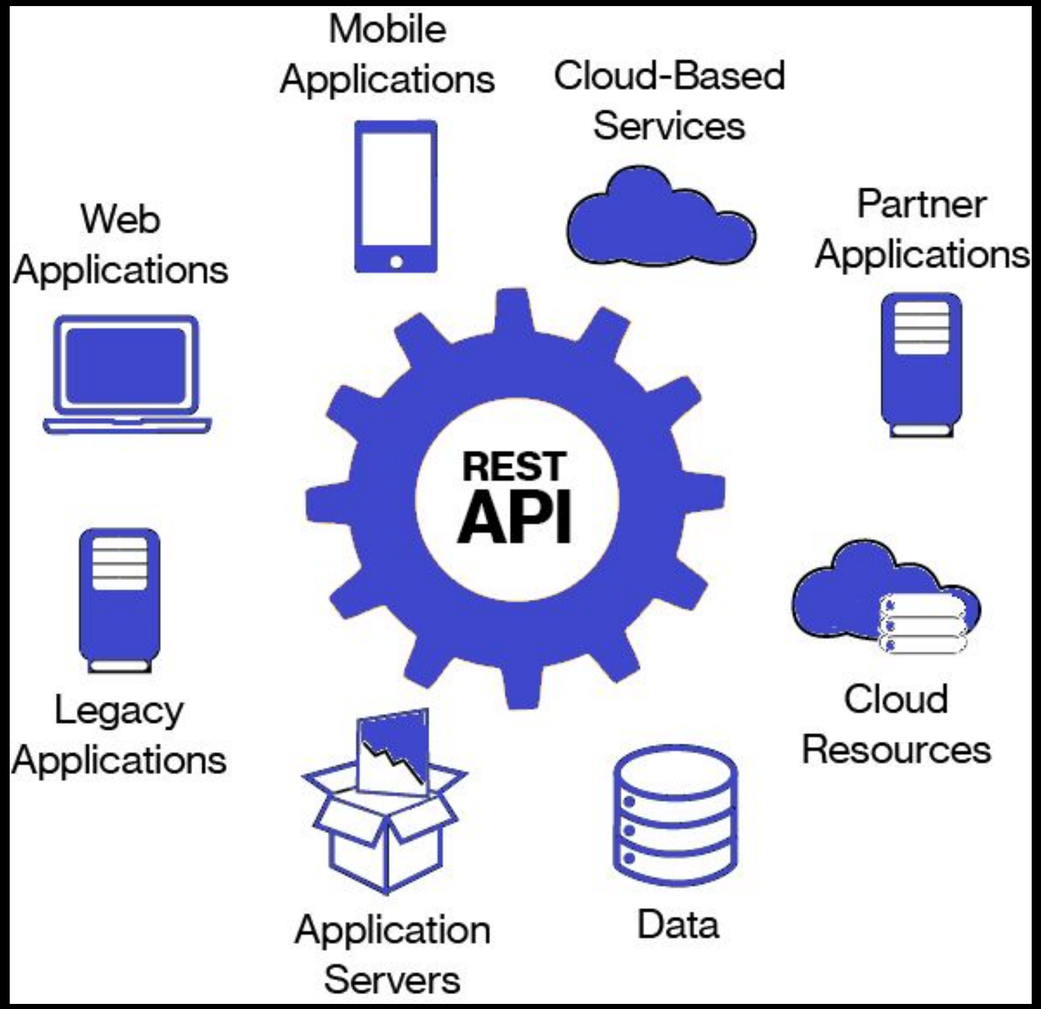
Проверить звук

Включить запись

Проектирование API
HTTP
REST

Что такое и зачем нужен API?



Принципы REST

- Клиент-серверная архитектура
- Независимость от платформы клиента
- Сервер не хранит состояние клиента
- В основе лежит *ресурс* (пост, автор, книга)
- Уникальные идентификаторы ресурсов
- Клиенты получают *представление* ресурса
- Операции над ресурсами HTTP «глаголами»

Методы HTTP и идемпотентность

GET	Получение ресурса	безопасный	идемпотентный	200_OK, 404_NOT_FOUND
POST	Создание ресурса	не безопасный	не идемпотентный	201_CREATED, 403_FORBIDDEN
PUT	Изменение/создание ресурса	не безопасный	идемпотентный	200_OK, 400_BAD_REQUEST
PATCH	Частичное изменение ресурса	не безопасный	не идемпотентный	200_OK, 400_BAD_REQUEST
DELETE	Удаление ресурса	не безопасный	идемпотентный	204_NO_CONTENT

Стажер Вася и его история об идемпотентности API: https://habr.com/ru/company/yandex/blog/442762/

Требования к API

- Консистентность
  - общие форматы
    - URL запросов
    - Данных

- Расширяемость
  - изменение параметров объекта
  - добавление дополнительных данных

- Документация
  - Open API

Требования к API - Консистентность

Post(id, text, pub_date, author, group)

GET /api/v7/posts/123

```
[123, “this is a post”, 1586069352, 1, 12]
```

GET /api/v7/groups/12

```
{
  “id”: 12,
  “title”: “Authors”,
  “description”: “users who write posts”
}
```

GET /api/v1/posts/123/comments

```
{
  “status”: 200000,
  “body”: {
    “comments”: [
      {“id”: 1, “text”: “baka”}
    ]
  }
}
```

Требования к API - Расширяемость (1)

Post(id, text, pub_date, author, group)

GET /api/v7/posts/123

[123, “this is a post”, 1586069352, 1, 12]

Требования к API - Расширяемость (2)

Post(id, text, pub_date, author, group)

GET /api/v7/posts/123
[123, “this is a post”, 1586069352, “2020-09-05”, 1, 12]

- Что если мы захотим вставить дату последнего изменения публикации после даты создания?

Требования к API - Расширяемость (3)

GET /api/v7/posts/123

```
[
  {
    "id": 123,
    "text": "this is a post",
    "pub_date": 1586069352,
    "author_id": 1,
    “group_id”: 12,
    “las_date”: “2020-09-05”
  }
]
```

Требования к API - Расширяемость (4)

GET /api/v7/posts/123

```
[
   {
      "id": 123,
      "text": "this is a post",
      "pub_date": 1586069352,
      "author_id": 1,
      “group_id”: 12
   }
]
```

Что если мы захотим возвращать ссылки на похожие посты?

Требования к API - Расширяемость (5)

GET /api/v7/posts/123

```
{
   “post”: {
      "id": 123, "text": "this is a post", "pub_date": 1586069352, "author_id": 1, “group_id”: 12
   }
}
```

Требования к API - Расширяемость (6)

GET /api/v7/posts/123

```
{
   “post”: {
      "id": 123, "text": "this is a post", "pub_date": 1586069352, "author_id": 1, “group_id”: 12
   },
   “_links”: {
      “similar_posts”: [
         {“href”: “/api/v7/posts/12”},
         {“href”: “/api/v7/posts/25”}
      ]
   }
}
```

Требования к API - Расширяемость (6)

GET /api/v7/posts/123

```
{
   “post”: {
      "id": 123, "text": "this is a post", "pub_date": 1586069352, "author_id": 1, “group_id”: 12
   },
   “_links”: {
      “similar_posts”: [
         {“href”: “/api/v7/posts/12”},
         {“href”: “/api/v7/posts/25”}
      ]
   }
}
```

Что если нам нужно передать метаданные?

Требования к API - Расширяемость (7)

GET /api/v7/posts/123

```
{
   “status”: 200000, // статус ошибки приложения - бизнес-логика
   “body”: {
      “post”: {
         "id": 123, "text": "this is a post", "pub_date": 1586069352,
         "author_id": 1, “group_id”: 12
      },
      “_links”: {
         “similar_posts”: [
            {“href”: “/api/v7/posts/12”}
         ]
      }
   }
}
```

Требования к API - Расширяемость (8)

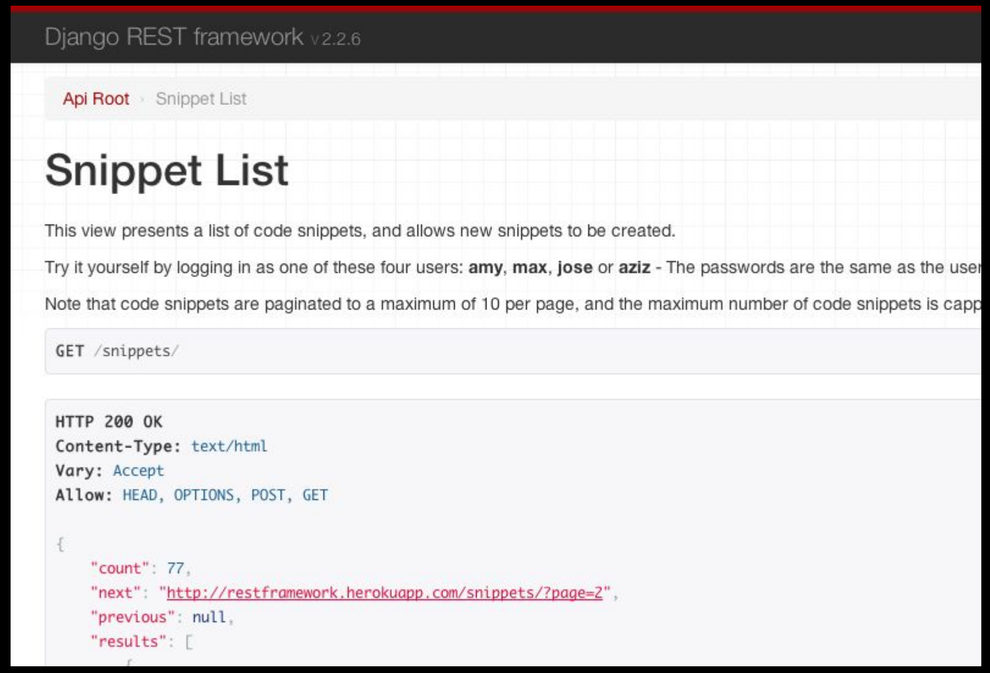
GET /api/v7/posts/123

```
{
   “status”: 200000, // успешный сценарий
   “body”: {
      “post”: {
         "id": 123, "text": "this is a post"
      }
   }
}
```

```
{
   “status”: 500001, // проблема с подключением к базе данных
   “body”: {
      “error”: “database connection failed”
   }
}
```

Требования к API - Документация

- Open API / Swagger / ReDoc
  - пример: https://flask-open-api.herokuapp.com/openapi/swagger-ui
- Django REST Framework - Self-describing API



Анатомия HTTP запросов:

- Конечная точка (endpoint)
- Метод (GET, POST, PUT, или DELETE)
- HTTP Заголовки (“User-Agent”, “Content-Type”, “Content-Length”)
- Данные (или тело; ответное сообщение)

curl https://flask-open-api.herokuapp.com/pets/ -v

Важные моменты:

- Использовать правильные HTTP методы
- В качестве ресурсов использовать существительные

  authors, books, orders
- Иерархия сущностей от общего к частному

  POST /api/v1/posts/123/comments/456
- Правильные HTTP коды с описанием ошибки
- Версионирование API
- Отдельный URL для API
- Пагинация

  GET /api/v1/posts?limit=500&offset=100

Получение данных POST запросом?

- необходимость фильтрации данных
- нужно указывать условия по которым фильтруем
- чем больше система, тем больше может быть фильтров
- ограничение объема передаваемой информации в адресной строке

Источник изображения: https://blazepress.com/2014/03/25-hilarious-photos-animals-looking-shocked/

Дополнительные материалы для ознакомления:

Обзорная серия статей по проектированию REST API

https://habr.com/en/post/483202/

Статус коды используемые в ответах на запросы

https://www.django-rest-framework.org/api-guide/status-codes/

https://ru.wikipedia.org/wiki/Список_кодов_состояния_HTTP