API

Регламент вебинара

- Выключаем микрофоны
- Вопросы задаем в чат
- Не отвлекаемся
- Активно работаем

План встречи

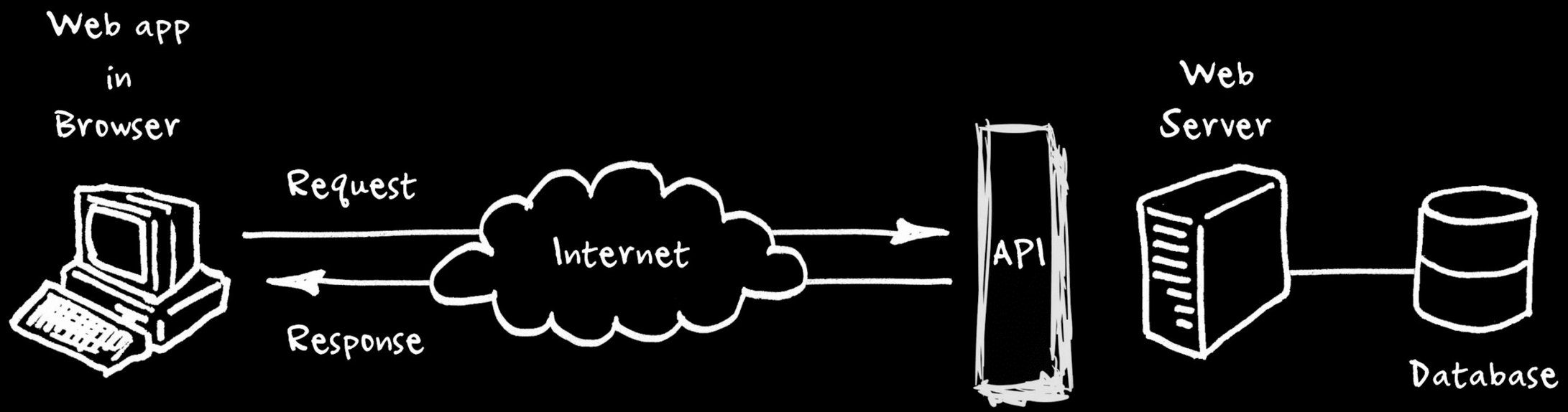
1. API,как устроены современные приложения?
2. REST/SOAP
3. Документация на API
4. Составление тестовой документации для работы с API

Технический слайд

Смотрим НН
Смотрим виды АПИ
Мемы

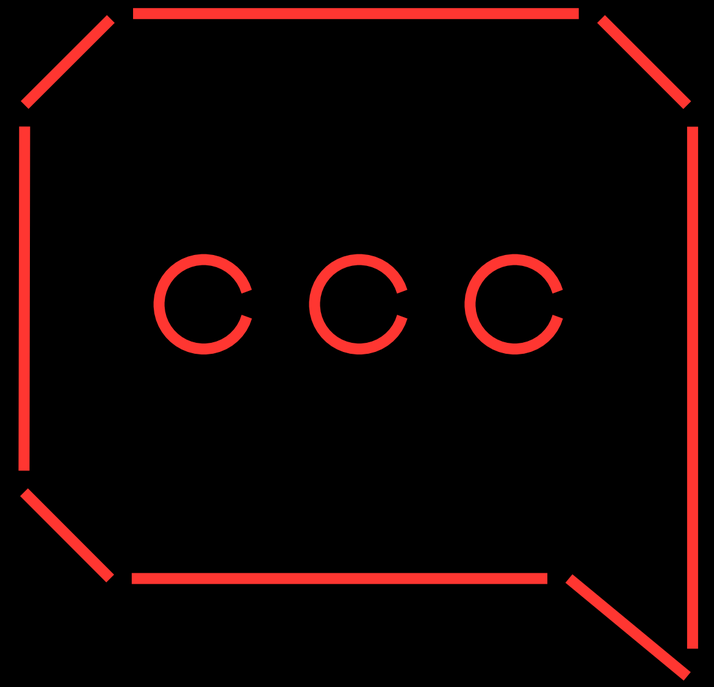
Что такое API?

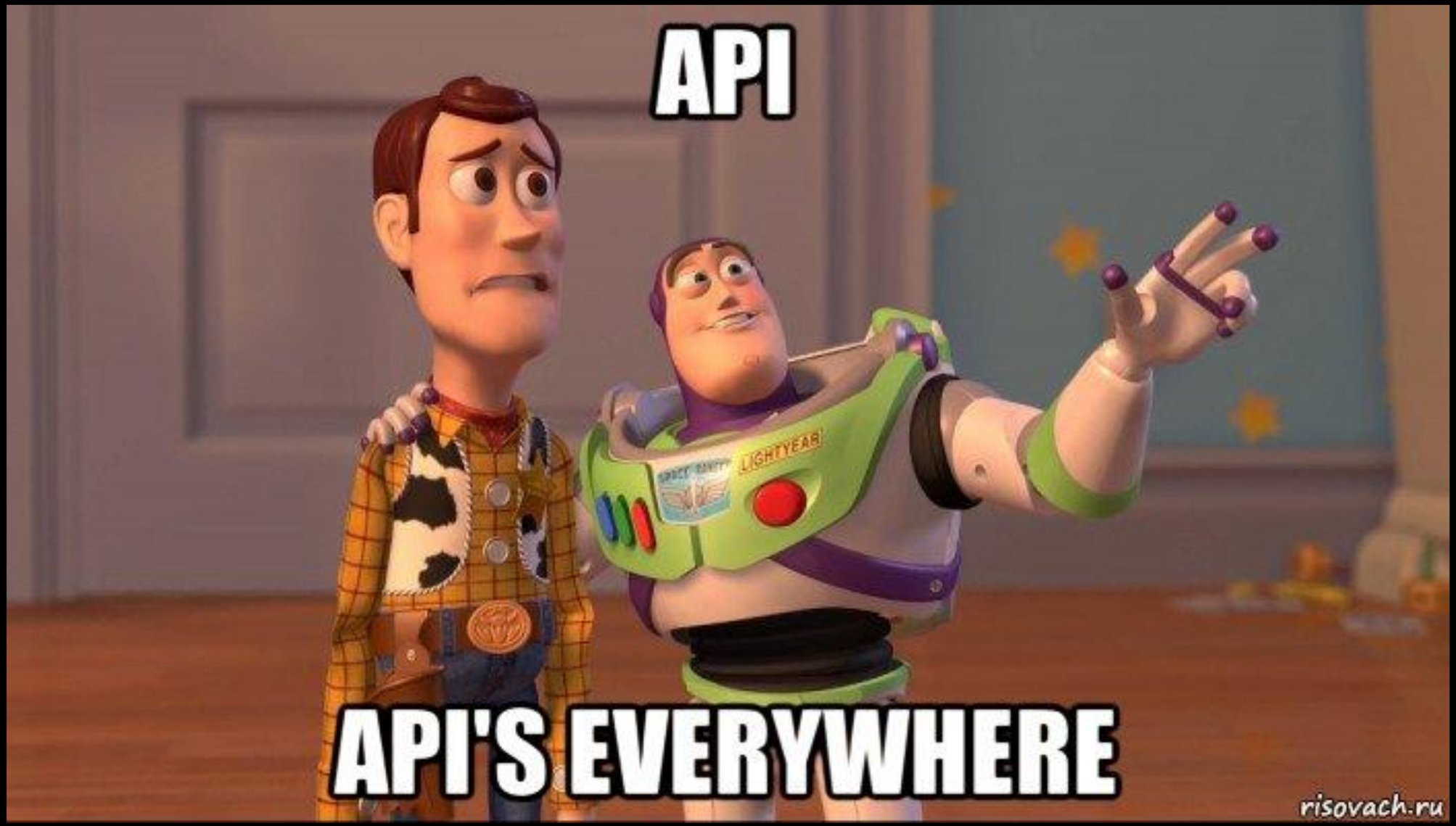
API - **A**pplication **P**rogramming **I**nterface



Загуглить
REST API
SOAP
gRPC
Graphql

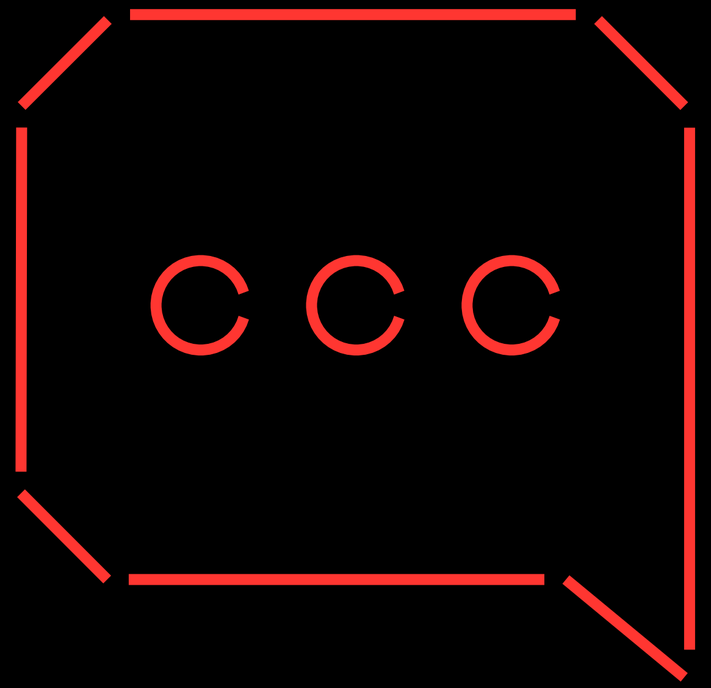
С какими API вы работаете каждый день?




API
API'S EVERYWHERE
risovach.ru

С какими API вы работаете каждый день?

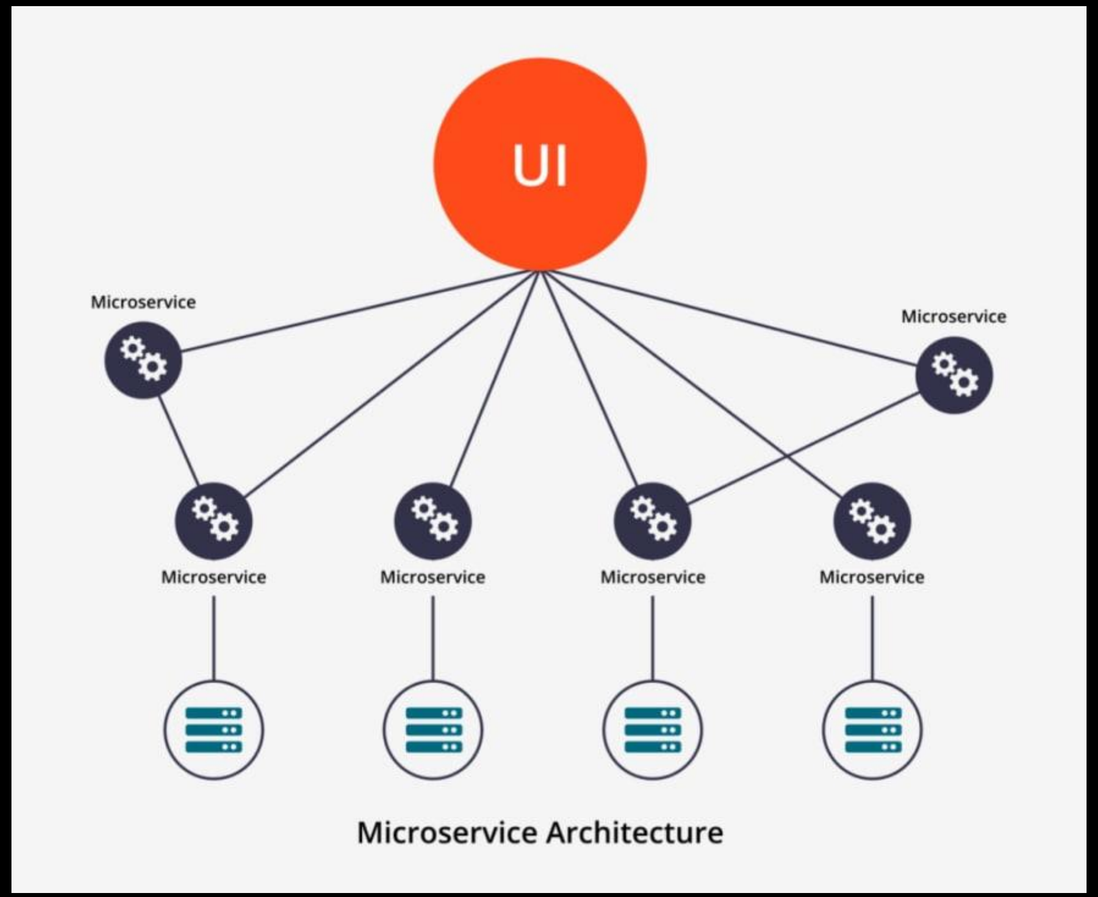
- Slack – **клиент на устройстве отправляет запросы и получает данные с серверов через API**
- Приложение банка
- Заказ <*что_угодно*> через сайт



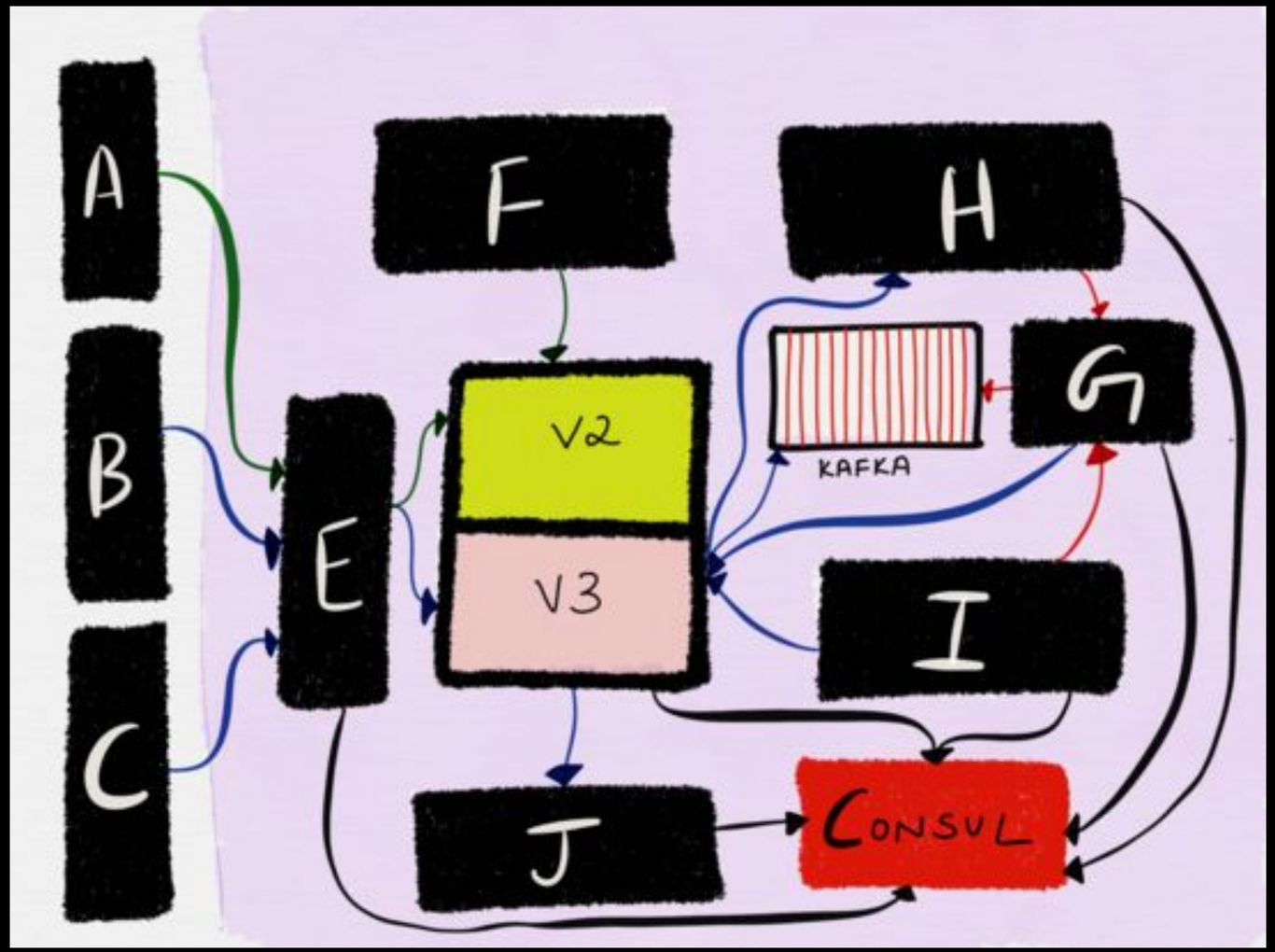
Почему важно уметь тестировать API?

- Существуют приложения без графического интерфейса
- Локализация ошибок и раздельное тестирование Бэк/Фронт
- Проверка сценариев, которые невозможно проверить через интерфейс – можно проверить больше
- Гибкий и быстрый способ генерации тестовых данных
- Не все тестировщики умеют тестировать API

Реальное приложение может быть устроено так



Но чаще оно устроено так



Загуглить
Monolith
SOA
MSA
KAFKA

Форматы данных в API (JSON, XML)

REST: что важно помнить

1. Работает на основании HTTP/HTTPS
2. Основной формат передачи данных —JSON (есть и другие)
3. 6 принципов RESTful
4. Простой и гибкий
5. Запрос/ответ состоит из 3 частей: Строка состояния, заголовки, тело

SOAP: что важно помнить

1. Работает на основании SOAP/HTTP/HTTPS
2. Основной формат передачи данных —XML строго!
3. Сообщение состоит из 3-х частей Envelop, Header, Body
4. WSDL – язык описания сервисов и доступа к ним

JSON *(JavaScript Object Notation)*

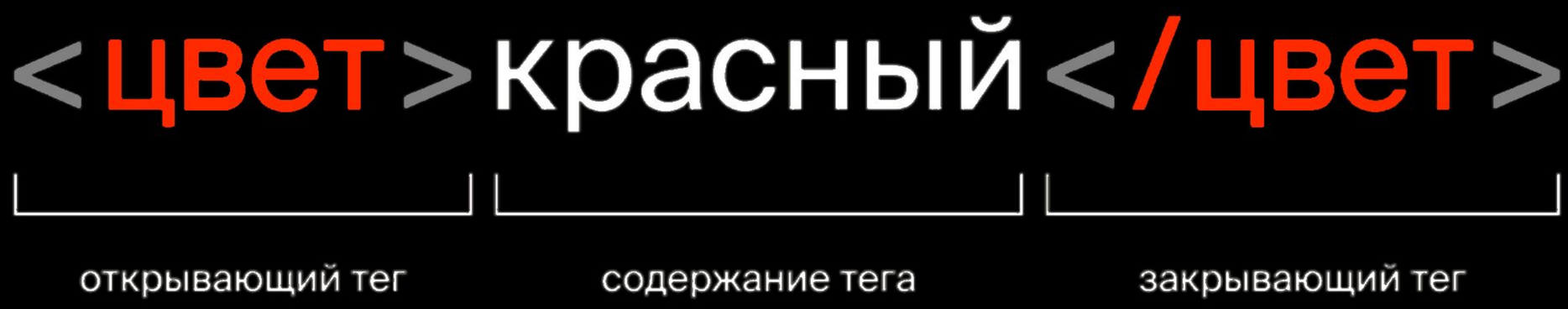
- **Набор пар ключ-значение (как в словаре)**
- **Порядок пар не важен, может меняться от запроса к запросу**
- **Ключ — всегда строка**
- Значение:
- строка
- число
- null
- Логический
- true или false
- массив (перечисление через запятую)
- объект (другой JSON)

```
{
  "first_name": "Ivan",
  "last_name": "Ivanov",
  "middle_name": "Ivanovich",
  "age": 35,
  "is_married": true,
  "has_kids": false,
  "emails": ["ivanov.i.i.@yandex-work.ru", "ivan@yandex-home.ru"],
  "favorite_numbers": [0, 7, 42],
  "pets": [{"type": "dog", "name": "Шарик"}, {"type": "cat", "name": "Мурлыка"}]
}
```

JSON *(пример)*

```
{
  "name": "Иван Иванович",
  "age": 24,
  "isMarried": false,
  "children": null,
  "cars": ["а123бв10", "б321ав01"],
  "mobilePhone": {
    "name": "Samsung",
    "made": 2015
  }
}
```

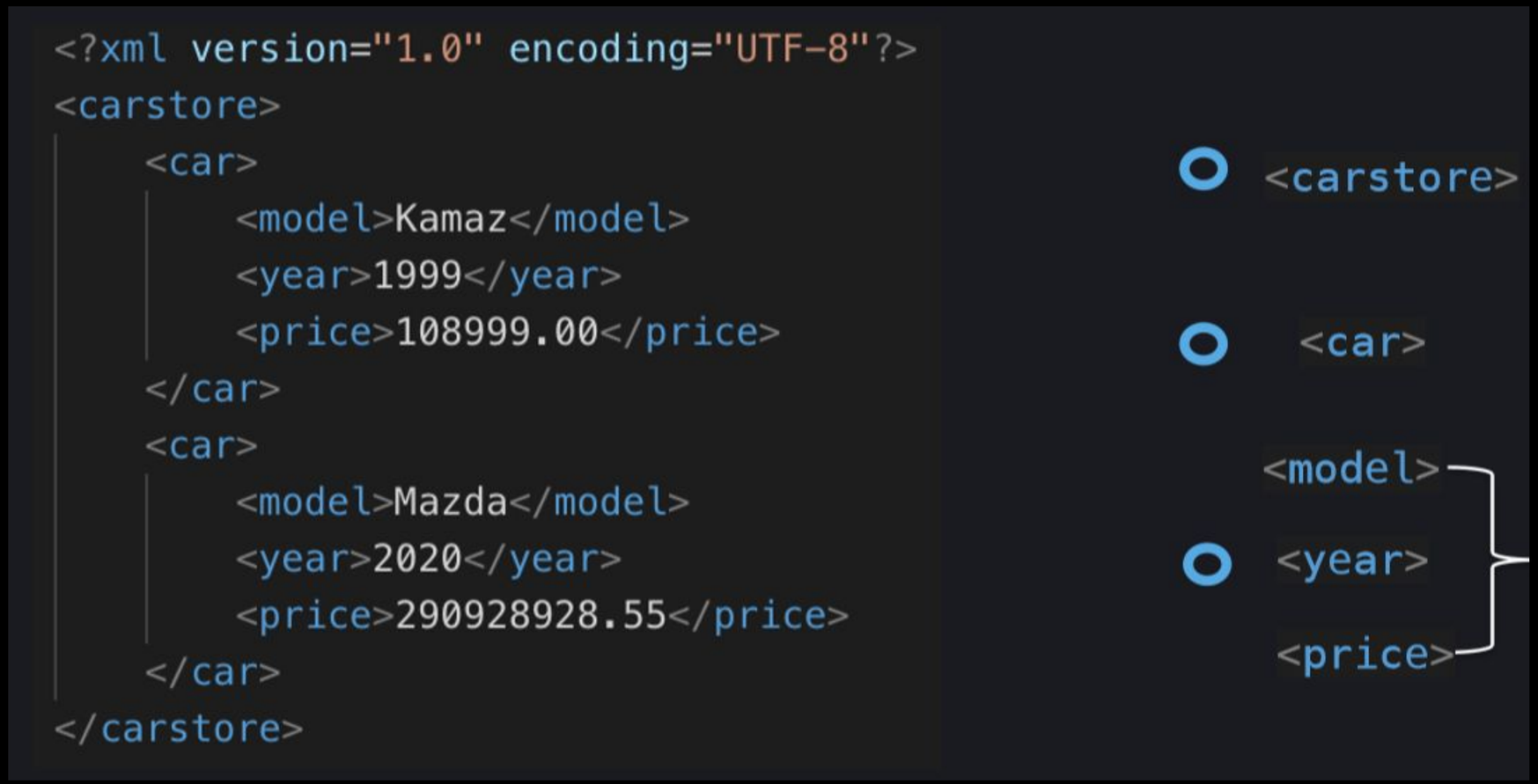
XML *(e**X**tensible **M**arkup **L**anguage)*



Каким может быть ***содержание тега***?

- Строка, число и т.д.
- Пустое
- *Открывающий тег может содержать атрибуты*
- Другой тег

Структура на примере простого XML



- Корневой тег (Родительский)

- Потомок и родитель

- Потомки (дочерние теги)

Структура XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<carstore>
    <car>
        <model lang="en">Kamaz</model>
        <year>1999</year>
        <price currency="RUB">108999.00</price>
    </car>
    <car>
        <model lang="en">Mazda</model>
        <year>2020</year>
        <price currency="RUB">290928928.55</price>
    </car>
</carstore>
```

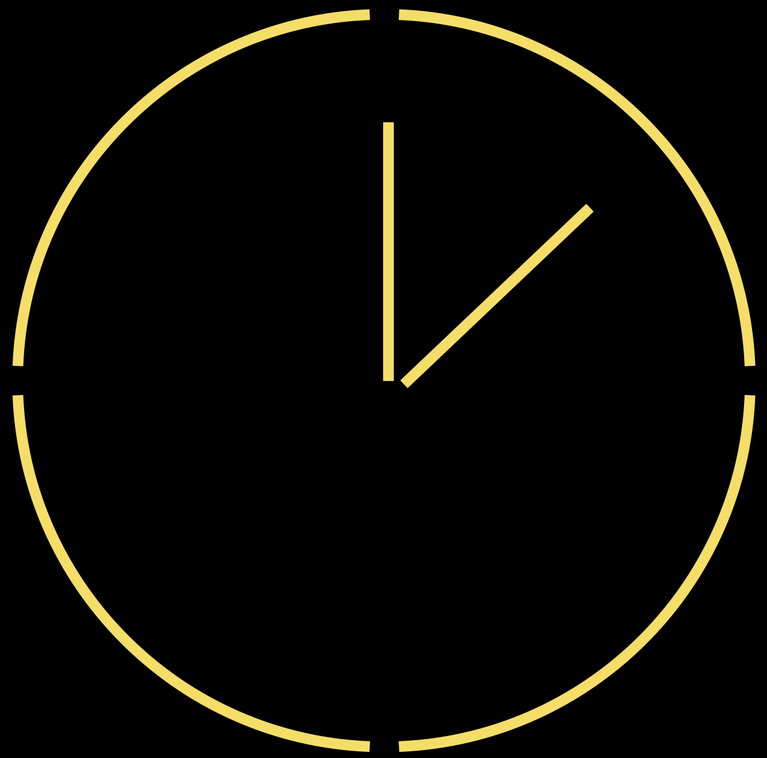
Декларация описывает версию XML документа и кодировку

```xml
<?xml version="1.0" encoding="UTF-8"?>
```

Атрибуты хранят в себе дополнительную информацию

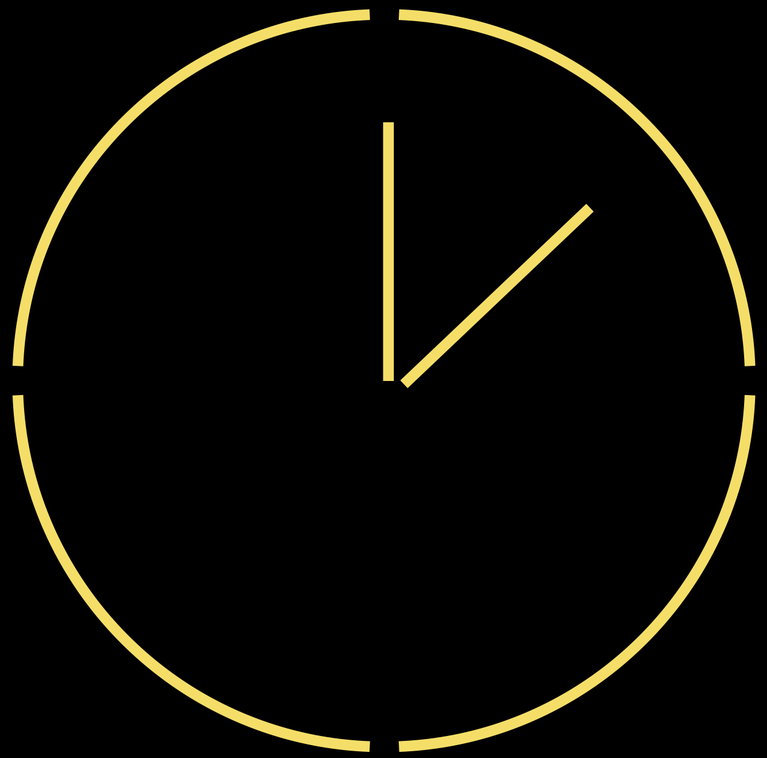
```xml
<model lang="en">
<price currency="RUB">
```

XML и HTML



Одно и то же?

XML и HTML



HTML – HyperText Markup Language

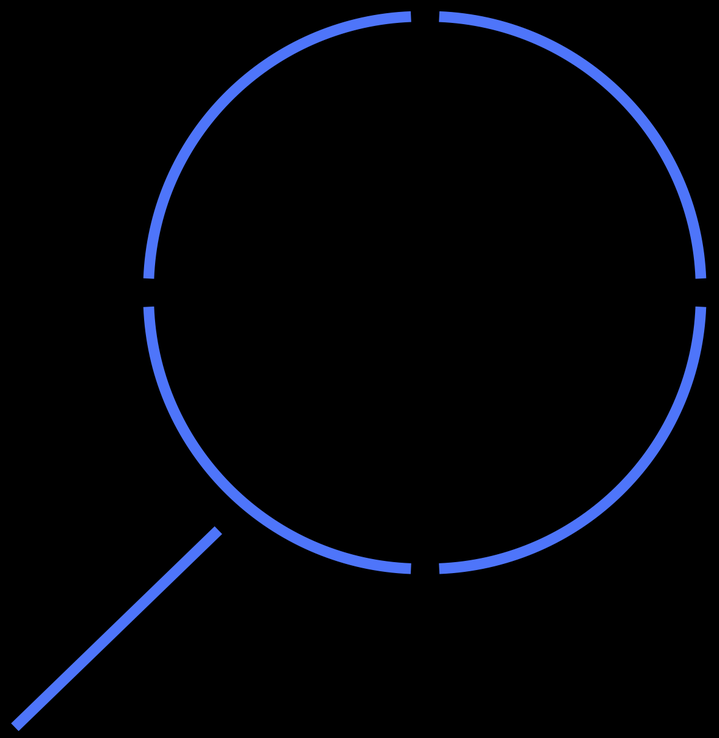
XML – eXtensible Markup Language

XML vs HTML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<carstore>
    <car>
        <model lang="en">Kamaz</model>
        <year>1999</year>
        <price currency="RUB">108999.00</price>
    </car>
    <car>
        <model lang="en">Mazda</model>
        <year>2020</year>
        <price currency="RUB">290928928.55</price>
    </car>
</carstore>
```

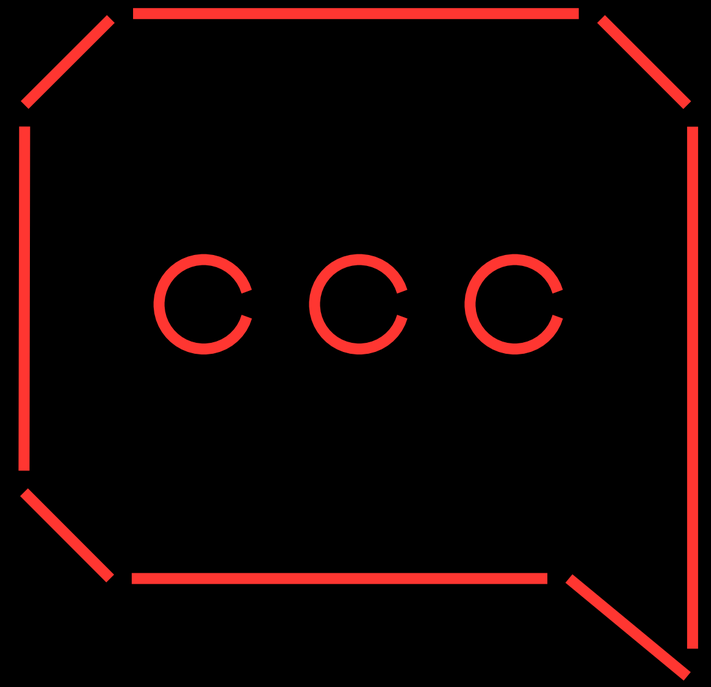
```html
<!DOCTYPE html>
    <html lang="en">
    ...
    <section>
        <p class="car">kamaz</p>
        <p>2005</p>
        <img src="" alt="foto">
    </section>
    <section>
        <p class="car">mazda</p>
        <p>2019</p>
        <p>zum-zum</p>
    </section>
    ...
</html>
```

Где писать JSON/XML?



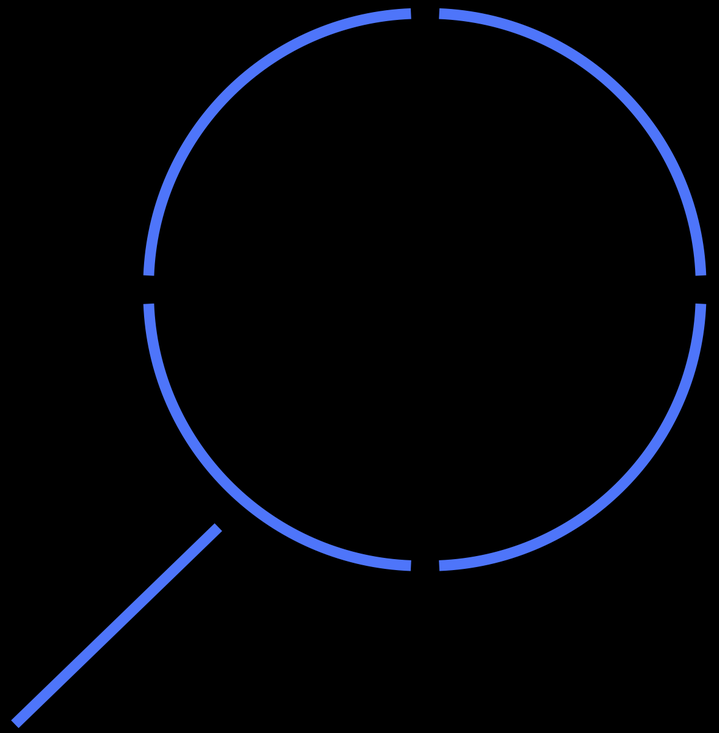
- Блокнот
- Notepad++
- Sublime Text
- Online: JSONViewer
- programmer's notepad
- VScode

Вопросы
про теорию API ?



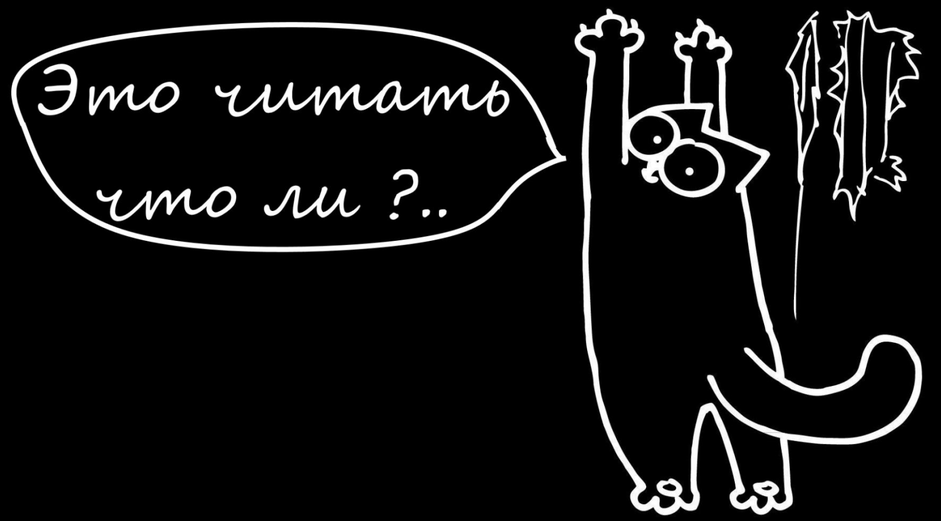
Документация API

Документация для API

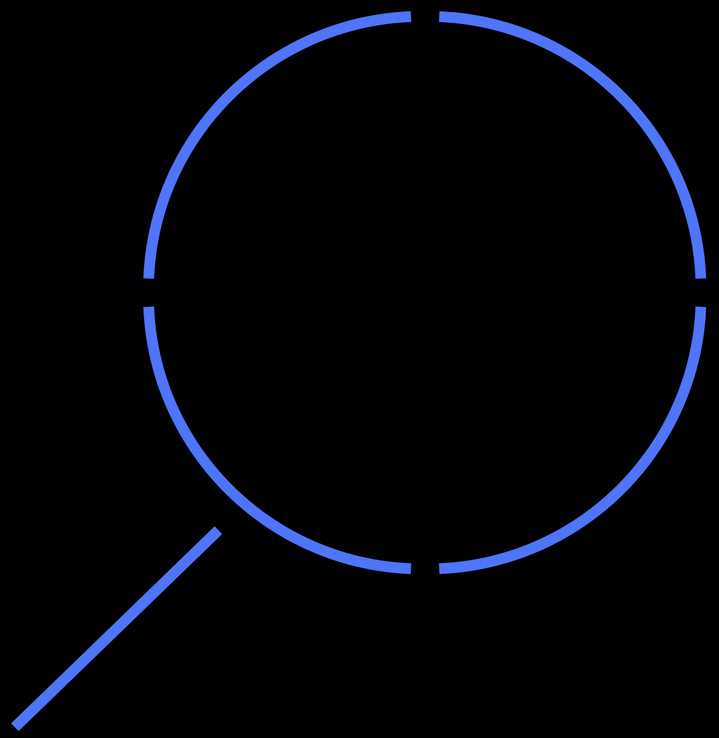


Зачем нужна?

Где взять?



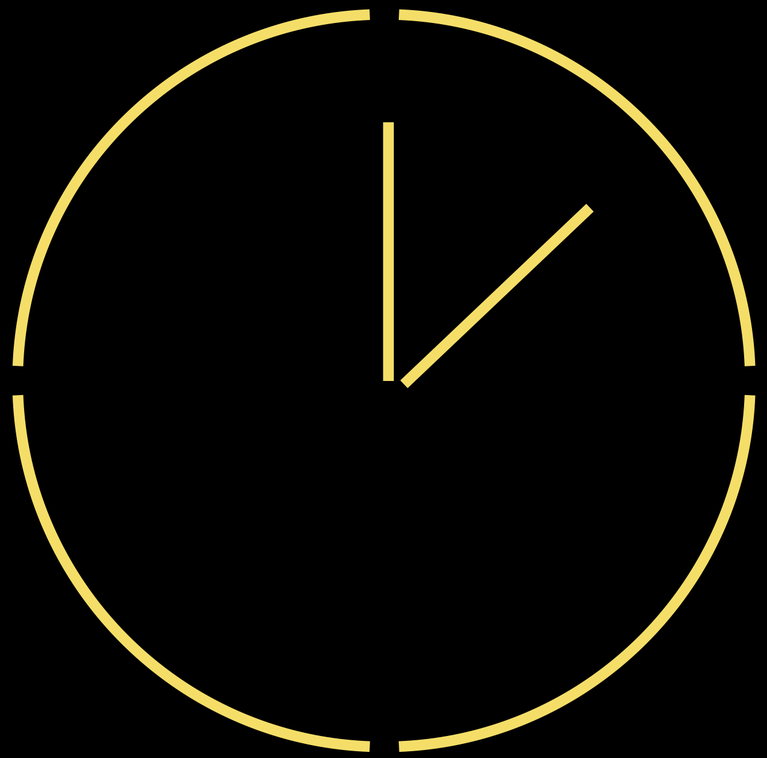
Документация для API



- Swagger
- Apidoc: пример
- Текстовые документы
- Разработчик
- …

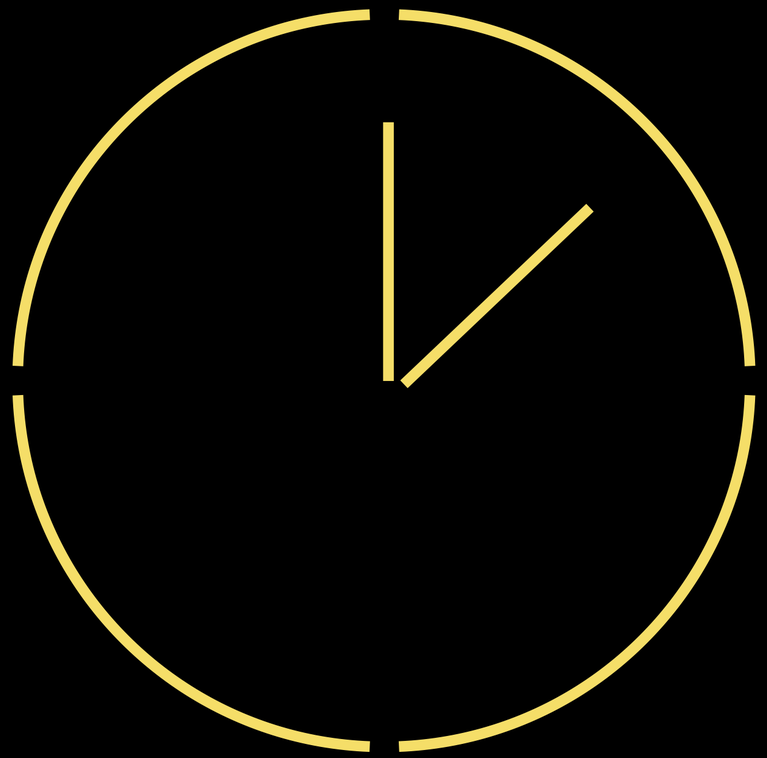
Немного про HTTP

Какие задачи решает HTTP?



- Структура передачи данных
- Передача документов
- Передача меты
- Авторизация
- Поддержка сессий
- Кеширование документов
- Согласование содержимого

Почему HTTP?



- Работает поверх TCP/TLS (нужно надежное соединение)
- Запрос ответ / нет стриминга
- Текстовый (человек может читать)
- Расширяемый

Из чего состоит HTTP-запрос

Запрос

```
GET /wiki/страница HTTP/1.1
Host: ru.wikipedia.org
User-Agent: Mozilla/5.0 (X11; U; Linux i686; ru; rv:1.9b5) Gecko/2008050509
Firefox/3.0b5
Accept: text/html
Connection: close
(пустая строка)
```

Ответ

```
HTTP/1.1 200 OK
Date: Wed, 11 Feb 2009 11:20:59 GMT
Server: Apache
X-Powered-By: PHP/5.2.4-2ubuntu5wm1
Last-Modified: Wed, 11 Feb 2009 11:20:59 GMT
Content-Language: ru
Content-Type: text/html; charset=utf-8
Content-Length: 1234
Connection: close
(пустая строка)
(далее следует запрошенная страница в HTML)
```

- **URL / endpoint / «ручка»**
- **Метод**
- **Query-параметры**
- **Заголовки**
- **Тело запроса**

Как тестировать API?

Как тестировать API?

Так же, как и все остальное — по документации

- **что должен делать конкретный сервис?**
- **какие есть заголовки и параметры?**
- **структура запроса**
- **какие параметры обязательны**
- **типы данных**
- **валидация**

Как тестировать API?

Что проверять?

- Статус-код ответа
- Структура ответа
- Если вносились изменения, то они видны:
- в приложении
- в базе-данных
- в ответе этой или другой ручки
- Прикладывай CURL!

Статус-коды:

1xx — информационные

2xx — успешное выполнение

3xx — перенаправление

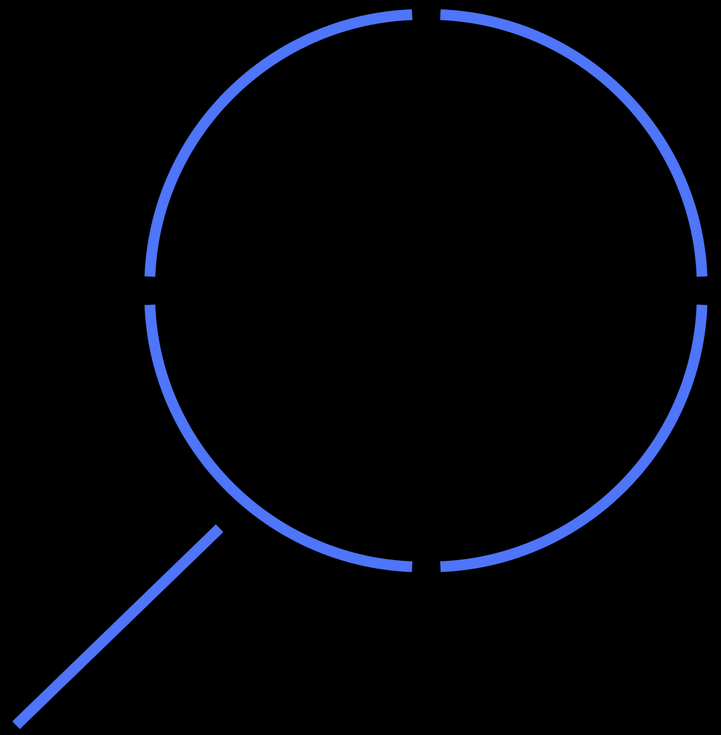
4xx — ошибка на стороне клиента

5xx — ошибка на стороне сервера

4xx иногда заменяют на 2xx

Тестовая документация: тестирование API

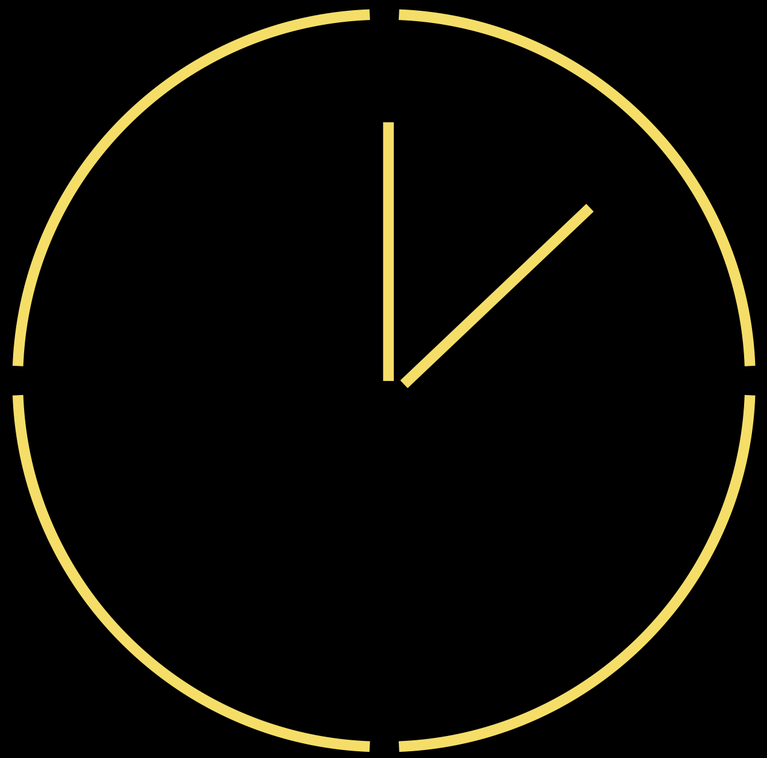
Тестовая документация: тестирование API



Документация всё та же ;)

- Чеклисты
- Тест-кейсы

Тестовая документация: тестирование API



Важно помнить:

- В основе – требования, а не реализация
- Применяем техники тест-дизайна
- Проверяем негативные сценарии

Вопросы
?