

# Тестирование API

Урок 1





## Яковлев Станислав

Team Lead QA в VK

- ✦ Руководил командой тестирования столичного оператора каршеринга.
- ✦ Декан факультета тестирования в GeekBrains.
- ✦ Автор Telegram-канала «Тестировщики нужны» — [\*\*@qa\\_chillout\*\*](#).



# Наш план



# План курса



Клиент-серверная  
архитектура



REST API



Soap API



## План курса



Клиент-серверная  
архитектура



REST API












Soap API



GraphQL, gRPC



## План урока

-  Клиент-серверная архитектура
-  Что такое Клиент
-  Что такое Сервер
-  Балансировщик
-  Виды клиент-серверной архитектуры
-  Принцип работы API
-  Типы API
-  Протоколы HTTP/HTTPS
-  Для чего необходим Swagger

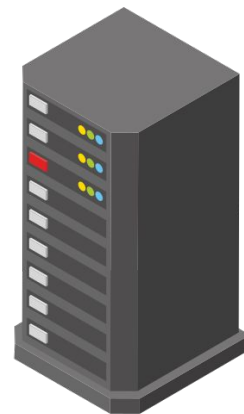
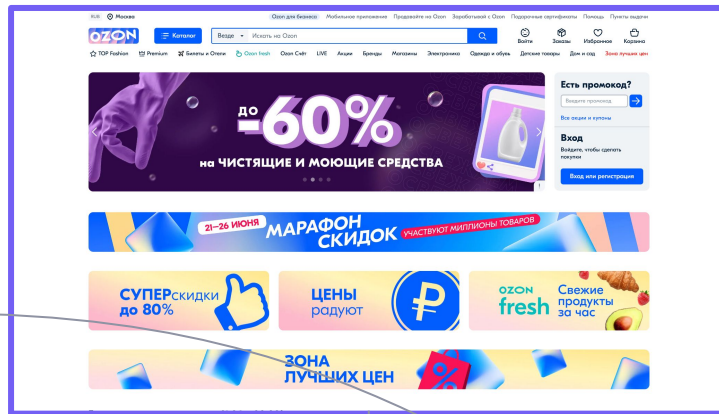


# Клиент-серверная архитектура



## Клиент-серверная архитектура

«Клиент — сервер» — вычислительная или сетевая архитектура, в которой сетевая нагрузка распределена между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами.





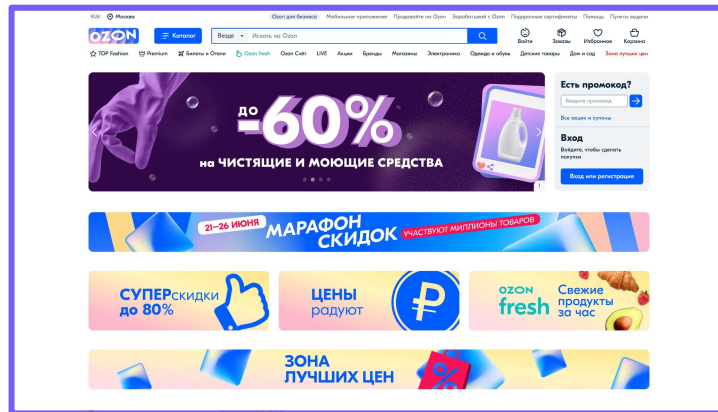


# Что такое Клиент

## Клиент

Это пользователь сервиса, который обращается к серверу для получения какой-то информации.

Клиент:  
*мобильное приложение*



Клиент:  
*веб-приложение*



# Что такое Сервер



# Сервер

Это место, где располагается веб-приложение или его серверная часть.





# Балансировщик



## Балансировщик

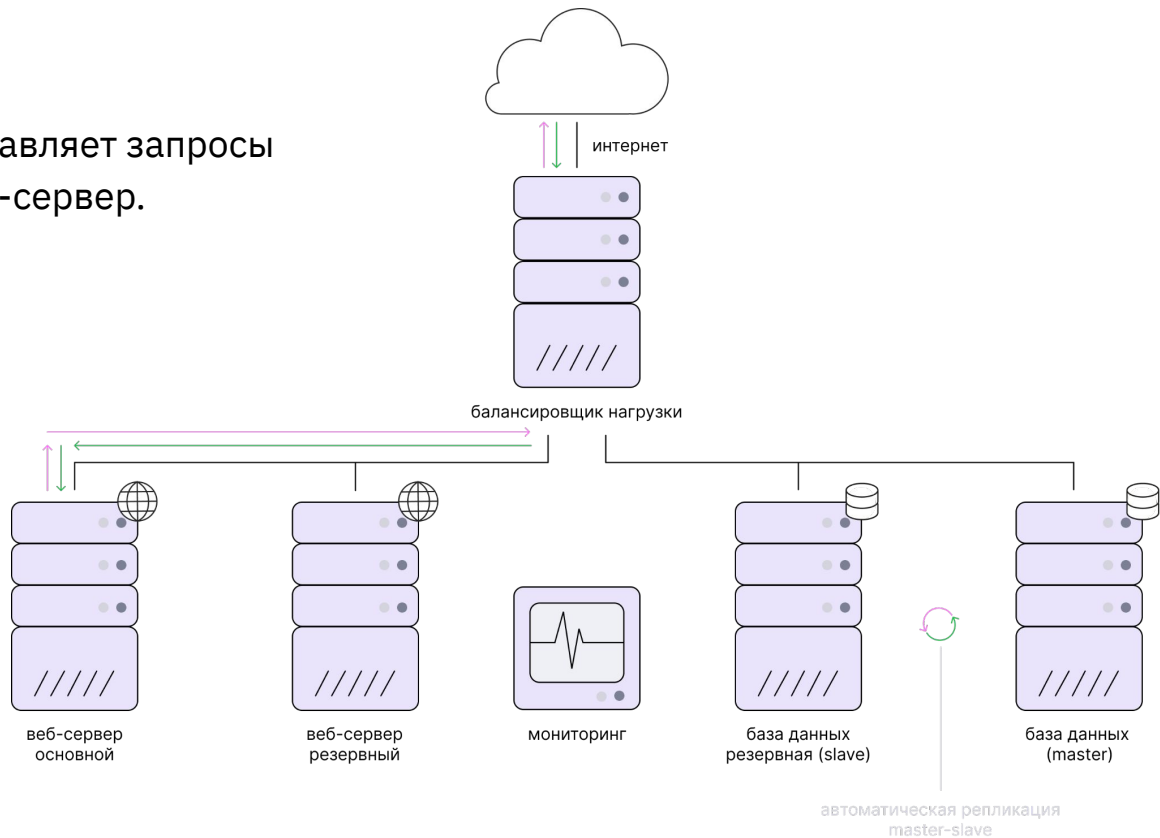
Балансировщик нагрузки (Load Balancer) — сервис, который помогает серверам эффективно перемещать данные, оптимизирует использование ресурсов доставки приложений и предотвращает перегрузки.

### Функции балансировщика:

- Разгрузка
- Прогнозная аналитика
- Запуск новых виртуальных хранилищ

## Принцип работы

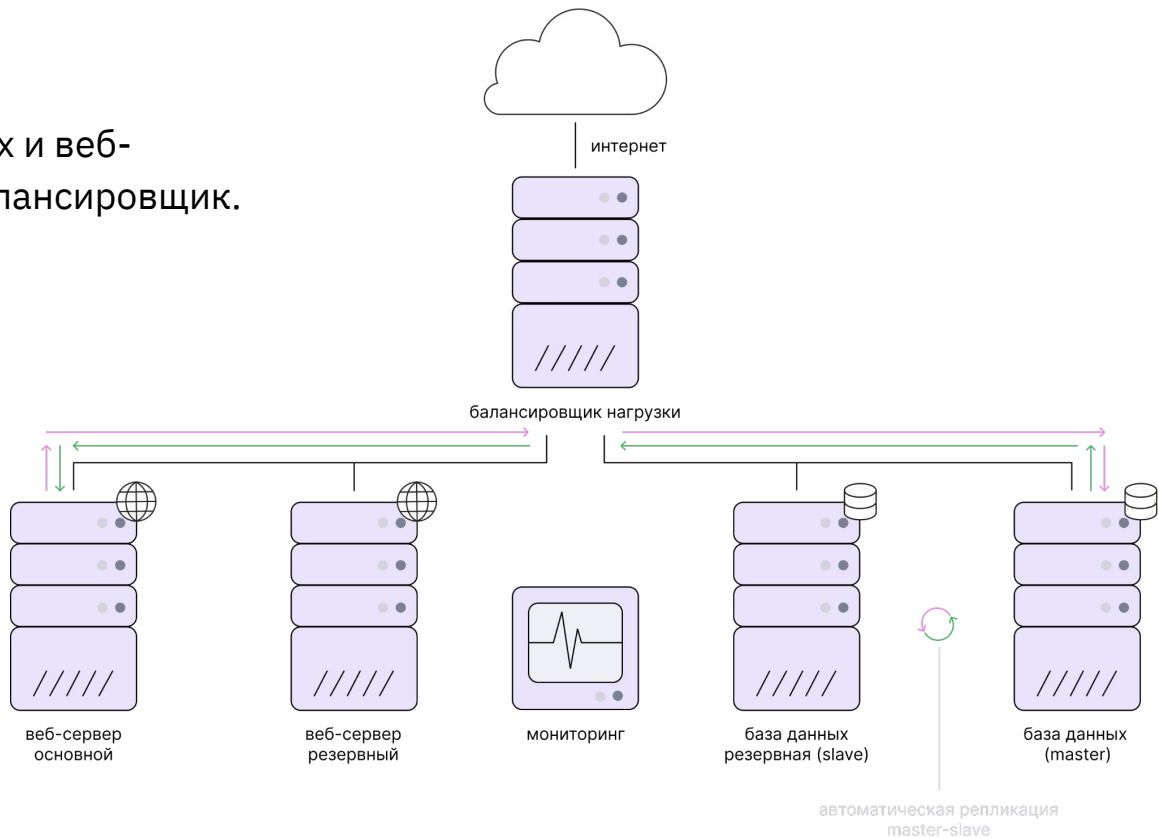
Обычно балансировщик направляет запросы посетителей на основной веб-сервер.





## Принцип работы

Общение между базой данных и веб-сервером тоже идёт через балансировщик.



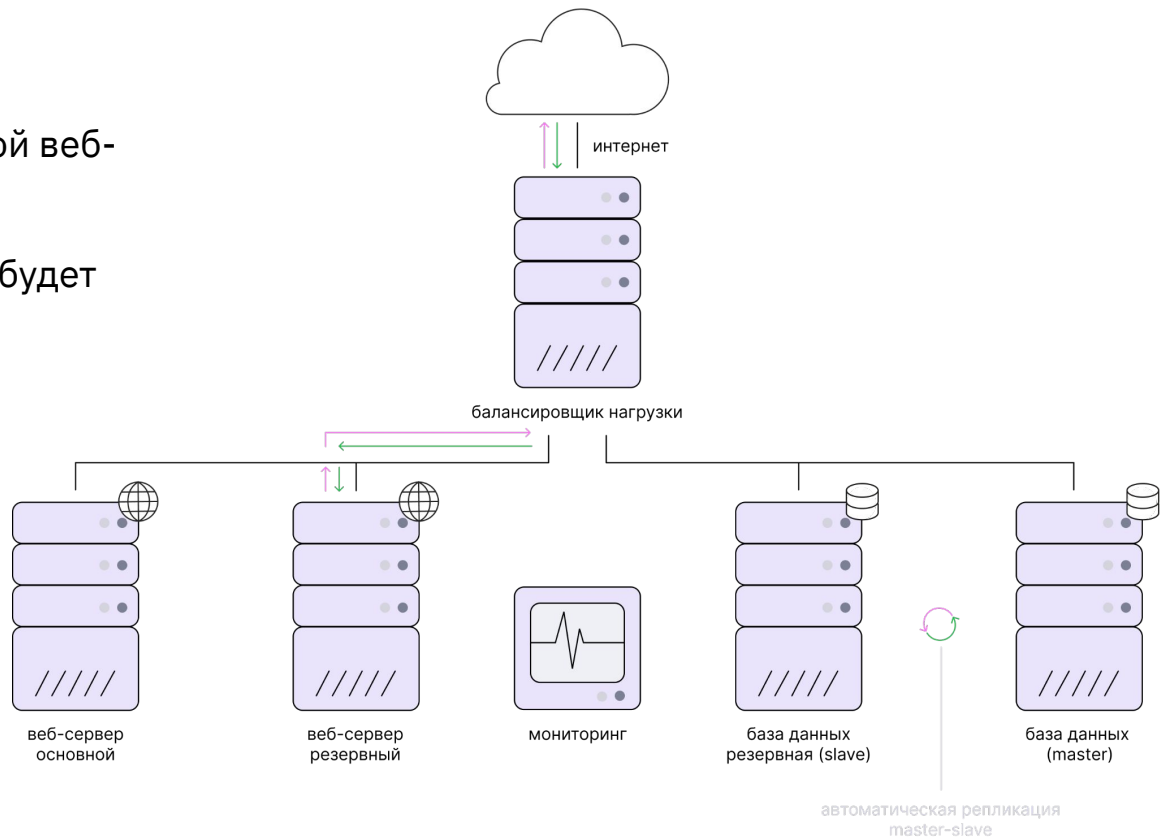




## Принцип работы

Что произойдёт, если основной веб-сервер выйдет из строя?

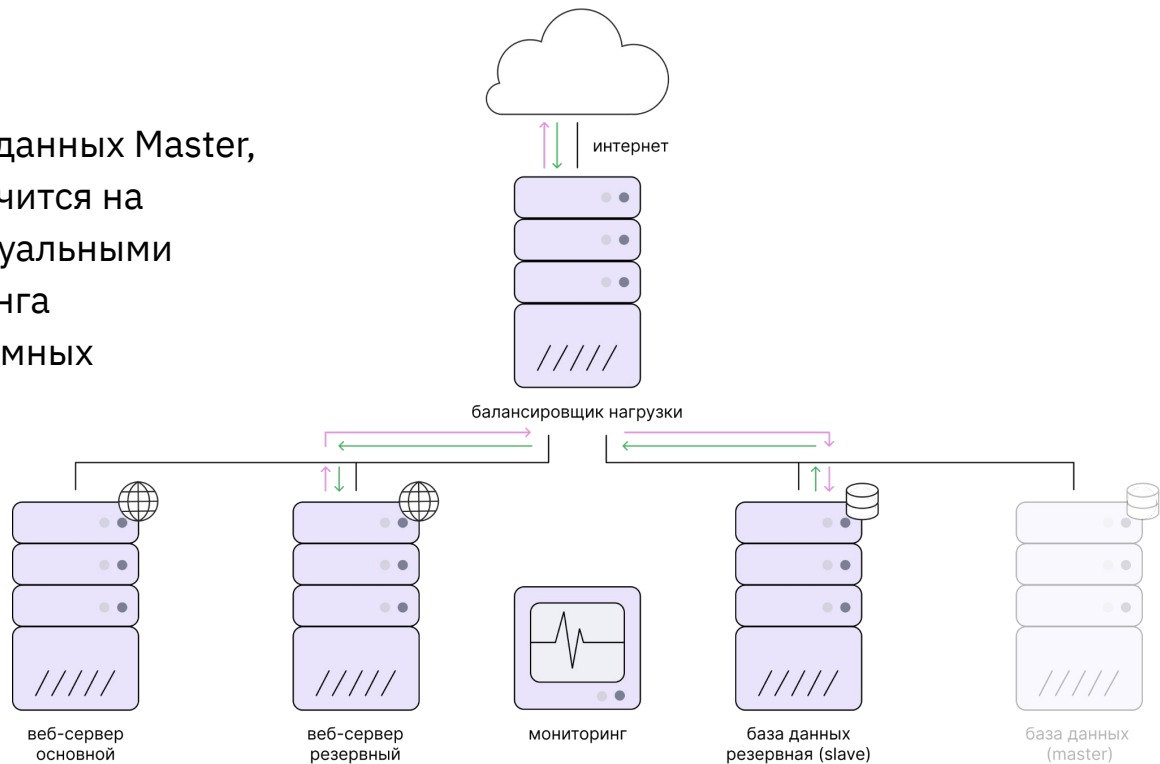
Балансировщик увидит это и будет перенаправлять все запросы на резервную копию.





## Принцип работы

Если откажет сервер с базой данных Master, балансировщик сам переключится на резервный сервер Slave с актуальными данными. А сервис мониторинга предупредит о событии системных администраторов.

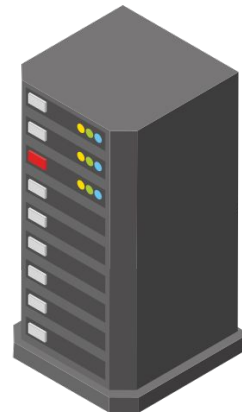
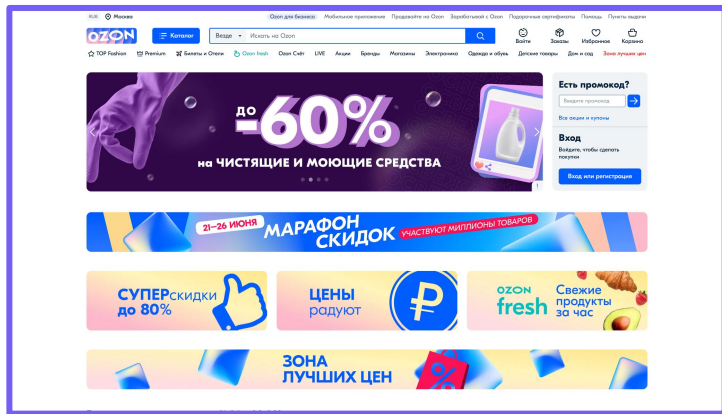




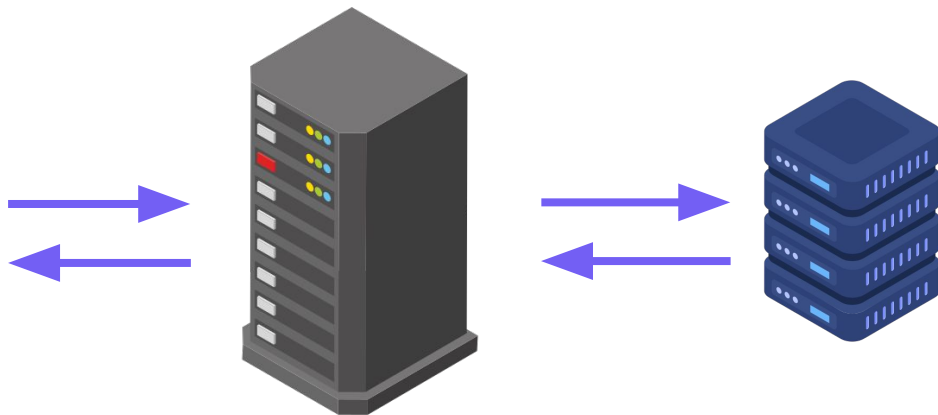
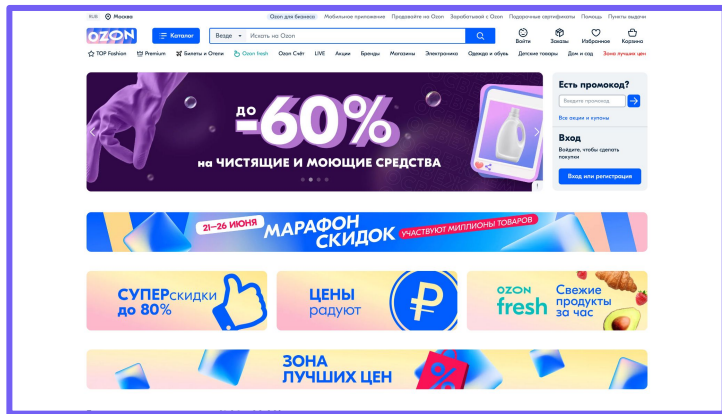
# Виды клиент-серверной архитектуры



## Двухзвенная (двухуровневая) архитектура

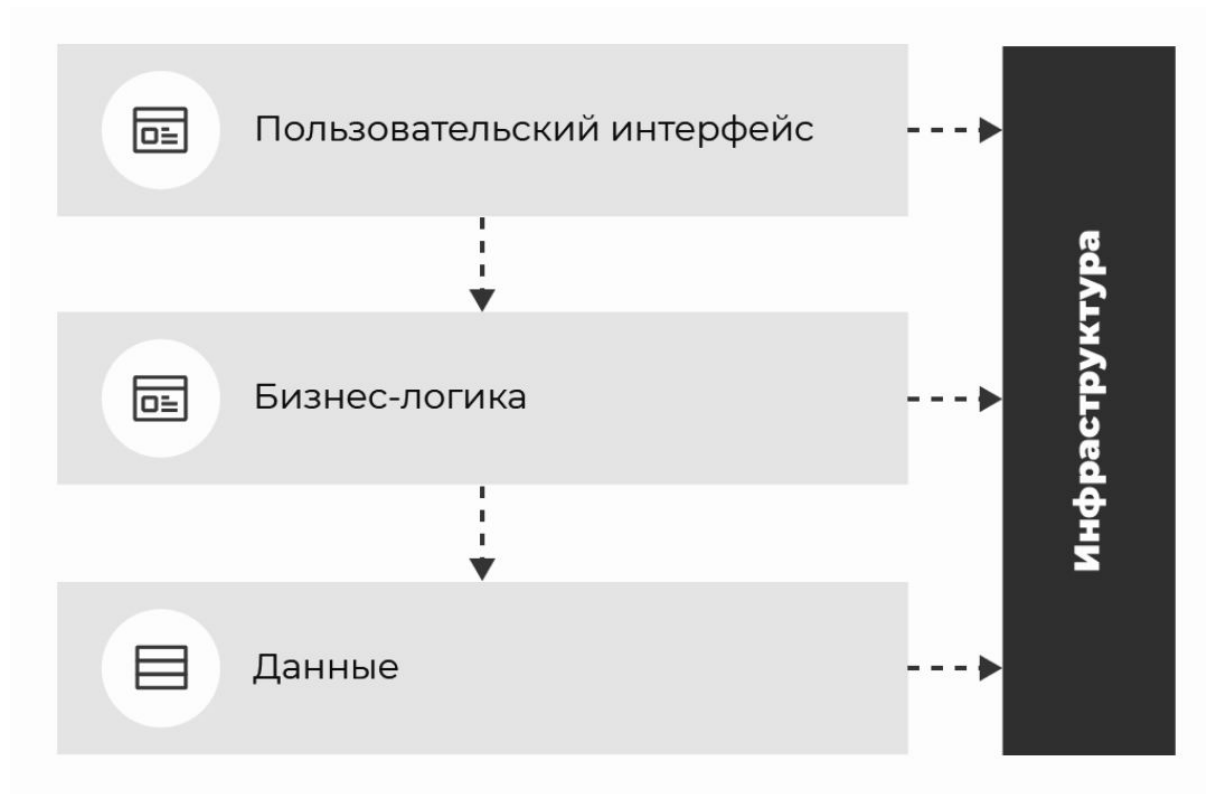


## Трёхзвенная (трёхуровневая) архитектура





## Многозвенная архитектура





## Монолитная архитектура

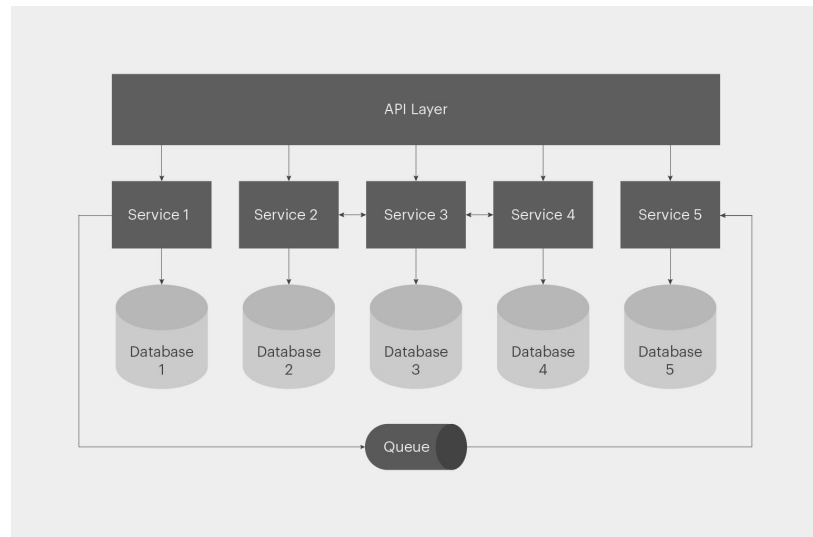
Монолитной называют архитектуру, где приложение представлено в виде единого компонента и представляет собой разрез модульно прошитой бизнес-логики. Все клиенты, как правило, обращаются только к одному приложению.





## Микросервисная архитектура

Это подход к созданию приложения, подразумевающий отказ от единой, монолитной структуры.





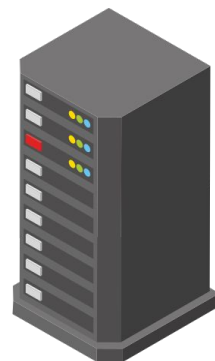
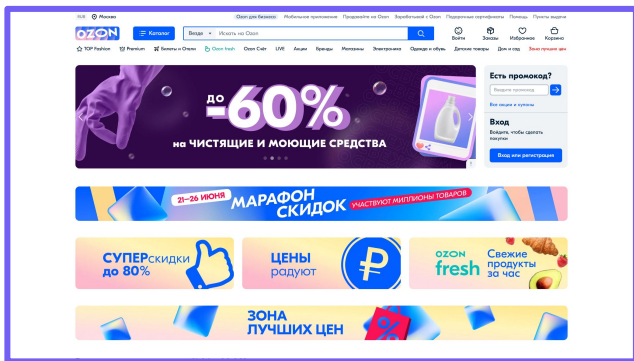


# Принцип работы API



## Принцип работы API

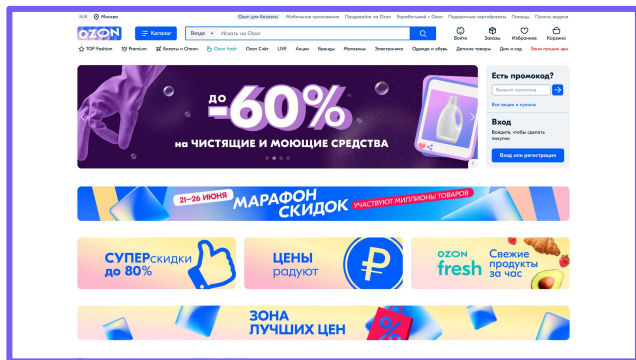
**API** — это контракт, предоставляемый программой. «Ко мне можно обращаться так и так, я обязуюсь делать то и это».



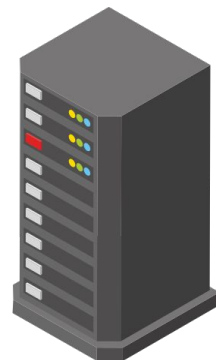
```
{  
  "iPhone" :  
  {  
    "discount" : "15%",  
    "memory" : "512",  
    "color" : "серый"  
  },  
  "price" : 200 000  
}
```



## Принцип работы API: позитивный сценарий

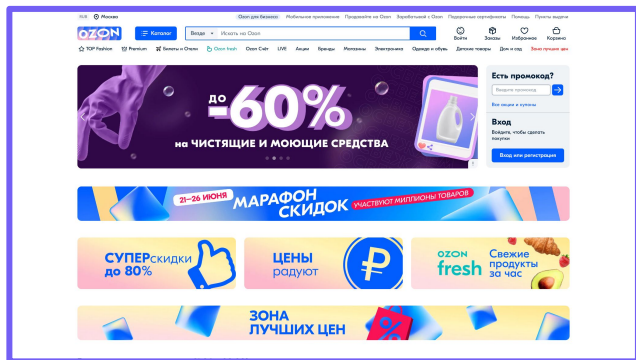


```
{  
  "message" : "Заказ сформирован успешно!"  
}
```

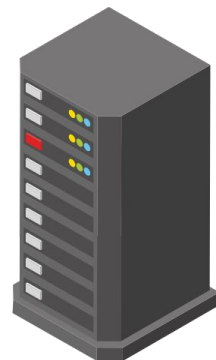




## Принцип работы API: негативный сценарий

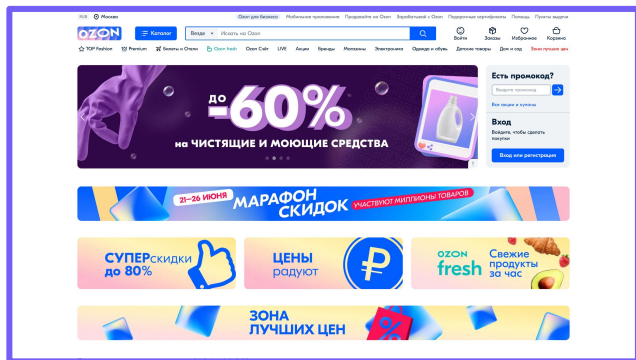


```
{  
  "iPhone" :  
  {  
    "discount" : "0",  
    "memory" : "утюг",  
    "color" : "512"  
  },  
  "price" : 0.200 000  
}
```

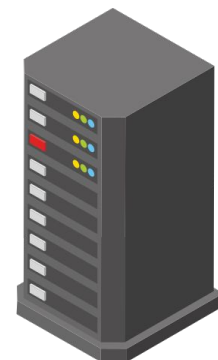




## Принцип работы API: негативный сценарий



```
{  
  "status" : 502  
}
```





## Принцип работы API

### Контракт включает в себя:

- саму операцию, которую мы можем выполнить;
- данные, которые поступают на вход;
- данные, которые оказываются на выходе (содержание данных или сообщение об ошибке).

### Преимущество подхода:

- такой подход позволяет большому количеству физических клиентских устройств и типов приложений взаимодействовать с данным приложением.

Один API можно использовать не только для вычислений на базе ПК, но также для мобильных телефонов.



# Типы API



# SOAP

**Simple Object Access Protocol (SOAP)** — простой протокол доступа к объектам.

```
<?xml version="1.0" encoding="utf-8" ?>
<SpellResult>
  <error code="1" pos="0" row="0" col="0" len="14">
    <word>синхрофазатрон</word>
    <s>синхрофазотрон</s>
  </error>
  <error code="3" pos="17" row="0" col="17" len="5">
    <word>дубне</word>
    <s>Дубне</s>
  </error>
</SpellResult>
```





# REST

**REST API (Representational State Transfer)** — передача состояния представления.

## Запрос:

GET <https://api.youla.io/api/v1/counters/622b65d14abcae3c015904be>

## Ответ:

```
{
  "data": {
    "id": "622b65d14abcae3c015904be",
    "chats": 0,
    "moderation": [],
    "archive": []
  },
  "status": 200,
  "detail": "Ok",
  "uri": ""
}
```



# GraphQL

Это язык запросов для получения данных через API.

## POST-запрос

```
{
  "extensions": {
    "persistedQuery": {
      "version": 1
    }
  },
  "id": "2332a58683ec9efc3c54a64489a7551d610cd1f7a3e485591bdf3fbf4a0c6cd9",
  "operationName": "Feed",
  "variables": {
    "after": "",
    "input": {
      "attributes": [{
        "slug": "categories",
        "value": []
      }],
      "datePublished": null,
      "exb": null,
      "location": {
        "city": null,
        "distanceMax": 50000,
        "latitude": 55.750629000000004,
        "longitude": 37.618665
      },
      "price": {
        "from": null,
        "to": null
      },
      "resultType": null,
      "search": null,
      "sort": "DEFAULT",
      "suggestedSubcategory": 0,
      "suggestedText": null,
      "viewType": "tile"
    },
    "skipAdvertisements": false,
    "skipCarousels": false,
    "skipStories": false
  }
}
```

## Ответ в формате GraphQL

```
{
  "data": {
    "feed": {
      "__typename": "Feed",
      "items": [],
      "pageInfo": {
        "__typename": "PageInfo",
        "threshold": 15,
        "hasNextPage": true,
        "cursor": "{\\\"page\\\":0,\\\"totalProductsCount\\\":30,\\\"dateUpdatedTo\\\":1647041341}",
        "personalSearchId": null,
        "productsAnalytics": {
          "__typename": "AllProductsAnalytics",
          "searchId": "5e3a601db28c310002e0911e31616fe5"
        }
      }
    },
    "extensions": {
      "tracing": {
        "version": 1,
        "startTime": "2022-03-11T23:29:05.731Z",
        "endTime": "2022-03-11T23:29:08.175Z",
        "duration": 2443633712,
        "execution": {
          "resolvers": []
        }
      }
    }
  }
}
```



## gRPC

Это актуальный фреймворк для разработки масштабируемых, современных и быстрых API. Часть RPC в названии расшифровывается как Remote Procedure Call и дословно переводится как «Система удалённого вызова процедур». Система впервые представлена корпорацией Google в 2015 году.

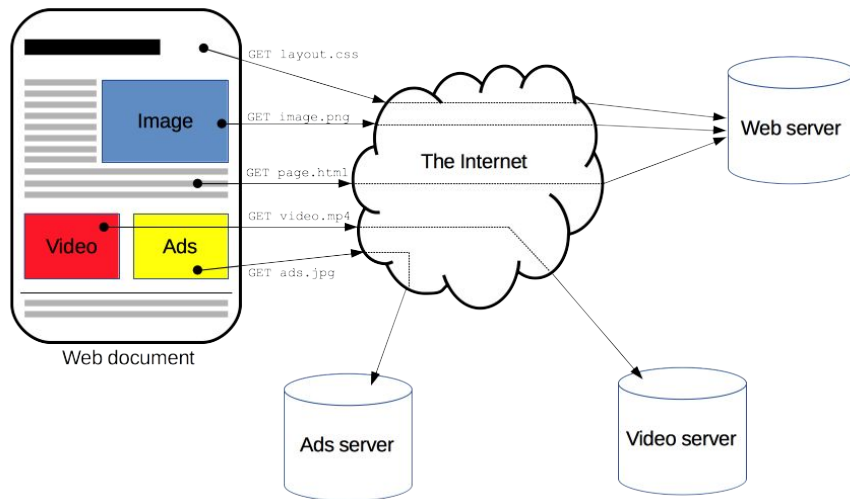


# Протоколы HTTP/HTTPS

## Протоколы HTTP/HTTPS

**HTTP** — это простой протокол, используемый для передачи содержимого в интернете. Подключение в HTTP устанавливается по стандартному TCP-порту 80.

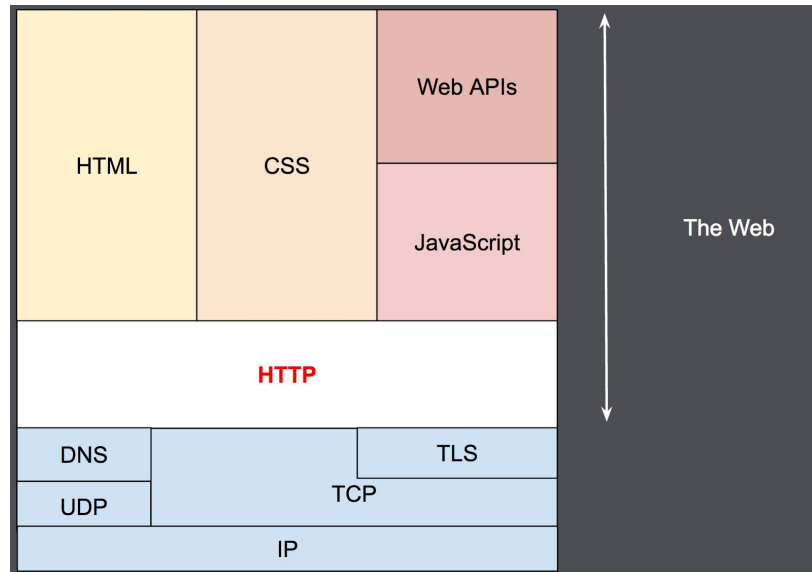
**HTTPS** — это безопасная версия протокола HTTP, которая реализует протокол HTTP с использованием протокола TLS для защиты базового TCP-подключения. Подключение в HTTPS устанавливается по TCP-порту 443.





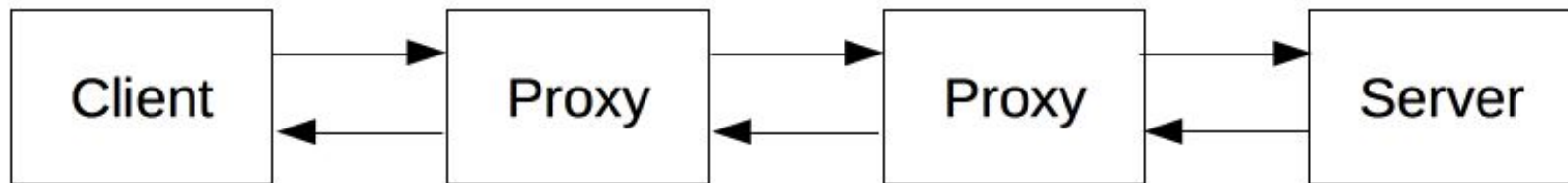
## Запросы и ответы

Сообщения, отправленные клиентом (обычно веб-браузером), называют запросами. А сообщения, отправленные сервером, — ответами.





## Составляющие систем на основе HTTP





## Основные аспекты HTTP

- HTTP — прост
- HTTP — расширяем
- HTTP не имеет состояния, но имеет сессию





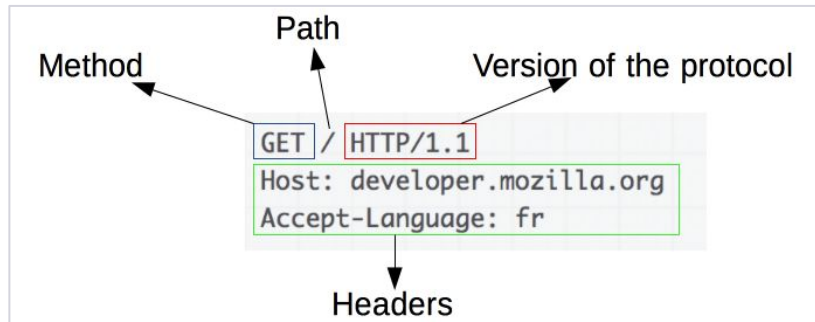
## HTTP-поток

1. Открытие TCP-соединения
2. Отправка HTTP-сообщения
3. Чтение ответа
4. Заккрытие или переиспользование соединение для дальнейших запросов

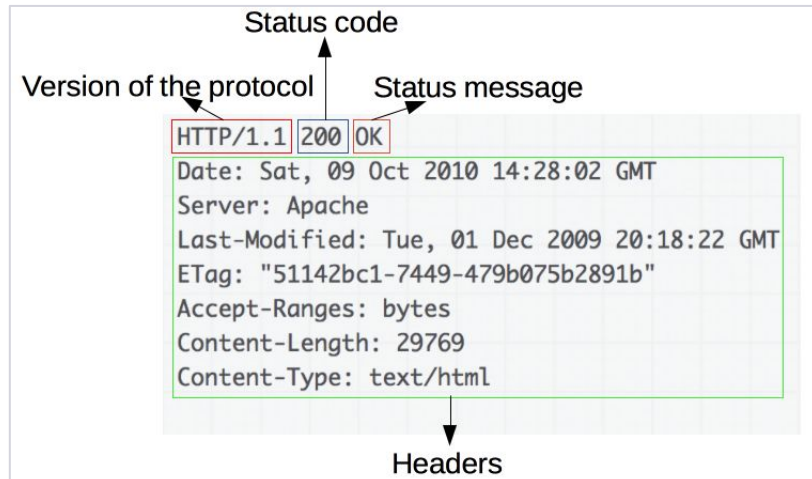


# HTTP-поток

## Запрос



## Ответ





# Для чего необходим Swagger



# Swagger

Это фреймворк для спецификации RESTful API.

## Swagger Petstore 1.0.6

| Base URL: [petstore.swagger.io/v2](https://petstore.swagger.io/v2) |  
<https://petstore.swagger.io/v2/swagger.json>

This is a sample server Petstore server. You can find out more about Swagger at <http://swagger.io> or on <irc://freenode.net, #swagger>. For this sample, you can use the api key `special-key` to test the authorization filters.

[Terms of service](#)  
[Contact the developer](#)  
[Apache 2.0](#)  
[Find out more about Swagger](#)

Schemes

HTTPS

Authorize

pet

Everything about your Pets

Find out more: <http://swagger.io>

POST	/pet/{petId}/uploadImage	uploads an image	✓	🔒
POST	/pet	Add a new pet to the store	✓	🔒
PUT	/pet	Update an existing pet	✓	🔒
GET	/pet/findByStatus	Finds Pets by status	✓	🔒
GET	/pet/findByTags	Finds Pets by tags	✓	🔒
GET	/pet/{petId}	Find pet by ID	✓	🔒
POST	/pet/{petId}	Updates a pet in the store with form data	✓	🔒
DELETE	/pet/{petId}	Deletes a pet	✓	🔒

<https://petstore.swagger.io>



Спасибо 😊  
за внимание