





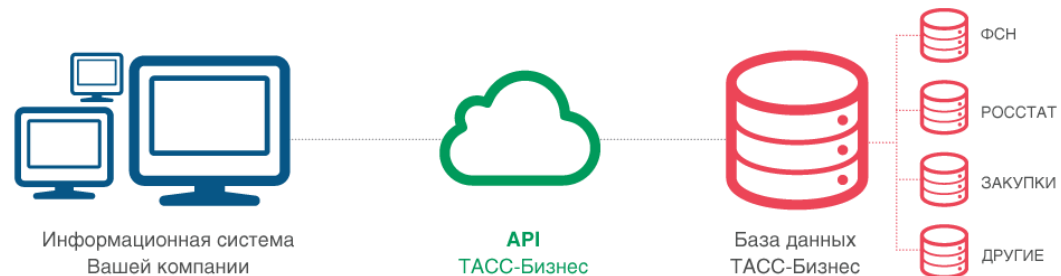
Web service

API

REST

API

Application Programming Interface



Это программный интерфейс, который позволяет двум приложениям взаимодействовать друг с другом без какого-либо вмешательства пользователя.

API предоставляют продукт или услугу для связи с другими продуктами и услугами, не зная, как они реализованы.



Главный принцип работы API.

Почему его называют интерфейсом



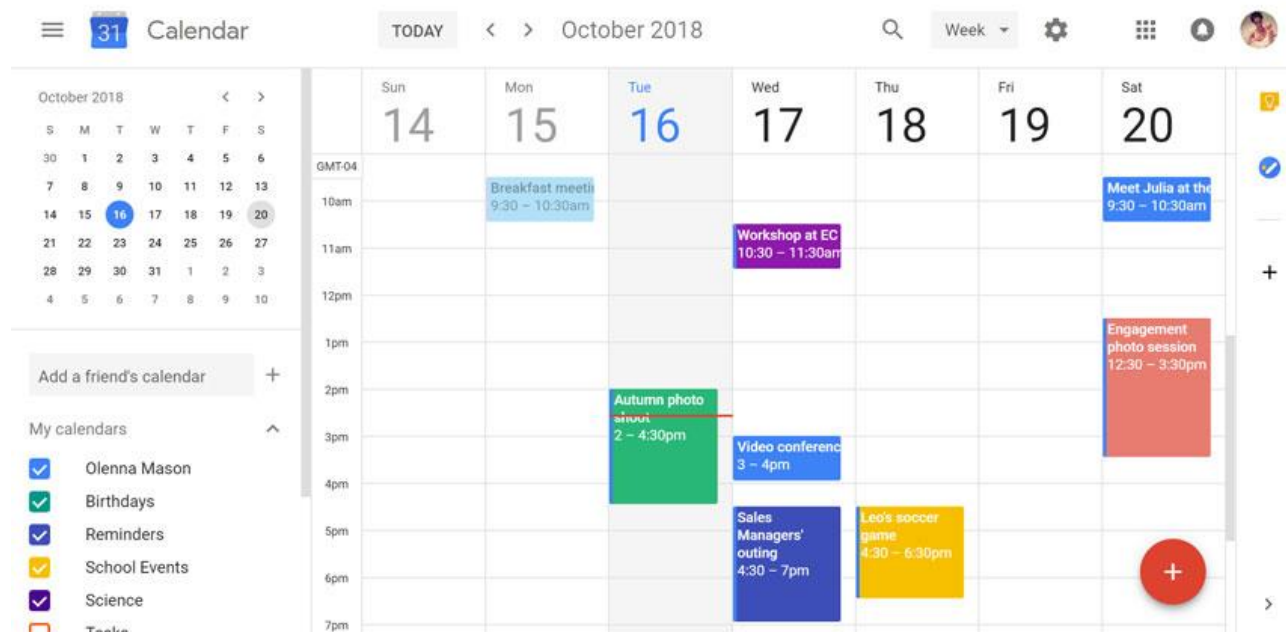
Инкапсуляция сокрытие части функций ради упрощения работы в целом и минимизации участков программного обеспечения, где один из разработчиков мог бы допустить ошибку.

Зачем нужен API?

Почему разработчики используют API?



Google Календарь




Погодное приложение





Сервис по заказу авиабилетов

РЕГИСТРАЦИЯ **ВОЙДИТЕ**
[Зарегистрироваться по Email](#)





 +7 Номер телефона

☒

ОТПРАВИТЬ SMS С КОДОМ

Мы отправим код подтверждения SMS-сообщением на номер телефона, который вы указали. Убедитесь, что правильно указали номер и код страны.

Быстрый доступ с

[Показать все](#)

WIFIKA.RU

Кнопки авторизации

Web-сервис (служба) – программа, которая организует взаимодействие между сайтами. Информация с одного портала передается на другой.



Задачи веб-сервисов



- Интеграция процессов идет сразу, без участия людей
- Если в компании используются корпоративные программы, то веб-сервис поможет настроить их совместную работу

Веб-сервис Web-API	API
Все веб-сервисы являются API.	Все API не являются веб-сервисами.
Он поддерживает XML.	Ответы форматируются с использованием MediaTypeFormatter Web API в XML, JSON или любой другой заданный формат.
Он может использоваться любым клиентом, который понимает XML.	Может использоваться клиентом, который понимает JSON или XML.
Веб-сервис использует три стиля: REST, SOAP и XML-RPC для связи.	API можно использовать для любого стиля общения.
Он обеспечивает поддержку только для протокола HTTP.	Он обеспечивает поддержку протокола HTTP / s: заголовки запроса / ответа URL и т. Д.



СПОСОБЫ РЕАЛИЗАЦИИ ВЕБ-СЕРВИСОВ

XML-RPC (XML Remote Procedure Call)

SOAP (Simple Object Access Protocol)

JSON-RPC (JSON Remote Procedure Call)

REST (Representational State Transfer)

Специализированные протоколы для конкретного вида задач,
такие как **GraphQL**.



REST



Representational

State

Transfer



**REST – это не протокол и не стандарт, а архитектурный
СТИЛЬ**





REST vs API



REST - это набор rules/standards/guidelines для создания веб-API. Поскольку существует множество способов сделать это, наличие согласованной системы структурирования API экономит время на принятии решений при ее создании и экономит время на понимании того, как ее использовать.

API-это очень широкий термин. Как правило, это то, как один фрагмент кода разговаривает с другим. В веб-разработке API часто относится к способу, которым мы получаем информацию из онлайн-сервиса.



РЕСУРС

URI

Ресурс — это ключевая абстракция, на которой концентрируется протокол HTTP. Ресурс — это все, что вы хотите показать внешнему миру через ваше приложение. Например, если мы пишем приложение для управления задачами, экземпляры ресурсов будут следующие:

Конкретный пользователь

- Конкретная задача
- Список задач

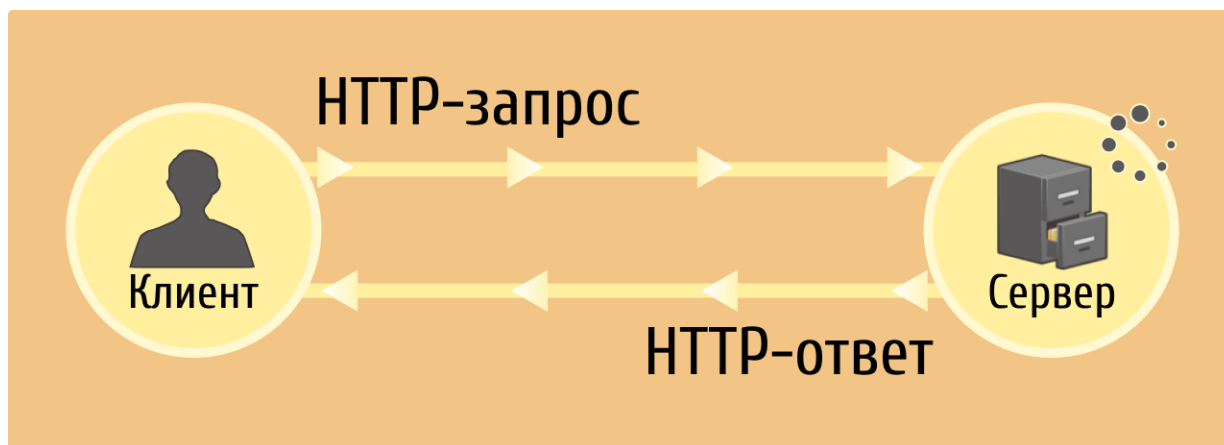
clients — клиенты;
orders — заказы клиентов;
items — товары.

/clients — URI всех имеющихся клиентов;

/clients/23 — URI конкретного клиента, а именно клиента с ID=23;

/clients/4/orders — URI всех заказов клиента №4;

/clients/1/orders/12 — URI заказа №12 клиента №1.



- **Формат обмена данными:** здесь нет никаких ограничений. JSON — очень популярный формат, хотя можно использовать и другие, такие как XML
- **Транспорт:** всегда HTTP. REST полностью построен на основе HTTP.
- **Определение сервиса:** не существует стандарта для этого, а REST является гибким.

МЕТОДЫ HTTP

Запрос	Описание
GET /clients/23 Accept : application/json, application/xml	Получить информацию о клиенте №23 в формате json или xml
POST /clients { "name" : "Amigo", "email" : "amigo@jr.com", "phone" : "+7 (191) 746-43-23" }	Создать нового клиента с полями: Имя — Amigo Email — amigo@jr.com Тел. — +7 (191) 746-43-23
PUT /clients/1 { "name" : "Ben", "email" : "bigben@jr.com", "phone" : "+380 (190) 346-42-13" }	Редактировать клиента №1 в следующем образом: Имя — Ben Email — bigben@jr.com Тел. — +380 (190) 346-42-13
DELETE /clients/12/orders/6	Удалить из системы заказ №6 у клиента №12

Принципы REST



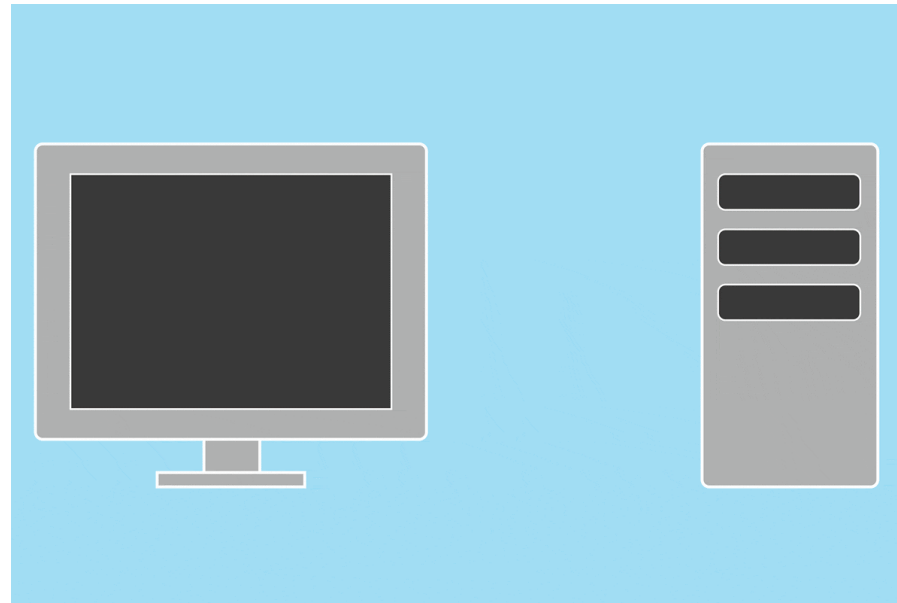
1. Client-Server
2. Stateless
3. Cacheable
4. Uniform interface
5. Layered system
6. Code on demand

1. Client-Server

2. Stateless
3. Cacheable
4. Uniform interface
5. Layered system
6. Code on demand

Приведение архитектуры к модели клиент-сервер

Необходимо отделять потребности клиентского интерфейса от потребностей сервера, хранящего данные. Само разграничение на “клиент” и “сервер” позволяет им развиваться независимо друг от друга.



1. Client-Server

2. Stateless

3. Cacheable

4. Uniform interface

5. Layered system

6. Code on demand



Отсутствие состояния

Все запросы от клиента должны быть составлены так, чтобы сервер получил всю необходимую информацию для выполнения запроса. Таким образом и сервер, и клиент могут "понимать" любое принятое сообщение, не опираясь при этом на предыдущие сообщения.



Кэширование

Клиенты могут выполнять кэширование ответов сервера. У тех, в свою очередь, должно быть явное или неявное обозначение как кэшируемых или некэшируемых, чтобы клиенты в ответ на последующие запросы не получали устаревшие или неверные данные.

1. Client-Server
2. Stateless

3. Cacheable

4. Uniform interface
5. Layered system
6. Code on demand



1. Client-Server
2. Stateless
3. Cacheable

4. Uniform interface

5. Layered system
6. Code on demand

Единообразие интерфейса

Клиент должен всегда понимать, в каком формате и на какие адреса ему нужно слать запрос, а сервер, в свою очередь, также должен понимать, в каком формате ему следует отвечать на запросы клиента. Этот единый формат клиент-серверного взаимодействия, который описывает, что, куда, в каком виде и как отсылать и является унифицированным интерфейсом



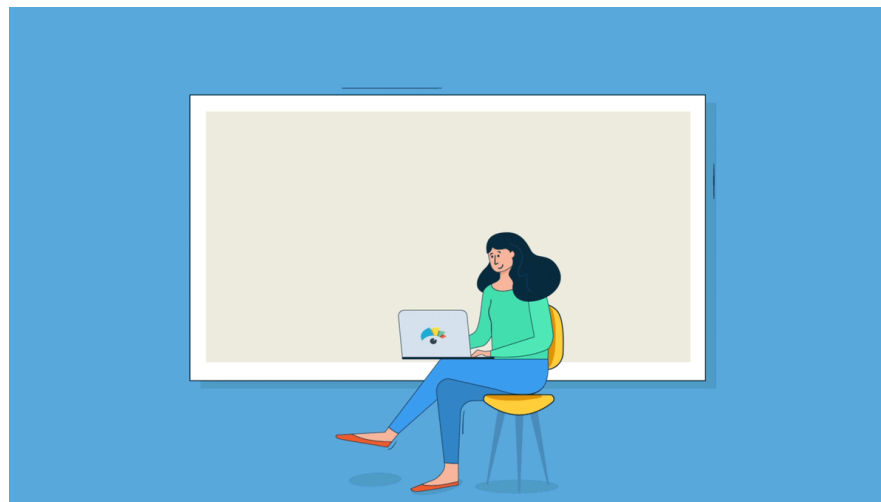
1. Client-Server
2. Stateless
3. Cacheable
4. Uniform interface

5. Layered system

6. Code on demand

Слои

Под слоями подразумевается иерархическая структура сетей. Иногда клиент может общаться напрямую с сервером, а иногда — просто с промежуточным узлом. Применение промежуточных серверов способно повысить масштабируемость за счёт балансировки нагрузки и распределённого кэширования.

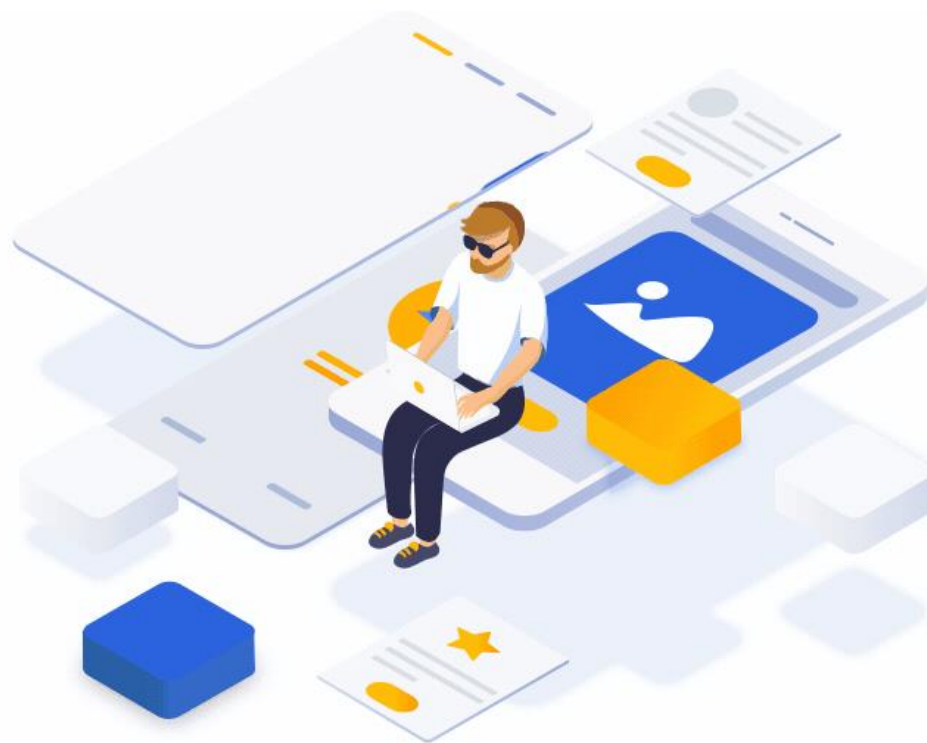


Код по требованию (необязательное ограничение)

Данное ограничение подразумевает, что клиент может расширять свою функциональность, за счет загрузки кода с сервера в виде апплетов или сценариев, скриптов

1. Client-Server
2. Stateless
3. Cacheable
4. Uniform interface
5. Layered system

6. Code on demand



RESTful

Всего лишь означает сервис, реализованный с использованием принципов REST.

ВОПРОСЫ

