**API** - is an interface to some data stored somewhere remotely on a different server. Because API- s are designed to be used by programs they dont have a friendly interface like a website or some app on the phone. API-s can only be used if you write some programming code, or use a terminal to write some complex command. When we are working with API we are working with JSON to transfer the data

**A web API** - is a set of rules and protocols that define how two systems can communicate with each other over the internet. It allows one system (such as a server) to expose (выставлять-передовать) certain functionality or data to other systems (such as mobile application). Web APIs are typically accessed over HTTP or HTTPS, and they can use a variety of data formats (such as JSON or XML) to exchange information. They are often used to enable integrations between different systems, or to allow developers to build applications that interact with a web-based service or platform.

к примеру есть клиент он обращается на сервер за какойто инф., сервер её оддаёт по запросу - это идеальный вариант НО, так на самом деле работать не будет так как , к примеру фронтэнд (клиент) у нас написан на JavaScript а backEnd на Java, данные приложения просто не смогут взаимодействовать. ВОТ для этого и существует API - это технология благодаря которой одно приложение может общаться/взаимодействовать с другим, особенно это актуально когда одно приложение написано на одном стэке технологий а другое на другой технологии. Или ещо пример, мобильная версия приложения обращается на сервер, опять же 2-е разные технологии они напрямую работать не будут, их надо подружить как то и здесь опять же API. или у нас есть приложение и мы хотим использовать это приложение В другом приложении как к примеру какуюто ф-цию (соединить), но при этом оба приложения написаны разными командами и на разных стэках, их можно подружить с помощью API. То есть API - это как внутренний интерфейс приложения для общения но не с конечным пользователем, а в роли пользователя выступает стороннее приложение, и даже возможно наличие у одного приложения несколько API для разных возможных интеграций с разными стэками технологий.

API - это описание способов, которыми одна компьютерная программа может взаимодействовать с другой компьютерной программой.

REST API - (Representational State Transfer) архитектурный стиль взаимодействия компонентов распределённого

приложения в сети. Т.Е. это не какая то технология это просто подход, это архитектурный стиль который диктует определённые правила, чтобы одни программы могли по определённым правилам общаться с другими программами, Опять же - это набор правил с помощью которых мы можем выстроить это взаимодействие.

ТРЕБОВАНИЯ К АРХИТЕКТУРЕ REST

**1**) - Клиент - серверное взаимодействие. Важно понимать, что сервер сам по себе никогда никакие запросы посылать никуда не будет. Ключивой момент - сервер отправляет ответ только когда клиент делает запрос. Пример: просматриваем страницу в youTube там определённое количество “лайков” оно не меняется динамически, чтоб увидеть изменения необходимо обновить страницу, а обновление это и есть запрос, по которому и придёт ответ - таже страница с обновлёнными данными (в том числе и с другим числом лайков).

**2**) - Операции над ресурсами. Ресурсами выступает например у нас есть интернет магазин и продукты всевозможные в нём: телефоны, телевизоры и т.д. это и есть ресурсы и мы можем их покупать, ложить в корзину, бронировать, совершать над ними какието действия эти действия и есть операции . Ещо к примеру YouTube, видео на этом сайте это и есть ресурсы которыми можно пользоватся - просматривать , а это и есть операции..

3) - Передача данных с помощью HTTP. HTTP - это основа любого API,

4) - Отсутствие состояния (STATELESS). Например я пользуюсь YouTube и просматривая контент сервер предлагает мне схожие тематически видео, вот этот момент и есть состояние (т.е. мои интересы это и есть состояние), но сам сервер не хранит информацию о клиентских предпочтениях (состоянии), это очень много инф., вместо этого состояние хранится на машине клиента, это называется COOCKIES, но сам принцип видимо по отношению к серверу называется STATELESS

5) - Форматы передачи данных:JSON, XML

**HTTP**

**любой текст на веб-страничке заключен в специальные теги, которые указывают браузеру какой размер текста использовать, его цвет, расположение на странице (слева, справа или по центру). Это касается не только текста, но и картинок, форм, активных элементов и вообще всего контента, т.е. того, что есть на страничке. Браузер, обнаруживая теги, действует согласно их предписанию, и показывает Вам обработанные данные, которые заключены в эти теги.**

Протокол прикладного уровня передачи данных. Говоря о клиент сервеной архитектуре, о запросах и ответах сервера, дак вот эти запросы и ответы это HTTP-запросы HTTP-ответы.

Благодаря протоколу HTTP Ваш браузер, получая данные с сервера, знает, как их требуется обработать, и успешно обрабатывает их, показывая Вам запрашиваемую страничку.

**Из чего состоят HTTP-запросы:**

1. Стартовая строка (request line).

а) Методы HTTP протокола:

**get**- получить данные из сервера,

**put**- изменение всей сущности, без её удаления

**post**- создание какой то новой сущности

**delete**- удаление существующей сущности,

**patch**-изменение/модернизация части сущности не всей

**Head**-не имеет response or request body, в основном используется для проверки, что endpoint is valid, при этом не ставится цель получения какой-либо инф., просто так сказать лёгкая проверка без нагромождения всякой логикой, чисто посмотреть что endpoint работает.

**!!!** APIs implement request methods differently. While one API may use PUT/PATCH for updating a resource, another API might use POST. For this reason, it is best to check the API documentation first.**!!!**

б)Цель / путь запроса (как правило URL - Uniform Recorse Locator) пример пути запроса:



Resource - это например товары в интернет магазине или видео на YouTube.

Path parameter - это детализация нашего resource

Query Parameters - задают необходимую фильтрацию, указываются (перечисляются) после вопросительного знака

в) версия HTTP

1. Заголовки (Headers) умное определение это метаданные. Другимим словами это информация о информации т.е. для корректного ответа нам надо сказать в каком формате мы ожидаем информацию + идёт доп инф. про клиента, т.е. те же кукис и пр.. т.е. чисто техническая инф между клиентом и сервером. Hedears Бывают:

| a)- General Headers | b)- Request headers, | c)- Response headers. |
| --- | --- | --- |
|  |  |  |

1. Тело (Body), обязательно присутствует когда мы используем методы REQUST-а put и post потому, что так как данные методы Request-а что-то создают или меняют(обновляют) нам же надо где то эту информацию “принести” на сервер чтоб он понял не только что мы от него хотим чтоб он сделал но и дали ему материал для этой работы. Нельзя же сказать создай нового пользователя и на этом точка. Надо дать ему имя пользователя, пол, возраст и т. д., т.е. данные. Для этого и существует body в request-е. И в ответе к стати тоже, я думаю.

**Из чего состоят HTTP- RESPONSE:**

1. Строка статуса (Status line)
2. Версия протокола (обычно HTTP /1.1)
3. Код статуса (Status code)

\*1XX - информационные

\*2ХХ - Успешная операция

\*3ХХ - Перенапрвление

\*4ХХ - Клиентские ошибки

\*5ХХ - Серверные ошибки

с) Пояснения (Status text)

2) Заголовки (Headears)

3) Тело (Body)

**ТИПЫ ДАННЫХ:**

Имеется ввиду типы данных которые могут передаваться по HTTP: XML; JSON (javascript object notation)

