**API = Application Programming Interface**

Простым языком - это интерфейс, который обеспечивает взаимодействие одной системы с другой.

API – это механизмы, которые позволяют двум программным компонентам взаимодействовать друг с другом, используя набор определений и протоколов. Например, система ПО метеослужбы содержит ежедневные данные о погоде. Приложение погоды на телефоне «общается» с этой системой через API и показывает ежедневные обновления погоды на телефоне.

Это набор методов, которые можно использовать для доступа к функц-сти других ПО. например интеграция с платежной системой

Например **оплата товара в онлайн-магазине. Покупка авиабилета онлайн**

**Типы API**

1. **Local API (Локальный)**

Позволяет компонентам одной системы взаимодействовать между собой.

1. **Remote API (Удаленный)**

Позволяет связывать между собой несколько систем (сервисов)

* ***SOAP ПРОТОКОЛ!!!!!***
* ***REST протокол!!!* Representational state transport - позволяет записывать информацию в более удобном виде ( чем XML).**

**REST API** - архитектурный стиль, который обеспечивает общение клента с сервером с помощью HTTP запросов (get, post ..). Использование методов HTTP позволяет реализовать типичный CRUD для любой информации. Самый популярный формат сообщений - JSON.

Это набор правил того, как программисту организовать написание кода серверного приложения, чтобы все системы легко обменивались данными и приложение можно было масштабировать.

**это когда серверное приложение дает доступ к своим данным клиентскому приложению по определенному юрл**

REST API — это способ взаимодействия сайтов и веб-приложений с сервером. Его также называют RESTful. API (Application Programming Interface) — это код, который позволяет двум приложениям обмениваться данными с сервера. REST (Representational State Transfer) — это способ создания API с помощью протокола HTTP.

***Существует шесть обязательных ограничений для построения распределённых REST-приложений.***

Эти ограничения определяют работу сервера в том, как он может обрабатывать и отвечать на запросы клиентов.

**1) Модель клиент-сервер**

Отделение потребности интерфейса клиента от потребностей сервера повышает переносимость кода клиентского интерфейса на другие платформы, а упрощение серверной части улучшает масштабируемость.

**2) Отсутствие состояния**

В период между запросами клиента никакая информация о состоянии клиента на сервере не хранится. Все запросы от клиента должны быть составлены так, чтобы сервер получил всю необходимую информацию для выполнения запроса. Состояние сессии при этом сохраняется на стороне клиента.

**3) Кэширование**

Клиенты и промежуточные узлы могут выполнять кэширование ответов сервера. Ответы сервера должны иметь явное или неявное обозначение как кэшируемые или некэшируемые с целью предотвращения получения клиентами устаревших или неверных данных в ответ на последующие запросы.

**4) Единообразие интерфейса**

Унифицированные интерфейсы позволяют каждому из сервисов развиваться независимо.

К унифицированным интерфейсам предъявляются четыре ограничительных условия:

- Идентификация ресурсов

- Манипуляция ресурсами через представление

- «Самоописываемые» сообщения

- Гипермедиа, как средство изменения состояния приложения

**5) Слои**

Клиент обычно не способен точно определить взаимодействует ли он напрямую с сервером, или же с промежуточным узлом в связи иерархической структурой сетей (слои).

**6) Код по требованию (необязательное ограничение)**

REST может позволить расширить функциональность клиента за счёт загрузки кода с сервера в виде апплетов или сценариев.

**API всегда появляется раньше GUI!!!**

**Как работает API:**

1. **Вызов самой операции (метод GET, POST)**
2. **Входные данные (запрос) = HTTP Request**
3. **Выходные данные = HTTP Response**

**Как вызывать API:**

1. **Вызов функции системой (прямой метод)**

**= локальный API, в котором компоненты как-то общаются между собой с помощью этого интерфейса. Не тестируем (на программном уровне)**

1. **Вызов метода другой системы (прямой метод).**

**Какая-то система обращается к другой системе. Сайт как агрегатор туристических путевок который перенаправляет на оплату путевки например**

1. **Вызов метода человеком (прямой метод)**

**с помощью Postman/Swagger где мы сам вносим данные для запроса**

1. **API > GUI (косвенный метод)**