**ОБЩЕЕ**

API – набор правил, методов и протоколов, предоставленной программой для взаимодействия с ней

**Используется:**

* Для взаимодействия с внешними сервисами
* Для расширения функциональности
* Для интеграции с аппаратным обеспечением
* Для получения и обмена данными

Типы API:

* WEB API: доступ к функциональности и данными осуществляется через интернет с использованием стандартных протоколов, например, HTTP
* Библиотеки и SDK (инструменты, которые разработчики внедряют в свое приложение и которые используются для работы с определенными языками программирования)
* API:
  + - Операционных систем
    - БД
    - Социальных сетей
    - Файловые
    - Микросервисные
    - Мобильные

**Составляющие API:**

* Операции;
* входящие данные;
* исходящие данные

**Использование API:**

* прямое (система внутри себя; взаимодействие двух систем)
* косвенно (через GUI)

Контракт – описание методов, список действий и операций, используемых для данного вида API.

**2 наиболее распространенные типа API:**

* REST
* SOAP

**REST (контракт WADL) – Representational state transfer**

REST — это архитектурный стиль; подход, ориентированный на использование HTTP в качестве транспортного протокола. Использует стандартные HTTP-методы для взаимодействия (GET, POST, PUT, DELETE и другие), что способствует единообразию интерфейса.

Наиболее распространенный тип API в клиент-серверной архитектуре. Обычно возвращает данные в формате JSON, хотя может поддерживать и другие форматы.

**Принципы REST:**

* Клиент-серверная архитектура
* Отсутствие состояния (stateless)
* Кеширование
* Единообразие интерфейса

**Использование:**

App with CRUD operations: creating, reading, updating, deleting data.

Apps for Real-time communications.

*Плюсы:*

* Проще в использовании: тесно связан с HTTP и имеет более простую структуру сообщений. Поддерживает несколько форматов данных
* Быстрее, доступно кэширование данных
* Легче интегрируется за счет более простой архитектуры

*Минусы:*

* Неэффективен при работе с большими объемами данных: для получения всех необходимых данных может требоваться несколько запросов
* Использует стандартный механизм безопасности (HTTPS), не поддерживает сценарии с высокими требованиями к безопасности
* Не поддерживает встроенные механизмы транзакций (транзакция - последовательность операций над данными имеющая начало и конец)

**SOAP (контракт WSDL Web Services Description Language) – Simple Object Access Protocol**

SOAP — это формат обмена структурированными данными между приложениями. Использует XML для кодирования запросов и ответов.

Обладает высокой расширяемостью, позволяет осуществлять обмен данными через несколько различных транспортных протоколов, включая HTTP, SMTP и другие.

Имеет встроенные механизмы обработки ошибок.

**Принципы SOAP:**

* Строгая спецификация
* Расширенные возможности безопасности
* Поддержка сложных операций
* Независимость от транспортного протокола

**Использование:**

Для приложений с требованиями высокого уровня безопасности и надежного метода обмена данными между приложениями, для фичей с электронной подписью и шифрованием данных.

*Плюсы:*

* Больше подходит для систем с требованиями высокого уровня безопасности и шифрования данных, включает в себя WS-Security
* Поддерживает сложные транзакционные процессы
* Лучше совместим с корпоративными стандартами и сложными процессами

*Минусы:*

* Сложнее в использовании из-за строгой спецификации, поддерживает только один формат данных
* Запросы «тяжелые» из-за структуры формата XML
* Сложнее интегрировать и масштабировать (особенно для систем с высокой нагрузкой)

XSD схема — это документ, который определяет структуру и правила валидации для XML документов.

Файл, содержащий XML Schema, обычно имеет расширение «.xsd» (XML Schema definition).

Необходим для определения правил, которым будет подчиняться XML-документ.

На основании XML Schema можно создать модель документа, которая включает:

* словарь (названия элементов и атрибутов);
* модель отношений между элементами и атрибутами и их структуру;
* типы данных элементов и атрибутов.

Упрощенно говоря, XML - это данные. Схема - это описание данных, формат данных.

WSDL описывает вашу службу и ее действия - что такое вызов службы, какие методы он предлагает, какие параметры и возвращаемые значения имеют эти методы. Это описание поведения службы - функциональность.

XSD (Xml Schema Definition) описывает статическую структуру сложных типов данных, обмениваемых этими сервисными методами. Он описывает типы, их поля, любое ограничение на эти поля (например, максимальную длину или шаблон регулярного выражения) и т.д. Это описание типов данных и, таким образом, статических свойств службы - это о данных.

**JSON:**

Текстовый формат обмена данными (основан на JS, но может быть использован и при работе с другими языками программирования).

Используется только в REST API, в SOAP не используется.

Что может быть использовано в качестве значений:

* JSON-объект {в фигурных скобках}
* JSON-массив [в квадратных скобках]
* Числа
* Строки
* Литералы true/false, null

JSON-объект – неупорядоченный набор множества пар ключ:значение.

Неупорядоченный – нет определенного порядка, в котором должны передаваться пары, один из пунктов тестирования API – это проверить, что пара принимается независимо от расположения в объекте.

Ключ – параметр, значение – значение параметра.

Значение может быть:

* Число
* Строка
* Массив
* Другой объект

JSON весит меньше, его структура проще для чтения как машиной, так и человеком.

JSON –массив – упорядоченная коллекция значений (т.к. нет ключа, то обратиться к какому-либо значению можно по его номеру в массиве, следовательно положение значения в массиве меняться не должно).

Значения в массиве разделяются запятыми.

Правила well formed JSON, полезные для тестировщика:

1. Данные написаны в виде пар «ключ:значение»
2. Данные разделены запятыми
3. Объект находится внутри фигурных скобок {}
4. Массив — внутри квадратных []

**XML**

В XML каждый элемент должен быть заключен в теги. Тег — это некий текст, обернутый в угловые скобки.

Теги открывающие и закрывающие.

В любом XML-документе есть корневой элемент. Это тег, с которого документ начинается, и которым заканчивается.

Значение элемента хранится между открывающим и закрывающим тегами. Это может быть число, строка, или даже вложенные теги.

XML-элементы могут иметь атрибуты. Атрибуты предоставляют дополнительную информацию об элементе. Атрибуты должны быть заключены в кавычки.

UTF-8 — кодировка XML документов по умолчанию.

Правила well formed XML:

* Есть корневой элемент.
* У каждого элемента есть закрывающийся тег.
* Теги регистрозависимы!
* Соблюдается правильная вложенность элементов.
* Атрибуты оформлены в кавычках.

Лекции, используемые для подготовки информации:

<https://www.youtube.com/watch?v=2mqN7ZhDsUA>

<https://qalearning.net/usefullresourses/raznica-mejdu-rest-i-soap>

<https://habr.com/ru/companies/otus/articles/737610/>

https://habr.com/ru/articles/90696/

<https://www.json.org/json-ru.html>

<https://habr.com/ru/articles/554274/>

https://habr.com/ru/articles/524288/