КЛИЕНТ-СЕРВЕРНОЕ ВЗАИМОДЕЙСТВИЕ. API.

Урок 1

API-ЭТО ИНТЕРФЕЙС ИЛИ СПОСОБ ВЗАИМОДЕЙСТВИЯ ОДНОЙ СИСТЕМЫ С ДРУГОЙ

- Типы api:

- - Local api , то что позволяет компонентам отделяй системы взаимодействовать между собой
- - Remote api , позволяет нам связывать между собой несколько систем (тут 2 протокола soap, rest)

- Api всегда появляется раньше gui (графической оболочки), то есть мы можем тестировать api, до того как фронт разработчик сделал интерфейс

- Api это своего рода контракт (т.е. между нашими программами составляются договора, которые указывают как к этой программе можно обращаться

- Как работает api

- - Вызов операции (метод)
- - Входные данные (http request) То есть контент-тело нашего запроса, сообщение об ошибке-статус код))
- - Выходные данные (http response)

- Способы вызова апи:
- Прямые:

- вызов функции самой системой (когда мы говорим про локальный апи)

- Вызываем метода другой системы (есть одна система, допустим сайт и при помощи апи можем обращаться к вызову платежной системы)

- вызов метода человеком (постман или свагер)

- Косвенные:

- GUI>API (даже если мы работаем с интерфейсом, то наш интерфейс взаимодействует с апи так как пользователя все равно интересует только фронт-энд, а то что под копотям ему не интересно)

- Апи используется не только для каких то удаленных задач и для связи и взаимодействия нескольких систем, но и для работы внутри одной системы когда один компонент обращается к другому

ВЕБ-СЕРВИСЫ

- Веб-сервисы это программа, которая организовывает взаимодействие между сайтами, то есть информация с одного портала передаётся на другой

- Веб-сервисы это такая веб ориентированная технология, которая позволяет программам общаться между собой используя стандартные форматы, такие как: xml и json, и посредством специального протокола soap и архитектурного стиля rest

SOAP

- SOAP - это протокол обмена структурированными сообщениями в распределённый вычислительной среде.
- Этот протокол использует для обмена произвольными сообщениями в формате XML.
- Soap может использоваться с любым протоколом прикладного уровня.

XSD

- XSD - описывает структуры нашего HTML документа и типы данных, которые там могут храниться. То есть в веб сервисе есть такой файлик XSD - и с помощью него мы можем задавать различные элементы, данные внутри наших XML-ек и передавать их посредством сети интернет.

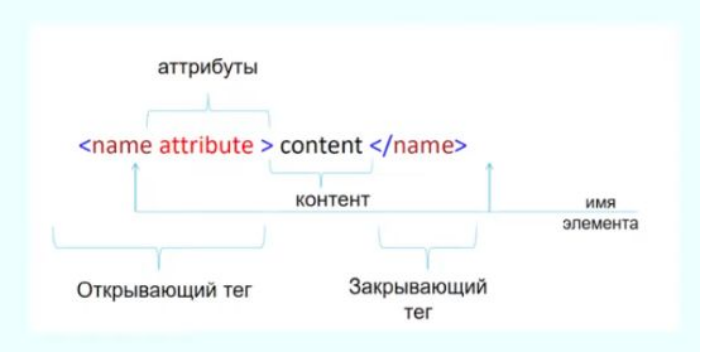
WSDL

- WSDL - это такой файлик, с таким расширением и с таким форматом. Он написан на языке WSDL. Он описывает сообщения , заголовки , события , которые свойственны для нашего веб сервиса. То есть данный файлик описывает структуру нашего веб сервиса и он обязателен для SOAP протокола. То есть без этого файлика мы просто не сможем использовать soap протокол и в дальнейшем при тестировании именно этот файл облегчает работу, так как в отличии от REST архитектурного стиля, где нет такого документа в котором было четко структурировано работа нашего веб сервиса тестировать там бывает тяжеловато.

- Каждый документ WSDL можно разделить на следующие логические части:
- - элементы данный, это message,то есть это сообщение которое использует наш веб сервис
- - Информация о типах данных, это type. Это информация определяет виды отправляемых и получаемых сервисом XML сообщений
- - Porttype это список операций , которые могут быть выполнены с нашими сообщениями

ПРАВИЛА НАПИСАНИЯ XML:

- - XML это язык , который очень похож на HTML, но в отличии от HTML у него другая расшифровка , это расширенный язык разметки и если HTML используется для разметки страницы (для верстки), то XML хранит в себе некоторую информацию и с помощью него эту информацию можно передавать при общении наших веб сервисов
- - Структурной единицей XML являются элементы (открывающий тег, закрывающий тег, контент который хранится между этими двумя тегами и атрибуты
- - У XML один корневой элемент
- - Все элементы должны имеет закрывающие теги
- - Названия наших тегов регистрозависимые
- - Элементы не должны пересекаться
- - Все значения атрибутов в кавычках
- - <, >, & нельзя использовать в тестовых блоках. Вместо этих символов их нужно прописывать в текстовом формате этих символов
- - Объявления XML - первая строка (номер версии, Калиновка наших символов и параметр stand-alone, который указывает запрещены ли ссылки на внешние документы



Элементы не должны пересекаться

```
<!-- valid -->
<b>This is bold text.</b> <i><b>This is bold italic text.</b> This is italic text.</i>

<!-- invalid -->
<b>This is bold text. <i>This is bold italic text.</b> This is italic text.</i>
```

Все элементы должны иметь закрывающие тэги

```
<?xml version="1.0" encoding="UTF-8"?>
<person>
    <familyName>Kress</familyName>
    <not_closed_element>
</person>
```

Все значения атрибутов в кавычках

```
<!- invalid -->
<person name=John/>
<!- valid -->
<person name="John" surname='Doe'/>
```

Названия регистрозависимые

```
<message>This is correct</message>

<Message>This is incorrect</message>
```

Только один корневой элемент

```
<?xml version="1.0" encoding="UTF-8"?>
<person>
    <givenName>Peter</givenName>
    <familyName>Kress</familyName>
</person>
<!-- Below is invalid element -->
<person>
    <givenName>John</givenName>
    <familyName>Doe</familyName>
</person>
<person/>
```

Объявления XML - первая строка

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

<,>,& нельзя использовать в текстовых блоках

```
<!- invalid -->
<text>I & my dog</text>
<!- valid -->
<text>I &amp; my dog</text>
```

&lt;	<
&gt;	>
&amp;	&
&apos;	'
&quot;	"

ОТЛИЧИЯ SOAP ОТ REST:

- - наличие WSDL
- - Сообщения обмениваются с помощью XML
- - Информация о том как должна структурироваться информация содержится в XCD

REST

- REST - архитектурный стиль
- К нему не применяются какие то жесткие правила
- Ему не нужны WSDL
- REST используют все больше и больше так как:
- - этот архитектурный стиль позволяет записывать информацию в более удобном формате, который занимает меньше места и повышает производительность нашей системы
- - Он направило зависим, к нему горазда меньше требований поэтому его используют все чаще

REST ОТ RESTFUL

- Если rest это архитектурный стиль с помощью которого у нас описывается структура передачи информации у веб сервисов, то RESTFUL это характеристика наших веб сервисов т.е. это такие веб сервисы которые полностью отвечают требованиям rest

ОТЛИЧИЯ REST И SOAP ВЕБ СЕРВИСАМИ:

- - Rest поддерживает различные форматы (json, xml, текстовые форматы). Soap поддерживает только xml
- - Rest работает только по протоколам http/https. Soap в отличии от rest может работать с различными протоколами
- - Soap на основе чтения не может быть помещён в кеш, в отличии от rest, который может быть закеширован
- - Rest это архитектурный стиль у которого нет огромного кол-ва правил, которым он может подчиняться
- - Soap это протокол, который сильно ограничен теми правилами, которые к нему предъявляются
- - Rest это простота, soap это стандарт
- - Soap более безопасный , ресурсоемкий , он медленнее и разработка дольше

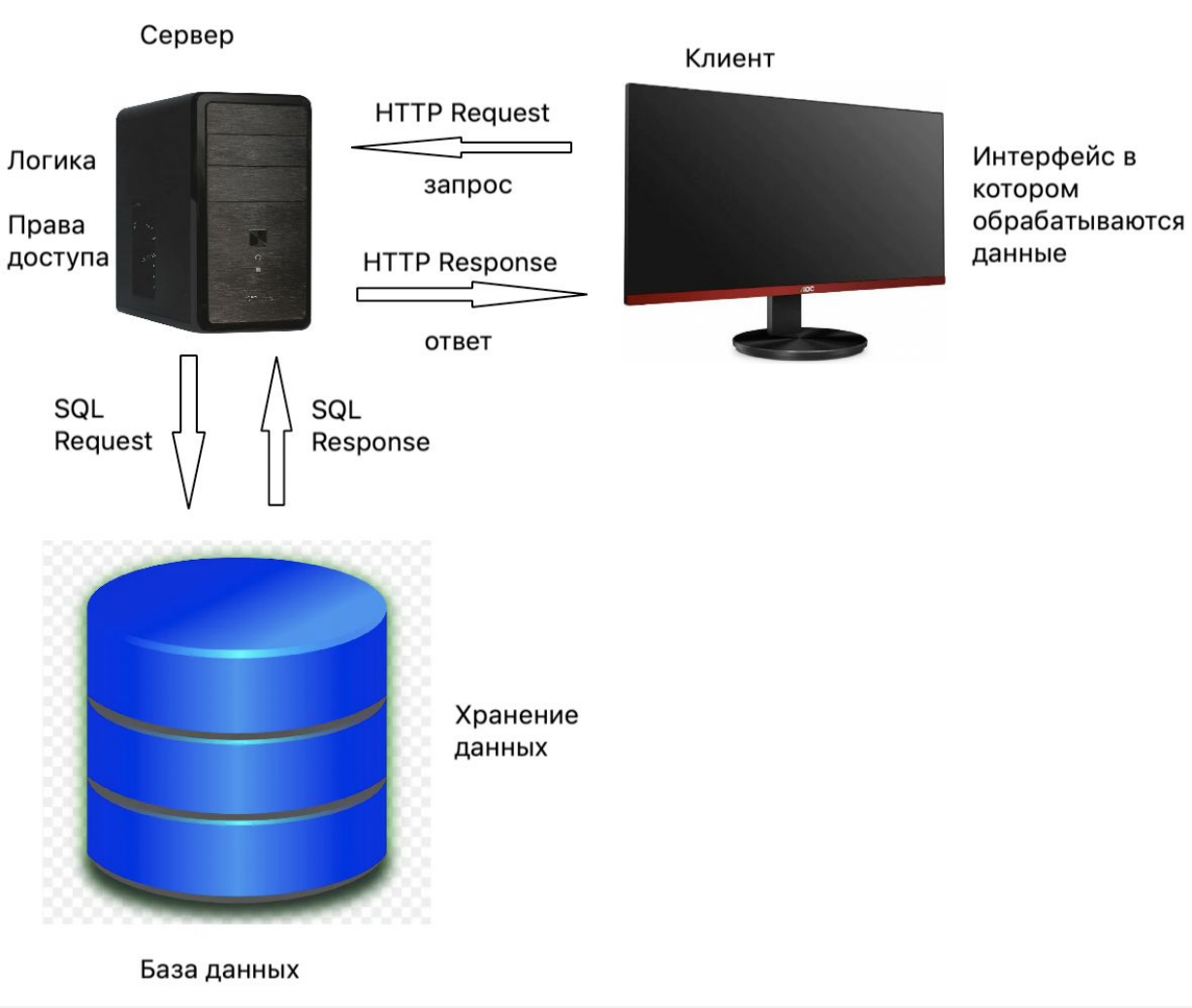
- Клиент(браузер, мобильное приложение)

Клиент запрашивает информацию с сервера (как правило через интерфейс (типо кнопки на сайте), который создал frontend-разработчик)

- Сервер (ПО установленный на компьютер)

Сервер принимает запросы от клиентов, обрабатывает их, если нужно извлекает данные из базы данных (БД) и отправляет их клиенту

- Обмен данными осуществляется по протоколу HTTP



HTTP

- Протокол Передачи Гипертекста — это протокол, который определяет язык для клиентов и серверов, чтобы общаться друг с другом.
- Как мы поняли HTTP – это язык, с помощью которого взаимодействуют между собой клиент и сервер.

ОСНОВНЫЕ СОСТАВЛЯЮЩИЕ ПРОТОКОЛА HTTP

- Данные (ЧТО отправить?) – то, что мы запрашиваем, например, запрос погоды в Москве

  -- то что создаем, например, сообщение в контакте

- URL (КУДА отправить?)- адрес сайта в интернете, т.е. куда мы отправляем запрос, например, https://yandex.ru

- Method (Что с этим сделать?) – то, что мы хотим сделать с данными

  Метод говорит серверу, что сделать с данными: извлечь из БД, создать новые данные в БД, обновить существующие или удалить данные из БД.

ОСНОВНЫЕ МЕТОДЫ ПРОТОКОЛА HTTP

- GET – получить данные с сервера
- POST – отправить данные на сервер
- PUT/PATCH – обновить данные на сервере, если они есть или создать новые, если их нет
- DELETE – удалить данные с сервера

СТРУКТУРА URL АДРЕСА

- Протокол – каким образом осуществляется обмен данными между клиентом и сервером

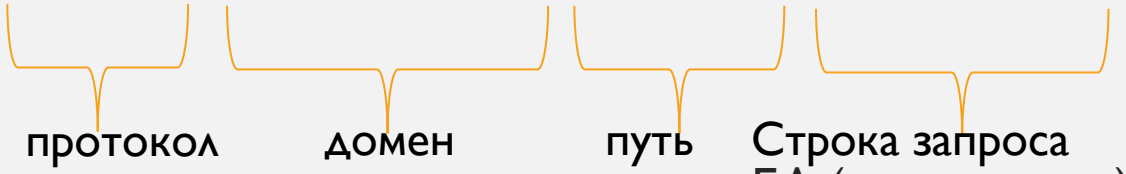
- Домен – адрес сайта в интернете

- Путь – указывает какие данные извлечь из БД (т.е. мы запрашивает friends (друзей), то для нашего профиля сервер извлекает всех друзей из БД для нашего профиля

ДАННЫЕ

- В зависимости от метода данные можно передавать в URL-адресе (метод GET), либо в теле запроса (метод POST, PUT, DELETE). Почему так? В методе GET в URL-адресе есть ограниченное кол-во передаваемых данных(символов)-> не более 255 шт., а в POST до 8кб (т.е. очень много)
- Передача данных методом GET в URL-адресе, а именно в троке запроса (метод GET используется, чтобы получить данные с сервера, т.е. извлечь или найти данные из БД)

https://google.com/search?q=погода



?q= это переменная т.е. то чему равно q мы ищем в БД (ищем погоду)

- Используя методы POST, PUT, DELETE данные отправляются в теле запроса

Данные в теле запроса передаются в формате Json

JSON

- Json – это формат обмена данными типа «ключ-значение»

(т.е. имя - Вася, фамилия – Иванов, возраст – 23, город – Москва)

{

”name” : ”Вася”,

“surname” :“Иванов”,

“age” : 23,

“city” :“Москва”

}

“ключ” :”значение”

СТРУКТУРА HTTP ПРОТОКОЛА

Метод GET		Метод POST, PUT, DELETE
URL: https://google.com/search?q=погода **Method: GET** **Status code: 200**	1	**URL: https://vk.com/registr** **Method: POST/PUT/DELETE** **Status code: 200**
Request headers	2	Request headers
Response headers	3	Response headers

	4	{
		”name” : ”Вася”,
		“surname” : “Иванов”,
		“age” : 23,
		“city” : “Москва”
		}

1. General headers – основной заголовок (что, куда, каким методом передаем)
2. Request headers – заголовок запроса (инфа о браузере: язык, время, браузер)
3. Response headers – заголовок ответа (инфа о сервере)
4. Тело запроса – данные, которые отправляем на сервер

СТАТУС КОДЫ

- 100-е (100-199) – информационные (102 Processing - "В обработке". Этот код указывает, что сервер получил запрос и обрабатывает его, но обработка ещё не завершена.)
- 200-е (200-299) – успешные (200 OK – "Успешно". Запрос успешно обработан.)
- 300-е (300-399) – перенаправление (302 Found - "Найдено". Этот код ответа значит, что запрошенный ресурс временно изменён. Например, при переходе на vk.com пользователя 302 статус кодом переведут на страницу vk.com/feed.)
- 400-е (400-499) – ошибка клиента (frontend) (404 Not Found – "Не найден". Сервер не может найти запрашиваемый ресурс. Код этого ответа, наверно, самый известный из-за частоты его появления в вебе.)
- 500-е (500-599) – ошибка сервера (backend) (500 Internal Server Error - "Внутренняя ошибка сервера". Сервер столкнулся с ситуацией, которую он не знает, как обработать.)

REST – РЕАЛИЗАЦИЯ ИНТЕРНЕТА НА ОСНОВЕ ПРОТОКОЛА HTTP

- Архитектура Rest = клиент-серверное взаимодействие

- Отличие методов Get и Post
- В методе get данные передаются url-адресе
- В методе post данные передаются в теле запроса

- Отличие методов Put и Patch
- В методе Put данные меняются целиком
- Метод Patch изменяет отдельные поля ресурса
- Отличие методов Put и Post
- Метод POST всегда создаёт новые данные.
- Метод PUT проверяет существуют ли уже такие данные, если да, то он обновляет их, если нет, то создаёт новые. (Например, при отправлении сообщений vk используется метод POST и каждый раз создаётся новое сообщение, а методом PUT мы обновляем имеющиеся сообщение, поэтому будет изменено одно и то же сообщение.)

- Отличие протоколов HTTP и HTTPS
- HTTPS не является отдельным протоколом передачи данных, а представляет собой расширение протокола HTTP с надстройкой шифрования;
- Передаваемые по протоколу HTTP данные не защищены, HTTPS обеспечивает конфиденциальность информации путем ее шифрования;
- HTTP использует порт 80, HTTPS — порт 443.