# **--- Back End ---**

**API (Програмный Интерфейс Приложения)** – Способ взаемодействия BACKEND и FRONTEND чаксти сайта через данные в формате JSON. Cпособ взаемодействия програм между собой или взамойдествие пользователей с пролижениями, как они могут общаться и передавать данные. Клиент сможет только выполнять только те действия которые заложил разработчик, залезть под капот приложения не сможет

**BackEND -> API -> JSON -> FrontEND**

**Клиент -> API -> Приложение(На Сервере)**

**Чужой Сервер -> API -> Приложение(На Севере)**

**RESTful API (Передача Репризентативного Состояния) –** тип интерфейса с помощью которого компьютеры передавют инормацию друг другу. Отправка данных на сервер(в наше приложение) или принятие данных от сервера(от нашего приложения) происходить по URL адресу который записывает в адресс клиент. GET , POST , PUT , DELETE запросы от клиента к серверу. Поддерживате обмен данных в таких форматах как JSON , HTML , XML , YAML . Сервер не запоминает ничего про перидущий запрос , каждий новый URL запрос нужно приписывать полностью с нуля

**JSON -** передача данных от клиента к серверу(или наоборот) или от сервера к серверу в формате ключ значение { “name”: “max” , “age”: 35 }. Такие файлы имеют разширение my\_file.json

**Кеширование(Caching)** - когда данные, которые принял клиент от нашего сервера(приложения) сохраняються в браузере клиента на несколько минут для повторного использования. Созданно чтоб облегчить работу нашего сервера и не оптравлять одну и туже информацию несколько раз

**↓** **Составные Части Запроса ↓**

**REQUEST LINE** - ето URL адресс в котором пользователь списиавет нужный ему метод взамодействия с серверов, ето GET , POST , PUT , DELETE. Например POST/my\_example/1.5

**HEADER** - информация клиенте, данные авторизации , информация об пк клиента. Например

Host: developer.mozilla.org

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:50.0) Gecko/20100101 Firefox/50.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate, br

**BODY MESSAGE** - информация, которую передает клиент на сервер. Ета информация создержит действия, которые необходимо выполнить серверу. Ето всегда файл в формате .json. Например

{“name” : ”mas” , “age”: 20 , “price” : 120, }

**↓** **Статус Коды Запроса ↓**

**Статус Код -** код который присылаеться в ответе пользователю от сервера, показывает успешность или не успешность запроса от пользователя к серверу

**1XX** - просто информация о запросе

**2XX -** показывает что запрос прошел успешно

**3XX -** показывает перенапралвение на другую страницу (redirect)

**4XX -** показывает что на стороне клиента произошла ошибка

**5XX -** показывает что на стороне сервера произошла ошибка

**↓ Типы Запросов ↓**

**GET Запрос -** [www.kondrich](http://www.kondrich) / category / top **?** name = max **&** age = 17  **-** служит для получение данных от сервера к клиенту

**POST Запрос** - www.kondrich / category / name\_category - служит для отправки данных от клиента на сервер (сохранение етих данных на сервер)

**PUT Запрос** - [www.kondrich](http://www.kondrich) / category / 1234 - служит для обновление запроса для отправки данных от клиента на сервер (изменение етих данных на сервере)

**DELETE Запрос -** [www.kondrich](http://www.kondrich) / category / 1234 - служит для удаления какой-то информации, которую отправил клиент на сервер (удаленные етих данных из серверв)

**Маршрутизация в Сети Интернет** - процесс опеределения найлучшего маршрута передачи данных от пользователя к серверу(и наоборот). Способ взаемодействия пользователся с сервером

**Протоколы** - набор правил(соглашений) с помощью которых сервера соединяються и обменються данными между собой

**Web Сервер (Apache , Nginx )** – програма запущеная на сервера, которая принимает постояно запросы от пользователся к сайту и перенапрявляет на обработку в фреймворки (Django) или языки програмирования (PHP)

**WSGI (Web Server Getaway Enterface**) - способ соединения(взаемодействия) web-сервера с python кодом в виде фреймворка Django. Передача запроса , который приняли от пользователя из web-сервера к фреймворку Django

**Local Host** **(127.0.0.1)** – доменное имя для компьютера пользователя. Позволяет использовать компьютер , как сервер где можна запускать программы или сайты

**Доменое Имя** - название сайта связаное с IP - адресом в интернете

**HTTP (Hyper Text Transfet Protocol)** - протокол по которым передаються страници написанные на HTML ( гипертекстовые документы) в интернете

**HTTPS (Hyper Text Transfer Protocol Secure)** - защищеный протокол по которым передаються страници написанные на HTML ( гипертекстовые документы) в интернете, который использует шифрование TLS

**Протокол Передачи Данных -** набор правил, по которым определяеться передача данные в сети интернет

**0.0.0.0 : 8000 --> IP адресс - 0.0.0.0 , Порт - 8000**

**IP (Inthernet Protocol)** - число, которое идентифицирует устройства в сети интернет. Ето устройство может быть пк , роутер , телефон , веб-сервер. Протокол сетевого уровня, его задача отправка пакетов от отправителя к получателю

**Порт** - число которое идентифицирует приложения , которые работают на устройствах в сети интернет

**Слушает Порт** - порт, который принимает на себя данные

**Сокет** - обьединение IP адреса и Порта, тоесть полный адресс

**Firewall (Брандмауер)** - програма для фильтрации пакетов нашего пк с удалеными сервера. Делит порты на свои и вражеские. Разрешеные порты свои , они подключены и от них можна принимать пакеты(данные) , а с вражеские порты нельзя подключать и следсвие нельзя принимать пакеты

**TCP / IP** - набор протоколов для обмена информацией в интернете. Связывание устройств происходит про IP адрессу компьютера(сервера) и Порту приложение, которые работают на нем. Внутри себя использует TCP , IP , UDP , HTTP , HTTPS , DNS протоколы

**TCP -** транспортный протокол с установлением соединение между устройствами и с гарантированой доставкой пакета. Если пакет не дошел, тогда посилаеться повторно

**UDP -** протокол без установление соединения между устройствмаи и без гарантированой доставкой пакетов

|  |  |
| --- | --- |
| Уровни Набора TCP / IP | Набор Протоколов TCP / IP |
| Прикладной | HTTP , HTTPS , FTP , DNS |
| Транспортый | TCP , UDP ↑ |
| Сетевой (Межсетевой) | IP (DHCP) ↑ |
| Уровень Сетевого Доступа (Канальный) | Ethernet , WLAN , WIFI ↑ |

Каждый уровень набора TCP / IP имеет свои протоколы передачи пакетов между устройствами. Каждый протокол зделан под свою задачу. Например передача пакетов между устройстами в одной сети или передача пакетов между устройствами в одном приложении