HW

Client\_Server

1) Прочитать про клиент-серверную архитектуру +

2) Что ткое HTTP и HTTPS

HTTP (Hiper Text Transfer Protokol) протокол, т.е. набор правил по которым осуществляется взаимодействие (передача информации) между клиентом и сервером. HTTPS в отличие от HTTP передает информацию в зашифрованном виде, т.о. является более защищенным.

3) HTTP методы

GET для получения информации от сервера. Нет тела, параметры передаются после знака ? в конце URL (из-за этого менее безопасный, т.к. виден). Кэшируемый и идемпотентный метод.

POST публикует информацию, содержащуюся в теле запроса, на сервере. Создает новую сущность. Не идемпотентный. Не кэшируется обычно.

PUT для обновления существующей записи. Если не находит старую запись, то создает новую. Идемпотентный. Не кэшируемый;

DELITE для удаления указанной сущности. Тело не обязательно. Идемпотентный. Не кэшируемый.

HEAD лайт-версия запроса Get, но не имеет тела. Запрашивает информацию с сервера, возвращает в ответе заголовки. Можно посмотреть там дополнительную информацию. Идемпотентный, кешируемый.

PATCH обновляет часть существующей записи. Обязательно содержит тело, в котором записаны параметры, по которым надо произвести обновление. Не кэшируемый, не идемпотентный.

CONNECT устанавливает связь с рессурсом, создает канал по которому клиент и сервер могут безопасно обмениваться сообщениями

OPTIONS позволяет получать сведения о деталях соединения с сервером

TRASE помогает отследить что происходит с запросом на пути к цели. Возвращает запрос в том виде, в котором он был получен. Нет тела. Идемпотентен, кэшируется

4) HTTP статус коды сервера 100 - информационные; 200 - информационные; 300 - перенаправление; 400 - ошибки на стороне клиента; 500 - ошибки на стороне сервера.

5) Что такое ядро браузера программа, которая отображает содержимое на основе языка разметки. Состоит из двух частей: движок рендера и Java Skript движок

6) Какие браузеры какие ядра используют Chromium (Blink + V8): Google Chrom, Yandex brouzer, Opera, Mikrosoft Edge, Vivaldi…

(WebKit + Nitro): Safari.

(Gecko + IonMonkey): Firefox, Netscape Navigator, Pale Moon…

7) Что такое API - правила, по которым приложения или части программы «общаются» друг с другом и обмениваются данными.

8. Что такое ендпоинты конечные точки - местоположение ресурсов к которым обращается API. Точки в которых клиент и сервер «общаются».

9) URL (URI, URL, URN) Uniform Resource Locator: адрес ресурса и способ обращения к нему.

URN: имя ресурса (только название, без определения как к нему подключаться)

URI = URL + URN

10) Идемпотентные HTTP методы Идемпотентность метода обозначает, что при отправке нескольких одинаковых запросов (с идентичным набором данных) один за другим, результат выполнения будет каждый раз одинаковым, не зависимо от количества запросов. Т.е. изменение базового состояния системы произойдет после первого запроса, а все последующие оставят состояние неизменным. К идемпотентным методам относят: GET, HEAD, PUT, OPTIONS. Методы DELETE и PATCH могут быть и идемпотентными и нет.

11) Безопасные HTTP методы - методы, которые не изменяют состояние сервера: HEAD, OPTIONS, GET.

12) Иденфикация - процесс, определяющий пользователя ( т.е. кто будет пользоваться системой - логин)

Аутентификация - проверка, что пользователь настоящий. ( Проверяется, что введенный пользователем пароль совпадает с паролем, соответствующем данному пользователю). Бывает двухфакторная и т.д.

Авторизация - предоставление соответствующих прав пользователю.

13) Что такое IP Интернет-протокол - набор правил, регулирующих адресацию, маршрутизацию и передачу данных в интернете. IP-адрес - уникальный идентификатор устройства в сети (локальной или в интернете)

14) Что такое октаты в DNS последовательности шестнадцатиричных чисел при помощи которых кодируются сообщения между клиентом и DNS-сервером

15) Что такое порт - идентефиктатор (числовой адрес) конкретной программы или процесса на данном IP-адресе (аналогия - квартира в почтовом адресе)

сколько портов у Linux сервера

Количество портов на Linux-сервере зависит от многих факторов, в том числе от используемых протоколов и настроек сетевых соединений. Обычно на Linux-сервере имеется 65535 портов, причем порты с номерами до 1023 зарезервированы для системных служб и приложений. Оставшиеся порты (с номерами от 1024 до 65535) доступны для пользовательских приложений. Чтобы узнать, какие порты используются на Linux-сервере, нужно воспользоваться командой netstat -tulpn, которая покажет список открытых портов и соответствующих им процессов.

16) Уровни OSI

7. Прикладной уровень (уровень приложений; [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *application layer*) — верхний уровень модели, обеспечивающий взаимодействие пользовательских приложений с сетью

6. Уровень представления ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *presentation layer*) обеспечивает преобразование протоколов и кодирование/декодирование данных. Запросы приложений, полученные с прикладного уровня, на уровне представления преобразуются в формат для передачи по сети, а полученные из сети данные преобразуются в формат приложений. На этом уровне может осуществляться сжатие/распаковка или шифрование/дешифрование, а также перенаправление запросов другому сетевому ресурсу, если они не могут быть обработаны локально.

5. Сеансовый уровень ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *session layer*) модели обеспечивает поддержание сеанса связи, позволяя приложениям взаимодействовать между собой длительное время. Уровень управляет созданием/завершением сеанса, обменом информацией, синхронизацией задач, определением права на передачу данных и поддержанием сеанса в периоды неактивности приложений.

4. Транспортный уровень ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *transport layer*) модели предназначен для обеспечения надёжной передачи данных от отправителя к получателю. При этом уровень надёжности может варьироваться в широких пределах. Существует множество классов протоколов транспортного уровня, начиная от протоколов, предоставляющих только основные транспортные функции (например, функции передачи данных без подтверждения приёма), и заканчивая протоколами, которые гарантируют доставку в пункт назначения нескольких пакетов данных в надлежащей последовательности, мультиплексируют несколько потоков данных, обеспечивают механизм управления потоками данных и гарантируют достоверность принятых данных.

3. Сетевой уровень ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *network layer*) модели предназначен для определения пути передачи данных. Отвечает за трансляцию логических адресов и имён в физические, определение кратчайших маршрутов, коммутацию и маршрутизацию, отслеживание неполадок и «заторов» в сети.

2. Канальный уровень ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *data link layer*) предназначен для обеспечения взаимодействия сетей на физическом уровне и контроля ошибок, которые могут возникнуть. Полученные с физического уровня данные, представленные в битах, он упаковывает в [кадры](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D0%B4%D1%80_(%D1%82%D0%B5%D0%BB%D0%B5%D0%BA%D0%BE%D0%BC%D0%BC%D1%83%D0%BD%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D0%B8)), проверяет их на целостность и, если нужно, исправляет ошибки (либо формирует повторный запрос повреждённого кадра) и отправляет на сетевой уровень. Канальный уровень может взаимодействовать с одним или несколькими физическими уровнями, контролируя и управляя этим взаимодействием.

1. Физический уровень ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *physical layer*) — нижний уровень модели, который определяет метод передачи данных, представленных в двоичном виде, от одного устройства (компьютера) к другому. Осуществляют передачу электрических или оптических сигналов в кабель или в радиоэфир и, соответственно, их приём и преобразование в биты данных в соответствии с [методами кодирования цифровых сигналов](https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%B7%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).

17) Хедеры http запросов используют только в запросах клиентов.

Referer: - это абсолютный или относительный URI ресурса, с которого сделан текущий запрос.

User-agent: показывает программное обеспечение клиента и его характеристики (браузер и его версия, язык, движок браузера и его версия, операционная система).

Server: указывает сервер клиента и его характеристики.

Via: указывает прокси клиента и его характеристики.

From: указывает имя/адрес пользователя, от чьего имени выполняется запрос.

Accept: содержит разделенные точкой с запятой список типов контента, которые клиент может понять.

Accept-Language: язык, установленный по умолчанию. Если несколько, то первый - предпочтительный

Authorization: данные пользователя для проверки пользовательского агента

Cookie: отправляет файлы куки, хранящиеся в браузере для этого домена

Accept-Encoding: сообщает серверу может ли браузер принимать сжатые данные