**Вопросы к лабораторной работе №1**

1. В каком документе описан протокол HTTP?

RFC (Request for Comments) 2616

1. Какого уровня HTTP-протокол в моделях OSI/ISO, TCP/IP?

На прикладном уровне модели OSI/ISO и на уровне приложений в модели TCP/IP.

1. Что такое web-приложение?

**Веб-приложение** – клиент-серверное приложение, в котором клиент и сервер взаимодействуют с помощью протокола HTTP.

1. Какие порты использует HTTP-протокол?

80 – HTTP

443 – HTTPS

1. Что такое RFC?

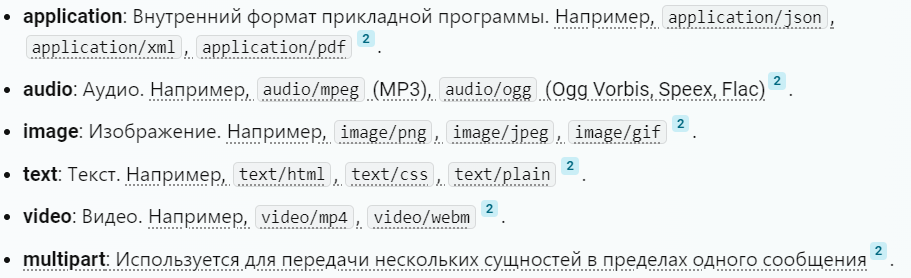
RFC (Request for Comments) – это серия документов, содержащих технические спецификации и стандарты, широко применяемые в Интернете.

1. Почему HTTP-протокол называют ассиметричным?

Потому что сообщения Request и Response отличаются (т.к. роли сервера и клиента разные при обмене сообщениями)

1. Что такое MIME?

MIME: Multipurpose Internet Mail Extensions - многоцелевые расширения Internet-почты. Используется и как стандарт кодирования Internet-сообщений.



**Заголовки:**

Cookie – содержит информацию, которую сервер отправляет браузеру пользователя. Браузер сохраняет эту информацию и отправляет её обратно на сервер с каждым последующим запросом. Это позволяет серверу узнать, что несколько запросов пришли от одного и того же браузера, что полезно, например, для аутентификации пользователя.

Postman-Token – используется инструментом Postman для предотвращения нежелательного кэширования запроса. Значение этого заголовка - это уникальный идентификатор, который генерируется для каждого запроса.

Host – содержит имя домена, для которого предназначен запрос, и, опционально, номер порта. Если порт не указан, то используется умолчательный порт протокола/сервиса (например, “80” для HTTP).

User-Agent – содержит информацию о клиенте, который отправляет запрос. Это может включать в себя детали о браузере, операционной системе и другую информацию, которая может быть полезна для сервера при формировании ответа.

Accept – какие типы контента (выраженные как MIME-типы), клиент может понять. Сервер затем выбирает одно из предложений и информирует клиента о своём выборе с помощью заголовка ответа Content-Type. (Если \*/\* - то клиент, у нас это браузер, может принять любой тип контента (html, json, png …)

Accept-Encoding – какие методы сжатия контента поддерживаются клиентом. Сервер затем может использовать эту информацию, чтобы определить, как сжать ответ для передачи по сети. (Если gzip, deflate, br – то это просто алгоритмы сжатия, инфа по каждому думаю излишне)

Connection – определяет, остаётся ли сетевое соединение активным после завершения текущей транзакции (запроса). Если в запросе отправлено значение keep-alive, то соединение остаётся и не завершается, позволяя выполнять последующие запросы на тот же сервер. (Кроме keep-alive еще может быть close (закрыть соединение), update (переключение на другой протокол))

Date – дата и время, когда было создано сообщение

Server – содержит информацию о программном обеспечении, используемом сервером для обработки запросов (В данном случае, у меня на сервере работает веб-сервер Apache, еще может быть nginx, GWS, Microsoft IIS)

Expires – содержит дату/время, по истечении которой ответ сервера считается устаревшим. Прошедшая или невалидная дата, например 0, обозначает, что ресурс уже устарел (Если в заголовке Expires указана дата в прошлом, это означает, что ресурс уже устарел. Однако, если в ответе сервера установлен заголовок Cache-Control с директивами max-age, заголовок Expires игнорируется)

Cache-Control – определяет поведение браузера при кэшировании. Он может содержать различные директивы, определяющие, должен ли браузер кэшировать данные и насколько свежими должны быть эти данные (Может содержать: no-store (запрещает кеширование), no-cache (необходимость отправить запрос на сервер для валидации ресурса перед использованием закешированных данных), must-revalidate (кешированный ответ должен быть проверен на актуальность перед использованием))

Pragma – используется для обратной совместимости с HTTP/1.0, где заголовок Cache-Control ещё не присутствует

Vary – определяет, как сопоставить будущие заголовки запроса, чтобы решить, можно ли использовать кешированный ответ, а не запрашивать новый с исходного сервера (Если установлен Accept-Encoding, то сервер может отправлять разные версии ресурса в зависимости от того, какие кодировки сжатия поддерживаются клиентом, может еще быть User-Agent (разные версии в зависимости от, к примеру, браузера), Accept-Language (разные версии в зависимости от языка пользователя), Origin (в зависимости от источника запроса)

Content-Encoding – указывает на кодировку, применённую к телу сообщения. Это позволяет клиенту информацию как декодировать тело, чтобы получить медиа-тип ссылающийся на заголовок Content-Type

Keep-Alive – используется для управления постоянными соединениями. Он может указывать, сколько времени и сколько запросов может оставаться открытым текущее неиспользованное соединение (Timeout – кол-во секунд между двумя последовательными запросами на одном и том же соединении, max – максимальное кол-во запросов, которое можно отправить по одному соединению)

Connection – определяет, остаётся ли сетевое соединение активным после завершения текущей транзакции (запроса). Если в запросе отправлено значение keep-alive, то соединение остаётся и не завершается, позволяя выполнять последующие запросы на тот же сервер. (Кроме keep-alive еще может быть close (закрыть соединение), update (переключение на другой протокол))

Transfer-Encoding – указывает форму кодирования, используемую для безопасной передачи полезной нагрузки пользователю (chunked – передается чанками (по частям), есть еще gzip, compress, deflate)

Content-Type – описывает MIME-тип содержимого, передаваемого в теле сообщения

Host – адрес сервера и порт, на который был отправлен запрос

Sec-Ch-UA – политика безопасности браузера

Sec-Ch-UA-Mobile – указывает, является ли браузер мобильным (1) или нет (0)

Sec-Ch-UA-Platform – указывает ОС, на которой работает браузер

DNT – указывает на наличие Do Not Track (DNT) заголовка, который может указывать на предпочтения пользователя относительно отслеживания

Upgrade-Insecure-Requests – сигнализирует серверу о том, что браузер поддерживает апгрейд запросов к защищённым версиям, таким как HTTPS