**HW**

**Client\_Server**

1. ***Прочитать про клиент-серверную архитектуру***

**Веб-приложение** – это клиент-серверное приложение, в котором клиентом выступает браузер, а сервером – веб-сервер (в широком смысле).

Основная часть приложения, как правило, находится на стороне веб-сервера, который обрабатывает полученные запросы в соответствии с бизнес-логикой продукта и формирует ответ, отправляемый пользователю. На этом этапе в работу включается браузер, именно он преобразовывает полученный ответ от сервера в графический интерфейс, понятный пользователю.

**Клиен**т – это программа, с которой работает пользователь, это может быть ваш интернет-браузер или десктоп-программа, установленная на ваш локальный компьютер или ноутбук.

**Сервер** – удаленная машина или мощный компьютер, на котором хранится вся логика, код, некое хранилище информации.

**База данных** – что-то вроде библиотеки данных, упорядоченных по таблицам.

Особенности такой модели заключаются в том, что пользователь отправляет определенный запрос на сервер, где тот системно обрабатывается и конечный результат отсылается клиенту. В возможности сервера входит одновременное обслуживание сразу нескольких клиентов.

Если одновременно поступает более одного запроса, то такие запросы устанавливаются в определенную очередь и сервером выполняются по очереди. Порой запросы могут иметь свои собственные приоритеты. Часть запросов с более высокими приоритетами будут постоянно выполняться в первоочередном порядке!

**Параметры, которые могут реализоваться на стороне сервера:**

Хранение, защита и доступ к данным;

Работа с поступающими клиентскими запросами;

Процесс отправки ответа клиенту.

**Параметры, которые могут реализоваться на стороне клиента:**

Площадка по предоставлению пользовательского графического интерфейса;

Формулировка запроса к серверу и его последующая отправка;

Получение итогов запроса и отправка дополнительной группы команд (запросы на добавление, обновление информации, удаление группы данных).

***Архитектура системы клиент-сервер*** формулирует принципы виртуального общения между локальными компьютерами, а все правила и принципы взаимодействия находятся внутри протокола.

***Сетевой протокол*** – это особый набор правил, на основании которого выполняется точное взаимодействие между компьютерами внутри виртуальной сети.

**TCP/IP** – совокупность протоколов передачи информации. TCP/IP – это особое обозначение всей сети, которая функционирует на основе протоколов TCP, а также IP.

**TCP** – вид протокола, который является связующим звеном для установки качественного соединения между 2 устройствами, передачи данных и верификации их получения.

**IP** – протокол, в функции которого входит корректность доставки сообщений по выбранному адресу. При этом информация делится на пакеты, которые могут поставляться по-разному.

**MAC** – вид протокола, на основании которого происходит процесс верификации сетевых устройств. Все устройства, которые подключены к сети Интернет, содержат свой оригинальный MAC-адрес.

**ICMP** – протокол, который ответственен за обмен данными, но не используется для процесса передачи информации.

**UDP** – протокол, управляющий передачей данных, но данные не проходят верификацию при получении. Этот протокол функционирует быстрее, чем протокол TCP.

**HTTP** – протокол для передачи информации (гипертекста), на базе которого функционируют все сегодняшние сайты. В его возможности входит процесс запрашивания необходимых данных у виртуально удаленной системы (файлы, веб-страницы и прочее).

**FTP** – протокол передачи информации из особого файлового сервера на ПК конечного пользователя.

**POP3** – классический протокол простого почтового соединения, который ответственен за передачу почты.

**SMTP** – вид протокола, который может устанавливать правила для передачи виртуальной почты. Он ответственен за передачу и верификацию доставки, а также оповещения о возможных ошибках.

**КОНЦЕПЦИИ ПОСТРОЙКИ КЛИЕНТ-СЕРВЕРНОЙ СИСТЕМЫ**

**Тонкий клиент** – производительный сервер.  
При такой модели весь процесс обработки информации перенесен на мощности сервера, а у пользователя права доступа очень строго ограничены. Сервер начинает отправлять ответ, который вообще не требует дополнительной работы по обработке. Клиент взаимодействует с пользователем: создает и отправляет запрос, принимает входящие итоги и выводит данные на экран пользователя.

**Толстый клиент.**  
Концепция, при которой часть обработки данных предоставляет клиенту. В такой ситуации сервер является простым хранилищем информации, а вся деятельность по обработке и предоставлению данных переносится на ПК пользователя.

**ЕСТЬ СРАЗУ 2 ВИДА КЛИЕНТ-СЕРВЕРНЫХ АРХИТЕКТУР:**

**1. Двухуровневая, состоящая сразу из 2 узлов:**

сервер, который ответственен за получение входящих запросов и отправку ответа пользователю, применяя при этом собственные ресурсы системы;

клиент, который может предоставлять пользовательский графический интерфейс.

Особенности работы заключаются в том, что на сервер приходит определенный запрос, потом его обрабатывают и дают напрямую, без дополнительного применения группы внешних ресурсов.

**2. Трехуровневая система состоит из использования таких компонентов:**

предоставление информации – графический пользовательский, прикладной объект в виде сервера приложения;

менеджмент ресурсов – сервер БД, который может предоставлять данные.

Особенность работы состоит в том, что сразу несколько серверов могут обрабатывать клиентские запросы. Процесс распределения операций может существенным образом снизить нагрузку на используемый сервер.

Трехуровневая архитектура может трансформироваться до многоуровневой, возможностью установки группы дополнительных серверов. Подобная виртуальная архитектура позволяет существенным образом повысить эффективность функционирования информационных систем, а также выполнить оптимизированное распределение части ее программно-аппаратных ресурсов.

***2) Что такое HTTP и HTTPS***

HTTPS и HTTP – два протокола, с помощью которых передается информация в Интернете. Они предназначены для передачи текстовых данных между клиентом и сервером, а главное различие между ними – в наличии и отсутствии шифрования передаваемых данных.

* **HTTP** (от англ. HyperText Transfer Protocol) ― это протокол передачи данных в интернете. С его помощью браузер получает информацию от сервера и показывает пользователю контент. Это первый протокол, который создали для работы в веб-пространстве.
* HTTPS (HyperText Transfer Protocol Secure) - расширение протокола HTTP для поддержки шифрования в целях повышения безопасности. Данные в протоколе HTTPS передаются поверх криптографических протоколов SSL или TLS. SSL (Secure Sockets Layer) и TLS (Transport Level Security)

Различия:

* HTTP работает по порту 80, а HTTPS ― по порту 443;
* HTTP по сравнению с HTTPS работает быстрее, поскольку последнему нужно время для шифрования канала связи.
* HTTP работает на уровне приложения, а HTTPS - на транспорном уровне.

**РАЗЛИЧИЯ SSL СЕРТИФИКАТОВ**

[**SSL**-сертификаты](https://sweb.ru/ssl/) имеют несколько разновидностей. Они классифицируются следующим доверительным факторам:

1. Проверка в упрощенном виде — DV (domain validation). Подтверждение права на пользование доменом. Обычно такой сертификат можно получить бесплатно.

2. Стандартная проверка OV (organization validation). Кроме права на владение доменом, подтверждается фактическое существование организации.

3. Расширенная проверка EV (extended validation). Подтверждает большую степень доверия – к перечисленным выше факторам прибавляется правомерное осуществление работы компании.

***3) HTTP методы***

В основе HTTP - клиент-серверная структура передачи данных․ Клиент формирует запрос (request) и отправляет на сервер; на сервере запрос обрабатывается, формируется ответ (response) и передается клиенту.

Для того, чтобы указать серверу на то, какое действие мы хотим произвести с ресурсом, используется тип HTTP-запроса, который также называется **HTTP метод**. Существует несколько HTTP методов, которые описывают действия с ресурсами. Наиболее часто используемыми являются **GET и POST.**

**Метод GET** запрашивает информацию из указанного источника и не влияет на его содержимое. Запрос доступен для кеширования данных и добавления в закладки. Длина запроса ограничена (макс. длина URL - 2048)

|  |  |
| --- | --- |
| **Метод** | **Что делает** |
| GET | Запрашивает представление ресурса. Такие запросы могут только извлекать данные. используется для получения со стороны севера определенного ресурса. Если вы производите этот запрос, сервер ищет информацию и отправляет ее вам назад. По сути, он производит операцию чтения на сервере. Дефолтный тип запросов. |
| HEAD | Запрашивает ресурс, как и GET , но ответ приходит без тела. |
| POST | Отправляет сущности к указанному ресурсу. Может вызывать побочные эффекты на сервере и изменение состояния. нужен для создания определенного ресурса на сервере. Сервер создает в базе данных новую сущность и оповещает вас, был ли процесс создания успешным. По сути, это операция создания. |
| PUT | Заменяет текущие представления ресурса данными запроса. используются для обновления определенной информации на сервере. В таком случае сервер просто изменяет информацию существующих сущностей в базе данных и оповещает об успехе выполнения операции. |
| DELETE | Удаляет указанный ресурс. |
| CONNECT | Настраивает туннель с сервером, определенным по ресурсу. |
| OPTIONS | Описывает параметры соединения с ресурсом. |
| TRACE | Вызывает возвращаемое текстовое сообщение с ресурса. |
| PATCH | Частично изменяет ресурс. используются для обновления определенной информации на сервере. В таком случае сервер просто изменяет информацию существующих сущностей в базе данных и оповещает об успехе выполнения операции. |

Все HTTP Methods можно разделить на три большие группы:

1. **Безопасные** — не меняют данные, можно выполнять их в любой последовательности. К ним относятся GET, HEAD и ОPTIONS.
2. **Идемпотентные**— при повторном выполнении результаты ожидаемо одинаковые. GET, HEAD, PUT, DELETE, OPTIONS, TRACE.
3. **Неидемпотентные** — при повторном выполении результаты будут отличаться. POST и PATCH

***4) HTTP статус коды сервера***

Выделяют 5 классов кодов по состоянию:

* 1хх – группа информационных кодов. Их задача состоит в обработке информации, создании сообщения о том, что текущий запрос принят, и передаче данных.
* 2хх – успешная обработка запроса. Запрос уже получен сервером и успешно выполнен без каких-либо ошибок.
* 3хх – перенаправление (англ. redirect). Данный код сообщает о текущей необходимости отправить запрос по иному адресу при смене расположения нужного файла.
* 4хх – ошибки на стороне пользователя. Такие ошибки могут указывать на то, что данный файл по запрашиваемому адресу не найден.
* 5хх – ошибки на стороне сервера. При выполнении запроса случился сбой сервера.

|  |  |  |  |
| --- | --- | --- | --- |
| *Информационные* | | | |
| 100 | Continue | "Продолжить". Этот промежуточный ответ указывает, что запрос успешно принят и клиент может продолжать присылать запросы либо проигнорировать этот ответ, если запрос был завершён. | Только HTTP/1.1 |
| 101 | Switching Protocol | "Переключение протокола". Этот код присылается в ответ на запрос клиента, содержащий заголовок Upgrade:, и указывает, что сервер переключился на протокол, который был указан в заголовке. Эта возможность позволяет перейти на несовместимую версию протокола и обычно не используется. | Только HTTP/1.1 |
| 102 | Processing | "В обработке". Этот код указывает, что сервер получил запрос и обрабатывает его, но обработка ещё не завершена. | Только HTTP/1.1 |
| 103 | Early Hints | "Ранние подсказки". В ответе сообщаются ресурсы, которые могут быть загружены заранее, пока сервер будет подготавливать основной ответ. [RFC 8297 (Experimental)](https://datatracker.ietf.org/doc/rfc8297). | Только HTTP/1.1 |
| *Успешные* | | | |
| 200 | OK | "Успешно". Запрос успешно обработан. Что значит "успешно", зависит от метода HTTP, который был запрошен:   * GET: "ПОЛУЧИТЬ". Запрошенный ресурс был найден и передан в теле ответа. * HEAD: "ЗАГОЛОВОК". Заголовки переданы в ответе. * POST: "ПОСЫЛКА". Ресурс, описывающий результат действия сервера на запрос, передан в теле ответа. * TRACE: "ОТСЛЕЖИВАТЬ". Тело ответа содержит тело запроса полученного сервером. | HTTP/0.9 и выше |
| 201 | Created | "Создано". Запрос успешно выполнен и в результате был создан ресурс. Этот код обычно присылается в ответ на запрос PUT "ПОМЕСТИТЬ". | HTTP/0.9 и выше |
| 202 | Accepted | "Принято". Запрос принят, но ещё не обработан. Не поддерживаемо, т.е., нет способа с помощью HTTP отправить асинхронный ответ позже, который будет показывать итог обработки запроса. Это предназначено для случаев, когда запрос обрабатывается другим процессом или сервером, либо для пакетной обработки. | HTTP/0.9 и выше |
| 203 | Non-Authoritative Information | "Информация не авторитетна". Этот код ответа означает, что информация, которая возвращена, была предоставлена не от исходного сервера, а из какого-нибудь другого источника. Во всех остальных ситуациях более предпочтителен код ответа 200 OK. | HTTP/0.9 и 1.1 |
| 204 | No Content | "Нет содержимого". Нет содержимого для ответа на запрос, но заголовки ответа, которые могут быть полезны, присылаются. Клиент может использовать их для обновления кешированных заголовков полученных ранее для этого ресурса. | HTTP/0.9 и выше |
| 205 | Reset Content | "Сбросить содержимое". Этот код присылается, когда запрос обработан, чтобы сообщить клиенту, что необходимо сбросить отображение документа, который прислал этот запрос. | Только HTTP/1.1 |
| 206 | Partial Content | "Частичное содержимое". Этот код ответа используется, когда клиент присылает заголовок диапазона, чтобы выполнить загрузку отдельно, в несколько потоков. | Только HTTP/1. |

| Код | Ошибка | Описание |
| --- | --- | --- |
| 300 | Множественный выбор | Затребованный URL обозначает более одного ресурса, и робот не смог однозначно определить, к какой странице URL относится (получен код 300 Multiple Choices).  Исправьте заголовки или укажите ресурс правильно, и тогда робот сможет проиндексировать страницу. |
| 301 | Ресурс перемещен навсегда | Документ уже не используется сервером, а ссылка перенаправляет на другую страницу (получен код 301 Moved Permanently).  Так как пользователи не смогут увидеть подобные документы, показывать их в поиске не имеет смысла, и робот их не индексирует. Однако робот проиндексирует страницу, на которую установлено перенаправление, если она доступна. |
| 302 | Ресурс временно перемещен | Запрошенный ресурс временно находится под другим адресом (получен код 302 Found).  Так как пользователи не смогут увидеть подобные документы, показывать их в поиске не имеет смысла, и робот их не индексирует. Однако робот проиндексирует страницу, на которую установлено перенаправление, если она доступна. |
| 303 | Смотрите другой ресурс | Запрошенный ресурс находится под другим адресом и его следует запрашивать, используя метод GET (получен код 303 See Other). Если вы хотите, чтобы указанная страница находилась в поиске, она должна отвечать кодом 200. |
| 304 | Ресурс не изменялся | Получен код 304 Not Modified. Если страница не изменилась с момента последнего обращения робота, рекомендуется выдавать этот код. Это ускорит индексирование и уменьшит трафик. |
| 305 | Следует использовать прокси | Доступ к затребованному ресурсу может осуществляться только через прокси-сервер, указанный в заголовке Location (получен код 305 Use Proxy). |
| 307 | Временное перенаправление | Затребованный ресурс был временно переведен на другой адрес, который необходимо прописать в Location (получен код 307 Temporary Redirect). |

## Коды 4xx (ошибка клиента)

Содержит перечень кодов HTTP-статуса, использующихся для обозначения возможных ошибок в клиентском запросе (коды 4xx).

| Код | Ошибка | Описание |
| --- | --- | --- |
| 400 | Неверный запрос/Bad Request | Запрос не может быть понят сервером из-за некорректного синтаксиса. |
| 401 | Неавторизованный запрос/Unauthorized | Для доступа к документу необходимо вводить пароль или быть зарегистрированным пользователем. |
| 402 | Необходима оплата за запрос/Payment Required | Внутренняя ошибка или ошибка конфигурации сервера. |
| 403 | Доступ к ресурсу запрещен/Forbidden | Доступ к документу запрещен. Если вы хотите, чтобы страница индексировалась, необходимо разрешить доступ к ней. |
| 404 | Ресурс не найден/Not Found | Документ не существует. Если вы удалили какой-то раздел сайта, можно с помощью robots.txt запретить роботу обращаться к нему. Если такой страницы на сайте никогда не существовало, игнорируйте эту ошибку, возможно, кто-то поставил некорректную ссылку на ваш сайт. |
| 405 | Недопустимый метод/Method Not Allowed | Метод, определенный в строке запроса (Request-Line), не дозволено применять для указанного ресурса, поэтому робот не смог его проиндексировать. |
| 406 | Неприемлемый запрос/Not Acceptable | Нужный документ существует, но не в том формате (язык или кодировка не поддерживаются роботом). |
| 407 | Требуется идентификация прокси, файервола/Proxy Authentication Required | Необходима регистрация на прокси-сервере. |
| 408 | Время запроса истекло/Request Timeout | Сайт не передал полный запрос в течение установленного времени и робот разорвал соединение. |
| 409 | Конфликт/Conflict | Запрос конфликтует с другим запросом или с конфигурацией сервера. |
| 410 | Ресурс недоступен/Gone | Затребованный ресурс был окончательно удален с сайта. |
| 411 | Необходимо указать длину/Length Required | Сервер отказывается принимать запрос без определенного заголовка Content-Length. Поправьте заголовки на своем сервере;— тогда в следующий раз робот сможет проиндексировать страницу. |
| 412 | Сбой при обработке предварительного условия/Precondition Failed | При проверке на сервере одного или более полей заголовка запроса обнаружено несоответствие (сбой или ошибка при обработке предварительного условия). |
| 413 | Тело запроса превышает допустимый размер/Request Entity Too Large | Сервер отказывается обрабатывать запрос потому, что размер запроса больше того, что может обработать сервер. |
| 414 | Недопустимая длина URI запроса/Request-URI Too Long | Сервер отказывается обслуживать запрос, потому что запрашиваемый роботом URI (Request-URI) длиннее, чем сервер может интерпретировать. |
| 415 | Неподдерживаемый MIME тип/Unsupported Media Type | Сервер отказывается обрабатывать запрос, потому что тело запроса имеет неподдерживаемый формат. |
| 416 | Диапазон не может быть обработан/Requested Range Not Satisfiable | Сервер отказывается обрабатывать запрос, потому что значение поля Range в заголовке запроса указывает на недопустимый диапазон байтов. |
| 417 | Сбой при ожидании/Expectation Failed | Сервер отказывается обрабатывать запрос, потому что значение поля Expect в заголовке запроса не соответствует ожиданиям. |
| 422 | Необрабатываемый элемент/Unprocessable Entity | Сервер не в состоянии обработать один (или более) элемент запроса. |
| 423 | Заблокировано/Locked | Сервер отказывается обработать запрос, так как один из требуемых ресурсов заблокирован. |
| 424 | Неверная зависимость/Failed Dependency | Сервер отказывается обработать запрос, так как один из зависимых ресурсов заблокирован. |
| 426 | Требуется обновление/Upgrade Required | Сервер запросил апгрейд соединения до SSL, но SSL не поддерживается клиентом. |
| 429 | Слишком много запросов/Too Many Requests | Отправлено слишком много запросов за короткое время. Это может указывать, например, на попытку [DDoS-атаки](https://ru.wikipedia.org/wiki/DoS-%D0%B0%D1%82%D0%B0%D0%BA%D0%B0). Ответ может сопровождаться заголовком Retry-After, который указывает, через какое время можно повторить запрос. Яндекс не учитывает этот заголовок. |

## Коды 5xx (ошибка сервера)

Перечень кодов HTTP-статуса, использующихся для обозначения возможных ошибок сервера (коды 5xx).

| Код | Ошибка | Описание |
| --- | --- | --- |
| 500 | Внутренняя ошибка сервера/Internal Server Error | Сервер столкнулся с непредвиденным условием, которое не позволяет ему выполнить запрос. |
| 501 | Метод не поддерживается/Not Implemented | Сервер не поддерживает функциональные возможности, требуемые для выполнения запроса. Этот ответ соответствует состоянию, когда сервер не распознает метод запроса и не способен обеспечить его для любого ресурса. |
| 502 | Ошибка шлюза/Bad Gateway | Сервер, действуя в качестве шлюза или прокси-сервера, получил недопустимый ответ от следующего сервера в цепочке запросов, к которому обратился при попытке выполнить запрос. |
| 503 | Служба недоступна/Service Unavailable | Возникла ошибка из-за временной перегрузки или отключения на техническое обслуживание сервера. |
| 504 | Время прохождения через межсетевой шлюз истекло/Gateway Timeout | Сервер, при работе в качестве внешнего шлюза или прокси-сервера, своевременно не получил отклик от вышестоящего сервера, к которому он обратился, пытаясь выполнить запрос. |
| 505 | Версия НТТР не поддерживается/Version Not Supported | Сервер не поддерживает или отказывается поддерживать версию HTTP-протокола, которая используется в сообщении запроса робота. |
| 507 | Недостаточно места/Not Extended | Сервер не может обработать запрос из-за недостатка места на диске). |
| 510 | Отсутствуют расширения/Not Extended | Сервер не может обработать запрос из-за того, что запрашиваемое расширение не поддерживается. |

***5) Что такое ядро браузера***

**Браузер = оболочка + ядро**  
Среди них относительно больше типов оболочек и меньше ядер. Оболочка относится к оболочке браузера: например, меню, панели инструментов и т. Д. Он в основном предназначен для работы с пользовательским интерфейсом, настройки параметров и так далее. Он вызывает ядро ​​для реализации различных функций. Ядро - это ядро ​​браузера. Ядро - это программа или модуль, который отображает контент на основе языка разметки.

**Браузерный движок** ([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" \o "Английский язык) [layout engine](https://en.wikipedia.org/wiki/layout_engine)) — представляет собой [программу](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), преобразующую содержимое веб-страниц (файлы [HTML](https://ru.wikipedia.org/wiki/HTML), [XML](https://ru.wikipedia.org/wiki/Extensible_Markup_Language), цифровые изображения и т. д.) и информацию о форматировании (в форматах [CSS](https://ru.wikipedia.org/wiki/Cascading_Style_Sheets), [XSL](https://ru.wikipedia.org/wiki/Extensible_Stylesheet_Language) и т. д.) в интерактивное изображение форматированного содержимого на экране. Браузерный [движок](https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA) обычно используется в [веб-браузерах](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%B1%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80) (отсюда название), [почтовых клиентах](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%87%D1%82%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0) и других программах, нуждающихся в отображении и редактировании содержимого веб-страниц.

Ядро браузера можно разделить на две части: движок рендеринга (инженер макета или движок рендеринга) и движок JS. Он отвечает за получение содержимого веб-страницы (HTML, XML, изображения и т. Д.), Организацию информации (например, добавление CSS и т. Д.) И расчет режима отображения веб-страницы, а затем вывод ее на монитор или принтер. Разница в ядре браузера будет по-разному интерпретировать синтаксис веб-страницы, поэтому эффект рендеринга будет другим. Все веб-браузеры, почтовые клиенты и другие приложения, которым необходимо редактировать и отображать сетевой контент, требуют ядра. (См. Википедия). Движок JS анализирует язык Javascript и выполняет язык Javascript для достижения динамических эффектов веб-страницы. Сначала не было четкого различия между движком рендеринга и движком JS, а позже движок JS становился все более независимым, и ядро ​​имело тенденцию ссылаться только на движок рендеринга. Механизм рендеринга определяет, как браузер отображает содержимое веб-страницы и информацию о формате страницы. Разные ядра браузеров по-разному интерпретируют синтаксис записи веб-страниц, поэтому эффект рендеринга (отображения) одной и той же веб-страницы в браузерах разных ядер также может быть различным. Именно поэтому авторам веб-страниц необходимо тестировать веб-страницы в браузерах разных ядер. Покажите причину эффекта. Движок JS отвечает за интерпретацию, компиляцию и выполнение JavaScript, чтобы заставить веб-страницу достигать некоторых динамических эффектов. Но обычные ядра браузера можно разделить на эти пять типов: Trident, Gecko, Presto, Webkit, Blink.

***6) Какие браузеры какие ядра используют***

* Blink: Google
* Gecko: Mozilla
* WebKit: Apple
* EdgeHTML: Microsoft

### Разработка прекращена

* [Trident](https://ru.wikipedia.org/wiki/Microsoft_Trident) — [проприетарный](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%BF%D1%80%D0%B8%D0%B5%D1%82%D0%B0%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) движок [Microsoft Internet Explorer](https://ru.wikipedia.org/wiki/Microsoft_Internet_Explorer).
* [Presto](https://ru.wikipedia.org/wiki/Presto_(Opera)) — проприетарный движок, разработанный [Opera Software](https://ru.wikipedia.org/wiki/Opera_Software). 13 февраля 2013 года было объявлено[[2]](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80%D0%BD%D1%8B%D0%B9_%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA" \l "cite_note-2) об отказе Opera от Presto в пользу Blink.

### Поддерживаемые

* [Blink](https://ru.wikipedia.org/wiki/Blink_(%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA)) — движок браузера [Chromium](https://ru.wikipedia.org/wiki/Chromium), браузера [Google Chrome](https://ru.wikipedia.org/wiki/Google_Chrome) с 28 версии, [Microsoft Edge](https://ru.wikipedia.org/wiki/Microsoft_Edge) с 79 версии, [Opera](https://ru.wikipedia.org/wiki/Opera) c 15 версии и [Vivaldi](https://ru.wikipedia.org/wiki/Vivaldi). Он является ответвлением [WebKit](https://ru.wikipedia.org/wiki/WebKit).
* [Gecko](https://ru.wikipedia.org/wiki/Gecko) — [открытый](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D1%81%D0%BE%D1%84%D1%82) движок проекта [Mozilla](https://ru.wikipedia.org/wiki/Mozilla); используется в большом числе программ, основанных на коде Mozilla (браузере [Firefox](https://ru.wikipedia.org/wiki/Mozilla_Firefox), почтовом клиенте [Thunderbird](https://ru.wikipedia.org/wiki/Mozilla_Thunderbird), наборе программ [SeaMonkey](https://ru.wikipedia.org/wiki/SeaMonkey)).
* [WebKit](https://ru.wikipedia.org/wiki/WebKit) — движок для браузера [Apple Safari](https://ru.wikipedia.org/wiki/Apple_Safari), включенного в операционную систему [Mac OS X](https://ru.wikipedia.org/wiki/Apple_Mac_OS_X), и браузера [Google Chrome](https://ru.wikipedia.org/wiki/Google_Chrome) (до 2013 года). Встроен в библиотеку [Qt](https://ru.wikipedia.org/wiki/Qt) (Начиная с Qt 5.6 признан устаревшим).
* [KHTML](https://ru.wikipedia.org/wiki/KHTML) — разработан в рамках проекта [KDE](https://ru.wikipedia.org/wiki/KDE), используется в браузере [Konqueror](https://ru.wikipedia.org/wiki/Konqueror) и послужил основой для [WebKit](https://ru.wikipedia.org/wiki/WebKit).

***7) Что такое API***

API — описание способов, которыми одна компьютерная программа может взаимодействовать с другой программой. Обычно входит в описание какого-либо интернет-протокола, программного каркаса или стандарта вызовов функций операционной системы.

API (Application Programming Interface — программный интерфейс приложения, или интерфейс программирования приложений) — специальный протокол для взаимодействия компьютерных программ, который позволяет использовать функции одного приложения внутри другого.

Дословно API расшифровывается как Application Programming Interface. Это набор правил, позволяющий программам «общаться» друг с другом. Разработчик создает API на сервере и позволяет клиентам обращаться к нему.

API повсеместно [используют](https://1ats.ru/blog/chto-takoe-integraciya-po-api/) для взаимодействия программ и приложений с операционными системами или интернет-сайтами. Если бы Application Programming Interface [отключились](https://thecode.media/api/), то почти все сервисы в интернете и большинство компьютерных программ перестали бы работать.

С помощью API программисты [могут использовать](https://skillbox.ru/media/code/chto_takoe_api/) возможности разных приложений, не задумываясь о том, как эти приложения обрабатывают запросы и какой у них код.

***8. Что такое ендпоинты***

Эндпоинт (в переводе с англ. — конечная точка) представляет собой некий шлюз, который соединяет серверные процессы приложения с внешним интерфейсом. Простыми словами, это адрес, на который отправляются сообщения. Чтобы понять, что такое эндпоинты, важно упомянуть работу API.

Эндпоинт (Endpoint - конечная точка) — это само обращение к маршруту отдельным HTTP методом. Эндпоинт выполняют конкретную задачу, принимают параметры и возвращают данные Клиенту.

Конечные точки указывают, как получить доступ к ресурсу, а метод указывает разрешенные взаимодействия (такие как GET, POST или DELETE) с ресурсом.

Каждый api-сервис имеет endpoint, к которому и нужно обращаться, например отправлять http-запрос. Обычно это url. Т.о. endpoint - это url. Или такая. url включает в себя маршрут (основная часть адреса сервиса) и endpoint (часть url, которая содержит например имя вызываемого метода). Существует один маршрут и у него несколько эндпоинтов. Т.о. здесь endpoint - это только часть url.

***9) URL (URI, URL, URN)***

**URI (Uniform Resource Identifier)** – это строка символов, которая используется для идентификации какого-либо ресурса по его адресу или по его имени, либо по тому и тому вместе.

**URL (Uniform Resource Locator)** – это строка символов, которая используется для идентификации какого-либо ресурса, но только по его адресу, по его местоположению.

**URN (Uniform Resource Name)** – это строка символов, которая используется для идентификации какого-либо ресурса, но только по его имени.

Каждый URL является URI. Каждый URN является URI. Но не каждый URI, к примеру, является URL (он может быть URN)

URL чаще всего используется в Интернете для поиска ресурсов на сервере. URL буквально точно показывает нам, как определить ресурс, именно по его адресу. Если ввести подобный URL в строке поиска браузера, то будет осуществлен поиск соответствующего ресурса. И хотя URL на картинке ниже немного отличаются друг от друга своей структурой, есть определенный формат, как должен быть построен любой URL.

Любой URL состоит из нескольких компонентов. Протокол и хост являются обязательными, все остальные - нет.

Подопытный URL выше можно прочитать как: используя протокол https обратиться к домену www.mysite.com по стандартному порту 80, в каталоге найти товар желтого цвета с идентификатором 15, в браузере пользователя сразу же переместиться в область где указана цена.

URN служит для обозначения уникального имени ресурса, неважно, где этот ресурс располагается в данный момент времени или вообще. Такая природа URN (независимость от адреса) позволяет ресурсам перемещаться с одного места на другое. URN позволяет получить доступ к ресурсу по различным сетевым протоколам, обращаясь к одному и тому же имени.

***10) Идемпотентные HTTP методы***

Метод HTTP является идемпотентным, если повторный идентичный запрос, сделанный один или несколько раз подряд, имеет один и тот же эффект, не изменяющий состояние сервера. Другими словами, идемпотентный метод не должен иметь никаких побочных эффектов (side-effects), кроме сбора статистики или подобных операций. Корректно реализованные методы [GET](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/GET), [HEAD](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/HEAD), [PUT](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/PUT) и [DELETE](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/DELETE) **идемпотентны**, но не метод [POST](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/POST). Также все [безопасные](https://developer.mozilla.org/ru/docs/Glossary/safe) методы являются идемпотентными.

Для идемпотентности нужно рассматривать только изменение фактического внутреннего состояния сервера, а возвращаемые запросами коды статуса могут отличаться: первый вызов [DELETE](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/DELETE) вернёт код [200](https://developer.mozilla.org/ru/docs/Web/HTTP/Status/200), в то время как последующие вызовы вернут код [404](https://developer.mozilla.org/ru/docs/Web/HTTP/Status/404). Из идемпотентности [DELETE](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/DELETE) неявно следует, что разработчики не должны использовать метод [DELETE](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/DELETE) при реализации RESTful API с функциональностью **удалить последнюю запись**.

Обратите внимание, что идемпотентность метода не гарантируется сервером, и некоторые приложения могут нарушать ограничение идемпотентности.

GET /pageX HTTP/1.1 идемпотентен. Вызвавший несколько раз подряд этот запрос, клиент получит тот же результат:

GET /pageX HTTP/1.1

GET /pageX HTTP/1.1

GET /pageX HTTP/1.1

GET /pageX HTTP/1.1

POST /add\_row HTTP/1.1 не идемпотентен; если его вызвать несколько раз, то он добавит несколько строк:

POST /add\_row HTTP/1.1

POST /add\_row HTTP/1.1 -> Adds a 2nd row

POST /add\_row HTTP/1.1 -> Adds a 3rd row

DELETE /idX/delete HTTP/1.1 идемпотентен, даже если возвращаемый код отличается:

DELETE /idX/delete HTTP/1.1 -> Returns 200 if idX exists

DELETE /idX/delete HTTP/1.1 -> Returns 404 as it just got deleted

DELETE /idX/delete HTTP/1.1 -> Returns 404

***11) Безопасные HTTP методы***

Безопасный запрос — это запрос, который не меняет состояние приложения.

Метод HTTP является безопасным, если он не меняет состояние сервера. Другими словами, безопасный метод проводит операции "только чтение" (read-only). Несколько следующих методов HTTP безопасные: GET , HEAD или OPTIONS . Все безопасные методы являются также идемпотентными, как и некоторые другие, но при этом небезопасные, такие как PUT или DELETE.

Даже если безопасные методы являются по существу “только для чтения”, сервер всё равно может сменить своё состояние: например, он может сохранять статистику. Что существенно, так то, когда клиент вызывает безопасный метод, то он не запрашивает никаких изменений на сервере, и поэтому не создаёт дополнительную нагрузку на сервер. Браузеры могут вызывать безопасные методы, не опасаясь причинить вред серверу: это позволяет им выполнять некоторые действия, например, предварительная загрузка без риска. Поисковые роботы также полагаются на вызовы безопасных методов.

|  |  |  |
| --- | --- | --- |
| HTTP-Метод | Безопасный | Идемпотентый |
| GET | Да | Да |
| HEAD | Да | Да |
| OPTIONS | Да | Да |
| PUT | Нет | Да |
| DELETE | Нет | Да |
| POST | Нет | Нет |

***12) Иденфикация, Аутентификация, Авторизация***

**Идентификация**— это процедура распознавания субъекта по его идентификатору (проще говоря, это определение имени, логина или номера).

**Аутентификация** – это процедура проверки подлинности (пользователя проверяют с помощью пароля, письмо проверяют по электронной подписи и т.д.) проверка подлинности пользователя путем сравнения введенного им пароля с паролем, сохраненным в базе данных.

**Авторизация** – это предоставление доступа к какому-либо ресурсу (например, к электронной почте). предоставление определенному лицу прав на выполнение определенных действий.

* Сачала система запрашивает [логин](http://security.mosmetod.ru/paroli/121-login-parol-i-uchetnaya-zapis), пользователь его указывает, система распознает его как существующий — это идентификация.
* После этого Google просит ввести [пароль](http://security.mosmetod.ru/paroli/62-zashchita-ot-moshennikov), пользователь его вводит, и система соглашается, что пользователь действительно настоящий, ведь пароль совпал, — это аутентификация.
* Возможно, Google дополнительно спросит еще и одноразовый код из SMS или приложения. Если пользователь и его правильно введет, то система окончательно согласится с тем, что он настоящий владелец аккаунта, — это [двухфакторная аутентификация](http://security.mosmetod.ru/paroli/93-dvukhfaktornaya-autentifikatsiya).
* После этого система предоставит пользователю право читать письма в его почтовом ящике и все остальное — это авторизация.

***13) Что такое IP***

**IP-адрес (**от англ. [Internet Protocol](https://ru.wikipedia.org/wiki/Internet_Protocol)) — уникальный числовой [идентификатор](https://ru.wikipedia.org/wiki/%D0%98%D0%B4%D0%B5%D0%BD%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%82%D0%BE%D1%80) устройства в компьютерной [сети](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C), работающей по протоколу [IP](https://ru.wikipedia.org/wiki/IP). IP означает «Интернет-протокол» – набор правил, регулирующих формат данных, отправляемых через интернет или локальную сеть. Без него невозможно существование Интернета или какой-либо внутренней IP-сети.

По сути, IP-адрес – это идентификатор, позволяющий передавать информацию между устройствами в сети: он содержит информацию о местоположении устройства и обеспечивает его доступность для связи. IP-адреса позволяют различать компьютеры, маршрутизаторы и веб-сайты в интернете и являются важным компонентом работы интернета.

IP-адрес IPv4 имеет 32-битную (4 байта) структуру. Он разделён на 4 части, каждая из которых состоит из 8 бит (1 байт) и называется октетом. Каждый бит IP-адреса – цифра двоичной системы.

Пример адреса (IPv4) в двоичном виде: 11000000.10101000.00110010.00000001.

IP-адрес в десятичном виде: 192.168.50.1.

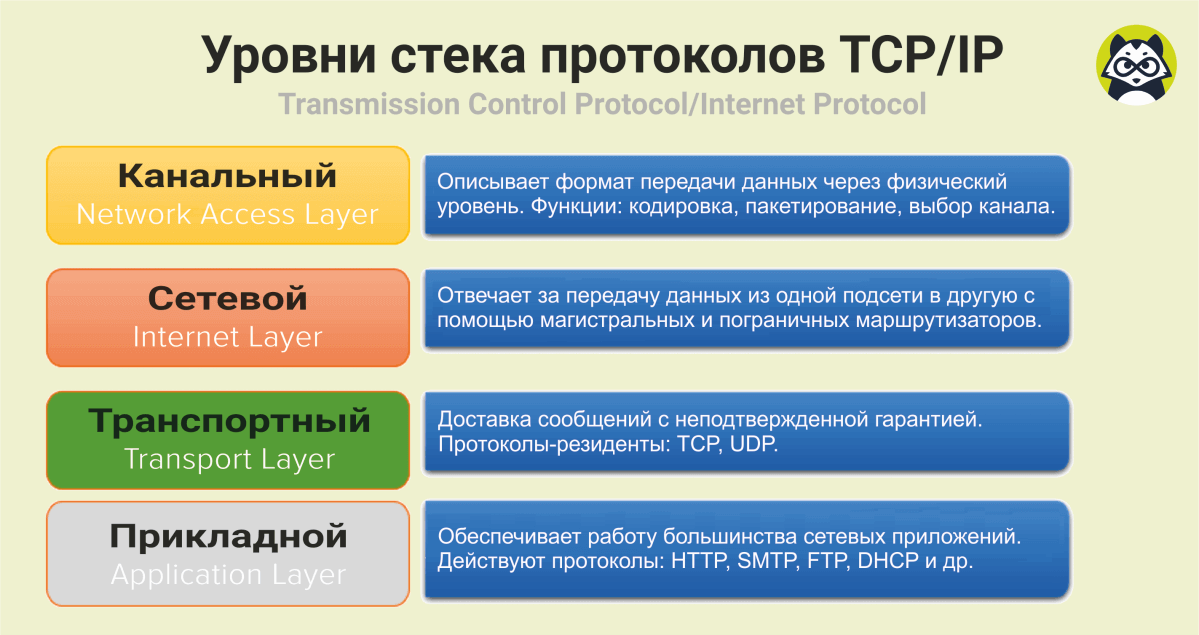
IP-адрес в классовой архитектуре сетевой адресации  состоит из двух частей:

Идентификатор сети. Определяет сеть, содержащую подключённые узлы.

Идентификатор узла. Отвечает за обозначение узла – сервера, маршрутизатора или любого другого **TCP/IP-**устройства.

**TCP/IP**

Любая сеть с IP-адресацией построена на основе TCP/IP – модели, включающей в себя стек протоколов, применяемых при передаче данных по сети. Основными протоколами являются TCP и IP, но имеется и масса других вариантов.



* **Канальный**. Отвечает за физическую передачу данных посредством использования таких протоколов, как Ethernet или WI-FI.
* **Сетевой (Интернет)**. На этом уровне находится система IP-адресов, и осуществляется [маршрутизация](https://eternalhost.net/base/vps-vds/tablitsa-marshrutizatsii-linux) – перемещение пакетов между устройствами. Сетевой уровень совмещает протоколы: IP, ICMP, IGMP.
* **Транспортный**. Здесь расположены протоколы TCP и UDP, отвечающие за передачу данных. Первый осуществляет гарантированное перемещение информации, предварительно устанавливая соединение с сетью. Второй же отправляет сообщения без осуществления «рукопожатия», что повышает скорость передачи данных, но также создаёт риск потери отдельных пакетов.
* **Прикладной**. Совмещает все высокоуровневые протоколы, взаимодействующие с системными приложениями. К таким относятся Telnet, FTP, SMTP, SNMP и подобные.
* Уникальные IP-адреса, которые назначаются специальными организациями (например, Интернет-провайдером), называются **внешними, белыми**или **публичными**. Публичные IP-адреса применяются для получения доступа к Интернету и осуществления взаимодействия с другими узлами через публичную сеть. Устройство с внешним IP-адресом видно другим пользователям в Интернете.
* Кроме того, существуют**частные**IP-адреса, именуемые также **серыми** или **внутренними**. Серые IP-адреса назначаются устройствам в локальной сети и не видны в Интернете. К примеру, можно представить дом, в котором к WI-FI роутеру подключено несколько устройств. Все они объединены в одну сеть и имеют серые IP-адреса.

|  |  |
| --- | --- |
| **Публичные IP-адреса** | **Частные IP-адреса** |
| Глобальный (внешний) охват. | Местный (внутренний) охват. |
| Используются для соединений через Интернет за пределами частной сети. | Используется для связи с другими устройствами в частной сети. |
| Уникальный числовой код, не используемый другими устройствами. | Неуникальный числовой код, который может использоваться другими устройствами в других частных сетях. |
| Можно узнать по поисковому запросу типа: «Мой IP-адрес» («What is my IP»). | Можно найти во внутренних настройках устройства. |
| Назначаются интернет-провайдером. | Присваиваются маршрутизатором конкретному устройству. |
| Платные. | Бесплатные. |
| Может использоваться любое число, не входящее в диапазон частных IP-адресов. | 10.0.0.0 — 10.255.255.255 172.16.0.0 — 172.31.255.255 192.168.0.0 — 192.168.255.255 |
| Пример: 8.8.8.8. | Пример: 10.11.12.13 |

**Присвоение IP-адресов**

**Динамическое назначение**

При подключении к сети через протокол динамической настройки узла (DHCP / Dynamic Host Configuration Protocol) все параметры стека TCP/IP автоматически устанавливаются на устройстве. Узлу назначается динамический IP-адрес, который меняется на другой при переподключении устройства. Диапазон IP-адресов указывается на сервере DHCP.

**Статическое назначение**

Статический IP-адрес присваивается вручную и не изменяется при переподключении к сети. Этот тип присваивания используется на устройствах, доступ к которым должен производится по одному адресу (например, на серверах).

**(IP) IPv4**. В сентябре 1981 года появился первый стандарт интернет-протокола (IP) IPv4, который положил начало современной сети Интернет. Ipv4 IP-адрес имеет вид: 192.168.50.1*.*

**IP-адрес IPv6** Интернет с 1980-х годов начал стремительно расти, поэтому появилась угроза истощения пула возможных адресов – их просто не хватило бы на все сети и узлы. Поэтому в 1995 году появился формат IPv6, при котором длина IP-адреса возросла с 32 до 128 бит, а десятичная система сменилась шестнадцатеричной.

IP-адрес IPv6 состоит из 16 октетов (8 блоков по 2 октета), раздёленных двоеточиями. В полном виде запись IPv6 выглядит следующим образом: 2001:0bd7:0ccf:0006:0000:0000:012f:002d.

**DNS и IP-адрес**

Путешествуя по Интернету, пользователь устанавливает соединение через браузер с другими серверами в основном не по IP-адресу, а с помощью [доменного имени](https://eternalhost.net/blog/domeny/chto-takoe-domen). **Система доменных имён (DNS)** служит для перенаправления на постоянный IP-адрес конечного веб-ресурса. Говоря простыми словами, она преобразовывает буквенные значения доменного имени в цифры IP-адреса.

Например, чтобы попасть на сайт поисковика Google, не нужно вводить сложный в запоминании числовой адрес «74.125.131.100». Достаточно набрать в адресной строке доменное имя «.google.com».

За осуществление подобной переадресации отвечает [DNS-сервер](https://eternalhost.net/base/domeny/chto-takoe-dns-server), который работает согласно информации из [DNS-записей](https://eternalhost.net/base/web-hosting/tipy-dns-zapisey). Продолжая «телефонную» аналогию можно сказать, что если IP-адрес — это номер телефона, то сервер DNS — это телефонная книга, содержащая все подобные номера.

***14) Что такое октаты в DNS***

***15) Что такое порт, сколько портов у Linux сервера***

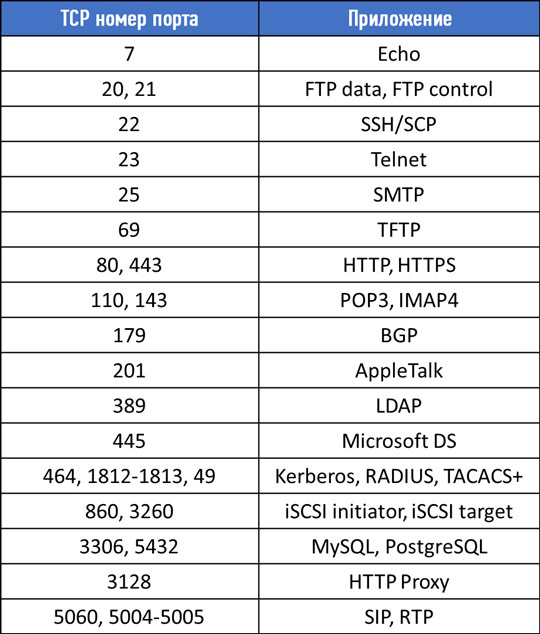
Порт (port) — это число в десятичной системе счисления от 0 до 65 535, которое записывается в заголовках протоколов транспортного уровня модели OSI, это те же, [TCP](https://anisim.org/articles/tcp-protokol/), [UDP](https://anisim.org/articles/udp-protokol/) и т.д. Применяется для идентификации программы или процесса для обмена пакетами данных в рамках одного IP-адреса. Т.е. по сути — айпи это уникальный адрес узла (устройства), а порт — это идентификатор его приложений или процессов, выходящих в сеть.

Просто идентификатор, чтобы определить получателя.  
Пакет идет до IP адреса вашего компьютера.  
А куда дальше? Кто будет этот пакет обрабатывать, кому он предназначен? Может это часть видефильма который вы смотрите, может это пакет сообщения для мессенджера, может это обновление для виндовс.  
Вот чтобы определить кому именно на компьютере адресован пакет, используют порт.  
Программы договариваются что будут пересылать данные друг для друга с определенной пометкой (номер порта).  
  
В итоге когда пакет приходит компьютер по номеру порта отдает его соответствующей программе которая подписана на этот порт.

IP адрес — это номер квартиры друга.  
Порт — это комната, в которой живёт друг

Теперь более понятным языком. Как вам известно еще из статьи про [TCP/IP протокол](https://anisim.org/articles/tcp-ip-protokol-chto-eto-takoe/) — каждое устройство, которое подключается к сети имеет свой собственный и уникальный идентификатор — айпи адрес. Все программы и процессы, которые работают на этом устройстве используют этот уникальный IP. И как они будут идентифицировать друг друга при выходе в сеть? Таким образом, каждая программа или процесс использует свой порт, так они различают друг друга в сети.

Существует ряд стандартных портов, которые используются по умолчанию. Их зарегистрировала организация IANA (Internet Assigned Numbers Authority). Именно IANA ответственна за ресурсы интернет-протоколов. Всего было зарезервировано 1 024 значения, от 0 до 1 023. Это значительно упростило маршрутизацию в интернете, т.к. они используются приложениями по умолчанию. Вот их небольшой список:



Когда вы устанавливаете соединение с каким-либо сервером, программой или другим компьютером — генерируется случайный PORT для соединения и вашей идентификации. В дальнейшем обмен пакетами данных будет происходить именно по нему.

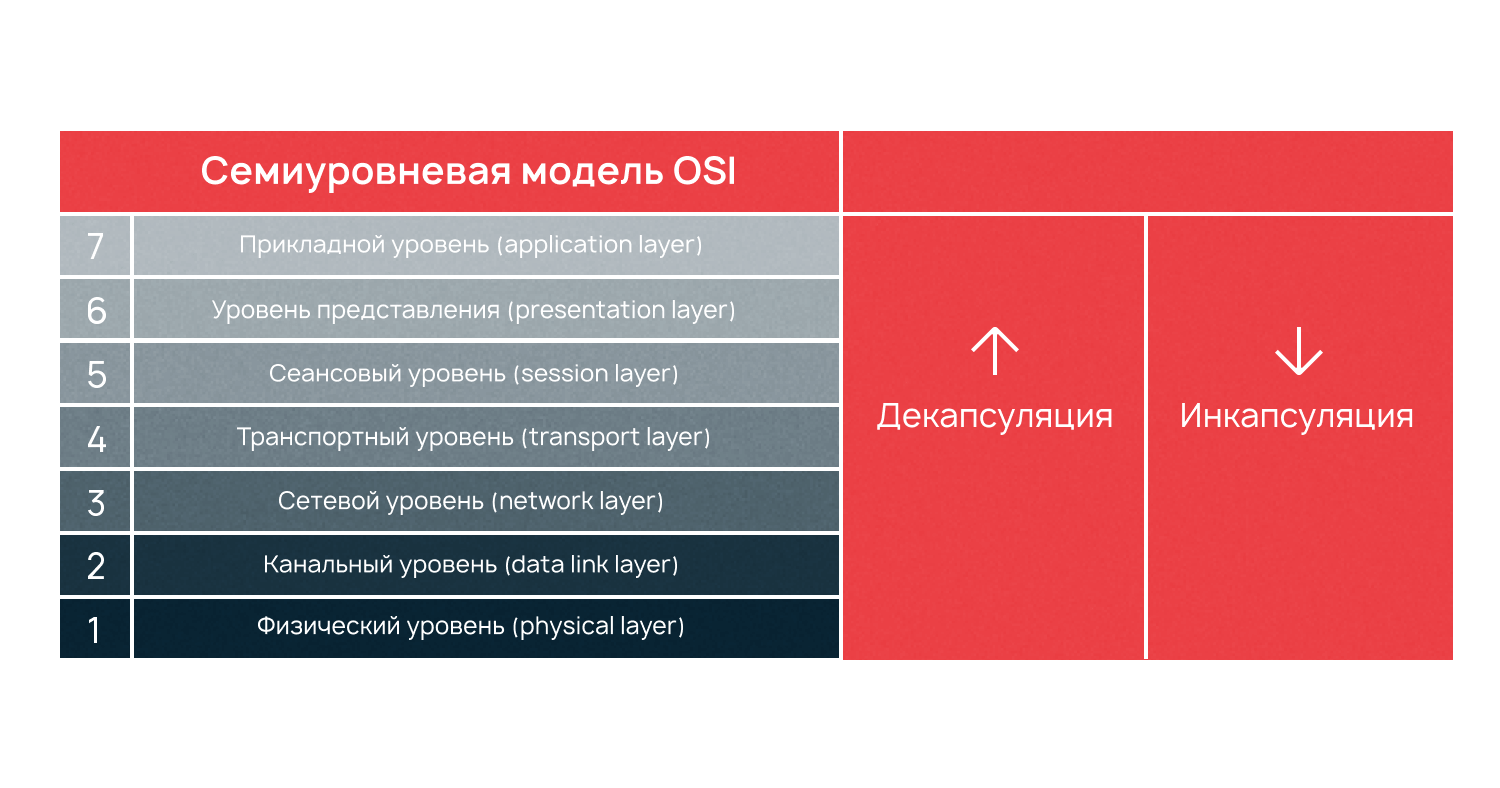
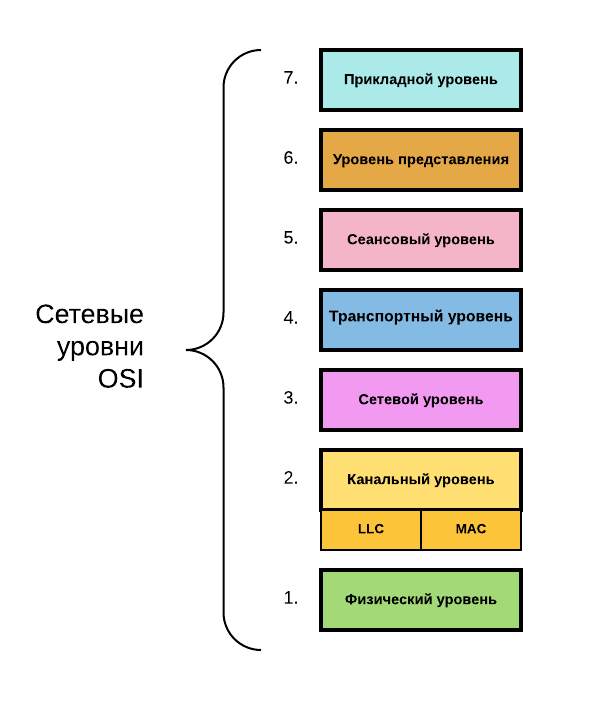
Для примера возьмем интернет-сервер со своим уникальным IP-адресом. На нем есть: веб-сервер, почтовый сервер, FTP-сервер. Если нам потребуется соединится с этим сервером, чтобы отправить почту просто по ИП адресу, то, как он поймет, что мы хотим почту? А вот PORT нам в этом, как раз и поможет.

Так, чтобы соединиться с каким-либо сервером, нам надо знать не только его [айпи](https://anisim.org/articles/ip-adres-chto-eto-takoe/), но и номер порта. И в то же время, чтобы кто-либо извне смог подключиться к нам, к приложению на нашем ПК для получения данных, он должен знать такие же данные. Благо все это происходит автоматически и нам это делать вручную не надо.

***16) Уровни OSI***

Сетевая модель OSI (или Open System Interconnection)  — сетевая модель стека сетевых протоколов OSI/ISO. Посредством данной модели различные сетевые устройства могут взаимодействовать друг с другом. Модель определяет различные уровни взаимодействия систем. Каждый уровень выполняет определённые функции при таком взаимодействии.

Модель OSI (или Open System Interconnection) характеризует взаимодействие сетевого оборудования. Иными словами, посредством неё общаются такие устройства, как ПК с сетевыми картами, коммутаторы, роутеры.



**Первый, физический уровень (physical layer, L1)**

Начнем с самого нижнего уровня. Он отвечает за обмен физическими сигналами между физическими устройствами, «железом».

железу картинка понятна только в виде набора нулей и единиц, то есть бит. В данном случае бит является блоком данных протокола, сокращенно PDU (Protocol Data Unit).

Каждый уровень имеет свои PDU, представляемые в той форме, которая будет понятна на данном уровне и, возможно, на следующем до преобразования.

Устройства физического уровня оперируют битами. Они передаются по проводам (например, через оптоволокно) или без проводов (например, через Bluetooth или IRDA, Wi-Fi, GSM, 4G и так далее). На текущем уровне речь идёт о кабелях, радиоэфире, кодировании нулей и единиц и др. Сигнал первого уровня – это группа напряжений различной амплитуды, волн либо радиочастот.

**Второй уровень, канальный (data link layer, L2)**

Второй уровень решает проблему адресации при передаче информации. Канальный уровень получает биты и превращает их в кадры (frame, также «фреймы»). Задача здесь — сформировать кадры с адресом отправителя и получателя, после чего отправить их по сети.

У канального уровня есть два подуровня — это MAC и LLC. MAC (Media Access Control, контроль доступа к среде) отвечает за присвоение физических MAC-адресов, а LLC (Logical Link Control, контроль логической связи) занимается проверкой и исправлением данных, управляет их передачей.

На втором уровне OSI работают коммутаторы, их задача — передать сформированные кадры от одного устройства к другому, используя в качестве адресов только физические MAC-адреса.

**Третий уровень, сетевой (network layer, L3)**

На третьем уровне появляется новое понятие — маршрутизация. Для этой задачи были созданы устройства третьего уровня — маршрутизаторы (их еще называют роутерами). Маршрутизаторы получают MAC-адрес от коммутаторов с предыдущего уровня и занимаются построением маршрута от одного устройства к другому с учетом всех потенциальных неполадок в сети.

На сетевом уровне активно используется протокол ARP (Address Resolution Protocol — протокол определения адреса). С помощью него 64-битные MAC-адреса преобразуются в 32-битные IP-адреса и наоборот, тем самым обеспечивается инкапсуляция и декапсуляция данных.

Здесь осуществляется маршрутизация трафика. Когда пользователь, к примеру, желает перейти на сайт и вводит его адрес, отправляется DNS-запрос. Ответом на него будет IP-адрес, который подставляется в пакет. Пакет данных – это новый термин, который появляется на 3-м сетевом уровне.

Устройствами здесь являются роутер или маршрутизатор.

**Четвертый уровень, транспортный (transport layer, L4)**

Уровни группы Media Layers (L1, L2, L3) занимаются передачей информации (по кабелю или беспроводной сети), используются сетевыми устройствами, такими как коммутаторы, маршрутизаторы и т.п. Уровни группы Host Layers (L4, L5, L6, L7) используются непосредственно на устройствах, будь то стационарные компьютеры или портативные мобильные устройства.

Здесь происходит доставка информации по каналам внешней сети. Блоки данных в данном случае делятся на отдельные фрагменты, размеры которых будут зависеть от используемого протокола. Для транспортного уровня это TCP и UDP. Какой из них лучше использовать, зависит от типа передаваемых данных.

TCP (Transmission Control Protocol) – протокол, хорошо подходящий для передачи трафика, для которого любые потери пакетов чувствительны. Процесс передачи контролируется, благодаря чему потерянные пакеты будут обнаружены и запрошены повторно. UDP (User Datagram Protocol) применяется, когда потеря нескольких пакетов не принципиальна. Например, при передаче видео, изображений.

**Пятый уровень, сеансовый (session layer, L5)**

Сеансовый уровень в ответе за организацию сеансов связи между приложениями на компьютерах. Он отвечает за создание и окончание сеанса, обмен данными, за синхронизацию и другие процессы.

 Пятый уровень оказывает услугу следующему: управляет взаимодействием между приложениями, открывает возможности синхронизации задач, завершения сеанса, обмена информации.

Примером работы пятого уровня может служить видеозвонок по сети. Во время видеосвязи необходимо, чтобы два потока данных (аудио и видео) шли синхронно. Когда к разговору двоих человек прибавится третий — получится уже конференция. Задача пятого уровня — сделать так, чтобы собеседники могли понять, кто сейчас говорит.

**Шестой уровень, представления данных (presentation layer, L6)**

На шестой ступени осуществляется преобразование форматов данных, например, сжатие и кодирование. Шестой уровень занимается тем, что представляет данные (которые все еще являются PDU) в понятном человеку и машине виде. Например, когда одно устройство умеет отображать текст только в кодировке ASCII, а другое только в UTF-8, перевод текста из одной кодировки в другую происходит на шестом уровне.

Шестой уровень также занимается представлением картинок (в JPEG, GIF и т.д.), а также видео-аудио (в MPEG, QuickTime). Помимо перечисленного, шестой уровень занимается шифрованием данных, когда при передаче их необходимо защитить.

**Седьмой уровень, прикладной (application layer)**

Прикладной уровень — это то, с чем взаимодействуют пользователи, своего рода графический интерфейс всей модели OSI, с другими он взаимодействует по минимуму. Все услуги, получаемые седьмым уровнем от других, используются для доставки данных до пользователя. Протоколам седьмого уровня не требуется обеспечивать маршрутизацию или гарантировать доставку данных, когда об этом уже позаботились предыдущие шесть. Задача седьмого уровня — использовать свои протоколы, чтобы пользователь увидел данные в понятном ему виде.

Протоколы здесь используют UDP (например, DHCP) или TCP (например, HTTP, HTTPS, SFTP (Simple FTP), DNS). Сюда относятся протоколы для просмотра страниц в интернете (HTTPS, HTTP), для работы с почтовыми службами (SMTP, POP3), для передачи файлов (FTP, TFTP) и другие.

***17) Хедеры http запросов***

Заголовки HTTP являются основной частью HTTP-запросов и ответов, и они содержат информацию о клиенте, закрытой странице, сервере и многих других.

Заголовки [HTTP](https://ru.wikipedia.org/wiki/HTTP) ([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) HTTP Headers) — это строки в HTTP-сообщении, содержащие разделённую двоеточием пару имя-значение. Формат заголовков соответствует общему формату заголовков текстовых сетевых сообщений [ARPA](https://ru.wikipedia.org/wiki/DARPA) (см. [RFC 822](https://tools.ietf.org/html/rfc822)). Заголовки должны отделяться от тела сообщения хотя бы одной пустой строкой.

**Все заголовки разделяются на четыре основных группы:**

1. [General Headers](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%B7%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BA%D0%BE%D0%B2_HTTP#%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%8B%D0%B5_%D0%B7%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BA%D0%B8) ([рус.](https://ru.wikipedia.org/wiki/%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Основные заголовки) — должны включаться в любое сообщение клиента и сервера.
2. [Request Headers](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%B7%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BA%D0%BE%D0%B2_HTTP#%D0%97%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BA%D0%B8_%D0%B7%D0%B0%D0%BF%D1%80%D0%BE%D1%81%D0%B0) ([рус.](https://ru.wikipedia.org/wiki/%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Заголовки запроса) — используются только в запросах клиента.
3. [Response Headers](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%B7%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BA%D0%BE%D0%B2_HTTP#%D0%97%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BA%D0%B8_%D0%BE%D1%82%D0%B2%D0%B5%D1%82%D0%B0) ([рус.](https://ru.wikipedia.org/wiki/%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Заголовки ответа) — только для ответов от сервера.
4. [Entity Headers](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%B7%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BA%D0%BE%D0%B2_HTTP#%D0%97%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BA%D0%B8_%D1%81%D1%83%D1%89%D0%BD%D0%BE%D1%81%D1%82%D0%B8) ([рус.](https://ru.wikipedia.org/wiki/%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Заголовки сущности) — сопровождают каждую [сущность](https://ru.wikipedia.org/wiki/%D0%A1%D1%83%D1%89%D0%BD%D0%BE%D1%81%D1%82%D1%8C_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) сообщения.

Именно в таком порядке рекомендуется посылать заголовки получателю.

Заголовки запроса, такие как Accept , Accept-\* или If-\* позволяют выполнять условные запросы; другие, такие как Cookie (en-US), User-Agent или Referer уточняют контекст, чтобы сервер мог адаптировать ответ. Не все заголовки, появляющиеся в запросе, являются заголовками запроса.

Header (хидер или хедер, шапка сайта) – это блок в верхней части сайта. Как правило, хедер и футер (подвал) выполнены в едином дизайне и содержат одну и ту же информацию на всех страницах ресурса.