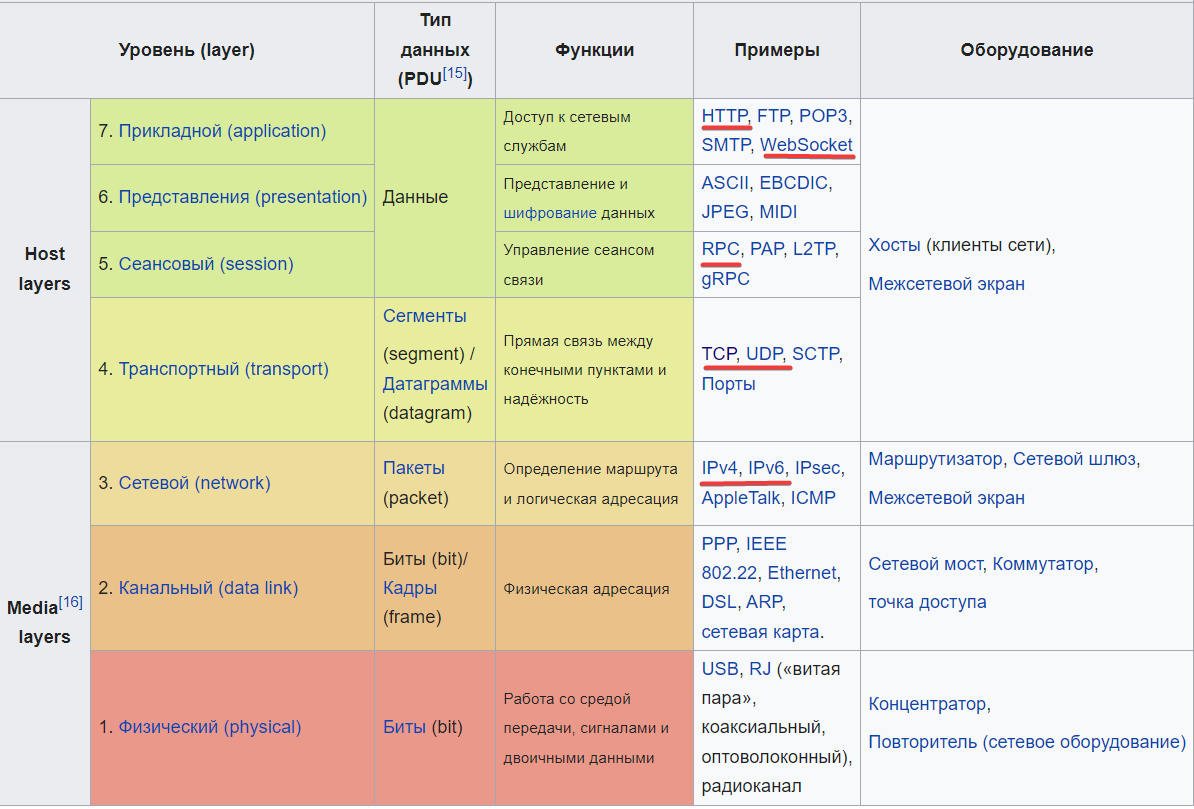
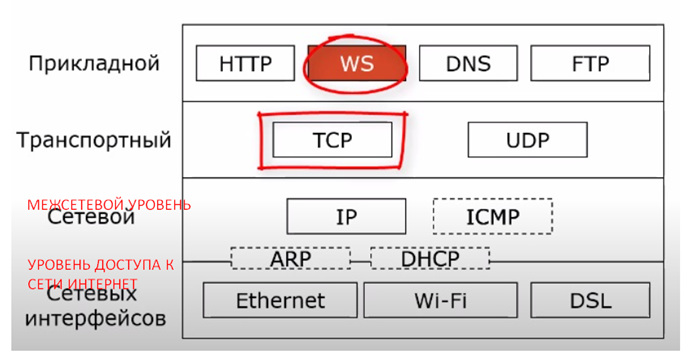
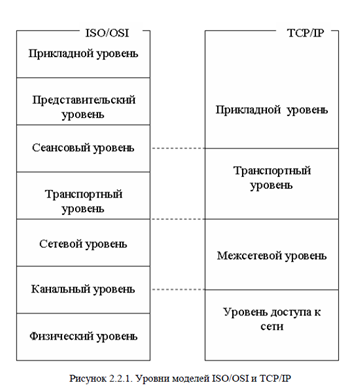
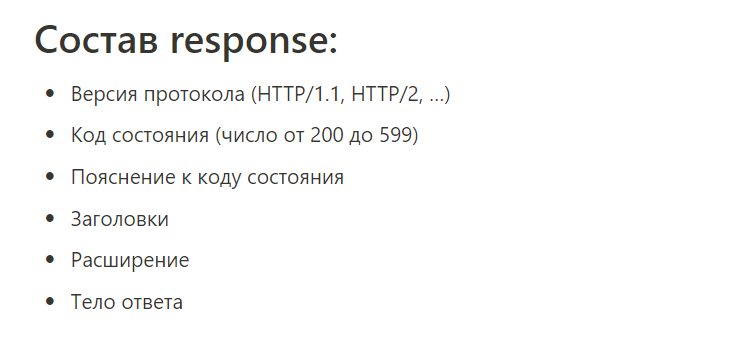
   .0

SMELOV DEBIL

1. **Сервер** – программное обеспечение, которое принимает запросы.
2. Интернет – всемирная сеть, построенная на стеке протоколов TCP/IP.
3. **Интернет** – 4 компонента:
   1. Стек протоколов TCP-IP — это основа Интернета.
   2. Интернет службы (DNS, DHCP, SMTP, POP3 и т.д.).
   3. Документация в формате RFC и STD.
   4. Система организаций, которые поддерживают сеть:
      * IETF (разрабатывает стандарты интернета в RFC)
      * ICANN (распространяет доменные имена)
      * IANA (регистрирует MIME)
      * ISOC (развитие, внедрение и распространение новых интернет-технологий)
      * W3C (WWW Consortium, некоммерческая организация для согласования стандартов) — согласовывают *всё*, что касается веб-программирования (HTTP, CSS, SVG, URI/URL, XML, PNG, JPEG, …). Также они изготавливают стандарты и регистрируют их в IETF.
4. **Интернет-служба** – это программа, система, предоставляющая услуги клиентам. Сервер + протокол. Стандартные серверы, которые прослушивают стандартные порты (*от 0 до 1024*).
5. **Веб-приложение** – клиент-серверное приложение, в котором клиент и сервер взаимодействуют с помощью протокола HTTP.
6. **HTTP** – *полудуплексный* протокол *прикладного* уровня.
7. Версии HTTP:
   1. 0.9 (methods: GET only, hypertext only, no HTTP headers (cannot transfer other content type files), no status/error codes, no URLs, no versioning)
   2. 1.0 (недействующий, нет постоянного соединения, methods: GET , HEAD , POST , имеет `Content-Type`)
   3. 1.1 (действующий, имеет постоянное соединение, has `Keep-alive`)
   4. 2.0 (бинарный, не распространен)
   5. 3.0 (не распространен)

For geeks: https://medium.com/platform-engineer/evolution-of-http-69cfe6531ba0

1. Два типа сообщений – request и response.
2. Два типа абонентов – клиент и сервер.  
   (кстати, клиент всегда является инициатором связи)
3. HTTPS (HTTP + SSL(TSL)) – специальный протокол, обеспечивающий шифрование и аутентификацию.   
   SSL – Secure Socket Layer, был заменен в 2014 на TSL   
   TSL – Transport Secure Layer
4. HTTP – порт *80*,  
    HTTPS – порт *443*.
5. Интернет-ресурс – *сущности*, которые имеют адрес в сети Интернет.  
   Бывают:  
    1) статические (html)  
    2) динамические (js)   
    3) полудинамические (html + js)  
    (адрес = IP + порт. например, 192.168.0.1:8000).  
    socket  = IP + port (example: 192.168.0.1:8000)
6. Виды каналов связи:
   1. симплексный – передача данных только в одну сторону (радио)
   2. полудуплексный – обе стороны могут передавать и принимать данные, но не одновременно. Когда один говорит, второй только слушает (например, рация)
   3. дуплексный – данные передаются и принимаются в обе стороны одновременно (e.g. телефон)
7. 
8. 
9. **Типы заголовков:**
   1. General (общие для запросов и ответов)
   2. Request
   3. Response
   4. Entity (для сущностей в запросах и ответах)
   5. User (пользовательский – тот, который не описан в протоколе HTTP)
10. **Уровни TCP/IP:**  
     1) Прикладной   
     2) Транспортный   
     3) Межсетевой   
     4) Уровень доступа к сети
11. **Уровни OSI:** 1) Прикладной  
     2) Представительский  
     3) Сеансовый  
     4) Транспортный  
     5) Сетевой  
     6) Канальный  
     7) Физический
12. **Протокол** – набор согласованных правил, которые устанавливают порядок установки соединения и обмена данными между пользователями сети Интернет.
13. Важные для нас протоколы каждого уровня:
    1. Прикладной: HTTP, WebSocket, DNS, DHCP
    2. Транспортный: TCP, UDP
    3. Сетевой: IP
14. Stateless протокол – который *не сохраняет* состояние между  
     запросом и ответом (например, HTTP или UDP).
15. Stateful протокол – который *сохраняет* состояние между запросом и  
     ответом (например, TCP).
16. Как происходит адресация на канальном уровне?    
       С помощью MAC-адреса (48-бит).
17. **WebSocket** – протокол *прикладного* уровня, основанный на TCP,   
     использующий *дуплексный* канал связи.
18. **Как устанавливается соединение WebSocket?**
    1. Клиент посылает обычный HTTP-запрос, называемый *рукопожатием*, с заголовками:  
           Connection: Upgrade  
           Upgrade: Websocket
    2. Сервер решает, устанавливать WebSocket-соединение или нет. Если да, то сервер пересылает те же заголовки с кодом ответа 101 Switching Protocols.
    3. HTTP-соединение заменяется на WebSocket-соединение, данные можно передавать.
19. Что значит заголовок Content-Type?  
          Это MIME-тип ресурса в request body или response body.
20. Какие методы запросов можно отправлять через тег <form>?  
          Только GET и POST.
21. **Как пересылаются данные через форму?***В виде пар ключ:значение.*  
    Дополнительная инфа:   
    У тега <form> задается атрибут enctype, равный либо   
     x-www-form-urlencoded, либо multipart/form-data, либо text/plain.
    1. *x-www-form-urlencoded:*Данные похожи на то, как кодируются параметры в URL. Отправляются, как пары ключ-значение. Эти пары разделяются через «&», пробелы заменяются на «+», а русские символы заменяются их шестнадцатеричным представлением (%D0%90%D0%BD%D1%8F). Ключи берутся из атрибута name, а значения – из атрибута value в теге input. (<input name=”someKey” value=”someValue”>).   
       Пример, как это выглядит на практике:  
       text1=hello+world&text2=a%CF%89b&file=a.txt
    2. *multipart/form-data:*  
       Используется в основном для пересылки файлов через форму. Данные отправляются в теле запроса и разделяются некоторой границей – boundary.
    3. *text/plain:*Не используется.
22. Узел Интернет – это устройство, имеющее IP-адрес и подключенное к сети Интернет (как правило, к сети Интернет-провайдера).
23. **Session (сессия, сеанс)** – серверный объект, хранящийся на стороне сервера и имеющий идентификатор Session ID (8-битное число).   
    Задача сессии – *идентифицировать серию запросов.*   
    Необходима для последовательных запросов, хранения промежуточных данных, например, чтобы не спрашивать у клиента логин и пароль при каждом запросе.
24. **Когда и как сессия создается и завершается?**
    1. Сервер создает сессию, когда на него приходит первый request:
       * без заголовка Cookie;
       * в заголовке Cookie не записан Session ID;
       * в заголовке Cookie записан неправильный или несуществующий Session ID.
    2. Сервер создаёт объект сессии, генерирует Session ID, и отправляет клиенту заголовок *Set-Cookie* со значением SessionID;
    3. Клиент, получивший заголовок *Set-Cookie*, обязан запомнить значение Session ID;
    4. Во все последующие запросы клиент будет добавлять заголовок *Cookie* со значением Session ID;
    5. Сам сервер никогда не отправляет заголовок *Cookie*, он только обрабатывает их от клиента.
25. Session Timeout – это максимальное время между двумя запросами.   
    Если время превышено, то сессия будет уничтожена, и будет создана новая сессия.
26. **Три уровня памяти:**
    1. Запрос — хранится в рамках одного запроса (пока не пришел ответ);
    2. Сессия — существует в рамках одной сессии;
    3. Контекст — в рамках контекста; может существовать условно бесконечно (пока работает сервер), можно передавать данные между сессиями. *Создается на основании config-файла.*
27. Контекст – создается на основе config-файла при старте приложения
28. Кроссплатформенное приложение – это приложение, способное работать на более чем одной программно-аппаратной (аппаратура + операционная система) платформе.
29. **Класс** – шаблон создания объекта.
30. **Объект** – специализированный объект времени исполнения.   
        Также объект – это экземпляр класса.
31. **Интерфейс** – поименованный набор сигнатур*.*  
    (У смелова понятие “поименованный набор *методов*”, но в интерфейсе могут быть методы, поля, свойства и т.д., поэтому мы говорим “поименованный набор сигнатур”)
32. **Платформа** – совокупность инструментов и библиотек   
       (ИЛИ совокупность инструментов и среды исполнения)
33. Рефлексия – выявления типа во время RunTime, метаданные. Применяется в Router для маппинга маршрутов и handler методом рефлексии.
34. AJAX – методология программирования, построенная на асинхронных запросах. Эта методология подразумевает такой подход к построению динамических приложений, при которых не осуществляется полная перезагрузка html-страниц.
35. **CLI** **(Common Language Infrastructure)** – спецификация общеязыковой инфраструктуры. Определяет архитектуру исполнительной системы и набор предоставляемых сервисов. Состоит из шести пунктов:
    1. Концепция  и архитектура:
       * CTS (Common Type System)
       * VES (Virtual Execution System)
       * CLS (Common Language Specification)
    2. Метаданные
    3. Инструкции CIL (Common Intermediate Language)
    4. Библиотеки:
       * BCL (Base Class Library)
    5. Формат взаимодействия с отладчиком
    6. Приложения
36. **CLR (Common Language Runtime)** – общеязыковая среда исполнения – виртуальная машина, на которой исполняются  все приложения, работающие в среде .NET. Является реализацией концепции VES от компании Microsoft. Содержит в себе JIT-компилятор(Just-In-Time).
37. *VES* (Virtual Execution System) — концепция виртуальной компиляции и запуска. VES – это только идея, концепция, спецификация, у которой есть конкретные реализации. Реализация VES на .NET – это CLR, на Java – это виртуальная Java-машина(JVM) и т.д.
38. *FCL* (Framework Class Library) — по аналогии, в то время как BCL – это просто библиотека стандартных (Basic) классов, то FCL – это конкретная реализация BCL для .NET Framework.   
    Можно рассматривать, как API CLR.
39. *MSIL* (Microsoft IL)  – реализация CIL (Common Intermediate Language) компанией Microsoft.
40. *NET Framework* – программная технология  программирования.  Это совокупность: CLR + FCL.
41. ASP.NET – набор  технологий для создания Web-приложений и Web-сервисов от компании Microsoft на основе платформы .NET Framework.
42. Типы приложений ASP.NET (т. е. какие проекты можно создать):
    1. HTTP-handler
    2. Web Form
    3. MVC
    4. MVC Web API
    5. ASMX-сервисы
    6. WCF-сервисы (отдельно).
43. Http-handler – это  простейшее приложение ASP.NET. Представлен классом, который реализует интерфейс *IHttpHandler* и имеет свойство *IsReusable* и метод *ProcessRequest(HttpContext httpContext)*. Внутри объекта HttpContext хранятся request, response и т.д.
44. IIS – Internet Information Server – набор серверов для Internet-служб компании Microsoft. Поддерживает протоколы: HTTP, HTTPS, FTP, SMTP, POP3.
45. Режимы публикации приложения:
    1. ручной
    2. автоматический (через VS Express или IIS)
46. Параметры при публикации IIS:
    1. имя сайта
    2. физический путь
    3. тип протокола, ip и port
47. Model-View-Controller – паттерн проектирования, в котором приложение состоит из 4 взаимодействующих компонентов: модель, представление, контроллер, роутер. Они относительно независимы.   
    Каждый компонент имеет свою зону ответственности:
    1. модель – данные;
    2. представление – динамическое формирование разметки для отправки клиенту;
    3. контроллер – обработка запроса, формирование экземпляра модели, вызов Razor Engine;
    4. роутер – маппинг URI.
48. **Какие требования должен соблюдать MVC контроллер?**  
         1) должен иметь в названии слово “Controller”  
         2) должен наследоваться от класса Controller (не обязательно)
49. Строго типизированные  представления – представления, которые привязаны к некоторой модели. В них используется ключевое слово @model в начале представления, которое говорит, к какой модели типизирован данное представление.
50. Способы передачи данных на View:  
         1) ViewData (глобальный key-value словарь, выглядит примерно   
              так: ViewData[“Foo”])   
         2) ViewBag (подобен ViewData, но использует свойства вроде   
             ViewBag.Foo)  
         3) ViewModel (специальный класс для передачи данных)  
         4) Передача модели через конструктор– для адекватных людей
51. **Что такое Redirect? Как он работает?**  
      Клиент отправляет запрос на сервер на доступ к ресурсу, который был перемещён. Сервер отправляет клиенту заголовок Location, в котором хранится URL, на который надо редиректнуться, а также один из статус-кодов: 301, 302, 307 или 308

https://developer.mozilla.org/en-US/docs/Web/HTTP/Redirections

1. Temporary Redirect – ***302 и 307***
2. Permanent Redirect – ***301 и 308***
3. Special Redirect - ***300, 304***
4. **В чем разница между 301, 302, 307 и 308 status codes?**  
    Во-первых, 301 и 302 считаются устаревшими. В спецификации HTTP/1.1 им на замену пришли аналогичные 307 и 308.   
    Вся разница в том, что 301 и 302 могли изменить метод запроса: например, делается запрос POST /api/users, возвращается 302 status code, и делается новый запрос GET /api/users. В 307 и 308 такой проблемы нет, и метод запроса всегда будет сохраняться при редиректе.
5. Где регистрируется таблица маршрутов?   
       В файле Global.asax.
6. По какому принципу Controller находит View?
   1. сначала ищет в Views/Controller\_name/Action\_name.cshtml
   2. потом в Views/Shared