# HTTP-запросы: структура, методы, строка статуса и коды состояния

**HTTP** (HyperText Transfer Protocol, дословно — «протокол передачи гипертекста») представляет собой протокол прикладного уровня, используемый для доступа к ресурсам Всемирной Паутины. Под термином *гипертекст* следует понимать текст, в понятном для человека представлении, при этом содержащий ссылки на другие ресурсы.

Данный протокол описывается спецификацией [RFC 2616](https://datatracker.ietf.org/doc/html/rfc2616). На сегодняшний день наиболее распространенной версией протокола является версия **HTTP/2**, однако нередко все еще можно встретить более раннюю версию **HTTP/1.1**.

В обмене информацией по HTTP-протоколу принимают участие клиент и сервер. Происходит это по следующей схеме:

* Клиент запрашивает у сервера некоторый ресурс.
* Сервер обрабатывает запрос и возвращает клиенту ресурс, который был запрошен.

По умолчанию для коммуникации по HTTP используется порт 80, хотя вместо него может быть выбран и любой другой порт. Многое зависит от конфигурации конкретного веб-сервера.

## HTTP-сообщения: запросы и ответы

Данные между клиентом и сервером в рамках работы протокола передаются с помощью HTTP-сообщений. Они бывают двух видов:

* **Запросы (HTTP Requests)** — сообщения, которые отправляются клиентом на сервер, чтобы вызвать выполнение некоторых действий. Зачастую для получения доступа к определенному ресурсу. Основой запроса является HTTP-заголовок.
* **Ответы (HTTP Responses)** — сообщения, которые сервер отправляет *в ответ* на клиентский запрос.

Само по себе сообщение представляет собой информацию в текстовом виде, записанную в несколько строчек.

В целом, как запросы HTTP, так и ответы имеют следующую структуру:

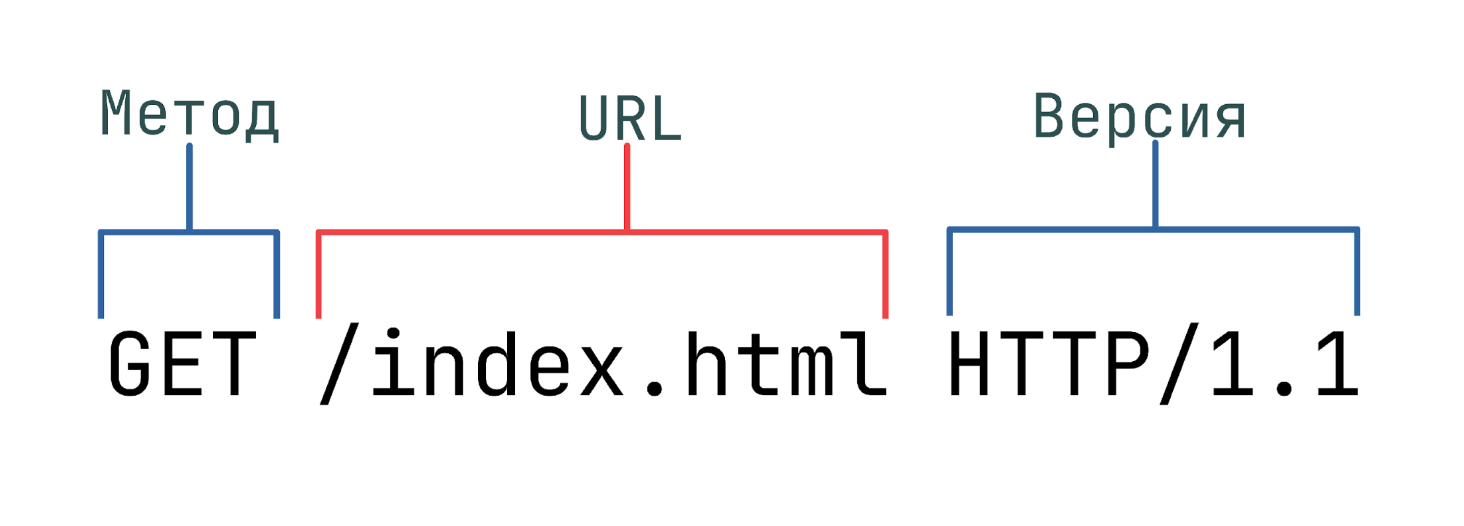
* *Стартовая строка (start line)* — используется для описания версии используемого протокола и другой информации — вроде запрашиваемого ресурса или кода ответа. Как можно понять из названия, ее содержимое занимает ровно одну строчку.
* *HTTP-заголовки (HTTP Headers)* — несколько строчек текста в определенном формате, которые либо уточняют запрос, либо описывают содержимое *тела* сообщения.
* Пустая строка, которая сообщает, что все метаданные для конкретного запроса или ответа были отправлены.
* Опциональное *тело сообщения*, которое содержит данные, связанные с запросом, либо документ (например HTML-страницу), передаваемый в ответе.

## Стартовая строка

Стартовая строка HTTP-запроса состоит из трех элементов:

* *Метод HTTP-запроса* (method, реже используется термин verb). Обычно это короткое слово на английском, которое указывает, что конкретно нужно сделать с запрашиваемым ресурсом. Например, метод GET сообщает серверу, что пользователь хочет получить некоторые данные, а POST — что некоторые данные должны быть помещены на сервер.
* Цель запроса. Представлена указателем ресурса URL, который состоит из протокола, доменного имени (или IP-адреса), пути к конкретному ресурсу на сервере. Дополнительно может содержать указание порта, несколько параметров HTTP-запроса и еще ряд опциональных элементов.
* Версия используемого протокола (либо HTTP/1.1, либо HTTP/2), которая определяет структуру следующих за стартовой строкой данных.

В примере ниже стартовая строка указывает, что в качестве метода используется GET, обращение будет произведено к ресурсу /index.html, по версии протокола HTTP/1.1:



### Методы

Методы позволяют указать конкретное действие, которое мы хотим, чтобы сервер выполнил, получив наш запрос. Так, некоторые методы позволяют браузеру (который в большинстве случаев является источником запросов от клиента) отправлять дополнительную информацию в теле запроса — например, заполненную форму или документ.

Ниже приведены наиболее используемые методы и их описание:



Разберемся с каждым из названных элементов подробнее.

|  |  |
| --- | --- |
| **Метод** | **Описание** |
| **GET** | Позволяет запросить некоторый конкретный ресурс. Дополнительные данные могут быть переданы через строку запроса (Query String) в составе URL (например ?param=value).О составляющих URL мы поговорим чуть позже. |
| **POST** | Позволяет отправить данные на сервер. Поддерживает отправку различных типов файлов, среди которых текст, PDF-документы и другие типы данных в двоичном виде. Обычно метод POST используется при отправке информации (например, заполненной формы логина) и загрузке данных на веб-сайт, таких как изображения и документы. |
| **HEAD** | Здесь придется забежать немного вперед и сказать, что обычно сервер в ответ на запрос возвращает заголовок и тело, в котором содержится запрашиваемый ресурс. Данный метод при использовании его в запросе позволит получить только заголовки, которые сервер бы вернул при получении GET-запроса к тому же ресурсу. Запрос с использованием данного метода обычно производится для того, чтобы узнать размер запрашиваемого ресурса перед его загрузкой. |
| **PUT** | Используется для создания (размещения) новых ресурсов на сервере. Если на сервере данный метод разрешен без надлежащего контроля, то это может привести к серьезным проблемам безопасности. |
| **DELETE** | Позволяет удалить существующие ресурсы на сервере. Если использование данного метода настроено некорректно, то это может привести к атаке типа «Отказ в обслуживании» (Denial of Service, DoS) из-за удаления критически важных файлов сервера. |
| **OPTIONS** | Позволяет запросить информацию о сервере, в том числе информацию о допускаемых к использованию на сервере HTTP-методов. |
| **PATCH** | Позволяет внести частичные изменения в указанный ресурс по указанному расположению. |

### URL

Получение доступа к ресурсам по HTTP-протоколу осуществляется с помощью указателя URL (Uniform Resource Locator). URL представляет собой строку, которая позволяет указать запрашиваемый ресурс и еще ряд параметров.

Использование URL неразрывно связано с другими элементами протокола, поэтому далее мы рассмотрим его основные компоненты и строение:

Поле **Scheme** используется для указания используемого протокола, всегда сопровождается двоеточием и двумя косыми чертами (://).

**Host** указывает местоположение ресурса, в нем может быть как доменное имя, так и IP-адрес.

**Port**, как можно догадаться, позволяет указать номер порта, по которому следует обратиться к серверу. Оно начинается с двоеточия (:), за которым следует номер порта. При отсутствии данного элемента номер порта будет выбран по умолчанию в соответствии с указанным значением **Scheme** (например, для http:// это будет порт 80).

Далее следует поле **Path**. Оно указывает на ресурс, к которому производится обращение. Если данное поле не указано, то сервер в большинстве случаев вернет указатель по умолчанию (например index.html).

Поле **Query String** начинается со знака вопроса (?), за которым следует пара «параметр-значение», между которыми расположен символ равно (=). В поле Query String могут быть переданы несколько параметров с помощью символа амперсанд (&) в качестве разделителя.

Не все компоненты необходимы для доступа к ресурсу. Обязательно следует указать только поля **Scheme** и **Host**.

### Версии HTTP

Раз уж мы упомянули версию протокола как элемента стартовой строки, то стоит сказать об основных отличиях версий HTTP/1.X от HTTP/2.X.

Последняя стабильная, наиболее стандартизированная версия протокола первого поколения (версия HTTP/1.1) вышла в далеком 1997 году. Годы шли, веб-страницы становились сложнее, некоторые из них даже стали приложениями в том виде, в котором мы понимаем их сейчас. Кроме того, объем медиафайлов и скриптов, которые добавляли интерактивность страницам, рос. Это, в свою очередь, создавало перегрузки в работе протокола версии HTTP/1.1.

Стало очевидно, что у HTTP/1.1 есть ряд значительных недостатков:

* Заголовки, в отличие от тела сообщения, передавались в несжатом виде.
* Часто большая часть заголовков в сообщениях совпадала, но они продолжали передаваться по сети.
* Отсутствовала возможность так называемого мультиплексирования — механизма, позволяющего объединить несколько соединений в один поток данных. Приходилось открывать несколько соединений на сервере для обработки входящих запросов.

С выходом HTTP/2 было предложено следующее решение: HTTP/1.X-сообщения разбивались на так называемые *фреймы*, которые встраивались в поток данных.

Фреймы данных (тела сообщения) отделялись от фреймов заголовка, что позволило применять сжатие. Вместе с появлением потоков появился и ранее описанный механизм мультиплексирования — теперь можно было обойтись одним соединением для нескольких потоков.

Единственное о чем стоит сказать в завершение темы: HTTP/2 перестал быть текстовым протоколом, а стал работать с «сырой» двоичной формой данных. Это ограничивает чтение и создание HTTP-сообщений «вручную». Однако такова цена за возможность реализации более совершенной оптимизации и повышения производительности.

### Заголовки

**HTTP-заголовок** представляет собой строку формата «Имя-Заголовок:Значение», с двоеточием(:) в качестве разделителя. Название заголовка не учитывает регистр, то есть между Host и host, с точки зрения HTTP, нет никакой разницы. Однако в названиях заголовков принято начинать каждое новое слово с заглавной буквы. Структура значения зависит от конкретного заголовка. Несмотря на то, что заголовок вместе со значениями может быть достаточно длинным, занимает он всего одну строчку.

В запросах может передаваться большое число различных заголовков, но все их можно разделить на три категории:

* О**бщего назначения**, которые применяются ко всему сообщению целиком.
* **Заголовки запроса** уточняют некоторую информацию о запросе, сообщая дополнительный контекст или ограничивая его некоторыми логическими условиями.
* **Заголовки представления**, которые описывают формат данных сообщения и используемую кодировку. Добавляются к запросу только в тех случаях, когда с ним передается некоторое тело.

Ниже можно видеть пример заголовков в запросе:

#### Самые частые заголовки запроса

|  |  |
| --- | --- |
| **Заголовок** | **Описание** |
| **Host** | Используется для указания того, с какого конкретно хоста запрашивается ресурс. В качестве возможных значений могут использоваться как доменные имена, так и IP-адреса. На одном HTTP-сервере может быть размещено несколько различных веб-сайтов. Для обращения к какому-то конкретному требуется данный заголовок. |
| **User-Agent** | Заголовок используется для описания клиента, который запрашивает ресурс. Он содержит достаточно много информации о пользовательском окружении. Например, может указать, какой браузер используется в качестве клиента, его версию, а также операционную систему, на которой этот клиент работает. |
| **Refer** | Используется для указания того, откуда поступил текущий запрос. Например, если вы решите перейти по какой-нибудь ссылке в этой статье, то вероятнее всего к запросу будет добавлен заголовок Refer: <https://selectel.ru> |
| **Accept** | Позволяет указать, какой тип медиафайлов принимает клиент. В данном заголовке могут быть указаны несколько типов, перечисленные через запятую (‘ , ‘). А для указания того, что клиент принимает любые типы, используется следующая последовательность — \*/\*. |
| **Cookie** | Данный заголовок может содержать в себе одну или несколько пар «Куки-Значение» в формате cookie=value. Куки представляют собой небольшие фрагменты данных, которые хранятся как на стороне клиента, так и на сервере, и выступают в качестве идентификатора. Куки передаются вместе с запросом для поддержания доступа клиента к ресурсу. Помимо этого, куки могут использоваться и для других целей, таких как хранение пользовательских предпочтений на сайте и отслеживание клиентской сессии. Несколько кук в одном заголовке могут быть перечислены с помощью символа точка с запятой (‘ ; ‘), который используется как разделитель. |
| **Authorization** | Используется в качестве еще одного метода идентификации клиента на сервере. После успешной идентификации сервер возвращает токен, уникальный для каждого конкретного клиента. В отличие от куки, данный токен хранится исключительно на стороне клиента и отправляется клиентом только по запросу сервера. Существует несколько типов аутентификации, конкретный метод определяется тем веб-сервером или веб-приложением, к которому клиент обращается за ресурсом. |

### Тело запроса

Завершающая часть HTTP-запроса — это его тело. Не у каждого HTTP-метода предполагается наличие тела. Так, например, методам вроде GET, HEAD, DELETE, OPTIONS обычно не требуется тело. Некоторые виды запросов могут отправлять данные на сервер в теле запроса: самый распространенный из таких методов — POST.

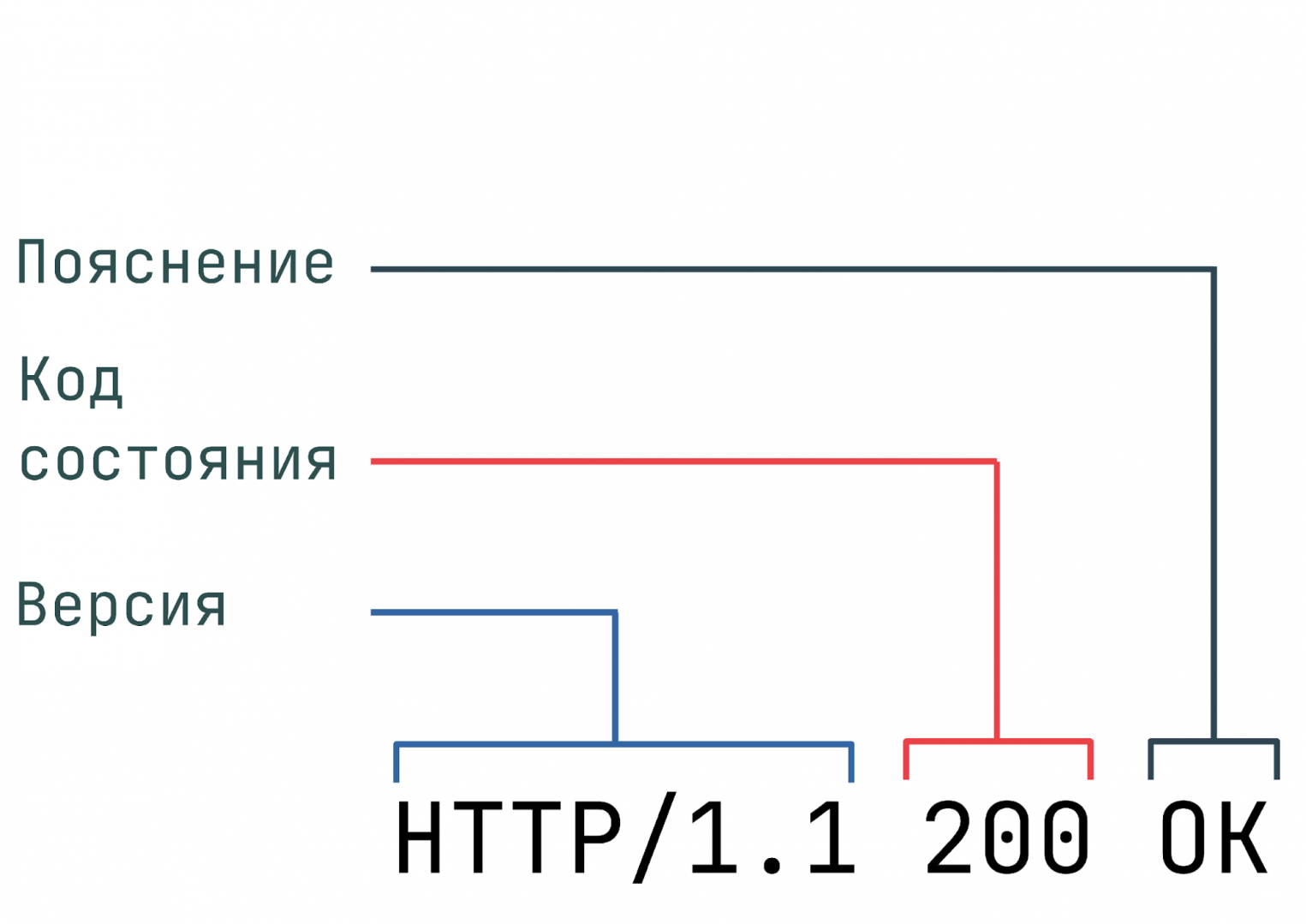
## Ответы HTTP

**HTTP-ответ** является сообщением, которое сервер отправляет клиенту *в ответ* на его запрос. Его структура равна структуре HTTP-запроса: стартовая строка, заголовки и тело.

### **Строка статуса (Status line)**

Стартовая строка HTTP-ответа называется **строкой статуса** (status line). На ней располагаются следующие элементы:

* Уже известная нам по стартовой строке запроса *версия протокола* (HTTP/2 или HTTP/1.1).
* *Код состояния*, который указывает, насколько успешно завершилась обработка запроса.
* *Пояснение* — короткое текстовое описание к коду состояния. Используется исключительно для того, чтобы упростить понимание и восприятие человека при просмотре ответа.

  
Так выглядит строка состояния ответа.

### Коды состояния и текст статуса

Коды состояния HTTP используются для того, чтобы сообщить клиенту статус их запроса. HTTP-сервер может вернуть код, принадлежащий одной из пяти категорий кодов состояния:

|  |  |
| --- | --- |
| **Категория** | **Описание** |
| **1xx** | Коды из данной категории носят исключительно информативный характер и никак не влияют на обработку запроса. |
| **2xx** | Коды состояния из этой категории возвращаются в случае успешной обработки клиентского запроса. |
| **3xx** | Эта категория содержит коды, которые возвращаются, если серверу нужно перенаправить клиента. |
| **4xx** | Коды данной категории означают, что на стороне клиента был отправлен некорректный запрос. Например, клиент в запросе указал не поддерживаемый метод или обратился к ресурсу, к которому у него нет доступа. |
| **5xx** | Ответ с кодами из этой категории приходит, если на стороне сервера возникла ошибка. |

Полный список кодов состояния доступен в спецификации к протоколу, ниже приведены только самые распространенные коды ответов:

|  |  |
| --- | --- |
| **Категория** | **Описание** |
| **200 OK** | Возвращается в случае успешной обработки запроса, при этом тело ответа обычно содержит запрошенный ресурс. |
| **302 Found** | Перенаправляет клиента на другой URL. Например, данный код может прийти, если клиент успешно прошел процедуру аутентификации и теперь может перейти на страницу своей учетной записи. |
| **400 Bad Request** | Данный код можно увидеть, если запрос был сформирован с ошибками. Например, в нем отсутствовали символы завершения строки. |
| **403 Forbidden** | Означает, что клиент не обладает достаточными правами доступа к запрошенному ресурсу. Также данный код можно встретить, если сервер обнаружил вредоносные данные, отправленные клиентом в запросе. |
| **404 Not Found** | Каждый из нас, так или иначе, сталкивался с этим кодом ошибки. Данный код можно увидеть, если запросить у сервера ресурс, которого не существует на сервере. |
| **500 Internal Error** | Данный код возвращается сервером, когда он не может по определенным причинам обработать запрос. |

Помимо основных кодов состояния, описанных в стандарте, существуют и коды состояния, которые объявляются крупными сетевыми провайдерами и серверными платформами. Так, коды состояния, используемые Selectel, можно посмотреть [здесь](https://developers.selectel.ru/docs/control-panel/status_panel_api/).

### Заголовки ответа

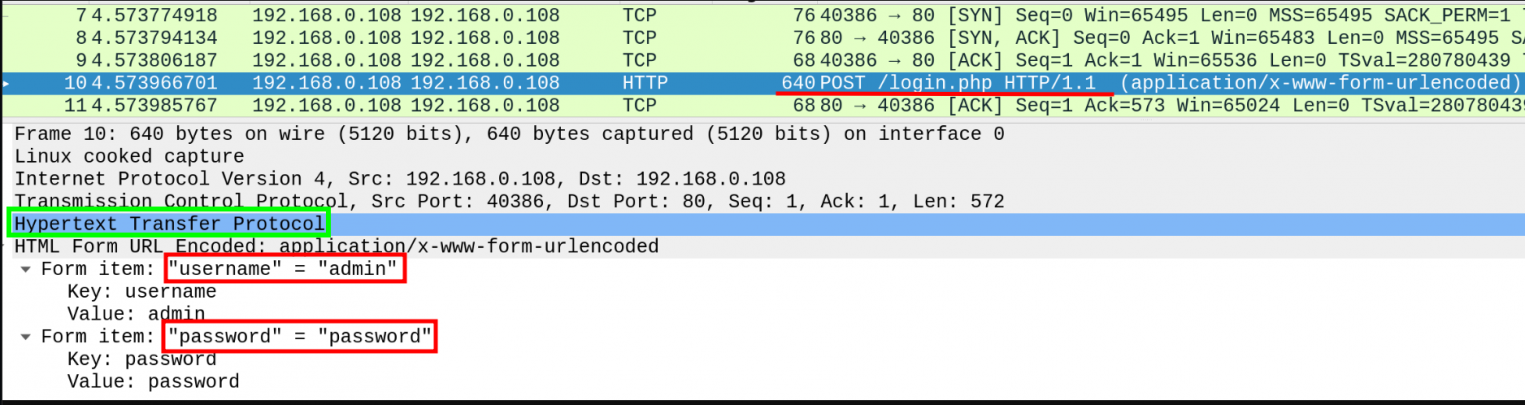
Response Headers, или заголовки ответа, используются для того, чтобы уточнить ответ, и никак не влияют на содержимое тела. Они существуют в том же формате, что и остальные заголовки, а именно «Имя-Значение» с двоеточием (:) в качестве разделителя.

Ниже приведены наиболее часто встречаемые в ответах заголовки:

|  |  |  |
| --- | --- | --- |
| **Категория** | **Пример** | **Описание** |
| **Server** | Server: ngnix | Содержит информацию о сервере, который обработал запрос. |
| **Set-Cookie** | Set-Cookie:PHPSSID=bf42938f | Содержит куки, требуемые для идентификации клиента. Браузер парсит куки и сохраняет их в своем хранилище для дальнейших запросов. |
| **WWW-Authenticate** | WWW-Authenticate: BASIC realm=»localhost» | Уведомляет клиента о типе аутентификации, который необходим для доступа к запрашиваемому ресурсу. Тело ответа Последней частью ответа является его тело. Несмотря на то, что у большинства ответов тело присутствует, оно не является обязательным. Например, у кодов «201 Created» или «204 No Content» тело отсутствует, так как достаточную информацию для ответа на запрос они передают в заголовке. |

### Безопасность HTTP-запросов, или что такое HTTPs

HTTP является расширяемым протоколом, который предоставляет огромное количество возможностей, а также поддерживает передачу всевозможных типов файлов. Однако, вне зависимости от версии, у него есть один существенный недостаток, который можно заметить если перехватить отправленный HTTP-запрос:



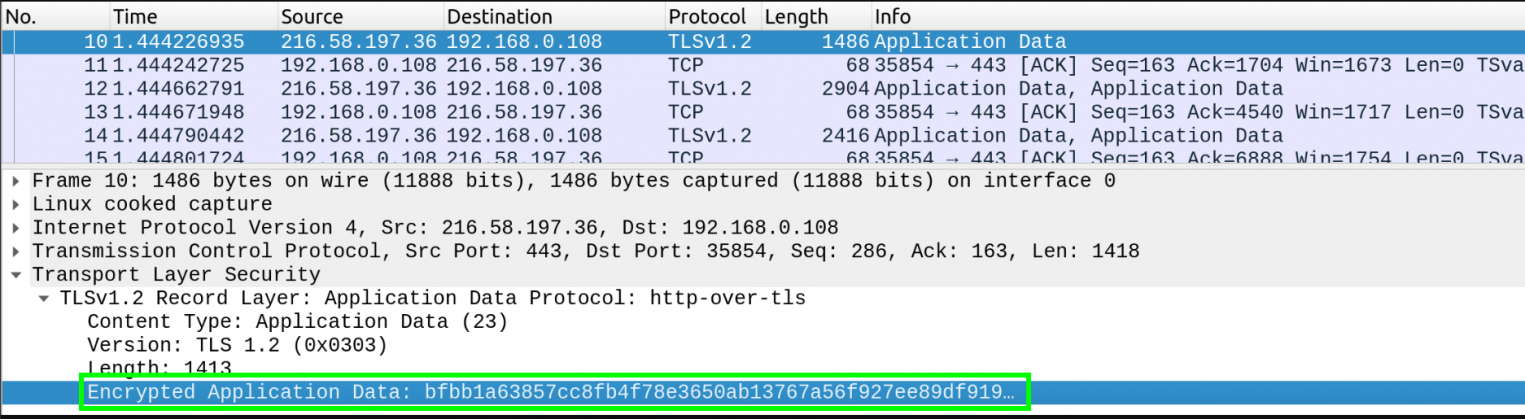
Да, все верно: данные передаются в открытом виде. HTTP сам по себе не предоставляет никаких средств шифрования.

Но как же тогда работают различные банковские приложения, интернет-магазины, сервисы оплаты услуг и прочие приложения, в которых циркулирует чувствительная информация пользователей?

Время рассказать про HTTPs!

**HTTPs (HyperText Transfer Protocol, secure)** является расширением HTTP-протокола, который позволяет шифровать отправляемые данные, перед тем как они попадут на транспортный уровень. Данный протокол по умолчанию использует порт 443.

Теперь если мы перехватим не HTTP , а HTTP**s**-запрос, то не увидим здесь ничего интересного:



Данные передаются в едином зашифрованном потоке, что делает невозможным получение учетных данных пользователей и прочей критической информации средствами обычного перехвата.

## Как отправить HTTP-запрос и прочитать его ответ

Теория это, конечно, отлично, но ничего так хорошо не закрепляет материал, как практика

Мы рассмотрим несколько способов, как написать HTTP-запрос в браузер, послать HTTP-запрос на сервер и получить ответ:

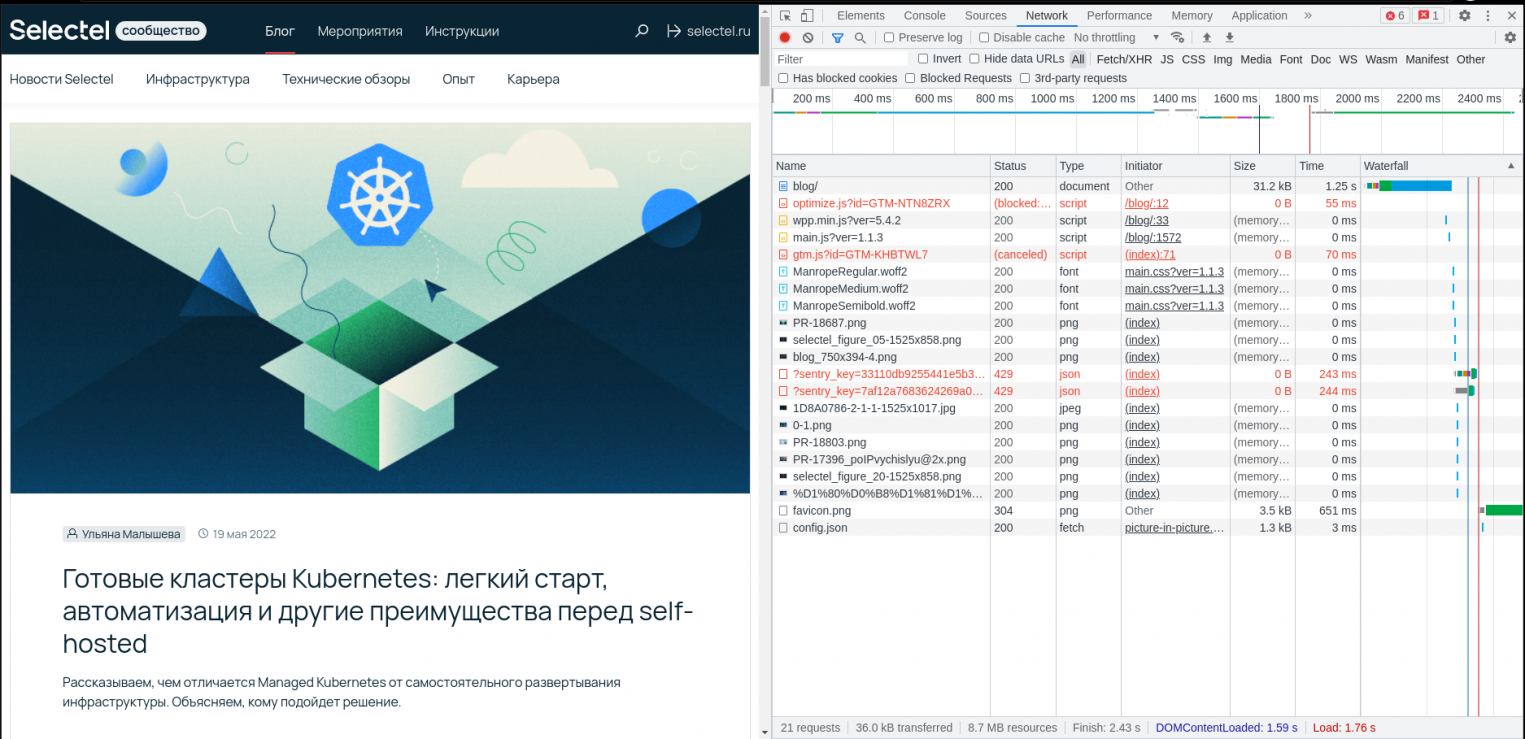
* Инструменты разработчика в браузере.
* Утилита cURL.

### Инструменты разработчика

Основной программой на наших устройствах, которая работает с HTTP-протоколом, в большинстве случаев является браузер. Помимо обычных пользователей, с браузерами часто работают и разработчики веб-приложений. Именно их инструментами мы воспользуемся для работы с запросами и ответами.

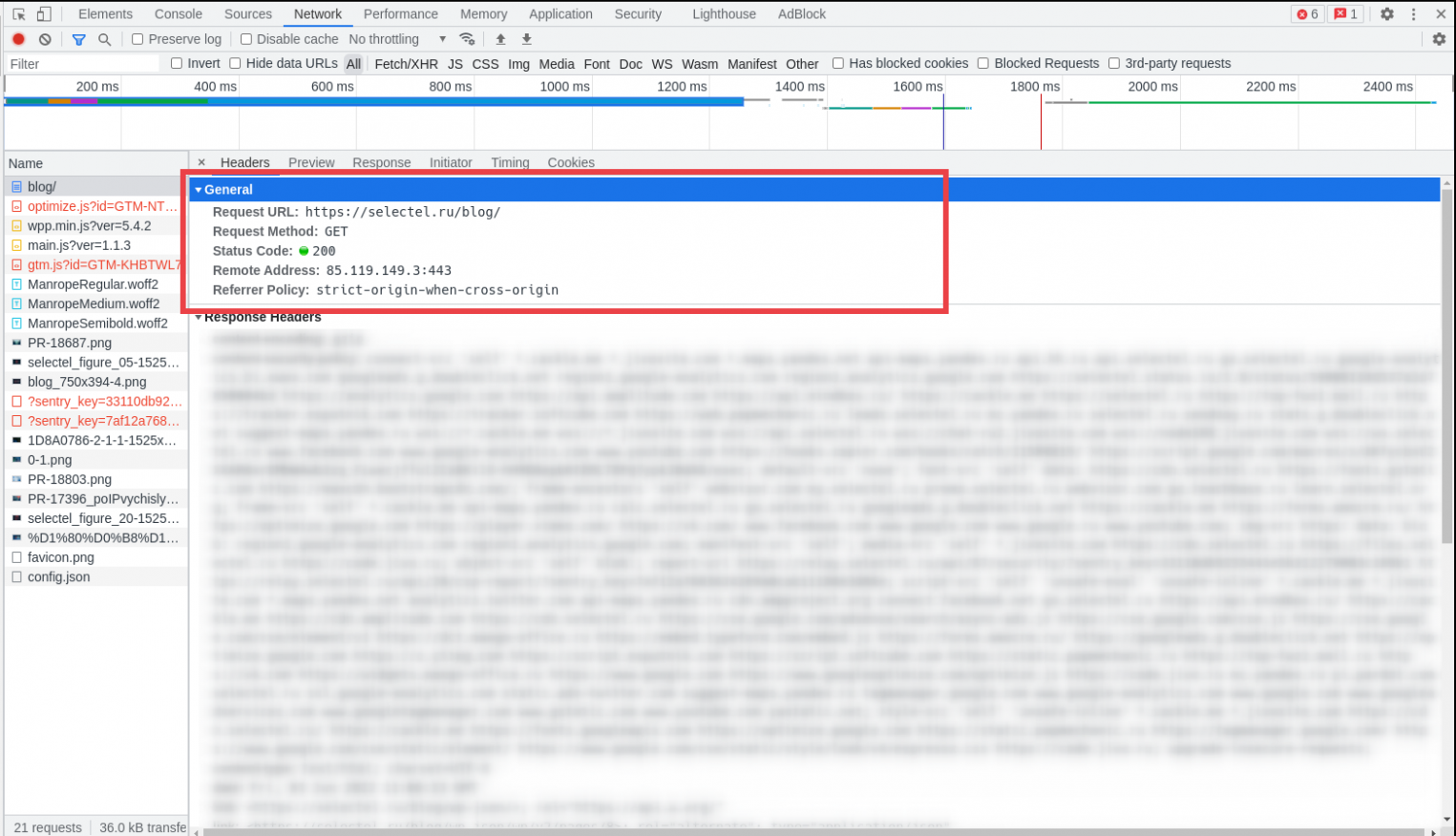
По нажатию комбинации клавиш [Ctrl+Shift+I] или просто [F12] в подавляющем большинстве современных браузеров у нас откроется окно инструментов разработчика, которая представляет собой панель с несколькими вкладками. Нужная нам вкладка обычно называется Network. Перейдем в нее, а в адресной строке введем URL того сайта, на который хотим попасть. В качестве примера воспользуемся сайтом блога Selectel — <https://selectel.ru/blog/>.

После нажатия Enter сайт начнет загружаться, а открытая вкладка Network — заполняться различными элементами, начиная все больше напоминать приборную панель самолета.



Не спешите пугаться. Это всего лишь список ресурсов, которые нужны для правильного отображения и работы сайта.

Нажав на любой из них, мы можем увидеть детали обработки отправленного запроса:



В данном запросе, например:

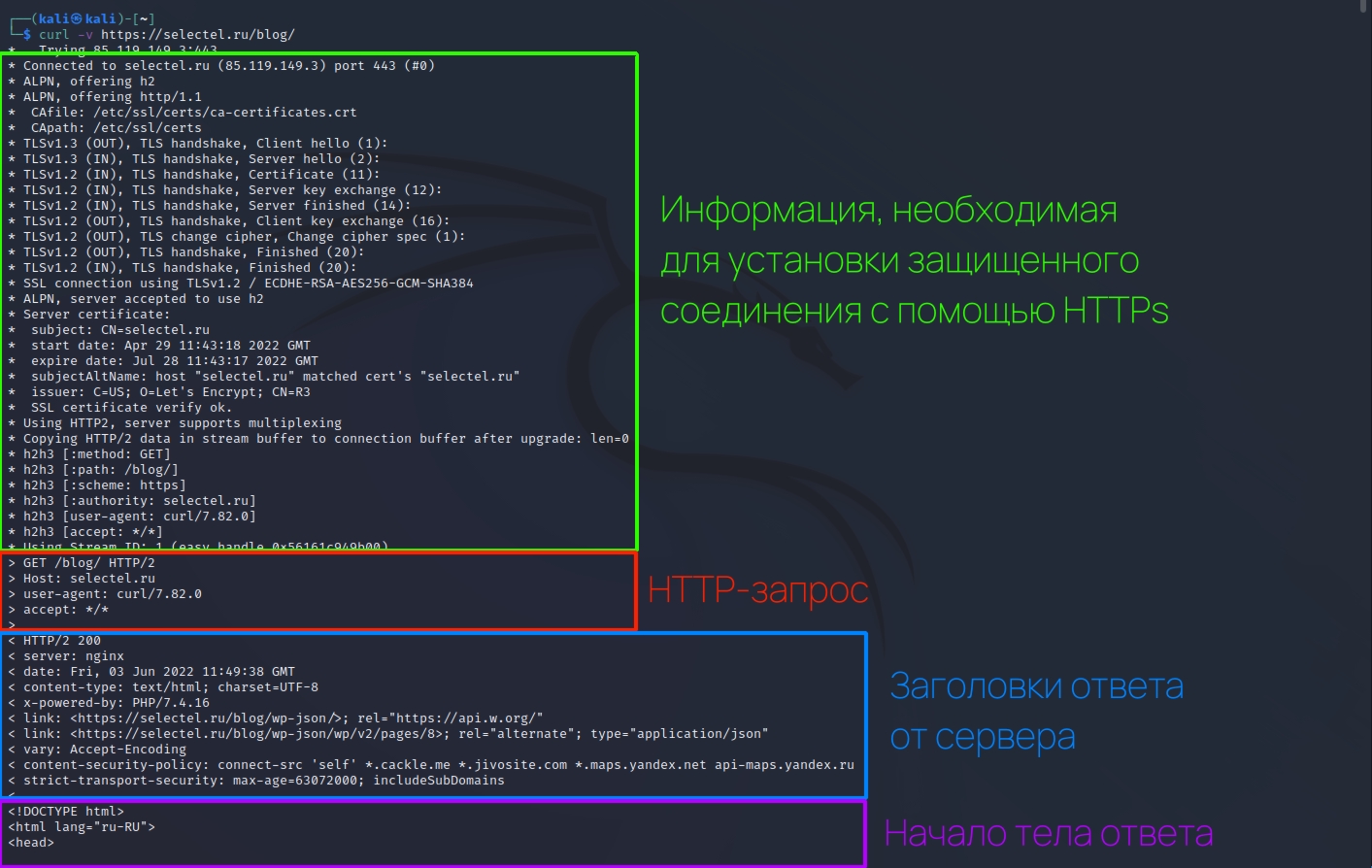
* URL, к которому было совершено обращение — <https://selectel.ru/blog>,
* Метод, который был использован в запросе — GET,
* И в качестве кода возврата сервер вернул нам страницу с кодом статуса — 200 OK

### Утилита cURL

Следующий инструмент, с помощью которого мы сможем послать запрос на тот или иной сервер, это утилита cURL.

**cURL (client URL)** является небольшой утилитой командной строки, которая позволяет работать с HTTP и рядом других протоколов.

Для отправки запроса и получения ответа мы можем воспользоваться флагом -v и указанием URL того ресурса, который мы хотим получить. «Схему» HTTP-запроса можно увидеть на скрине ниже:



После запуска утилита выполняет:

* подключение к серверу,
* самостоятельно разрешает все вопросы, необходимые для отправки запроса по HTTPs,
* отправляет запрос, содержимое которого мы можем видеть благодаря флагу -v,
* принимая ответ от сервера, отображает его в командной строке «как-есть».

Помимо этого, у данной утилиты есть огромное количество опций, которые предоставляют возможности по детальной настройке отправляемых запросов. Все эти возможности и делают ее такой популярной у веб-разработчиков и других специалистов, которым приходится работать с протоколом HTTP.

## Заключение

HTTP представляет собой расширяемый протокол прикладного уровня, на котором работает весь веб-сегмент интернета. В данной статье мы рассмотрели принцип его работы, структуру, «компоненты» HTTP-запросов. Коснулись вопросов отличия версий протокола, HTTPs — его расширения, добавляющего шифрование. Разобрали устройство запроса, поняли, как можно отправить HTTP-запрос и получить ответ от сервера