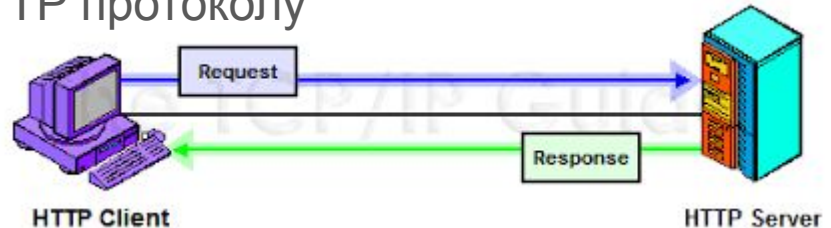


6. Веб-приложения

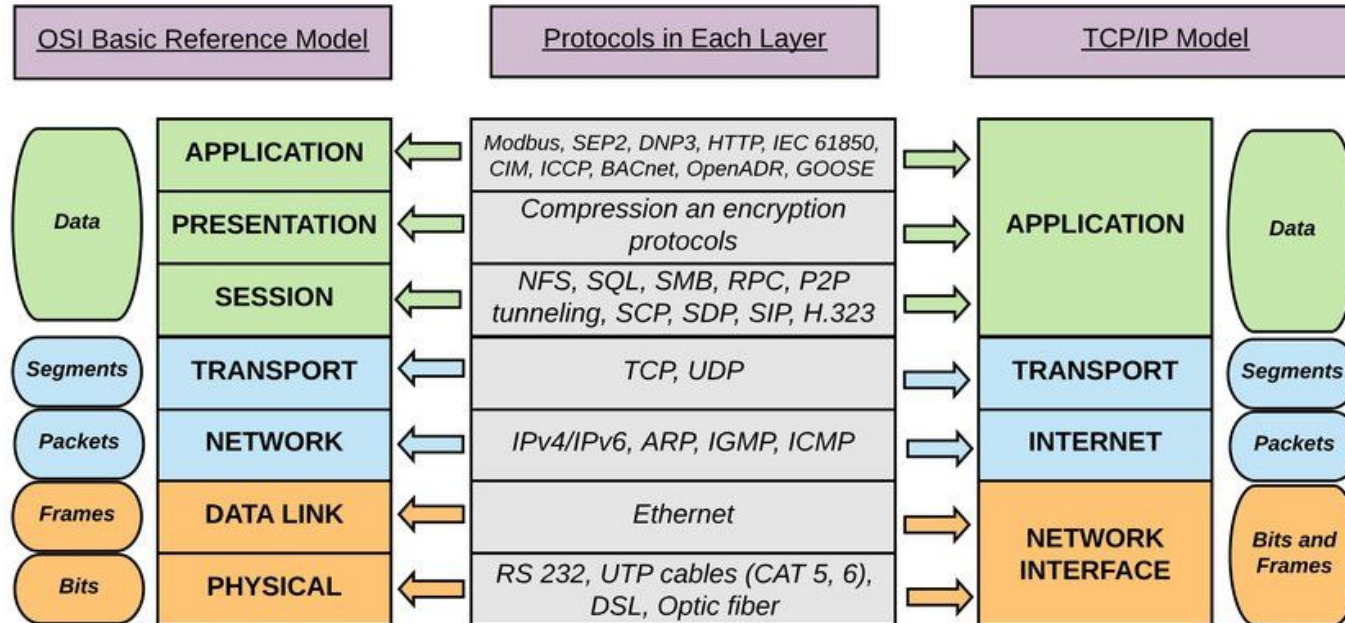
Когда заходим в браузер

- делаем запрос – открываем ссылку в интернете
 - на разные ресурсы разные ссылки
 - иногда возвращаются ошибки
 - 404 Not Found
 - 503 unavailbale
 - иногда вводим какие-то данные
- на запрос отвечает сервер
- клиент-серверная модель работы
 - клиент – запрос, сервер – ответ
- очень часто взаимодействие по HTTP протоколу



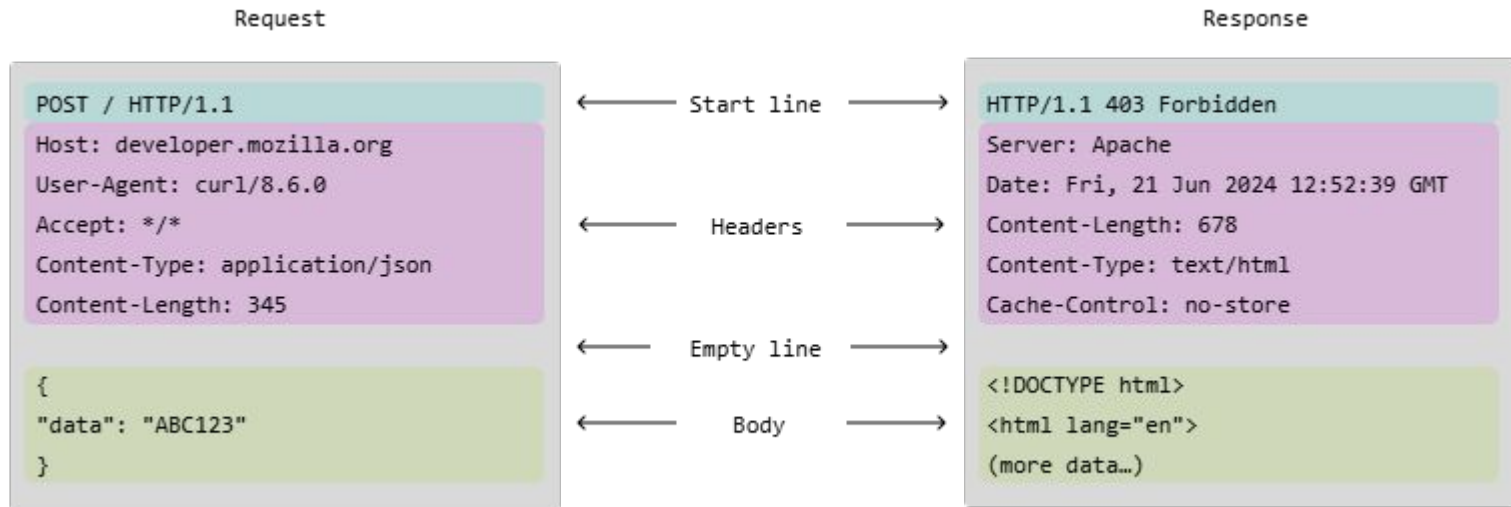
HTTP протокол в модели OSI

- OSI - open systems interconnection
- HTTP: APPLICATION уровень – взаимодействие сети и пользователя



Структура HTTP пакета

- структура:
 - стартовая строка (разный формат для клиента и сервера)
 - Метод URI HTTP/Версия / HTTP/Версия КодСостояния Пояснение
 - заголовки: Host, Content-type, и т.д.
 - тело сообщения



Стартовая строка

- **Клиент:** Метод URI HTTP/Версия | POST / HTTP/1.1
- **Метод:**
 - GET – получить содержимое ресурса
 - POST – обновить ресурс введенными данными
 - PUT – загрузить содержимое запроса на указанный URI
 - DELETE - удалить
 - PATCH, OPTIONS, HEAD, TRACE, CONNECT
- **URI (Uniform Resource Identifier)**
- **HTTP Версия**
 - HTTP/0.9 – 1991г. ; HTTP/1.1 - 1999г ; HTTP/2.0 - 2015г.
- **Сервер:** HTTP/Версия КодСостояния | HTTP/1.1 403 Forbidden
 - код состояния

Виды кодов состояния

Код	Класс	Назначение
<div>1xx</div>	<div>Информационный</div> <div>(англ. informational)</div>	<p>Информирование о процессе передачи.</p> <p>В HTTP/1.0 — сообщения с такими кодами должны игнорироваться.</p> <p>В HTTP/1.1 — клиент должен быть готов принять этот класс сообщений как обычный ответ, но ничего отправлять серверу не нужно.</p> <p>Сами сообщения от сервера содержат только стартовую строку ответа и, возможно, несколько полей заголовка. Прокси-серверы подобные сообщения должны отправлять дальше от сервера к клиенту.</p>
<div>2xx</div>	<div>Успех</div> <div>(англ. Success)</div>	<p>Информирование о случаях успешного принятия и обработки запроса клиента. В зависимости от статуса, сервер может ещё передать заголовки и тело сообщения.</p>
<div>3xx</div>	<div>Перенаправление</div> <div>(англ. Redirection)</div>	<p>Сообщает клиенту, что для успешного выполнения операции необходимо сделать другой запрос (как правило по другому URI). Из данного класса пять кодов 301, 302, 303, 305 и 307 относятся непосредственно к перенаправлениям (редирект). Адрес, по которому клиенту следует произвести запрос, сервер указывает в заголовке <code>Location</code>. При этом допускается использование фрагментов в целевом URI.</p>
<div>4xx</div>	<div>Ошибка клиента</div> <div>(англ. Client Error)</div>	<p>Запрос клиента содержит ошибку. При использовании всех методов, кроме <code>HEAD</code>, сервер должен вернуть в теле сообщения гипертекстовое пояснение для пользователя.</p>
<div>5xx</div>	<div>Ошибка сервера</div> <div>(англ. Server Error)</div>	<p>Операция не выполнена по вине сервера. Для всех ситуаций, кроме использования метода <code>HEAD</code>, сервер должен включать в тело сообщения объяснение, которое клиент отобразит пользователю.</p>

Практика

- <https://jsonplaceholder.typicode.com/>
 - туда можно слать разные запросы на GET / POST / PUT и т.д.
- <https://jsonplaceholder.typicode.com/posts>
- `curl -X POST https://jsonplaceholder.typicode.com/posts -H "Content-Type: application/json" -d "{\"title\":\"POST from CMD\", \"body\":\"Test body\", \"userId\":1}"`
- `curl -X PUT https://jsonplaceholder.typicode.com/posts/1 -H "Content-Type: application/json" -d "{\"id\":1, \"title\":\"Updated Title\", \"body\":\"Updated body\", \"userId\":1}"`
- `curl -X GET https://jsonplaceholder.typicode.com/posts/1`
- `curl -X DELETE https://jsonplaceholder.typicode.com/posts/1`

Как это сделать на Go?

- Пример: <https://go.dev/doc/articles/wiki/>
- net/http <https://pkg.go.dev/net/http>
 - виды методов: <https://pkg.go.dev/net/http#pkg-constants>
 - виды запросов: <https://pkg.go.dev/net/http#pkg-constants>
 - запрос: <https://pkg.go.dev/net/http#Request>
 - ответ: <https://pkg.go.dev/net/http#Response>
- Сервер: <https://gobyexample.com/http-server>
- Клиент: <https://gobyexample.com/http-client>

Ссылки

- OSI:

https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI

- HTTP:

https://ru.wikipedia.org/wiki/HTTP#%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_HTTP-%D1%81%D0%BE%D0%BE%D0%B1%D1%89%D0%B5%D0%BD%D0%B8%D1%8F

- Еще один пример как сделать HTTP сервер

<https://dev.to/stungnet/making-a-basic-http-server-with-golang-37lk>