

Южно-Уральский
государственный
университет

Национальный
исследовательский
университет

КРОК

Архитектура клиент-сервер (HTTP)

КРОК


Челябинск, ул. Карла Маркса, д. 38

Смирнов Анатолий
Технический менеджер

Антонов Сергей
Тимлид группы разработчиков,
старший инженер-разработчик

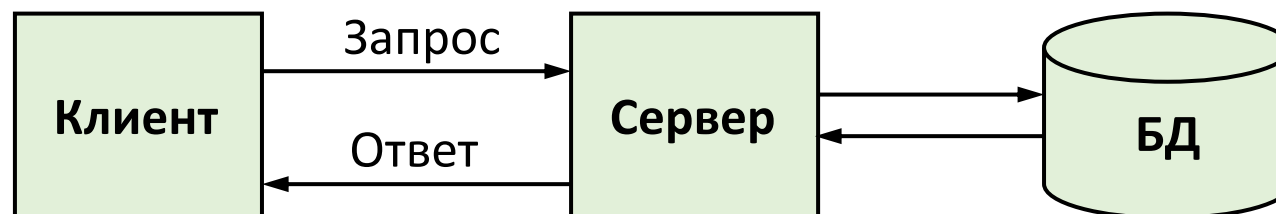
Кузнецов Сергей
Инженер-разработчик



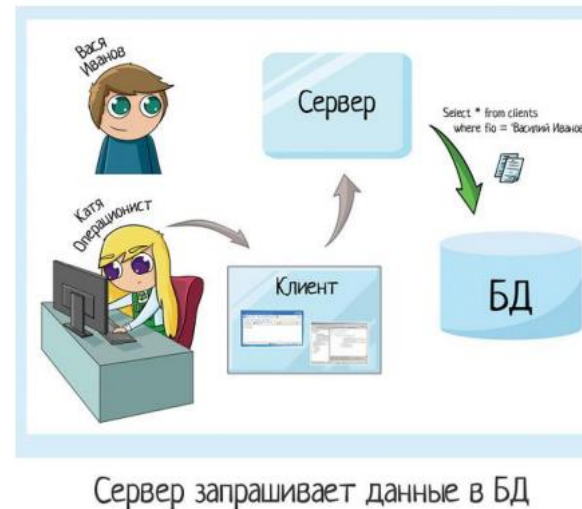
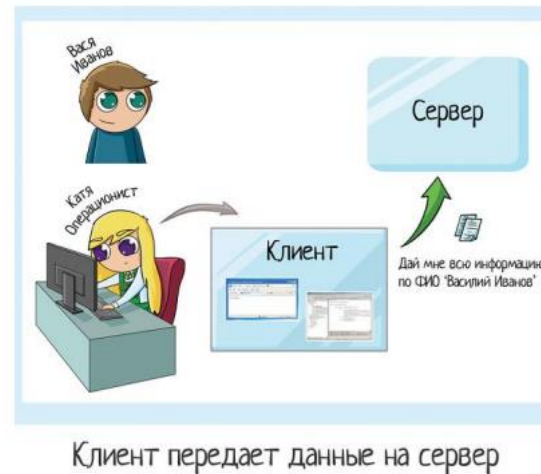
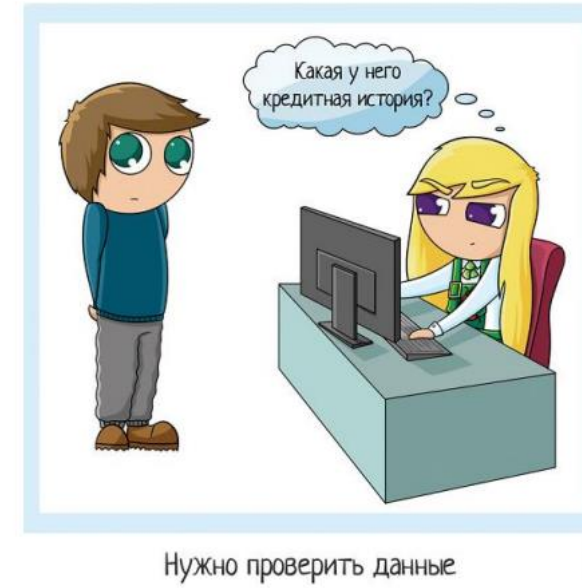
- HTML/CSS
- Клиент-сервер (HTTP)  Сейчас мы тут
- Установка ПО (самостоятельно)
- Git (самостоятельно)
- SQL, СУБД PostgreSQL
- PHP
- JavaScript
- Каркас веб-приложения (вместе)
- Выполнение проекта (самостоятельно)



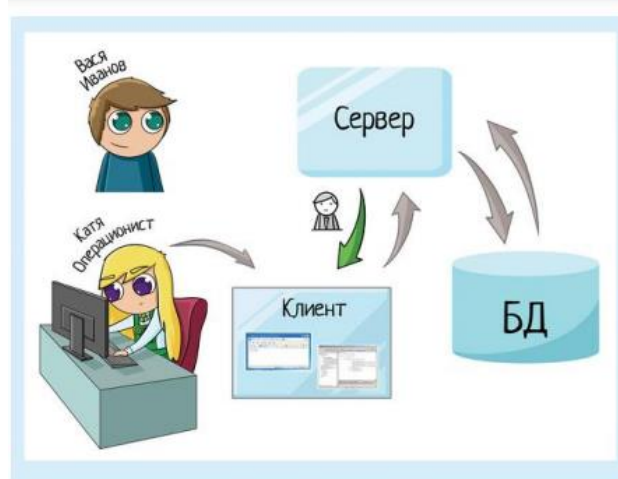
- Веб-приложение – это клиент-серверное приложение, в котором клиентом выступает браузер, а сервером – веб-сервер (в широком смысле)
- Архитектура «клиент-сервер» определяет общие принципы организации взаимодействия в сети, где имеются серверы, узлы-поставщики некоторых специфичных функций (сервисов) и клиенты (потребители этих функций)



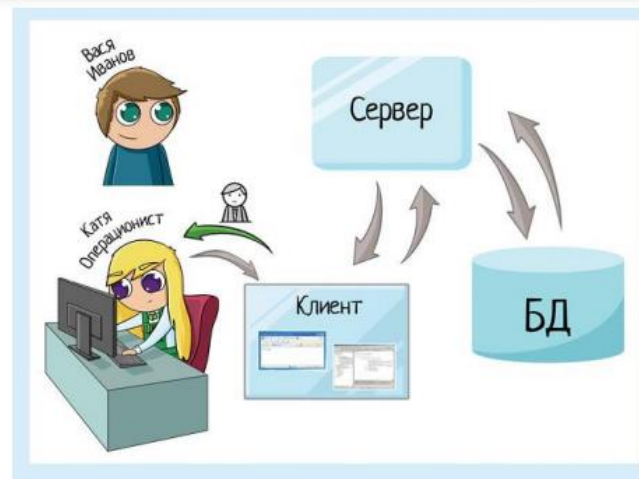
Пример клиент-серверной архитектуры в банке



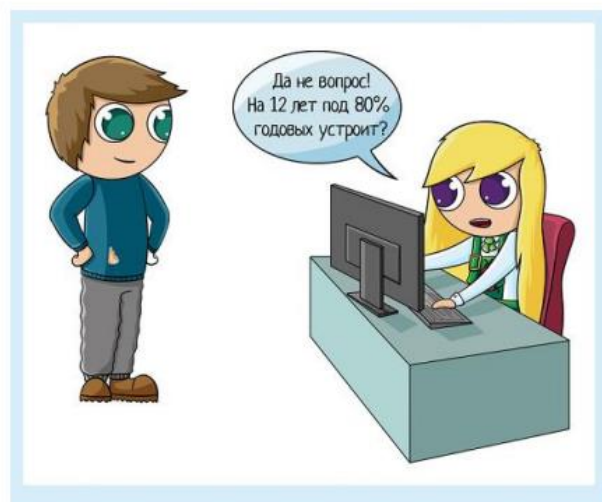
Пример клиент-серверной архитектуры в банке



Сервер передает ее клиенту



Клиент показывает инфу Кате



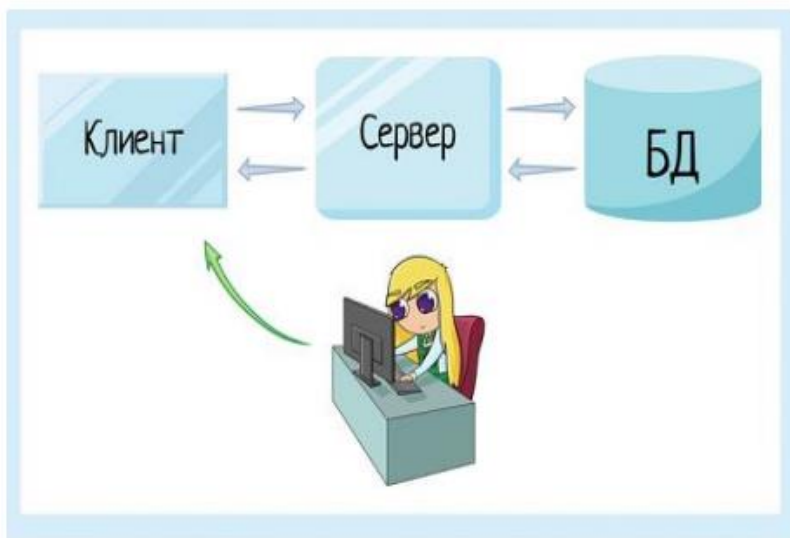
Катя дает ответ Васе



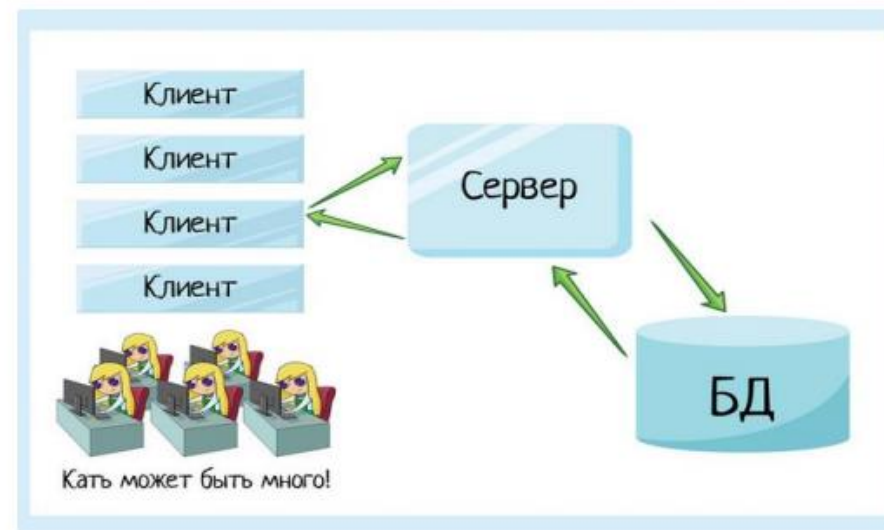
Все счастливы, ура!



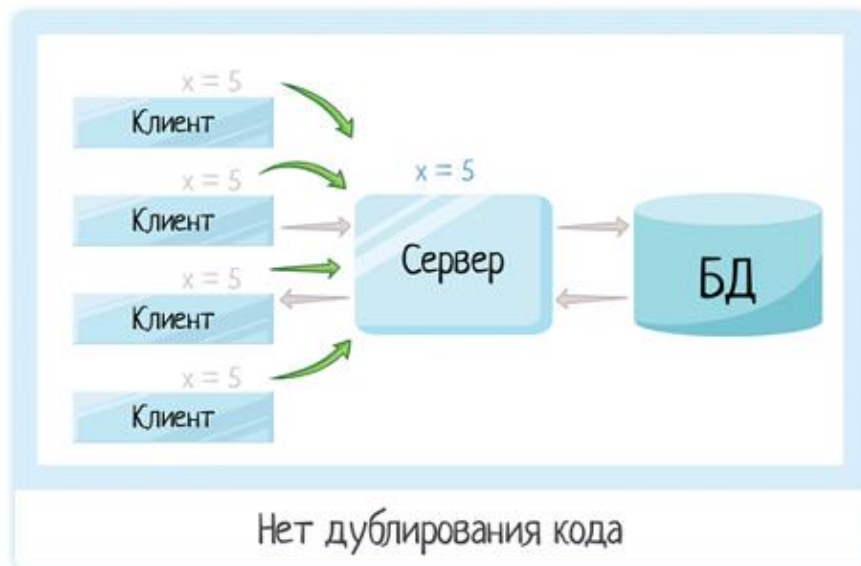
- Клиент — та программа, с которой работает пользователь. Он даже не знает, это у него на компьютере программа целиком, или где-то за ней прячутся сервер с базой. Он работает в браузере или с desktop-приложением.
- Сервер — компьютер, на котором хранится само приложение. Весь код, вся логика, все дополнительные материалы и справочники.



Пользователь использует клиент

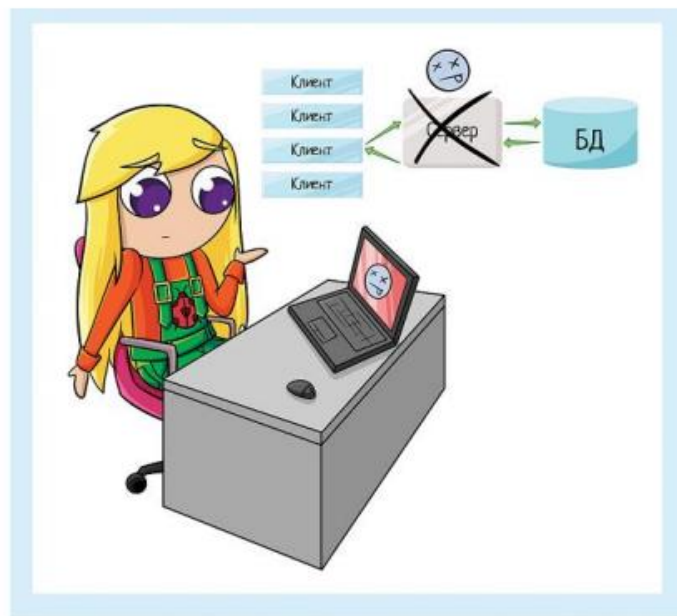


Плюсы клиент-серверной архитектуры

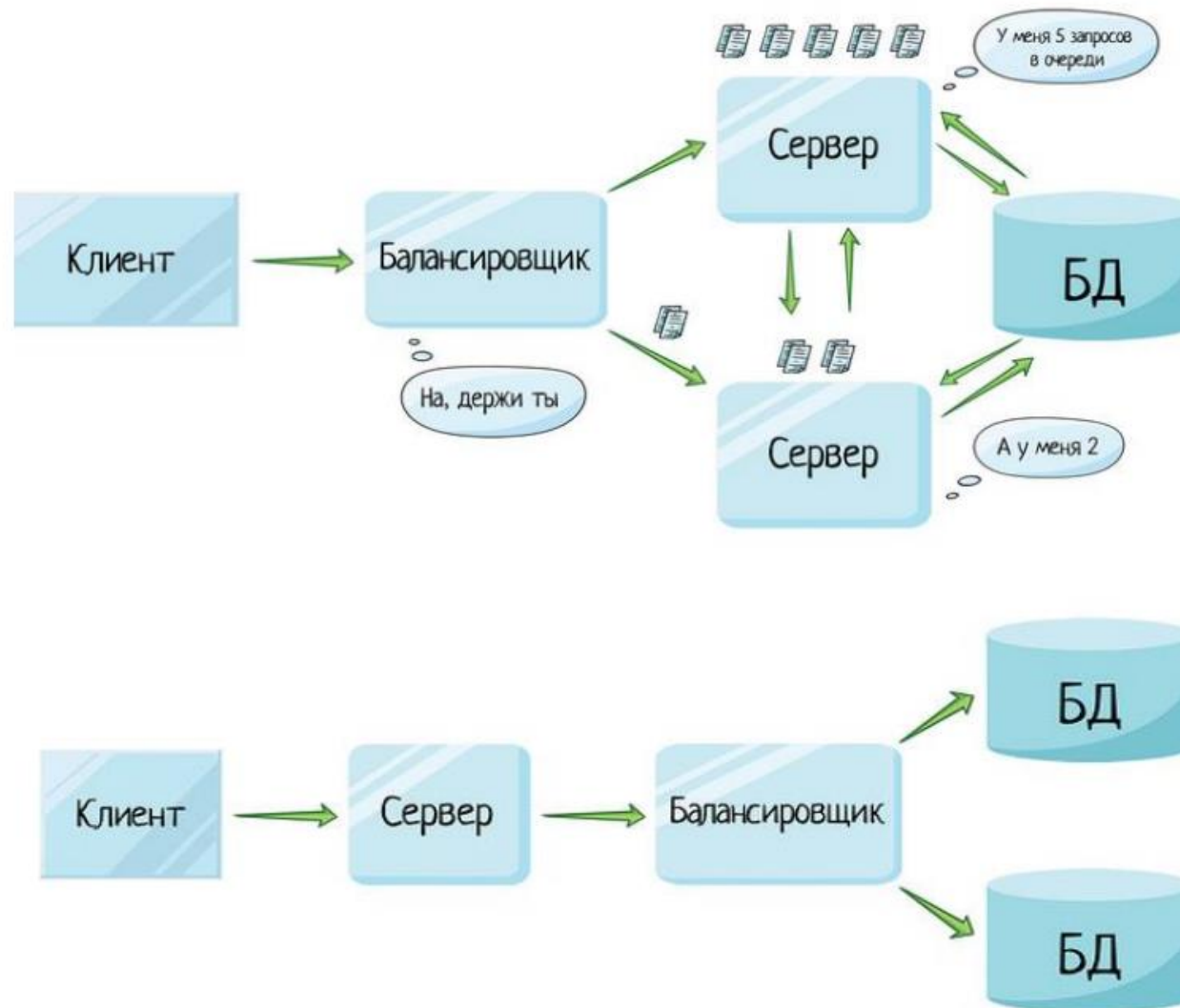


Выдавать простому операционисту много прав небезопасно





Сломался сервер — все клиенты не работают



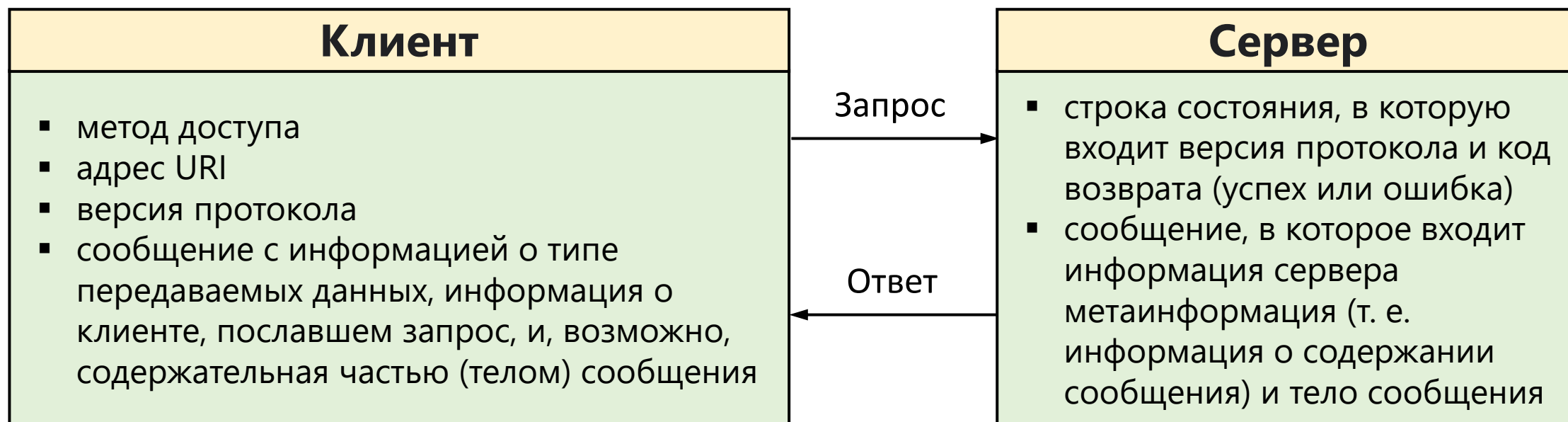


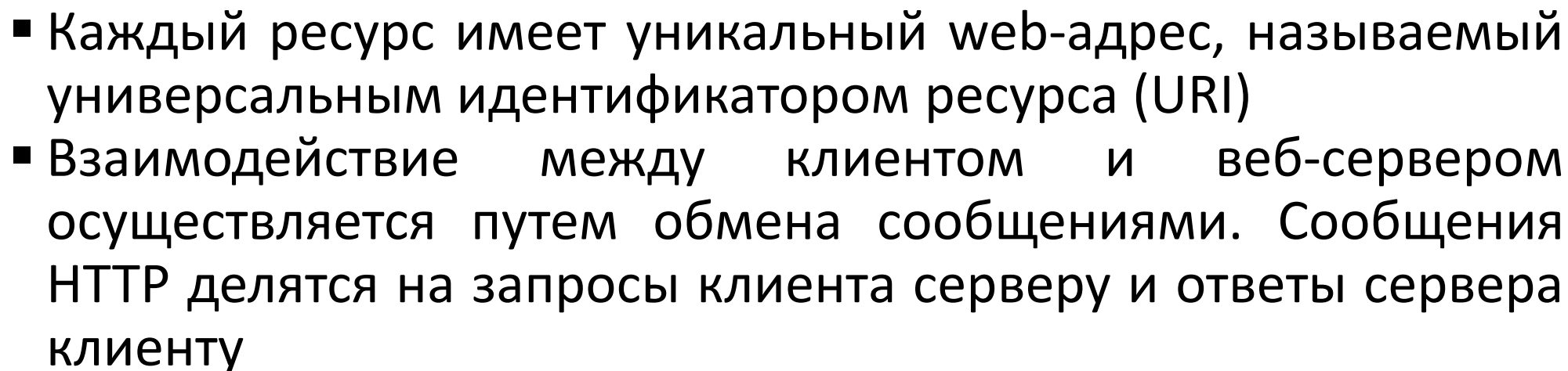
- HTTP (HyperText Transfer Protocol, протокол передачи гипертекста) — протокол прикладного уровня
- Описание протокола: <https://www.w3.org/Protocols>

Уровень OSI	Пример
Прикладной	HTTP, HTTPS, SMTP, DNS, FTP
Транспортный	TCP, UDP
Сетевой	IP
Сетевых интерфейсов	Ethernet, Wi-Fi

В первую очередь, предназначен для передачи данных в виде текстовых сообщений. Основой протокола HTTP является технология «клиент-сервер».

- Клиенты инициируют соединение и посылают запрос серверу
- Сервер ожидает соединения для получения запроса, производит необходимые действия и возвращают обратно сообщение с результатом





11

protocol://user:password@host:port/path/file?parameters#fragment

protocol	Прикладной протокол, посредством которого получают доступ к ресурсу (например, http, https, ftp)
user	Пользователь, от имени которого получают доступ к ресурсу либо сам пользователь в качестве ресурса (необязательный параметр);
password	Пароль пользователя для аутентификации при доступе к ресурсу (необязательный параметр)
host	IP-адрес или имя сервера, на котором расположен ресурс
port	Номер порта, на котором работает сервер, предоставляющий доступ к ресурсу (может не указываться, если используется стандартный порт протокола)
path	Путь к файлу, содержащему ресурс
file	Файл, содержащий ресурс
parameters	Параметры для обработки ресурсом-программой (необязательный параметр)
fragment	Точка в файле, начиная с которой следует отображать ресурс (необязательный параметр, например, заголовок в тексте)



Сообщения запроса и ответа имеют общий формат. Оба типа сообщений выглядят следующим образом:

- сначала идет начальная строка (**Start-line**);
- затем одно или несколько полей заголовка, называемых, также, просто заголовками (**Headers**);
- затем пустая строка (то есть строка, состоящая из символов **CR** и **LF**), указывающая конец полей заголовка;
- а затем, возможно, тело сообщения (**Body**).

Start-line <CR<LF> (стартовая строка)
Headers <CR<LF> (заголовки)
<CR<LF> (пустая строка)
Body (тело)

← Типовая структура
формата HTTP



Заголовки бывают четырех видов:

- общие заголовки (general-headers), могут присутствовать как в запросе, так и в ответе;
- заголовки запросов (request-headers), могут присутствовать только в запросе;
- заголовки ответов (response-headers), могут присутствовать только в ответе;
- заголовки объекта (entity-headers), относятся к телу сообщения и описывают его содержимое.

Каждый заголовок состоит из названия, символа двоеточия ":" и значения.

Заголовок	Назначение
Host	Адрес и, при необходимости, порт (через двоеточие) сервера, к которому делается запрос (обязателен для версии протокола HTTP 1.1)
User-Agent	Название программного обеспечения клиента
Cookie	Cookie, которые сохранены у клиента, связанные с данным сервером
Content-Type	MIME-тип содержимого (например, text/html или application/javascript)
Content-Length	Длина тела сообщения в байтах
Set-Cookie	Ответ сервера клиенту для установки значения Cookie в браузере

	Метод	Назначение
Самые популярные	GET	Получить определённый ресурс (например, HTML-файл, содержащий информацию о товаре или список товаров)
	POST	Создать новый ресурс (например, добавить новую статью на вики, добавить новый контакт в базу данных)
	PUT	Обновить существующий ресурс (или создать новый, если таковой не существует)
	DELETE	удалить определённый ресурс
	HEAD	получить метаданные об определённом ресурсе без получения содержания, как это делает запрос GET
	TRACE, OPTIONS, CONNECT, PATCH	Эти команды используются для менее популярных/более сложных задач (запрос информации об опциях соединения (например, методах, типах документов, кодировках))



Первая строка ответа сервера - это строка состояния (Status-Line). Она состоит из версии протокола, числового кода состояния, поясняющей фразы, разделенных пробелами и завершающих символов конца строки:

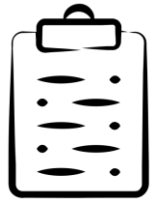
<Версия HTTP> <Код состояния> <Поясняющая фраза><CR><LF>

↑
Пробел

↑
Пробел



- Элемент «Код состояния» (Status-Code) – это целочисленный трехразрядный (трехзначный) код результата понимания и удовлетворения запроса
- Поясняющая фраза (Reason-Phrase) представляет собой короткое текстовое описание кода состояния



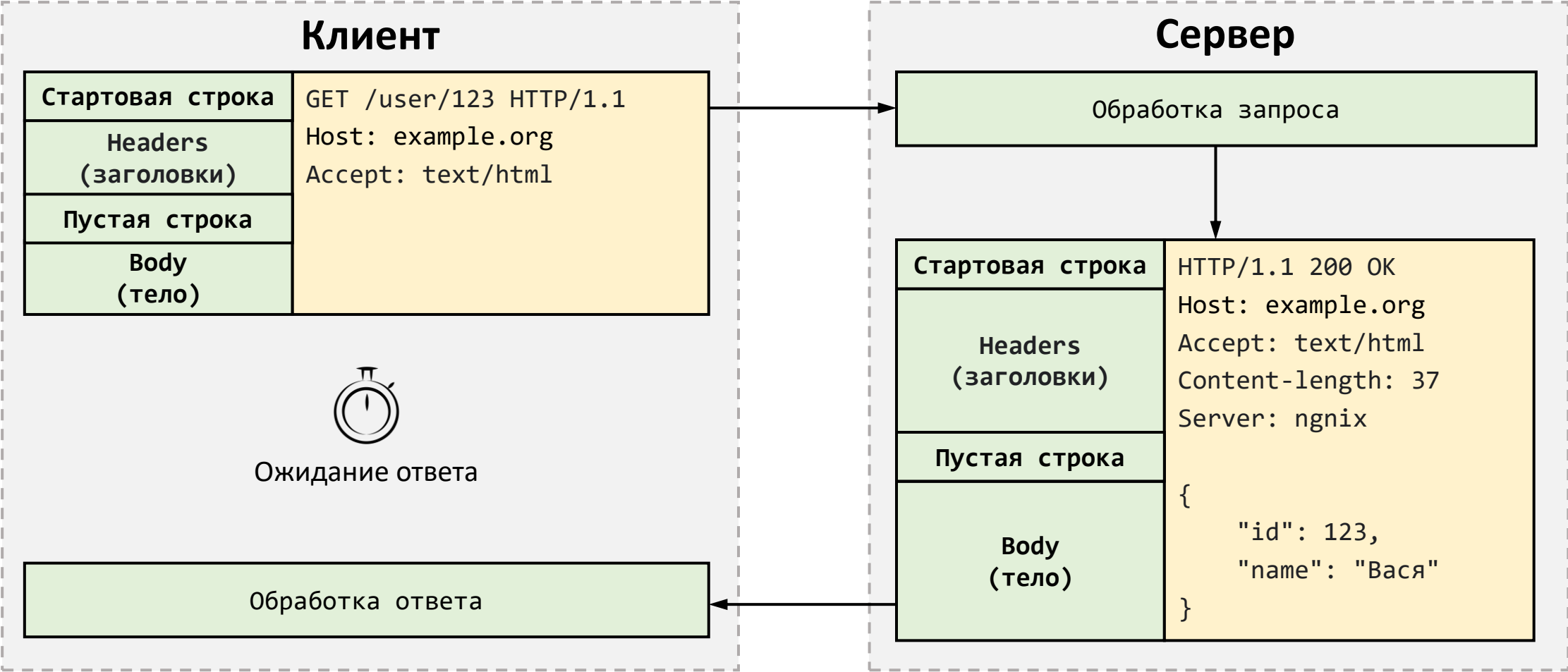
Первая цифра кода состояния определяет класс ответа. Последние две цифры не имеют определенной роли в классификации. Имеется 5 значений первой цифры:

Группа	Назначение
1xx	Информационные коды – запрос получен, продолжается обработка
2xx	Успешные коды – действие было успешно получено, понято и обработано
3xx	Коды перенаправления – для выполнения запроса должны быть предприняты дальнейшие действия
4xx	Коды ошибок клиента – запрос имеет ошибку синтаксиса или не может быть выполнен
5xx	Коды ошибок сервера – сервер не в состоянии выполнить допустимый запрос

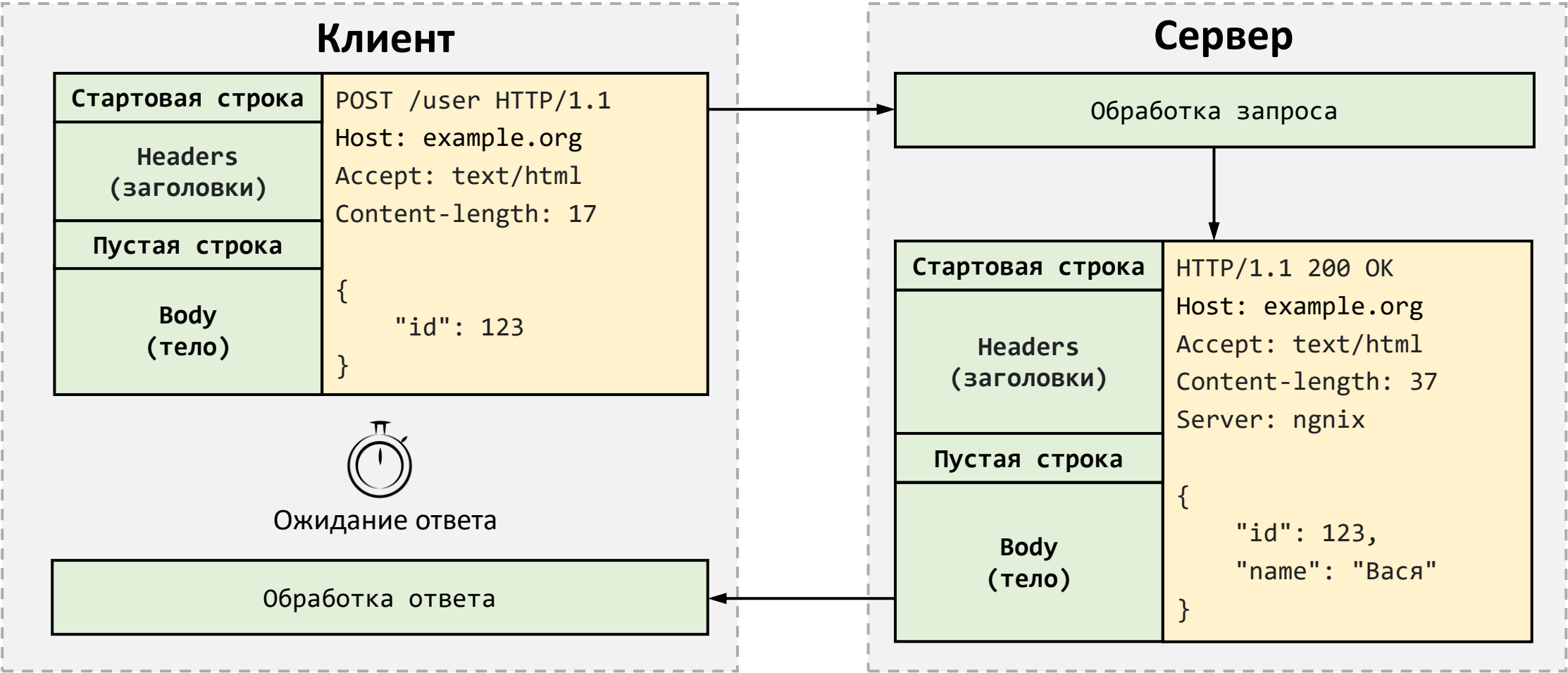
Код	Поясняющая фраза согласно RFC 2068	Перевод
100	Continue	Продолжить
200	OK	ОК
201	Created	Создано
204	No Content	Нет содержимого
400	Bad Request	Неверный запрос
401	Unauthorized	Несанкционированно
404	Not Found	Не найдено
405	Method Not Allowed	Метод не разрешен
500	Internal Server Error	Внутренняя ошибка сервера
503	Service Unavailable	Сервис недоступен

Пример GET-запроса

http://example.org/user/123



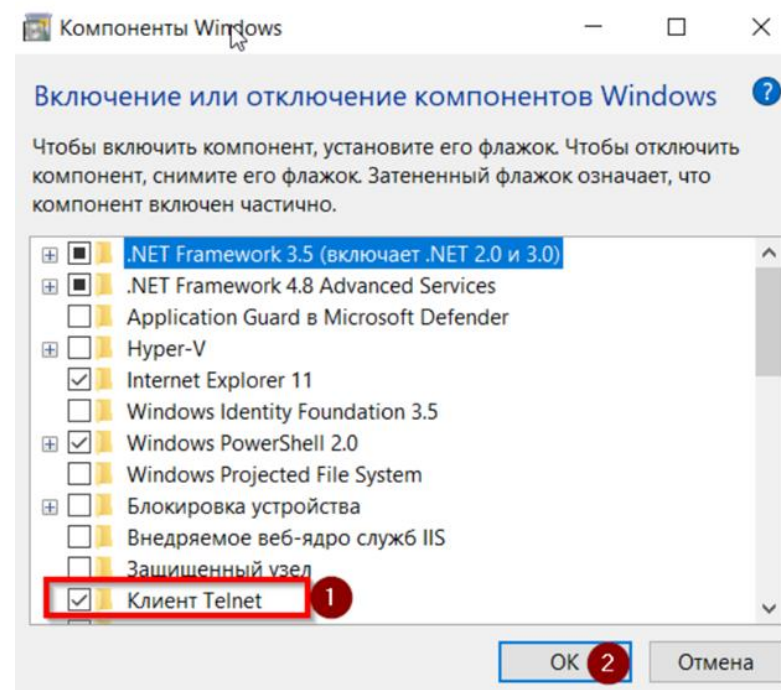
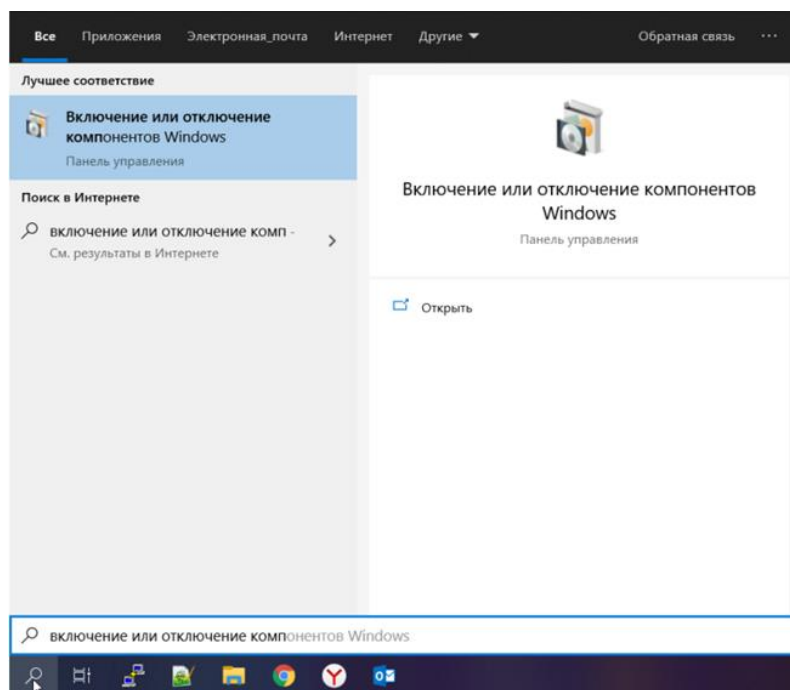
http://example.org/user



Браузеры – это тонкие клиенты и они реализуют обмен с веб-серверами по протоколу HTTP. Когда вы вводите запрос в адресную строку, отправляется запрос с методом GET, на указанный сервер (хост).

Чтобы понять, как это делает браузер, давайте сформируем запросы к серверу вручную при помощи терминального клиента для обмена незашифрованными текстовыми сообщениями telnet

1. Включим компонент telnet в MS Windows (на примере 10й версии):



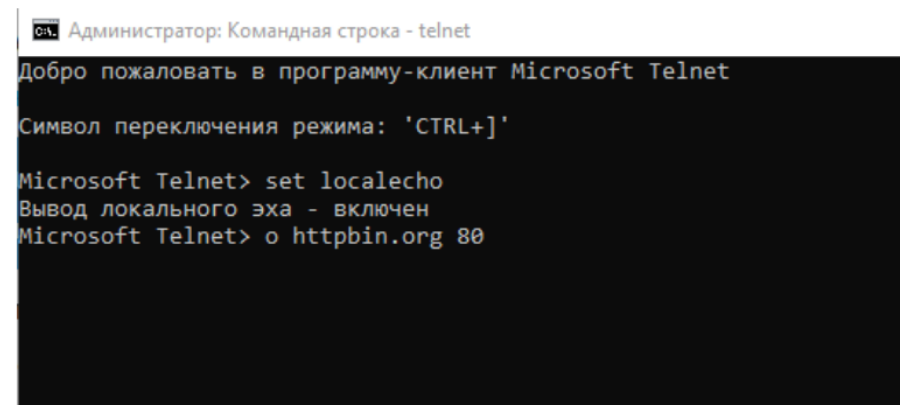
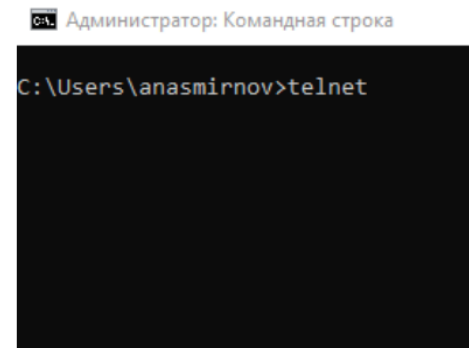
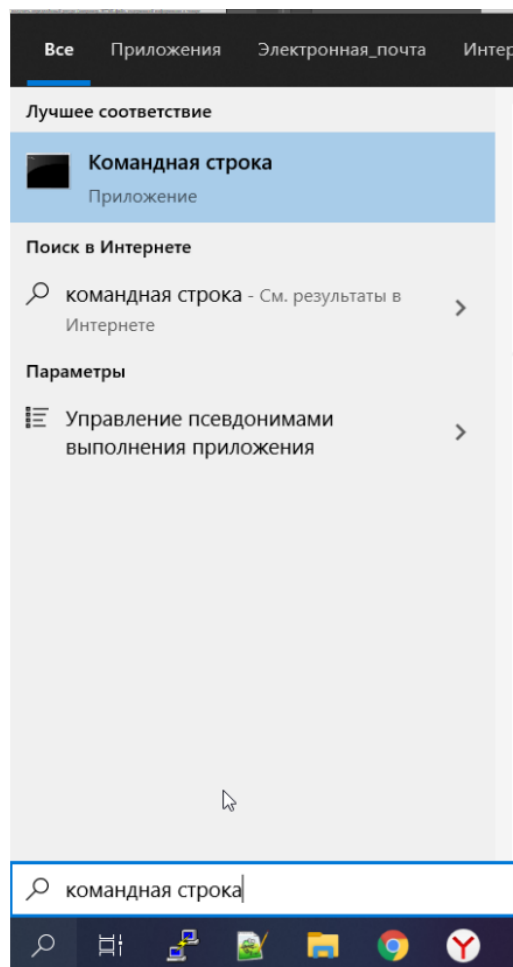
2. Запустим командную строку

3. Запустим telnet (вводим в командную строку **telnet** и нажимаем Enter)

4. Далее устанавливаем режим отображения вводимых символов:

set localecho

5. Подключаемся к тестовому серверу **httpbin.org**, вводим команду и нажимаем Enter:
o httpbin.org 80



6. После подключения вводим HTTP-запрос (метод GET) и завершаем ввод, нажав 2 раза Enter:

```
GET /get HTTP/1.1
Host: httpbin.org
<Enter>
<Enter>
```

7. Получаем HTTP-ответ от сервера

0% Telnet httpbin.org

```
Welcome to Microsoft Telnet Client

Escape Character is 'CTRL+]'

Microsoft Telnet> o httpbin.org 80
Connecting To httpbin.org...
GET /get HTTP/1.1
Host: httpbin.org
```

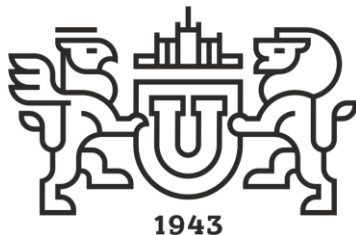
```
HTTP/1.1 200 OK
Date: Tue, 22 Mar 2022 05:41:47 GMT
Content-Type: application/json
Content-Length: 198
Connection: keep-alive
Server: gunicorn/19.9.0
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true

{
  "args": {},
  "headers": {
    "Host": "httpbin.org",
    "X-Amzn-Trace-Id": "Root=1-6239619b-4438f8dc3edfd356631372f7"
  },
  "origin": "31.162.211.62",
  "url": "http://httpbin.org/get"
}
```

Данный сервер возвращает отправленные данные клиентом обратно в теле ответа:

- <https://httpbin.org/ip> Возвращает IP-адрес.
- <https://httpbin.org/anything> Возвращает почти всё из нижеперечисленного.
- <https://httpbin.org/user-agent> Возвращает user-agent.
- <https://httpbin.org/headers> Возвращает словарь с заголовками.
- <https://httpbin.org/get> Возвращает данные из GET запроса.
- <https://httpbin.org/post> Возвращает данные из POST запроса.
- <https://httpbin.org/put> Возвращает данные из PUT запроса.
- <https://httpbin.org/delete> Возвращает данные из DELETE запроса
- <https://httpbin.org/gzip> Возвращает данные сжатые в gzip.
- <https://httpbin.org/status/:code> Возвращает статус-код HTTP равный параметру :code.
- <https://httpbin.org/redirect/:n> Перенаправляет n раз.
- <https://httpbin.org/relative-redirect/:n> Относительное перенаправление n раз.
- <https://httpbin.org/cookies> Возвращает переданные cookie.
- <https://httpbin.org/cookies/set/:name/:value> Устанавливает указанные cookie (name=value).
- <https://httpbin.org/stream/:n> Передаёт от n до 100 строк.
- <https://httpbin.org/delay/:n> Задерживает ответ от n до 10 секунд

Протокол	Год	Описание
HTTP/0.9	1991	HTTP был предложен Тимом Бернерсом-Ли. Были задокументированы основные синтаксические и семантические положения.
HTTP/1.0	1996	В мае 1996 года для практической реализации HTTP был выпущен информационный документ RFC 1945
HTTP/1.1	1999	Новым в этой версии был режим «постоянного соединения»: TCP-соединение может оставаться открытым после отправки ответа на запрос, что позволяет посылать несколько запросов за одно соединение. Клиент теперь обязан посылать информацию об имени хоста, к которому он обращается
HTTP/2	2015	HTTP/2 является бинарным. Среди ключевых особенностей: мультиплексирование запросов, расстановка приоритетов для запросов, сжатие заголовков, загрузка нескольких элементов параллельно посредством одного TCP-соединения, поддержка push-уведомлений со стороны сервера
HTTP/3	2018 – н.в.	Основан на UDP вместо TCP в качестве транспортного протокола. Как и HTTP/2, он не объявляет устаревшими предыдущие основные версии протокола



Южно-Уральский
государственный
университет

Национальный
исследовательский
университет

КРОК

Спасибо за внимание!



КРОК

Челябинск, ул. Карла Маркса, д. 38