

Основы клиент-серверного взаимодействия



Владимир
Тесленко



Владимир Тесленко

Automation QA Engineer в компании Unicore

 teslvova@gmail.com

 [@teslvova](https://t.me/teslvova)




План занятия

1. [Как работает интернет?](#)
2. [Что такое сервер и как он работает?](#)
3. [Основы веб-технологий](#)
4. [Dev Tools \(Девелоперская Консоль / Консоль Разработчика\)](#)



Вспоминаем прошлые занятия

- Цели тестирования;
- Цикл тестирования ПО;
- Методы и виды тестирования.



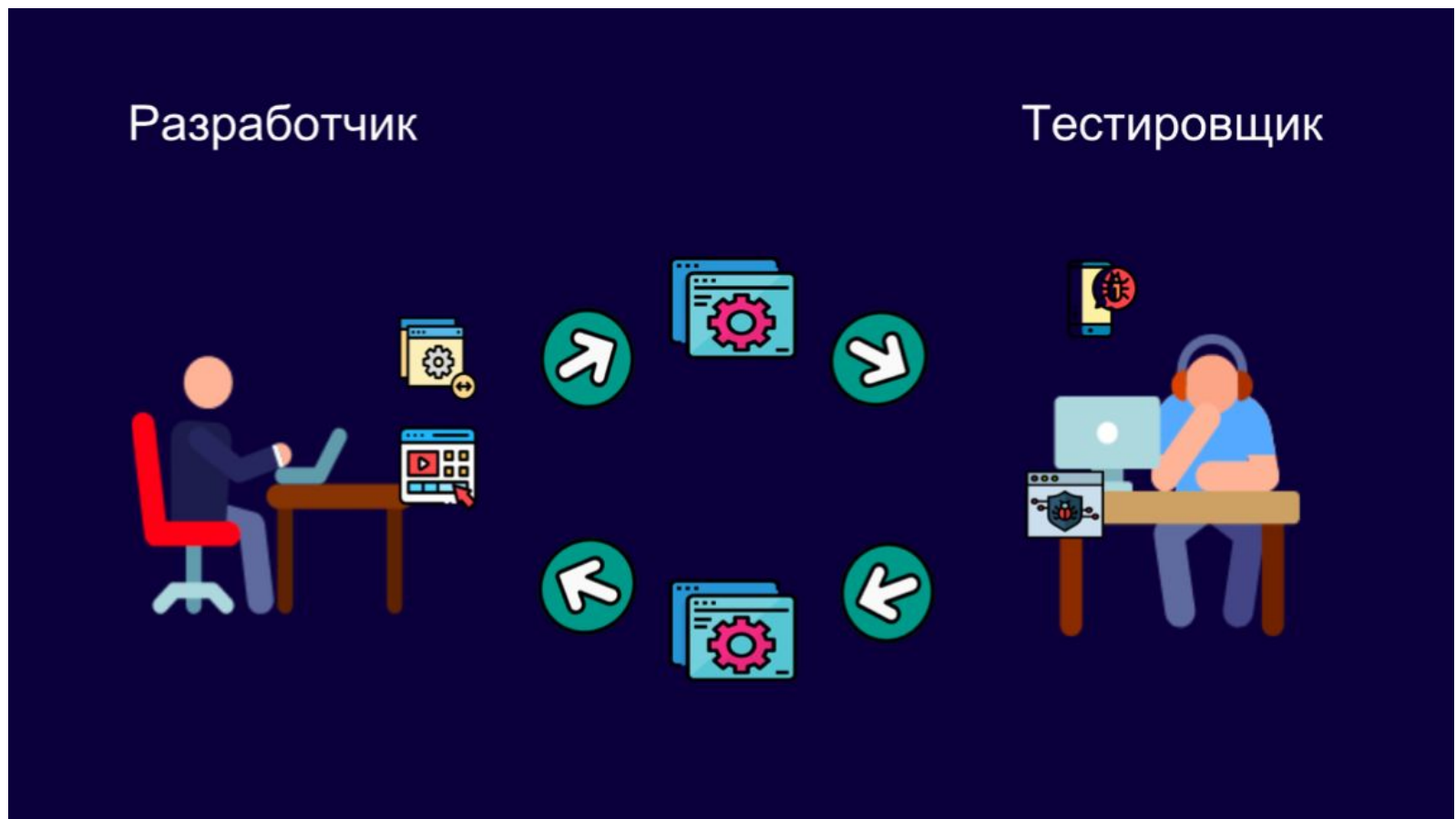
Зачем QA-инженеру понимать основы клиент-серверного взаимодействия?

Большинство современного ПО строится на клиент-серверном взаимодействии.

Тестировщику важно быть в контексте этой архитектуры, понимать принципы её работы, основные преимущества и недостатки.

Это позволит не только лучше проводить испытания в ходе своей работы, но и при необходимости понимать, каким образом найденные дефекты нужно локализовывать, правильно определяя, на чьей они стороне.

Зачем QA-инженеру понимать основы клиент-серверного взаимодействия?





Что нас сегодня ждет?

- Понятия Frontend и Backend
- Как работает Интернет?
- Что такое Клиент?
- Что такое Веб-Сервер
- Модель взаимодействия клиент-сервер
- Балансировщик
- Базы данных
- Основные понятия про URL | HTTP/HTTPS
- Основы JSON | XML
- Коды ответов клиента и сервера
- Типы запросов
- Основы SQL
- Logs
- Локализация багов
- Основы HTML, CSS, JavaScript
- Девелоперская консоль



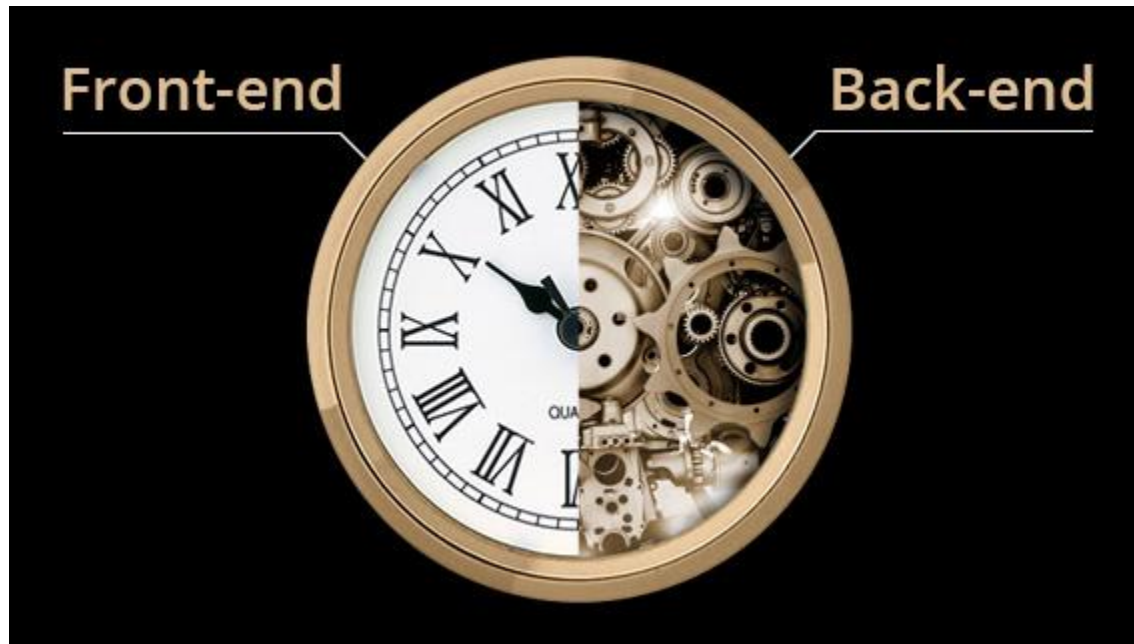
Frontend и Backend

Frontend — система скриптов, которая работает на стороне пользователя (в браузере). Используются технологии HTML, CSS и JavaScript.

- создание дизайн-макета сайта
- верстка сайтов и шаблонов для CMS
- привязка к пользовательскому интерфейсу специальных скриптов, отвечающих за визуализацию и web-анимацию.

Frontend и Backend

К **Backend** (бэкенд) относится работа самой программы: как работает ее код, как взаимодействует приложение с сервером, работа с базами данных, и множество других действий, скрытых от глаз пользователя.





Как работает интернет?

Интернет – глобальная сеть соединённых между собой компьютеров, если оставить за скобками все дополнительные устройства (роутеры или свитчи).

В этой сети нет определённого центра, все компьютеры равноправны между собой.



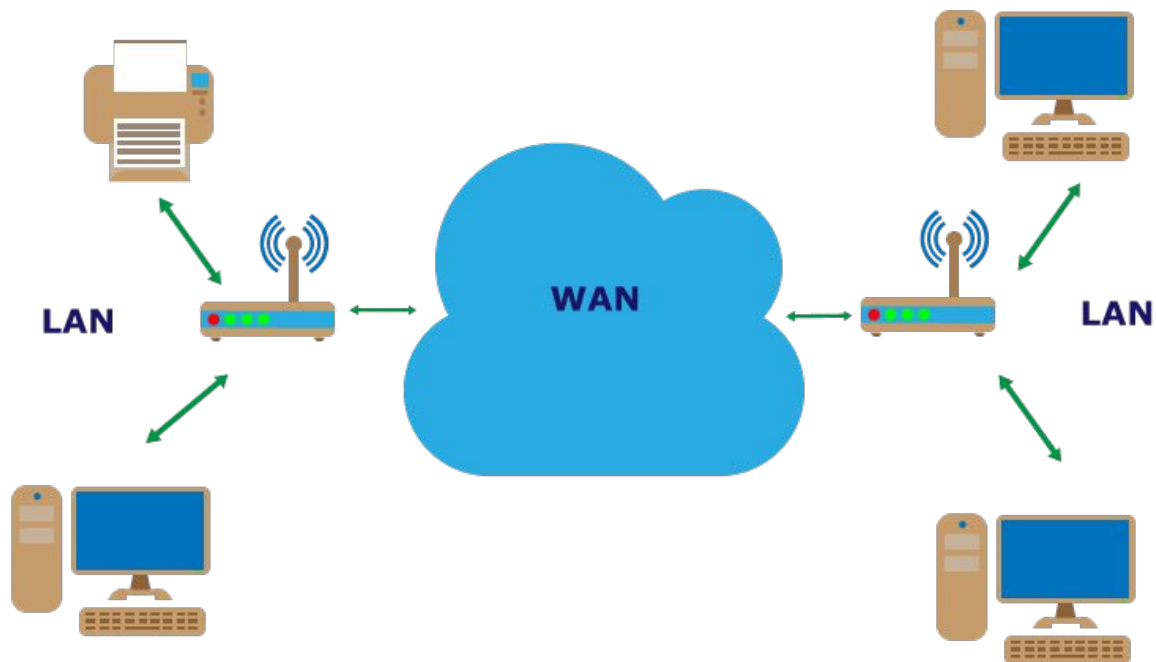
Как работает интернет?

Компьютеры соединяются между собой напрямую с помощью локальной сети (LAN) или через глобальное соединение (WAN).

LAN (Local Area Network) - это локальная сеть. Проще говоря, это компьютеры, которые соединены между собой на не очень большом расстоянии.

WAN (Wide Area Network) – это глобальная компьютерная сеть интернет.

Как работает интернет?

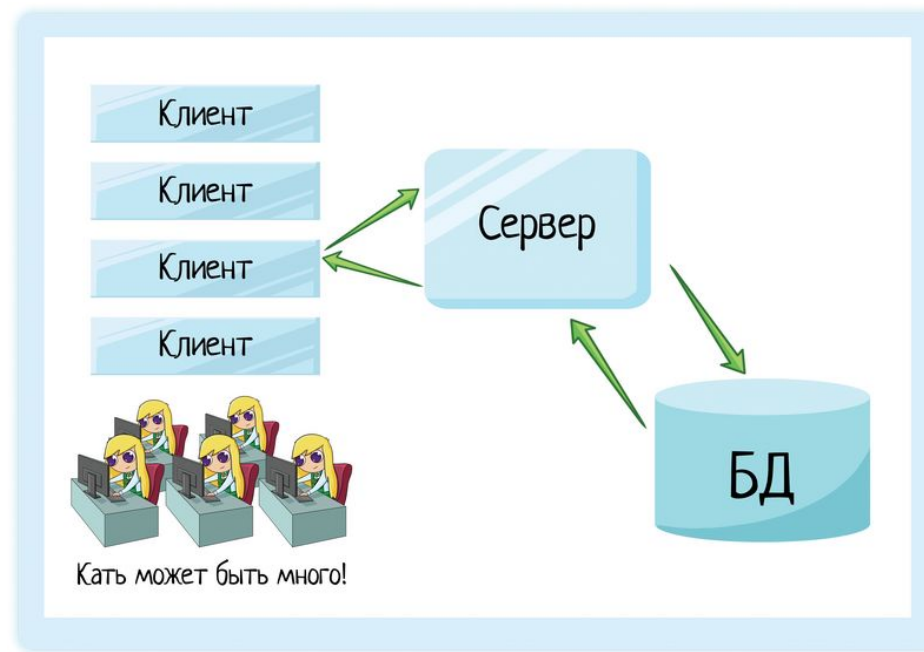


Клиент

Под клиентом понимают обычный браузер (их может быть много).

Клиент – процесс, который запрашивает обслуживание от сервера.

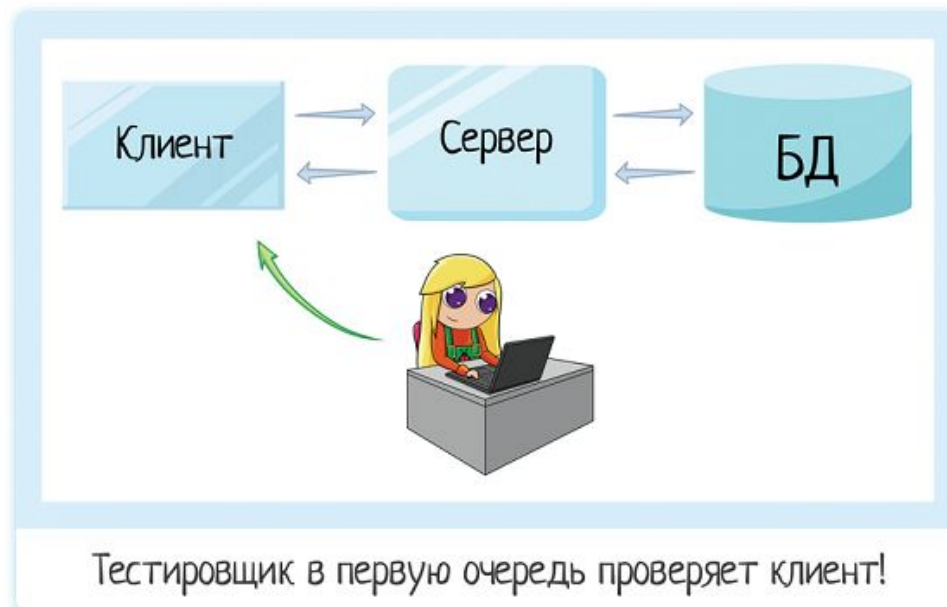
Процесс не является клиентом по каким-то параметрам своей структуры, он является клиентом только по отношению к серверу.



Клиент

Клиентом для сервера может выступать и другой сервер.

При взаимодействии клиента и сервера инициатором диалога с сервером, как правило, является клиент. Сервер сам не инициирует совместную работу. Это не исключает, однако, того, что сервер может извещать клиентов о каких-то зарегистрированных им событиях.



Веб-сервер

Веб-сервер - это сервер, который может получать входящие веб-запросы, а также знает, как их следует обрабатывать и отвечать на них.

Некоторые запросы являются статическими (html-файлы, изображения и т.д.), другие - динамическими. В случае динамических запросов веб-сервер будет знать, куда направлять обработку запроса.



Веб-сервер

Веб-сервер предназначен для обслуживания содержимого HTTP.

Его задача - принять запрос и понять, что с ним делать: определить, к какому файлу перешел запрос, отправить его по нужному адресу, обработать этот файл и выдать ответ пользователю.

Дополнительные задачи веб-сервера:

- реализация журнала ошибок и обращений (логи);
- какую папку с файлами надо обрабатывать, по каким правилам;
- аутентификация и авторизация (может определить, кто обращается к файлам и выдает им доступ к ним).



Веб-сервер

Аутентификация — это установление соответствия лица названному им идентификатору, а авторизация — предоставление этому лицу возможностей в соответствии с положенными ему правами или проверка наличия прав при попытке выполнить какое-либо действие.

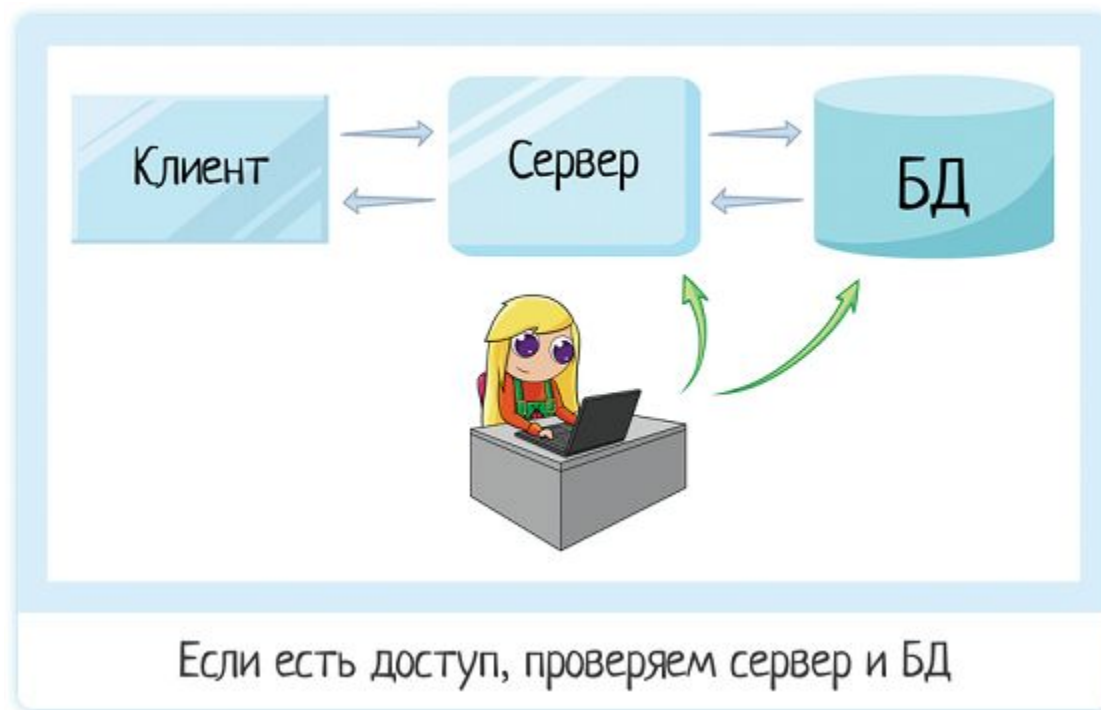
Модель взаимодействия клиент-сервер

Если мы посмотрим на архитектуру с позиции сайта, то первый уровень можно считать браузером, с помощью которого посетитель заходит на сайт. Второй уровень – это ПО сервера, а третий уровень – это база данных.



Пример текста на слайде

И конечно, когда тестировщик имеет опыт работы с кодом и базой данных, то он начинает проверять информацию на сервере и в БД.



Балансировщик

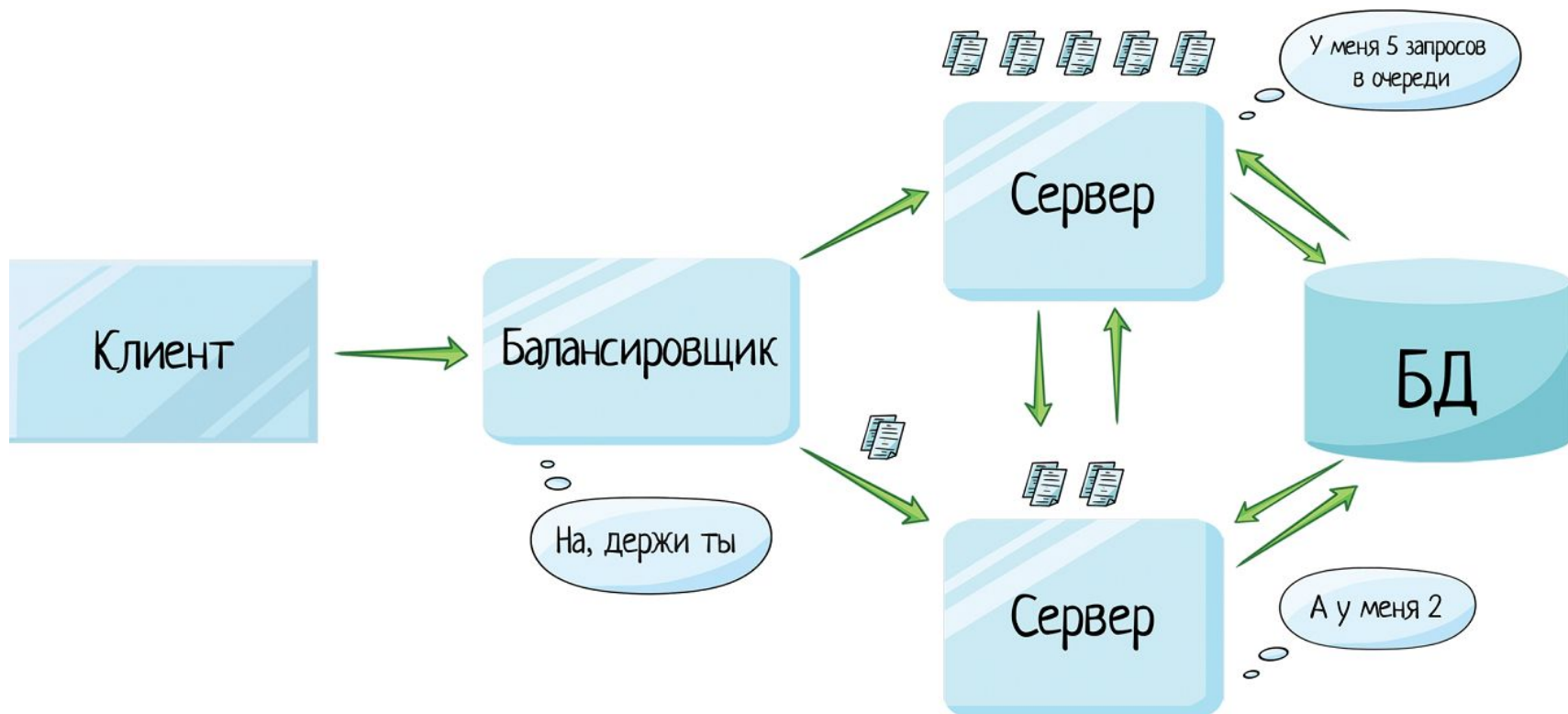
Для бесперебойной работы веб-приложений важно, чтобы взаимодействие между клиентом и серверами было надежным.

Для этого система имеет несколько серверов, между клиентом и серверами установлен балансировщик, который обеспечивает качественное взаимодействие между ними.



Балансировщик

Если один сервер будет очень нагружен, второй сможет взять часть обработки данных на себя, и приложение не будет «тормозить».



Балансировщик

Если один из серверов перестанет работать, то балансировщик определит это и будет работать с другими серверами, а пользователь даже не заметит, что в системе произошла проблема.

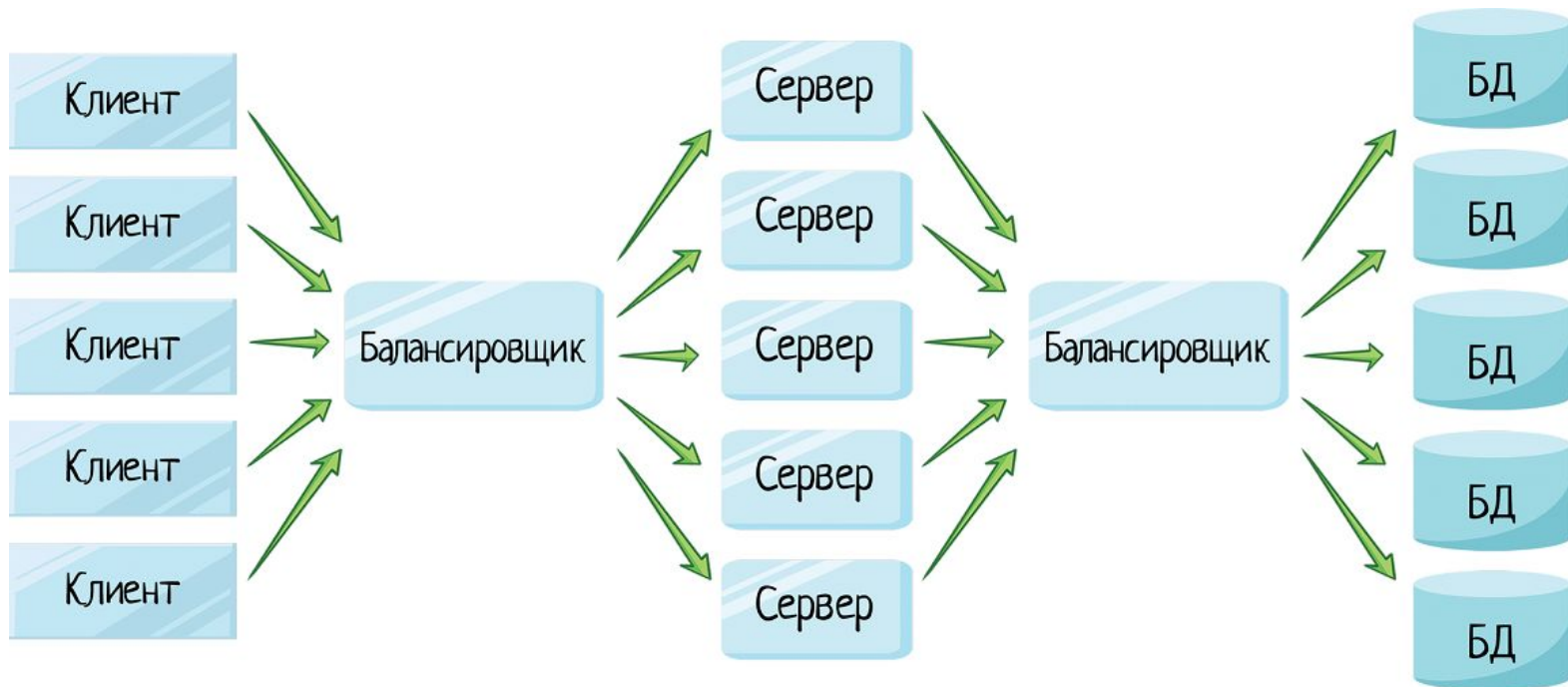
Тестировщик периодически перепроверяет работу серверов и балансировщика, чтобы понимать, что все работает исправно.



Многоуровневая модель

Суть многоуровневой архитектуры в том, что запрос клиента обрабатывается сразу несколькими серверами.

Хороший пример многоуровневых приложений - Facebook, VK, LinkedIn.





СУБД

СУБД (система управления базами данных) – комплекс ПО, с помощью которого можно создавать базы данных (БД) и проводить над ними различные операции: обновлять, удалять, выбирать, редактировать и т. д. СУБД гарантирует сохранность, целостность, безопасность хранения данных и позволяет выдавать доступ к администрированию БД.

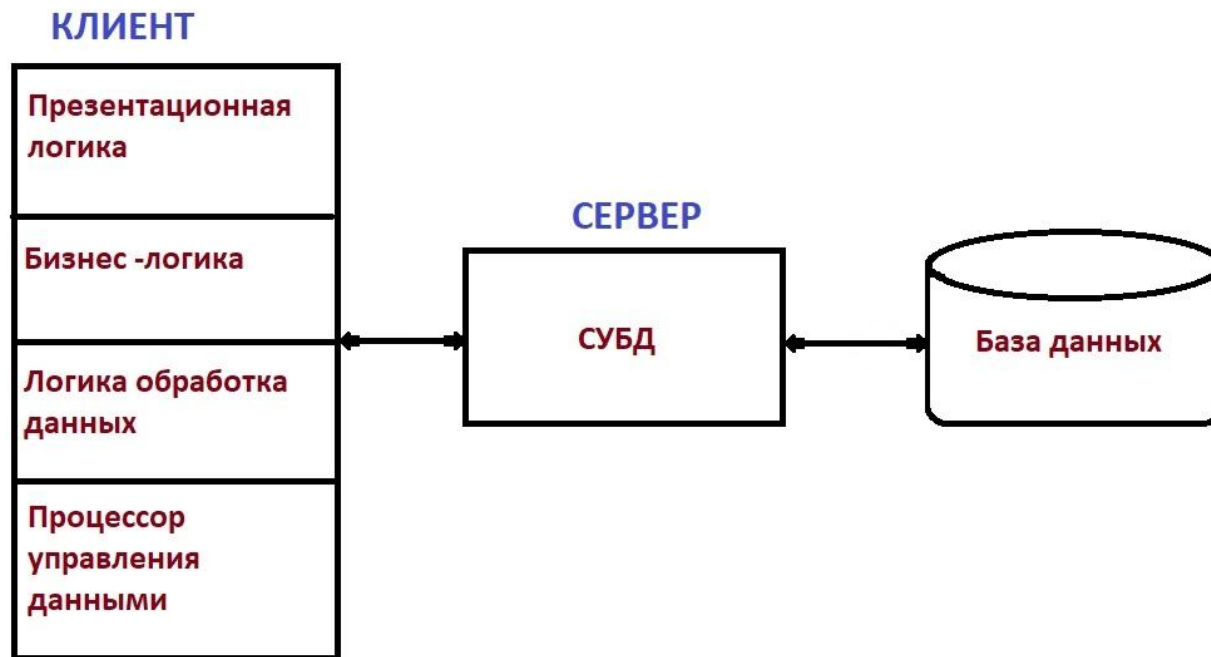


СУБД

- Для выполнения операций используется механизм запросов.
- Результатом выполнения запросов является либо отобранное по определенным критериям множество записей, либо изменения в таблицах.
- Запросы к базе формируются на специально созданном для этого языке, который так и называется «язык структурированных запросов» (SQL – Structured Query Language).

СУБД

Распространенные СУБД: Oracle Database, PostgreSQL, MySQL



URL | HTTP/HTTPS


URL (Uniform Resource Locator) — это адрес, который выдан уникальному ресурсу в интернете. В теории каждый корректный URL ведет на уникальный ресурс.

- Например, ресурсами могут быть HTML-страница, CSS-файл, изображение.
- На практике существуют некоторые исключения, когда, например, URL ведет на ресурс, который больше не существует или который был перемещён.
- Ресурс, доступный по URL, и сам URL обрабатываются веб-сервером.

URL | HTTP/HTTPS

URL состоит из различных частей, некоторые из которых являются обязательными, а некоторые - факультативными. Рассмотрим наиболее важные части на примере:

http://www.example.com:80/path/



Protocol

URL | HTTP/HTTPS

«**http://**» — часть URL, которая указывает браузеру, какой протокол использовать.

Обычно это HTTP-протокол или его безопасная версия - HTTPS.

Интернет требует эти 2 протокола, но браузеры часто могут использовать и другие протоколы, например **mailto:** (чтобы открыть почтовый клиент) или **ftp:** для запуска передачи файлов, так что не стоит удивляться, если вы вдруг увидите другие протоколы.



HTTP/HTTPS

Для передачи веб-страниц используется **HyperTextTransferProtocol (HTTP)** — протокол передачи гипертекста. Текст веб-страниц назван гипертекстом из-за того, что веб-страницы связаны между собой ссылками, образуя тем самым гиперпространства.

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.

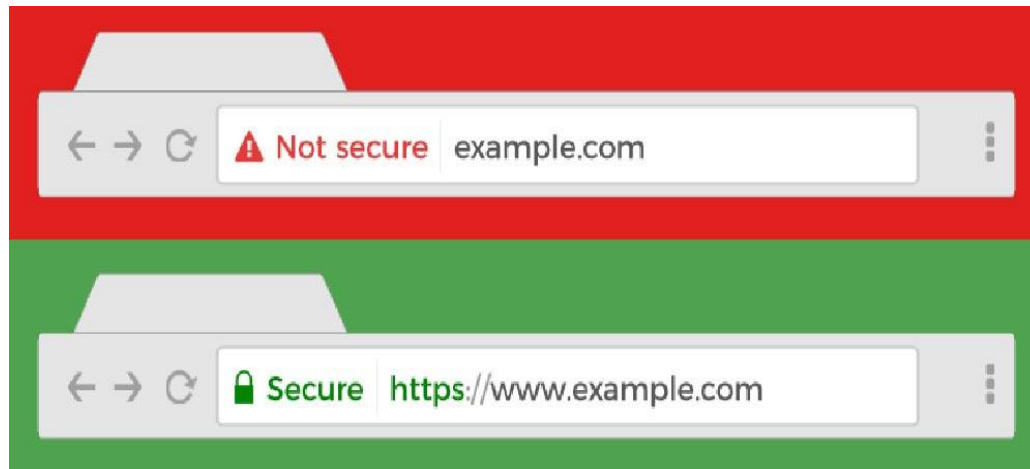


HTTP/HTTPS

TLS (transport layer security – Протокол защиты транспортного уровня), как и его предшественник **SSL** (secure sockets layer – слой защищённых сокетов) – это криптографические протоколы, обеспечивающие защищённую передачу данных между узлами в сети Интернет.

TLS и SSL используют асимметричное шифрование для аутентификации для сохранения целостности сообщений.

HTTP/HTTPS





HTTP/HTTPS

Техническое отличие: **HTTP** обычно использует порт соединения **80**, тогда как **HTTPS** — порт **443**. При необходимости системный администратор может указать другой порт, но они всегда будут разными для HTTP и HTTPS протоколов.

Функции HTTP-протокола

- передаёт содержимое веб-страниц;
- передаёт ответ сервера на запрос (например, «всё ОК», «страница не найдена», «нет прав» и т.д.);
- передаёт «заголовки» технической информации, например, кодировку или cookies;
- передаёт данные со стороны клиента, в том числе загружает файлами и многое другое.

JSON | XML

JSON (JavaScript Object Notation) — это текстовый формат обмена структурированными данными, основанный на JavaScript.

Его легко преобразовать в структуру данных для большинства языков программирования: числа, строки, логические переменные, массивы.

JSON Example:

```
{
  "employees": [
    { "firstName": "John", "lastName": "Doe" },
    { "firstName": "Anna", "lastName": "Smith" },
    { "firstName": "Peter", "lastName": "Jones" }
  ]
}
```

JSON | XML

Каждый раз, когда мы вводим пароль на Facebook или в другом веб-приложении, сервер воспринимает это как следующий текст:

```
{  
  "login": "Netoologe@gmail.com",  
  "password": "net190!93theBestF0RStu41ent$",  
  "admin": true  
}
```

JSON | XML

XML (eXtensible Markup Language — расширяемый язык разметки) — это язык описания данных. Он используется для хранения и передачи информации в удобном для человека и компьютера виде.

XML Example:

```
<employees>
  <employee>
    <firstName>John</firstName> <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>
```



Разница между JSON и XML

Наиболее частое распространенное использование JSON | XML — пересылка данных от сервера к браузеру.

XML – это язык разметки, который используется для добавления дополнительной информации в обычный текст, а **JSON** – это способ представления структурированных данных в удобном формате.

Разница между JSON и XML

XML

```
<empinfo>
  <employees>
    <employee>
      <name>James Kirk</name>
      <age>40</age>
    </employee>
    <employee>
      <name>Jean-Luc Picard</name>
      <age>45</age>
    </employee>
    <employee>
      <name>Wesley Crusher</name>
      <age>27</age>
    </employee>
  </employees>
</empinfo>
```

JSON

```
{ "empinfo" :
  {
    "employees" : [
      {
        "name" : "James Kirk",
        "age" : 40,
      },
      {
        "name" : "Jean-Luc Picard",
        "age" : 45,
      },
      {
        "name" : "Wesley Crusher",
        "age" : 27,
      }
    ]
  }
}
```

Коды ответов клиента и сервера

HTTP Status Codes





Коды ответов клиента и сервера

1x — информационный код, отвечающий за передачу данных. Такие коды временны и показывают, что запрос принят и обрабатывается.

2x — код успешной обработки запроса. Сервис получил и обработал запрос.

3x — код редиректа. Сервер сигнализирует, что для выполнения запроса нужно предпринять дополнительные действия, например, перейти на другой адрес. По какой-то внутренней причине сервер не может выполнить пользовательский запрос.



Коды ответов клиента и сервера

4x — клиентская ошибка. Ошибка на стороне клиента. Возможно, пользователь что-то сделал неправильно, и поэтому запрос не может быть успешно обработан.

5x — серверная ошибка. По какой-то внутренней причине сервер не может выполнить пользовательский запрос.

Типы запросов

Method	Description
GET	Request to read a Web page
HEAD	Request to read a Web page's header
PUT	Request to store a Web page
POST	Append to a named resource (e.g., a Web page)
DELETE	Remove the Web page
TRACE	Echo the incoming request
CONNECT	Reserved for future use
OPTIONS	Query certain options



Типы запросов

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

POST позволяет передавать большие объемы данных в бинарном виде, т. е. без искажений и изменений передаваемых данных.

PUT загружает содержимое запроса на указанный в запросе URL. Если по заданному URI ресурса нет, то сервер создает его, возвращая статус 201 (Created).



Типы запросов

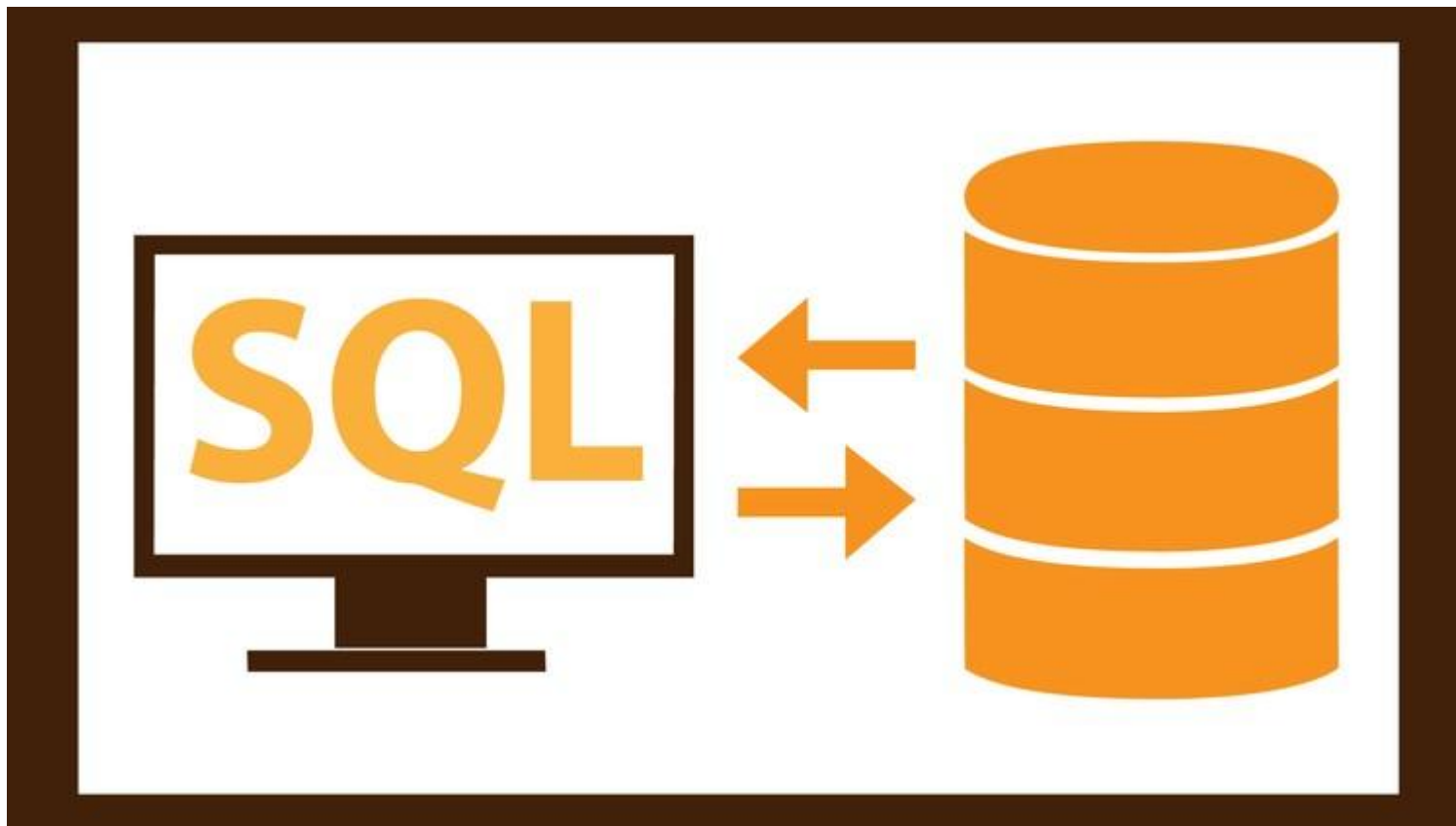
DELETE удаляет указанный ресурс.

OPTIONS используется для описания параметров коммуникации между клиентом и сервером.

HEAD аналогичен методу GET, однако в ответе сервера содержится только заголовок, без тела. Обычно применяется для того, чтобы проверить, существует ли ресурс по указанному адресу, а также не изменился ли он с момента последнего обращения.

CONNECT преобразует соединение запроса в прозрачный TCP/IP-туннель.

ОСНОВЫ SQL



ОСНОВЫ SQL

SQL (Structured Query Language — структурированный язык запросов) — это язык управления базами данных.

```
SELECT * FROM table_name;
```

```
UPDATE table_name SET column1 = value1, column2 = value2, WHERE id=2;
```

```
DELETE FROM table_name WHERE id=10;
```



Виды SQL-запросов

SELECT - извлекает данные из базы данных

UPDATE - обновляет данные в базе данных

DELETE - удаляет данные из базы данных

INSERT INTO - вставляет новые данные в базу данных

CREATE DATABASE - создает новую базу данных

ALTER DATABASE - изменяет базу данных

CREATE TABLE - создает новую таблицу

ALTER TABLE - изменяет таблицу

DROP TABLE - удаляет таблицу

Как работает запрос к серверу?

GET	POST	PUT	PATCH	DELETE
find	create	replace	update	destroy
SELECT	INSERT	UPDATE	UPDATE	DELETE

Logs

```

Терминал Файл Правка Вид Поиск Терминал Справка

1 [|||||69.2%] Tasks: 194, 234 th
2 [|||||85.1%] Load average: 2.25
3 [|||||74.0%] Uptime: 00:10:18
4 [|||||98.1%]
Mem[1371/3952M]
Swp[ 0/3147MB]

PID USER      PRI  NI  VIRT   RES   SH
9500 vladimir  39   19 23936 6108 356
F1help F2Setup F3Search F4Filter F5Free

4 /          м ,          9
p          A % \          G
#          :          A a Q          ,
}          C w d - M          q
Z          A l E o v P          &
]          z n | % % " #          y *
W          ] = Q i t l E 9 #          %
$ y s R / l # Q ! Y _ ^ q
l d # ! 7 q ) ) * = g A
# ; P W b i U k 8 W 5 f
m          g 4 | % 5 & 1 : j

XDG-DESKxdg-desktopXDG-DESKTOP-MENU(1)
NAME
xdg-desktop-menu - command line
tool for (un)installing desktop
menu items

SYNOPSIS
xdg-desktop-menu install
[--noupdate]
line 1 (press h for help or q to quit)

Ссылки: 1
Доступ: (0644/-rw-r--r--) Uid: ( 0/
root) Gid: ( 0/ root)
Доступ: 2015-01-29 09:22:01.819592922 +0
100
Модифицирован: 2015-01-29 09:22:01.81959
2922 +0100
Изменён: 2015-01-29 09:22:01.819592922 +
0100
Создан: -

М
ENOTTY 25 Неприменимый к данному устройст
ву ioctl
EISDIR 21 Это каталог
ENOKEY 126 Требуемый ключ недоступен
EDOTDOT 73 Специфичная для RFS ошибка
ENOSYS 38 Функция не реализована
ENOTEMPTY 39 Каталог не пуст
EMEDIUMTYPE 124 Неправильный тип носителя

A: 50,4 V: 50,4 A-V: -0,000 ct: 0,00
%.8:
58.8
X..
% t%

victor echo india golf Whiskey echo unif
orm golf
Juliett india charlie Oscar romeo golf O
scar november NINE
Echo charlie kilo alfa romeo delta unif
orm kilo
sierra echo tango Hotel yankee kilo foxt
rot alfa tango
Hotel echo foxtrot echo november India g
olf yankee alfa sierra

card0
0 directories, 1 file
/dev/fd
0 -> /dev/pts/38
1 -> /dev/pts/38
2 -> /dev/pts/38
3 -> /proc/10089/fd

1 directory, 3 files

#include <string.h>
#include <errno.h>
#include <ctype.h>
struct security_class_mapping {
const char *name;
const char *perms[sizeof(unsigne
d) * 8 + 1];
};
#include "classmap.h"
#include "initial_sid_to_string.h"
< чтения] 138L, 3517C 14,1 2%

lo
Interfaces | RX bps | pps | % TX bps | pps | %
lo | 0 | 0 | 0 | 0 | 0
etho | 3B | 0 | 2B | 0 | 0
qdisc none (pfifo_fast) | 0 | 0 | 2B | 0 | 0
----- Increase screen height to see graphical statistics -----
----- Press d to enable detailed statistics -----
----- Press i to enable additional information -----
Thu Jan 29 09:22:50 2015 Press ? for help
10m 2.25 4x2.3GHz 3.9G35% 2015-01-29 09:22:50

```

Зачем QA-инженеру понимать Логи?

Анализируя данные сервера или браузера, QA-инженер всегда использует запротоколированную информацию от пользователя или от сервера, которая называется «Логи».

Логи - это история операций, в которую заносятся данные о действиях пользователей и программы.

Данные в логах решающие для тестировщика при определении характера бага.

Зачем QA Engineer должен понимать Логи?

The screenshot displays the Chrome DevTools interface for the URL `netology.ru/`. The **Network** tab is active, showing a list of requests. The selected request is from `mc.yandex.ru/webvisor` with a status code of 200. The **Headers** sub-tab is open, displaying request details: **Request URL**: `https://netology.ru/`, **Request Method**: `GET`, **Status Code**: `200`, **Remote Address**: `104.26.8.143:443`, and **Referrer Policy**: `strict-origin-when-cross-origin`. Below the headers, the **Response Headers** section is visible. The **Console** tab is also open, showing several messages: a deprecation warning for `Synchronous XMLHttpRequest`, an `Uncaught ReferenceError: $ is not defined` error from `caltatscript.aspx?id=1012018:94`, and a message about a script being refused to execute from `https://top-fwz1.mail.ru/js/code.js` due to its MIME type.



Локализация дефекта

Когда собрана общая информация о системе, мы можем локализовать баг и собрать необходимое максимальное количество информации о его воспроизведении:

- **Выявить причины возникновения дефекта**

Например, не проходит восстановление пароля. Необходимо выявить, откуда приходит запрос на сервер в неверном формате — от backend либо frontend. Эти данные можно получить из логов.



Локализация дефекта

- **Проанализировать возможность влияния найденного дефекта на другие области**

Например, в одной из форм, которую редко используют, возникает ошибка при нажатии на кнопку «Редактировать».

- **Отклонение от ожидаемого результата**

Нужно проверить, соответствует ли результат тестирования ожидаемому результату. Справиться с этой задачей помогает техническое задание.



Локализация дефекта

- **Исследовать окружение**

Необходимо воспроизвести баг в разных операционных системах (iOS, Android, Windows и т.д.) и браузерах (Google Chrome, Mozilla, Internet Explorer и др.).

При этом нужно проверить требования к продукту, чтобы выяснить, какие системы должны поддерживаться.



Локализация дефекта

- **Проверить на разных устройствах**

Например, desktop-приложение предназначено для использования на компьютерах, поэтому зачастую нет необходимости тестировать его на мобильных устройствах.

Для смартфонов в идеале должна быть разработана отдельная мобильная версия, которую, в свою очередь, нет смысла тестировать на компьютерах. Кроме того, есть web-приложения, которые могут работать и на компьютерах, и на мобильных устройствах, а тестировать их нужно на всех типах устройств.



Локализация дефекта

- **Проверить в разных версиях ПО**

Для того, чтобы не запутаться в реализованных задачах, в разработке используют версию ПО.

- **Проанализировать ресурсы системы**

Возможно, дефект был найден при нехватке внутренней или оперативной памяти устройства. В таком случае баг может воспроизводиться на идентичных устройствах по-разному.



Frontend

Основы HTML

HTML (HyperText Markup Language — язык гипертекстовой разметки) не является языком программирования, это язык разметки.

Он используется, чтобы сообщать вашему браузеру, как отображать веб-страницы, которые вы посещаете.

Разметка

```
<ol>  
<li>HTML</li>  
<li>CSS</li>  
<li>JavaScript</li>  
<li>PHP</li>  
</ol>
```

Результат

1. HTML
2. CSS
3. JavaScript
4. PHP

Оснoвы HTML

Структура документа HTML

- ▶ `<html>`
- ▶ `<head>`
- ▶ `<title>Заголовок</title>`
- ▶ `<meta charset="UTF-8">`
- ▶ `<link rel="icon" href="favicon.ico">`
- ▶ `</head>`
- ▶ `<body>`
- ▶ Тело документа
- ▶ `</body>`
- ▶ `</html>`



Основы CSS

CSS (Cascading Style Sheets — каскадные таблицы стилей) — формальный язык описания внешнего вида документа. Он позволяет применять стили выборочно к элементам, т.е. оформлять внешний вид веб-страниц, написанных с помощью языков разметки HTML.

HTML



HTML - CSS



ОСНОВЫ CSS

Inline CSS

```
<p style="color: blue;">This is a paragraph.</p>
```

Internal CSS

```
<head>  
  <style type = text/css>  
    body {background-color: blue;}  
    p { color: yellow;}  
  </style>  
</head>
```

External CSS

```
<head>  
  <link rel="stylesheet" type="text/css" href="style.css">  
</head>
```

Основы JavaScript

JavaScript — мультипарадигменный язык программирования, позволяющий создавать скрипты, которые встраиваются в HTML-страницы и выполняются в браузере посетителя страницы, а не на сервере.

Все современные браузеры поддерживают язык JavaScript.

```
<script type="text/javascript">
  function getInspiration() {
    if (morningDay == "depressed")
      start.Coding & get.Awesome;
  }
</script>
```

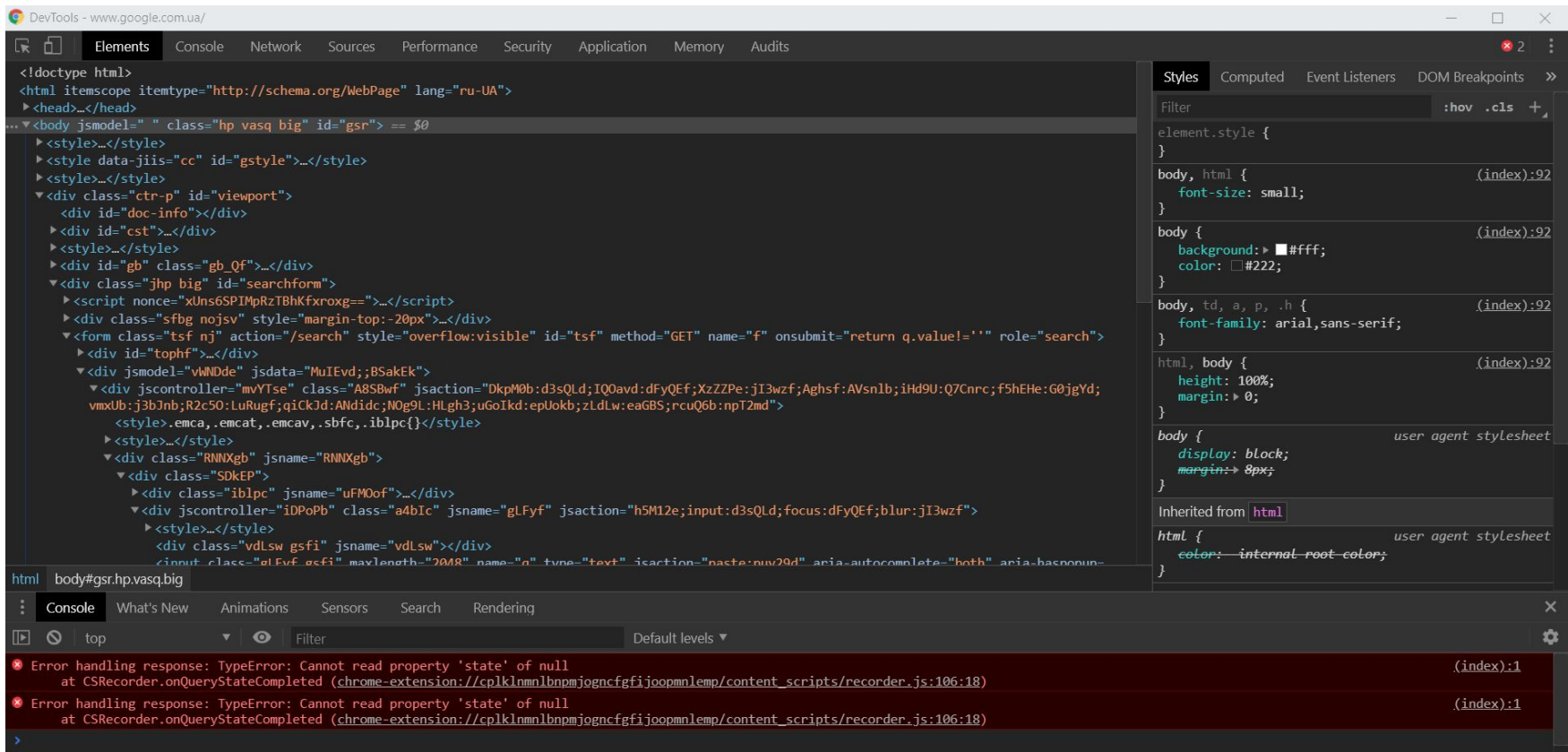


Что такое девелоперская консоль?

Консоль разработчика помогает:

- читать ошибки в удобном виде;
- получать массу полезной информации о выполнении скриптов;
- запускать команды JavaScript и пр.

Что такое девелоперская консоль?





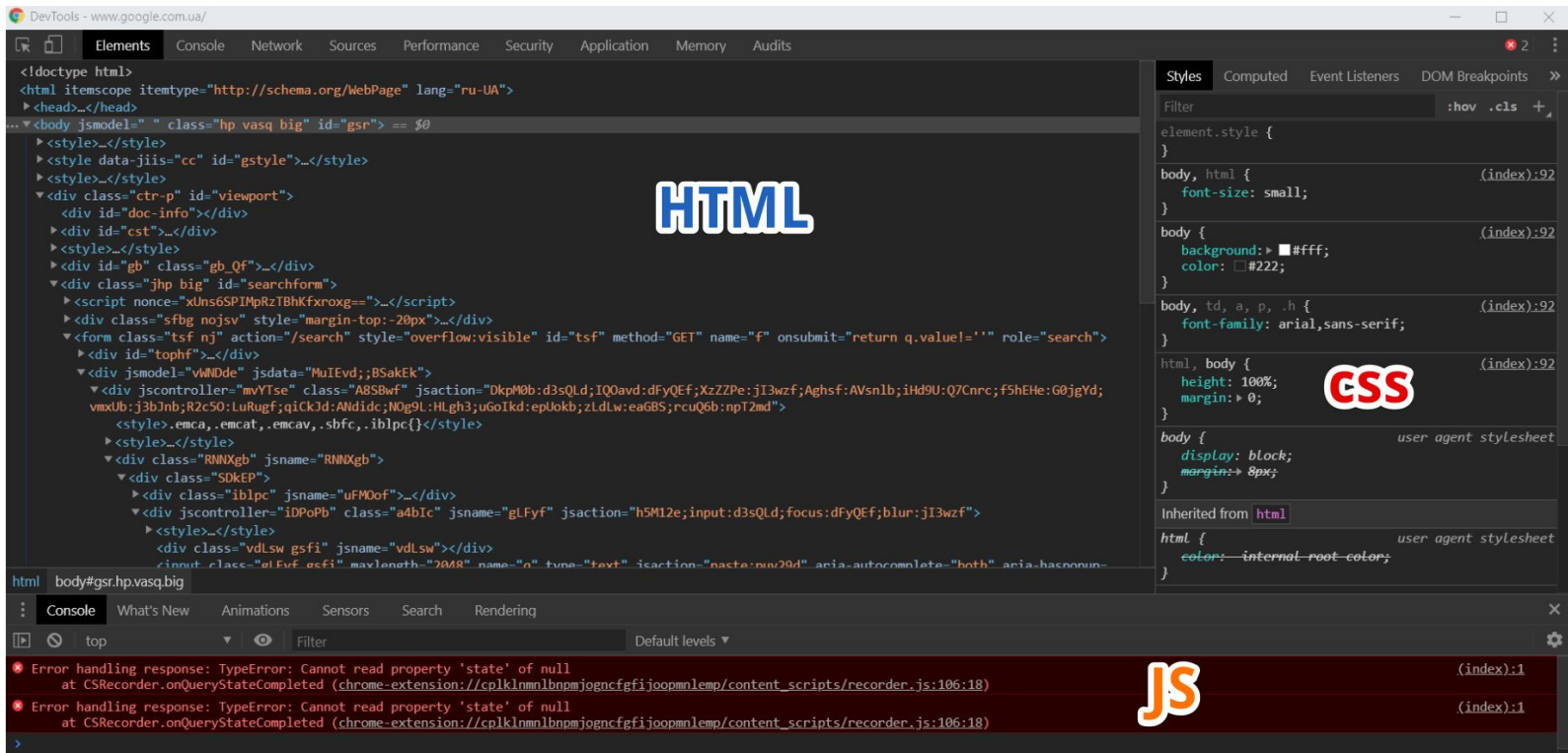
Что такое девелоперская консоль?

В каждом браузере есть свой инструмент разработчика — это то, что многие называют консолью.

Консоль в браузере — это инструмент, с помощью которого можно:

- посмотреть наполнение страницы, выводимой браузером;
- найти существующие ошибки (что чаще всего и делают тестировщики);
- исправить эти ошибки просто и быстро (что чаще всего делают девелоперы);
- замерить различные показатели и манипулировать страницей.

Что такое девелоперская консоль?



Дополнительные материалы

- [Что такое IP адрес](#)
- [Разница между IPv4 и IPv6](#)
- [TCP/IP: что это и зачем это тестировщику](#)
- [DNS сервер - что это и как работает?](#)
- [Что такое Ajax и зачем он нужен?](#)
- [Отличие архитектуры «клиент-сервер» от архитектуры «файл-сервер»](#)
- [Протокол HTTP](#)



Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

**Задавайте вопросы и
пишите отзыв о лекции!**

Владимир Тесленко

✉ teslvoya@gmail.com

|  [@teslvoya](https://t.me/teslvoya)