МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
"МИРЭА – Российский технологический университет"

РТУ МИРЭА

Институт информационных технологий
Кафедра инструментального и прикладного программного обеспечения

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 3
по дисциплине
«Разработка серверных частей интернет ресурсов»

Выполнил студент группы ИКБО-01-20 Баикин К. Е.

Принял Волков М.Ю.

Практическая работа выполнена «_ _»_____2022 г. _____

«Зачтено» «__»_____2022 г. _____

Москва 2022

СОДЕРЖАНИЕ

Цель работы

Получить навыки создания сложных конфигураций серверов и их оптимизации.

Задание на практическую работу

Создать сложную серверную конфигурацию, состоящую из связки apache+nginx+php+База данных. Возможно использование связки apache+php как единый компонент. В данной конфигурации предполагается создание как минимум 3 элементов(контейнеров) или использование как основы серверной конфигурации, созданной в практической работе No1. В этой конфигурации предполагается акселерированное проксирование без кэширования.

Предполагается, что сервер nginx будет отображать статический контент, а apache динамический и в связке мы получим быстродейственную и эффективную систему. Также по необъяснимым обстоятельствам следует изменить root- директории с базовых на другие. Для доступа к администрированию предусмотреть базовую авторизацию и аутентификацию с применением httaccess и хранением пользователей в БД (без использования htpasswd). Для тестирования данной конфигурации предполагается создать тестовое веб-приложение на тему по варианту:

1. Кофейня
2. *Строительный магазин*
3. Автосервис
4. Магазин электроники
5. Портфолио
6. Библиотека
7. Ресторан
8. Погода
9. Ломбард
10. Магазин игрушек
11. Индивидуальная тема

Тестовое веб-приложение предполагает создание как минимум 2 веб-страниц со статическим контентом и двух веб-страниц с динамическим контентом: взятом из базы данных, например.

Выполнение практической работы

Структура проекта представляет из себя 3 контейнера, каждый из которых выполняет свою функцию. Первый содержит сервер apache, интерпритатор языка php, динамические шаблоны php и служебные скрипты php, данный контейнер обслуживает динамический контент. Второй содержит сервер nginx, статические html страницы и служебные javascript скрипты, он обслуживает статический контент. Третий же содержит в себе лишь базу данных mysql.

Первый контейнер обслуживает такой инамический контент как: каталог ценностей, которые продает ламбард, страница просмотра ценности и страница администрирования, доступ к которой доступен лишь администратору, получающий доступ посредством аутенфикации через страницу входа.

Второй контейнер обслуживает статический контент, который включает: главную страницу, которая перенаправляет пользователя в разные разделы веб-приложения, страницу, содержащую информацию о сервисе/веб-приложении и страницу входа.

Второй контейнер также реализует обратное проксирование, при котором происходит проверка поступающих ему запросов на имена файлов, если файл оканчивается на .php, то запрос обрабатывает сервер динамики, если же файл оканчивается на .html или .js, то запрос обрабатывает сервер статики.

Аутенфикация происходит средствами веб-сервера apache, а именно с помощью модуля libaprutil1-dbd-mysql, который позволяет серверу обращаться к базе данных и брать от туда имя пользователя и его праль, хешированный по алгоритму sha1. Аутенфикация защищает страницу администрирования, остальные файлы открыты всем.

Контейнеры также производят инъекцию конфигурационных файлов в оба сервера, которые содержат новые конфигурации портов и каталогов с файлами, для удовлетворения условия задачи.

Результаты выполнения работы представлены на рисунках 1-5 и листингах 1-14.

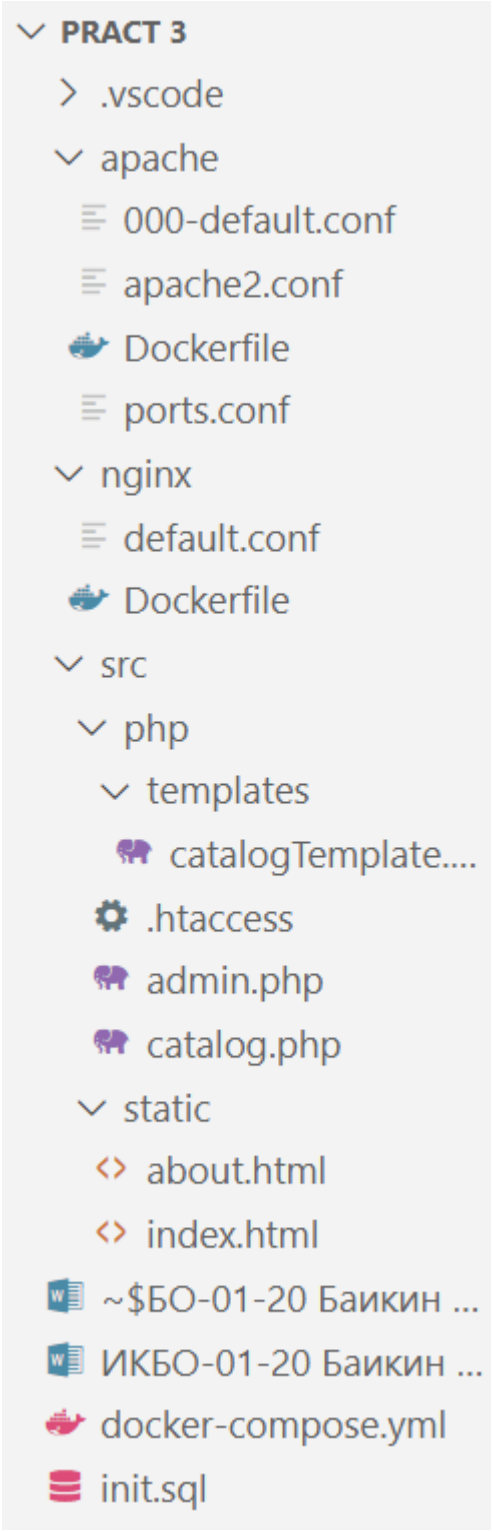

Рисунок 1 — Структура проекта

Листинг 1. Файл docker-compose.yml

```
version: '3'
services:
  apache:
    container_name: apache
    build: apache
    volumes:
      - ./src/php:/var/www/html-dynamic
    ports:
      - 8081:8081
  nginx:
    container_name: nginx
    build: nginx
    volumes:
      - ./src/static:/usr/share/nginx/html-static
    ports:
      - 8082:8082
  db:
    image: 'mysql:latest'
    ports:
      - 3306:3306
    volumes:
      - ./init.sql:/data/application/init.sql
    command: --init-file /data/application/init.sql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: root

  adminer:
    image: adminer
    restart: always
    ports:
      - 8080:8080
```

Листинг 2. Файл init.sql

```
CREATE DATABASE IF NOT EXISTS appDB;
CREATE USER IF NOT EXISTS 'user'@'%' IDENTIFIED BY 'password';
GRANT SELECT,UPDATE,INSERT,DELETE ON appDB.* TO 'user'@'%';
FLUSH PRIVILEGES;

USE appDB;
CREATE TABLE IF NOT EXISTS users (
  ID INT(11) NOT NULL AUTO_INCREMENT,
  name VARCHAR(20) NOT NULL,
  password VARCHAR(40) NOT NULL,
  PRIMARY KEY (ID)
);

INSERT INTO users (name, password)
SELECT * FROM (SELECT 'admin', '{SHA}0DPiKuNIrrVmD8IUCuw1hQxNqZc=') AS tmp
WHERE NOT EXISTS (
    SELECT name FROM users WHERE name = 'admin' AND password =
'{SHA}0DPiKuNIrrVmD8IUCuw1hQxNqZc='
) LIMIT 1;

CREATE TABLE IF NOT EXISTS catalog (
  ID INT(11) NOT NULL AUTO_INCREMENT,
  product_name TEXT NOT NULL,
  product_desc TEXT NOT NULL,
  product_price INT(11) NOT NULL,
  PRIMARY KEY (ID)
);

INSERT INTO catalog (`product_name`, `product_desc`, `product_price`)
SELECT * FROM (SELECT 'Веник', 'Связка прутьев или веток, используемая для
подметания помещений, но иногда и уличных территорий. Прежде употреблялся также
для чистки одежды, для опрыскивания водой белья или цветов.', 70) AS tmp
WHERE NOT EXISTS (
    SELECT product_name FROM catalog WHERE product_name = 'Веник'
) LIMIT 1;

INSERT INTO catalog (`product_name`, `product_desc`, `product_price`)
SELECT * FROM (SELECT 'Корзина', 'Корзина для мусора', 100) AS tmp
WHERE NOT EXISTS (
    SELECT product_name FROM catalog WHERE product_name = 'Корзина'
) LIMIT 1;

INSERT INTO catalog (`product_name`, `product_desc`, `product_price`)
SELECT * FROM (SELECT 'Дрель', 'Ручной, пневматический или электрический
инструмент, предназначенный для придачи вращательного движения сверлу или другому
режущему инструменту для сверления отверстий в различных материалах при
проведении строительных, отделочных, столярных, слесарных и других работ.', 5600)
AS tmp
WHERE NOT EXISTS (
```

```
    SELECT product_name FROM catalog WHERE product_name = 'Дрель'
) LIMIT 1;

INSERT INTO catalog (`product_name`, `product_desc`, `product_price`)
SELECT * FROM (SELECT 'Краска', 'Краска для стен', 500) AS tmp
WHERE NOT EXISTS (
    SELECT product_name FROM catalog WHERE product_name = 'Краска'
) LIMIT 1;

INSERT INTO catalog (`product_name`, `product_desc`, `product_price`)
SELECT * FROM (SELECT 'Шпатель', 'Шпатель для штукатурки', 100) AS tmp
WHERE NOT EXISTS (
    SELECT product_name FROM catalog WHERE product_name = 'Шпатель'
) LIMIT 1;

INSERT INTO catalog (`product_name`, `product_desc`, `product_price`)
SELECT * FROM (SELECT 'Шуруповерт', 'Шуруповерт для болтов', 3000) AS tmp
WHERE NOT EXISTS (
    SELECT product_name FROM catalog WHERE product_name = 'Шуруповерт'
) LIMIT 1;

INSERT INTO catalog (`product_name`, `product_desc`, `product_price`)
SELECT * FROM (SELECT 'Шпатлевка', 'Шпатлевка для стен', 200) AS tmp
WHERE NOT EXISTS (
    SELECT product_name FROM catalog WHERE product_name = 'Шпатлевка'
) LIMIT 1;

INSERT INTO catalog (`product_name`, `product_desc`, `product_price`)
SELECT * FROM (SELECT 'Шуруп', 'Шуруп для болтов', 10) AS tmp
WHERE NOT EXISTS (
    SELECT product_name FROM catalog WHERE product_name = 'Шуруп'
) LIMIT 1;

INSERT INTO catalog (`product_name`, `product_desc`, `product_price`)
SELECT * FROM (SELECT 'Болт', 'Болт для шуруповерта', 20) AS tmp
WHERE NOT EXISTS (
    SELECT product_name FROM catalog WHERE product_name = 'Болт'
) LIMIT 1;
```

Листинг 3. Файл nginx/default.conf

```
server {
    listen       8082;
    listen  [::]:8082;
    server_name  localhost;

    location / {
        root   /usr/share/nginx/html-static;
        index  index.html index.htm;

        location ~ .php$ {
            proxy_set_header Host apache:8081;
            proxy_pass http://apache:8081;
        }

        location ~ (.html)|(.js)$ {
            try_files $uri $uri/;
        }
    }

    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html-static;
    }
}
```

Листинг 4. Файл nginx/Dockerfile

```
FROM nginx
WORKDIR /usr/share/nginx/html-static
COPY default.conf /etc/nginx/conf.d/default.conf
CMD ["nginx", "-g", "daemon off;"]
EXPOSE 8082
```

Листинг 5. Файл apache/000-default.conf

```
<VirtualHost *:8081>
    ServerAdmin test@localhost
    DocumentRoot /var/www/html-dynamic

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Листинг 6. Файл apache/apache2.conf

```
DBDriver mysql
DBDParams host=db,port=3306,user=user,pass=password,dbname=appDB
DBDMin  2
DBDKeep 8
DBDMax  20
DBDExptime 300

DefaultRuntimeDir ${APACHE_RUN_DIR}

PidFile ${APACHE_PID_FILE}

Timeout 300

KeepAlive On

MaxKeepAliveRequests 100

KeepAliveTimeout 5

User ${APACHE_RUN_USER}
Group ${APACHE_RUN_GROUP}

HostnameLookups Off

ErrorLog ${APACHE_LOG_DIR}/error.log

LogLevel warn

IncludeOptional mods-enabled/*.load
IncludeOptional mods-enabled/*.conf

Include ports.conf

<Directory />
    Options FollowSymLinks
    AllowOverride None
    Require all denied
</Directory>

<Directory /usr/share>
    AllowOverride None
    Require all granted
</Directory>

<Directory /var/www/html>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
```

```
<Directory /var/www/html-dynamic>
    AuthBasicProvider dbd
    AuthDBDUserPWQuery "select password from users where name = %s"
</Directory>

AccessFileName .htaccess

<FilesMatch "^\.ht">
    Require all denied
</FilesMatch>

LogFormat "%v:%p %h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\""
vhost_combined
LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %O" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

IncludeOptional conf-enabled/*.conf
IncludeOptional sites-enabled/*.conf
```

Листинг 7. Файл apache/Dockerfile

```
FROM php:8.0-apache
WORKDIR /var/www/html-dynamic
RUN docker-php-ext-install mysqli
COPY ports.conf /etc/apache2/ports.conf
COPY 000-default.conf /etc/apache2/sites-available/000-default.conf
COPY apache2.conf /etc/apache2/apache2.conf
RUN a2enmod authn_dbd && apt-get update && apt-get install -y apache2-utils
libaprutil1-dbd-mysql
EXPOSE 8081
```

Листинг 8. Файл apache/ports.conf

```
Listen 8081
<IfModule ssl_module>
        Listen 443
</IfModule>
<IfModule mod_gnutls.c>
        Listen 443
</IfModule>
```

Листинг 9. Файл src/static/about.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Строительный магазин</title>
  </head>
  <body>
    <h1>Строительный магазин</h1>
    <p>
      Lorem ipsum dolor, sit amet consectetur adipisicing elit. Voluptatibus,
      consectetur? Ad, recusandae similique magnam eligendi aliquid placeat
      minima voluptatem illum aliquam incidunt facilis delectus, voluptate quia
      molestiae voluptates provident corrupti!
    </p>
  </body>
</html>
```

Листинг 10. Файл src/static/index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Строительный магазин</title>
  </head>
  <body>
    <h1>Строительный магазин</h1>
    <ul>
      <li><a href="catalog.php">Каталог</a></li>
      <li><a href="about.html">О магазине</a></li>
      <li><a href="admin.php">Вход для админа</a></li>
    </ul>
  </body>
</html>
```

Листинг 11. Файл src/php/.htaccess

```
<Files admin.php>
    AuthName "Login required"
    AuthType Basic
    Require valid-user
</Files>
<Files catalog.php>
    Require all granted
</Files>
```

Листинг 12. Файл src/php/admin.php

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>Administrate</title>
        <style>span { margin: 10px; }</style>
    </head>
    <body>
        <h1>List of users</h1>
        <?php
            $mysqli = new mysqli("db", "user", "password", "appDB");
                        $result = $mysqli->query("SELECT * FROM users");
                        while ($row = mysqli_fetch_assoc($result)) {
                                $users[] = $row;
                        }
        ?>
        <div style="
            display: flex;
            flex-direction: column;
        ">

                    <?php foreach ($users as $user) : ?>
                                <?php echo implode(" | ",$user) ?>
                       <?php endforeach; ?>

                </div>
        <?php $mysqli->close(); ?>
    </body>
</html>
```

Листинг 13. Файл src/php/catalog.php

```
<?php

$mysqli = new mysqli("db", "user", "password", "appDB");

// get data from db
$result = $mysqli->query("SELECT * FROM catalog");
while ($row = mysqli_fetch_assoc($result)) {
    $products[] = $row;
}

// загружаем шаблон
include('templates/catalogTemplate.php');
```

Листинг 14. Файл src/php/templates/catalogTemplate.php

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Каталог</title>
  <style>
    table{
      border-collapse: collapse;
      border: 1px solid black;
    }
    tr, td, th{
      padding: 7px 10px;
      border: 1px solid black;
    }

  </style>
 </head>
 <body>
  <h1>Каталог товаров</h1>
  <table>
     <tr>
      <th>Название</th>
      <th>Описание</th>
      <th>Цена</th>
    </tr>
       <?php foreach ($products as $product) : ?>
        <tr>
          <td>
            <?php echo $product['product_name'] ?>
          </td>
          <td>
            <?php echo $product['product_desc'] ?>
          </td>
          <td>
            <?php echo $product['product_price'] ?>
          </td>
        </tr>
      <?php endforeach; ?>

  </table>
 </body>
</html>
```

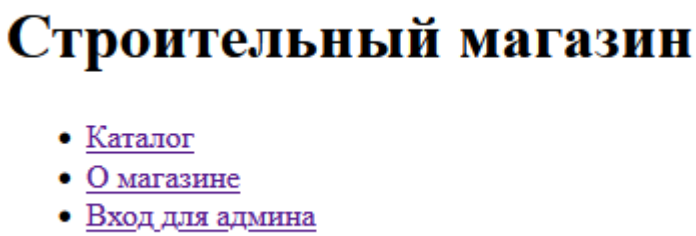

Рисунок 2 — Стартовая страница

Каталог товаров

Название	Описание	Цена
Веник	Связка прутьев или веток, используемая для подметания помещений, но иногда и уличных территорий. Прежде употреблялся также для чистки одежды, для опрыскивания водой белья или цветов.	70
Корзина	Корзина для мусора	100
Дрель	Ручной, пневматический или электрический инструмент, предназначенный для придачи вращательного движения сверлу или другому режущему инструменту для сверления отверстий в различных материалах при проведении строительных, отделочных, столярных, слесарных и других работ.	5600
Краска	Краска для стен	500
Шпатель	Шпатель для штукатурки	100
Шуруповерт	Шуруповерт для болтов	3000
Шпатлевка	Шпатлевка для стен	200
Шуруп	Шуруп для болтов	10
Болт	Болт для шуруповерта	20

Рисунок 3 — Каталог товаров

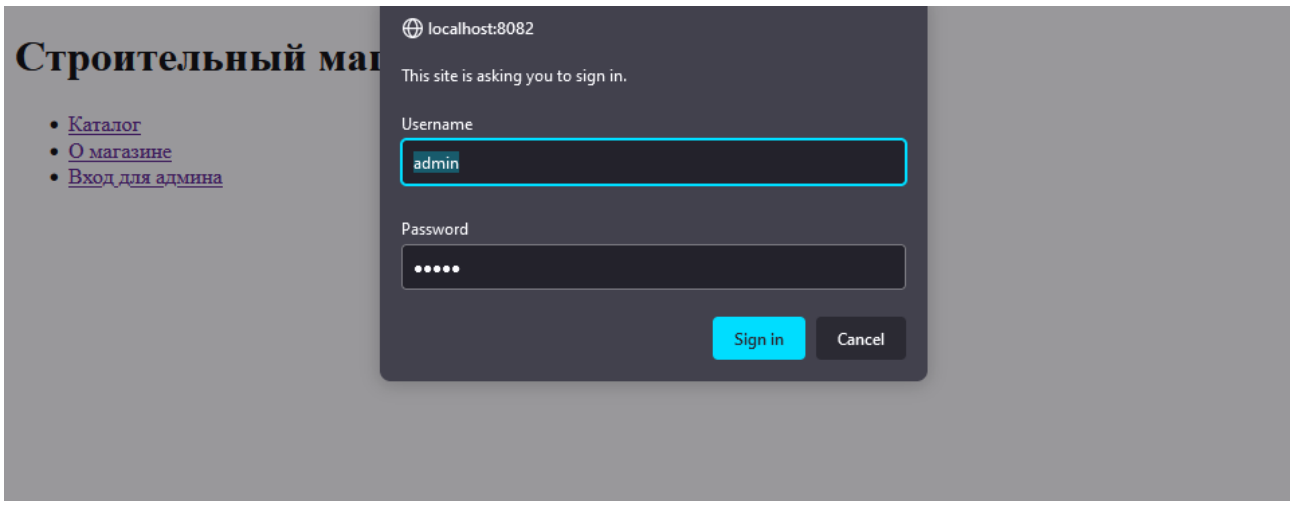

Рисунок 4 — Вход администратора

List of users

1 | admin | {SHA}0DPiKuNIrrVmD8IUCuw1hQxNqZc=

Рисунок 5 — Список пользователей

Вывод

В ходе выполнения данной работы были укреплены навыки проектирования, разработки, конфигурирования и развертки веб-серверного ПО, а также его контейнеризация средствами Docker. Была создана сложная конфигурация из 2х веб-серверов и одной базы данных. У веб-серверов были настроены нетипичные конфигурации, такие как коренной каталог и порт для подключения.

Ответы на вопросы

1. Веб-сервер — сервер, принимающий HTTP-запросы от клиентов, обычно веб-браузеров, и выдающий им HTTP-ответы, как правило, вместе с HTML- страницей, изображением, файлом, медиа-потоком или другими данными. Веб-сервером называют как программное обеспечение, выполняющее функции веб-сервера, так и непосредственно компьютер, на котором это программное обеспечение работает.
2. Сервер приложений — это программная платформа, предназначенная для эффективного исполнения процедур, на которых построены приложения. Сервер приложений действует как набор компонентов, доступных разработчику программного обеспечения через API, определённый самой платформой.
3. Изначально, основой Всемирной паутины были web-сервера CERN httpd, написанные Тимом Бернерсом-Ли на языке программирования Си. В 1995 году Брайан Белендорф объединил патчи веб-сервера CERN и создал первую версию сервера Apache (сокращение от «a patchy server»). В декабре 1995 года вышел релиз Apache 1.0. В 1999 году была создана некоммерческая организация Apache Software Foundation. В 1994 году датский программист Расмус Лердорф создал набор сценариев на Perl/CGI для вывода и учёта посетителей его онлайн-резюме, обрабатывающий шаблоны HTML-документов. Лердорф назвал его PHP (англ. Personal Home Page – «личная домашняя страница»). В 2000-м году вышла разработанная компанией Zend Technologies четвертая версия интерпретатора PHP, дополненная множеством новых функций. Разработка первого клиентского языка велась в 1992-1995 годах компанией Nombas. Этот язык, названный Cmm («Си- минус-минус») не получил широкого распространения, так как сценарии, написанные на нем работали только в 16-битовом Netscape Navigator под управлением Windows. В 1996 году компания Microsoft выпустила технологию ASP (англ. Active Server Pages — «активные серверные страницы»). В 2005 году была разработана технология AJAX (от англ. Asynchronous JavaScript and XML — «асинхронный JavaScript и XML»). В 1995 году Джон Гей и Роберт Татцуми из компании Future Wave разработали программу FutureSplash для создания анимации методом векторного морфинга (плавной трансформации изображения). В 2002 году выходит 6 версия Adobe Flash – Flash MX. С 2005 года, когда корпорация Adobe

купила компанию Macromedia, начался новый виток развития Flash. В 2007 году компания Microsoft выпустила браузерную мультимедийную платформу Silverlight, работающую с .NET Framework.

4. HTTP — протокол прикладного уровня передачи данных, изначально — в виде гипертекстовых документов в формате HTML, в настоящее время используется для передачи произвольных данных. Основным объектом манипуляции в HTTP является ресурс, на который указывает URI (Uniform Resource Identifier) в запросе клиента. Обычно такими ресурсами являются хранящиеся на сервере файлы, но ими могут быть логические объекты или что-то абстрактное. Особенностью протокола HTTP является возможность указать в запросе и ответе способ представления одного и того же ресурса по различным параметрам: формату, кодировке, языку и т. д. (в частности, для этого используется HTTP-заголовок). Именно благодаря возможности указания способа кодирования сообщения клиент и сервер могут обмениваться двоичными данными, хотя данный протокол является текстовым.
5. Клиент получает от сервера некую информацию от сервера, а также может сам отправлять информацию на сервер. Пользователь взаимодействует с клиентом посредствам графического интерфейса пользователя.
6. Веб-сервера служат для централизованного хранения и обработки информации, а также для синхронизации множества клиентов, подключающихся к ним.
7. SSL — криптографический протокол, который подразумевает более безопасную связь. Он использует асимметричную криптографию для аутентификации ключей обмена, симметричное шифрование для сохранения конфиденциальности, коды аутентификации сообщений для целостности сообщений.
8. Система управления содержимым — информационная система или компьютерная программа, используемая для обеспечения и организации совместного процесса создания, редактирования и управления содержимым, иначе — контентом.
9. Верно при наличии у сервера приложения соответствующего программного модуля для обработки запросов и отправки ответов по протоколу http, обычно он встроен в программную платформу.
10. Common Gateway Interface — «интерфейс общего шлюза») — стандарт интерфейса, используемого внешней программой для связи с веб-сервером. Программу, которая работает по такому интерфейсу совместно с веб- сервером, принято называть шлюзом, хотя многие предпочитают названия « скрипт » (сценарий) или «CGI-программа».
11. Все скрипты, как правило, помещают в каталог cgi (или cgi-bin) сервера, но это не обязательно: скрипт может располагаться где угодно, но при этом большинство веб-серверов требует специальной настройки. В веб-сервере Apache, например, такая настройка может производиться при

помощи общего файла настроек httpd.conf или с помощью файла .htaccess в том каталоге, где содержится этот скрипт.

12. CGI не налагает особых условий на платформу, поэтому работает на всех популярных платформах и WEB серверах. Также технология не привязана к конкретному языку программирования и может быть использована на любом языке, работающем со стандартными потоками ввода/вывода. Производительность программ CGI не очень высока. Основной причиной этого является то, что при каждом обращении к серверу создается отдельный процесс, требующий большого количества системных ресурсов. Кроме этого встроенных средств масштабируемости данная технология не предусматривает, и об этом разработчик должен заботиться отдельно. CGI программа представляет из себя готовый к исполнению файл, что препятствует легкому расширению системы. Данные причины привели к тому, что сейчас разработке приложений в технологии CGI предпочитают более развитые платформы, предоставляющие больше удобства разработчикам и обладающие повышенной производительностью. Тем не менее знание CGI необходимо для понимания работы высокоуровневых платформ.
13. Интерфейс FastCGI — клиент-серверный протокол взаимодействия веб-сервера и приложения, дальнейшее развитие технологии CGI. По сравнению с CGI является более производительным и безопасным.
14. Отличия от CGI : Недостаток CGI-программ в том, что они должны быть перезапущены веб-сервером при каждом запросе, что приводит к понижению производительности. FastCGI, вместо того чтобы создавать новые процессы для каждого нового запроса, использует постоянно запущенные процессы для обработки множества запросов. Это позволяет экономить время.
15. Менеджер процессов тесно взаимодействует с Микроядром, чтобы обеспечить услуги, составляющие сущность операционной системы. Хотя он и является единственным процессом, который использует то же адресное пространство, что и Микроядро, Менеджер процессов выполняется как истинный процесс. И он, как и все остальные процессы, подвергается диспетчеризации со стороны Ядра и использует предоставляемые Микроядром примитивы передачи сообщений для связи с другими процессами в системе. Менеджер процессов отвечает за создание новых процессов в системе и за управление основными ресурсами, связанными с процессом. Все эти услуги предоставляются посредством сообщений. Так, например, если процесс хочет породить новый процесс, он делает это, посылая сообщение с указанием атрибутов создаваемого процесса. Обратите внимание, что т.к. сообщения передаются по сети, вы можете легко создать процесс на другом узле сети, послав сообщение Менеджеру процессов на этом узле.
16. PHP-FPM — простыми словами программный пакет, позволяющий выполнить обработку скриптов, написанных на языке PHP. Включен в

состав PHP с версии 5.3.3, для более ранних версий необходима установка отдельно. Является альтернативой FastCGI — протоколу взаимодействия веб-сервера с программами. FPM расшифровывается как F astcgi P rocess M anager.

17. Spawn-fcgi используется для запуска удаленных и локальных FastCGI процессов. Почему лучше использовать spawn-fcgi Разделение привилегий без необходимости suid -исполняемого файла или запуска сервера с привилегиями root. Lighttpd — веб-сервер, разрабатываемый с расчётом на скорость и защищённость, а также соответствие стандартам. Это свободное программное обеспечение, распространяемое по лицензии BSD. lighttpd работает в Linux и других Unix-подобных операционных системах, а также в Microsoft Windows.
18. chroot — операция изменения корневого каталога в Unix-подобных операционных системах. Программа, запущенная с изменённым корневым каталогом, будет иметь доступ только к файлам, содержащимся в данном каталоге.
19. PHP интерпретатор запускается как независимый сервер, обрабатывающий входящие запросы на исполнение PHP скриптов по протоколу FastCGI, что позволяет ему работать с любым веб-сервером, поддерживающим этот протокол.
20. На практике выбирая между встроенным менеджером и внешним надо оценить ситуацию и выбрать именно тот инструмент который наиболее подходит запросам. Например создавая простой сервер для нескольких сайтов на типовых движках применение внешнего менеджера будет явно излишним. Данный подход хорош именно тем что можно как из конструктора собрать именно то что нужно для решения конкретной задачи.
21. Общий интерфейс шлюза это технология, которая позволяет веб-браузерам отправлять формы и подключаться к программам через веб-сервер. Лучший способ, которым веб-браузеры отправляют формы и просто подключаются к программам на сервере.
22. Simple Common Gateway Interface (SCGI) протокол по взаимодействию приложений с веб (http) серверами, разработанный как альтернатива Common Gateway Interface. Он похож на FastCGI, но проще в реализации.
23. PCGI (Perl Common Gateway Interface) — библиотека к языку программирования Perl для работы с интерфейсом CGI (Common Gateway Interface). Библиотека позволяет с высокой скоростью обрабатывать входящий поток данных.
24. PSGI или Perl Web Server Gateway Interface - это спецификация, предназначенная для отделения среды веб-сервера от кода веб-фреймворка. PSGI не является программным интерфейсом (API) для веб-приложений.

25. WSGI — стандарт взаимодействия между Python-программой, выполняющейся на стороне сервера, и самим веб-сервером, например Apache.
26. Веб-сервер apache превращает запросы браузера в конечные веб-страницы и знает, как обрабатывать программный код PHP. PHP – это всего лишь язык программирования, и без поддержки веб-сервера, например Apache, у пользователей Сети нет никакой возможности получить страницы, содержащие программный код PHP.
27. Это полностью бесплатное ПО, ничего не нужно платить даже в случае коммерческого использования продукта. Совместимость с различным ПО, написанным на языках Python, PHP, Perl и многих других. Отличная масштабируемость. Высокий уровень стабильности и отказоустойчивости. Здесь, правда, есть нюанс – в кривых руках даже Mercedes не заведется. Нужно с умом подключать модули к Apache и грамотно его конфигурировать, в противном случае можно легко получить нестабильную машину с кучей проблем. Apache может интерпретировать запросы как физический ресурс в файловой системе, требующий дополнительной обработки.
28. Apache считается не самым удачным вариантом для сайтов с очень высоким трафиком, в таких случаях лучше подойдет Nginx. Отдельные модули могут быть уязвимы с точки зрения безопасности, поэтому их нужно внимательно конфигурировать. Большинство встроенных функций и возможностей в веб- сервер Apache могут оказаться избыточными и ненужными для держателей сайтов.
29. Основной функциональной особенностью Apache является модульная система, которая позволяет отключать или подключать отдельные компоненты по своему усмотрению. Кроме того, модули мультипроцессинга Apache, отвечающие за обработку запросов клиентов, позволяют гибко настраивать политику обработки подключений. Вот ключевые MPM-модули Apache: mpm_prefork – формирует однопоточные процессы на входящие запросы; mpm_worker – генерирует процессы, которые обеспечивают управление несколькими потоками одновременно, по одному потоку на одно соединение; mpm_event – в чем-то похож на предыдущий модуль, но в данном случае реализована оптимизация под keep-alive соединения.
30. Ядро apache полностью написано на языке программирования C, его функциональные возможности ограничены обработкой конфигурационных файлов и исполнение протокола HTTP. Также ядро оснащено системой загрузки модулей, этот функционал никак не отключается и является фундаментальным. Различные модули имеют, как правило, узкую специализацию – например, это может быть кэширование входящих запросов или аутентификация. Для Apache существуют сотни динамических модулей, которые конфигурируются через ядро.

31. Система конфигурации apache осуществляется через текстовые конфиги, в которых хранятся параметры и настройки для работы системных компонентов и ПО. Например, в файле httpd.conf содержатся директивы, управляющие функционированием всего веб-сервера, а .htaccess хранит в себе данные о настройках Apache в рамках того каталога, где размещен файл, а также в его дочерних директориях.
32. URI — унифицированный идентификатор ресурса. По-русски иногда говорят [у́ри]. URI — последовательность символов, идентифицирующая абстрактный или физический ресурс. Ранее назывался Universal Resource Identifier — универсальный идентификатор ресурса. Унифицированный указатель ресурса — система унифицированных адресов электронных ресурсов, или единообразный определитель местонахождения ресурса. Используется как стандарт записи ссылок на объекты в Интернете.

Список информационных источников:

1. Документация по языку PHP, электронный ресурс URL: https://php.net (дата обращения: 11.09.2022) – Текст: электронный;
2. Документация по системе Docker, электронный ресурс URL: https://docs.docker.com (дата обращения: 11.09.2022) – Текст: электронный.
3. Документация по HTML, CSS, JavaScript, электронный ресурс URL: https://developer.mozilla.org (дата обращения 11.09.2022) – Текст: электронный.