Изображение выглядит как текст

Автоматически созданное описание

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**"МИРЭА – Российский технологический университет"**

**РТУ МИРЭА**

Институт Информационных технологий (ИТ)

Кафедра Инструментального и прикладного программного обеспечения

**ПРАКТИЧЕСКАЯ РАБОТА № 3**

**«Настройка сервера по заданным характеристикам, размещение проекта»**

по дисциплине «Разработка серверных частей интернет-ресурсов»

**Студент группы** ИКБО-24-20 Ефимцев С. М.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись студента)

**Руководитель практической работы** ассистент Благирев М.М.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Работа представлена «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Допущен к работе «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

**Содержание**

[**Цель работы**](#_5gn2qel16529) **3**

[**Теоретическое введение**](#_3w3yv4wpfc61) **3**

[**Ход работы**](#_jw5nkko4epz3) **4**

[**Вывод**](#_8fri2k5sotb6) **18**

[**Ответы на вопросы**](#_olqyyneydf0h) **19**

[**Ссылка на удалённый репозиторий**](#_nkboy28c5zd) **38**

[**Список использованной литературы**](#_fbwqarltqpzb) **39**

# **Цель работы**

Научиться создавать сложную серверную конфигурацию, настраивать сервер по заданным характеристикам.

# **Теоретическое введение**

Веб-сервер работает в качестве «посредника» между пользователем и физическим сервером. При получении запроса от посетителя он ищет необходимую страницу в каталоге с сайтом и отправляет её в ответ. Браузер принимает полученный файл, обрабатывает его и отображает на экране посетителя.

Передача информации веб-сервера выполняется по протоколу HTTP (HyperText Transfer Protocol), изначально созданного для работы с HTML-страницами. Уже позже стало возможным отправлять через HTTP файлы любых типов. В последнее время преобладают сайты, работающие через HTTPS. Это улучшенная версия HTTP, которая отличается от предшественника тем, что поддерживает шифрование трафика TLS/SSL между пользователем и сервером.

Звание самого популярного веб-сервера в мире уже более 25 лет удерживает за собой Apache HTTP Server, который принято называть сокращенно Apache или «Апач». Сегодня программа обслуживает более 40% всех существующих серверов, включая проекты IBM, eBay, PayPal и Facebook.

Apache – это свободное программное обеспечение для размещения веб-сервера. Он хорошо показывает себя в работе с масштабными проектами, поэтому заслуженно считается одним из самых популярных веб-серверов. Кроме того, Apache очень гибок в плане настройки, что даёт возможность реализовать все особенности размещаемого веб-ресурса.

Nginx — мощный инструмент для развертывания веб-сервера, который при правильной настройке превосходит Apache. Области применения Nginx весьма обширны — от кэширования HTTP до создания инвертированного прокси-сервера.

Сейчас Nginx обслуживает примерно 30,8% всех существующих сайтов мира, о чьих веб-серверах есть информация в открытом доступе. Понимание, что из себя представляет Nginx и как этот программный продукт можно применять на практике, помогает эффективно решать задачи во многих областях IT-индустрии.

MySQL — это реляционная система управления базами данных, популярная СУБД с моделью «клиент-сервер». Поэтому часто можно услышать от других разработчиков словосочетание «MySQL-сервер».

# **Ход работы**

Индивидуальный вариант 5 - Портфолио.

На рисунках 1 - 5 представлены созданные страницы веб-сервиса.

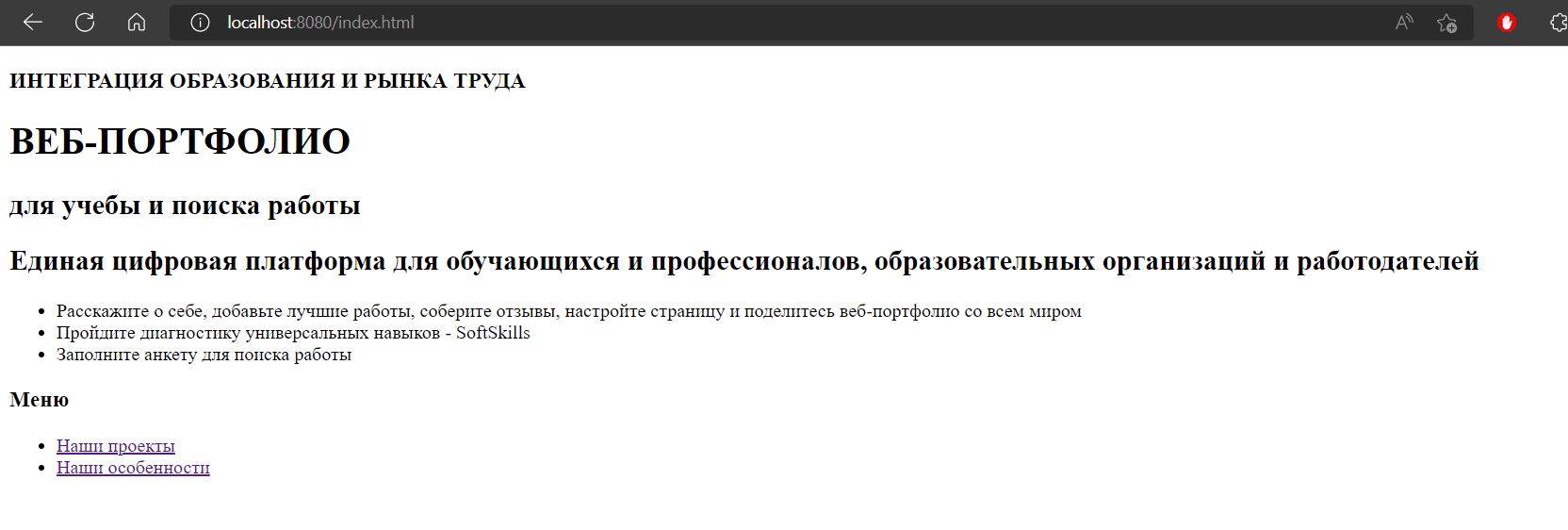


Рисунок 1 - Главная страница

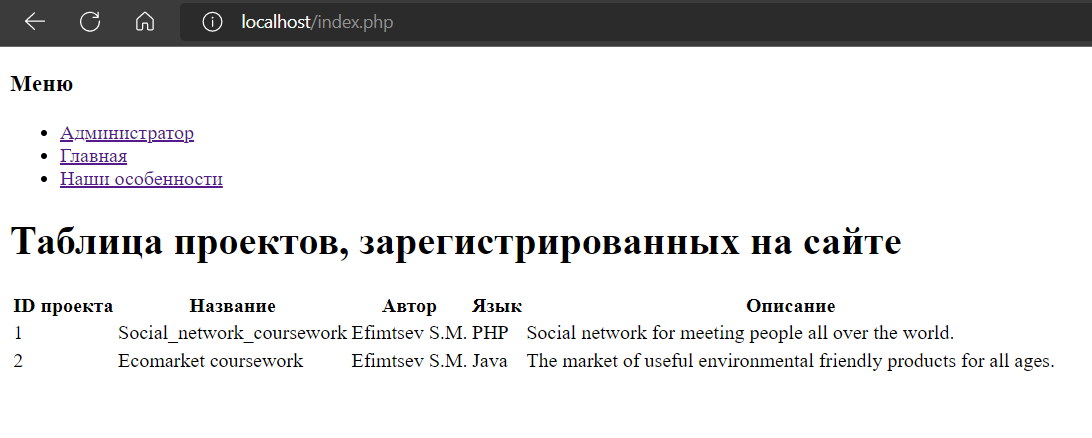


Рисунок 2 - Страница с проектами

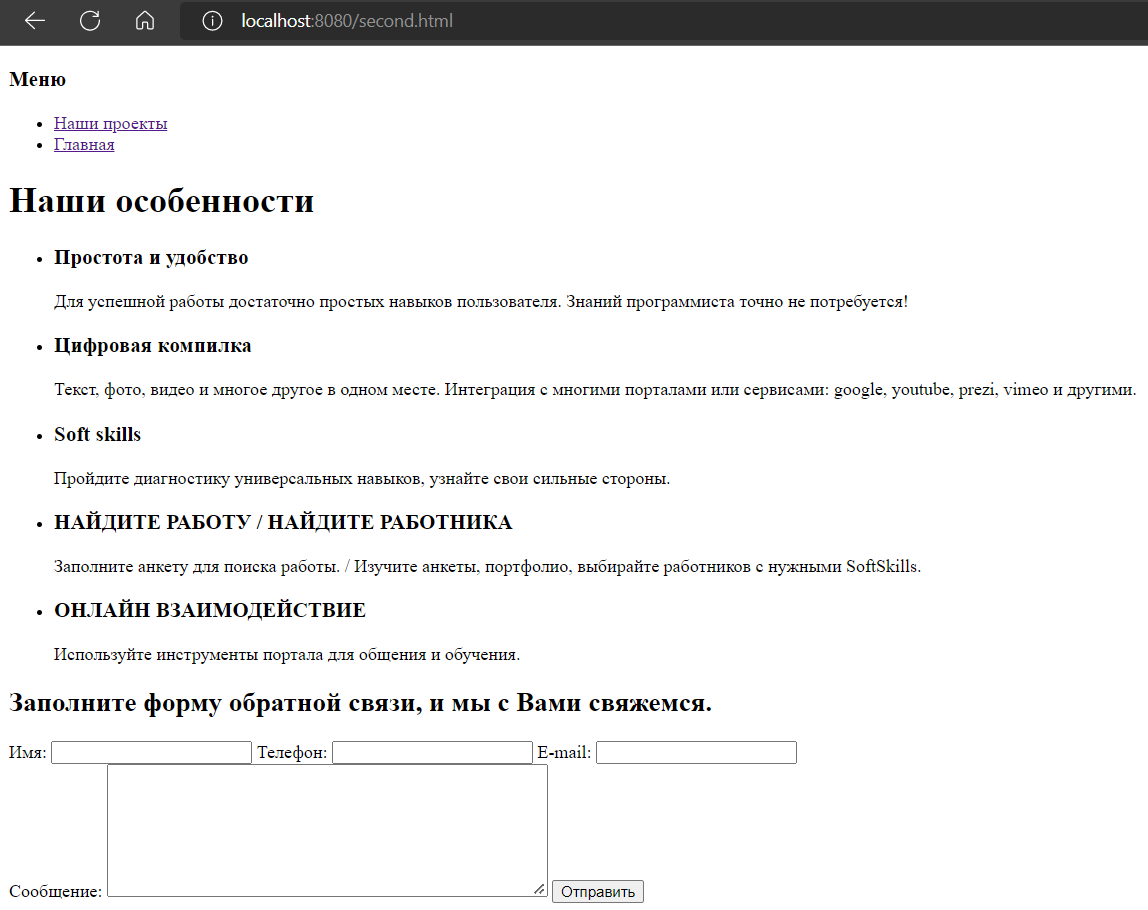


Рисунок 3 - Страница с формой обратной связи

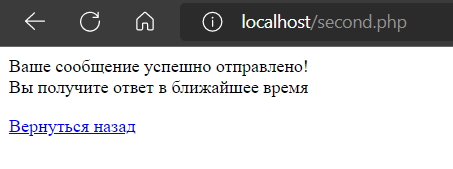


Рисунок 4 - Страница с обработкой формы обратной связи

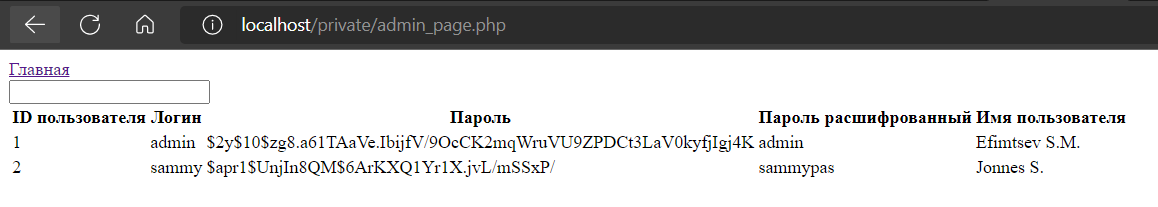


Рисунок 5 - Страница администратора

На рисунке 6 представлена страница авторизации, которую проходит пользователь при попытке зайти на страницу администратора.

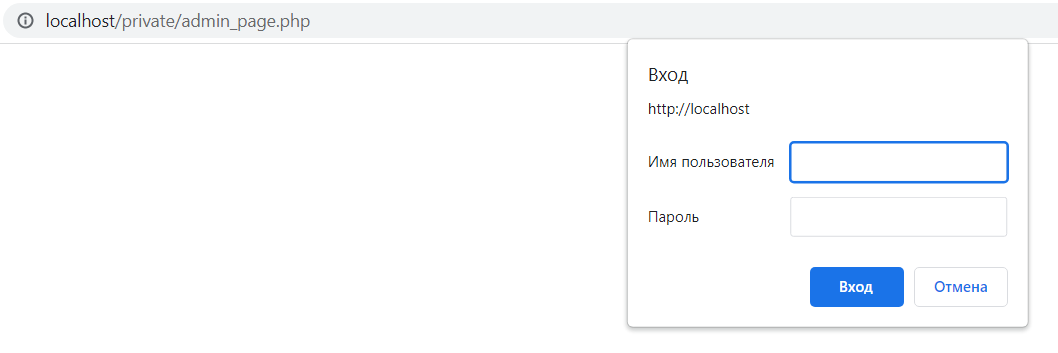


Рисунок 6 - Авторизация для входа на страницу администратора

На рисунке 7 показана структура проекта.

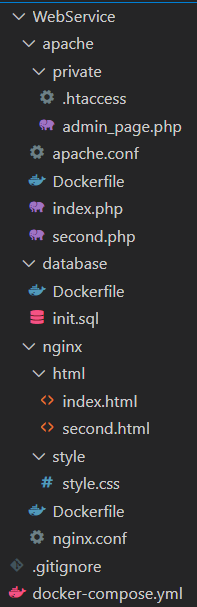


Рисунок 7 - Структура проекта

В листинге 1 показан файл, необходимый для пометки директории, как требующей авторизацию.

Листинг 1 - Файл .htaccess

AuthType Basic

AuthName "Restricted Access"

Require valid-user

Файл admin\_page.php содержит в себе терминал и таблицу пользователей с их данными. Он приведён в листинге 2.

Листинг 2 - Файл admin\_page.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">

<link rel="stylesheet" href="./style.css" />

<title>Информационно-административная страница о сервере</title>

</head>

<body>

<a href="http://localhost:80/index.php">Главная</a>

<br>

<form action="" method="post">

<input name="command" class="input" type="text">

</form>

<?php

$output = null;

$retval = null;

if (!empty($\_POST['command'])) {

$command = $\_POST['command'];

exec($command, $output, $retval);

foreach ($output as $i) {

echo $i, " ";

}

}

?>

<table>

<tr>

<th>ID пользователя</th>

<th>Логин</th>

<th>Пароль</th>

<th>Пароль расшифрованный</th>

<th>Имя пользователя</th>

</tr>

<?php

$mysqli = new mysqli("database1", "user", "password", "portfolioDB");

$result = $mysqli->query("SELECT \* FROM auth\_users");

foreach ($result as $row){

echo "<tr>

<td>{$row['id\_user']}</td>

<td>{$row['login\_user']}</td>

<td>{$row['password\_user']}</td>

<td>{$row['password\_user\_encr']}</td>

<td>{$row['name\_user']}</td>

</tr>";

}

?>

</table>

</body>

</html>

В листинге 3 представлен файл apache.conf, содержащий данные для подключения к базе данных пользователей и получении оттуда паролей для авторизации.

Листинг 3 - Файл apache.conf

<VirtualHost \*:80>

ServerName apache

# ServerAdmin user@localhost

DocumentRoot /var/www/html

ErrorLog ${APACHE\_LOG\_DIR}/error.log

CustomLog ${APACHE\_LOG\_DIR}/access.log combined

# <IfModule mod\_dbd.c>

DBDriver mysql

DBDParams "host=database1 port=3306 dbname=portfolioDB user=user pass=password"

# </IfModule>

<Directory "/var/www/html/private">

# AuthType Basic

# AuthName "Restricted Content"

# AuthUserFile /etc/apache2/.htpasswd

# Require valid-user

AuthBasicProvider dbd

AuthDBDUserPWQuery "SELECT password\_user FROM auth\_users WHERE login\_user = %s"

</Directory>

# AuthType Basic

# AuthName "Restricted Content"

# AuthUserFile /etc/apache2/.htpasswd

# Require valid-user

</VirtualHost>

В листинге 4 содержится файл, создающий образ с сервером Apache внутри.

Листинг 4 - Файл apache/Dockerfile

FROM php:7.4.3-apache

WORKDIR /var/www/html/

COPY . .

COPY ./apache.conf /etc/apache2/sites-available/000-default.conf

RUN apt-get update &&\

apt-get install -y libaprutil1-dbd-mysql &&\

a2enmod authn\_dbd &&\

a2enmod authn\_socache &&\

docker-php-ext-install mysqli &&\

service apache2 restart

В листинге 5 содержится файл с отображением таблицы проектов, зарегистрированных на сайте.

Листинг 5 - Файл index.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>Портфолио - Наши проекты</title>

</head>

<body>

<h3>Меню</h3>

<ul>

<li><a href="./private/admin\_page.php">Администратор</a></li>

<li><a href="http://localhost:8080/index.html">Главная</a></li>

<li><a href="http://localhost:8080/second.html">Наши особенности</a></li>

</ul>

<h1>Таблица проектов, зарегистрированных на сайте</h1>

<table>

<tr>

<th>ID проекта</th>

<th>Название</th>

<th>Автор</th>

<th>Язык</th>

<th>Описание</th>

</tr>

<?php

$mysqli = new mysqli("database1", "user", "password", "portfolioDB");

$result = $mysqli->query("SELECT \* FROM projects");

foreach ($result as $row){

echo "<tr><td>{$row['id\_project']}</td><td>{$row['title\_project']}</td><td>{$row['author\_name']}</td>

<td>{$row['main\_lang']}</td><td>{$row['descrip']}</td></tr>";

}

?>

</table>

<?php

// phpinfo();

?>

</body>

</html>

В листинге 6 показан скрипт обработки формы обратной связи.

Листинг 6 - Файл second.php

<!doctype html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />

<title>Ваше сообщение успешно отправлено</title>

</head>

<body>

<?php

$back = "<p><a href=\"javascript: history.back()\">Вернуться назад</a></p>";

if(!empty($\_POST['name']) and !empty($\_POST['phone']) and !empty($\_POST['mail'])

and !empty($\_POST['message'])){

$name = trim(strip\_tags($\_POST['name']));

$phone = trim(strip\_tags($\_POST['phone']));

$mail = trim(strip\_tags($\_POST['mail']));

$message = trim(strip\_tags($\_POST['message']));

mail('evendot@yandex.ru', 'Письмо с Портфолио',

'Вам написал: '.$name.'<br />Его номер: '.$phone.'<br />Его почта: '.$mail.'<br />

Его сообщение: '.$message,"Content-type:text/html;charset=windows-1251");

echo "Ваше сообщение успешно отправлено!<Br> Вы получите ответ в

ближайшее время<Br> $back";

exit;

}

else {

echo "Для отправки сообщения заполните все поля! $back";

exit;

}

?>

</body>

</html>

В листинге 7 представлен код написания Dockerfile для создания образа, содержащего СУБД MySQL.

Листинг 7 - Файл database/Dockerfile

FROM mysql

# ENV MYSQL\_DATABASE: portfolioDB

# ENV MYSQL\_USER: root

# ENV MYSQL\_PASSWORD: password

# ENV MYSQL\_ROOT\_USER: root

# ENV MYSQL\_ROOT\_PASSWORD: password

# ADD ./init.sql /docker-entrypoint-initdb.d/

EXPOSE 3306

В листинге 8 показан файл с запросами SQL для создания базы данных, таблиц внутри неё и наполнения этих таблиц.

Листинг 8 - Файл init.sql

CREATE DATABASE IF NOT EXISTS portfolioDB;

CREATE USER IF NOT EXISTS 'user'@'%' IDENTIFIED BY 'password';

GRANT ALL PRIVILEGES ON portfolioDB.\* TO 'user'@'%';

FLUSH PRIVILEGES;

USE portfolioDB;

CREATE TABLE IF NOT EXISTS projects (

id\_project int NOT NULL AUTO\_INCREMENT,

title\_project varchar(50) NOT NULL,

author\_name varchar(100) NOT NULL,

main\_lang varchar(20) NOT NULL,

descrip varchar(200),

PRIMARY KEY (id\_project)

);

INSERT INTO projects(title\_project, author\_name, main\_lang, descrip) VALUES(

"Social\_network\_coursework",

"Efimtsev S.M.",

"PHP",

"Social network for meeting people all over the world."

),

(

"Ecomarket coursework",

"Efimtsev S.M.",

"Java",

"The market of useful environmental friendly products for all ages."

);

CREATE TABLE IF NOT EXISTS auth\_users (

id\_user int NOT NULL AUTO\_INCREMENT,

login\_user varchar(30) NOT NULL,

password\_user varchar(100) NOT NULL,

password\_user\_encr varchar(100) NOT NULL,

name\_user varchar(100) NOT NULL,

PRIMARY KEY (id\_user)

);

INSERT INTO auth\_users (login\_user, password\_user, password\_user\_encr, name\_user) VALUES(

'admin',

'$2y$10$zg8.a61TAaVe.IbijfV/9OcCK2mqWruVU9ZPDCt3LaV0kyfjIgj4K',

'admin',

'Efimtsev S.M.'

),

(

'sammy',

'$apr1$UnjIn8QM$6ArKXQ1Yr1X.jvL/mSSxP/',

'sammypas',

'Jonnes S.'

);

В листинге 9 показан код статической HTML-страницы, являющейся главной в данном веб-сервисе.

Листинг 9 - Файл 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>Портфолио</title>

</head>

<body>

<header>

<h3>ИНТЕГРАЦИЯ ОБРАЗОВАНИЯ И РЫНКА ТРУДА</h3>

<h1>ВЕБ-ПОРТФОЛИО</h1>

<h2>для учебы и поиска работы</h2>

</header>

<main>

<h2>

Единая цифровая платформа для обучающихся и профессионалов,

образовательных организаций и работодателей

</h2>

<ul>

<li>

Расскажите о себе, добавьте лучшие работы, соберите отзывы, настройте

страницу и поделитесь веб-портфолио со всем миром

</li>

<li>Пройдите диагностику универсальных навыков - SoftSkills</li>

<li>Заполните анкету для поиска работы</li>

</ul>

<h3>Меню</h3>

<ul>

<li><a href="http://localhost:80/index.php">Наши проекты</a></li>

<li><a href="./second.html">Наши особенности</a></li>

</ul>

</main>

</body>

</html>

Листинг 10 содержит код файла статической HTML-страницы с формой обратной связи.

Листинг 10 - Файл second.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>Портфолио - Наши особенности</title>

</head>

<body>

<main>

<h3>Меню</h3>

<ul>

<li><a href="http://localhost:80/index.php">Наши проекты</a></li>

<li><a href="./index.html">Главная</a></li>

</ul>

<h1>Наши особенности</h1>

<ul>

<li>

<h3>Простота и удобство</h3>

<p>

Для успешной работы достаточно простых навыков пользователя. Знаний

программиста точно не потребуется!

</p>

</li>

<li>

<h3>Цифровая компилка</h3>

<p>

Текст, фото, видео и многое другое в одном месте. Интеграция с

многими порталами или сервисами: google, youtube, prezi, vimeo и

другими.

</p>

</li>

<li>

<h3>Soft skills</h3>

<p>

Пройдите диагностику универсальных навыков, узнайте свои сильные

стороны.

</p>

</li>

<li>

<h3>НАЙДИТЕ РАБОТУ / НАЙДИТЕ РАБОТНИКА</h3>

<p>

Заполните анкету для поиска работы. / Изучите анкеты, портфолио,

выбирайте работников с нужными SoftSkills.

</p>

</li>

<li>

<h3>ОНЛАЙН ВЗАИМОДЕЙСТВИЕ</h3>

<p>Используйте инструменты портала для общения и обучения.</p>

</li>

</ul>

<h2>Заполните форму обратной связи, и мы с Вами свяжемся.</h2>

<form method="post" action="http://localhost:80/second.php">

<div class="left">

<label for="name">Имя:</label>

<input maxlength="30" type="text" name="name" />

<label for="phone">Телефон:</label>

<input maxlength="30" type="text" name="phone" />

<label for="mail">E-mail:</label>

<input maxlength="30" type="text" name="mail" />

</div>

<div class="right">

<label for="message">Сообщение:</label>

<textarea rows="7" cols="50" name="message"></textarea>

<input type="submit" value="Отправить" />

</div>

</form>

</main>

</body>

</html>

В листинге 11 содержится файл для создания Docker-образа с сервером NginX.

Листинг 11 - Файл nginx/Dockerfile

FROM nginx

CMD ["nginx", "-g", "daemon off;"]

В листинге 12 находится файл конфигурации сервера NginX.

Листинг 12 - Файл nginx.conf

user www-data;

worker\_processes auto;

pid /run/nginx.pid;

include /etc/nginx/modules-enabled/\*.conf;

events {

worker\_connections 768;

}

http {

charset UTF-8;

sendfile on;

tcp\_nopush on;

tcp\_nodelay on;

keepalive\_timeout 65;

types\_hash\_max\_size 2048;

include /etc/nginx/mime.types;

default\_type application/octet-stream;

ssl\_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3;

ssl\_prefer\_server\_ciphers on;

access\_log /var/log/nginx/access.log;

error\_log /var/log/nginx/error.log;

gzip on;

include /etc/nginx/conf.d/\*.conf;

include /etc/nginx/sites-enabled/\*;

server{

listen 81;

listen [::]:81;

server\_name localhost;

charset UTF-8;

location / {

charset UTF-8;

root /usr/share/nginx/html-static;

index index.html index.htm;

}

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

charset UTF-8;

root /usr/share/nginx/html-static;

}

}

}

Листинг 13 представляет код файла docker-compose.yml, собирающий все Dockerfile’ы воедино и создающий готовый веб-сервис Портфолио.

Листинг 13 - Файл docker-compose.yml

version: '3.9'

services:

web-server-apache:

build:

context: ./WebService/apache

dockerfile: Dockerfile

# restart: always

volumes:

- './WebService/apache:/var/www/html/'

# - ./WebService/apache/apache2.conf:/etc/apache2/apache2.conf

ports:

- 80:80

# network\_mode: host

depends\_on:

- database1

- web-server-nginx

web-server-nginx:

container\_name: web-server-nginx

build:

context: ./WebService/nginx

dockerfile: Dockerfile

# restart: always

# links:

# - database

volumes:

- ./WebService/nginx/html:/usr/share/nginx/html

- ./WebService/nginx/nginx.conf:/etc/nginx/nginx.conf

ports:

- "8080:80"

depends\_on:

- database1

database1:

build:

context: ./WebService/database

dockerfile: Dockerfile

container\_name: database1

environment:

MYSQL\_ROOT\_PASSWORD: password

# MYSQL\_DATABASE: portfolioDB

volumes:

- ./WebService/database/:/docker-entrypoint-initdb.d

# ports:

# - 3307:33060

# restart: always

# **Вывод**

В процессе выполнения данной практической работы мною были получены навыки и умения построения и настройки сложной серверной конфигурации.

# **Ответы на вопросы**

**1. Что такое веб-сервер?**

Веб-сервер-это компьютерное программное обеспечение и базовое оборудование, которое принимает запросы от клиентов через HTTP (сетевой протокол, созданный для распространения веб-контента) или его безопасный вариант HTTPS и выдаёт им HTTP-ответы, как правило, вместе с HTML-страницей, изображением, файлом.

**2. Что такое сервер приложения и чем он отличается от веб-сервера?**

Сервер приложений - это программа сервера, работающая в распределенной сети и обеспечивающая среду выполнения для прикладных программ. Сервер приложений представляет собой основной компонент среды выполнения во всех конфигурациях - на нем выполняются приложения. Сервер приложений взаимодействует с веб-сервером и возвращает динамический, настраиваемый ответ на клиентский запрос.

Ключевое отличие: Сервер приложений - это программный механизм, который передает приложение с одного устройства на другое, в то время как веб-сервер - это компьютер, на котором выполняются программы, принимающие запросы HTTP от клиентов.

**3. Кратко опишите историю развития интернета в рамках развития веб серверов.**

Традиционно принято считать, что современный интернет начала развиваться в конце 80-х и начале 90-ых годов. В то время интернет-сайты были просто хранилищем документов, которые имели специальную разметку. В частности, использовался язык разметки SGML. SGML (Standard Generalized Markup Language) — это стандартный обобщенный язык разметки. Помимо размеченных документов сайты сохраняли некоторых связанных с документом данные, например файлы, изображения и т.д. Для создания ссылок между документами и ссылок между документом и связанными файлами, был предложен специальный язык гипертекстовой разметки − HTML (HyperText Markup Language). Для доступа к таким документам посредством сети − протокол HTTP (HyperText Transfer Protocol). Несмотря на развитие интернета и модификацию, многие принципы, заложенные в эти языки существенно, не изменились до настоящего времени. Только в 1999 году при смене протокола HTTP/1.1 на более совершенный HTTP/2 произошли некоторые кардинальные изменения, которые были необходимы для развития сети. Протокол HTTP реализован по клиент-серверной технологии и работает по принципу запрос-ответ без сохранения состояния. Целью запроса служит некий ресурс, который определяется единым идентификатором ресурса − URI (Uniform Resource Identifier), HTTP использует одну из разновидностей URI − URL (Uniform Resource Locator) − универсальный указатель ресурса, который помимо сведений о ресурсе определяет также его физическое местоположение.

**4. Кратко опишите протокол HTTP.**

HTTP — это протокол, позволяющий получать различные ресурсы, например HTML-документы. Протокол HTTP лежит в основе обмена данными в Интернете. HTTP является протоколом клиент-серверного взаимодействия, что означает инициирование запросов к серверу самим получателем, обычно веб-браузером (web-browser). Полученный итоговый документ будет (может) состоять из различных поддокументов, являющихся частью итогового документа: например, из отдельно полученного текста, описания структуры документа, изображений, видео-файлов, скриптов и многого другого.

**5. Опишите механизм взаимодействия HTTP-сервера, HTTP-клиента и пользователя.**

Клиент инициирует взаимодействие с сервером и посылает запрос, содержащий:

• метод доступа;

• адрес URI (Uniform Resource Identifier, универсальный идентификатор ресурса);

• версию протокола;

• сообщение с информацией о типе передаваемых данных, информацией о клиенте, пославшем запрос, и, возможно, с содержательной частью (телом) сообщения.

Ответ сервера содержит:

• строку состояния, в которую входит версия протокола и код возврата (успех или ошибка);

• сообщение, в которое входит информация сервера, метаинформация (т.е. информация о содержании сообщения) и его тело.

**6. Опишите цели и задачи веб-сервера.**

Цель веб-сервера проста - обслуживать одновременно большое количество клиентов, максимально эффективно используя hardware. Главная задача веб сервера принимать HTTP-запросы от пользователей, обрабатывать их, переводить в цифровой компьютерный код. Затем выдавать HTTP-ответы, преобразуя их из миллионов нолей и единичек в изображения, медиа-потоки, буквы, HTML страницы.

**7. Опишите технологию SSI.**

SSL (secure sockets layer — уровень защищённых cокетов) представляет собой криптографический протокол для безопасной связи.

Цель протокола — обеспечить защищенную передачу данных. При этом для аутентификации используются асимметричные алгоритмы шифрования (пара открытый — закрытый ключ), а для сохранения конфиденциальности — симметричные (секретный ключ). Когда пользователь заходит на веб-сайт, браузер запрашивает информацию о сертификате у сервера, который высылает копию SSL-сертификата с открытым ключом. Далее, браузер проверяет сертификат, название которого должно совпадать с именем веб-сайта.

Кроме того, проверяется дата действия сертификата и наличие корневого сертификата, выданного надежным центром сертификации. Если браузер доверяет сертификату, то он генерирует предварительный секрет (pre-master secret) сессии на основе открытого ключа, используя максимально высокий уровень шифрования, который поддерживают обе стороны.

Сервер расшифровывает предварительный секрет с помощью своего закрытого ключа, соглашается продолжить коммуникацию и создать общий секрет (master secret), используя определенный вид шифрования. Теперь обе стороны используют симметричный ключ, который действителен только для данной сессии. После ее завершения ключ уничтожается, а при следующем посещении сайта процесс рукопожатия запускается сначала.

**8. Что такое система управления контентом?**

CMS или «Система управления контентом» (еще ее иногда называют движком сайта) — это массивный кусок софта, который представляет собой полный набор инструментов для создания сайта и управления им же. То есть то, что обычно делается с помощью HTML, CSS и JavaScript-файлов, тут делается нажатием кнопок в графическом интерфейсе.

Естественно, такой подход значительно ускоряет скорость разворачивания полноценного ресурса и сильно облегчает жизнь тем, кто этим ресурсом управляет. По сути, программист никакой и не нужен. Практически любую функциональность на сайт можно добавить своими руками в интуитивно понятном интерфейсе.

**9. Верно ли, что сервер приложения умеет работать с протоколом HTTP?**

Для хранения данных обычно используются СУБД, что обусловлено необходимостью доступа к большому количеству взаимосвязанной информации. Однако сервер приложений не умеет работать с протоколом HTTP и обрабатывать пользовательские запросы, так как это задача веб-сервера. Чтобы обеспечить их взаимодействие был разработан общий интерфейс шлюза - CGI (Common Gateway Interface).

**10.Что такое CGI?**

CGI (от англ. Common Gateway Interface — «общий интерфейс шлюза») — стандарт интерфейса, используемого для связи внешней программы 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».

Поскольку гипертекст статичен по своей природе, веб-страница не может непосредственно взаимодействовать с пользователем. До появления JavaScript, не было иной возможности отреагировать на действия пользователя, кроме как передать введенные им данные на веб-сервер для дальнейшей обработки. В случае CGI эта обработка осуществляется с помощью внешних программ и скриптов, обращение к которым выполняется через стандартизованный (см. RFC 3875: CGI Version 1.1) интерфейс — общий шлюз.

**11.Как работает система с использованием интерфейс шлюза - CGI?**

Клиент получает от сервера и отдает серверу данные через стандартные stdin и stdout.

Получив запрос от браузера пользователя, веб-сервер определяет, что запрошено динамическое содержимое и формирует специальный запрос, который через интерфейс CGI передается веб-приложению. При его получении приложение запускается и выполняет запрос, результатом которого служит HTML-код динамически сформированной страницы, который передается назад веб-серверу, после чего приложение завершает свою работу.

**12.Назовите достоинства и недостатки CGI.**

CGI не налагает особых условий на платформу и web - сервер, 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 CGI - программ не высока. Основной причиной этого является то, что при очередном обращении к серверу для работы CGI – программы создается отдельный процесс, что требует большого количества системных ресурсов.

· Встроенных средств масштабируемости технология не предусматривает.

· CGI - программа представляет из себя готовый к исполнению файл, что препятствует легкому расширению системы.

**13.Что такое FastCGI?**

Интерфейс FastCGI — клиент-серверный протокол взаимодействия веб-сервера и приложения, дальнейшее развитие технологии CGI.

**14.Назовите основные отличия CGI от FastCGI.**

По сравнению с CGI является более производительным и безопасным. FastCGI снимает множество ограничений CGI-программ. Недостаток CGI-программ в том, что они должны быть перезапущены веб-сервером при каждом запросе, что приводит к понижению производительности. FastCGI. Использует постоянно запущенные процессы для обработки множества запросов. Это позволяет экономить время. FastCGI-программы могут быть запущены не только на конктетном сервере, но и где угодно в сети. Также возможна обработка запросов несколькими FastCGI-процессами, работающими параллельно.

**15.Что такое менеджер процессов?**

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.Что такое PHP-FPM?**

PHP-FPM - это разновидность SAPI для PHP.

FPM

SAPI, он же Server API. В php есть несколько таких API для разных вариантов его работы:

* CLI SAPI - в качестве консольной команды `php` для запуска наших кронов и других cli-программ (Command Line Interface)
* apxs2 SAPI - в качестве модуля к apache2
* CGI SAPI - в качестве запускаемого на каждом запросе CGI (сейчас так почти никто не делает)
* FPM SAPI - Fast Process Manager, написанный для PHP разработчиками из комании Badoo и теперь поддерживаемый сообществом

Работа с FPM отличается от работы с Apache в первую очередь тем, что FPM - это только PHP. Это не веб-сервер, не что-то умное. Это наоборот - максимально простой, легкий и быстрый менеджер процессов для PHP. В отличие от апача, он даже не использует http-протокол, а работает со специальным fastcgi-протоколом. В первую очередь FPM быстрее обрабатывает запросы благодаря его легковесности и простоте.

Во вторую очередь, FPM действительно умный менеджер процессов. Он контролирует количество работающих PHP-процессов, частоту их перезапуска для борьбы с утечками памяти (да, модули php как и всегда текут) и прочие простые вещи, необходимые для контроля сервера.

**17.Что такое Spawn-fcgi?**

Создание процесса FastCGI прямо в web-сервере имеет следующие недостатки

* процесс FastCGI может быть запущен только локально
* имеет тебе права что и web-сервер
* имеет туже base-dir что и web-сервер

Как только вы начнёте использовать отдельный FastCGI сервер чтобы снять нагрузку с web-сервера, вы сможете контролировать процесс FastCGI внешними программами, такими как spawn-fcgi.

spawn-fcgi используется чтобы запустить FastCGI процесс в своём окружении, выставить ему user-id, group-id и сменить корневую директорию (chroot).

Для большего удобства должен быть использван wrapper скрипт берущий на себя заботу обо всех опциях. Такой скрипт включён в состав lighttpd,- spawn-php.sh.

**18.Что такое Lighttpd?**

lighttpd (также «lighty», «лайти») — 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.

lighttp предостовляет интерфейс ко внешним программам которые поддерживают FastCGI интерфейс. FastCGI Интерфейс определён http://www.fastcgi.com/ и это платформо-независимый и серверо-независимый интерфейс между web-приложением и web-сервером.

Это подразумевает что FastCGI программы запущенные с Apache web-сервером также запустятся с lighttpd и наоборот.

lighttpd включает в себя внутренный FastCGI распределитель нагрузки который может использоваться для распределения сразу на несколько FastCGI серверов. В отличие от иных решений только FastCGI процесс должен находиться в кластере, а не целый web-сервер. Это позволяет использовать FastCGI процессу больше резурсов чем, например, load-balancer+apache+mod\_php.

**19.Что такое chroot окружение?**

Чаще всего администраторы сталкиваются с утилитой chroot, которая позволяет подменить местонахождение корня для дочерних программ.

Администратор выполняет команду chroot /mnt, где /mnt — это точка монтирования накопителя с ОС, и происходит чудо: отныне все программы, включая интерпретатор, действуют от лица «неисправной» системы. Теперь grub2-install отработает корректно.

Утилита chroot в современных дистрибутивах Linux является частью проекта GNU Coreutils, а исходный код умещается в один файл. Утилита выполняет следующие действия:

* разбирает аргументы командной строки;
* выполняет системный вызов chroot;
* выполняет переход в другой каталог;
* изменяет пользователя, от имени которого необходимо выполнить действия;
* выполняет заданную команду, а если команды нет, то запускает интерпретатор командной строки по умолчанию.

Утилита основана на одноименном системном вызове, который делает единственную вещь: меняет один из компонентов разрешения имен файлов. Назначение других системных вызовов, связанных с файловыми системами, очевидно: программы хотят уметь работать с файлами и каталогами.

**20.Опишите механизм взаимодействия серверов с использованием FastCGI.**

Связь же между web-сервером и FastCGI-процессом осуществляется через один сокет, который процесс должен слушать на предмет входящих подключений от web-сервера.

После приема соединения от web-сервера FastCGI-процесс обменивается данными с использованием простого протокола, решающего две задачи: организация двунаправленного обмена в рамках одного соединения (для эмуляции STDIN, STDOUT, STDERR) и организация нескольких независимых FastCGI-сессий в рамках одного соединения.

FastCGI приложение выполняет одну из трех определенных протоколом ролей. Наиболее привычная (и основная) — Responder, в которой приложение получает данные HTTP-запроса и формирует HTTP-ответ (аналогично CGI 1.1). Вторая роль — Authorizer: приложение получает данные HTTP-запроса и генерирует решение авторизован / не авторизован. Третья — Filter, в которой приложение также получает данные HTTP-запроса и дополнительно поток данных, хранящихся на сервере, после чего выполняет фильтрацию (преобразование) данных и формирует HTTP-ответ. Может использоваться для изменения «на лету» статического контента (HTML-документов, изображений), а также для их кэширования.

**21.Опишите процесс выбора встроенного или внешнего менеджера процессов.**

На практике, выбирая между встроенным менеджером и внешним, надо оценить ситуацию и выбирать 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.

**22.Что такое интерфейс 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» или «CGI-программа». По сути позволяет использовать консоль ввода и вывода для взаимодействия с клиентом.

**23.Что такое SCGI?**

SCGI-это протокол, который определяет связь между веб - сервером и сервером приложений. Это в отличие от CGI, который представляет собой более ранний интерфейс приложения (шлюза), предназначенный для того, чтобы позволить прикладному программисту избежать сложности сокетов и длительных сервисных процессов, когда плохая масштабируемость и высокие накладные расходы приемлемы.

Клиент подключается к SCGI-серверу по протоколу передачи потоков, допускающем передачу 8-битных байтов. Клиент начинает с отправки запроса. Когда SCGI-сервер видит конец запроса, он посылает обратно ответ и закрывает соединение. Формат ответа не специфицирован протоколом.

**24.Что такое PCGI?**

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. Основное достоинство заключается в том, что библиотека позволяет совершенно безопасно принимать сколь угодно крупные объёмы данных, при этом очень экономично потребляя оперативную память.

Кроме того, решается один из серьёзных вопросов безопасности: злонамеренно сформированный сложный запрос такого размера не перегрузит сервер. В качестве сравнения можно рассмотреть стандартные методы защиты:

* наложение ограничения на объём принимаемых данных;
* наложение ограничения на объём используемой на эти цели оперативной памяти;
* наложение ограничений на ресурсы сервера по времени разбора или по степени загрузки системы.

**25.Что такое PSGI?**

PSGI или Perl Web Server Gateway Interface - спецификация, предназначенная для отделения среды веб-сервера от кода веб-фреймворка. PSGI не является программным интерфейсом (API) для веб-приложений.

PSGI сервер - программа на Perl, предоставляющая среду для запуска в ней PSGI приложения. Его часто называют PSGI Application Container, так как он похож на Java Servlet Container, который представляет собой Java-процесс, предоставляющий среду для сервлетов Java.

PSGI, как говорит его разработчик Tatsuhiko Miyagawa, это «Перловый суперклей для веб-фреймворков и веб-серверов». Ближайшие родственники — WSGI (Python) и Rack (Ruby). Идея тут вот в чем. Разработчик очень часто тратит довольно много времени, чтобы адаптировать свое приложение под как можно большее количество движков, а PSGI предоставляет единый интерфейс для работы с различными серверами, что сильно упрощает жизнь.

Особенности:

* для обмена информацией между клиентом и сервером используется $env (представляет из себя ссылку на хеш);
* PSGI приложение — ссылка на Perl-функцию, которая принимает в качестве параметра $env;
* функция возвращает ссылку на массив, который состоит из 3 элементов: HTTP статус, [HTTP заголовки], [Тело ответа];
* функция может вернуть и ссылку на другую функцию, но это будет рассмотрено в других более углубленных статьях;
* расширение файла, содержащего код запуска приложения, должно быть .psgi.

**26.Что такое WSGI?**

WSGI (англ. Web Server Gateway Interface) — стандарт взаимодействия между Python-программой, выполняющейся на стороне сервера, и самим веб-сервером, например Apache.

WSGI предоставляет простой и универсальный интерфейс между большинством веб-серверов и веб-приложениями или фреймворками.

По стандарту, WSGI-приложение должно удовлетворять следующим требованиям:

* должно быть вызываемым (callable) объектом (обычно это функция или метод)
* принимать два параметра:
  + словарь переменных окружения (environ)
  + обработчик запроса (start\_response)
* вызывать обработчик запроса с кодом HTTP-ответа и HTTP-заголовками
* возвращать итерируемый объект с телом ответа

**27.Опишите механизм взаимодействия серверов Apache и PHP.**

Веб-сервер выполняет только один однопоточный процесс, поэтому приложения PHP будут останавливаться, если запрос заблокирован.

URI запросы обслуживаются из текущей директории, в которой был запущен PHP, если не используется опция -t для явного указания корневого документа. Если URI запроса не указывает на определённый файл, то будет возвращён index.php или index.html в указанной директории. Если ни один из файлов не существует, то поиск этих файлов будет продолжен в родительской директории и так далее до тех пор, пока они не будут найдены или был достигнут корень документа. Если найден index.php или index.html, он возвращается, а в $\_SERVER['PATH\_INFO'] будет находится последняя часть URL. В противном случае возвращается 404 код ответа.

Если PHP-файл указывается в командной строке, когда запускается веб-сервер, то он рассматривается как скрипт "маршрутизации" (router). Скрипт выполняется в самом начале каждого HTTP-запроса. Если этот скрипт возвращает false, то запрашиваемый ресурс возвращается как есть. В противном случае браузеру будет возвращён вывод этого скрипта.

**28.Опишите преимущества веб-сервера Apache.**

* Доступность. Это программное обеспечение с открытым исходным кодом. Значит, его может бесплатно использовать или модифицировать любой желающий. Разработчики по всему миру создают конфигурации и модули веб-сервера для своих специфических нужд. По этой же причине Apache регулярно получает полезные дополнения, расширяющие его базовый функционал.
* Гибкость настройки. Apache использует несколько конфигурационных файлов для управления веб-сервером. Это позволяет настроить ПО под узконаправленные задачи.
* Функциональность. У Apache динамическая модульная структура. Можно быстро подключать дополнительный функционал в виде скачиваемых модулей, даже без обращения к внешним источникам. Это позволяет решать целый комплекс важнейших задач в области безопасности, кэширования, редактирования URL, распределения нагрузки. Благодаря гибридным модулям MPM, Apache может одинаково успешно обслуживать статический и динамический контент. Есть возможность оперативно отключать ненужные модули и ускорять работу веб-сервера
* Кроссплатформенность. Apache работает как на Windows, так и на всех Unix-подобных системах. Администрирование веб-сервером не имеет серьёзных отличий на разных ОС. Индивидуален только процесс установки и расположение директорий с файлами программы.
* Совместимость. Apache работает на базе скриптовых или веб-ориентированных языков (PHP, Python, Tcl, Ruby, Perl, ASP), что делает его совместимым с самым широким спектром баз данных и серверного ПО. Многие веб-приложения и инструменты сразу выходят со средствами запуска из-под Apache в виде PHP-модуля. Веб-сервер, поддерживает технологии FastCGI и CGI, позволяющие пользоваться программными продуктами на объектно-ориентированных языках Java, sh, C, C++.
* Масштабируемость. Подходит для веб-ресурсов любого масштаба. Apache хорошо работает как на одностраничном сайте (лендинге), так и на многостраничном сайте с ежедневной аудиторией в десятки тысяч посетителей.
* Поддержка пользователей. Apache удерживает первенство популярности среди веб-серверов с 1996 года. За прошедшее время для него создана обширнейшая база документации – как официальной, так и созданной сторонними разработчиками. Готовые, подробно описанные руководства можно найти практически на любой сценарий.

**29.Опишите недостатки веб-сервера Apache.**

* Производительность. Скорость обработки запросов Apache несколько ниже, по сравнению со своими конкурентами. Гибкость веб-сервера в некоторых случаях вредит производительности. Например, Apache приходится каждый раз считывать несколько конфигурационных файлов на сервере, затрачивая системные ресурсы и время. Но этот и многие другие факторы можно исправить, отключив ненужные опции. Правда в таком случае функциональность Apache не будет сильно отличаться от других веб-серверов.
* Сложная конфигурация повышает уязвимость. Возможность подключать модули в Apache это не всегда преимущество. Чем больше модулей, тем сложнее становятся настройки. Соответственно, больше шансов допустить критические пробелы в контуре безопасности.
* Синтаксис конфигов.. В файлах с параметрами программы используются разнообразные переменные, поэтому настройка и управление веб-сервером может показаться сложной новичкам. Упростить администрирование Apache можно с помощью бесплатного инструмента Apache GUI.
* Излишний функционал. Даже без дополнительных модулей Apache предоставляет пользователям массу возможностей. Правда, большинство использует лишь небольшую часть базового функционала приложения. Поэтому часто после установки приходится тратить время на отключение «лишних» модулей.

**30.Опишите архитектуру веб-сервера Apache.**

Apache состоит из ядра и динамической модульной системы. Параметры системы изменяются с помощью конфигурационных файлов.

Ядро

Ядро Apache разработано Apache Software Foundation на языке C.

Модульная система

Модуль – отдельный файл, подключение которого расширяет изначальный функционал ядра. Они могут включаться в состав ПО при первоначальной установке или подгружаться позже через изменение конфигурационного файла.

Большинство из них отвечает за определенный аспект обработки клиентского запроса – поддержку различных языков программирования, безопасность, кэширование, аутентификацию и т.д. Таким образом, большая задача разбивается на несколько фаз, каждую из которых решает отдельный, узкоспециализированный модуль.

Для Apache существует больше 500 модулей. Многие популярные веб-приложения сразу выпускаются в виде модуля к Apache. Например, ISPmanager и VDSmanager.

Конфигурация

Система конфигурации Apache работает на текстовых файлах с прописанными настройками. Она подразделяется на три условных уровня, для каждого из которых имеется свой конфигурационный файл:

1. Уровень конфигурации сервера (файл httpd.conf) – основной конфигурационный файл. Действие распространяется на весь механизм веб-сервера.
2. Уровень каталога (файл .htaccess) – дополнительный конфигурационный файл. Его директивы охватывают только каталог, где расположен файл, а также вложенные подкаталоги.
3. Уровень виртуального хоста (файл httpd.conf> или extra/httpd-vhosts.conf).

Обычно конфигурационные файлы Apache находятся в папке «conf», а дополнительные конфигурационные файлы во вложенной в нее папке «extra». Внести изменения можно как через редактирование самого файла, так и через командную строку.

**31.Опишите функции ядра веб-сервера Apache.**

Основные функции — обработка конфигурационных файлов, протокол HTTP/HTTPS и загрузка модулей. Ядро может работать без модулей, но будет иметь ограниченный функционал.

**32.Опишите конфигурацию веб-сервера Apache.**

Система конфигурации Apache работает на текстовых файлах с прописанными настройками. Она подразделяется на три условных уровня, для каждого из которых имеется свой конфигурационный файл:

Уровень конфигурации сервера (файл httpd.conf) – основной конфигурационный файл. Действие распространяется на весь механизм веб-сервера.

Уровень каталога (файл .htaccess) – дополнительный конфигурационный файл. Его директивы охватывают только каталог, где расположен файл, а также вложенные подкаталоги.

Уровень виртуального хоста (файл httpd.conf> или extra/httpd-vhosts.conf).

Обычно конфигурационные файлы Apache находятся в папке «conf», а дополнительные конфигурационные файлы во вложенной в нее папке «extra». Внести изменения можно как через редактирование самого файла, так и через командную строку.

**33.Что такое URI, URL и чем они различаются.**

* URI - Uniform Resource Identifier (унифицированный идентификатор ресурса)
* URL - Uniform Resource Locator (унифицированный определитель местонахождения ресурса)
* URI – имя и адрес ресурса в сети, включает в себя URL и URN
* URL – адрес ресурса в сети, определяет местонахождение и способ обращения к нему
* URI https://wiki.merionet.ru/images/vse-chto-vam-nuzhno-znat-pro-devops/1.png
* URL - https://wiki.merionet.ru

# **Ссылка на удалённый репозиторий**

https://github.com/el-hombre0/Dev\_of\_serv\_parts\_of\_Internet\_resources

# 

# **Список использованной литературы**

1. Docker - Документация к Docker и Docker-compose [Электронный ресурс]. Режим доступа: https://docs.docker.com/compose/
2. PHP - Документация к PHP [Электронный ресурс]. Режим доступа: https://www.php.net/docs.php
3. Metanit - Сайт о программировании [Электронный ресурс]. Режим доступа: https://metanit.com/php/tutorial