|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра Инструментального и прикладного программного обеспечения

**ПРАКТИЧЕСКАЯ РАБОТА №3**

по дисциплине «Разработка серверных частей интернет-ресурсов»

**Тема практической работы:** Настройка сервера по заданным характеристикам, размещение проекта

**Студент группы** ИКБО-16-20 Пак Сергей Андреевич

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

(подпись студента)

**Руководитель практической работы** преподаватель Волков М.Ю.

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

(подпись руководителя)

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

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

Москва 2022

**СОДЕРЖАНИЕ**

[1. ЦЕЛЬ РАБОТЫ 3](#_Toc116480019)

[2. ХОД РАБОТЫ 4](#_Toc116480020)

[3. ВЫВОДЫ 15](#_Toc116480021)

[4. ОТВЕТЫ НА ВОПРОСЫ 16](#_Toc116480022)

[5. ССЫЛКА НА УДАЛЁННЫЙ РЕПОЗИТОРИЙ 22](#_Toc116480023)

[6. СПИСОК ЛИТЕРАТУРЫ 23](#_Toc116480024)

# 1. ЦЕЛЬ РАБОТЫ

Предлагается создать сложную серверную конфигурацию, состоящую из связки apache+nginx+php+База данных. Возможно использование 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 №1. 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 rootдиректории с базовых на другие. Для доступа к администрированию предусмотреть базовую авторизацию и аутентификацию с применением httaccess и хранением пользователей в БД (без использования htpasswd).

# 2. ХОД РАБОТЫ

Для начала необходимо создать структуру проекта (рисунок 1).

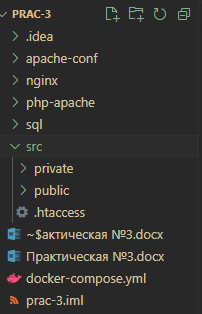


Рисунок 1 – Файловая структура проекта

В директории «apache-conf» будет находиться конфигурация для веб-сервера «Apache».

В директории «nginx» будет находиться конфигурация для веб-сервера «Nginx».

В директории «php-apache» будет находиться «Dockerfile» для сборки образа.

В директории «sql» будет находиться скрипт для инициализации базы данных.

В директории «src» будут находиться все страницы.

Теперь нужно начать с настройки веб-сервера «Apache». Его конфигурация находится в файле «apache-conf/httpd.conf», содержимое которого представлено на листинге 1.

Листинг 1 – Конфигурация веб-сервера «Apache»

ServerRoot "/etc/apache2"

ServerName 127.0.0.1

DocumentRoot "/var/www/apache-server/html"

ErrorLog /var/log/apache2/error.log

IncludeOptional mods-available/auth\_form.load

IncludeOptional mods-available/request.load

IncludeOptional mods-available/session.load

IncludeOptional mods-available/session\_cookie.load

Timeout 300

KeepAlive On

# Alias "/" "/var/www/apache-server/html/private/index.html"

Продолжение листинга 1

User ${APACHE\_RUN\_USER}

Group ${APACHE\_RUN\_GROUP}

ServerAdmin s.park190802@gmail.com

<IfModule mpm\_prefork\_module>

StartServers 5

MinSpareServers 5

MaxSpareServers 10

MaxClients 150

MaxRequestsPerChild 0

</IfModule>

<IfModule mpm\_worker\_module>

StartServers 2

MaxClients 150

MinSpareThreads 25

MaxSpareThreads 75

ThreadsPerChild 25

MaxRequestsPerChild 0

</IfModule>

<IfModule mod\_authn\_dbd.c>

DBDriver mysql

DBDParams host=MYSQL,port=3306,user=admin,pass=admin,dbname=appDB

DBDMin 2

DBDKeep 8

DBDMax 20

DBDExptime 300

</IfModule>

<Directory "/var/www/apache-server/html/public">

AuthType None

Require all granted

</Directory>

<Directory "/var/www/apache-server/html/private">

AuthType Form

AuthName "Login"

Require valid-user

AuthFormProvider dbd

AuthDBDUserPWQuery "SELECT password FROM users WHERE username = %s"

Session On

SessionCookieName session path=/

ErrorDocument 401 /public/login.php

</Directory>

<Location "/logout">

SetHandler form-logout-handler

AuthFormLogoutLocation "http://localhost/public/logout.php"

SessionMaxAge 1

SessionCookieName session path=/

Session on

</Location>

<Location "/login">

SetHandler form-login-handler

AuthFormLoginRequiredLocation "http://localhost/public/login.php"

AuthFormLoginSuccessLocation "http://localhost/private/index.html"

AuthFormProvider dbd

AuthDBDUserPWQuery "SELECT password FROM users WHERE username = %s"

Продолжение листинга 1

AuthType form

AuthName "Login"

Session On

SessionCookieName session path=/

ErrorDocument 401 /login

</Location>

Настройка веб-сервера «Nginx» осуществляется в файле «nginx/nginx.conf», содержимое которого представлено на листинге 2.

Листинг 2 – Конфигурация веб-сервера «Nginx»

worker\_processes 1;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

server {

listen 80;

server\_name localhost;

location ~ \.(jpg|jpeg|gif|png|ico|css|zip|tgz|gz|rar|bz2|doc|xls|exe|pdf|ppt|txt|tar|mid|midi|wav|bmp|rtf|js)$ {

root /var/www/apache-server/html;

}

location ~ /\.ht {

deny all;

}

location / {

proxy\_pass http://apache;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $remote\_addr;

proxy\_connect\_timeout 120;

proxy\_send\_timeout 120;

proxy\_read\_timeout 180;

}

}

}

Для сборки образа «php-apache» нужно создать «Dockerfile». Его содержимое представлено на листинге 3.

Листинг 3 – «Dockerfile» для сборки образа «php-apache»

FROM php:7.2-apache

RUN mkdir -p /var/www/apache-server/html

RUN apt-get update -y && apt-get install -y apache2-utils libaprutil1-dbd-mysql

RUN a2enmod authn\_dbd

RUN docker-php-ext-install mysqli pdo pdo\_mysql

Теперь необходимо создать скрипт для инициализации базы данных. Содержимое скрипта представлено на листинге 4.

Листинг 4 – Скрипт инициализации базы данных

CREATE DATABASE IF NOT EXISTS appDB;

CREATE USER IF NOT EXISTS 'admin'@'%' IDENTIFIED WITH mysql\_native\_password BY 'admin';

GRANT SELECT,UPDATE,INSERT ON appDB.\* TO 'admin'@'%';

FLUSH PRIVILEGES;

USE appDB;

CREATE TABLE IF NOT EXISTS users (

ID INT(11) NOT NULL AUTO\_INCREMENT,

username VARCHAR(32) NOT NULL,

password VARCHAR(256) NOT NULL,

email VARCHAR(64) NOT NULL,

PRIMARY KEY (ID)

);

CREATE TABLE IF NOT EXISTS `goods`(

ID INT(11) PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(255) NOT NULL,

`description` TEXT NOT NULL

);

INSERT INTO users (

username,

password,

email

)

VALUES

('ezh1k', '$2y$10$8tMdwAC2xM9PSLqev5gBGODjQEnHduLh0mXhp2OYOWjdPTa4OPTwi', 's.park190802@gmail.com');

INSERT INTO `goods`(`title`, `description`)

VALUES

('Title #1', 'Description number 1'),

('Title #2', 'Description number 2'),

('Title #3', 'Description number 3'),

('Title #4', 'Description number 4'),

('Title #5', 'Description number 5');

Теперь нужно организовать все нужные контейнеры с помощью «Docker compose». Содержимое файла «docker-compose.yml» представлено на листинге 5.

Листинг 5 – Содержимое файла «docker-compose.yml»

version: '3'

services:

apache:

build: ./php-apache

volumes:

- ./src:/var/www/apache-server/html

- ./apache-conf/httpd.conf:/etc/apache2/sites-enabled/000-default.conf

depends\_on:

mysql:

condition: service\_healthy

nginx:

image: nginx:1.13

Продолжение листинга 5

ports:

- "80:80"

volumes:

- ./src:/var/www/apache-server/html

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

depends\_on:

- apache

mysql:

image: mysql

hostname: MYSQL

environment:

MYSQL\_ROOT\_PASSWORD: root

HOSTNAME: MYSQL

volumes:

- "./sql:/docker-entrypoint-initdb.d"

- "dbdata:/var/lib/mysql"

healthcheck:

test: [ "CMD", "mysqladmin" ,"ping", "-h", "localhost" ]

timeout: 1s

retries: 10

volumes:

dbdata:

Для того чтобы проверить работоспособность серверной конфигурации, нужно создать несколько страниц.

Создадим динамическую страницу «users.php». Содержимое файла представлено на листинге 6.

Листинг 6 – Страница «users.php»

<?php

try {

$conn = new PDO(

"mysql:host=MYSQL;dbname=appDB",

"admin",

"admin"

);

$query = "SELECT `ID`, `username`, `email` FROM `users`";

}

catch (PDOException $e) {

echo "<pre>";

echo "[ERROR]: Ошибка подключения к БД!" . PHP\_EOL;

echo $e->getMessage() . PHP\_EOL;

echo "</pre>";

die();

}

?>

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport"

Продолжение листинга 6

content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Пользователи</title>

</head>

<body>

<table border="1">

<thead>

<tr>

<th>ID</th>

<th>username</th>

<th>email</th>

</tr>

</thead>

<tbody>

<?php foreach ($conn->query($query) as $row): ?>

<tr>

<td><?= $row["ID"] ?></td>

<td><?= $row["username"] ?></td>

<td><?= $row["email"] ?></td>

</tr>

<?php endforeach; ?>

</tbody>

</table>

</body>

</html>

Создадим ещё одну динамическую страницу «goods.php». Содержимое файла представлено на листинге 7.

Листинг 7 – Страница «goods.php»

<?php

try {

$conn = new PDO(

"mysql:host=MYSQL;dbname=appDB",

"admin",

"admin"

);

$query = "SELECT `ID`, `title`, `description` FROM `goods`";

print\_r($conn->query($query));

}

catch (PDOException $e) {

echo "<pre>";

echo "[ERROR]: Ошибка подключения к БД!" . PHP\_EOL;

echo $e->getMessage() . PHP\_EOL;

echo "</pre>";

die();

}

?>

<!DOCTYPE html>

Продолжение листинга 7

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport"

content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Товары</title>

</head>

<body>

<table border="1">

<thead>

<tr>

<th>ID</th>

<th>title</th>

<th>description</th>

</tr>

</thead>

<tbody>

<?php foreach ($conn->query($query) as $row): ?>

<tr>

<td><?= $row["ID"] ?></td>

<td><?= $row["title"] ?></td>

<td><?= $row["description"] ?></td>

</tr>

<?php endforeach; ?>

</tbody>

</table>

</body>

</html>

Создадим две страницы для авторизации и регистрации соответственно. При этом для страницы регистрации ещё нужно написать обработчик формы.

Код страницы авторизации представлен на листинге 8.

Листинг 8 – Страница «login.php»

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Авторизация</title>

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

</head>

<body>

<div class="container">

<div class="form-box">

Продолжение листинга 8

<form action="/login" class="form" method="post">

<!-- Логин -->

<div class="form-field">

<label for="login" class="label">Логин</label>

<div class="form-controls">

<input

type="text"

id="login"

name="httpd\_username"

class="input"

placeholder="Логин"

required

>

</div>

</div>

<!-- Пароль -->

<div class="form-field">

<label for="password" class="label">Пароль</label>

<div class="form-controls">

<input

type="password"

id="password"

name="httpd\_password"

class="input"

placeholder="Пароль"

required

>

</div>

</div>

<!-- Отправка формы -->

<div class="form-field --text-center">

<input

type="submit"

name="login"

value="Авторизоваться"

class="button"

>

</div>

</form>

</div>

</div>

</body>

</html>

Код страницы регистрации представлен на листинге 9.

Листинг 9 – Страница «register.php»

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Регистрация</title>

Продолжение листинга 9

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

</head>

<body>

<div class="container">

<div class="form-box">

<form action="scripts/register.php" class="form" method="post">

<!-- Имя пользователя -->

<div class="form-field">

<label for="username" class="label">Имя пользователя</label>

<div class="form-controls">

<input

type="text"

id="username"

name="username"

class="input"

placeholder="Имя"

required

>

</div>

</div>

<!-- Email -->

<div class="form-field">

<label for="email" class="label">Email</label>

<div class="form-controls">

<input

type="email"

id="email"

name="email"

class="input"

placeholder="Email"

required

>

</div>

</div>

<!-- Пароль -->

<div class="form-field">

<label for="password" class="label">Пароль</label>

<div class="form-controls">

<input

type="password"

id="password"

name="password"

class="input"

placeholder="Пароль"

required

>

</div>

</div>

<!-- Отправка формы -->

<div class="form-field --text-center">

<input

type="submit"

Продолжение листинга 9

name="register"

value="Регистрация"

class="button"

>

</div>

</form>

</div>

</div>

</body>

</html>

Для неё есть обработчик формы, который обрабатывает данные формы и делает запрос к базе данных. Код обработчика представлен на листинге 10.

Листинг 10 – Код обработки формы регистрации

<?php

if (isset($\_REQUEST["register"])) {

$username = $\_REQUEST["username"];

$email = $\_REQUEST["email"];

$password = $\_REQUEST["password"];

// все поля формы должны быть заполнены

if (

empty($username)

|| empty($email)

|| empty($password)

) {

echo "[ERROR]: Неполные данные!" . PHP\_EOL;

die();

}

// установка соединения с БД

try {

$conn = new PDO(

"mysql:host=MYSQL;dbname=appDB",

"admin",

"admin"

);

// хеширование пароля

$encryptedPassword = password\_hash($password, PASSWORD\_DEFAULT);

$query = "INSERT INTO `users`(`username`, `password`, `email`) VALUES (:username, :password, :email)";

$preparedStmt = $conn->prepare($query);

$preparedStmt->bindParam(":username", $username);

$preparedStmt->bindParam(":password", $encryptedPassword);

$preparedStmt->bindParam(":email", $email);

// если удалось зарегистрироваться, то происходит редирект на страницу авторизации

if ($preparedStmt->execute()) {

header("Location: /public/login.php");

die();

}

// всё плохо

Продолжение листинга 10

echo "<pre>";

echo "[ERROR]: Ошибка выполнения запроса!" . PHP\_EOL;

echo "</pre>";

}

catch (PDOException $e) {

echo "<pre>";

echo "[ERROR]: Ошибка подключения к БД!" . PHP\_EOL;

echo $e->getMessage() . PHP\_EOL;

echo "</pre>";

}

}

# 3. ВЫВОДЫ

В ходе работы была создана сложная серверная конфигурация, состоящая из:

1. Apache;
2. Nginx;
3. PHP;
4. MySQL.

В этой конфигурации сервер nginx отображает статический контент, а apache – динамический.

# 4. ОТВЕТЫ НА ВОПРОСЫ

1. Что такое веб-сервер?

Веб-сервер — это 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.

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

Сервер приложений — это программная платформа, предназначенная для эффективного исполнения процедур (программ, скриптов), на которых построены приложения. Сервер приложений действует как набор компонентов, доступных разработчику программного обеспечения через API, определённый самой платформой.

Основное отличие веб-сервера от сервера приложений заключается в том, что веб-сервер предназначен для обслуживания статических страниц, например HTML и CSS, тогда как сервер приложений отвечает за генерацию динамического содержимого путем выполнения кода на стороне сервера, например, JSP, EJB и т. п.

1. Кратко опишите историю развития интернета в рамках развития веб-серверов.

От статических web-страниц, трудных в изменении, мы пришли в web-серверам, способным работать с динамическими данными.

1. Кратко опишите протокол HTTP.

HTTP — протокол прикладного уровня передачи данных, изначально — в виде гипертекстовых документов в формате HTML, в настоящее время используется для передачи произвольных данных.

1. 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.

1. Опишите цели и задачи веб-сервера.

Основная цель и задача веб-сервера - обработка HTTP-запросов и возврат пользователю их результатов.

1. Опишите технологию SSI.

SSI (Server Side Includes — включения на стороне сервера) - язык для динамической «сборки» веб-страниц на сервере из отдельных составных частей и выдачи клиенту полученного HTML документа.

1. Что такое система управления контентом?

Система управления содержимым — информационная система или компьютерная программа, используемая для обеспечения и организации совместного процесса создания, редактирования и управления содержимым, иначе — контентом.

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

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).

1. Что такое CGI?

CGI — стандарт интерфейса, используемого внешней программой для связи с веб-сервером.

1. Как работает система с использованием интерфейс шлюза - CGI?

Обобщенный алгоритм работы через CGI можно представить в следующем виде: клиент запрашивает CGI-приложение по его URI; Веб-сервер принимает запрос и устанавливает переменные окружения, через них приложению передаются данные и служебная информация; Веб-сервер перенаправляет запросы через стандартный поток ввода (stdin) на вход вызываемой программы; CGI-приложение выполняет все необходимые операции и формирует результаты в виде HTML; Сформированный гипертекст возвращается веб-серверу через стандартный поток вывода (stdout). Сообщения об ошибках передаются через stderr; Веб-сервер передает результаты запроса клиенту.

1. Назовите достоинства и недостатки CGI.

Достоинства: процесс CGI скрипта не зависит от Веб-сервера и, в случае падения, никак не отразится на работе последнего; может быть написан на любом языке программирования; поддерживается большинством Веб-серверов.

Недостатки: самым большим недостатком этой технологии являются повышенные требования к производительности веб-сервера. Дело в том, что каждое обращение к CGI-приложению вызывает порождение нового процесса, со всеми вытекающими отсюда накладными расходами. Если же приложение написано с ошибками, то возможна ситуация, когда оно, например, зациклится. Браузер прервет соединение по истечении тайм-аута, но на серверной стороне процесс будет продолжаться, пока администратор не снимет его принудительно.

1. Что такое FastCGI?

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

1. Назовите основные отличия CGI от FastCGI.

В отличие от CGI, в FastCGI для каждого скрипта не запускается отдельный процесс, благодаря чему меньше расходуются ресурсы.

1. Что такое менеджер процессов?

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.

1. Что такое PHP-FPM?

php-fpm — это fastcgi process manager. Он представляет из себя отдельную службу, которая работает независимо от какого-либо веб сервера. Он может сам принимать запросы от веб сервера через unix-сокет или через сетевое соединение. Другими словами, вы можете держать сайты на одном сервере, а php-скрипты исполнять на другом.

1. Что такое Spawn-fcgi?

Spawn-fcgi используется для запуска удаленных и локальных FastCGI процессов.

1. Что такое Lighttpd?

lighttpd — веб-сервер, разрабатываемый с расчетом на скорость и защищенность, 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.

1. Что такое chroot окружение?

Chroot окружение — способ запуска программ, системный вызов и просто команда, позволяющая изменить корневой каталог в системе.

1. Опишите механизм взаимодействия серверов с использованием FastCGI.

В то время как CGI-программы взаимодействуют с сервером через STDIN и STDOUT запущенного CGI-процесса, FastCGI-процессы используют Unix Domain Sockets или TCP/IP для связи с сервером. Это даёт следующее преимущество над обычными CGI-программами: FastCGI-программы могут быть запущены не только на этом же сервере, но и где угодно в сети. Также возможна обработка запросов несколькими FastCGI-процессами, работающими параллельно.

1. Опишите процесс выбора встроенного или внешнего менеджера процессов.

Надежнее встроенный менеджер.

1. Что такое интерфейс шлюза?

Аппаратный маршрутизатор или программное обеспечение для сопряжения компьютерных сетей, использующих разные протоколы (например, локальной и глобальной).

1. Что такое SCGI?

SCGI (Simple Common Gateway Interface) — простой общий интерфейс шлюза — разработан как альтернатива CGI и во многом аналогичен FastCGI, но более прост в реализации. Все, что применимо к FastGCI, справедливо и для SCGI.

1. Что такое PCGI?

PCGI (Perl Common Gateway Interface) - библиотека Perl для работы с интерфейсом CGI, долгое время являлась основным вариантом работы с Perl Приложениями через CGI, отличается хорошей производительностью при скромных потребностях в ресурсах и неплохой защиты от перегрузки.

1. Что такое PSGI?

PSGI (Perl Web Server Gateway Interface) — технология взаимодействия веб-сервера и сервера приложений для Perl. Если PCGI представляет собой инструмент для работы с классическим CGI-интерфейсом, то PSGI более напоминает FastCGI. PSGI-сервер представляет среду для выполнения Perl-приложений которая постоянно запущена в качестве службы и может взаимодействовать с веб-сервером через TCP/IP или UNIХ-сокеты и предоставляет Perl приложениям те же преимущества, что и FastCGI.

1. Что такое WSGI?

WSGI (Web Server Gateway Interface) — предназначен для взаимодействия веб-сервера с сервером приложений для программ, написанных на языке Python.

1. Опишите механизм взаимодействия серверов Apache и PHP.

Apache обычно обслуживает файлы, извлекая файл и отправляя поток вниз по HTTP-соединению. Однако с PHP Apache извлекает файл, передает его в двоичный файл PHP и отправляет выход поток из команды вниз по HTTP-соединению.

1. Опишите преимущества веб-сервера Apache.

* Высокий уровень надежности;
* Гибкие настройки;
* Свободный доступ к программе;
* Регулярные обновления и патчи;
* Удобство и легкость настройки.

1. Опишите недостатки веб-сервера Apache.

* Проблемы с производительностью на высоконагруженных сайтах;
* Большое кол-во параметров настройки может привести к уязвимости в конфигурации;
* Некоторая вероятность наличия вредоносного кода в модулях от независимых разработчиков.

1. Опишите архитектуру веб-сервера Apache.

Ядро Apache включает в себя основные функциональные возможности, такие как обработка конфигурационных файлов, протокол HTTP и система загрузки модулей.

Система конфигурации Apache, основанная на текстовых конфигурационных файлах.

Apache имеет встроенный механизм виртуальных хостов. Он позволяет полноценно обслуживать на одном IP-адресе множество сайтов (доменных имён), отображая для каждого из них собственное содержимое.

1. Опишите функции ядра веб-сервера Apache.

Основные функции ядра:

* Передача данных по HTTP;
* Обработка файлов;
* Загрузка и поддержка модулей.

1. Опишите конфигурацию веб-сервера Apache.

Конфигурацию Apache можно разделить на три основных уровня:

* Конфигурация сервера;
* Конфигурация виртуального хоста;
* Конфигурация уровней каталога.

1. Что такое URI, URL и чем они различаются.

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.

URN – имя ресурса в сети, определяет только название ресурса, но не говорит как к нему подключиться.

# 5. ССЫЛКА НА УДАЛЁННЫЙ РЕПОЗИТОРИЙ

Ссылка на удалённый репозиторий проекта <https://github.com/Sergey-hub02/RSCHIR>.

# 6. СПИСОК ЛИТЕРАТУРЫ

1. Официальная документация докера [Электронный ресурс] – URL: <https://docs.docker.com> (дата обращения 03.09.2022)
2. Установка и настройка PHP [Электронный ресурс] – URL: <https://www.php.net/manual/ru/install.php> (дата обращения 03.09.2022)
3. Docker Compose [Электронный ресурс] – URL: <https://habr.com/ru/company/ruvds/blog/450312> (дата обращения 03.09.2022)
4. Docker Hub [Электронный ресурс] – URL: <https://hub.docker.com> (дата обращения 03.09.2022)