|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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:**

**Студент группы** ИКБО-20-19 Анваржонов Ж Т

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

(подпись студента)

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

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

(подпись руководителя)

Работа представлена «20» ноября 2021 г.

Допущен к работе «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Москва 2021

**Цель работы**

В задании предлагается 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 №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 root-директории с базовых на другие. Для доступа к администрированию предусмотреть базовую авторизацию и аутентификацию с применением httaccess и хранением пользователей в БД.

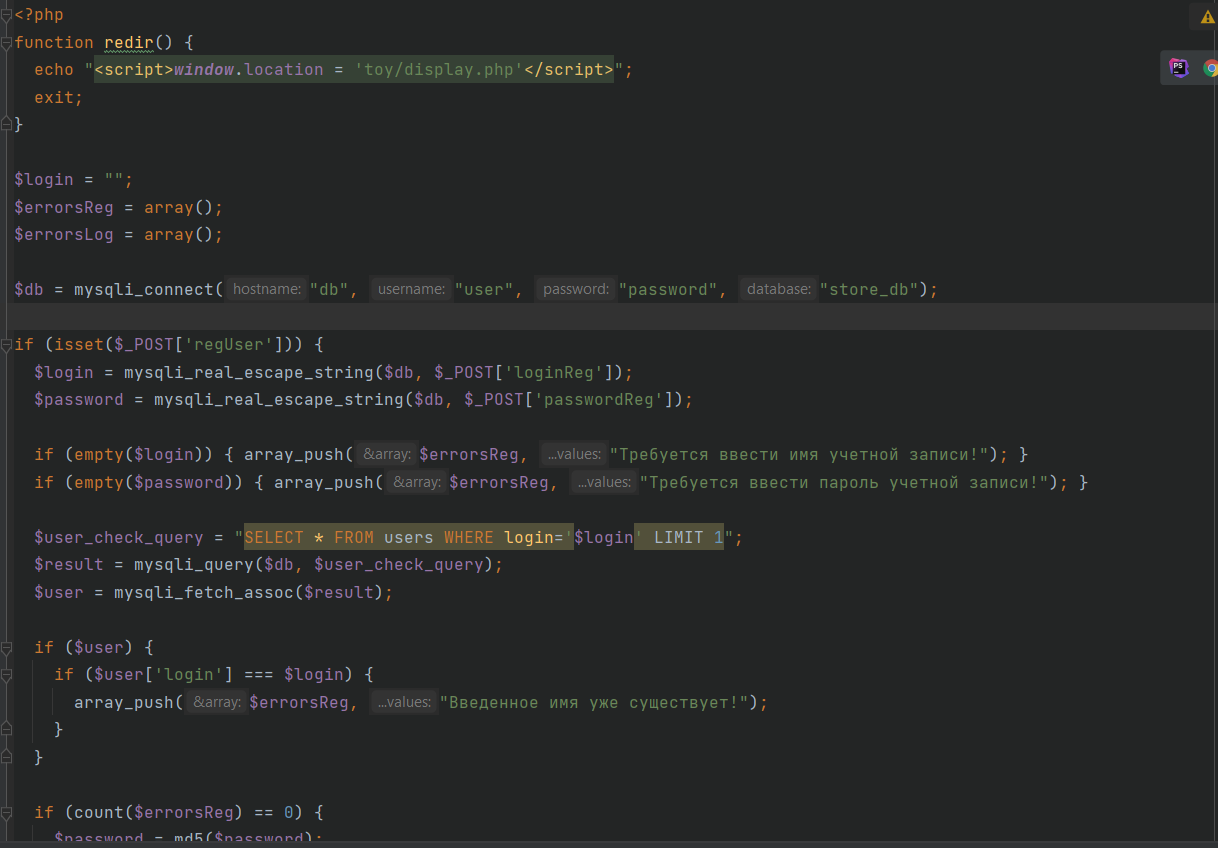
Для тестирования данной конфигурации предполагается создать тестовое веб-приложение на тему “Магазин игрушек”

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 задания**

<!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" />  
 <script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.1.1/js/bootstrap.min.js"></script>  
 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.1.1/css/bootstrap.min.css" />  
 <?php include('server.php') ?>  
 <title>Cеть магазинов игрушек "ToyBoy"</title>  
</head>  
<style type="text/css">  
 .btn-primary {  
 background-color: green **!important**;  
 }  
</style>  
  
<body>  
 <div class="container">  
 <div class="container row mw-100">  
 <div class="justify-content-center">  
 <br>  
 <strong>Для входа после регистрации:</strong>  
 </div>  
 <div class="justify-content-center">  
 <form method="post" action="index.php" class="border p-3">  
 <?php include('errorsLog.php'); ?>  
 <div class="">  
 <label for="login" class="form-label">Логин</label>  
 <input type="text" class="form-control" id="login" name="loginLog" aria-describedby="loginHelp" />  
 </div>  
 <div class="">  
 <label for="passwordInput" class="form-label">Пароль</label>  
 <input type="password" class="form-control" id="passwordInput" name="passwordLog" />  
 </div>  
 <div class="text-center">  
 <button type="submit" class="btn btn-primary mt-2" name="loginUser">Войти</button>  
 </div>  
 </form>  
 </div>  
 <div class="justify-content-center">  
 <br>  
 <strong>Для первичной регистрации:</strong>  
 </div>  
 <div class="justify-content-center">  
 <form method="post" action="index.php" class="border p-3">  
  
 <?php include('errorsReg.php'); ?>  
 <div class="">  
 <label for="login" class="form-label">Логин</label>  
 <input type="text" class="form-control" id="login" name="loginReg" aria-describedby="loginHelp" />  
 </div>  
 <div class="">  
 <label for="passwordInput" class="form-label">Пароль</label>  
 <input type="password" class="form-control" id="passwordInput" name="passwordReg" />  
 </div>  
 <div class="text-center">  
 <button type="submit" class="btn btn-primary mt-2" name="regUser">  
 Зарегистрироваться  
 </button>  
 </div>  
 </form>  
 </div>  
 </div>  
 </div>  
</body>  
  
</html>

Рисунок 1 – файл index.php

 Рисунок 2 – файл server.php

<?php  
  
$con = new mysqli('db','user', 'password','store\_db');  
  
if (!$con){  
 die(mysqli\_error($con));  
}  
  
?>

Рисунок 2 – файл connect.php

CREATE DATABASE IF NOT EXISTS store\_db default charset cp1251;  
CREATE USER IF NOT EXISTS 'user'@'%' IDENTIFIED BY 'password';  
GRANT SELECT,UPDATE,INSERT ON store\_db.\* TO 'user'@'%';  
FLUSH PRIVILEGES;  
  
use store\_db;  
CREATE TABLE IF NOT EXISTS users (  
 ID INT(11) NOT NULL AUTO\_INCREMENT,  
 login VARCHAR(20) NOT NULL,  
 password VARCHAR(40) NOT NULL,  
 PRIMARY KEY (ID)  
);  
  
INSERT INTO users(login, password)  
VALUES ('admin', 'admin');  
  
  
CREATE TABLE IF NOT EXISTS toys (  
 ID INT(11) NOT NULL AUTO\_INCREMENT,  
 name VARCHAR(30) NOT NULL,  
 price int(10) NOT NULL,  
 PRIMARY KEY (ID)  
);  
CREATE TABLE IF NOT EXISTS stores (  
 ID INT(11) NOT NULL AUTO\_INCREMENT,  
 name VARCHAR(40) NOT NULL,  
 address VARCHAR(100) NOT NULL,  
 PRIMARY KEY (ID)  
);  
  
INSERT INTO toys (name, price)  
VALUES  
('Milas','10312'),  
('Fasan','421'),  
('Iron man','2131'),  
('Spider man','552'),  
('Joker toy','213'),  
('Jonks Boy','3000'),  
('Near of Plan','603'),  
('Kangeroi','5004'),  
('Baby Stel','4190');  
  
  
INSERT INTO stores (name, address) VALUES  
('Ручеек','ул. Ручейкова, д.9'),  
('Cолнышнко','ул. Салютов, д. 228к12'),  
('Антошка','ул. Антонова, д. 10к1'),  
('Чайка','ул. Илфайза , д. 2'),  
('Колокольчик','ул. Красный Казанец, д. 1');

Рисунок 3 – файл init.sql

<?php  
include '../connect.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>CRUD operation</title>  
 <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"  
 integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">  
  
</head>  
  
<body>  
  
  
<div class="container">  
 <ul class="nav nav-tabs mt-2 justify-content-center">  
 <li class="nav-item">  
 <a class="nav-link" href="../toy/display.php">Toys</a>  
 </li>  
 <li class="nav-item">  
 <a class="nav-link active" href="#">Stores</a>  
 </li>  
 <li class="nav-item dropdown">  
 <a class="nav-link dropdown-toggle" data-toggle="dropdown" href="#" role="button" aria-haspopup="true"  
 aria-expanded="false">Static</a>  
 <div class="dropdown-menu">  
 <a class="dropdown-item" href="http://localhost:8001/about.html">About</a>  
 <a class="dropdown-item" href="http://localhost:8001/contacts.html">Contacts</a>  
 <div class="dropdown-divider"></div>  
 </div>  
 </li>  
  
 </ul>  
  
 <div class="text-center">  
 <button class="btn btn-primary my-5"><a href="user.php" class="text-light">ADD Store</a></button>  
 </div>  
 <table class="table table-striped table-dark">  
 <thead>  
 <tr>  
 <th scope="col">ID</th>  
 <th scope="col">Store name</th>  
 <th scope="col">Store address</th>  
 <th scope="col">Operations</th>  
  
 </tr>  
 </thead>  
 <tbody>  
  
 <?php  
 $sql = "select \* from `stores`";  
 $result = mysqli\_query($con, $sql);  
 if ($result) {  
 while ($row = mysqli\_fetch\_assoc($result)) {  
 $id = $row['ID'];  
 $name = $row['name'];  
 $address = $row['address'];  
 echo '<tr>  
 <th scope="row">' . $id . '</th>  
 <td>' . $name . '</td>  
 <td>' . $address . '</td>  
 <td>  
 <button class = "btn btn-primary"><a href="update.php?updateid=' . $id . '" class="text-light">Update</a></button>  
 <button class = "btn btn-danger"> <a href="delete.php?deleteid=' . $id . '" class="text-light">Delete</a></button>  
 </td>   
 </tr>';  
 }  
 }  
 ?>  
  
 </tbody>  
 </table>  
</div>  
</body>  
  
</html>

Рисунок 4 – файл stores/display.php

Статические файлы



Рисунок 5 – файл about.php

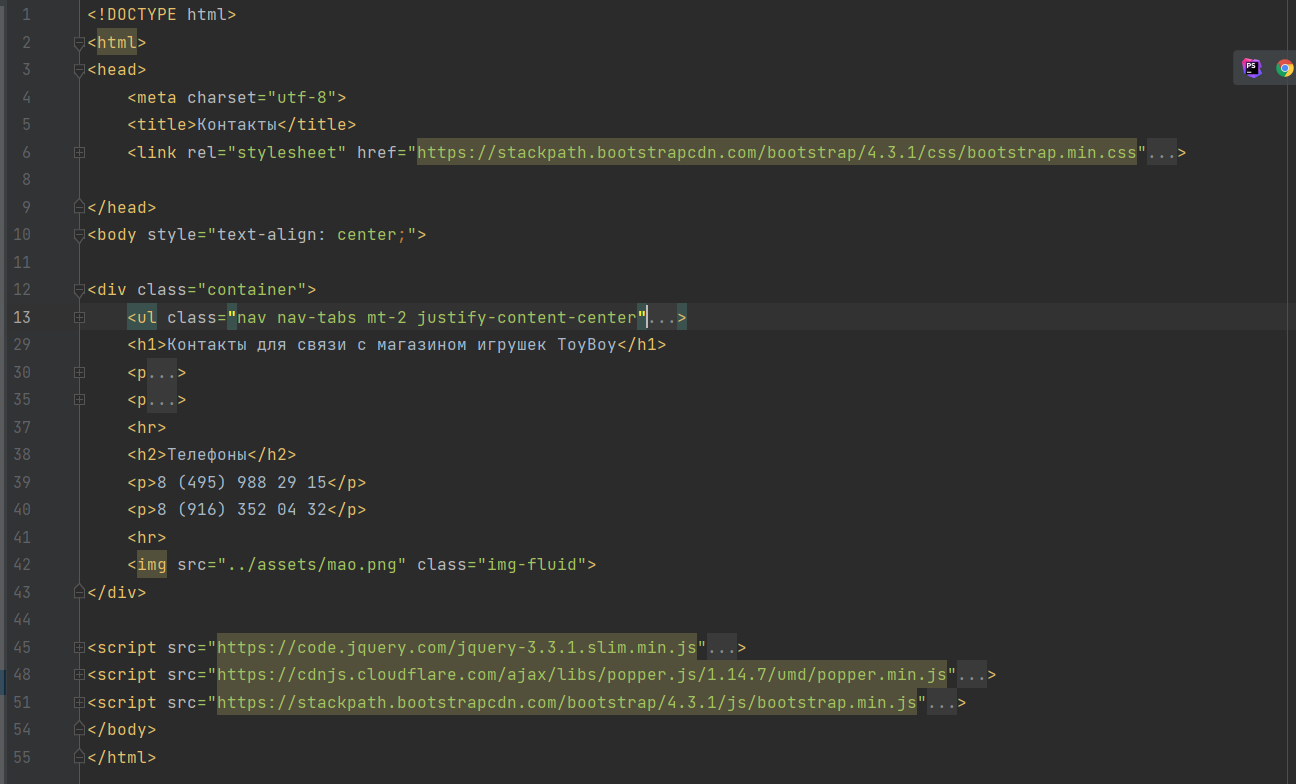


Рисунок 6 – файл contacts.php

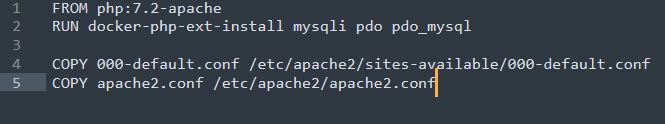


Рисунок 5 – файл Dockerfile в директории apache

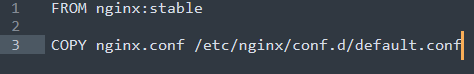


Рисунок 6 – файл Dockerfile в директории nginx

version: "3"  
  
services:  
 db:  
 image: mariadb  
 volumes:  
 - ./database:/docker-entrypoint-initdb.d  
 restart: always  
 environment:  
 - MYSQL\_ALLOW\_EMPTY\_PASSWORD=true  
 static\_nginx:  
 build: ./nginx  
 volumes:  
 - ./nginx:/var/nginx  
 - ./assets:/var/nginx/assets  
 ports:  
 - 8001:80  
 dynamic\_apache:  
 build: ./apache  
 volumes:  
 - ./apache:/var/apache  
 - ./assets:/var/apache/assets  
 ports:  
 - 8002:80  
 depends\_on:  
 - db  
 phpmyadmin:  
 image: phpmyadmin  
 restart: always  
 ports:  
 - 8080:80  
 environment:  
 - PMA\_ARBITRARY=1  
 depends\_on:  
 - db

Рисунок 7 – файл docker-compose.yml

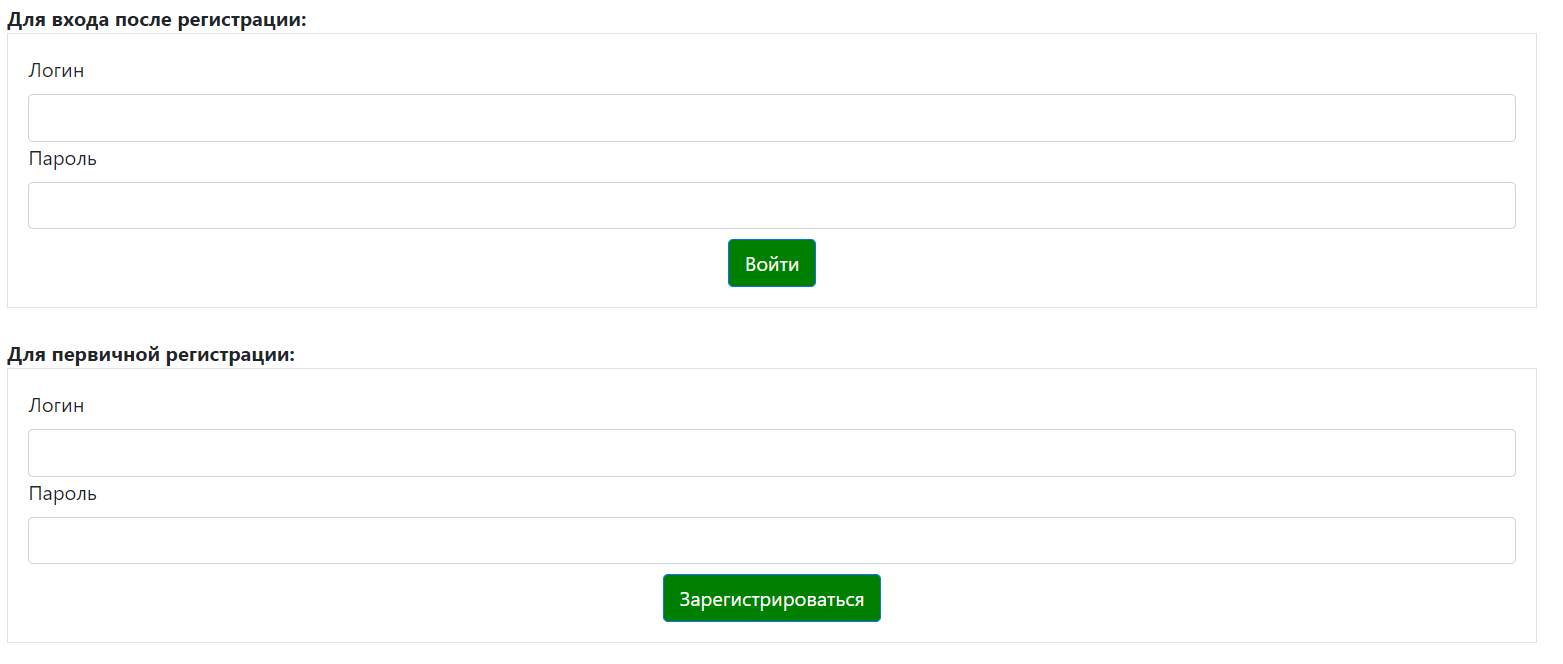


Рисунок 8 – Страница с регистрацией и авторизацией

******

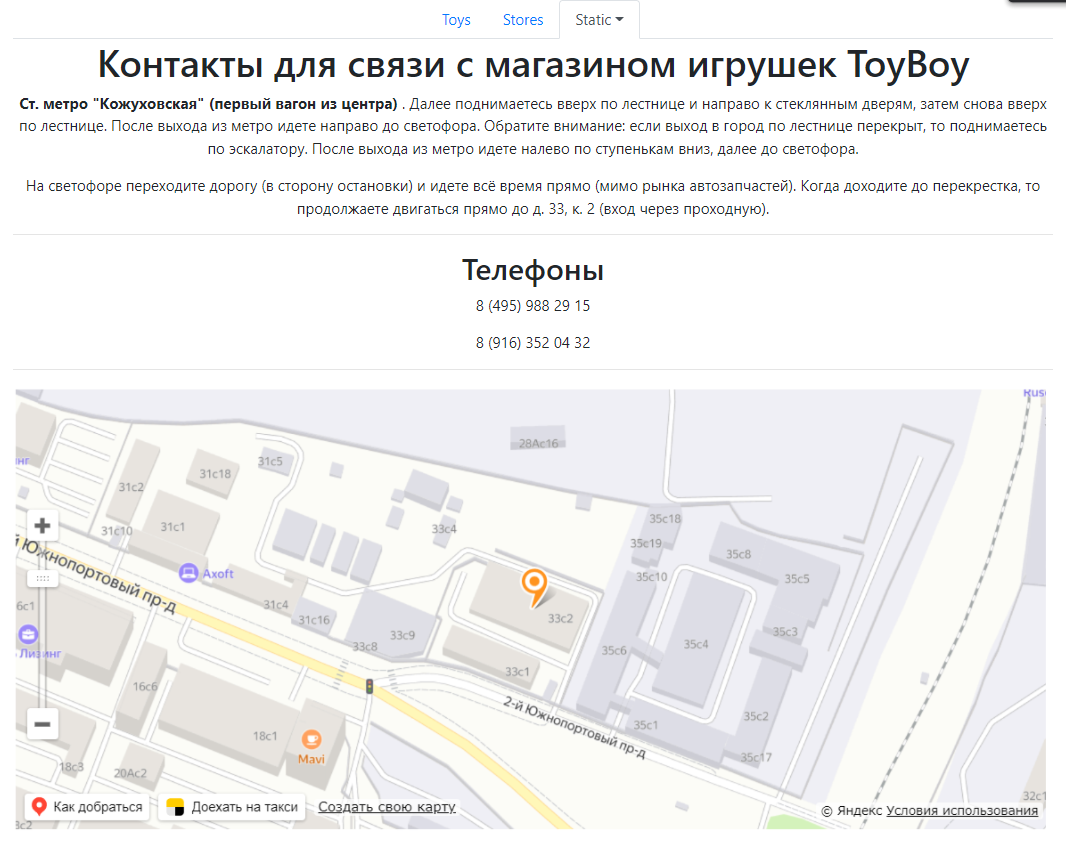
******

Рисунок 9 – Страницы со статическим содержимым

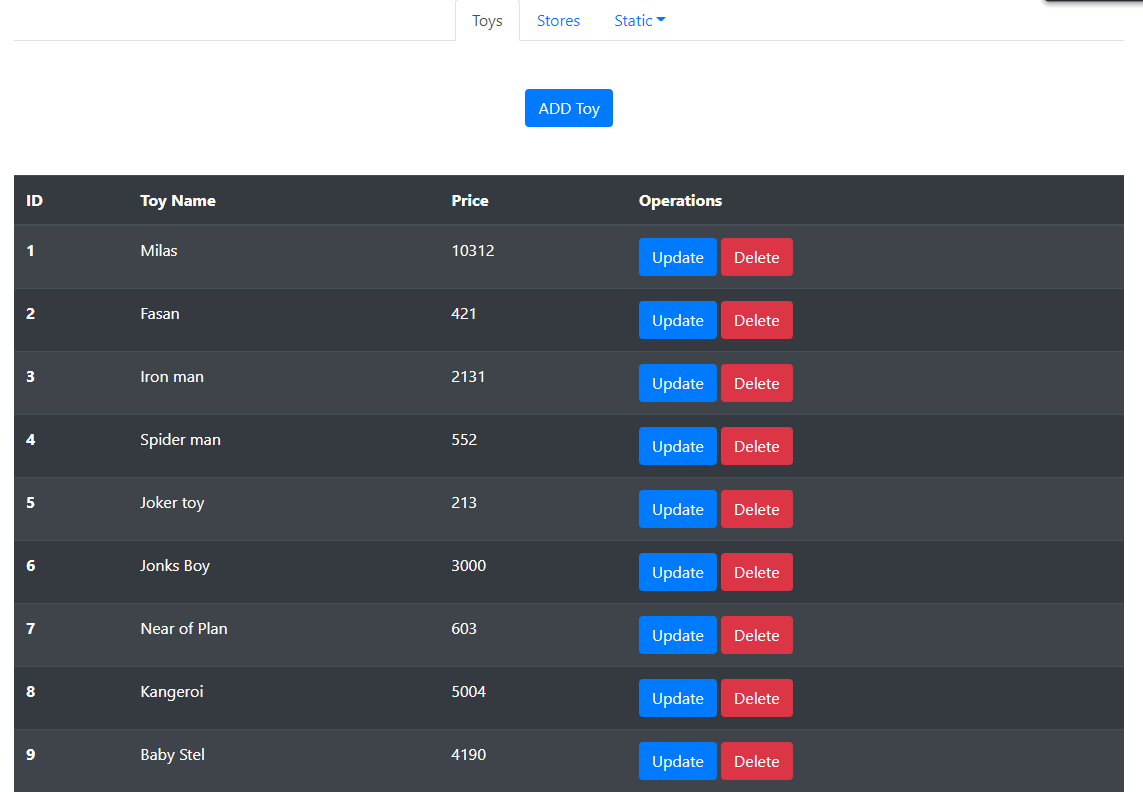


Рисунок 10 – Страница с выводом всех записей таблицы toys из базы данных

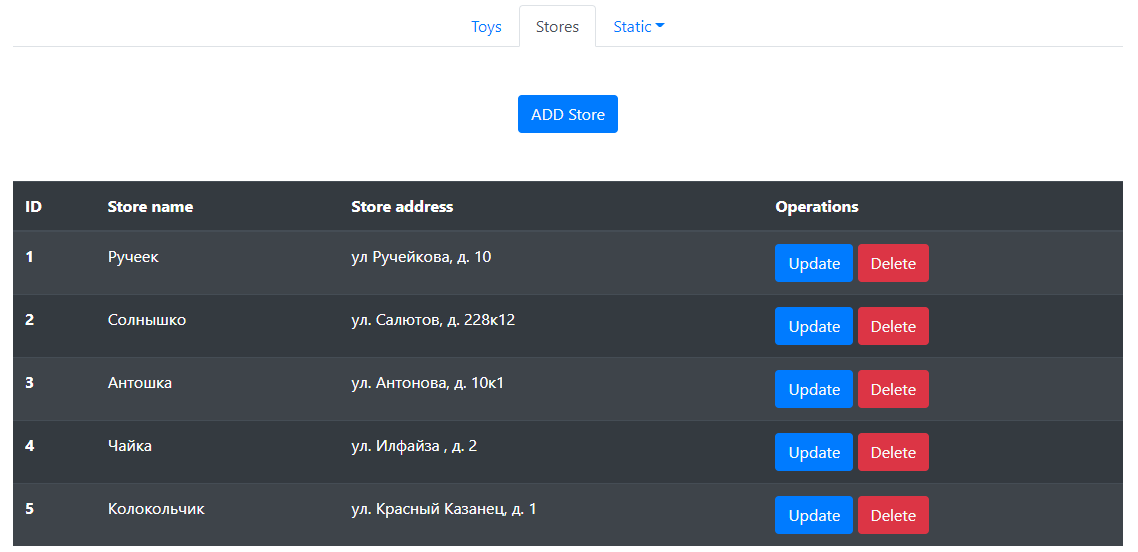


Рисунок 11 – Страница с выводом всех записей таблицы stores из базы данных

Вывод

В ходе выполнения практической была создана сложная серверная конфигурация, состоящая из связки apache+nginx+php+база данных. В этой конфигурации предполагается акселерированное проксирование без кэширования.

**Ответы на вопросы к практической работе**

1. *Что такое веб-сервер?*

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.

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

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 путём выполнения кода на стороне сервера, например, JSP, EJB и т. п.

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

Самый первый сервер появился в прошлом веке - 6 августа 1991 года. Его создатель – учёный из Британии Тимоти Джон Бернерс-Ли. Идеи о том, чтобы создать всемирную компьютерную сеть под названием World Wide Web (WWW), появились раньше.

Первая идея была выдвинута учёным из Британии в 1989 году. Именно тогда началась история серверов. Что касается детальной разработки, то она началась в 1990 году, когда у учёного появился доступ к компьютеру NeXT (именно он выступил в качестве первого интернет-сервера, браузера и редактора сети). После этого британец получил возможность воплощать свои идеи в реальность. Финальная доработка проекта произошла в 1991 году, после чего он был рассмотрен комитетом CS. В августе этого года мир увидел первый Интернет-сервер.

До конца года появление серверов состоялось также в других исследовательских центрах Соединенных Штатов Америки и Европы. По состоянию на ноябрь 1992 года в мире функционировало уже 26 интернет-серверов, а ещё год спустя их количество перевалило за две сотни. В 1995 году интернет WWW начал масштабироваться и стал глобальным. Сегодня Интернет используют сотни миллионов пользователей по всему миру, а количество серверов постоянно растёт. Первый Web-сайт был создан по адресу http://info.cern.ch, однако сейчас этот сайт находится на хранении в архиве. На сайте располагалась информация о Всемирной паутине, рекомендации по установке веб-сервера и получению браузера. Также этот сайт стал первым каталогом, так как позже на нём были размещены ссылки на другие сайты.

1. *Кратко опишите протокол HTTP.*

HTTP (англ. HyperText Transfer Protocol — «протокол передачи гипертекста») — 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.*

Основой HTTP является технология «клиент-сервер», то есть предполагается существование: Потребителей (клиентов), которые инициируют соединение и посылают запрос; Поставщиков (серверов), которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом.

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

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.

Любой веб сервер, для удобства его использования пользователями, должен иметь удобный веб-браузер. Он передает веб серверу запросы, преобразованные в URL-адреса интернет — ресурсов.

Наряду со стандартными функциями, некоторые веб серверы имеют дополнительные. Так, к примеру, соответствующее программное обеспечение может фиксировать число обращений пользователей к тому или иному ресурсу, записывать их в отдельный журнал. А еще они могут поддерживать HTTPS, что не маловажно для защищенного соединения между сайтами и пользователями. Зачастую веб-сервер устанавливается вместе с мейл-сервером. Это позволяет пользователям быстро переходить на страничку почты прямо с сайта, нажав всего лишь на одну гиперссылку.

1. *Опишите технологию SSI.*

SSI (Server Side Includes — включения на стороне сервера) — несложный язык для динамической «сборки» веб-страниц на сервере из отдельных составных частей и выдачи клиенту полученного HTML-документа. Реализован в веб-сервере Apache при помощи модуля mod\_include. Включённая в настройках по умолчанию веб-сервера возможность позволяет подключать HTML-файлы, поэтому для использования инструкций файл должен оканчиваться расширением .shtml, .stm или .shtm

1. *Что такое система управления контентом?*

Систе́ма управле́ния содержи́мым (англ. Content management system, CMS, система управления контентом) — информационная система или компьютерная программа, используемая для обеспечения и организации совместного процесса создания, редактирования и управления содержимым, иначе — контентом (от англ. content).

1. *Верно ли, что сервер приложения умеет работать с протоколом HTTP?* Да, веб-сервер
2. *Что такое CGI?*

CGI (от англ. 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». По сути позволяет использовать консоль ввода и вывода для взаимодействия с клиентом.

1. *Как работает система с использованием интерфейс шлюза - CGI?* Рисунок (17)

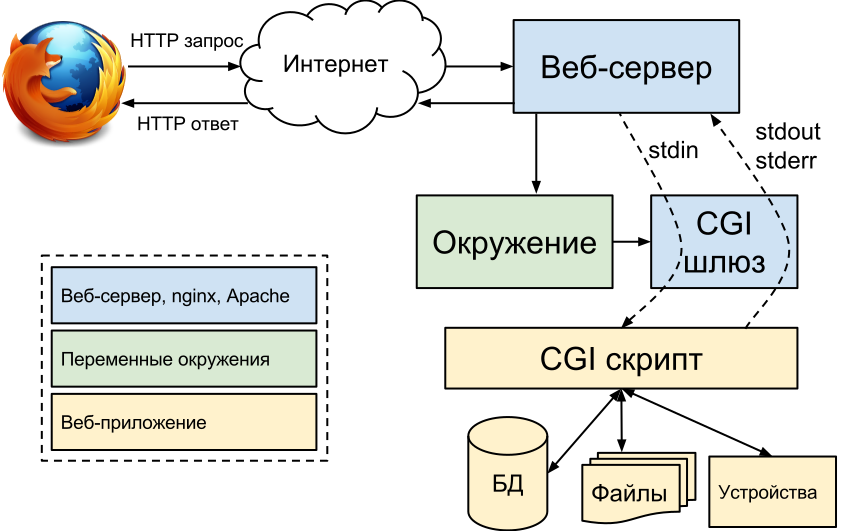
**

Рисунок 17 – Демонстрация работы CGI

1. *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.

1. *Что такое FastCGI?*

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

1. *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.

1. *Что такое менеджер процессов?*

ПО для контроля над текущими процессами веб-сайта

1. *Что такое PHP-FPM?*

FPM (FastCGI Process Manager, менеджер процессов FastCGI) является альтернативной реализацией PHP FastCGI с несколькими дополнительными возможностями, обычно используемыми для высоконагруженных сайтов.

1. *Что такое Spawn-fcgi?*

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

1. *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.

1. *Что такое chroot окружение?*

Chroot-окружение – это системный вызов, который временно перемещает root каталог в новую папку. Как правило, root каталог находится в «/». Но при помощи chroot можно задать другой каталог, который будет служить как root-каталог в окружении chroot. Любые приложения, которые запускаются внутри изолированного окружения, в принципе не могут взаимодействовать с остальной операционной системой. Кроме того, не-рутовый пользователь (non-root), помещённый в chroot-окружение, не сможет перемещаться по иерархии каталогов.

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

Отдача запроса от браузера к FastCGI-приложению

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

Процесс выбора отличается для разных задач, быстродействие, ОС и т.д

1. *Что такое интерфейс шлюза?*

Интерфейс используемый внешней программой для связи с веб-сервером

1. *Что такое SCGI?*

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.

1. *Что такое PCGI*

PCGI (Perl Common Gateway Interface) — библиотека к языку программирования Perl для работы с интерфейсом CGI (Common Gateway Interface).

1. *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.

1. *Что такое WSGI?*

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

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

Apache получает запрос, после того как Apache решает что это PHP-файл он отдает его PHP интерпретатору. Когда PHP получает файл о читает его и выполняет весь PHP код, затем он отдает выходные данные серверу Apache, который в свою очередь отсылает их браузеру.

1. *Опишите преимущества веб-сервера Apache.*

Основные достоинства Apache - надежность, безопасность и гибкость настройки. Apache позволяет подключать различные модули, добавляющие в него новые возможности - например, можно подключить модуль, обеспечивающий поддержку РНР или любого другого Web-ориентированного языка программирования.

1. *Опишите недостатки веб-сервера Apache.*

Недостатки - отсутствие удобного графического интерфейса администратора. Настройка Apache осуществляется путем редактирования его конфигурационного файла. В Интернете можно найти простые конфигураторы Apache, но их возможностей явно не хватает для настройки всех функций Web-сервера.

1. *Опишите архитектуру веб-сервера Apache.*

Ядро, Система конфигурации (Конфигурация сервера, Конфигурация виртуального хоста, Конфигурация уровня каталога), Многопроцессорные модели (MPM), Система модулей, Механизм виртуальных хостов.

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

Ядро Apache включает в себя основные функциональные возможности, такие как обработка конфигурационных файлов, протокол HTTP и система загрузки модулей. Ядро (в отличие от модулей) полностью разрабатывается Apache Software Foundation, без участия сторонних программистов.

Теоретически ядро apache может функционировать в чистом виде, без использования модулей. Однако функциональность такого решения крайне ограничена.

Ядро Apache полностью написано на языке программирования C.

1. *Опишите конфигурацию веб-сервера Apache.*

Система конфигурации Apache основана на текстовых конфигурационных файлах. Имеет три условных уровня конфигурации:

* Конфигурация сервера (httpd.conf). Директивы конфигурации сгруппированы в три основных раздела: директивы, управляющие процессом Apache в целом (глобальное окружение); директивы, определяющие параметры «главного» сервера, или сервера «по умолчанию», который отвечает на запросы, которые не обрабатываются виртуальными хостами (определяют также установки по умолчанию для всех остальных виртуальных хостов); установки для виртуальных хостов, позволяющие обрабатывать запросы Web одним-единственным сервером Apache, но направлять по раздельным адресам IP или именам хостов.
* Конфигурация виртуального хоста (httpd.conf c версии 2.2, extra/httpd-vhosts.conf).
* Конфигурация уровня каталога (.htaccess).

Имеет собственный язык конфигурационных файлов, основанный на блоках директив. Практически все параметры ядра могут быть изменены через конфигурационные файлы, вплоть до управления MPM. Большая часть модулей имеет собственные параметры.

Часть модулей использует в своей работе конфигурационные файлы операционной системы (например /etc/passwd и /etc/hosts). Помимо этого, параметры могут быть заданы через ключи командной строки.

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

URI (англ. Uniform Resource Identifier) — унифицированный (единообразный) идентификатор ресурса. URI — последовательность символов, идентифицирующая абстрактный или физический ресурс. Ранее назывался Universal Resource Identifier — универсальный идентификатор ресурса.

URI — символьная строка, позволяющая идентифицировать какой-либо ресурс: документ, изображение, файл, службу, ящик электронной почты и т. д. Прежде всего, речь идёт о ресурсах сети Интернет и Всемирной паутины. URI предоставляет простой и расширяемый способ идентификации ресурсов. Расширяемость URI означает, что уже существуют несколько схем идентификации внутри URI, и ещё больше будет создано в будущем.

URI является либо URL, либо URN, либо одновременно обоими.

URL — это URI, который, помимо идентификации ресурса, предоставляет ещё и информацию о местонахождении этого ресурса. А URN — это URI, который только идентифицирует ресурс в определённом пространстве имён (и, соответственно, в определённом контексте), но не указывает его местонахождение. Например, URN urn:ISBN:0-395-36341-1 — это URI, который указывает на ресурс (книгу) 0-395-36341-1 в пространстве имён ISBN, но, в отличие от URL, URN не указывает на местонахождение этого ресурса: в нём не сказано, в каком магазине её можно купить или на каком сайте скачать. Впрочем, в последнее время появилась тенденция говорить просто URI о любой строке-идентификаторе, без дальнейших уточнений. Так что, возможно, термины URL и URN скоро уйдут в прошлое.

Поскольку URI не всегда указывает на то, как получить ресурс, в отличие от URL, а только идентифицирует его, это даёт возможность описывать с помощью RDF (Resource Description Framework) ресурсы, которые не могут быть получены через Интернет (например, личность, автомобиль, город и проч.).

Список использованных источников

1. Сейерс, Э. Х. Docker на практике / Э. Х. Сейерс, А. Милл ; перевод с английского Д. А. Беликов. — Москва : ДМК Пресс, 2020. — 516 с. — ISBN 978-5-97060-772-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/131719 (дата обращения: 02.10.2021). — Режим доступа: для авториз. пользователей.
2. Одиночкина, С. В. Web-программирование PHP : учебно-методическое пособие / С. В. Одиночкина. — Санкт-Петербург : НИУ ИТМО, 2012. — 79 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/43562 (дата обращения: 02.10.2021). — Режим доступа: для авториз. пользователей.
3. Ульман, Л. MySQL / Л. Ульман. — Москва : ДМК Пресс, 2008. — 352 с. — ISBN 5-94074-229-7. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/1241 (дата обращения: 02.10.2021). — Режим доступа: для авториз. пользователей.
4. Даева, С. Г. Информационные системы и технологии: работа с веб-сервером nginx : учебно-методическое пособие / С. Г. Даева. — Москва : РТУ МИРЭА, 2020. — 75 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/167619 (дата обращения: 02.10.2021). — Режим доступа: для авториз. пользователей.