|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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»

**Студент группы** ИКБО-01-19 Кузин Данил Олегович

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

(подпись студента)

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

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

(подпись руководителя)

Работа представлена «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

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

Москва 2021

**Оглавление**

[Цель работы 3](#_Toc85248709)

[Ход работы 3](#_Toc85248710)

[Вывод 10](#_Toc85248711)

[Ответы на вопросы к практической работе 11](#_Toc85248712)

[Список использованной литературы 17](#_Toc85248713)

# Цель работы

Используя серверную конфигурацию, разработанную в первой работе, научиться создавать сложную серверную конфигурацию, состоящую из связки apache+nginx+php+База данных.

# Ход работы

1. Был создана сложная серверная конфигурация, имеющая файловую систему представленную на Рисунке 1.

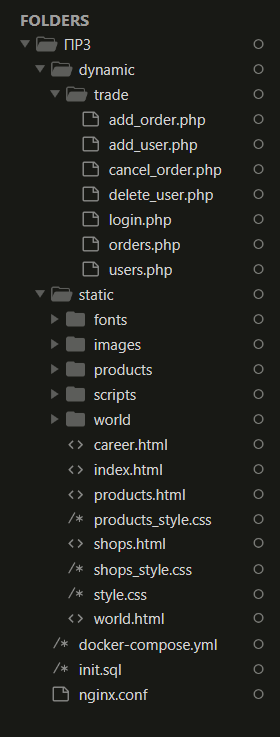


Рисунок - Файловая система проекта

1. Был создан файл nginx.conf для настройки nginx (Листинг 1).

Содержимое php-файлов и файла конфигурации nginx представлено на листингах 1-8.

Листинг 1 – Файл nginx.conf

server {

location / {

root /data/static/;

expires -1;

}

location ~ \.php$ {

proxy\_pass http://web;

proxy\_no\_cache 1;

proxy\_cache\_bypass 1;

}

}

Листинг 2 – Файл login.php

<?php

if (!isset($\_SERVER['PHP\_AUTH\_USER'])) {

header('WWW-Authenticate: Basic realm="My Realm"');

header('HTTP/1.0 401 Unauthorized');

echo 'Авторизация отменена';

exit;

}

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

$stmt = $mysqli->prepare("SELECT `password` FROM users WHERE `name`=?");

$res = $stmt->bind\_param('s', $\_SERVER['PHP\_AUTH\_USER']);

$stmt->execute();

$result = $stmt->get\_result();

$user = $result->fetch\_array(MYSQLI\_ASSOC);

if ($\_SERVER['PHP\_AUTH\_PW'] !== $user['password']) {

header('WWW-Authenticate: Basic realm="My Realm"');

header('HTTP/1.0 401 Unauthorized');

exit;

}

?>

Листинг 3 – Файл users.php

<?php

include 'login.php';

?>

<html lang="ru">

<head>

<title>Управление пользователями</title>

</head>

<body>

<h1>Управление пользователями</h1>

<h2>Заказчики</h2>

<form action="/trade/add\_user.php" method="post">

<p>Логин: <input type="text" name="name" /></p>

<p>Пароль: <input type="password" name="password" /></p>

<p><input type="submit" /></p>

</form>

<?php

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

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

echo "<ul>";

foreach ($result as $row) {

echo "<li>";

echo "<a href=\"/trade/delete\_user.php?ID={$row['ID']}\">";

echo "Удалить пользователя ↓";

echo "</a><br>";

echo "{$row['ID']} {$row['name']}</li>";

}

echo "</ul>";

?>

</body>

</html>

Листинг 4 – Файл orders.php

<?php

include 'login.php';

?>

<html lang="ru">

<head>

<title>Управление заказами</title>

</head>

<body>

<h1>Управление заказами</h1>

<h2>Заказы</h2>

<form action="/trade/add\_order.php" method="post">

<p>Название товара: <input type="text" name="title" /></p>

<p>Количество: <input type="number" name="count"></input></p>

<p>Адрес доставки: <input type="text" name="address" /></p>

<p><input type="submit"/></p>

</form>

<?php

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

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

echo "<ul>";

foreach ($result as $row) {

echo "<li>";

if($row['status'] != "Отменен") {

echo "<a href=\"/trade/cancel\_order.php?ID={$row['ID']}\">";

echo "Отменить заказ ↓";

echo "</a><br>";

}

echo "{$row['ID']} {$row['title']} {$row['count']}<div>{$row['address']}</div><div>{$row['status']}</li>";

}

echo "</ul>";

?>

</body>

</html>

Листинг 5 – Файл add\_user.php

<?php

include 'login.php';

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

$stmt = $mysqli->prepare("INSERT INTO users (name, password) VALUES (?, ?)");

$res = $stmt->bind\_param('ss', $\_POST['name'], $\_POST['password']);

$stmt->execute();

header('Location: /trade/users.php');

?>

Листинг 6 – Файл add\_order.php

<?php

include 'login.php';

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

$stmt = $mysqli->prepare("INSERT INTO orders (title, count, address, status) VALUES (?, ?, ?, 'В обработке')");

$res = $stmt->bind\_param('sss', $\_POST['title'], $\_POST['count'], $\_POST['address']);

$stmt->execute();

header('Location: /trade/orders.php');

?>

Листинг 7 – Файл delete\_user.php

<?php

include 'login.php';

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

$stmt = $mysqli->prepare("DELETE FROM `users` WHERE `ID`=?");

$res = $stmt->bind\_param('i', $\_GET['ID']);

$stmt->execute();

header('Location: /trade/users.php');

?>

Листинг 8 – Файл cancel\_order.php

<?php

include 'login.php';

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

$stmt = $mysqli->prepare("UPDATE orders SET status = 'Отменен' WHERE `ID`=?");

$res = $stmt->bind\_param('i', $\_GET['ID']);

$stmt->execute();

header('Location: /trade/orders.php');

?>

1. Программа была запущена и протестирована (Рисунок 2 - 6).

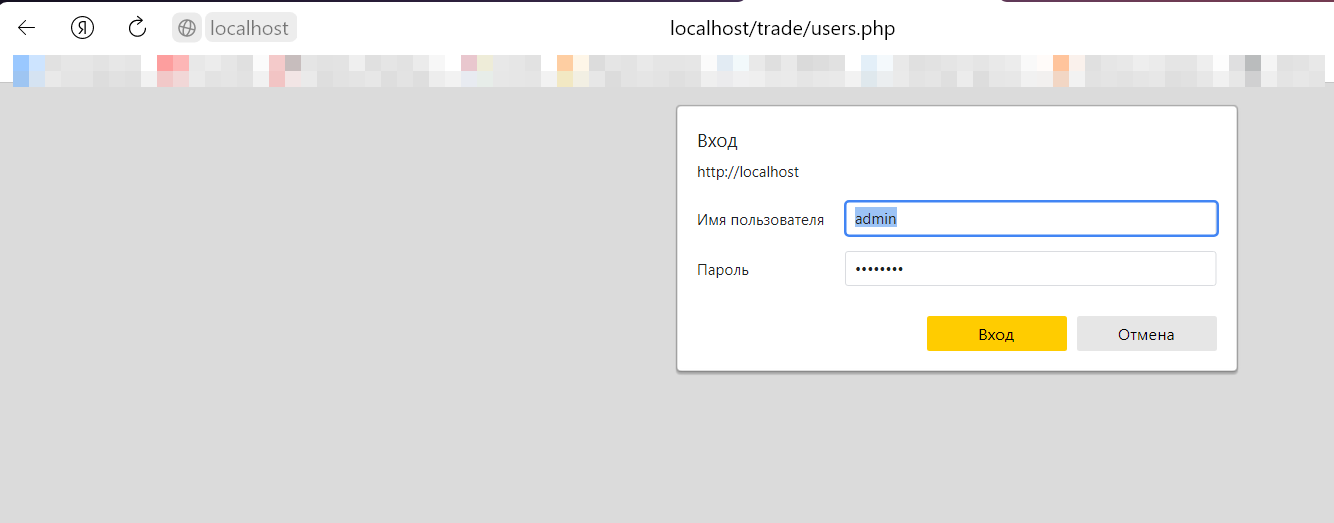


Рисунок – Авторизация

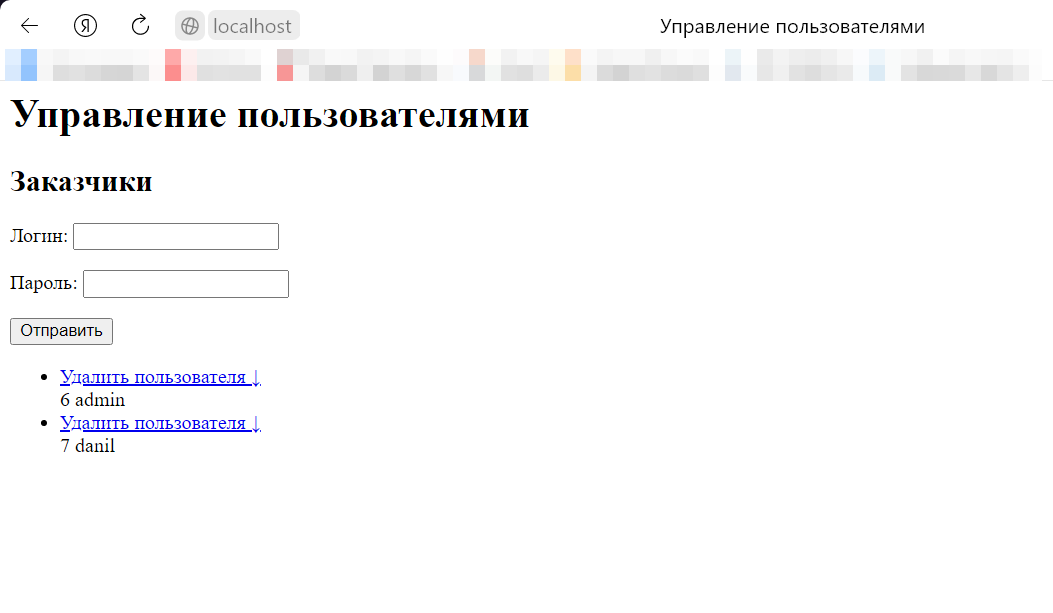


Рисунок – Управление пользователями

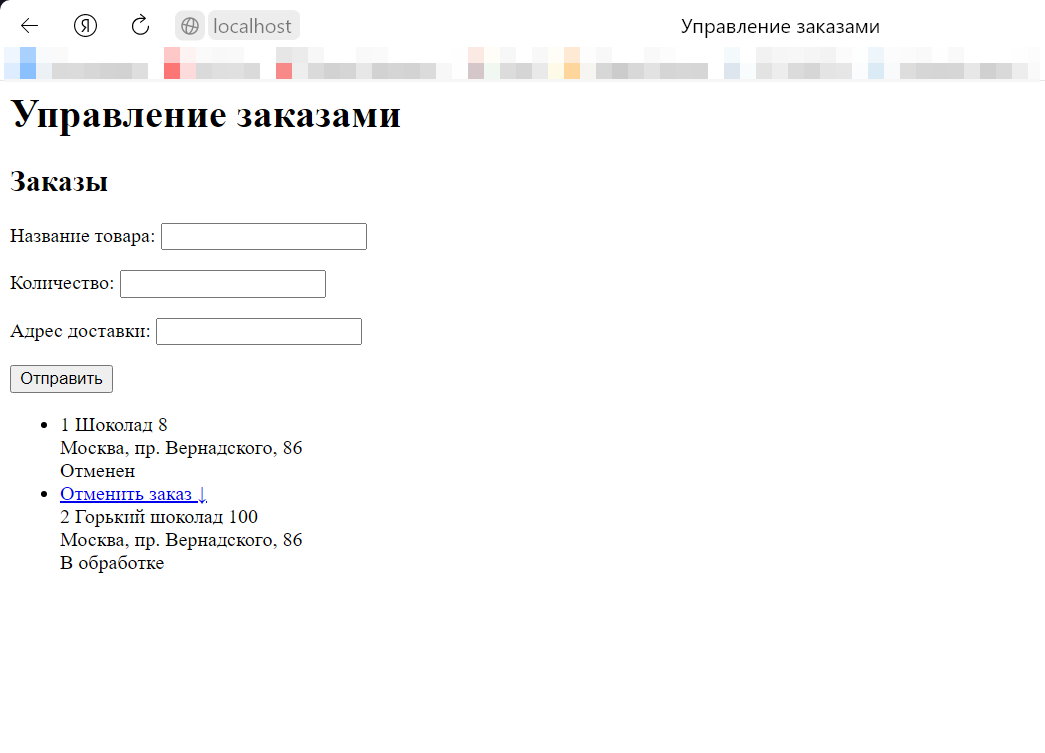


Рисунок 4 – Управление заказами

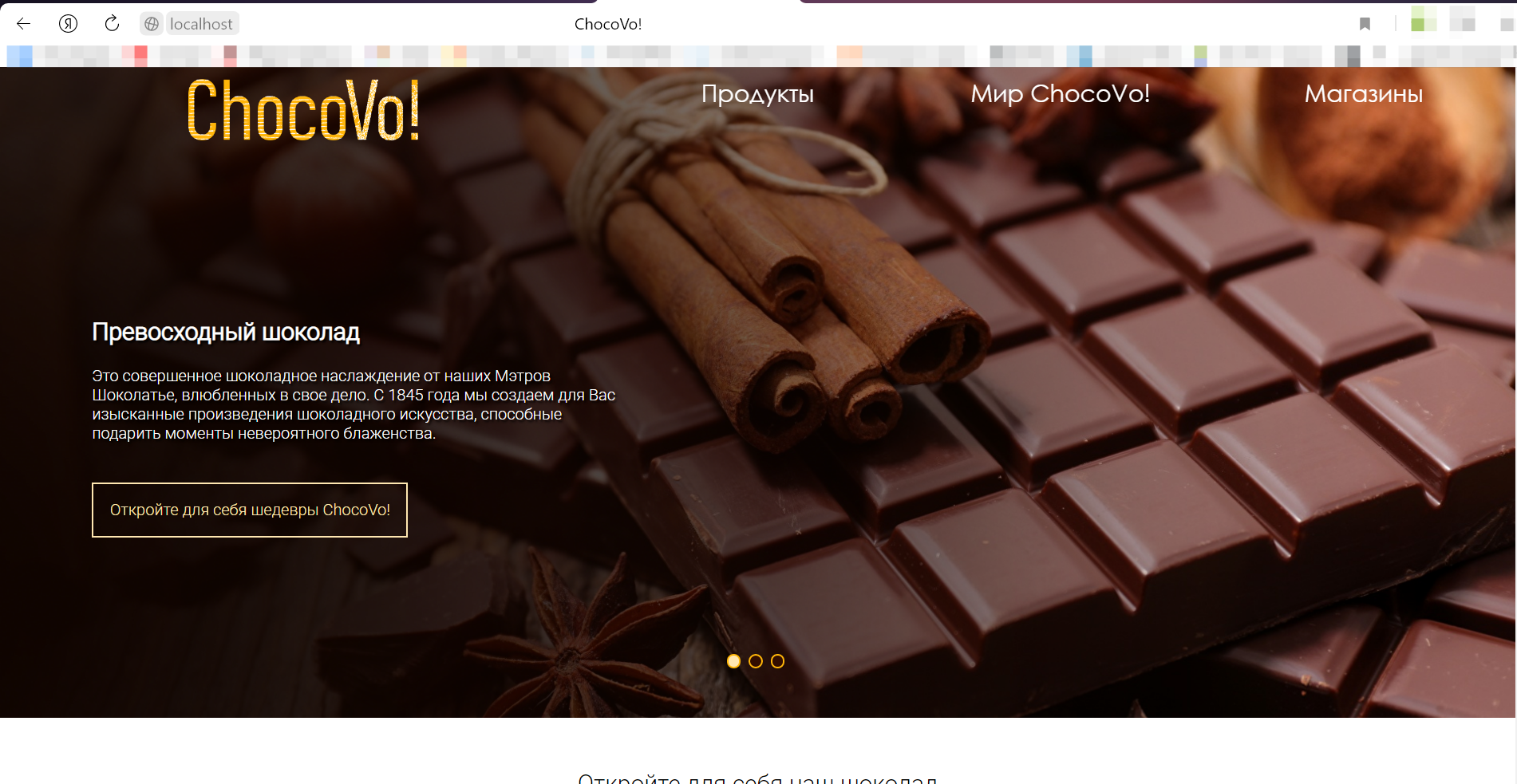


Рисунок – Веб-страница со статическим контентом

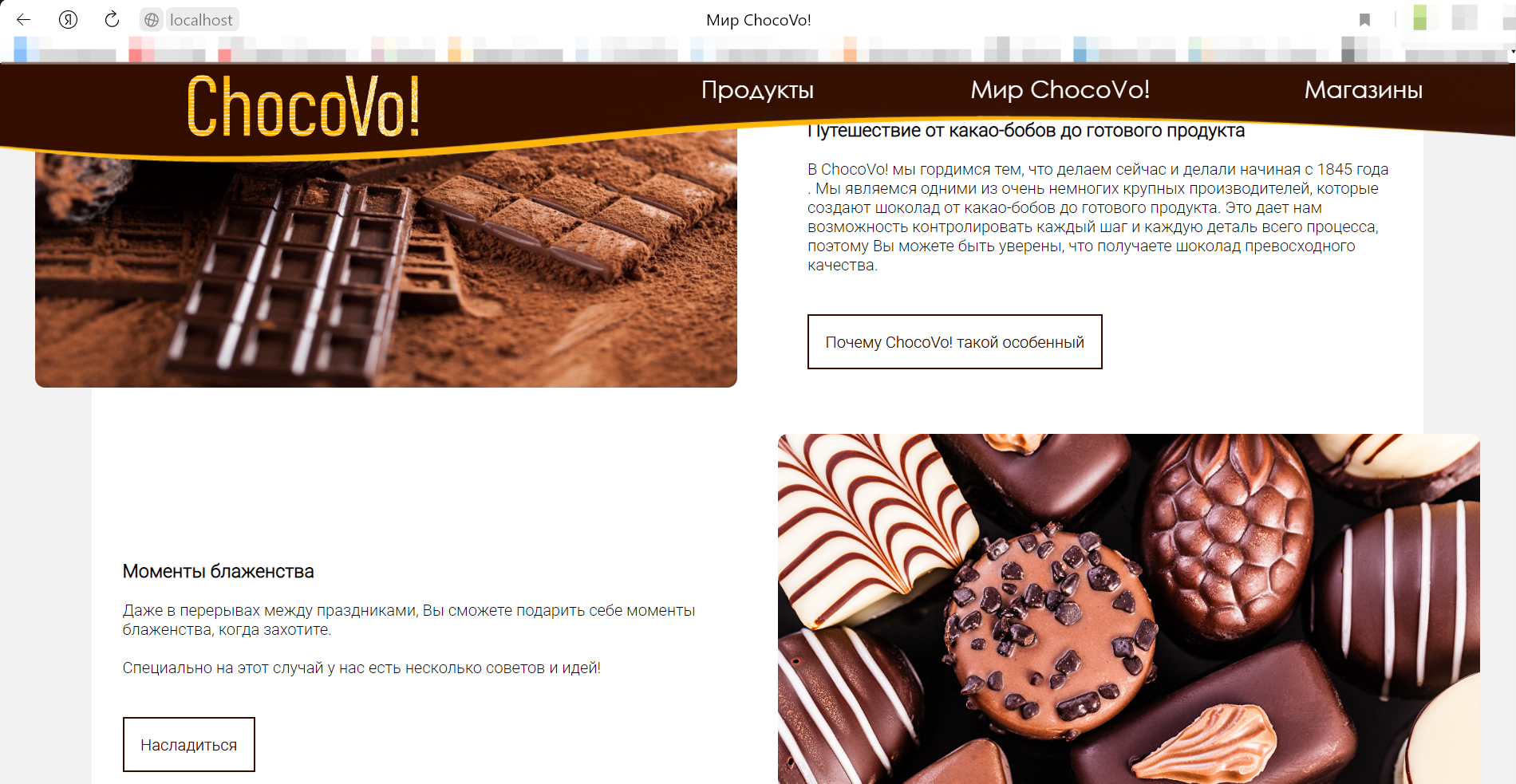


Рисунок 6 – Веб-страница со статическим контентом

# Вывод

В работе была продолжена настройка сложной серверной конфигурации, состоящей из связки apache+nginx+php+База данных. В результате работы было создано веб-приложение с динамическими веб-страницами и веб-страницами со статическим контентом.

# Ответы на вопросы к практической работе

1) Что такое веб-сервер?

Веб-сервер — это сервер, принимающий HTTP-запросы от клиентов, обычно веббраузеров, 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.

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

3) 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. Со временем стали появляться все новые и новые серверы. Сейчас наиболее популярные – nginx и apache.

4) 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.

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

Пользователь придумывает запрос, клиент осуществляет этот запрос к серверу, сервер его обрабатывает и отсылает ответ клиенту.

6) 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.

7) 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 HTMLдокумента.

8) 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.

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

Верно

10) Что такое CGI?

CGI — Common Gateway Interface — «интерфейс общего шлюза» — стандарт интерфейса, используемого внешней программой для связи с вебсервером.

11) Как работает система с использованием интерфейс шлюза - CGI?

С помощью CGI можно создавать CGI-программы, называемые шлюзами, которые во взаимодействии с такими прикладными системами, как система управления базой данных, электронная таблица, деловая графика и др., смогут выдать на экран пользователя динамическую информацию. Программа-шлюз запускается сервером в реальном масштабе времени. Сервер обеспечивает передачу запроса пользователя шлюзу, а она в свою очередь, используя средства прикладной системы, возвращает результат обработки запроса на экран пользователя.

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. По сравнению с CGI является более производительным и безопасным.

14) Назовите основные отличия CGI от FastCGI.

FastCGI не только наследует преимущества CGI, но также добавляет несколько новых функций.

Распределенные вычисления: мы можем запускать приложения FastCGI на машинах, отличных от веб-сервера. Распределенные вычисления - это проверенная технология, которая применяется для: настройки и подключения существующих совместных систем, полного использования производительности системы и повышения безопасности системы (например, брандмауэров)

Множество ролей и расширяемых ролей: приложения CGI обрабатывают HTTP-запросы и предоставляют возвращаемые значения.Функции FastCGI не ограничиваются этим, например, он выполняет модульную аутентификацию и проверки идентичности, а также конвертирует форматы данных. Эти особенности FastCGI позволяют играть больше ролей в будущем.

15) 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.

16) Что такое PHP-FPM?

PHP-FPM — программный пакет, позволяющий выполнить обработку скриптов, написанных на языке PHP. Включен в состав PHP с версии 5.3.3, для более ранних версий необходима установка отдельно. Является альтернативой FastCGI — протоколу взаимодействия веб-сервера с программами. FPM расшифровывается как Fastcgi Process Manager.

17) Что такое Spawn-fcgi?

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

18) Что такое Lighttpd?

Lighttpd — веб-сервер, разрабатываемый с расчётом на скорость и защищённость, а также соответствие стандартам. Это свободное программное обеспечение, распространяемое по лицензии BSD.

19) Что такое chroot окружение?

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

20) Опишите механизм взаимодействия серверов с использованием FastCGI.

Веб сервер через FastCGI отправляет запрос к другому серверу (веб серверу, серверу баз данных и др), и получает ответ.

21) Опишите процесс выбора встроенного или внешнего менеджера процессов.

22) Что такое интерфейс шлюза?

Интерфейс общего шлюза — стандарт интерфейса, используемого внешней программой для связи с вебсервером.

23) 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.

24) Что такое PCGI?

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

25) 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 Perlприложениям те же преимущества, что и FastCGI.

26) Что такое WSGI?

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

27) Опишите механизм взаимодействия серверов Apache и PHP.

Mod\_php - позволяет Apache выполнять php скрипты. Является наверно самым популярным и простым способом подружить Apache и PHP. Модуль не использует ни CGI, ни FastCGI. Есть свои минусы — скрипты работают под пользователем веб-сервера, невозможно использовать

больше одной версии PHP.

28) Опишите преимущества веб-сервера Apache.

Высокий уровень надежности, гибкие настройки, свободный доступ к программе, регулярные обновления и патчи, удобство и легкость настройки

29) 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

30) Опишите архитектуру веб-сервера Apache.

Apache имеет ядро, систему конфигураций, систему моделей, многопроцессорные модели и механизм виртуальных хостов.

31) Опишите функции ядра веб-сервера Apache.

Передача данных по HTTP, обработка файлов, загрузка и поддержка модулей

32) Опишите конфигурацию веб-сервера Apache.

Конфигурацию Apache, можно разделить на три основных уровня: конфигурация сервера, конфигурация виртуального хоста, конфигурация уровней каталога

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

URI — символьная строка, позволяющая идентифицировать какой-либо ресурс: документ, изображение, файл, службу, ящик электронной почты и т. д.

URL — это URI, который, помимо идентификации ресурса, предоставляет ещё и информацию о местонахождении этого ресурса.

# Список использованной литературы

1. Документация Docker — Текст : электронный // Docker docs : [Электронный ресурс]. — URL: https://docs.docker.com/ (дата обращения: 04.09.2021).
2. Руководство по php — Текст : электронный // php : [Электронный ресурс]. — URL: https://www.php.net/ (дата обращения: 22.09.2021).
3. Docker Hub — Текст : электронный // Docker Hub : [Электронный ресурс]. — URL: https://hub.docker.com/ (дата обращения: 04.09.2021).
4. Справочник по php — Текст : электронный // Docker Hub : [Электронный ресурс]. — URL: https://php.ru/manual/ (дата обращения: 22.09.2021).