|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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 №1**

по дисциплине «Разработка серверных частей интернет-ресурсов»

**Тема практической работы:**

**Студент группы** ИКБО-16-20 Лобанков Андрей Александрович

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

(подпись студента)

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

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

(подпись руководителя)

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

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

Москва 2022

# **Цель работы:**

Создать свою конфигурацию серверного программного обеспечения, в которой должны присутствовать веб-сервер, операционная система, язык программирования и база данных.

# **Постановка задачи:**

Для проверки работоспособности конфигурации требуется инициализировать базу данных: создать отдельного пользователя для работы с ней, создать базу данных, в которой создать таблицу пользователи с полями: идентификационный номер, имя, фамилия. Также для проверки конфигурации требуется сгенерировать тестовую страничку, содержащую выборку из созданной таблицы и информационное сообщение о версии языка программирования, его настройках и конфигурации.

# **Ход работы:**

Мною было принято решение использовать технологии LAMP — Linux OS, Apache, MySQL, PHP. Для этого необходимо было создать файл docker-compose.yml , в котором будет прописаны инструкции для запуска и настройки необходимых нам сервисов. Структура проекта будет выглядеть следующим образом: директория database, в котором будет хранится скрипт для создания и инициализирования базы данных, который был предоставлен в качестве проверки работоспособности конфигурации, директория server, в котором хранится index.php и style.css (Рисунок 1)

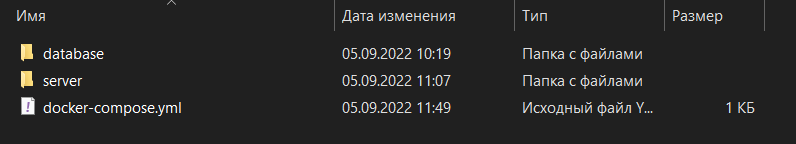


Рисунок 1 – Структура проекта.

Также в директории server присутствует Dockerfile, необходимый для закачки образа php apache и установки нужных расширений (Рисунок 2).

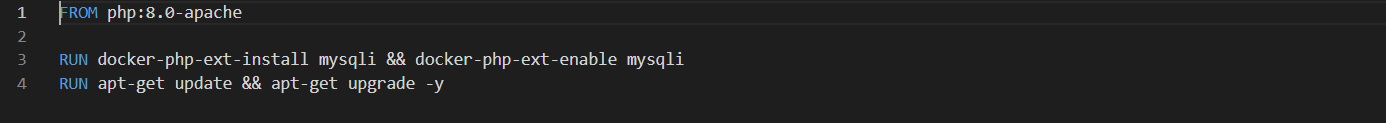


Рисунок 2 – Содержимое Dockerfile.

В файле docker-compose.yml прописаны инструкции для запуска необходимых нам технологий (Рисунок 3).

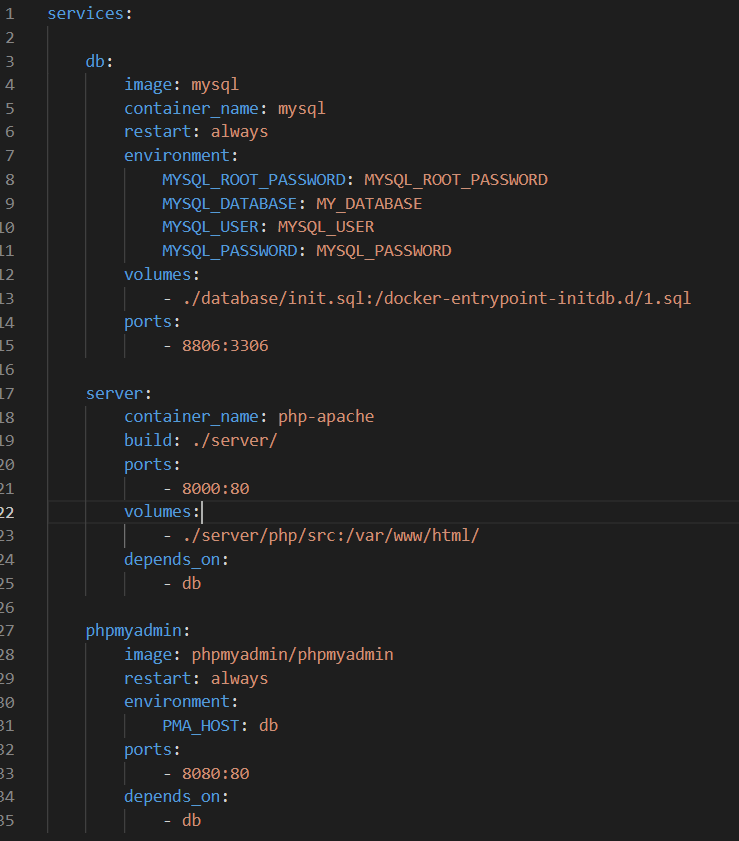


Рисунок 3 – Инструкции для базы данных.

Через консоль осуществим запуск с помощью команды docker-compose up (Рисунок 4).

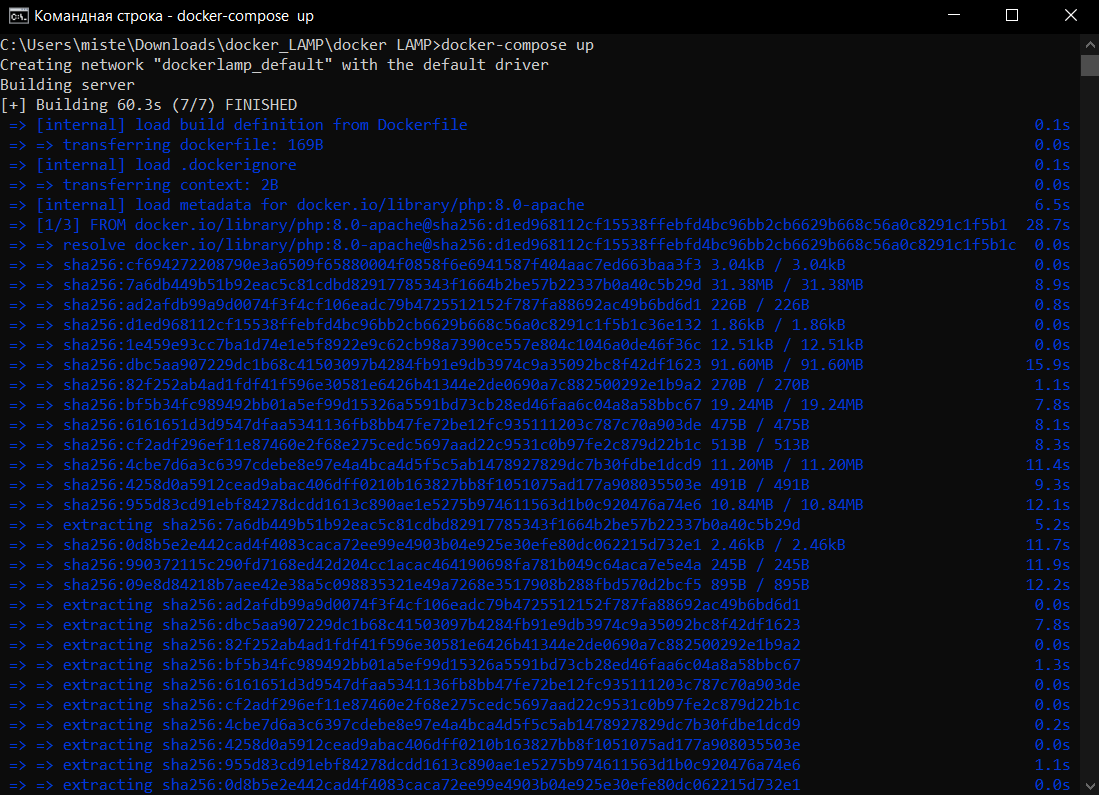


Рисунок 4 – Запуск docker-compose.

Переходим по localhost:8000 и видим наш готовый результат (Рисунок 5).

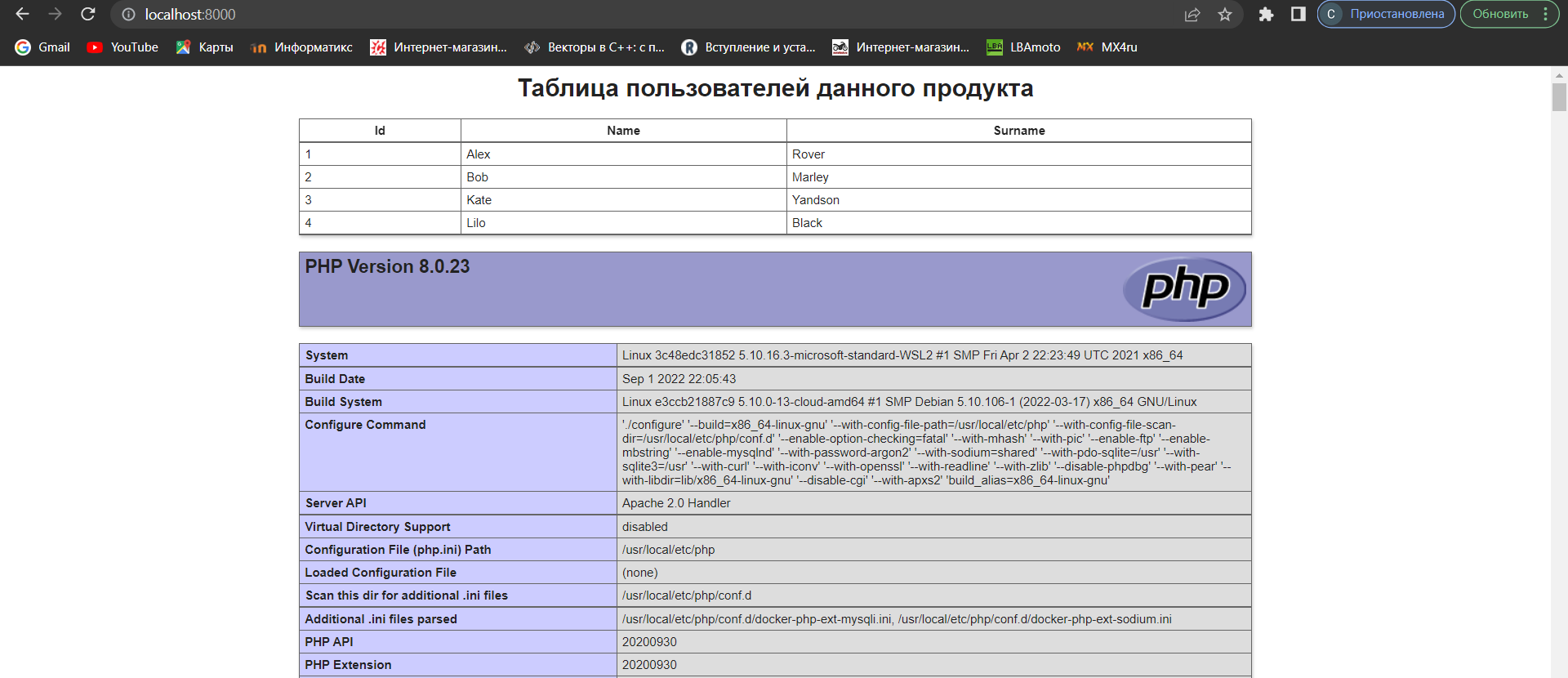


Рисунок 5 – Результат.

# **Выводы:**

В ходе выполнения практической работы была успешна установлена конфигурация LAMP и выполнены поставленные задачи.

# **Ответы на вопросы:**

1. Сервер - (программное обеспечение) - программный компонент вычислительной системы, выполняющий сервисные (обслуживающие) функции по запросу клиента, предоставляя ему доступ к определённым ресурсам или услугам. Клиент — это аппаратный или программный компонент вычислительной системы, посылающий запросы серверу.
2. База данных — это информационная модель, позволяющая упорядоченно хранить данные об объекте или группе объектов, обладающих набором свойств, которые можно категорировать. Базы данных функционируют под управлением систем управления базами данных (сокращенно СУБД).
3. API (Application Programming Interface - прикладной программный интерфейс) - набор функций и подпрограмм, обеспечивающий взаимодействие клиентов и серверов.
4. Сервис - легко заменяемый компонент сервисно-ориентированной архитектуры со стандартизированными интерфейсами. В отличие от сервиса, сервер это виртуальная машина, которая может хостить множество сервисов.
5. Архитектура клиент-сервер - данная модель — это идея разделения системы или приложения на отдельные задачи, размещаемые на различных платформах для большей эффективности. Уже применение данной идеи лежит в основе архитектуры клиент-сервер, распределенных вычислений, архитектуры приложений и т.д.

Если рассматривать в общем случае как модель, архитектура клиент-сервер — это сетевое окружение, в котором управление данными осуществляется на серверном узле, а другим узлам (клиентам) предоставляется доступ к данным.

1. Виды сервисов: серверы приложений, веб-серверы, серверы баз данных, файл серверы, прокси-серверы, файрволы, почтовые серверы.
2. Масштабируемость - способность работать с увеличенной нагрузкой путем наращивания ресурсов без фундаментальной перестройки архитектуры и/или модели реализации при добавлении ресурсов. Система называется масштабируемой, если она способна увеличивать производительность пропорционально дополнительным ресурсам.
3. Протоколы передачи данных - набор определенных правил или соглашений интерфейса логического уровня, который определяет обмен данными между различными программами. Эти правила задают единообразный способ передачи сообщений и обработки ошибок.
4. Тонкий и толстый клиенты. Главное отличие тонкого клиента от толстого заключается в том, что выполнение самого приложения происходит на сервере. Толстый клиент совмещает компонент представления данных и прикладной компонент.
5. Паттерн MVC: общие тезисы - схема разделения данных приложения и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.

***Модель*** (*Model*) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.

***Представление*** (*View*) отвечает за отображение данных модели пользователю, реагируя на изменения модели.

***Контроллер*** (*Controller*) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

1. Паттерн MVP: Model-View-Presenter - Особенностью паттерна Model-View-Presenter является то, что он позволяет создавать абстракцию представления. Для реализации данного метода выделяется интерфейс представления. А презентер получает ссылку на реализацию интерфейса, подписывается на события представления и по запросу меняет модель.

Признаки подхода с использованием презентера:

● двусторонняя коммуникация с представлением;

●представление взаимодействует напрямую с презентером, путем вызова соответствующих функций или событий экземпляра презентера;

● презентер взаимодействует с View путем использования специального интерфейса, реализованного представлением;

● одному презентеру соотвествует одно отображение.

1. Паттерн MVVM: Model-View-View Model - Особенностью паттерна Model-View-View Model является связывание элементов представления со свойствами и событиями View-модели.

Признаками данного подхода являются:

● Двусторонняя коммуникация с представлением.

● View-модель — это абстракция представления. Означает, что свойства представления совпадают со свойствами View-модели / модели.

● View-модель не имеет ссылки на интерфейс представления (IView). Изменение состояния View-модели автоматически изменяет представление и наоборот, поскольку используется механизм связывания данных (Bindings).

● Одному экземпляру View-модели соответствует одно отображение.

1. Паттерн MVC: Model-View-Controller - Особенностью паттерна Model-View-Controller является то, что контроллер и представление зависят от модели, но при этом сама модель не зависит от двух других компонентов.

Признаками данного подхода являются:

● Контроллер определяет, какое представление должно быть отображено в требуемый момент. Если рассматривать применение для разработки веб-приложений, то контроллер управляет запросами пользователя. Его основная функция — вызывать и координировать действие необходимых ресурсов и объектов, нужных для выполнения действий, задаваемых пользователем. Обычно контроллер вызывает соответствующую модель для задачи и выбирает подходящий вид.

● События представления могут повлиять только на контроллер. Контроллер может повлиять на модель и определить другое представление.

● Возможно несколько представлений только для одного контроллера.

1. Docker: общие тезисы и определения - программное обеспечение для автоматизации развёртывания и управления приложениями в среде виртуализации на уровне операционной системы; позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер, а также предоставляет среду по управлению контейнерами. Ключевое преимущество Докера в том, что он позволяет пользователям упаковать приложение со всеми его зависимостями в стандартизированный модуль для разработки. В отличие от виртуальных машин, контейнеры не создают такой дополнительной нагрузки, поэтому с ними можно использовать систему и ресурсы более эффективно.
2. Dockerfile - это текстовый файл с инструкциями, необходимыми для создания образа контейнера. Эти инструкции включают идентификацию существующего образа, используемого в качестве основы, команды, выполняемые в процессе создания образа, и команду, которая будет выполняться при развертывании новых экземпляров этого образа контейнера.
3. Docker Compose - это инструментальное средство, входящее в состав Docker. Оно предназначено для решения задач, связанных с развёртыванием проектов.
4. LAMP - набор серверного программного обеспечения, широко используемый в интернете. LAMP назван по первым буквам входящих в его состав компонентов:

● Linux — операционная система Linux;

● Apache — веб-сервер;

● MariaDB / MySQL — СУБД;

● PHP — язык программирования, используемый для создания вебприложений (помимо PHP могут подразумеваться другие языки, такие как Perl и Python).

**Ссылка на удаленный репозиторий проекта:** https://github.com/exploitMasterAndrey/PHP

**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. Дергачев А. М. Проблемы эффективного использования сетевых сервисов / Научно-технический вестник СПбГУ ИТМО. 2011. № 1 (71). С. 83-87.

2. Розенфельд Л., Морвиль П. Информационная архитектура в Интернете, 2 е издание. – Пер. с англ. – СПб: Символ Плюс, 2005 – 544 с., ил.

3. Спинеллис Д., Гусиос Г. Идеальная архитектура. Ведущие специалисты о красоте программных архитектур. – Пер. с англ. – СПб.: Символ Плюс, 2010 – 528 с., ил.

4. Фаулер, Мартин. Ф28 Архитектура корпоративных программных приложений.: Пер. с англ. — М.: Издательский дом "Вильяме", 2006 — 544 с.: ил. — Парал. тит. англ.

5. Голицына О.Л., Максимов Н.В., Партыка Т.Л., Попов И.И. ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ. - 2-е изд. - Москва: ФОРУМ - ИНФРА-М, 2008. - 395 с.

6. Распределенные системы. Принципы и парадигмы / Э. Таненбаум, М. ван Стеен. — СПб.: Питер, 2003. — с. 83-93 — (Серия «Классика computer science»). ISBN 5-272-00053-6-.