|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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 Голубев Григорий Александрович

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

(подпись студента)

**Руководитель практической работы** преподаватель

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

(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, 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)

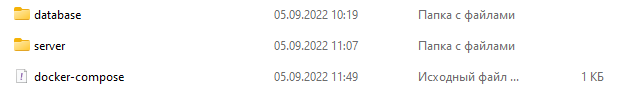


Рисунок 1 – Структура проекта.

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

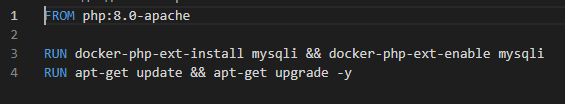


Рисунок 2 – Содержимое Dockerfile.

В файле docker-compose.yml прописаны инструкции для запуска необходимых нам технологий. На рисунке 3 прописаны инструкции для MYSQL базы данных.

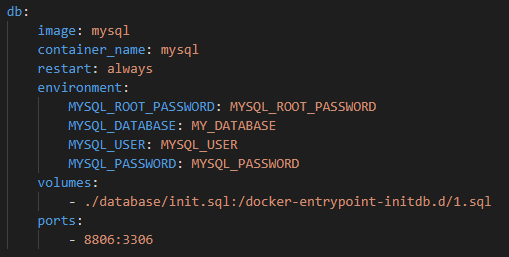


Рисунок 3 – Инструкции для базы данных.

На рисунке 4 прописаны инструкции для php-apache.

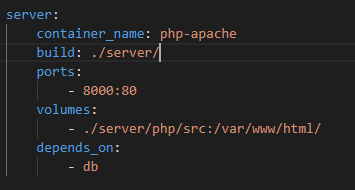


Рисунок 4 – Инструкции для базы данных.

На рисунке 5 прописаны инструкции для phpMyAdmin (Рисунок 5).

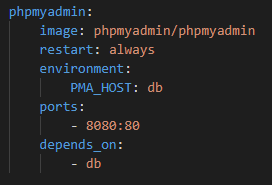


Рисунок 5 – Инструкции для phpMyAdmin.

Через консоль осуществим запуск с помощью команды docker-compose up (Рисунок 6).

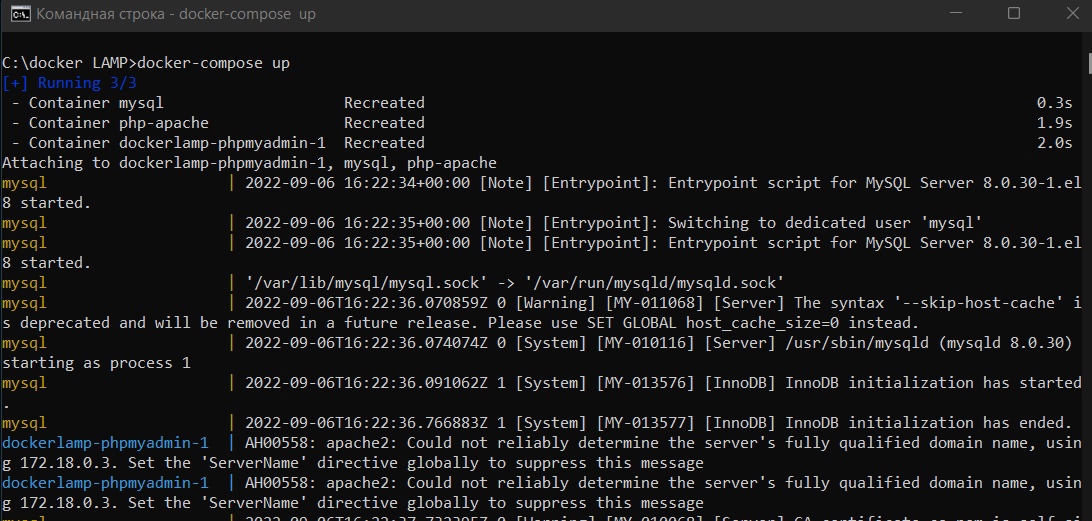


Рисунок 6 – Запуск docker-compose.

Переходим по localhost:8080 и видим наш готовый результат (Рисунок 7).

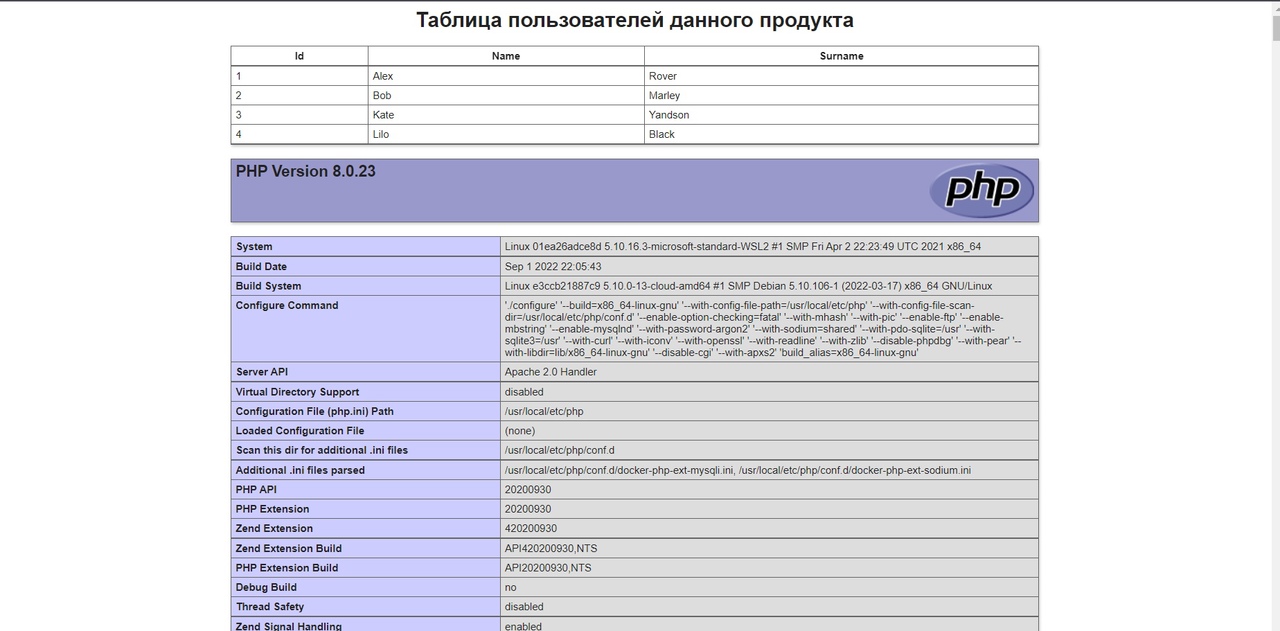


Рисунок 7 – Результат.

# **Выводы:**

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

# **Ссылка на репозиторий:**

https://github.com/Grigalgol/PHP-works

# **Ответы на вопросы:**

1. **Сервер и клиент**

Технология клиент-сервер - это сетевая архитектура, в которой процессы обмена данными или файлами распределена между так называемыми поставщиками и заказчиками.

В технологии клиент-сервер есть два главных действующих лица:

* клиент - компьютерное или мобильное устройство при управлении пользователем, которое отправляет запрос или команду серверу (например, ввод поискового запроса в Google тоже относится к этому процессу);
* сервер - аппаратный или облачный сервер, который принимает запрос и выполняет его (обработка данных на сайтах, приложениях и в сервисах происходит через веб-узлы).

1. **База данных**

База данных (БД) — это имеющая название совокупность данных, которая отражает состояние объектов и их отношений в рассматриваемой предметной области.

1. **API**

API – это механизмы, которые позволяют двум программным компонентам взаимодействовать друг с другом, используя набор определений и протоколов.

1. **Сервис, отличия от сервера**

Сервер находится уровнем выше, чем сервис. Любой полученный запрос прежде, чем попадёт в сервис, проходит через сервер. И уже сервер передаёт запрос сервису. И сервис отдаёт ответ именно серверу. А уже сервер отправляет ответ клиенту.

1. **Архитектура клиент-сервер**

В архитектуре «Клиент-Сервер» несколько компьютеров-клиентов (удалённые системы) посылают запросы и получают услуги от централизованной служебной машины – сервера

1. **Виды сервисов**

Сервер приложений — это программная платформа (фреймворк), предназначенная для эффективного исполнения процедур (программ, скриптов), на которых построены приложения.

Веб-серверы. Являются подвидом серверов приложений. Изначально предоставляли доступ к гипертекстовым документам по протоколу HTTP. Сейчас поддерживают расширенные возможности, в частности, передачу произвольных данных.

Серверы баз данных. Серверы баз данных используются для обработки запросов. На сервере находится СУБД для управления БД и ответов на запросы.

Файл-серверы. Файл-сервер хранит информацию в виде файлов и предоставляет пользователям доступ к ней.

Прокси-сервер - промежуточный сервер (комплекс программ) в компьютерных сетях, выполняющий роль посредника (без шифрования)

Файрволы (брандмауэры). Межсетевые экраны, анализирующие и фильтрующие проходящий сетевой трафик, с целью обеспечения безопасности сети.

Почтовые серверы. Предоставляют услуги по отправке и получению электронных почтовых сообщений.

1. **Масштабируемость**

Система называется масштабируемой, если она способна увеличивать производительность пропорционально дополнительным ресурсам.

1. **Протоколы передачи данных**

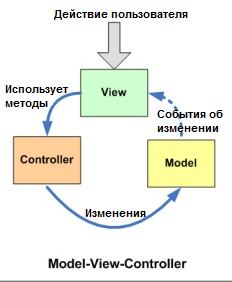
Сетевые протоколы передачи данных — перечень правил, определяющих особенности и порядок передачи информации.

1. **Тонкий и толстый клиенты**

Толстый клиент — клиент, выполняющий запрашиваемые со стороны пользователя манипуляции независимо от ведущего сервера. Основной сервер в такой вариации системной архитектуры может применяться как особое хранилище информации, обработка и конечное предоставление которых просто переносится на локальную машину пользователя.

Тонкий клиент — вид клиента, который может переносить выполнение задач по обработке информации на сервер, не применяя свои мощности по вычислению для их внедрения. Все вычислительные ресурсы подобного клиента максимально ограничены, важно, чтобы их хватало для старта нужного сетевого ПО, применяя, к примеру, веб-интерфейс.

1. **Паттерн MVC: общие тезисы**



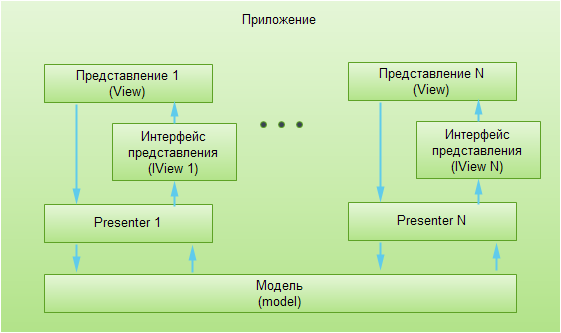
MVC расшифровывается как модель-представление-контроллер (от англ. model-view-controller). Это способ организации кода, который предполагает выделение блоков, отвечающих за решение разных задач. Один блок отвечает за данные приложения, другой отвечает за внешний вид, а третий контролирует работу приложения. Компоненты MVC:

Модель — этот компонент отвечает за данные, а также определяет структуру приложения. Например, если вы создаете To-Do приложение, код компонента model будет определять список задач и отдельные задачи.

Представление — этот компонент отвечает за взаимодействие с пользователем. То есть код компонента view определяет внешний вид приложения и способы его использования.

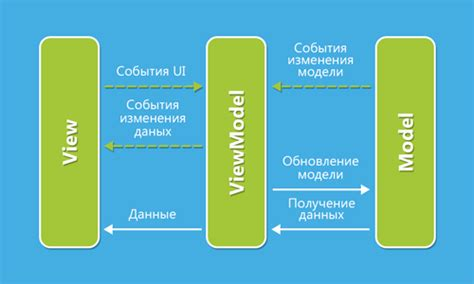
Контроллер — этот компонент отвечает за связь между model и view. Код компонента controller определяет, как сайт реагирует на действия пользователя. По сути, это мозг MVC-приложения.

1. **Паттерн MVC: Model-View-Presenter**



**Model View Presenter (MVP)** - шаблон, который впервые появился в IBM, а затем использовался в Taligent в 1990-х. MVP является производным от MVC, при этом имеет несколько иной подход. В MVP представление не тесно связано с моделью, как это было в MVC. Presenter общается с представлением через интерфейс, который позволяет увеличить тестируемость, так как модель может быть заменена на специальный макет для модульных тестов.

1. **Паттерн MVC: Model-View-View Model**



MVVM (Model-View-ViewModel) — способ организации кода. Он помогает отделить пользовательский интерфейс от логики.

View содержит только код пользовательского интерфейса. Он отображает экран (текстовые поля, ярлыки, кнопки) и пользовательский ввод.

Model — это классы. Это «вещи» в вашей программе. Например, в программе ввода заказов вашими классами моделей могут быть: Customer, Order, InventoryItem. Эти классы будут содержать логику, необходимую для выполнения функций.

ViewModel используются для связи между View и Model. Они обычно не содержат много логики. Они содержат Model, который должен быть использован во View. Когда человек взаимодействует с пользовательским интерфейсом, ViewModel узнает, какое действие необходимо выполнить, и просит Model сделать это.

1. **Паттерн MVC: Model-View-Controller**

MVC — подход к проектированию приложения, который предполагает выделение кода в блоки модель, представление и контроллер. Контроллер обрабатывает входящие запросы. Модель достаёт из базы данных информацию, нужную для выполнения конкретных запросов. Представление определяет результат запроса, который получает пользователь.

1. **Docker: общие тезисы и определения**

Docker — это программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации.

Docker состоит из нескольких компонентов:

Docker Daemon — то самое Container Engine; запускает контейнеры.

Docker CII — утилита по управлению Docker.

Dockerfile — инструкция по тому, как собирать образ.

Image — образ, из которого раскатывается контейнер.

Container - это способ стандартизации развертки приложения и отделения его от общей инфраструктуры. Экземпляр приложения запускается в изолированной среде, не влияющей на основную операционную систему.

Docker registry — хранилище образов.

Docker daemon (демон) — это серверная часть, она работает на хост-машине: скачивает образы и запускает из них контейнеры, создаёт сеть между контейнерами, собирает логи. Когда мы говорим «создай образ», этим тоже занимается демон.

Docker CLI — клиентская часть Docker, консольная утилита для работы с демоном. Повторю, она может работать не только локально, но и по сети.

1. **Dockerfile**

Dockerfile — это текстовый сценарий инструкций, который используется для создания образа контейнера.

1. **Docker Compose**

Docker Compose — это средство для определения и запуска приложений Docker с несколькими контейнерами.

1. **LAMP**

LAMP – это комплекс серверного программного обеспечения, состоящий из операционной системы Linux, веб-сервера Apache, СУБД MySQL, языка программирования PHP (помимо него также могут использоваться Perl, Python).