МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий (ИТ)

Кафедра Инструментального и Прикладного Программного Обеспечения (ИиППО)

ПРАКТИЧЕСКАЯ РАБОТА №1

по дисциплине

«Разработка серверных частей интернет ресурсов»

Тема: **«Установка PHP»**

Студент группы ИКБО-13-20 Лянной А.П.

_____

(подпись студента)

Принял руководитель работы ???????????.

_____

(подпись руководителя)

Практическая работа выполнена «___»_____2022 г.

Зачтено «___»_____2022 г.

Москва 2022 г.

ОГЛАВЛЕНИЕ

1. Цель работы

Создать свою конфигурацию серверного программного обеспечения, в которой должны присутствовать веб-сервер, операционная система, язык программирования и база данных. Данная конфигурация будет использоваться для выполнения следующих практических работ по данной дисциплине и для выполнения курсового проектирования.

Дается рекомендация использовать ОС Linux, язык программирования PHP, веб-сервер Apache и СУБД MySQL.

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.

Также для выполнения задания рекомендуется использовать технологию контейнеризации и оркестровки контейнеров.

2. Ход работы

От оркестровки контейнеров требуется во-первых объединение выше рекомендованных технологий, а во-вторых их применимость, для чего требуется выполнить некоторые настроечные команды при запуске контейнеров и предоставить им доступ к некоторым томам. Оркестровку контейнеров составим по следующему файлу *docker-compose.yml* (листинг 1).

Листинг 1 – docker-compose.yml

```
version: '3.7'
services:
    db:
        image: mysql:latest
        command: --default-authentication-
plugin=mysql_native_password
        environment:
            MYSQL_ROOT_PASSWORD: Franklin5
        volumes:
        - "./test/:/var/www/html"
    www:
        depends_on:
        - db
        image: php:7.4.18-apache
        volumes:
        - "./test/:/var/www/html"
        ports:
        - 80:80
        - 443:443
```

Установленные и оркестрованные образы все ещё необходимо донастроить. В контейнер с PHP-Apache требуется установить модуль *mysqli* для доступа к MySQL, что делается командой `docker-php-ext-install mysqli`. Файл *index.php* будет запущен в качестве точки входа на сайт автоматически. В контейнере с MySQL требуется зайти в MySQL и выполнить инициализационный скрипт *init.sql* с помощью `source`. Также для подчёркивания индивидуальности работы добавим дополнительную запись в таблицу с именем-фамилией исполнителя работы (рис. 1).

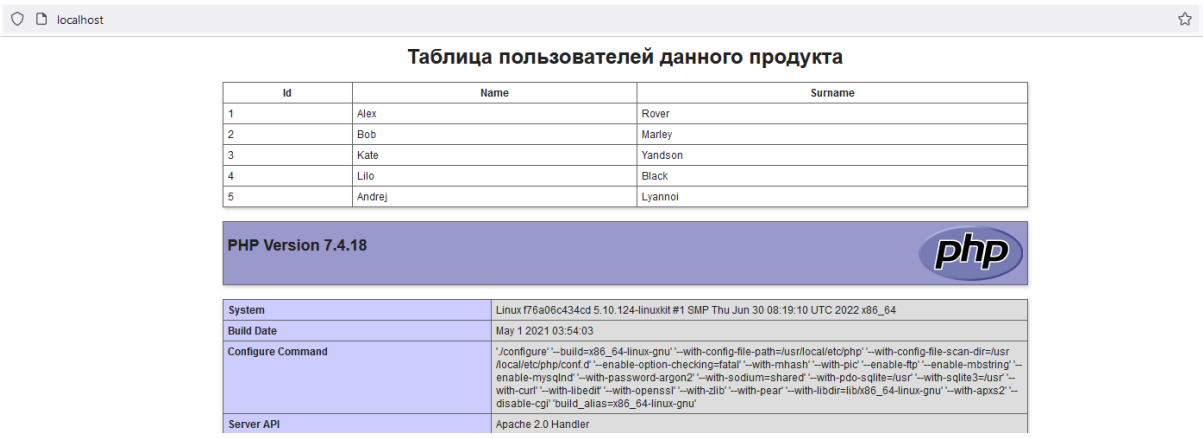

Рисунок 1 – Результат выполнения работы

3. Вывод

Была создана своя конфигурация серверного программного обеспечения, в которой присутствуют веб-сервер, операционная система, язык программирования и база данных.

4. Ответы на вопросы к практической работе

1. Сервер и клиент

Сервер — 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

База данных — организованная в соответствии с определёнными правилами и поддерживаемая в памяти компьютера совокупность данных, характеризующая актуальное состояние некоторой предметной области и используемая для удовлетворения информационных потребностей пользователей.

3. API

API (Application Programming Interface) – описание способов взаимодействия одной компьютерной программы с другими.

4. Сервис, отличия от сервера

Сервер - программный компонент вычислительной системы, выполняющий сервисные (обслуживающие) функции по запросу клиента, предоставляя ему доступ к определённым ресурсам или услугам.

Сервис - легко заменяемый компонент сервисно-ориентированной архитектуры со стандартизированными интерфейсами.

Эффективный сервер состоит из нескольких сервисов, связанных друг с другом.

5. Архитектура клиент-сервер

Архитектура клиент-сервер – вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами.

6. Виды сервисов

- Серверы приложений
- Веб-серверы
- Серверы баз данных
- Файл-серверы
- Прокси-сервер
- Файрволы
- Почтовые серверы

7. Масштабируемость

Масштабируемость – способность работать с увеличенной нагрузкой путем наращивания ресурсов без фундаментальной перестройки архитектуры и/или модели реализации при добавлении ресурсов.

8. Протоколы передачи данных

- RTP (Real-time Transport Protocol)
- HTTP (Hyper-Text Transfer Protocol)
- FTP (File Transport Protocol)
- POP3 (Post Office Protocol)
- SMTP (Simple Mail Transfer Protocol)
- TELNET

9. Тонкий и толстый клиенты

Тонкий клиент несёт ответственность за отображение данных, сервер – за хранение и обработку.

Толстый клиент несёт ответственность за обработку и отображение данных, сервер – за хранение.

10. Паттерн MVC: общие тезисы.

MVC – Model View Controller.

Первая часть данного паттерна это модель. Это представление содержания функциональной бизнес-логики приложения. Модель, как и любой компонент архитектуры под управлением данного паттерна не зависит от остальных частей продукта. То есть слой, содержащий модель ничего не знает об элементах дизайна и любом другом отображении пользовательского интерфейса.

Представление это есть отображение данных, получаемых от модели. Никакого влияния на модель представление оказать не может. Данное разграничение является разделением компетенций компонентов приложения и влияет на безопасность данных.

Третьим компонентом системы является контроллер. Данный компонент является неким буфером между моделью и представлением. Обобщенно он управляет представлением на основе изменения модели.

11. Паттерн MVC: 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.

12. Паттерн MVC: Model-View-ViewModel.

Особенностью паттерна Model-View-ViewModel является связывание элементов представления со свойствами и событиями View-модели.

13. 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.

14. Docker: общие тезисы и определения.

Существует проблема разработки того или иного приложения и его развертывания на других машинах. Самыми частыми решениями данной проблемы является установочные скрипты, облачные сервисы и виртуальные машины. Описанные подходы не являются оптимальными что раздувает техническую поддержку до максимума, а также медленны и тяжеловесны. Одним из вариантов решения данной задачи является докер, который представляет технологию контейнеризации. Подобно виртуальной машине докер запускает свои процессы в собственной, заранее настроенной операционной системе. Но при этом все процессы докера работают на физическом host-сервере, деля все процессоры и всю доступную память со всеми другими процессами, запущенными в host-системе. Подход, используемый Docker, находится посередине между запуском всего на физическом сервере и полной виртуализацией, предлагаемой виртуальными машинами. Этот подход называется контейнеризацией.

15. Dockerfile.

Часто возникает ситуация, когда конфигурации уже существующего не хватает. Чтобы создавать свои собственные образы нужен специальный скрипт. Образы наследуются и, обычно, для создания своего первого образа мы берём готовый образ и наследуемся от него. Чтобы запустить данный скрипт он должен иметь имя Dockerfile и не должен иметь типа.

16. Docker Compose.

Когда идет работа с несколькими контейнерами, то требуется механизм их объединения и оркестровки. Таким инструментом является Docker Compose. Это средство для решения задач развертывания проектов. Docker Compose используется для одновременного управления несколькими контейнерами, входящими в состав приложения. Этот инструмент предлагает те же возможности, что и Docker, но позволяет работать с более сложными приложениями.

17. LAMP

Для полноценной работоспособности конфигурации нужны: операционная система, Веб-сервер, язык программирования и База данных. Из всего этого следует идея технологии LAMP — акроним, обозначающий набор (комплекс) 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 веб-приложений (помимо PHP могут подразумеваться другие языки, такие как Perl и Python).

5. Ссылка на удалённый репозиторий проекта

https://github.com/Kvadr0n/NOT-JAVA

6. Список использованной литературы

1. Руководство по PHP – Установка mysqli [Электронный ресурс] – Режим доступа: https://www.php.net/manual/ru/mysqli.installation.php, свободный

2. Habr – Про docker compose [Электронный ресурс] – Режим доступа: https://habr.com/ru/company/ruvds/blog/450312/, свободный