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-03-20 Цемкало Алёна Ростиславовна

_____

(подпись студента)

Руководитель практической работы преподаватель Благирев М.М.

_____

(подпись руководителя)

Работа представлена «____»_____ 2022 г.

Допущен к работе «___» _____ 2022 г.

Москва 2022

СОДЕРЖАНИЕ

РТУ МИРЭА 1
Цель работы 3
Ход работы 4
ВЫВОД 9
Ответы на вопросы к практической работе 10
Ссылка на удаленный репозиторий проекта 17
Список использованной литературы 18

Цель работы

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

Для проверки работоспособности конфигурации 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 сгенерировать тестовую страничку, содержащую выборку из созданной таблицы и информационное сообщение о версии языка программирования, его настройках и конфигурации.

Ход работы

alena@DESKTOP-LETE34I: ~/apache-project

```
version: '3.8'
services:
  php-apache-environment:
    container_name: php-apache
    image: php:8.0-apache
    volumes:
      - ./php/src:/var/www/html
    ports:
      - 8000:80
```

Рисунок 1 - Создание PHP контейнера в файле docker-compose.yml

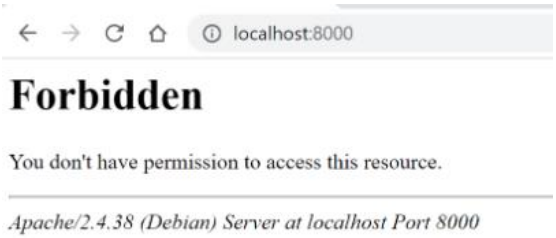

Рисунок 2 - Запущенный контейнер на хосте 8000

alena@DESKTOP-LETE34I: ~/apache-project/php/src

```
<html lang="en">
<head>
<title>Hello world page</title>
    <link rel="stylesheet" href="style.css" type="text/css"/>
</head>
        <body>
        <h1>Таблица пользователей данного продукта</h1>
<table>
<tr><th>Id</th><th>Name</th><th>Surname</th></tr>
<?php
$mysqli = new mysqli("db", "user", "password", "appDB");
$result = $mysqli->query("SELECT * FROM users");
foreach ($result as $row){
            echo "<tr><td>{$row['ID']}</td><td>{$row['name']}</td><td>{$row['surname']}</td></tr>";
}
?>
</table>
<?php
phpinfo();
?>
</body>
</html>
```

Рисунок 3 - Запись в файл index.php

alena@DESKTOP-LETE34I: ~/apache-project/php

```
FROM php:8.0-apache
RUN docker-php-ext-install mysqli && docker-php-ext-enable mysqli
RUN apt-get update && apt-get upgrade -y
```

Рисунок 4 - Установка и подключение mysqli в Dockerfile

alena@DESKTOP-LETE34I: ~/apache-project

```
version: '3.8'
services:
  php-apache-environment:
    container_name: php-apache
    build:
      context: ./php
      dockerfile: Dockerfile
    depends_on:
      - db
    volumes:
      - ./php/src:/var/www/html
    ports:
      - 8000:80
  db:
    container_name: db
    image: mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: "rootPswd"
      MYSQL_DATABASE: "appDB"
      MYSQL_USER: "user"
      MYSQL_PASSWORD: "password"
    ports:
      - "9906:3306"
```

Рисунок 5 - Настройка контейнера базы данных MySQL в docker-compose.yml

alena@DESKTOP-LETE34I: ~/apache-project

```
<html lang="en">
<head>
<title>Hello world page</title>
    <link rel="stylesheet" href="style.css" type="text/css"/>
</head>
        <body>
        <h1>Таблица пользователей данного продукта</h1>
<table>
<tr><th>Id</th><th>Name</th><th>Surname</th></tr>
<?php
$mysqli = new mysqli("db", "user", "password", "appDB");
$result = $mysqli->query("SELECT * FROM users");
foreach ($result as $row){
            echo "<tr><td>{$row['ID']}</td><td>{$row['name']}</td><td>{$row['surname']}</td></tr>";
}
?>
</table>
<?php
phpinfo();
?>
</body>
</html>
```

Рисунок 6 - Настройка страницы index.php

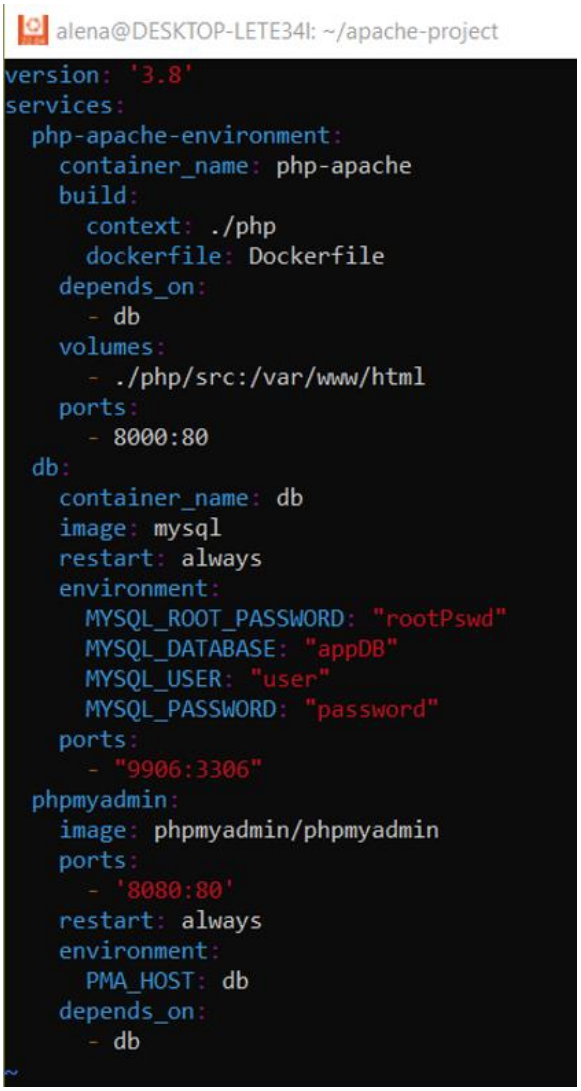

Рисунок 7 - Настройка образа PHPMyAdmin в docker-compose.yml

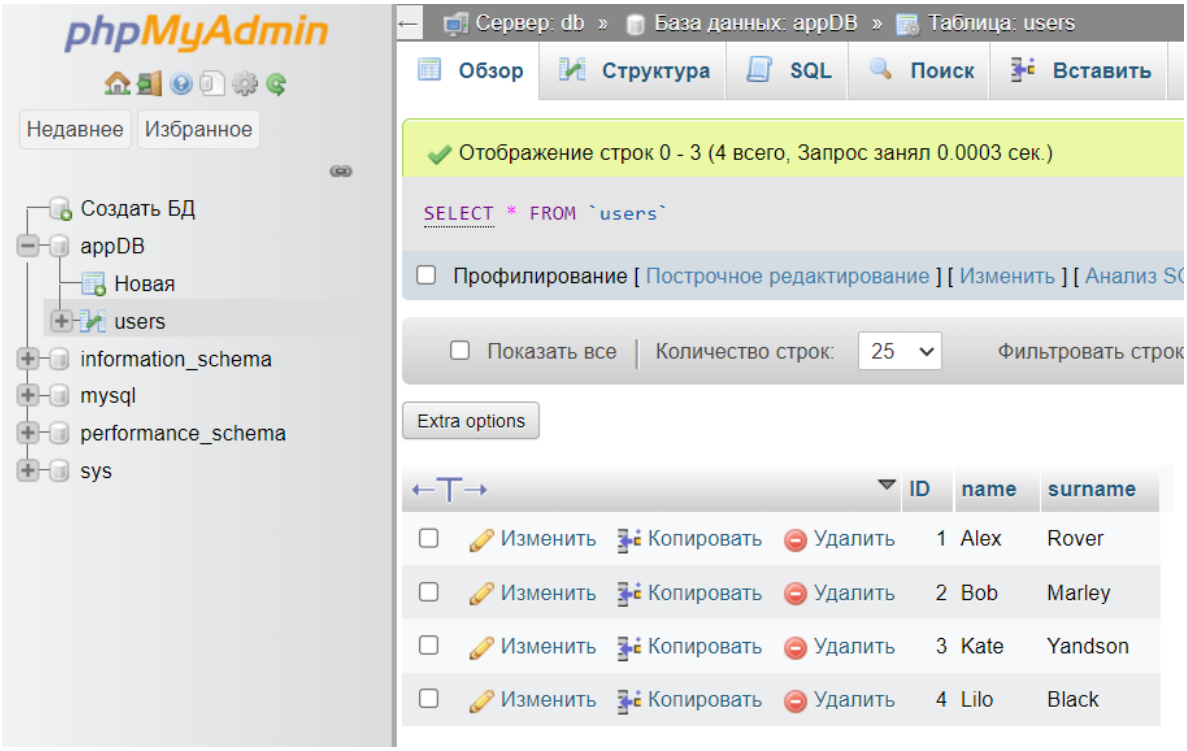

Рисунок 8 - Просмотр таблицы users базы данных appDB в PHPMyAdmin

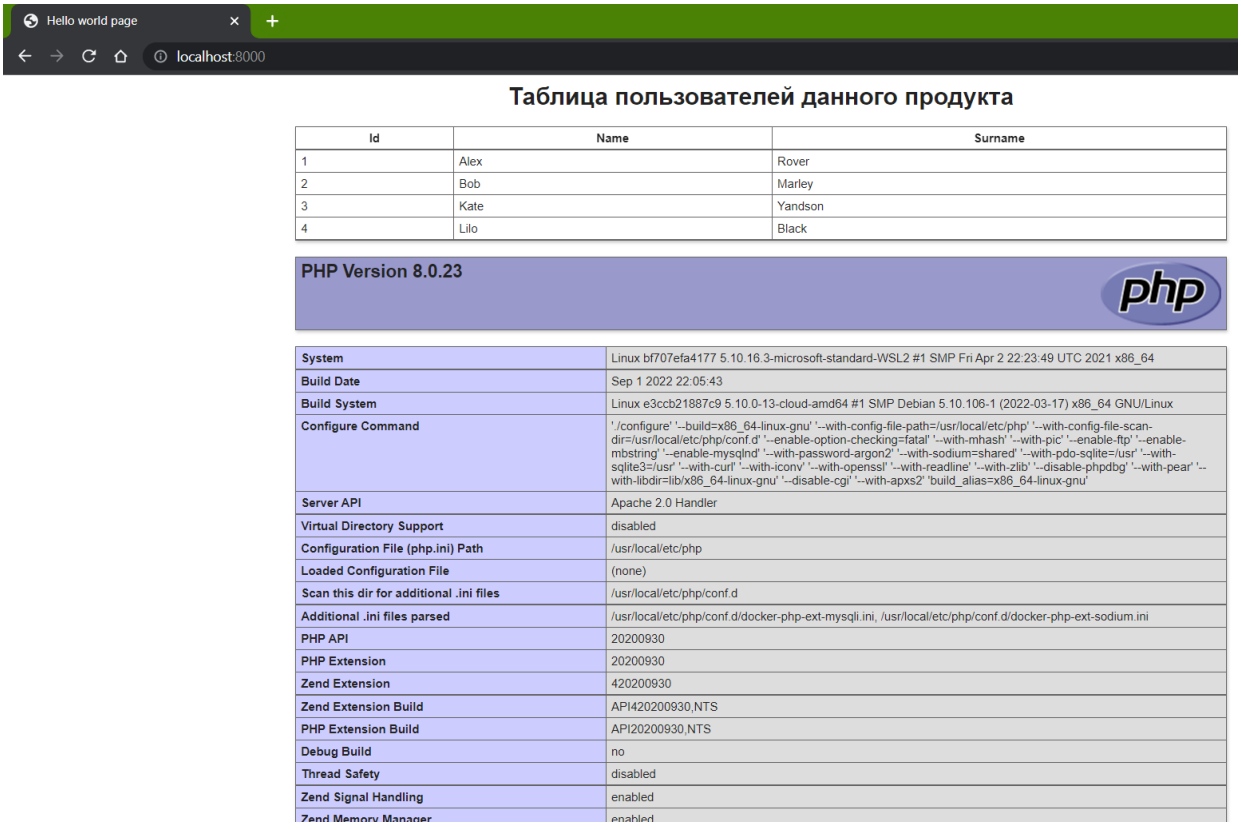

Рисунок 9 - Тестовая страница

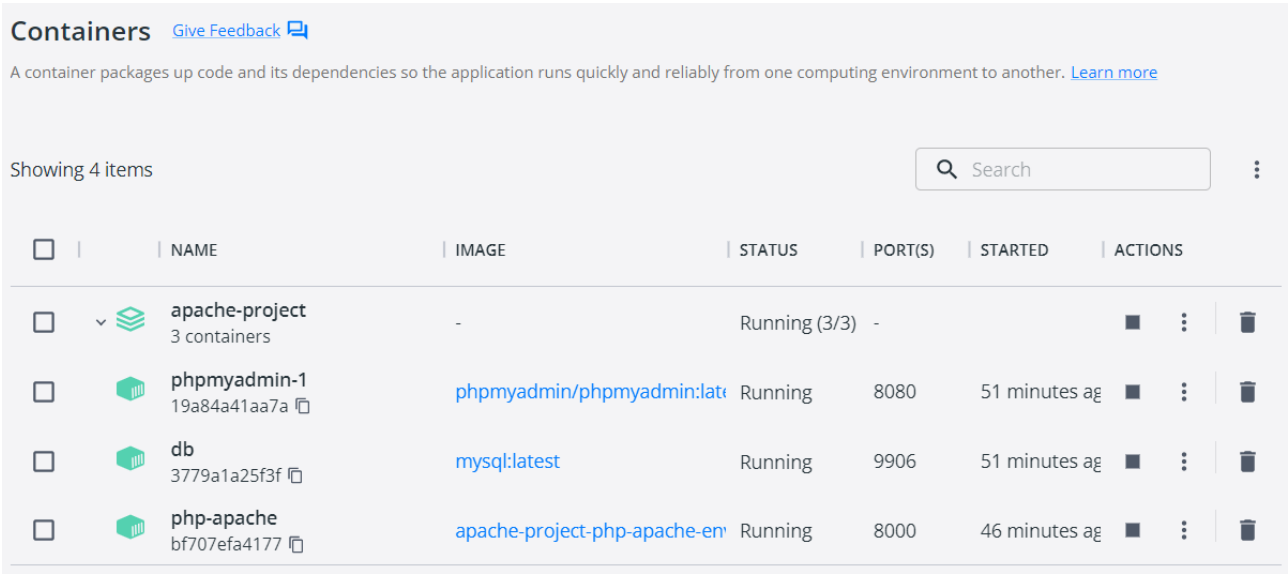

Рисунок 10 - Контейнеры в приложении Docker Desktop

ВЫВОД

Создана своя конфигурация серверного программного обеспечения, в которой присутствует веб-сервер, операционная система, язык программирования и база данных.

Для проверки работоспособности конфигурации инициализирована база данных. Также для проверки конфигурации сгенерирована тестовая страничка, содержащая выборку из созданной таблицы и информационное сообщение о версии языка программирования, его настройках и конфигурации.

Ответы на вопросы к практической работе

1. Сервер и клиент.

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.

Программа, являющаяся клиентом, взаимодействует с сервером, используя определённый протокол. Она может запрашивать с сервера какие-либо данные, манипулировать данными непосредственно на сервере, запускать на сервере новые процессы и т. п. Полученные от сервера данные клиентская программа может предоставлять пользователю или использовать как-либо иначе, в зависимости от назначения программы. Программа-клиент и программа-сервер могут работать как на одном и том же компьютере, так и на разных. Во втором случае для обмена информацией между ними используется сетевое соединение.

2. База данных.

База данных — это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе. База данных обычно управляется системой управления базами данных (СУБД). Данные вместе с СУБД, а также приложения, которые с ними связаны, называются системой баз данных, или, для краткости, просто базой данных.

Данные в наиболее распространенных типах современных баз данных обычно хранятся в виде строк и столбцов, формирующих таблицу. Этими данными можно легко управлять, изменять, обновлять, контролировать и упорядочивать. В большинстве баз данных для записи и запросов данных используется язык структурированных запросов (SQL).

3. API.

API (Application Programming Interface - прикладной программный интерфейс) - набор функций и подпрограмм, обеспечивающий взаимодействие клиентов и серверов.

API (в клиент-сервере) - описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой.

4. Сервис, отличия от сервера.

Сервер (программное обеспечение) - 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.

5. Архитектура клиент-сервер.

Модель клиент-сервер — это идея разделения системы или приложения на отдельные задачи, размещаемые на различных платформах для большей эффективности.

Клиент представляет собой программу представления данных, которая для их получения посылает запросы серверу, который в свою очередь может делать запрос к базе данных, обрабатывает данные и возвращает их к клиенту. Возможны случаи разделение обработки данных, когда часть работы сервера в виде обработки данных выполняет клиент. Но нужно понимать, что в этом случае очень важно разделение обязанностей и уровней доступа к данным на стороне клиента.

6. Виды сервисов.

- Сервисы приложений

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

- Веб-сервисы

Являются подвидом сервисов 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. Как правило, файл-сервер обеспечивает и определенный уровень защиты от несанкционированного доступа.

- Прокси-сервер

Прокси-сервер (от англ. proxy - представитель, уполномоченный; часто просто прокси, сервер-посредник) - промежуточный сервер (комплекс программ) в компьютерных сетях, выполняющий роль посредника.

- 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.

7. Масштабируемость.

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.

8. Протоколы передачи данных.

Протокол 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.

Сетевой протокол - набор правил и действий (и их очередности), позволяющий осуществлять соединение и обмен данными между двумя и более включенными в сеть устройствами.

9. Тонкий и толстый клиенты.

При применении толстого клиента полная функциональность приложения обеспечивается вне зависимости от сервера.

Тонким клиентом называют компьютеры и программы, функционирующие в терминальной или серверной сети. Множество задач по обработке данных осуществляются на главных компьютерах, к которым присоединено приложение и компьютер. Тонкий клиент же в отличие от толстого только отображает данные, принятые от сервера. Вся логика приложения выполняется на более производительном сервере, что не требует клиентских мощностей, кроме хорошего и стабильного канала связи.

10. Паттерн MVC: общие тезисы.

Первая часть данного паттерна – это модель (Model). 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 о элементах дизайна и любом другом отображении пользовательского интерфейса.

Представление (View) 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. Если рассматривать интернет-ресурсы представлением является html-страница

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.

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.

12. Паттерн MVC: 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.

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.

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. Данный вариант чаще всего используется при разработке веб-приложений.

14. Docker: общие тезисы и определения.

- docker daemon — сердце докера. Это демон, работающий на хост-машине, и умеющий сохранять с удалённого репозитория и загружать на него образы, запускать из них контейнеры, следить за запущенными контейнерами, собирать логи и настраивать сеть между контейнерами (а с версии 0.8 и между машинами). А еще именно демон создает образы контейнеров, хоть и может показаться, что это делает docker-client.
- docker — это консольная утилита для управления docker-демоном по HTTP. Она устроена очень просто и работает предельно быстро. Вопреки заблуждению, управлять демоном докера можно откуда угодно, а не только с той же машины. В сборке нового образа консольная утилита docker принимает пассивное участие: архивирует локальную папку в tar.gz и передает по сети docker-daemon, который и делает всю работу. Именно из-за передачи контекста демону по сети, лучше собирать тяжелые образы локально.

- docker Hub централизованно хранит образы контейнеров. Когда вы пишете “docker run ruby”, docker скачивает самый свежий образ с ruby именно из публичного репозитория. Изначально хаба не было, его добавили уже после очевидного успеха первых двух частей.

15. Dockerfile.

В файлах Dockerfile содержатся инструкции по созданию образа. С них, набранных заглавными буквами, начинаются строки этого файла. После инструкций идут их аргументы. Инструкции, при сборке образа, обрабатываются сверху вниз.

Слои в итоговом образе создают только инструкции FROM, RUN, COPY, и ADD. Другие инструкции что-то настраивают, описывают метаданные, или сообщают Docker о том, что во время выполнения контейнера нужно что-то сделать, например — открыть какой-то порт или выполнить какую-то команду.

16. 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 вебприложений (помимо PHP могут подразумеваться другие языки, такие как Perl и Python).

Ссылка на удаленный репозиторий проекта

https://github.com/tsemkaloalena-mirea/server_parts_of_internet_resources_development

Список использованной литературы

1. Видео “Введение в Докер” на английском языке от создателя: Introduction to Docker (https://www.youtube.com/watch?v=Q5POuMHxW-0)
2. Статья о назначении докера простыми словами: https://habr.com/ru/post/309556/
3. Более сложная и подробная статья про докер: https://habr.com/ru/post/277699/
4. Хорошая статья с пингвинами для прочтения после туториала по докеру: https://habr.com/ru/post/250469/
5. Официальная документация докера: https://docs.docker.com/
6. Статья о конкретном опыте использования докер контейнеров: https://habr.com/ru/post/247969/
7. Туториал по докеру: https://badcode.ru/docker-tutorial-dlia-novichkovrassmatrivaiem-docker-tak-iesli-by-on-byl-ighrovoi-pristavkoi/
8. Туториал по докеру с Хабра: https://habr.com/ru/post/310460/
9. Шпаргалка с командами Docker: https://habr.com/ru/company/flant/blog/336654/
10. Ссылка на скачивание докера с официального сайте: https://www.docker.com/products/docker-desktop
11. Отличная статья про dockerfile: https://habr.com/ru/company/ruvds/blog/439980/
12. Установка и настройка PHP: https://www.php.net/manual/ru/install.php
13. Настройка среды PhpStorm и полезные фичи: https://habr.com/ru/post/282003/
14. Про docker compose: https://habr.com/ru/company/ruvds/blog/450312/
15. Docker hub: https://hub.docker.com/