|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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:**

**«Конфигурация серверного программного обеспечения»**

**Студент группы** ИКБО-30-20 Лазонов А.Е.

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

(подпись студента)

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

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

(подпись руководителя)

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

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

Москва 2022

**Содержание**

1. Содержание 2

2. Цель работы 3

3. Ход работы 4

4. Выводы 8

5. Ответы на вопросы к практической работе 8

6. Ссылка на удаленный репозиторий проекта 14

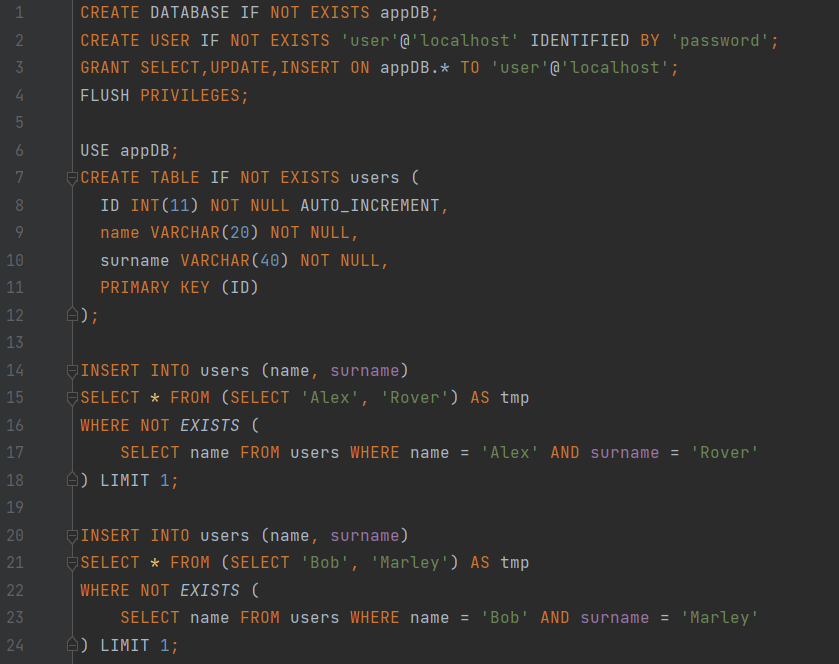
7. Список использованной литературы 15

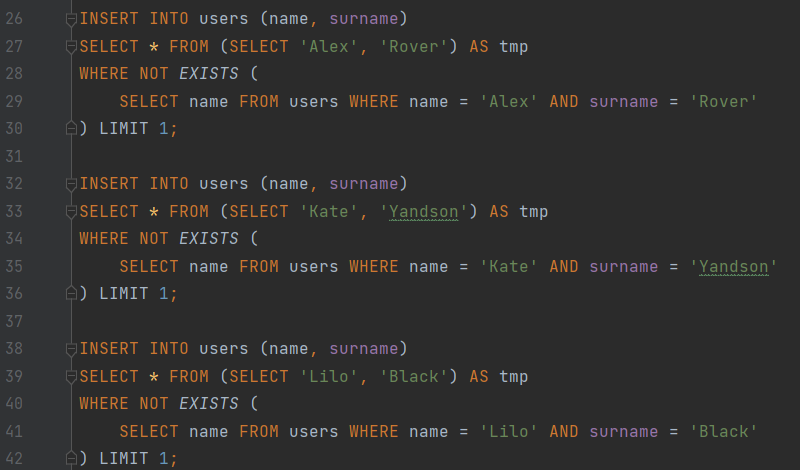
**Цель работы**

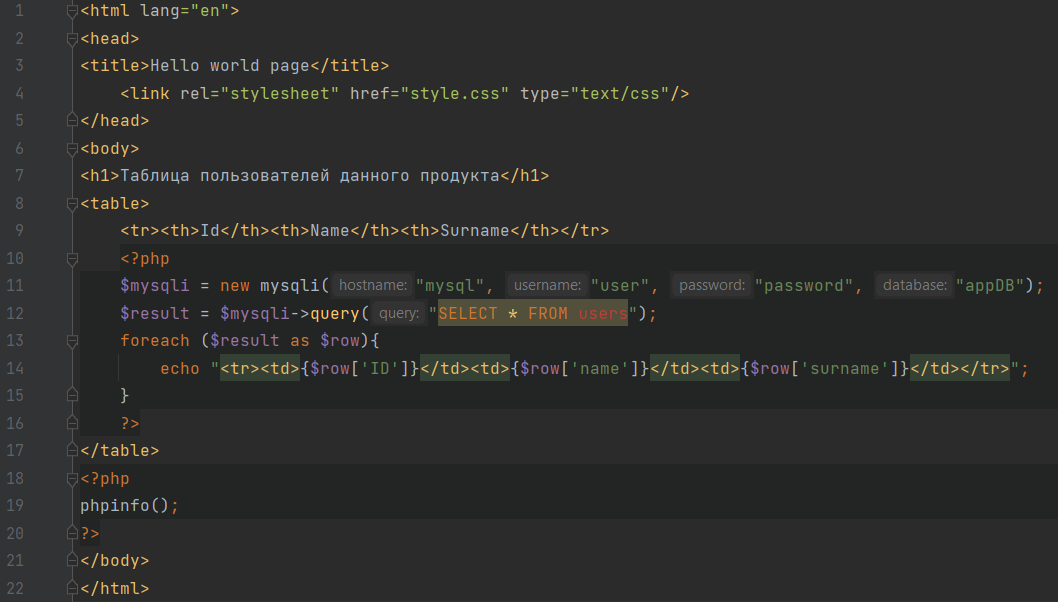
Создать свою конфигурацию серверного программного обеспечения, в которой должны присутствовать веб-сервер, операционная система, язык программирования и база данных. Данная конфигурация будет использоваться для выполнения следующих практических работ по данной дисциплине и для выполнения курсового проектирования. Для этого использовать программное обеспечение для автоматизации развертывания и управления приложениями в средах с поддержкой контейнеризации Docker.

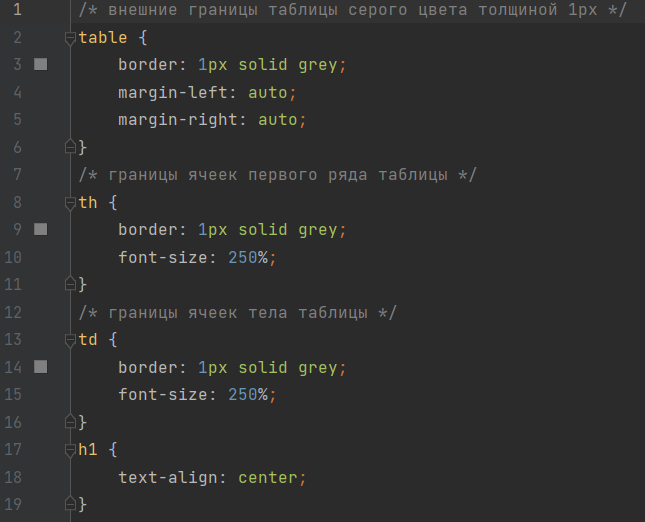
**Ход работы**

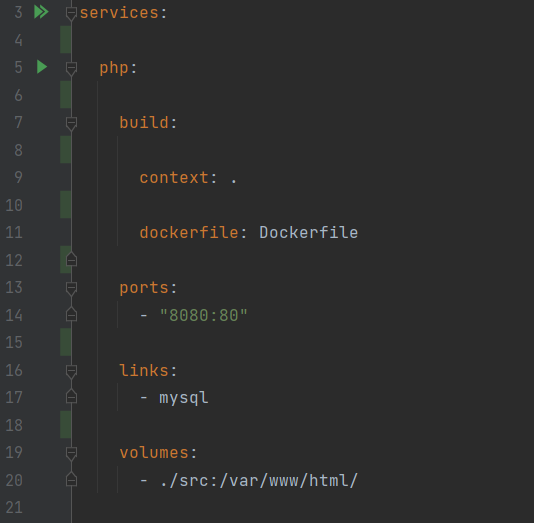
Существует проблема разработки того или иного приложения и его развертывания на других машинах. Самыми частыми решениями данной проблемы является установочные скрипты, облачные сервисы и виртуальные машины. Описанные подходы не являются оптимальными что раздувает техническую поддержку до максимума, а 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 hostсистеме. Подход, используемый Docker, находится посередине между запуском всего на физическом сервере и полной виртуализацией, предлагаемой виртуальными машинами. Этот подход называется контейнеризацией, что и требовалось применить в текущей практической работе. Мною были написаны файлы Dockerfile и docker-compose.yml для создания и совмещения собственных контейнеров в единое «приложение», файл с базой данных, оформленная таблица для неё и выведение содержимого в формате веб-страницы было создано автоматически.

Рисунок 1 — Содержимое файла создания БД

Рисунок 2 — Содержимое файла с созданием БД

Рисунок 3 — Описание графической части страницы

Рисунок 4 — Описание стилей таблицы

Рисунок 5 — Описание сервиса php в docker-compose.yml

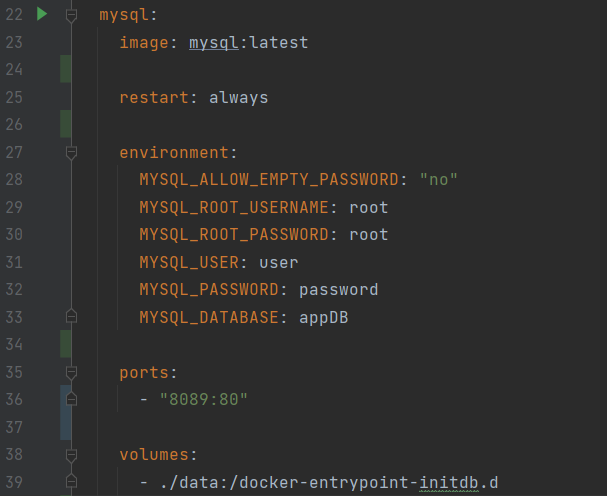
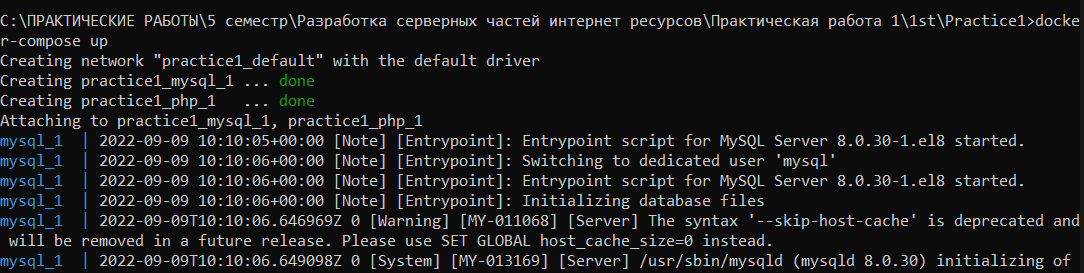
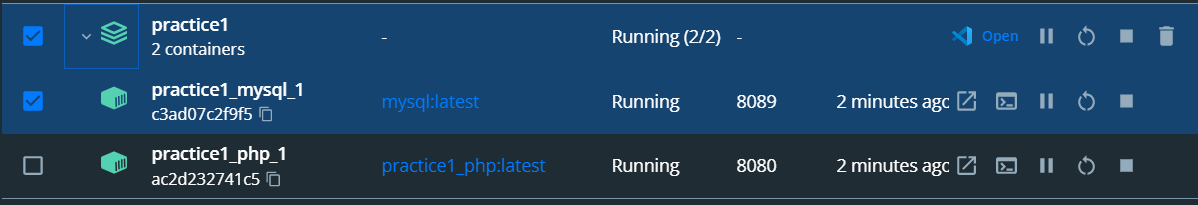
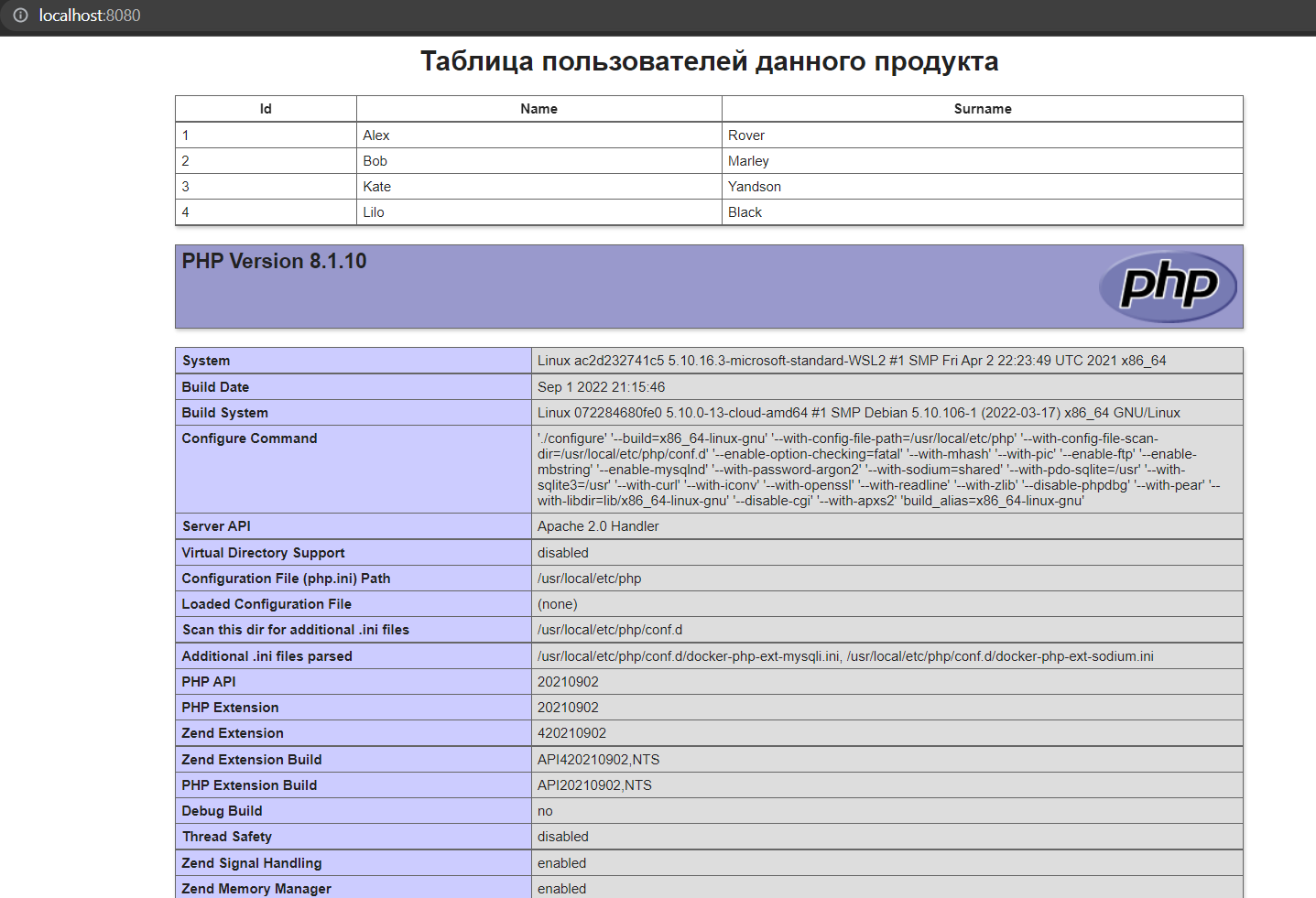
Рисунок 6 — Описание сервиса mysql в docker-compose.yml

Рисунок 7 — Содержимое Dockerfile

Рисунок 8 — Сборка контейнеров

Рисунок 9 — Проверка правильности сборки контейнеров

Рисунок 10 — Тестовая страница

**Выводы**

В результате выполнения текущей практической работы были получены практические навыки по развертыванию собственного контейнера с помощью Docker, создана конфигурация серверного программного обеспечения в качестве основы для дальнейшего построения полноценного проекта.

**Ответы на вопросы к практической работе**

1. Сервер и клиент:

* + - * Сервер — выделенный или специализированный компьютер для выполнения сервисного программного обеспечения.
      * Клиент — компьютерное устройство, которое отсылает запросы серверу, касающиеся выполнения определенных задач или предоставления конкретной информации.

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 СУБД . Данные вместе с ней, а также приложения, которые с ними связаны, называются системой баз данных, или просто базой данных.

3. API:

* + - * API (Application Programming Interface — программный интерфейс приложения) — специальный протокол для взаимодействия компьютерных программ, который позволяет использовать функции одного приложения внутри другого.

4. Сервис, отличия от сервера:

* + - * Веб-сервер - это сервер реализующий http протокол. Веб-сервис - это технология для взаимодействия между системами. REST подход подразумевает передачу представления ресурсов, это может быть HTML,JSP,XML, реализует api. Веб-сервер же реализует http протокол.

5. Архитектура клиент-сервер:

* + - * Архитектура «Клиент-Сервер» предусматривает разделение процессов предоставление услуг и отправки запросов на них на разных компьютерах в сети, каждый из которых выполняет свои задачи независимо от других. В ней несколько компьютеров-клиентов посылают запросы и получают услуги от централизованной служебной машины — сервера, которая также называется хост-системой. Клиентская машина предоставляет пользователю «дружественный интерфейс» для облегчения взаимодействия с сервером.

6. Виды сервисов:

* + - * Сервер приложений — это программная платформа, предназначенная для эффективного исполнения программ, скриптов, на которых построены приложения.
      * Веб-серверы изначально предоставляли доступ к гипертекстовым документам по протоколу HTTP. Сейчас поддерживают расширенные возможности, в частности, передачу произвольных данных.
      * Серверы баз данных используются для обработки запросов. На сервере находится СУБД для управления БД и ответов на запросы.
      * Файл-сервер хранит информацию в виде файлов и предоставляет пользователям доступ к ней.
      * Прокси-сервер — промежуточный сервер в компьютерных сетях, выполняющий роль посредника. Существует несколько видов прокси-серверов.
      * Файрволы — 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:

* + - * 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. Основными являются горизонтальная и вертикальная масштабируемость.

8. Протоколы передачи данных:

* + - * Сетевой протокол — это набор правил, определяющий принципы взаимодействия устройств в сети. Чтобы отправка и получение информации прошли успешно, все устройства-участники процесса должны принимать условия протокола и следовать им. В сети их поддержка встраивается или в аппаратную или программную части.

9. Тонкий и толстый клиенты:

* + - * Толстый клиент — у него полная функциональность приложения обеспечивается вне зависимости от сервера. В данном случае сервер чаще всего выступает в роли хранилища информации, а вся логика приложения, как и механизм отображения данных располагаются и выполняются на клиенте. Даже при отсутствии соединения с сервером работа ведется с локальными копиями данных, а при возобновлении соединения происходит синхронизация данных.
      * Тонкий клиент — 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. Любой сбой на сервере и в канале связи влечет “падение” всего приложения.

10. Паттерн MVC (общие тезисы):

* + - * Для решения проблем проектирования были созданы архитектурные конструкции, называемые паттернами проектирования или часто коротко паттернами. 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. Третьим компонентом системы является **контроллер (Controller).** Данный компонент является неким буфером между моделью и представлением. Обобщенно он управляет представлением на основе изменения модели.

11. Патерн 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: 1. Двусторонняя коммуникация с представлением; 2. Представление взаимодействует напрямую с презентером, путем вызова соответствующих функций или событий экземпляра презентера. 3 Презентер взаимодействует с View путем использования специального интерфейса, реализованного представлением. 4. 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: 1. Двусторонняя коммуникация с представлением. 2. View-модель — это абстракция представления. Означает, что свойства представления совпадают со свойствами View-модели / модели. 3. View-модель не имеет ссылки на интерфейс представления (IView). Изменение состояния View-модели автоматически изменяет представление и наоборот, поскольку используется механизм связывания данных (Bindings). 4. 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: 1. Контроллер определяет, какое представление должно быть отображено в требуемый момент. 2. События представления могут повлиять только на контроллер. Контроллер может повлиять на модель и определить другое представление. 3. Возможно несколько представлений только для одного контроллера.

14. Docker (общие тезисы и определения):

* + - * Существует проблема разработки того или иного приложения и его развертывания на других машинах. 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. Основные компоненты: docker-daemon, docker, docker Hub.

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.

16. Docker Compose:

* + - * Docker Compose механизм объединения нескольких контейнеров и их оркестровки. Таким инструментом является Docker Compose. Это средство для решения задач развертывания проектов. Также 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:

* + - * LAMP — акроним, обозначающий 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 веб-приложений.

**Ссылка на удаленный репозиторий проекта**

* + 1. Репозиторий проекта: <https://github.com/ISilentI/Practice1>

**Список использованной литературы — ИЗМЕНИТЬ + СДЕЛАТЬ АДЕКВАТНЫЕ ПЕРЕНОСЫ**

* 1. Официальная документация докера: <https://docs.docker.com/>
  2. Команды Docker: <https://habr.com/ru/company/flant/blog/336654/>
  3. Статья о Dockerfile: <https://habr.com/ru/company/ruvds/blog/439980/>
  4. Статья о docker-compose: <https://habr.com/ru/company/ruvds/blog/450312/>
  5. Настройка среды PhpStorm: <https://habr.com/ru/post/282003/>