|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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-01-19 Кузин Данил Олегович

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

(подпись студента)

**Руководитель практической работы** преподаватель Волков М.Ю.

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

(подпись руководителя)

Работа представлена «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Допущен к работе «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Москва 2021

**Оглавление**

[Ход работы 2](#_Toc82175485)

[Вывод 3](#_Toc82175486)

[Ответы на вопросы к практической работе 5](#_Toc82175487)

[Список использованной литературы 9](#_Toc82175488)

# Ход работы

1. В рабочей директории был создан файл docker-compose.yml (Рисунок 1)

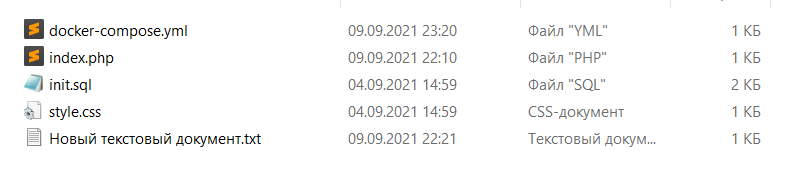


Рисунок 1 – Структура проекта

Содержимое файла docker-compose.yml представлено в листинге 1.

Листинг 1 – Файл docker-compose.yml

version: "3"

services:

apache:

image: php:7.4-apache

depends\_on:

- mysql

ports:

- 8080:80

volumes:

- .:/var/www/html

command: 'sh -c "docker-php-ext-install mysqli && exec apache2-foreground"'

links:

- mysql:db

mysql:

image: 'mysql:latest'

restart: 'always'

volumes:

- ./init.sql:/init/init.sql

command: --init-file /init/init.sql

environment:

MYSQL\_ROOT\_PASSWORD: password

1. С помощью команды docker-compose up, вызванной в рабочей директории, был запущен процесс сборки, создания и запуска контейнеров с php, apache и MySQL (Рисунок 2).

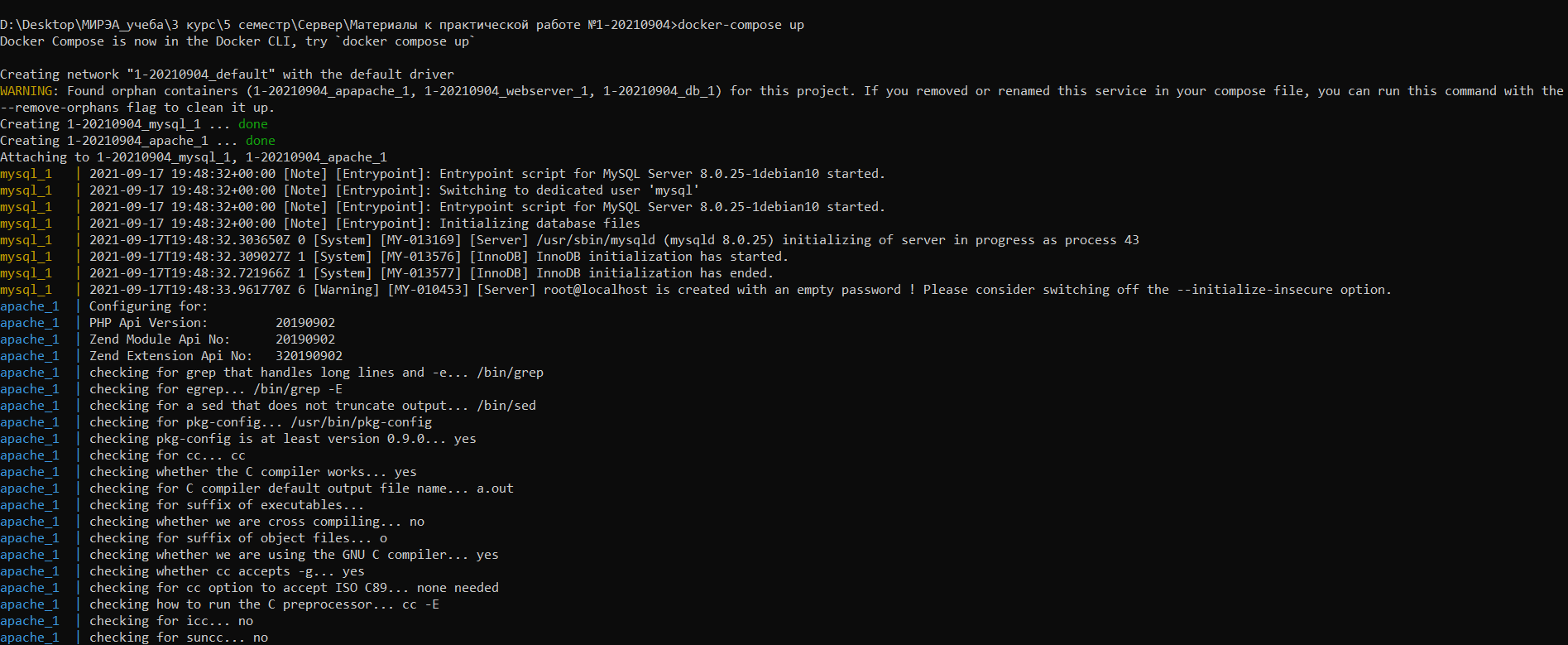


Рисунок 2 - Запуск контейнеров

В результате на порту 8080 был запущен веб-сервер, отображающий выборку из базы данных и информационное сообщение о версии языка программирования, его настройках и конфигурации (Рисунок 3).



Рисунок 3 - Проверка работы

# Вывод

В результате работы для создания конфигурации серверного программного обеспечения был использован инструмент docker-compose для оркестровки контейнеров, который запускает мультиконтейнерное приложение с ОС Linux, языком программирования php, веб-сервером apache и MySQL. Для проверки конфигурации была сгенерирована тестовая страница с необходимыми данными.

# Ответы на вопросы к практической работе

1. Сервер и клиент.

Сервер – компьютер, обслуживающий другие компьютеры в сети и предоставляющий им свои ресурсы для решения определенных задач. Клиент – компьютер, использующий ресурсы сервера и предоставляющий пользователю возможность взаимодействия с системой и сетью.

1. База данных.

База данных — совокупность данных, организованных в соответствии с концептуальной структурой, описывающей характеристики этих данных и взаимоотношения между ними, которая поддерживает одну или более областей применения.

1. API.

API (программный интерфейс приложения, интерфейс прикладного программирования) — описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой. Обычно входит в описание какого-либо интернет-протокола (например, RFC), программного каркаса (фреймворка) или стандарта вызовов функций операционной системы.

1. Сервис, отличия от сервера.

Сервис – программное обеспечение, функциональность или набор программных функций, используемое клиентами для выполнения каких-либо операций (получение, обновление информации) совместно с политиками, определяющими их использование.

Сервер – компьютер, обслуживающий другие компьютеры в сети и предоставляющий им свои ресурсы для решения определенных задач.

1. Архитектура клиент-сервер.

«Клиент — сервер» — вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами. Фактически клиент и сервер — это программное обеспечение. Обычно эти программы расположены на разных вычислительных машинах и взаимодействуют между собой через вычислительную сеть посредством сетевых протоколов, но они могут быть расположены также и на одной машине.

1. Виды сервисов.

Сервисы можно классифицировать как «Технические» и «Профессиональные». Под «Техническим Сервисом» понимается основанная на технологии возможность, которую использует или потребляет заказчик для облегчения бизнес-процесса или какой-либо функции. Под «Профессиональным Сервисом» понимаются приносящие пользу действия ИТ персонала, которые он выполняет для поддержки, сопровождения, мониторинга и обеспечения работоспособности технических сервисов.

1. Масштабируемость.

Масштабируемость — это наращивание инфраструктуры его компонентов (вычислительных ресурсов, хранилища, сети), чтобы оно могло единовременно обслуживать больше запросов.

1. Протоколы передачи данных.

**TCP/IP**

Это стек протоколов TCP и IP. Первый обеспечивает и контролирует надёжную передачу данных и следит за её целостностью. Второй же отвечает за маршрутизацию для отправки данных. Протокол TCP часто используется более комплексными протоколами.

**UDP**

Протокол, обеспечивающий передачу данных без предварительного создания соединения между ними. Этот протокол является ненадёжным. В нём пакеты могут не только не дойти, но и прийти не по порядку или вовсе продублироваться.

Основное преимущество UDP протокола заключается в скорости доставки данных. Именно поэтому чувствительные к сетевым задержкам приложения часто используют этот тип передачи данных.

**FTP**

Протокол передачи файлов. Его использовали ещё в 1971 году — задолго до появления протокола IP. На текущий момент этим протоколом пользуются при удалённом доступе к хостингам. FTP является надёжным протоколом, поэтому гарантирует передачу данных. Этот протокол работает по принципу клиент-серверной архитектуры. Пользователь проходит аутентификацию (хотя в отдельных случаях может подключаться анонимно) и получает доступ к файловой системе сервера.

**HTTP**

Изначально протокол передачи HTML-документов. Сейчас же он используется для передачи произвольных данных в интернете. Он является протоколом клиент-серверного взаимодействия без сохранения промежуточного состояния. В роли клиента чаще всего выступает веб-браузер, хотя может быть и, например, поисковый робот. Для обмена информацией протокол HTTP в большинстве случаев использует TCP/IP. HTTP имеет расширение HTTPS, которое поддерживает шифрование. Данные в нём передаются поверх криптографического протокола TLS.

1. Тонкий и толстый клиенты.

Толстый клиент обеспечивает полное функционирование независимо от сервера, а все расчёты и операции выполняются на устройстве клиента.

Тонкий клиент зависит от сервера и служит лишь приложением для получения или отправки информации на сервер, где выполняются все операции и вычисления.

1. Паттерн MVC: общие тезисы.

MVC – это фундаментальный паттерн (шаблон для проектирования), который используется во многих технологиях. Состоит из модели, представления и контроллера.

Модель – это часть, содержащая в себе функциональную бизнес-логику приложения. Она полностью независима от остальных частей продукта.

Представление – это часть, отвечающая за отображение данных полученных от модели. Также оно не может напрямую влиять на модель.

Контроллер – часть, обрабатывающая запросы, собирающая и отсылающая View-модель конечному пользователю.

1. Паттерн MVC: Model-View-Presenter.

Данный подход позволяет создавать абстракцию представления. Выделяется интерфейс представления с определенным набором свойств и методов, а Presenter получает ссылку на реализацию интерфейса, подписывается на события представления и по запросу изменяет модель. Используется в ситуации, когда невозможно связывание данных.

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

Данный подход позволяет связывать элементы представления со свойствами и событиями View-модели. Используется в ситуации, когда возможно связывание данных без необходимости ввода специальных интерфейсов представления.

1. Паттерн MVC: Model-View-Controller.

Основная идея этого паттерна в том, что и контроллер, и представление зависят от модели, но модель никак не зависит от этих двух компонентов. Используется в ситуации, когда связь между представлением и другими частями приложения невозможна.

1. Docker: общие тезисы и определения.

Docker – программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации. Позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер, который может быть перенесён на любую Linux-систему с поддержкой cgroups в ядре, а также предоставляет среду по управлению контейнерами.

1. Dockerfile.

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.

1. Docker Compose.

Docker-compose позволяет запускать несколько контейнеров, связывать их и определять различные свойства контейнера в одном файле. Этот файл называется docker-compose.yml написанный в формате YAML.

1. LAMP.

LAMP – акроним, обозначающий набор (комплекс) серверного программного обеспечения, широко используемый в интернете: Linux, Apache, MariaDB/MySQL, PHP.

# Список использованной литературы

1. Сейерс, Э. Х. Docker на практике / Э. Х. Сейерс, А. Милл ; перевод с английского Д. А. Беликов. — Москва : ДМК Пресс, 2020. — 516 с. — ISBN 978-5-97060-772-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/131719 (дата обращения: 10.09.2021). — Режим доступа: для авториз. пользователей.
2. Одиночкина, С. В. Web-программирование PHP : учебно-методическое пособие / С. В. Одиночкина. — Санкт-Петербург : НИУ ИТМО, 2012. — 79 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/43562 (дата обращения: 10.09.2021). — Режим доступа: для авториз. пользователей.
3. Филиппов, Ф. В. HTTP + PHP в примерах и задачах : учебное пособие / Ф. В. Филиппов, А. Н. Губин. — Санкт-Петербург : СПбГУТ им. М.А. Бонч-Бруевича, 2015. — 67 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/180044 (дата обращения: 18.09.2021). — Режим доступа: для авториз. пользователей.