|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра инструментального и прикладного программного обеспечения (ИППО)**

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**

по дисциплине «Разработка серверных частей интернет-ресурсов»

**Практическая работа № 1**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент группы ИНБО-01-17 | ИКБО-20-20 Мажарова А.А. | (подпись) | |
| Преподаватель | Благирев М.М. | (подпись) | |
| Отчет представлен | «\_\_\_»\_\_\_\_\_\_\_\_2022\_г. | |  | |

Москва 2022 г.

**Оглавление**

[Ход работы 3](#_Toc82263759)

[Вывод 4](#_Toc82263760)

[Список используемой литературы 4](#_Toc82263761)

# Ход работы

1. В рабочей директории был создан файл для практической.

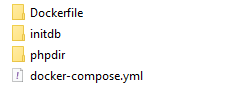


Рисунок - Файловая система проекта

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

Листинг 1 – Файл docker-compose.yml

version: '3.7'

services:

  db:

    image: mysql:latest

    command: --default-authentication-plugin=mysql\_native\_password

    environment:

      MYSQL\_ROOT\_PASSWORD: Alis

    volumes:

    - "./initdb/:/docker-entrypoint-initdb.d"

    - db\_volume:/var/lib/mysql

  www:

    depends\_on:

    - db

    build: ./Dockerfile

    volumes:

    - "./phpdir/:/var/www/html"

    ports:

    - 80:80

    - 443:443

volumes:

  db\_volume:

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).

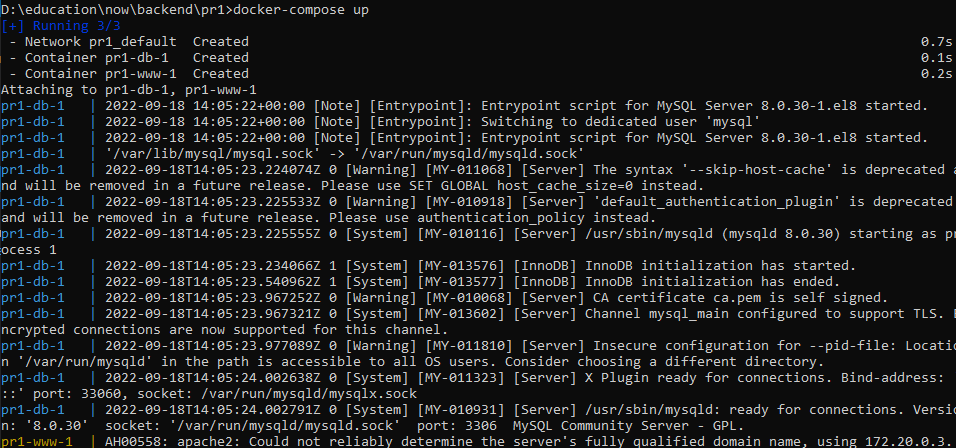


Рисунок - Запуск контейнеров

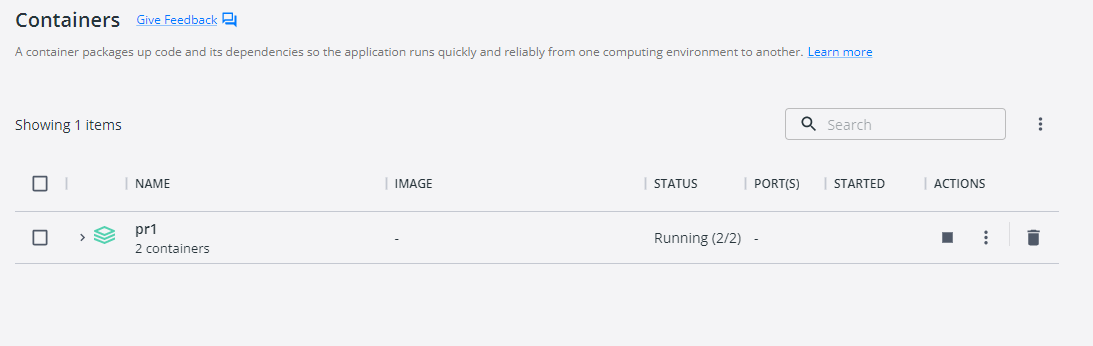


Рисунок 3 – контейнеры в докере

В результате на порту 80 был запущен веб-сервер, отображающий выборку из базы данных и информационное сообщение о версии языка программирования, его настройках и конфигурации (Рисунок 4).

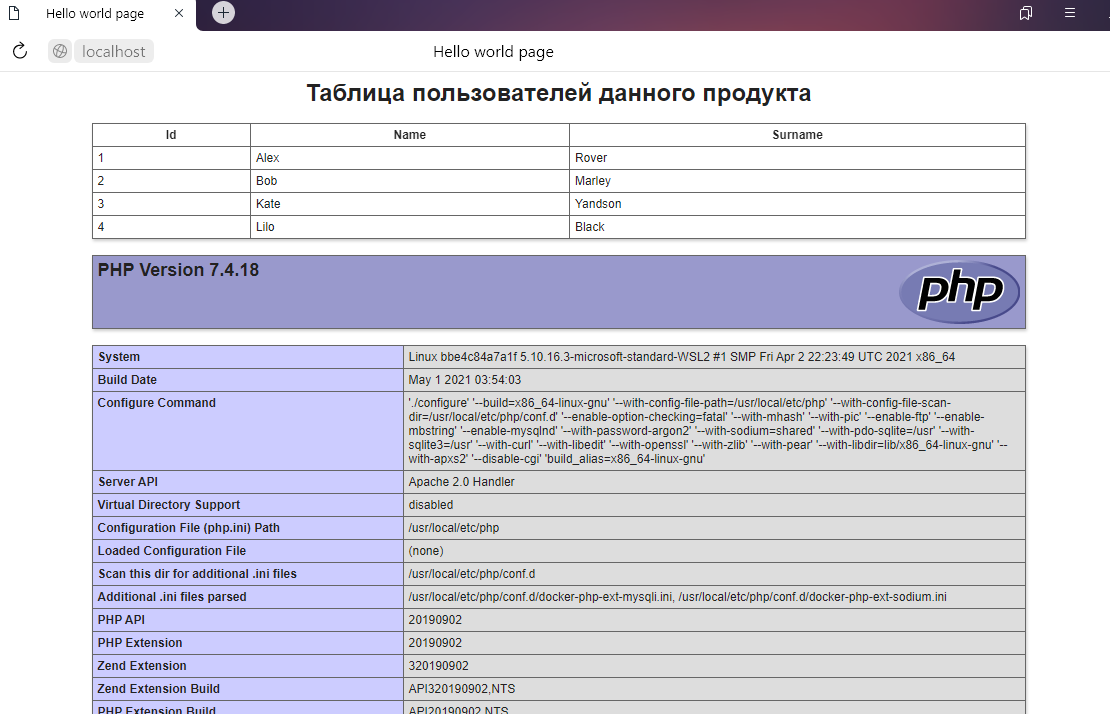


Рисунок 4 - Проверка работы

# Вывод

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

# Список используемой литературы

1. Статья о назначении докера простыми словами: <https://habr.com/ru/post/309556/>
2. Статья о конкретном опыте использования докер контейнеров: <https://habr.com/ru/post/247969/>
3. Официальная документация докера: <https://docs.docker.com/>

# 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.

1. База данных.

**База данных** – это информационная модель, позволяющая упорядоченно хранить данные об объекте или группе объектов, обладающих набором свойств, которые можно категорировать.

1. API.

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

1. Сервис, отличия от сервера.

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

Отличается тем, что можно быстро и легко заменить на новый, обновить, в отличие от сервера, так как сервер – это главный связующий элемент.

1. Архитектура клиент-сервер.

Клиент общается с сервером (клиент шлёт запрос, сервер его обрабатывает и отвечает клиенту)

Есть двухзвенная архитектура: клиент, сервер и база данных.

Есть трёхзвенная архитектура: клиент, сервер приложения, сервер управления БД и база данных

1. Виды сервисов.

Виды сервисов: файл-серверы, файрволы, серверы баз данных, серверы приложений, веб-серверы, почтовые серверы, прокси-серверы.

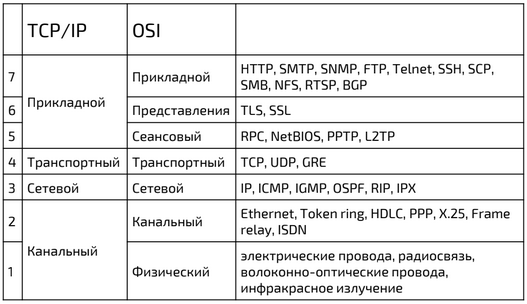
1. Масштабируемость.

**Вертикальная масштабируемость** – увеличение производительности компонентов серверной системы в интересах повышения производительности всей системы

**Горизонтальная масштабируемость** – разбиение системы на более мелкие структурные компоненты и разнесение их, так и увеличение количества компонентов, параллельно выполняющих одну и ту же функцию

1. 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.



1. Тонкий и толстый клиенты.

**Толстый клиент** – логика приложения и отображение данных обрабатывается на клиенте, а хранение данных – на сервере.

**Тонкий клиент** – на клиенте только отображение данных, остальное на сервере.

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.

Пример использования:Windows Forms.

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.

Пример использования: WPF.

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.

Пример использования: MVC ASP.NET, Spring MVC, Django.

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.

В файлах Dockerfile содержатся инструкции по созданию и сборке образа. 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.

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.