A gold eagle with two heads and a blue circle with a white circle and a blue circle with a blue circle with a blue circle with a white circle with a blue circle with a blue circle with

AI-generated content may be incorrect.

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 «Проектирование и разработка серверных частей интернет- ресурсов»

**Студент группы** ИКБО-21-23 Муравьев А.О.

(подпись студента)

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

(подпись руководителя)

Работа представлена « » 2025 г.

Допущен к работе « » 2025 г.

Москва 2025

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 1](#_Toc194713266)

[ВВЕДЕНИЕ Ошибка! Закладка не определена.](#_Toc194713267)

[ВЫПОЛНЕНИЕ РАБОТЫ 4](#_Toc194713268)

[ВЫВОД Ошибка! Закладка не определена.](#_Toc194713269)

[ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ 10](#_Toc194713270)

# ЦЕЛЬ РАБОТЫ

Предлагается 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 вашей 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 вашей конфигурации требуется сгенерировать тестовую страничку, содержащую выборку из созданной таблицы и информационное сообщение о версии языка программирования, его настройках и конфигурации.

# ХОД РАБОТЫ

Для создания образа необходимого веб-сервера, был использован Dockerfile, изображённый на рисунке 1.

### Рисунок 1 – Dockerfile для задания

Здесь в качестве основы для нашего образа мы используем официальный образ PHP, затем копируем содержимое сервера, находящееся в текущей директории, в файловую систему веб-сервера и устанавливаем mysqli для корректной работы приложения.

Для связи сервера и его базы данных мы будем использовать dockercompose. Его содержимое показано на рисунке 2.

### Рисунок 2 – Файл docker-compose.yml

Здесь есть два сервиса: веб-сервер и база данных. При этом веб-сервер зависит от базы данных. В настройках сервера мы указываем Dockerfile, который создаст нужный нам образ, а также порты и тома для работы с нужными нам файлами сервера.

В сервисе базы данных мы, в свою очередь, задаем ее базовый образ и указываем переменные окружения: пользователя, название базы данных, корневой и пользовательский пароли. Помимо этого, мы указываем том для корректной инициализации базы данных на основе файла init.sql. На рисунке 3 показана итоговая файловая структура проекта.

### Рисунок 3 – Структура проекта

Соберём проект с помощью команды docker compose build (рисунок 4).

### Рисунок 4 – Сборка проекта

Запустим проект с помощью команды docker compose up (рисунок 5).

### Рисунок 5 – Запуск проекта

Оба сервиса запущены и работают, что видно в Docker Desktop (рисунок 6).

### Рисунок 6 – Сервисы в Docker Desktop

Теперь по адресу localhost:8000 (или 127.0.0.1:8000) можно увидеть сайт, продемонстрированный на рисунке 7.

### Рисунок 7 – Проверка работы проекта

# ВЫВОД

Таким образом, был произведен корректный запуск приложенного к практической работе php скрипта генерации страницы с характеристиками веб-сервера.

Исходный код проекта расположен по адресу: <https://github.com/alexomur/MireaBackend/tree/master/Prac1>

# ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Сервер и клиент – Сервер предоставляет ресурсы/услуги, клиент их потребляет (например, веб-сервер и браузер).

2. База данных – Организованное хранилище данных (например, MySQL), управляемое СУБД.

3. API – Интерфейс для взаимодействия между программными компонентами (например, REST API).

4. Сервис, отличия от сервера – Сервис выполняет конкретную функцию (например, авторизация), сервер — физическая/виртуальная система, предоставляющая сервисы.

5. Архитектура клиент-сервер – Модель взаимодействия, где клиент запрашивает услуги, а сервер их предоставляет.

6. Виды сервисов – Веб-сервисы, микросервисы, облачные сервисы и т.д.

7. Масштабируемость – Способность системы работать при увеличении нагрузки (горизонтальная/вертикальная).

8. Протоколы передачи данных – Правила обмена данными (HTTP, TCP/IP, FTP).

9. Тонкий и толстый клиенты – Тонкий клиент минимально загружен логикой (браузер), толстый — содержит больше функций (десктопприложение).

10. Паттерн MVC: общие тезисы – Разделение приложения на Model (данные), View (отображение), Controller (логика).

11. MVC: Model-View-Presenter – Presenter mediates between View and Model, обрабатывает пользовательский ввод.

12. MVC: Model-View-View Model (MVVM) – View Model обеспечивает связь данных с View через привязки. 7

13. MVC: Model-View-Controller – Controller обрабатывает input, обновляет Model и View.

14. Docker: общие тезисы – Технология контейнеризации для изоляции и развёртывания приложений.

15. Dockerfile – Скрипт для сборки Docker-образа (инструкции: FROM, RUN, COPY и т.д.).

16. Docker Compose – Инструмент для оркестровки многоконтейнерных приложений.

17. LAMP – Стандартный стек серверного ПО: Linux, Apache, MySQL, PHP.