|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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 Голиков А.Л.

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

(подпись студента)

**Руководитель практической работы** преподаватель Благиев М.М.

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

(подпись руководителя)

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

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

Москва 2022

[Оглавление](#__RefHeading___Toc182_395009482)

[​ Цель работы 3](#__RefHeading___Toc182_395009482)

[​ Ход работы 4](#__RefHeading___Toc184_395009482)

[​ Вывод 7](#__RefHeading___Toc186_395009482)

[​ Ответы на вопросы к парктической работе 8](#__RefHeading___Toc188_395009482)

[​ Ссылка на удаленный репозиторий проекта 14](#__RefHeading___Toc190_395009482)

[​ Список использованной литературы 15](#__RefHeading___Toc192_395009482)

# 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.

Язык программирования PHP, веб-сервер Apache и СУБД MySQL

# Ход работы

Создадим файл docker-compose.yaml (рис. 1), чтобы задать параметры необходимых образов для запуска приложения.

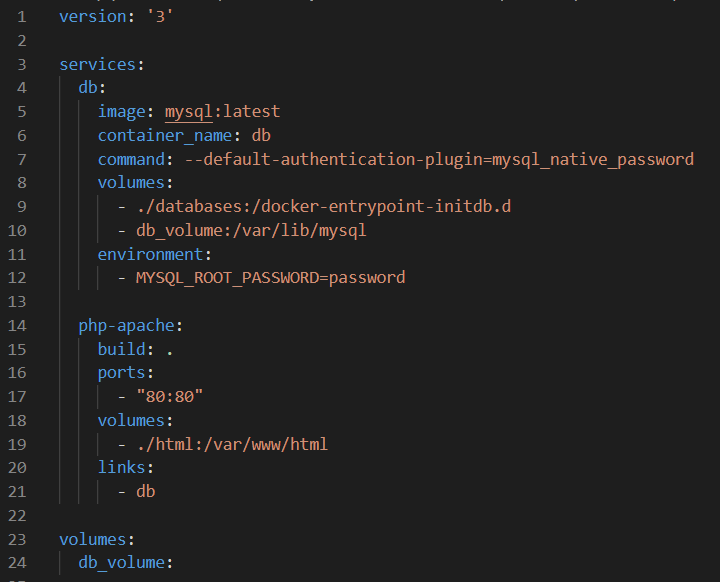


Рисунок 1 – файл docker-compose.yaml

Для добавления инструментов поддержки MySQL внутри контейнера PHP внутри каталога проекта создадим файл Dockerfile и создадим собственный образ PHP Apache и среду, которая установит mysqli и подключит PHP Apache к серверу MySQL (рис. 2).



Рисунок 2 – файл Dockerfile

Запуск контейнера представлен на рисунках 3 и 4.



Рисунок 3 – результат запуска контейнера

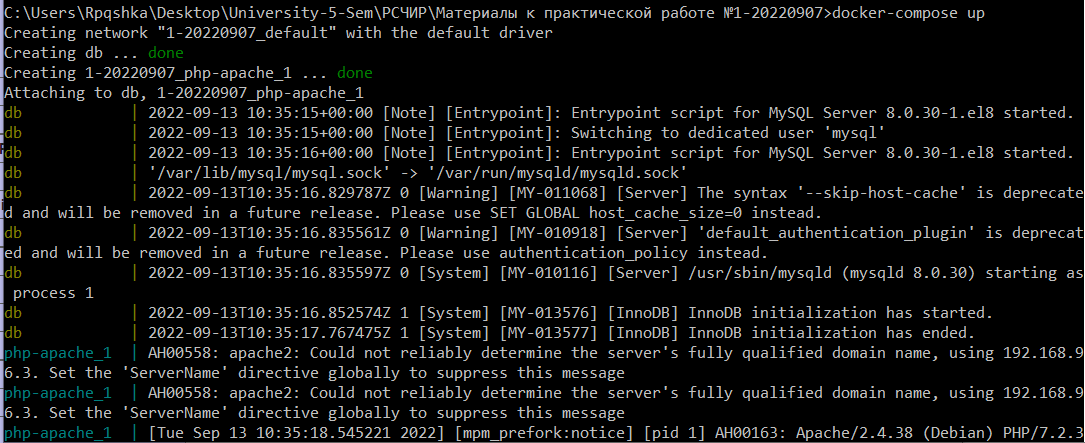


Рисунок 4 – запуск контейнеров

# Вывод

В данной практической работе была создана конфигурация серверного программного обеспечения, в которой присутствуют веб-сервер, операционная система, язык программирования и база данных.

# 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.  
 Сервер (аппаратное обеспечение) - выделенный или специализированный компьютер для выполнения сервисного программного обеспечения без непосредственного участия человека.  
 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 (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.

1. Сервис, отличия от сервера.

Сервис - легко заменяемый компонент сервисно-ориентированной  
архитектуры со стандартизированными интерфейсами - это технология для взаимодействия между системами.   
 Веб-сервис(веб-служба) - идентифицируемая уникальным веб-адресом  
(URL-адресом) программная система со стандартизированными интерфейсами — это сервер реализующий http протокол.

1. Архитектура клиент-сервер.

Данная модель — 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.

1. Виды сервисов.

Виды сервисов:

* Сервер приложений (англ. application server) — это программная  
  платформа (фреймворк), предназначенная для эффективного исполнения процедур (программ, скриптов), на которых построены приложения.
* Веб-серверы - подвид серверов приложений.
* Серверы баз данных используются для обработки запросов (СУБД).
* Файл-сервер хранит информацию в виде файлов и предоставляет пользователям доступ к ней.
* Прокси-сервер - промежуточный сервер (комплекс программ) в компьютерных сетях, выполняющий роль посредника (Веб-прокси служащий для кэширования данных. Обратный прокси - межсетевой экран).
* Файрволы (брандмауэры).
* Почтовые серверы.

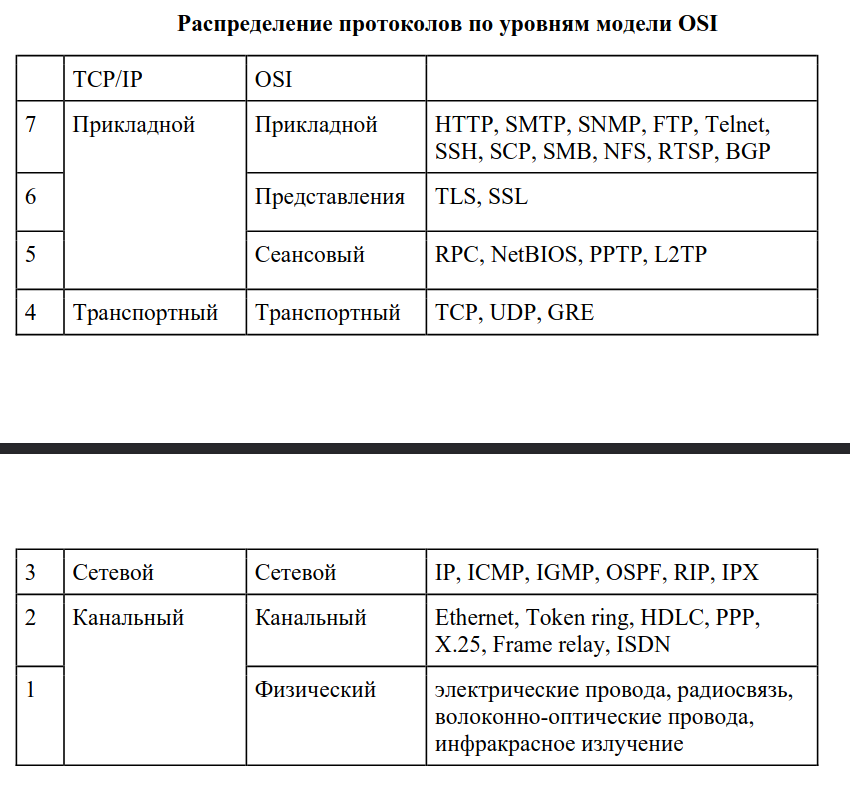
1. Масштабируемость.

Масштабируемость - способность работать с  
увеличенной нагрузкой путем наращивания ресурсов без фундаментальной  
перестройки архитектуры и/или модели реализации при добавлении ресурсов (вертикальная, горизонтальная).

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.

В общей классификации протоколы делятся на низкоуровневые,  
протоколы верхнего уровня и протоколы промежуточного уровня.

Р

1. Тонкий и толстый клиенты.

При применении толстого клиента полная функциональность приложения обеспечивается вне зависимости от сервера. Тонкий клиент же в отличие от толстого только отображает данные, принятые от сервера.

1. Паттерн MVC: общие тезисы.

Первая часть данного паттерна это модель (Model). Это представление  
содержания функциональной бизнес-логики приложения, не зависит от  
остальных частей продукта.

Представление (View) это есть отображение данных, получаемых от  
модели. Никакого влияния на модель представление оказать не может.

Третьим компонентом системы является контроллер. Данный компонент является неким буфером между моделью и представлением. Обобщенно он управляет представлением на основе изменения модели.

1. 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

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

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

1. 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.

1. Docker: общие тезисы и определения.

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.

1. Dockerfile.

Dockerfile - это текстовый файл с инструкциями, необходимыми для создания образа контейнер

1. Docker Compose.

Когда идет работа с несколькими контейнерами, то требуется механизм их объединения и оркестровки. Таким инструментом является Docker Compose. Это средство для решения задач развертывания проектов. Docker Compose используется для одновременного управления несколькими контейнерами, входящими в состав приложения.

1. 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 веб-  
  приложений (помимо PHP могут подразумеваться другие языки,  
  такие как Perl и Python).

# Ссылка на удаленный репозиторий проекта

Ссылка на GitHUB - https://github.com/Rpqshka/University-5-Sem/tree/main/РСЧИР/Материалы%20к%20практической%20работе%20№1-20220907

# Список использованной литературы

1. Статья о назначении докера простыми словами: <https://habr.com/ru/post/309556>/
2. Официальная документация докера: https://docs.docker.com/
3. Туториал по докеру: https://badcode.ru/docker-tutorial-dlia-novichkov-  
   rassmatrivaiem-docker-tak-iesli-by-on-byl-ighrovoi-pristavkoi/
4. Туториал по докеру с Хабра: https://habr.com/ru/post/310460/
5. Шпаргалка с командами Docker:  
   https://habr.com/ru/company/flant/blog/336654/