



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»

**Студент группы ИКБО-02-20**

**Тарарина Ольга  
Владимировна**

---

(подпись студента)

**Руководитель практической работы**

**преподаватель Благирев М.М.**

---

(подпись руководителя)

Работа представлена

«\_\_\_»\_\_\_\_\_ 2022 г.

Допущен к работе

«\_\_\_»\_\_\_\_\_ 2022 г.

Москва 2022

## СОДЕРЖАНИЕ

1. Цель работы .....	3
2. Ход работы.....	3
3. Ответы на вопросы к практической работе.....	6
3.1. Сервер и клиент.....	6
3.2. База данных.....	6
3.3. API. ....	6
3.4. Сервис, отличия от сервера.....	6
3.5. Архитектура клиент-сервер. ....	7
3.6. Виды сервисов.....	7
3.7. Масштабируемость. ....	8
3.8. Протоколы передачи данных.....	9
3.9. Тонкий и толстый клиенты. ....	9
3.10. Паттерн MVC: общие тезисы. ....	10
3.11. Паттерн MVC: Model-View-Presenter. ....	10
3.12. Паттерн MVC: Model-View-View Model. ....	10
3.13. Паттерн MVC: Model-View-Controller.....	11
3.14. Docker: общие тезисы и определения. ....	11
3.15. Dockerfile.....	12
3.16. Docker Compose.....	12
3.17. LAMP.....	12
4. Ссылка на удаленный репозиторий проекта .....	13
ЗАКЛЮЧЕНИЕ .....	13
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ .....	13

## 1. Цель работы

Предлагается создать свою конфигурацию серверного программного обеспечения, в которой должны присутствовать веб-сервер, операционная система, язык программирования и база данных.

Для проверки работоспособности 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.

## 2. Ход работы

Для облегчения работы с рекомендуемыми инструментами используются предоставленные скрипт инициализации БД для СУБД MySQL и скрипт генерации тестовой страницы вместе с оформлением на языке PHP. Данные файлы были помещены в папку “php” и создан файл “Dockerfile” (рис. 2.1).

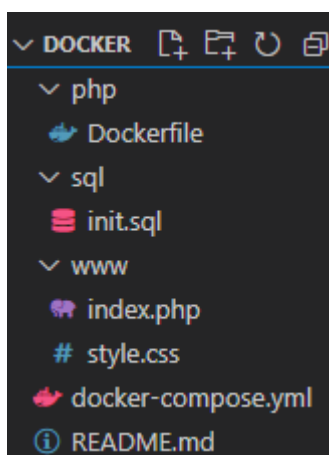


Рисунок 2.1 – Папка “php”

Dockerfile был настроен для создания образа запуска базового php проекта с поддержкой MySQL (рис 2.2).

```
php > Dockerfile > ...
1 FROM php:7.2-apache
2 RUN apt-get update && docker-php-ext-install mysqli
```

Рисунок 2.2 – Файл Dockerfile

Для совместной работы контейнера сервера и контейнера базы данных был создан файл “docker-compose.yml” (рис 2.3). Для контейнера базы данных используется готовый образ MariaDB.


```
docker-compose.yml
1 version: '3' # версия Docker
2
3 services: # Контейнеры, которые запускаем
4
5     php: # Имя контейнера
6         build: # Процесс билдинга
7             ./php
8         ports: # Порты устройство:docker
9             - 9000:80
10        volumes: # Связываем папку с устройства с папкой на виртуальной машине
11            - ./www:/var/www/html
12        depends_on: # Зависимость, после чего запускаем
13            - datab
14
15        datab: # Имя контейнера
16            image: mariadb:latest # Образ бд
17            restart: always # Поведение при крахе
18            volumes: # Связываем папку с устройства с папкой на виртуальной машине
19                - "/sql:/docker-entrypoint-initdb.d"
20            environment:
21                MARIADB_ROOT_PASSWORD: password
```

Рисунок 2.3 – Файл docker-compose.yml

С помощью команды “docker-compose up --build” можно создать и запустить контейнеры. Затем можно проверить работоспособность сервера и базы данных перейдя по ссылке “localhost:9000” (рис 2.4 - 2.5).

Таблица пользователей данного продукта		
Id	Name	Surname
1	Alex	Rover
2	Bob	Marley
3	Kate	Yandson
4	Lilo	Black

PHP Version 7.2.34



System	Linux e44c808aa484 5.10.124-linuxkit #1 SMP Thu Jun 30 08:19:10 UTC 2022 x86_64
Build Date	Dec 11 2020 10:50:00
Configure Command	./configure '--build=x86_64-linux-gnu' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/etc/php/conf.d' '--enable-option-checking=fatal' '--with-mhash' '--with-pic' '--enable-ftp' '--enable-mbstring' '--enable-mysqlnd' '--with-password-argon2' '--with-sodium=shared' '--with-pdo-sqlite=/usr' '--with-sqlite3=/usr' '--with-curl' '--with-libedit' '--with-openssl' '--with-zlib' '--with-libdir=lib/x86_64-linux-gnu' '--with-apxs2' '--disable-cgi' 'build_alias=x86_64-linux-gnu'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	(none)
Scan this dir for additional .ini files	/usr/local/etc/php/conf.d
Additional .ini files parsed	/usr/local/etc/php/conf.d/docker-php-ext-mysqli.ini /usr/local/etc/php/conf.d/docker-php-ext-sodium.ini

Рисунок 2.4 – Работа сервера и базы данных

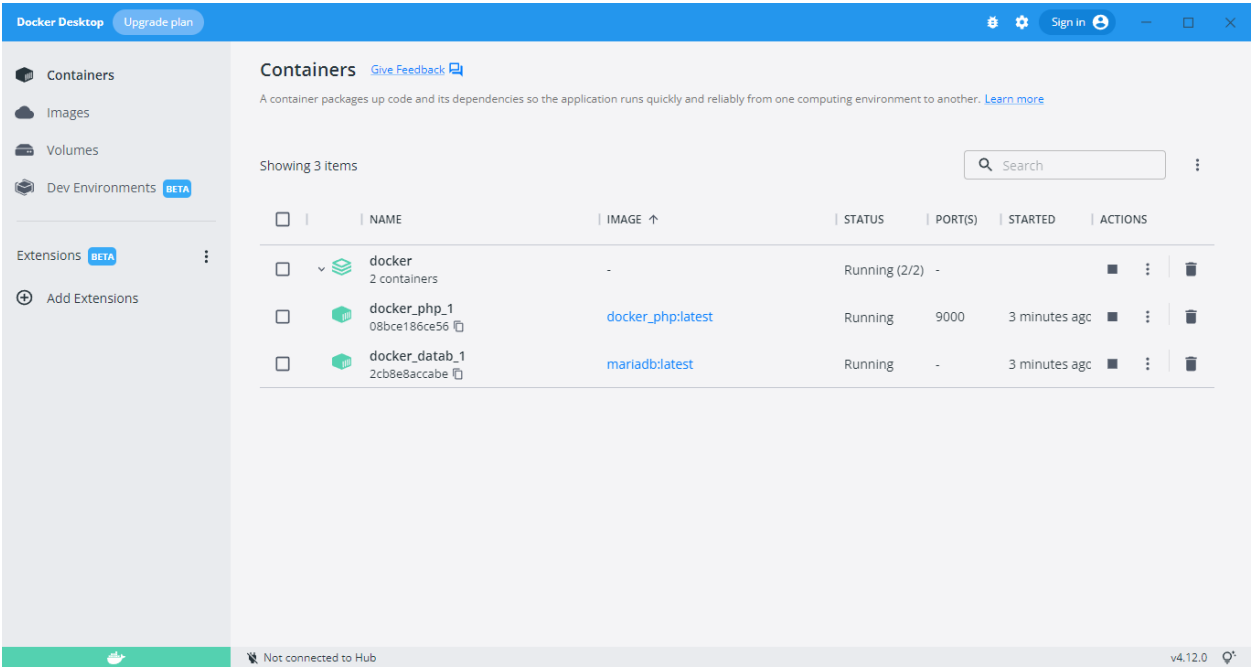


Рисунок 2.5 – Работа сервера и базы данных

### **3. Ответы на вопросы к практической работе**

#### **3.1. 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.

#### **3.2. База данных.**

База данных — это информационная модель, позволяющая упорядоченно хранить данные об объекте или группе объектов, обладающих набором свойств, которые можно категоризировать. Базы данных функционируют под управлением систем управления базами данных (сокращенно СУБД).

#### **3.3. 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.

#### **3.4. Сервис, отличия от сервера.**

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

Веб-сервис (веб-служба) - идентифицируемая уникальным веб-адресом (URL-адресом) программная система со стандартизированными интерфейсами.

### **3.5. Архитектура клиент-сервер.**

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. Но нужно понимать, что в этом случае очень важно разделение обязанностей и уровней доступа к данным на стороне клиента

### **3.6. Виды сервисов.**

- Сервер приложений (англ. application server) — это программная платформа (фреймворк), предназначенная для эффективного исполнения процедур (программ, скриптов), на которых построены приложения.
- Веб-серверы. Являются подвидом серверов приложений. Изначально предоставляли доступ к гипертекстовым документам по протоколу HTTP. Сейчас поддерживают расширенные возможности, в частности, передачу произвольных данных.
- Серверы баз данных. Серверы баз данных используются для обработки запросов. На сервере находится СУБД для управления БД и ответов на запросы.
- Файл-серверы. Файл-сервер хранит информацию в виде файлов и предоставляет пользователям доступ к ней. Как правило, файл-сервер обеспечивает и определенный уровень защиты от несанкционированного доступа
- Прокси-сервер. Прокси-сервер (от англ. proxy - представитель, уполномоченный; часто просто прокси, сервер-посредник) - промежуточный

сервер (комплекс программ) в компьютерных сетях, выполняющий роль посредника. Существует несколько видов прокси-серверов:

- Веб-прокси — широкий класс прокси-серверов, служащий для кэширования данных.
- Обратный прокси — прокси-сервер, который, в отличие от веб-прокси, ретранслирует запросы клиентов из внешней сети на один или несколько серверов, логически расположенных во внутренней сети.
- Файрволы (брандмауэры). Межсетевые экраны, анализирующие и фильтрующие проходящий сетевой трафик, с целью обеспечения безопасности сети.
- Почтовые серверы. Предоставляют услуги по отправке и получению электронных почтовых сообщений.

### **3.7. Масштабируемость.**

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.

- Вертикальная масштабируемость (англ. scaling up) представляет собой увеличение производительности компонентов серверной системы в интересах повышения производительности всей системы. Данный метод не снимает нагрузку на всю систему, но является самым простым. Ярким примером является увеличение оперативной памяти, установка более мощного процессора.
- Горизонтальная масштабируемость (англ. scaling out) представляет собой как разбиение системы на более мелкие структурные компоненты и разнесение их, так и увеличение количества компонентов,



параллельно выполняющих одну и ту же функцию. Частым примером является добавление еще одного сервера тех же характеристик к существующему.

### **3.8. 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.

Наиболее известные прикладные протоколы, используемые в сети Интернет:

- Протокол RTP (Real-time Transport Protocol), протокол работает на прикладном уровне (OSI - 7) и используется при передаче трафика реального времени.
- HTTP (Hyper Text Transfer Protocol) — это протокол передачи гипертекста.
- FTP (File Transfer Protocol) — это протокол передачи файлов со специального файлового сервера на компьютер пользователя.
- POP3 (Post Office Protocol) — это стандартный протокол почтового соединения.
- SMTP (Simple Mail Transfer Protocol) — протокол, который задает набор правил для передачи почты.
- TELNET — это протокол удаленного доступа

### **3.9. Тонкий и толстый клиенты.**

- Толстый клиент

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

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. К сожалению, любой сбой на сервере и в канале связи влечет “падение” всего приложения

### **3.10. Паттерн MVC: общие тезисы.**

Название паттерну дают первые буквы его основных компонентов: Model View Controller.

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 об 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. Если рассматривать интернет-ресурсы представлением является html-страница.

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

### **3.11. 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.

### **3.12. Паттерн MVC: Model-View-View Model.**

Особенностью паттерна Model-View-View Model является связывание элементов представления со свойствами и событиями View-модели.

### **3.13. 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.

### **3.14. 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 hostsистеме. Подход, используемый Docker, находится посередине между запуском всего на физическом сервере и полной виртуализацией, предлагаемой виртуальными машинами. Этот подход называется контейнеризацией.

Основными компонентами докера является:

- `docker daemon` — сердце докера. Это демон, работающий на хост-машине, и умеющий сохранять с удалённого репозитория и загружать на него образы, запускать из них контейнеры, следить за запущенными контейнерами, собирать логи и настраивать сеть между контейнерами (а с версии 0.8 и между машинами). А еще именно демон создает образы контейнеров, хоть и может показаться, что это делает `docker-client`.
- `docker` — это консольная утилита для управления `docker-демоном` по HTTP. Она устроена очень просто и работает предельно быстро. Вопреки заблуждению управлять демоном докера можно откуда угодно, а не только с той же машины. В сборке нового образа консольная утилита `docker` принимает пассивное участие: архивирует локальную папку в `tar.gz` и передает по сети `docker-daemon`, который и делает всю работу. Именно из-за передачи контекста демону по сети, лучше собирать тяжелые образы локально.

- docker Hub централизованно хранит образы контейнеров. Когда вы пишете “`docker run ruby`”, docker скачивает самый свежий образ с ruby именно из публичного репозитория. Изначально хаба не было, его добавили уже после очевидного успеха первых двух частей.
- docker registry предназначен для хранения и дистрибуции готовых образов.

### **3.15. 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.

### **3.16. Docker Compose.**

Это средство для решения задач развертывания проектов. Docker Compose 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

### **3.17. LAMP**

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).

#### 4. Ссылка на удаленный репозиторий проекта

Полный код проекта можно найти по ссылке:

<https://github.com/01ga2001/docker>

### ЗАКЛЮЧЕНИЕ

Была создана конфигурация серверного программного обеспечения на наборе LAMP и проверена ее работоспособность. Данная конфигурация будет использоваться для выполнения следующих практических работ по данной дисциплине и для выполнения курсового проектирования.

### СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. «Конспект лекции №1»: [Электронный ресурс]. URL: <https://online-edu.mirea.ru/mod/resource/view.php?id=403993>

Дата обращения 05.09.2022

2. «Руководство по Docker Compose»: [Электронный ресурс]. URL: <https://habr.com/ru/company/ruvds/blog/450312/>

Дата обращения 06.09.2022

3. «Docker самый простой и понятный tutorial»: [Электронный ресурс]. URL: [https://badtry.net/docker-tutorial-dlia-novichkov-rassmatrivaem-docker-tak-iesli-by-on-byl-ighrovoi-pristavkoi/#what\\_is\\_docker\\_container](https://badtry.net/docker-tutorial-dlia-novichkov-rassmatrivaem-docker-tak-iesli-by-on-byl-ighrovoi-pristavkoi/#what_is_docker_container)

Дата обращения: 05.09.2022

4. «Шпаргалка с командами Docker»: [Электронный ресурс]. URL: <https://habr.com/ru/company/flant/blog/336654/>

Дата обращения: 06.09.2022

5. «Docker. Как и зачем»: [Электронный ресурс]. URL: <https://habr.com/ru/post/309556/>

Дата обращения: 05.09.2022