



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-30-20

**Тайибов Чингиз
Гусейнович**

(подпись студента)

Руководитель практической работы

преподаватель Волков М.Ю.

(подпись руководителя)

Работа представлена

«___» _____ 2022 г.

Допущен к работе

«___» _____ 2022 г.

Москва 2022

СОДЕРЖАНИЕ

Постановка задачи.....	1
Выполнение работы	1
Ссылка на удаленный репозиторий	4
Ответы на вопросы к практической:	4
ВЫВОДЫ.....	9
СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ	9

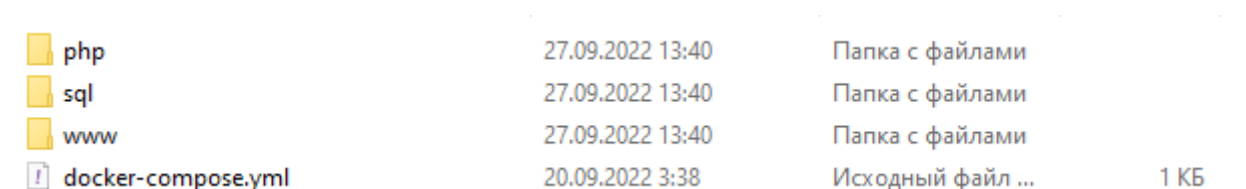
Постановка задачи

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

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.

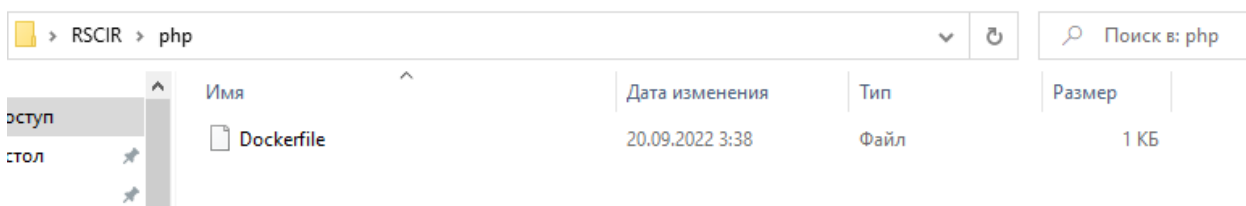
Выполнение работы

Структура проекта показана на следующих рисунках 1-4:



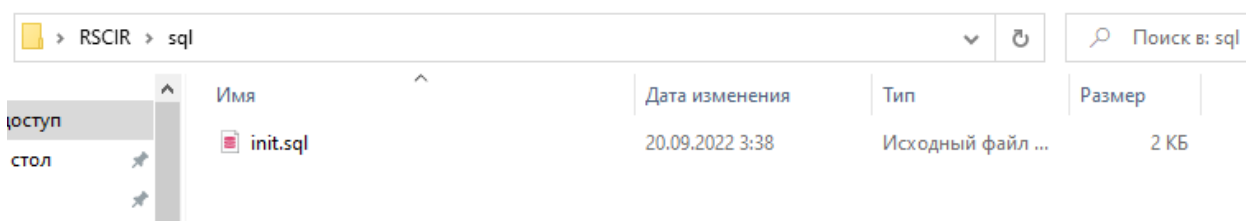
php	27.09.2022 13:40	Папка с файлами	
sql	27.09.2022 13:40	Папка с файлами	
www	27.09.2022 13:40	Папка с файлами	
docker-compose.yml	20.09.2022 3:38	Исходный файл ...	1 КБ

Рисунок 1 – Структура проекта.



RSCIR > php				Поиск в: php
Имя	Дата изменения	Тип	Размер	
Dockerfile	20.09.2022 3:38	Файл	1 КБ	

Рисунок 2 – Содержимое папки php.



RSCIR > sql				Поиск в: sql
Имя	Дата изменения	Тип	Размер	
init.sql	20.09.2022 3:38	Исходный файл ...	2 КБ	

Рисунок 3 – Содержимое папки sql.

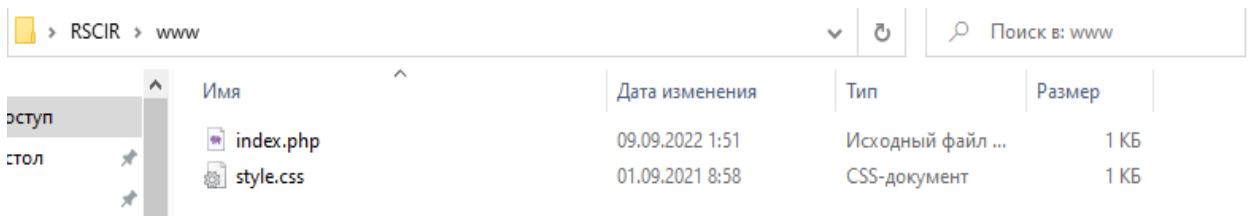


Рисунок 4 – Содержимое папки www.

В директории serv присутствует Dockerfile, необходимый для загрузки образа php-apache и установки нужных расширений (Рисунок 5).

Листинг 1 – Dockerfile

```
1 FROM php:7.4-apache
2 RUN apt-get update && docker-php-ext-install mysqli
```

Исправим стандартный файл index.php. В итоге содержимое файла будет выглядеть следующим образом (Листинг 2).

Листинг 2 –index.php

```
1 <html lang="en">
2 <head>
3 <title>Hello world page</title>
4 <link rel="stylesheet" href="style.css" type="text/css"/>
5 </head>
6 <body>
7 <h1>Таблица пользователей данного продукта</h1>
8 <table>
9 <tr><th>Id</th><th>Name</th><th>Surname</th></tr>
10 <?php
11 $mysqli = new mysqli("localhost", "user", "password", "appDB");
12 $result = $mysqli->query("SELECT * FROM users");
13 foreach ($result as $row){
14     echo "<tr><td>{$row['ID']}</td><td>{$row['name']}</td><td>{$row['surname']}</td></tr>";
15 }
16 ?>
17 </table>
18 <?php
19 phpinfo();
20 ?>
21 </body>
22 </html>
```

Напишем docker-compose.yml. Содержимое файла представлено в листинге 3.

Листинг 3 – docker-compose.yml

```

1  version: '3'
2
3  services:
4    php:
5      build:
6        ./php
7      ports:
8        - 8080:80
9      volumes:
10       - ./www:/var/www/html
11      depends_on:
12        - data
13    data:
14      image: mysql
15      container_name: DB
16      volumes:
17        - " ./sql/init.sql:/docker-entrypoint-initdb.d/1.sql"
18      environment:
19        MYSQL_ROOT_PASSWORD: password

```

Переходим в рабочую директорию и с помощью команды docker-compose up собираем, создаем и запускаем контейнеры (рис.2).

```

C:\Users\Ching\OneDrive\Рабочий стол\RSCIR>docker-compose up
[+] Running 3/3
- Network rscir_default      Created                                0.0s
- Container DB               Created                                0.1s
- Container rscir-serv-1     Created                                0.1s
Attaching to DB, rscir-serv-1
DB | 2022-09-27 10:32:24+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.30-1.el8 started.
DB | 2022-09-27 10:32:24+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
DB | 2022-09-27 10:32:24+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.30-1.el8 started.
rscir-serv-1 | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.18.0.3. Set the 'ServerName' direct
rscir-serv-1 | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.18.0.3. Set the 'ServerName' direct
DB | 2022-09-27 10:32:24+00:00 [Note] [Entrypoint]: Initializing database files
DB | 2022-09-27T10:32:24.932100Z 0 [Warning] [MY-011068] [Server] The syntax '--skip-host-cache' is deprecated and will be removed in a fut
ure release. Please use SET GLOBAL host_cache_size=0 instead.
DB | 2022-09-27T10:32:24.933254Z 0 [System] [MY-013169] [Server] /usr/sbin/mysqld (mysqld 8.0.30) initializing of server in progress as pro
cess 80
DB | 2022-09-27T10:32:24.955521Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
rscir-serv-1 | [Tue Sep 27 10:32:25.005373 2022] [mpm_prefork:notice] [pid 1] AH00163: Apache/2.4.54 (Debian) PHP/8.0.23 configured -- resuming norma
l operations
rscir-serv-1 | [Tue Sep 27 10:32:25.005494 2022] [core:notice] [pid 1] AH00094: Command line: 'apache2 -D FOREGROUND'
DB | 2022-09-27T10:32:25.820815Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
DB | 2022-09-27T10:32:27.590680Z 6 [Warning] [MY-010453] [Server] root@localhost is created with an empty password ! Please consider switch
ing off the --initialize-insecure option.
DB | 2022-09-27 10:32:32+00:00 [Note] [Entrypoint]: Database files initialized
DB | 2022-09-27 10:32:32+00:00 [Note] [Entrypoint]: Starting temporary server
DB | 2022-09-27T10:32:32.816215Z 0 [Warning] [MY-011068] [Server] The syntax '--skip-host-cache' is deprecated and will be removed in a fut
ure release. Please use SET GLOBAL host_cache_size=0 instead.
DB | 2022-09-27T10:32:32.817353Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.30) starting as process 129

```

Рисунок 1 - Запуск контейнеров.

Перейдем по адресу localhost:8080 и убедимся в работоспособности веб-сервера, отображающего выборку из базы данных и информационное сообщение о версии языка программирования, его настройках и конфигурации (рис.3).

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


PHP Version 8.0.23	
	
System	Linux 3c48edc31852 5.10.16.3-microsoft-standard-WSL2 #1 SMP Fri Apr 2 22:23:49 UTC 2021 x86_64
Build Date	Sep 1 2022 22:05:43
Build System	Linux e3ccb21887c9 5.10.0-13-cloud-amd64 #1 SMP Debian 5.10.106-1 (2022-03-17) x86_64 GNU/Linux
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-iconv' '--with-openssl' '--with-readline' '--with-zlib' '--disable-phpdbg' '--with-pear' '--with-libdir=lib/x86_64-linux-gnu' '--disable-cgi' '--with-apxs2' '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
PHP API	20200930
PHP Extension	zendengine

Рисунок 2— Проверка конфигурации

Ссылка на удаленный репозиторий

<https://github.com/ChiNiz/RSCIR/tree/main/RSCIRpr1>

Ответы на вопросы к практической:

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.

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. API.

API (ApplicationProgrammingInterface - 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.

4. Сервис, отличия от сервера.

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

Сервер находится уровнем выше, чем сервис. Любой полученный запрос прежде, чем попадёт в сервис, проходит через сервер. И уже сервер передаёт запрос сервису. И сервис отдаёт ответ именно серверу. А уже сервер отправляет ответ клиенту.

5. Архитектура клиент-сервер.

Данная модель — это идея разделения системы или приложения на отдельные задачи, размещаемые на различных платформах для большей эффективности. Уже применение данной идеи лежит в основе архитектуры клиент-сервер, распределенных вычислений, архитектуры приложений и т.д.

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.

6. Виды сервисов.

Существует великое множество возможных сервисов, как самостоятельных, так и в составе приложений. Рассмотрим возможные виды сервисов:

- Серверы приложений;
- Веб-серверы;
- Серверы баз данных;
- Файл-серверы;
- Прокси-серверы;
- Файрволы (брандмауэры);
- Почтовые серверы.

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. Основными являются горизонтальная и вертикальная масштабируемость.

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.

9. Тонкий и толстый клиенты.

При классификации компонентов архитектуры клиент-сервер существует понятия “толстый” и “тонкий” клиент. При применении толстого клиента полная функциональность приложения обеспечивается вне зависимости от сервера. 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. Тонким клиентом называют компьютеры и программы, функционирующие в терминальной или серверной сети. Множество задач по обработке данных осуществляются на главных компьютерах, к которым присоединено приложение и компьютер. Тонкий клиент же в отличие от толстого только отображает данные, принятые от сервера.

10. Паттерн MVC: общие тезисы.

Первая часть данного паттерна это модель (Model). Это представление содержания функциональной бизнес-логики приложения.

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.

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.

12. Паттерн MVC: Model-View-View Model.

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

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.

14. Docker: общие тезисы и определения.

Существует проблема разработки того или иного приложения и его развертывания на других машинах. Самыми частыми решениями данной проблемы являются установочные скрипты, облачные сервисы и виртуальные машины. Описанные подходы не являются оптимальными что раздувает техническую поддержку до максимума, а также медленны и тяжеловесны. Одним из вариантов решения данной задачи является докер, который представляет технологию контейнеризации.

15. Dockerfile.

Часто возникает ситуация, когда конфигурации уже существующего не хватает. Чтобы создавать свои собственные образы нужен специальный скрипт. Образы наследуются и, обычно, для создания своего первого образа мы берём готовый образ и наследуемся от него. Чтобы запустить скрипт он должен иметь имя `Dockerfile` и не должен иметь типа.

Чтобы собрать новый образ, нужно, в папке, где находится `Dockerfile`, выполнить команду: `dockerbuild -tmy_image .`. Тег `-t` задает название образа или тег, а точка обозначает работу в текущей директории.

16. DockerCompose.

Когда идет работа с несколькими контейнерами, то требуется механизм их объединения и оркестровки. Таким инструментом является `DockerCompose`. Это средство для решения задач развертывания проектов. `DockerCompose` 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.

17. 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, Apache, MySQL, PHP.

ВЫВОДЫ

В процессе выполнения практической работы мы создали свою конфигурацию серверного программного обеспечения, в которой присутствуют веб-сервер, операционная система, язык программирования и база данных. Для проверки работоспособности конфигурации мы сгенерировали тестовую страничку с необходимыми данными.

СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Установка и настройка PHP URL:
<https://www.php.net/manual/ru/install.php> (Дата обращения 06.09.2022)
2. Официальная документация докера: URL: <https://docs.docker.com/>
(Дата обращения 06.09.2022)
3. Статья о назначении докера простыми словами. URL:
<https://habr.com/ru/post/309556/> (Дата обращения 06.09.2022)
4. Настройка связки Apache + PHP + MySQL + phpMyAdmin URL:
<https://puzzleweb.ru/other/apache.php> (Дата обращения 06.09.2022)
5. Создание базы данных в MySQL URL:
<https://selectel.ru/blog/tutorials/how-to-create-databases-in-mysql/> (Дата обращения 06.09.2022)