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 (ИиППО)**

# ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ

по дисциплине «Разработка серверных частей интернет ресурсов»

## Практическое задание № 3

Студента группы *ИКБО-24-20 Постнов Никита*

(подпись)

Старший преподаватель Волков М. Ю.

(подпись)

Отчет представлен «___»_____2022 г.

Москва 2022 г

**СОДЕРЖАНИЕ**

## 1. Цель работы

Вариант 6 - библиотека

В задании предлагается создать сложную серверную конфигурацию, состоящую из связки apache+nginx+php+База данных. Возможно использование связки apache+php как единый компонент. В данной конфигурации предполагается создание как минимум 3 элементов(контейнеров) или использование как основы серверной конфигурации, созданной в практической работе No1. В этой конфигурации предполагается акселерированное проксирование без кэширования.

## 2. Ход работы

Создам файл docker-compose в котором будет описана логика контейнеризации приложения(рис. 1).

```yaml
version: '3'
services:
  db:
    image: mysql:latest
    container_name: db
    command: --default-authentication-plugin=mysql_native_password
    volumes:
      - ./databases:/docker-entrypoint-initdb.d
      - db_volume:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: password
      HOSTNAME: db
    healthcheck:
      test: [ "CMD", "mysqladmin" ,"ping", "-h", "localhost" ]
      timeout: 1s
      retries: 10
  apache:
    build: './php-apache'
    volumes:
      - ./public_html:/home/public_html
      - ./conf/8000-default.conf:/etc/apache2/sites-enabled/000-default.conf
    # env_file: ./.env
    depends_on:
      db:
        condition: service_healthy
  nginx:
    build: './conf/nginx'
    volumes:
      - ./conf/nginx/conf/default.conf:/etc/nginx/nginx.conf
      - ./nginx-public_html:/home/public_html
    ports:
      - 8000:80
    depends_on:
      - apache

volumes:
  db_volume:
```

Рисунок 1 – Код docker-compose

В конфигурации nginx укажем файлы которые должны обрабатываться на стороне nginx, а какие на стороне apache.

```
worker_processes  1;

events {
    worker_connections  1024;
}

http {
    include       mime.types;
    default_type  application/octet-stream;
    sendfile        on;
    keepalive_timeout  65;

    server {
        listen       80;
        server_name  localhost;

        location ~ \.(jpg|jpeg|gif|png|ico|css|zip|tgz|gz|rar|bz2|doc|xls|exe|pdf|ppt|t
            root /var/www/apache-server/html;
        }

        location ~ /\.ht {
            deny  all;
        }

        location / {
            proxy_pass http://apache;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $remote_addr;
            proxy_connect_timeout 120;
            proxy_send_timeout 120;
            proxy_read_timeout 180;
        }
    }
}
```

Рисунок 2 – конфигурация nginx

В файле 8000-default.conf будет реализована авторизация пользователя используя созданную базу данных (Рисунок 3)

```
    MaxRequestsPerChild   0
</IfModule>

<IfModule mod_authn_dbd.c>
DBDriver mysql
DBDParams "host=db,port=3306,user=user,pass=password,dbname=appDB"
DBDMin  2
DBDKeep 8
DBDMax  20
DBDExptime 300
</IfModule>

# конфигурация для конкретной папки
<Directory "/home/public_html">
    # не требуется авторизация
    AuthType None
    Require all granted
</Directory>

<Directory "/home/public_html/admin/">
    AuthType Basic
    AuthName "login"

    # To cache credentials, put socache ahead of dbd here
    AuthBasicProvider dbd

    # Also required for caching: tell the cache to cache dbd lookups!

 # mod_authz_core configuration
    Require valid-user
    AuthDBDUserPWQuery "SELECT password FROM users WHERE name = %s"
    Session On
    SessionCookieName session path=/
#     неавторизованный вход
    ErrorDocument 401 /public/account-login.php
</Directory>
```

Рисунок 3 – Конфигурация apache

В файле init.sql создадим таблицу пользователей а также таблицу книг в библиотеке согласно варианту

```sql
INSERT INTO users (name, password)
SELECT * FROM (SELECT 'Kate', 'Yandson') AS tmp
WHERE NOT EXISTS (
    SELECT name FROM users WHERE name = 'Kate' AND password = 'Yandson'
) LIMIT 1;

INSERT INTO users (name, password)
SELECT * FROM (SELECT 'Lilo', 'Black') AS tmp
WHERE NOT EXISTS (
    SELECT name FROM users WHERE name = 'Lilo' AND password = 'Black'
) LIMIT 1;


USE appDB;
CREATE TABLE IF NOT EXISTS books (
  ID INT(11) NOT NULL AUTO_INCREMENT,
  title VARCHAR(20) NOT NULL,
  author VARCHAR(40) NOT NULL,
  taken tinyint not NULL,
  PRIMARY KEY (ID)
);


INSERT INTO books (title, author,taken)
SELECT * FROM (SELECT '1984', 'Some Author',0) AS tmp
WHERE NOT EXISTS (
    SELECT title FROM books WHERE title = '1984' AND author = 'Some Author'
) LIMIT 1;
```

Рисунок 4 – Пример создания базы данных

В файле index.php (Рисунок 5) реализуем логику библиотеки, в которой пользователь может узнать наличие книги по названию.

```
<html lang="ru">
<head>
<title>Hello world page</title>
    <link rel="stylesheet" href="style.css" type="text/css"/>
</head>
<body>
<h1>Доступность книги</h1>
<?php
$ar = htmlspecialchars($_GET["title"]);
$mysqli = new mysqli("db", "user", "password", "appDB");
$result = $mysqli->query("SELECT * FROM books");
$t = 0 ;
foreach ($result as $row){
    if ($ar == $row['title']){
        $t = $t+1;
        $tk =  $row['taken'];
        if ($tk == 1){
            echo "OOPS this book is already taken";
            break;
        }
        else {
            echo "Come to us! You can take this book";
            break;
        }
    }

}
if ($t == 0){
    echo "Not found :(";
}
?>
</body>
</html>
```

Рисунок 5 – Файл index.php

Пример работы представленного кода (Рисунок 6)

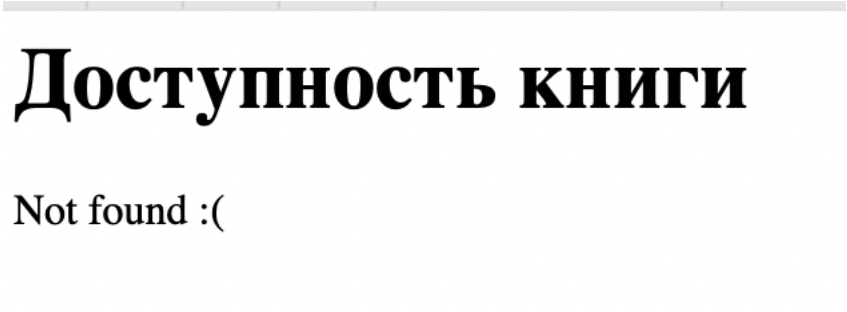


Рисунок 6 – Пример работы index.php

Файл Dockerfile содержит установку требуюмых для выполнения упражнений библиотек (Рисунок 7).

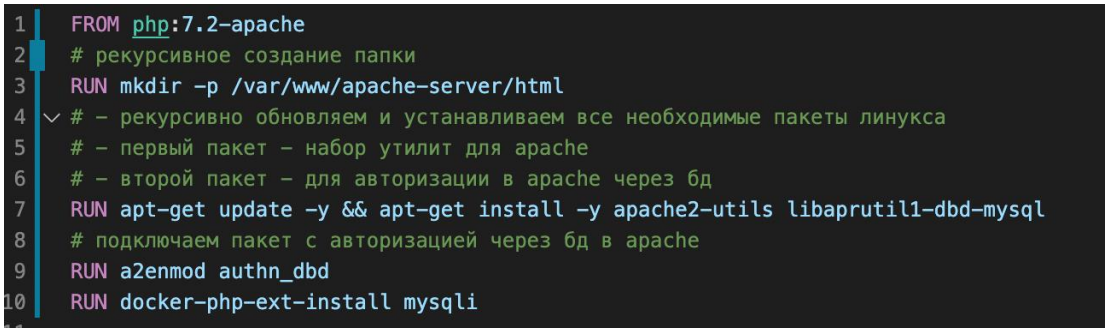


Рисунок 7 – Файл Dockerfile

## 3. Выводы

В ходе работы была создана конфигурация серверного программного обеспечения, состоящего из:

1. Веб-сервера Apache;
2. Сервера Nginx;
3. СУБД MySQL;
4. Языка программирования PHP.

А также были приобретены знания в области создания авторизации с помощью модуля .htaccess

## 5. Ответы на вопросы к практической работе

## 5. Ссылка на удалённый репозиторий проекта

https://github.com/1TSOP/RSCHIR_3PR.git

## 6. Список использованных источников


1. Видео “Введение в Докер” на английском языке от создателя: Introduction to Docker (https://www.youtube.com/watch?v=Q5POuMHxW-0)
2. Статья о назначении докера простыми словами: https://habr.com/ru/post/309556/
3. Более сложная и подробная статья про докер: https://habr.com/ru/post/277699/