#### Министерство науки и высшего образования Российской Федерации

#### Федеральное государственное бюджетное образовательное учреждение

#### высшего образования

#### «Владимирский государственный университет

#### имени Александра Григорьевича и Николая Григорьевича Столетовых»

**(ВлГУ)**

**Кафедра информационных систем и программной инженерии**

Лабораторная работа №6

по дисциплине

"Основы разработки веб-приложений"

Выполнил:

ст. гр. ПРИ-117

Подолец А.А.

Принял:

Козаев А.

Владимир, 2020 г.

## **Цель работы**

Познакомиться со средством контейнеризации Docker и использовать его для запуска приложения на Laravel.

**Ход работы**

Вариант 13 (1)

Материалы находятся: <https://github.com/taki-jeden-karol/ORWP>

В папке прошлой лабораторной работы создаем файл Dockerfile:

FROM php:7.4.5-fpm

RUN apt-get update \

&& docker-php-ext-install pdo mysqli pdo\_mysql && \

pecl install -o -f redis \

&& rm -rf /tmp/pear \

&& docker-php-ext-enable redis

WORKDIR /var/www

COPY . /var/www

EXPOSE 9000

CMD ["php-fpm"]

Так же добавляем файл docker-compose.yml для работы с бд:

version: "3.2"

services:

nginx:

image: nginx:alpine

volumes:

- .:/var/www

- ./nginx/conf.d/:/etc/nginx/conf.d/

ports:

- 81:81

- 443:443

depends\_on:

- php

php:

build:

context: .

volumes:

- .:/var/www

ports:

- 9000:9000

depends\_on:

- db

- redis

db:

image: mysql:8.0

ports:

- 3307:3306

environment:

MYSQL\_DATABASE: lab\_laravel

MYSQL\_ROOT\_PASSWORD: 12232

redis:

image: redis:4.0

ports:

- 6379:6379

Создаем файл default.conf по пути nginx/conf.d/:

server {

listen 81;

index index.php product.plade.php;

error\_log /var/log/nginx/error.log;

access\_log /var/log/nginx/access.log;

root /var/www/public;

location ~ \.php$ {

try\_files $uri =404;

fastcgi\_split\_path\_info ^(.+\.php)(/.+)$;

fastcgi\_pass php:9000;

fastcgi\_index index.php;

include fastcgi\_params;

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

fastcgi\_param PATH\_INFO $fastcgi\_path\_info;

}

location / {

try\_files $uri $uri/ /index.php?$query\_string;

gzip\_static on;

}

}

В файле .env меняем подключение к бд, меняем redis\_host и cache\_driver:

APP\_NAME=Laravel

APP\_ENV=local

APP\_KEY=base64:IPl6ogB09V9CM6PcADF3Cj5Fzuve+nPDtXAS1C6etB4=

APP\_DEBUG=true

APP\_URL=http://localhost

LOG\_CHANNEL=stack

DB\_CONNECTION=mysql

DB\_HOST=db

DB\_PORT=3306

DB\_DATABASE=lab\_laravel

DB\_USERNAME=root

DB\_PASSWORD=12232

BROADCAST\_DRIVER=log

CACHE\_DRIVER=redis

QUEUE\_CONNECTION=sync

SESSION\_DRIVER=file

SESSION\_LIFETIME=120

REDIS\_HOST=redis

REDIS\_PASSWORD=null

REDIS\_PORT=6379

MAIL\_MAILER=smtp

MAIL\_HOST=smtp.mailtrap.io

MAIL\_PORT=2525

MAIL\_USERNAME=null

MAIL\_PASSWORD=null

MAIL\_ENCRYPTION=null

MAIL\_FROM\_ADDRESS=null

MAIL\_FROM\_NAME="${APP\_NAME}"

AWS\_ACCESS\_KEY\_ID=

AWS\_SECRET\_ACCESS\_KEY=

AWS\_DEFAULT\_REGION=us-east-1

AWS\_BUCKET=

PUSHER\_APP\_ID=

PUSHER\_APP\_KEY=

PUSHER\_APP\_SECRET=

PUSHER\_APP\_CLUSTER=mt1

MIX\_PUSHER\_APP\_KEY="${PUSHER\_APP\_KEY}"

MIX\_PUSHER\_APP\_CLUSTER="${PUSHER\_APP\_CLUSTER}"

В ProductController.php добавляем сбор данных их кэша:

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Product;

use Session;

use App\Cart;

class ProductsController extends Controller

{

public function index()

{

if(\Cache::get('products')==null)

\Cache::put('products', Product::all());

$products = \Cache::get('products');

return view('products', compact('products'));

}

public function addProducts(Request $req)

{

$product = new Product();

$product->name = $req->input('name');

$product->description = $req->input('description');

$product->photo = $req->input('photo');

$product->price = (float)$req->input('price');

$product->save();

\Cache::put('products', null);

return redirect()->intended('/');

}

public function delete($id)

{

Product::find($id)->delete();

\Cache::put('products', null);

return redirect()->intended('/');

}

public function addToCart($id, Request $request)

{

$product = Product::find($id);

$oldCart = Session::has('cart') ? Session::get('cart') : null;

$cart = new Cart($oldCart);

$cart->add($product);

$request->session()->put('cart', $cart);

$request->session()->save();

return redirect()->intended('/');

}

public function showShoppingCart(Request $request){

if (Session::has('cart')){

$cart = $request->session()->get('cart');

return view('shoppingCart', ['products'=> $cart->items, 'totalPrice' => $cart->totalPrice]);

}

}

public function deleteFromCart($id, Request $request){

$oldCart = Session::get('cart');

$cart = new Cart($oldCart);

$product = Product::find($id);

$cart->delete($product);

$request->session()->put('cart', $cart);

$request->session()->save();

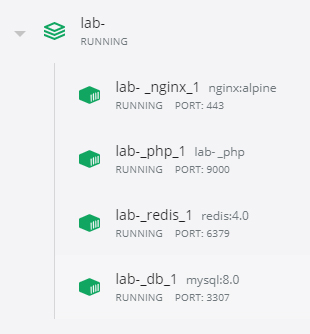
return redirect()->intended('showShoppingCart');

}

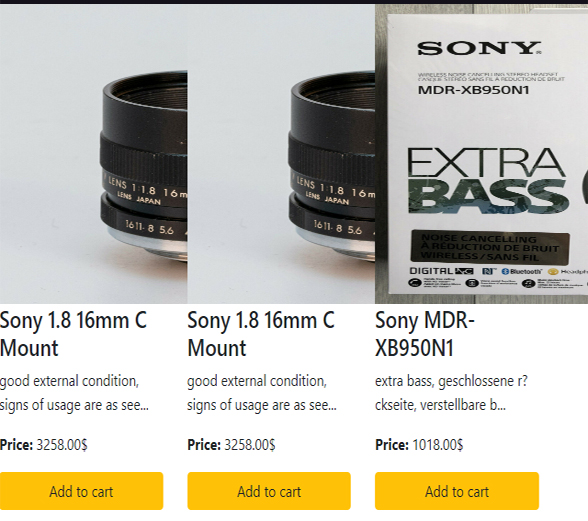
}

Запускаем сервисы в докер-контейнере





Просматриваем результат



**ВЫВОД**

В ходе лабораторной работы было изучено средство контейнеризации Docker и использование его для запуска приложения на Laravel.