Sklep internetowy z filmami

Laravel – Projekt Zaliczeniowy

Sebastian Winiarski

Piotr Wróblewski

Wymagania:

Do działania projektu wymagane są:

1. PHP 7.4
2. Laravel (composer global require laravel/laravel)
3. Laravel UI (composer require laravel/ui)

Do działania strony potrzebna jest włączona baza MySQL. Przed uruchomieniem strony należy migrować dane do bazy danych(php artisan migrate) lub wczytać do bazy danych przygotowaną tabelę(movie_store.sql). Następnie uruchomić projekt poleceniem php artisan serve.

Funkcjonalności strony:

Klient:

1. Wyświetlanie bazy dostępnych w sklepie filmów. Możliwość sortowania i filtrowania filmów według kategorii.
2. Podglądania szczegółów o wybranym filmie
3. Dodawanie filmów do koszyka zakupów, zarządzanie koszykiem.
4. Rejestracja konta i logowanie, możliwość resetowania hasła.
5. Dokonanie zakupu filmów dodanych do koszyka(wymagany proces podania danych wysyłkowych).

Admin:

1. To samo co klient +
2. Panel Admina(zabezpieczony tylko dla kont z rolą admin):
    a. Podgląd wszystkich filmów z szybkim dostępem do usuwania i edycji
    b. Dodawanie nowego filmu do katalogu sklepu
    c. Edytowanie szczegółów i kategorii wybranego filmu

By wejść do panelu admina należy się zalogować na konto admina(email: admin@admin.com,

hasło: password) i przejść do url 127.0.0.1:8000/admin

Opis działania kodu projektu:

Lista wszystkich Routes projektu:

```php
<?php

use Illuminate\Support\Facades\Route;

Route::get('/', function () {
    return redirect('/movies');
});

// Auth builtin routes
Route::get('/home', 'HomeController@index')->name('home');
Auth::routes();

// showing movies
Route::get('/movies', 'MoviesController@show_all')->name('movie.store');
Route::post('/movies', 'MoviesController@show_by_category');
Route::get('/movies/{movie}', 'MoviesController@show_one_movie')->name('show.movie');

// adding/removing from cart
Route::get('/add/{movie}', 'CartController@addToCart')->name('cart.add');
Route::get('/cart', 'CartController@showCart')->name('cart.show');
Route::get('/cart/remove/{movie}', 'CartController@removeFromCart')->name('cart.remove');

//checkout
Route::get('/checkout', 'CartController@shippingInfo')->name('shippingForm')->middleware('auth');
Route::put('/checkout', 'CartController@checkout')->name('checkoutAddress')->middleware('auth');

//Admin panel routes with authorization
Route::middleware('admin')->group(function(){
    Route::view('/admin', 'admin/admin');

    // List all movies in catalog
    Route::get('/admin/movies', 'AdminPanelController@show_all')->name('admin.listMovies');

    // Routes for showing, updating and deleting movies from catalog
    Route::get('/admin/movies/{movie}', 'AdminPanelController@show_movie')->name('admin.showMovie');
    Route::put('/admin/movies/{movie}', 'AdminPanelController@update_movie')->name('admin.updateMovie');
    Route::delete('/admin/movies/{movie}', 'AdminPanelController@delete_movie')->name('admin.deleteMovie');
    Route::post('/admin/movies/{movie}', 'AdminPanelController@add_category_to_movie')->name('admin.addCategory');
    Route::delete('/admin/movies/{movie}/{category}', 'AdminPanelController@delete_movie_category')->name('admin.deleteCategory');

    // Adding new movie to catalog
    Route::view('/admin/movie/create', 'admin/createMovie')->name('admin.createMovieForm');
    Route::post('/admin/movie/create', 'AdminPanelController@create_movie')->name('admin.createMovie');
});

Route::get('logout', '\App\Http\Controllers\Auth\LoginController@logout')->name('logout');

// if wrong url then show string
Route::fallback(function(){
    return "Wygląda na to, że zawędrowałeś za daleko!";
});
```

MoviesController odpowiedzialny za wyświetlanie filmów na stronie głównej i szczegółów jednego, wybranego filmu:

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use App\Movies;
use App\Categories;


class MoviesController extends Controller
{
    // get all movie from Movies table, order by release date and paginate
    function show_all(){
        $movies = Movies::orderBy('release_date', 'desc')->paginate(20);
        $categories = Categories::orderBy('name')->get();

        return view('store', ['movies'=>$movies, 'categories'=>$categories, 'chosen_category'=>'Wszystkie']);
    }

    // get all info and categories about one movie with given id
    function show_one_movie($id){
        $movie = Movies::find($id);
        $categories = $movie->Categories;

        return view('pages.movie', ['movie'=>$movie, 'categories'=>$categories]);
    }

    // show movies from chosen category and sort
    function show_by_category(Request $request){
        $category = $request->input('categories');
        $sorting = $request->input('sort');

        if($category == 'Wszystkie'){
            $movies = Movies::orderBy($sorting, 'desc')->paginate(20);
        } else {
            $movies = Movies::whereHas('categories', function($q) use($category){
                $q->where('name', $category);
            })
            ->orderBy($sorting, 'desc')
            ->paginate(20);
        }

        $categories = Categories::all();
        return view('/store', ['movies'=>$movies, 'categories'=>$categories, 'chosen_category'=>$category]);
    }
}

```

Cart Controller

CartController odpowiada za funkcje związane z zakupem filmów, a więc z zarządzaniem koszykiem i przeprowadzaniem procesu zamówienia.

Na zarządzanie koszykiem składają się funkcje pokazywania zawartości koszyka, dodawania filmów do koszyka i usuwania dodanych do koszyka filmów.

Część odpowiedzialna za zarządzanie koszykiem:

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Movies;
use App\Cart;
use App\User;
use App\Order;
use App\ShippingInfo;
use Carbon\Carbon;
use Session;
use Auth;


class CartController extends Controller
{
    // func checking if there already is cart in Sessio, if there is then creates new cart with old cart data, otherwise creates new cart with default data
    function addToCart(Request $request, Movies $movie){
        $oldCart = Session::has('cart') ? Session::get('cart') : null;
        $cart = new Cart($oldCart);
        $cart->add($movie, $movie->id);

        Session::put('cart', $cart);

        return redirect()->route('movie.store');
    }

    // func showing all items stored in cart session
    function showCart(){
        if(!Session::has('cart')){
            return view('pages.shopping-cart');
        }

        $oldCart = Session::get('cart');
        $cart = new Cart($oldCart);
        return view('pages.shopping-cart', ['products' => $cart->items, 'totalPrice' => $cart->totalPrice]);
    }

    // func for removing item from cart
    function removeFromCart($id){
        $oldCart = Session::has('cart') ? Session::get('cart') : null;
        $cart = new Cart($oldCart);
        $cart->removeItem($id);

        //check if there is any item left in session cart, if not then cart clear session data
        if(count($cart->items) > 0){
            Session::put('cart', $cart);
        } else {
            Session::forget('cart');
        }

        return redirect()->route('cart.show');
    }
```

Cart Controller cd.

Na zarządzanie procesem zamówienia składa się wyświetlenie formularza do wprowadzenia danych wysyłkowych. Jeżeli użytkownik wcześniej już takie dane wprowadził, to pola formy będą domyślnie wypełnione tymi danymi.

Po wprowadzeniu danych wysyłkowych i zatwierdzeniu złożenia zamówienia koszyk(działający w sesji) zostaje wyczyszczony, a do bazy danych zostaje dodane zamówienie, a także dane wysyłkowe.

Część odpowiedzialna za zarządzanie procesem zamówienia:

```php
    // func for checking if currently logged user already has shipping info. If he has, then we will print it in input form fields in blade.
    function shippingInfo(){
        if(!Session::has('cart')){
            return view('pages.shopping-cart');
        }
        $oldCart = Session::get('cart');
        $cart = new Cart($oldCart);
        $user = Auth::user();
        $shipping_data = ShippingInfo::where('user_id', $user->id)->get();

        // dd($shipping_data);
        return view('pages.checkout', ['products' => $cart->items, 'totalPrice' => $cart->totalPrice, 'shippingInfo' => $shipping_data]);
    }

    function checkout(Request $request){
        if(!Session::has('cart')){
            return view('pages.shopping-cart');
        }
        $oldCart = Session::get('cart');
        $cart = new Cart($oldCart);
        $user = Auth::user();
        // $shipping_data = User::find($user->id)->shipping_info;
        $shipping_data = ShippingInfo::updateOrCreate(
            ['user_id' => $user->id],
            ['user_id' => $user->id,
            'name' => $request->input('name'),
            'surname' => $request->input('surname'),
            'country' => $request->input('country'),
            'city' => $request->input('city'),
            'street' => $request->input('street')]
        );

        $order = new Order;
        $order->user_id = $user->id;
        $order->shipping_id = $shipping_data->id;
        $order->total_cost = $cart->totalPrice;
        $order->paid = FALSE;
        $order->payment_type = $request->payment;
        $order->created_at = Carbon::now()->toDateTimeString();
        $order->save();
        Session::forget('cart');

        return redirect()->route('movie.store');
    }
```

AdminPanelController

AdminPanelController odpowiada za funkcjonalności związane z panelem admina. Na ten moment są to funkcje związane z zarządzaniem katalogiem produktów sklepu.

Poniżej kod funkcji odpowiedzialnych za wyświetlanie wszystkich filmów, konkretnego filmów z jego danymi(gotowymi do aktualizacji) i kategoriami:

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use App\Movies;
use App\Categories;


class AdminPanelController extends Controller
{
    // get all movies in database, order them by title and paginate
    function show_all(){
        $movies = Movies::orderBy('title', 'asc')->paginate(20);

        return view('admin.adminlistMovies', ['movies'=>$movies]);
    }

    // show one movie info, based on given id. return infos about movie, movie category(pivot between movies and category) and all categories
    function show_movie($id){
        $movie = Movies::find($id);
        $movie_categories = $movie->Categories;
        $categories = Categories::all();

        return view('admin.adminEditMovie', ['movie'=>$movie, 'movie_categories'=>$movie_categories, 'categories'=>$categories]);
    }

    // update all info of movie based in sent POST input(with PUT method) and given id
    function update_movie(Request $request, $id){
        $movie = Movies::find($id);
        $movie->title = $request->input('title');
        $movie->picture = $request->input('picture');
        $movie->release_date = $request->input('release_date');
        $movie->price = $request->input('price');
        $movie->description = $request->input('description');
        $movie->save();
        $movie_categories = $movie->Categories;
        $categories = Categories::all();

        return view('admin.adminEditMovie', ['movie'=>$movie, 'movie_categories'=>$movie_categories, 'categories'=>$categories]);
    }
```

AdminPanelController cd.

Dalsza część AdminPanelController odpowiedzialna za dodawanie nowego filmu do katalogu sklepowego, usuwanie filmu z katalogu oraz zarządzanie kategoriami danego filmu:

```php
    // add to db movie with data taken from post form inputs, return created movie page
    function create_movie(Request $request){
        $movie = new Movies;
        $movie->title = $request->input('title');
        $movie->picture = $request->input('picture');
        $movie->release_date = $request->input('release_date');
        $movie->price = $request->input('price');
        $movie->description = $request->input('description');
        $movie->save();

        $movie_categories = $movie->Categories;
        $categories = Categories::all();

        return redirect()->route('admin.showMovie', ['movie'=>$movie, 'movie_categories'=>$movie_categories, 'categories'=>$categories]);
        // return view('admin.adminEditMovie', ['movie'=>$movie, 'movie_categories'=>$movie_categories, 'categories'=>$categories]);
    }

    // delete movie with given id
    function delete_movie($id){
        $movie = Movies::find($id);
        $movie->delete();

        return redirect()->route('admin.listMovies');
    }

    // get movie_id and category_id and detach pivot table from movie
    function delete_movie_category($movie_id, $category_id){
        $movie = Movies::find($movie_id);
        $movie->categories()->detach($category_id);

        $movie_categories = $movie->Categories;
        $categories = Categories::all();

        return view('admin.adminEditMovie', ['movie'=>$movie, 'movie_categories'=>$movie_categories, 'categories'=>$categories]);
    }

    // find movie by given id, then take category id from request and attach both to create pivot if no such pivot exists
    function add_category_to_movie(Request $request, $movie_id){
        $movie = Movies::find($movie_id);
        $movie->categories()->attach($request->input('categories'));

        $movie_categories = $movie->Categories;
        $categories = Categories::all();

        return view('admin.adminEditMovie', ['movie'=>$movie, 'movie_categories'=>$movie_categories, 'categories'=>$categories]);
```

Modele

W projekcie znajduje się 6 modeli:

1. Cart – przechowywanie danych o koszyku
2. Categories – przechowywanie danych o kategoriach filmów
3. Movies – przechowywanie danych i filmach
4. Order – przechowywanie danych o zamówieniach
5. ShippingInfo – przechowywanie danych wysyłkowych klientów
6. User – przechowywanie danych o użytkownikach

Cart:

Model kart odpowiada za zarządzanie koszykiem przechowywującym dane w sesji(w pliku typu file).

```php
<?php

namespace App;

class Cart
{

    public $items;
    public $totalQty = 0;
    public $totalPrice = 0;

     // upon object creation we check if there is already a cart in session - if true, then write it's info to variables
    public function __construct($oldCart){
        if($oldCart){
            $this->items = $oldCart->items;
            $this->totalQty = $oldCart->totalQty;
            $this->totalPrice = $oldCart->totalPrice;
        }
    }

    // adding item to cart
    public function add($item, $id){
        $storedItem = ['qty' => 0, 'price' => $item->price, 'item' => $item];
        // if there is already data in items(which implies that there was oldCart) then check if there is a movie with given id
        // if there is, then overwrite storedItem with existing movie info
        if($this->items){
            if(array_key_exists($id, $this->items)){
                $storedItem = $this->items[$id];
            }
        }
        // increase number of item in cart by one, then count total price of given number of item
        $storedItem['qty']++;
        $storedItem['price'] = $item->price * $storedItem['qty'];
        // saving storeditem data to variables to pass it later to Session data
        $this->items[$id] = $storedItem;
        $this->totalQty++;
        $this->totalPrice += $item->price;
    }

    // removing item from cart
     function removeItem($id){
         $this->totalQty -= $this->items[$id]['qty'];
         $this->totalPrice -= $this->items[$id]['price'];
         unset($this->items[$id]);
     }

}

```

Categories:

Posiada relację many-to-many z modelem Movies.

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Categories extends Model
{
    public $timestamps = false;

    public function movies(){
        return $this->belongsToMany('App\Movies');
    }
}

```

Movies:

Posiada relację many-to-many z modelem Categories, oraz relację one-to-many z modelem Order.

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Movies extends Model
{
    public $timestamps = false;

    public function categories(){
        return $this->belongsToMany('App\Categories');
    }

    public function order(){
        return $this->belongsToMany('App\Order');
    }
}

```

Order:

Posiada relację many-to-many z modelem Movie, oraz relację many-to-one z modelem User.

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Order extends Model
{
    public function user()
    {
        return $this->belongsTo('App\User');
    }

    public function movies(){
        return $this->belongsToMany('App\Movies');
    }

    protected $attributes = [
        'paid' => FALSE,
    ];
}

```

ShippingInfo:

Posiada relację one-to-one z modelem User. Tablica fillable została zdefiniowana do wykorzystania w kontrolerze funkcji UpdateOrCreate.

```php
namespace App;

use Illuminate\Database\Eloquent\Model;

class ShippingInfo extends Model
{

    protected $fillable = [
        'user_id', 'name', 'surname', 'country', 'city', 'street'
    ];

    public function user(){
        return $this->belongsTo('App\User');
    }
}

```

User:

Model stworzony przez wbudowaną funkcję Laravela „Auth”. Została nieznacznie zmodyfikowana. Posiada relację one-to-one z ShippingInfo oraz relację one-to-many z modelem Order.

```php
<?php

namespace App;

use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

class User extends Authenticatable
{
    use Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'email', 'password',
    ];

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password', 'remember_token',
    ];

    protected $attributes = [
        'role' => 'customer',
    ];

    public function shippingInfo(){
        return $this->hasOne('App\ShippingInfo');
    }

    public function order(){
        return $this->hasMany('App\Order');
    }

}
```

Middleware

W middleware użyliśmy wbudowanych funkcji auth, jak również własnej, prostej funkcji autoryzacyjnej sprawdzającej czy dany użytkownik ma prawo dostępu do panelu admina(do tego wymagana jest posiadana rola admin).

movieshop > app > Http > Middleware > AdminMiddleware.php > AdminMiddleware

```php
<?php

namespace App\Http\Middleware;

use Closure;

class AdminMiddleware
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @return mixed
     */
    public function handle($request, Closure $next)
    {
        if ($request->user() && $request->user()->role != 'admin'){
            return redirect()->route('movie.store');
        } elseif ( !($request->user()) ){
            return redirect()->route('login');
        }
        return $next($request);
    }

}
```