

Laravel - Tworzenie Aplikacji, Wzorzec MVC

Projekt zaliczeniowy - dokumentacja

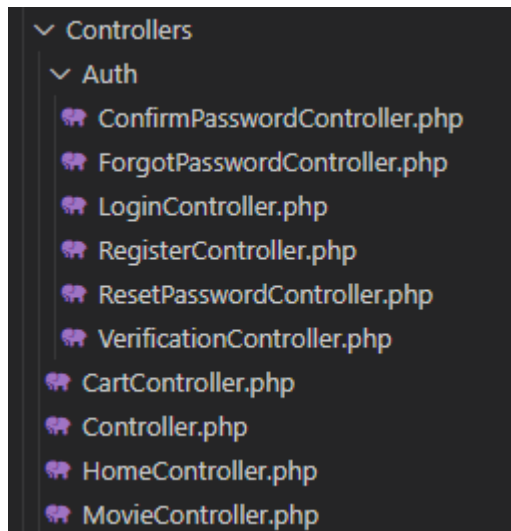
Artur Spychalla, nr albumu: 23546

Paweł Wąsowski, nr albumu: 23131

1. Aplikacja „MovieShop” umożliwia takie funkcjonalności jak:
 - pobieranie danych na temat danego filmu z bazy danych,
 - wyświetlanie kart z filmami,
 - pobieranie kategorii filmu i sortowanie po wybranej kategorii,
 - dodawanie nowego filmu do bazy danych,
 - usuwanie filmu z bazy danych,
 - edycja filmu w bazie danych,
 - dodawanie filmów do koszyka sprzedażowego,
 - usuwanie filmów z koszyka sprzedażowego,
 - sumowanie cen za filmy dodane do koszyka.

2. Aplikacja składa się z:

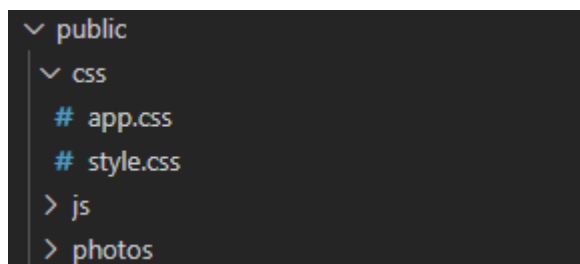
a) Kontrolerów:



b) Modeli:



c) CSS:



d) Widoków:

```
views
├── auth
├── layouts
│   ├── app.blade.php
│   ├── log.blade.php
│   ├── create.blade.php
│   ├── edit.blade.php
│   ├── home.blade.php
│   ├── login.blade.php
│   ├── shopping-cart.blade.php
│   └── welcome.blade.php
```

3. Poniżej przedstawione są poszczególne funkcje aplikacji:

MovieController.php

Poniższa funkcja index odpowiada za pobranie filmów i kategorii aby przekazać je do kolejnego szablonu blade'a.

```
public function index($id){

    $genres = app('App\Http\Controllers\HomeController')->getGenres();
    $movies = Movie::where('id', $id)->get();
    return view('edit',['movies'=>$movies, 'genres' => $genres]);
}
```

Funkcja insert odpowiada za dodawanie nowego filmu do bazy danych. Użyliśmy mapowania do wprowadzania danych do bazy. Na końcu wykonujemy redirect do głównej strony.

```
public function insert(Request $request){
    $path = $request->input('path');
    $title = $request->input('title');
    $category = $request->input('category');
    $production_year = $request->input('production_year');
    $description = $request->input('description');
    $price = $request->input('price');

    $data=array('path' => $path, "title" => $title, "category" => $category,
    "production_year" => $production_year, "description" => $description, "price" => $price);
    DB::table('movies')->insert($data);

    $genres = app('App\Http\Controllers\HomeController')->getGenres();
    return redirect()->route('home');
}
```

Funkcja destroy odpowiada za usunięcie wybranego filmu. Na końcu wykonujemy redirect do głównej strony.

```
public function destroy($id) {
    $deleting = Movie::where('id',$id)->delete();
    return redirect()->route('home');
}
```

Funkcja edit odpowiada za edycję istniejącego filmu. Użyliśmy mapowania do wprowadzania danych do bazy. Na końcu wykonujemy redirect do głównej strony.

```
public function edit(Request $request,$id) {
    $path = $request->input('path');
    $title = $request->input('title');
    $category = $request->input('category');
    $production_year = $request->input('production_year');
    $description = $request->input('description');
    $price = $request->input('price');

    $update = Movie::where('id', $id)->update(['path' => $path, 'title'=> $title,
    'category' => $category, 'production_year' => $production_year,
    'description' => $description, 'price'=> $price]);

    return redirect()->route('home');
}
```

HomeController.php

Funkcja index odpowiada za pobranie wszystkich filmów z bazy danych i przekazanie ich do widoku „home” – tak aby można było pobrać te dane do wyświetlenia. Następnie w ten sam sposób pobieramy kategorie, które widoczne są w navbarze.

Funkcja getByGenre służy do pobierania danych na temat danej kategorii i dopasowanych do niej filmów. Następnie przekazujemy te informacje do widoku „home” – tak aby można było pobrać dane do wyświetlenia.

Funkcja getGenres pobiera wszystkie kategorie.

```
public function index()
{
    $movies = Movie::get();
    $genres = HomeController::getGenres();
    return view('home', ['movies' => $movies, 'genres' => $genres]);
}

//Grouping by genre
public function getByGenre(Request $request){
    $category = $request -> path();
    $category = substr($category, 5);
    $moviesByGenre = Movie::where('category', $category) -> get();
    $genres = HomeController::getGenres();
    return view('home', ['movies' => $moviesByGenre, 'genres' => $genres]);
}

public function getGenres(){
    $genres = Movie::select('category as genre') -> get();
    return $genres;
}
```

CartController.php

Funkcja getIndex odpowiada za pobranie wszystkich filmów:

```
public function getIndex()
{
    $movies = Movie::all();
    return view('home', ['movies'=> $movies]);
}
```

Funkcja `getAddToCart` służy do dodawania wybranego filmu do koszyka sprzedażowego.

Funkcja `getSubstractedCart` służy do usuwania wybranych filmów z koszyka sprzedażowego.

```
//Adding new movie to cart
public function getAddToCart(Request $request, $id){
    $movie = Movie::find($id);
    $oldCart = Session::has('cart') ? Session::get('cart') : null;
    $cart = new Cart($oldCart);
    $cart -> add($movie, $movie->id);

    $request->session()->put('cart',$cart);
    return redirect()->route('home');
}

//Deleting movie from the cart
public function getSubstractedCart($id){
    $oldCart = Session::has('cart') ? Session::get('cart') : null;
    $cart = new Cart($oldCart);
    $cart -> subtractOneMovie($id);

    if(count($cart->items) > 0){
        Session::put('cart', $cart);
    }else{
        Session::forget('cart');
    }
    return redirect()->route('shoppingCart');
}
```

Funkcja `getCart` odpowiada za wyświetlanie listy dodanych do koszyka filmów.

```
//Shopping Cart
public function getCart(){

    $oldCart = Session::get('cart');
    $cart = new Cart($oldCart);
    $genres = app('App\Http\Controllers\HomeController')->getGenres();
    return view('shopping-cart', ['movies' => $cart->items,
    'totalPrice'=>$cart->totalPrice, 'genres' => $genres]);
}
```

Model Movie.php

```
class Movie extends Model
{
    public $timestamps = false;

    protected $fillable = ['path','title','category','production_year','description','price'];

    protected $table = 'movies';
}
```

Fragment kodu z szablonu [home.blade.php](#), który odpowiada za wyświetlenie danych z bazy movies.

```
<h3 class="card-title" > {{$movie->title}}</h3>
<p class="card-text" > Category: {{$movie->category}}</p>
<p class="card-text"> Year of production: {{$movie->production_year}}</p>
<p class="card-text" > Price: {{$movie->price}}$</p>
```

Web.php

Poniżej znajdują się wszystkie routy, dzięki którym możemy wywołać wybrane funkcje zdefiniowane w kontrolerach:

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

Auth::routes();

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

Route::get('/home/{genre}', 'HomeController@getByGenre')->name('genre');

Route::get('/home/Add-to-cart/{id}', 'CartController@getAddToCart')->name('add');

Route::get('/home/substract-from-cart/{id}', 'CartController@getSubstractedCart')->name('substract');

Route::get('/home/shop/Cart', 'CartController@getCart')->name('shoppingCart');

Route::get('/home/movie/insert','MovieController@insertform')->name('insertData');

Route::post('create','MovieController@insert');

Route::get('delete/{id}','MovieController@destroy');

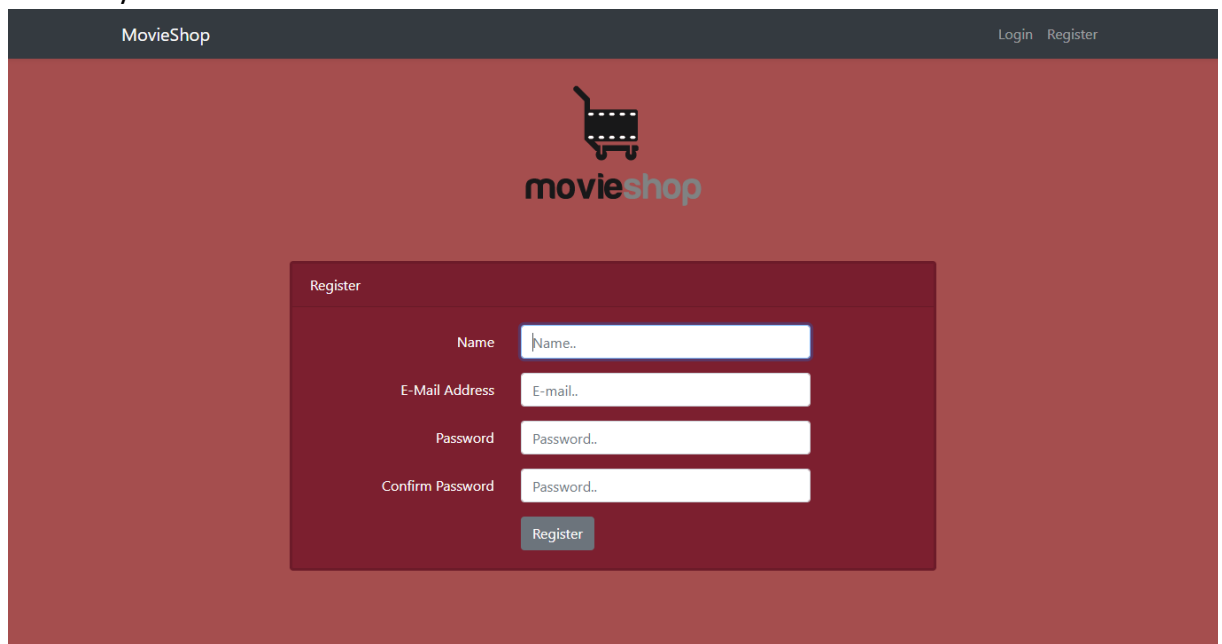
Route::get('/home/movie/edit/{id}','MovieController@index')->name('editData');

Route::post('edit/{id}','MovieController@edit');
```

4. Aplikacja prezentuje się następująco:

Ekran rejestracji nowego użytkownika:

W tym ekranie można utworzyć konto. Jeśli nie wypełnimy inputów, pojawi się stosowny komunikat.



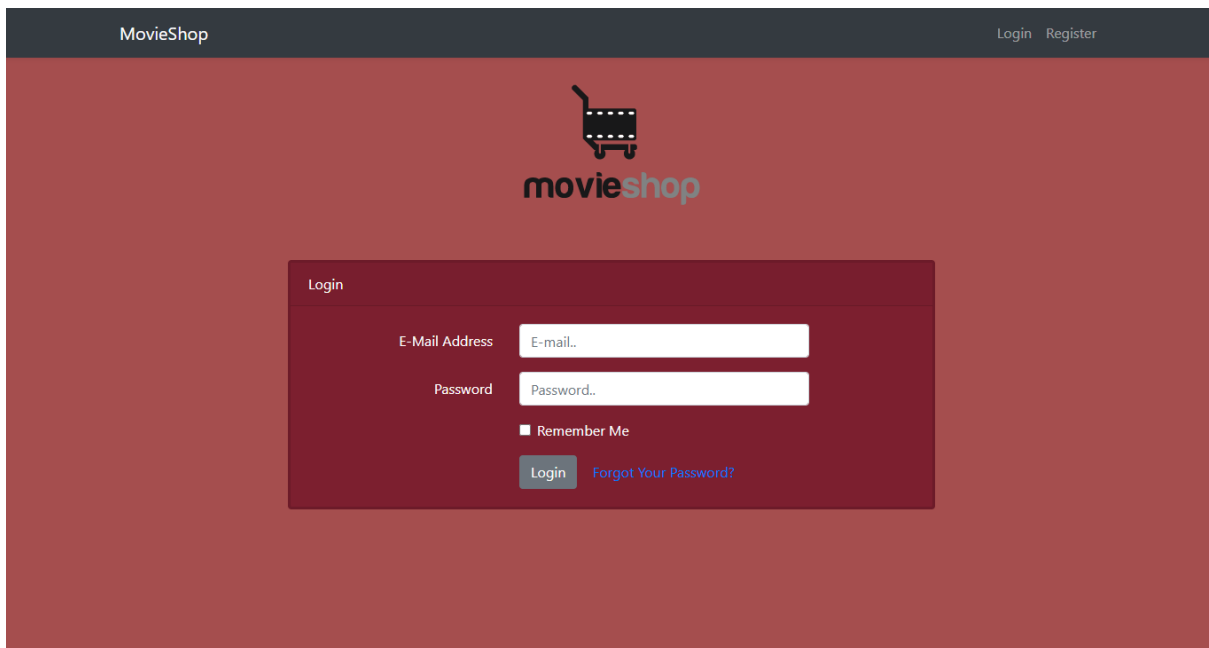
The screenshot shows the 'Register' form on the MovieShop website. The form is white and centered on a red background. It contains the following fields and labels:

- Name: Name..
- E-Mail Address: E-mail..
- Password: Password..
- Confirm Password: Password..

At the bottom of the form is a 'Register' button.

Ekran logowania:

W tym ekranie można zalogować się do aplikacji. Jeśli konto nie jest utworzone pokaże się stosowny komunikat. Jeśli nie wypełnimy inputów, pojawi się stosowny komunikat.



Główny ekran:

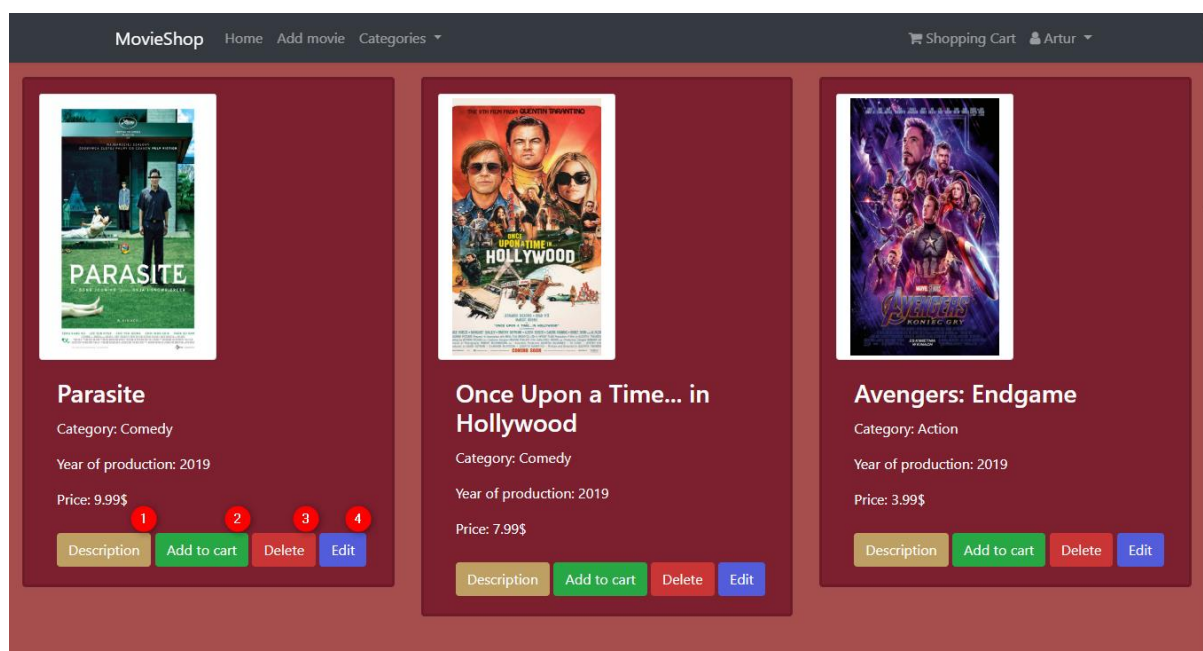
W tym ekranie można zobaczyć wszystkie filmy, które są w bazie danych. Dodatkowo za pomocą navbaru można przejść do kolejnych zakładek oraz za pomocą przycisków:

Description – pobrać opis z bazy danych i go wyświetlić.

Add to cart – dodać dany film do koszyka zakupowego.

Delete – usunąć film z bazy danych.

Edit – wyedytować wybrany filmu.



Ekran dodawania nowego filmu:

Wszystkie pola muszą zostać wypełnione.

MovieShop

Home

Add movie

Categories ▾

Shopping Cart

Artur ▾

You can add a new movie here. Please fill all of below fields.

Enter path to the image..

Enter title..

Enter category..

Enter year of production..

Enter description..

Enter price..

Add

© 2020 Copyright:

Paweł Wąsowski & Artur Spychalla CDV 2020

Ekran edycji filmu:

Domyślnie wypełnione są wszystkie dane danego filmu, który edytujemy.

MovieShop

Home

Add movie

Categories ▾

Shopping Cart

Artur ▾

You can edit a movie here. Please fill all of below fields.

parasite.jpg

Parasite

Comedy

2019

Greed and class discrimination threaten the newly formed symbiotic relationship between the wealthy Park family and the destitute Kim clan.

9.99

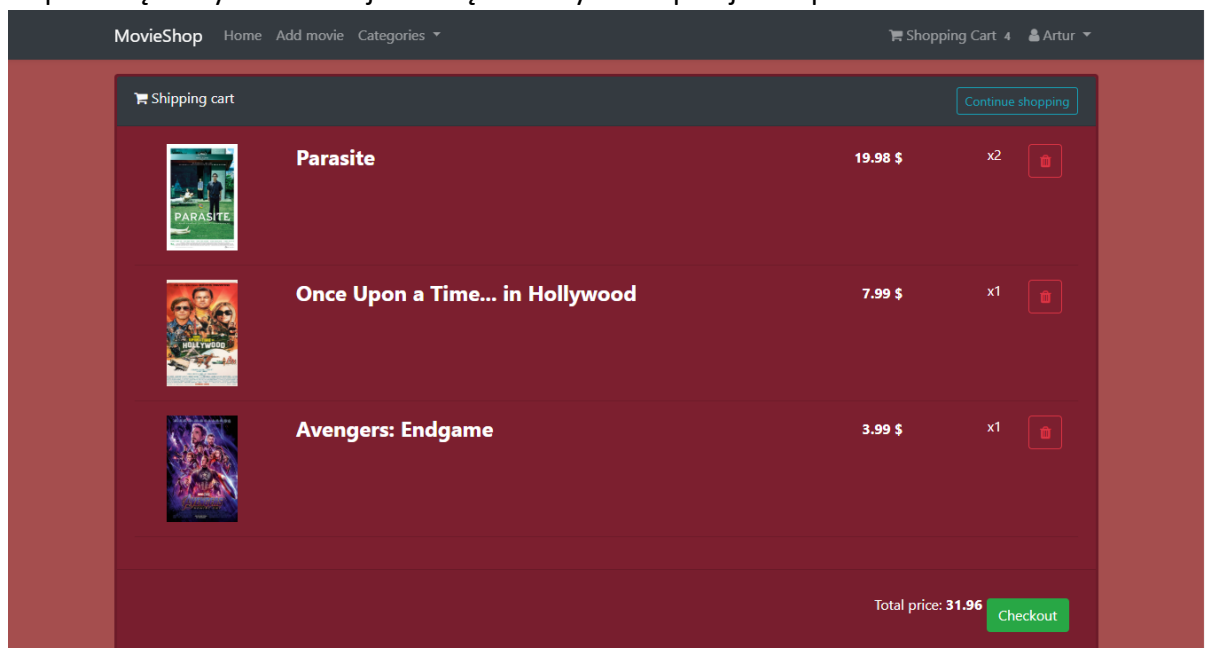
Edit

© 2020 Copyright:

Paweł Wąsowski & Artur Spychalla CDV 2020

Ekran koszyka sprzedażowego:

W tym ekranie widzimy wszystkie filmy, które dodaliśmy do koszyka zakupowego. Możemy za pomocą ikony śmietnika je usunąć z koszyka lub przejść do płatności.



Baza danych MySQL:

