

AI1	Dokumentacja projektu
Autor	Dawid Olko, 125148
Kierunek, rok	Informatyka, II rok, st. stacjonarne (3,5-I)
Temat projektu	<i>Platforma wypożyczania filmów VOD</i>

Spis treści

1. Wstęp	4
Zakres Działania Aplikacji	4
Główne Elementy Techniczne	5
2. Narzędzia i technologie	6
Framework Laravel	6
Baza danych	6
Konfiguracja połączenia z serwerem bazodanowym:	6
Frontend	7
Middleware	7
Kontrolery	7
Modele	8
Usługi	8
Walidacja Danych	9
Seedowanie Danych	9
3. Baza danych	10
Diagram ERD	10
Opis rozwiązań zastosowanych w bazie	10
MOVIES	10
USERS	10
LOANS	10
LOAN_MOVIE	10
OPINIONS	11
CATEGORIES	11
PREMIUM_MOVIES	11
REFERRAL_CODES	11
LOYALTY_POINTS	11
RECOMMENDATIONS	11
FRIENDSHIPS	11

Migracje	11
Seedery	12
Opis Przykładowej Relacji w Modelach	12
4. GUI	14
Opis Interfejsu Użytkownika	14
Główne Widoki Aplikacji	14
5. Uruchomienie aplikacji	22
Wymagania	22
Kroki przy pierwszym uruchomieniu	22
Kroki przy kolejnych uruchomieniach	23
6. Funkcjonalności Aplikacji	24
Logowanie i Rejestracja	24
Logowanie:	24
Rejestracja:	25
Przykładowi użytkownicy:	25
Przeglądanie Zasobów	26
Strona Główna:	26
Katalog Filmów:	26
Szczegóły Filmu:	27
Regulamin Strony:	29
Zarządzanie przez Administratora	29
CRUD Filmów:	29
Zarządzanie Użytkownikami:	30
Zarządzanie Wypożyczeniami:	32
Funkcjonalności dla Zalogowanych Użytkowników	32
Profil Użytkownika:	32
Koszyk:	33
Opinie:	33
Proces Wypożyczania Filmów	34
Dodawanie do Koszyka:	34
Finalizacja Zamówienia:	34
Historia Wypożyczeń:	35
7. Walidacja danych	36
Walidacja dla różnych typów żądań	36

1. Wstęp

Projekt dotyczy stworzenia aplikacji internetowej do wypożyczania filmów VOD (Video on Demand) przy użyciu frameworka Laravel. Aplikacja jest przeznaczona do obsługi małego przedsiębiorstwa zajmującego się wypożyczaniem filmów online. Użytkownicy aplikacji mogą przeglądać katalog dostępnych filmów, wypożyczać je na określony czas oraz zarządzać swoimi zamówieniami i danymi. Aplikacja zapewnia pełną automatyzację procesu wypożyczania filmów, obliczania kosztów, obsługi promocji oraz przyznawania punktów lojalnościowych.

Zakres Działania Aplikacji

Aplikacja obejmuje następujące funkcjonalności:

1. Rejestracja i logowanie użytkowników:

- Tylko zarejestrowani i zalogowani użytkownicy mogą korzystać z pełnych funkcji aplikacji, takich jak wypożyczanie filmów.
- Nowi użytkownicy mogą samodzielnie się zarejestrować, a po rejestracji uzyskują dostęp do swojego konta.

2. Zarządzanie filmami i zamówieniami przez administratora:

- Administrator ma możliwość dodawania, edytowania oraz usuwania filmów w bazie danych.
- Administrator zarządza również zamówieniami oraz zmianą danych użytkowników czy ich usunięciem.

3. Automatyzacja procesów biznesowych:

- Aplikacja automatycznie oblicza koszt wypożyczenia filmów, uwzględniając dynamiczne ceny promocyjne.
- Przyznawanie punktów lojalnościowych za wypożyczenia oraz zarządzanie ich stanem jest również zautomatyzowane.
- Obsługa zmiany dostępności filmów oraz inne operacje związane z zarządzaniem katalogiem filmów są wykonywane automatycznie.

4. Funkcjonalności dla użytkowników:

- Użytkownicy mogą przeglądać dostępne filmy, korzystając z różnych filtrów i sortowań.
- Mogą dodawać filmy do koszyka, wypożyczać je oraz zarządzać swoimi zamówieniami.
- Użytkownicy mają dostęp do swojego profilu, gdzie mogą edytować swoje dane oraz przeglądać historię wypożyczeń.

5. Interfejs użytkownika:

- Aplikacja posiada responsywny interfejs użytkownika, zaprojektowany z użyciem silnika szablonów Blade.
- Interfejs zawiera różnorodne elementy, takie jak formularze, przyciski, listy, oraz dynamiczne elementy, które poprawiają doświadczenie użytkownika.

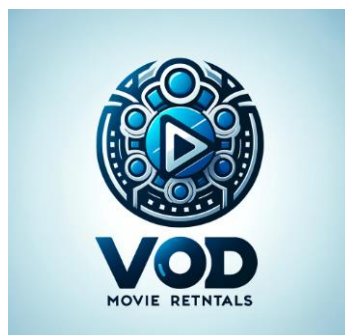
6. Bezpieczeństwo:

- Aplikacja implementuje mechanizmy uwierzytelniania i autoryzacji, zapewniając bezpieczeństwo danych użytkowników.
- Wszystkie wrażliwe operacje są zabezpieczone, a dostęp do funkcji administracyjnych jest ograniczony do uprawnionych użytkowników.

Główne Elementy Techniczne

- **Framework:** Laravel 11.x
- **Baza danych:** MySQL
- **Front-end:** Blade (szablony), CSS, JavaScript
- **Backend:** PHP, Laravel
- **Inne technologie:** Faker (do seedowania danych czy migracji tabel do bazy), Bootstrap (do stylizacji interfejsu)

Aplikacja "Wypożyczalnia Filmów VOD" jest zaprojektowana tak, aby zapewnić wygodę użytkownika zarówno dla klientów, jak i administratorów. Automatyzacja wielu procesów biznesowych znacznie upraszcza zarządzanie zamówieniami i katalogiem filmów, co przekłada się na efektywność działania całego systemu.



RentalVOD

2. Narzędzia i technologie

Framework Laravel

Laravel to popularny framework aplikacji webowych oparty na języku PHP, stworzony przez Taylora Otwell. Jest to zaawansowane narzędzie, które umożliwia szybkie i efektywne tworzenie skalowalnych aplikacji webowych, wykorzystując nowoczesne techniki programowania oraz wzorce projektowe, co ułatwia tworzenie czytelnego i łatwego w utrzymaniu kodu.

W projekcie wykorzystujemy Laravel w wersji 11.x, który oferuje liczne funkcjonalności i usprawnienia w porównaniu do poprzednich wersji. Laravel jest udostępniany na podstawie licencji MIT, co oznacza, że możemy swobodnie korzystać, modyfikować i rozpowszechniać framework. Oficjalna dokumentacja Laravel jest dostępna na stronie Laravel Docs, gdzie można znaleźć szczegółowe informacje na temat funkcji, klas, interfejsów i sposobu użycia różnych komponentów frameworka.

Baza danych

Aplikacja wykorzystuje silnik bazodanowy MySQL. MySQL jest oprogramowaniem typu open source, dostępnym na licencji GNU General Public License (GPL). Możesz pobrać najnowszą wersję MySQL ze strony MySQL Downloads.

Konfiguracja połączenia z serwerem bazodanowym:

Aby skonfigurować połączenie z serwerem bazodanowym MySQL, należy edytować plik konfiguracyjny `config/database.php`. W tym pliku znajduje się sekcja `connections`, w której można zdefiniować ustawienia połączenia dla różnych środowisk (np. `development`, `production`).

Konfiguracja połączenia z bazą danych MySQL:

```
'mysql' => [
    'driver' => 'mysql',
    'url' => env('DB_URL'),
    'host' => env('DB_HOST', '127.0.0.1'),
    'port' => env('DB_PORT', '3306'),
    'database' => env('DB_DATABASE', 'laravel'),
    'username' => env('DB_USERNAME', 'root'),
    'password' => env('DB_PASSWORD', ''),
    'unix_socket' => env('DB_SOCKET', ''),
    'charset' => env('DB_CHARSET', 'utf8mb4'),
    'collation' => env('DB_COLLATION', 'utf8mb4_unicode_ci'),
    'prefix' => '',
    'prefix_indexes' => true,
    'strict' => true,
    'engine' => null,
    'options' => extension_loaded('pdo_mysql') ? array_filter([
        PDO::MYSQL_ATTR_SSL_CA => env('MYSQL_ATTR_SSL_CA'),
    ]) : [],
],
```

W powyższym kodzie należy ustawić odpowiednie wartości dla parametrów host, port, database, username i password, zgodnie z konfiguracją Twojego serwera bazodanowego MySQL.

Frontend

Aplikacja wykorzystuje Blade jako silnik szablonów do generowania widoków HTML. Blade jest prosty w użyciu, a jednocześnie bardzo potężny, pozwala na wykorzystanie pełnych możliwości PHP bez komplikacji związanych z innymi silnikami szablonów. W połączeniu z CSS i JavaScript, zapewnia dynamiczny i responsywny interfejs użytkownika.

Middleware

Middleware w Laravelu służy do wykonywania różnych zadań pośrednich w trakcie przetwarzania żądań HTTP. W projekcie używamy middleware do sprawdzania uprawnień użytkowników, na przykład czy użytkownik jest administratorem:

```
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class IsAdmin
{
    public function handle(Request $request, Closure $next)
    {
        if (Auth::check() && Auth::user()->role_id == 1) {
            return $next($request);
        }
        abort(403);
    }
}
```

Kontrolery

Kontrolery w Laravelu organizują logikę aplikacji i odpowiadają za obsługę żądań HTTP. Przykładem może być kontroler AdminController, który zarządza użytkownikami i filmami:

```
class AdminController extends Controller
{
    protected $movieService;

    public function __construct(MovieService $movieService)
    {
        $this->movieService = $movieService;
    }

    public function index() {
        return view('admin.index');
    }

    public function users() {
        $users = User::paginate(10);
        return view('admin.editUsers', compact('users'));
    }

    public function movies()
    {
        $movies = Movie::paginate(10);

        foreach ($movies as $movie) {
            $dynamicPrice = $this->movieService->calculateDynamicPrice($movie);
            $movie->price_day = $dynamicPrice;
            $movie->save();
        }

        return view('admin.editMovies', compact('movies'));
    }
}
```

Modele

Modele w Laravelu odpowiadają za interakcje z bazą danych. Przykładowy model Movie może wyglądać tak:

```
<?php

namespace App\Models;

// use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

class User extends Authenticatable
{
    use HasFactory, Notifiable;

    public $timestamps = false;
    protected $fillable = [
        'id',
        'first_name',
        'last_name',
        'email',
        'address',
        'password',
        'role_id',
    ];

    protected $hidden = [
        'password',
        'remember_token',
    ];

    protected $casts = [
        'email_verified_at' => 'datetime',
        'password' => 'hashed',
    ];

    public function opinions()
    {
        return $this->hasMany(Opinion::class);
    }

    public function role()
    {
        return $this->belongsTo(Role::class);
    }

    public function isAdmin() : bool
    {
        return $this->role && $this->role->name == 'admin';
    }
}
```

Usługi

Aplikacja korzysta z usług (services), które zawierają logikę biznesową, taką jak obliczanie dynamicznych cen filmów:

```
class MovieService
{
    public function calculateDynamicPrice(Movie $movie)
    {
        // Liczba wypożyczeń danego filmu
        $loansCount = Loan::whereHas('movies', function ($query) use ($movie) {
            $query->where('movie_id', $movie->id);
        })->count();

        // Liczba zakupów wersji premium danego filmu
        $premiumCount = PremiumMovie::where('movie_id', $movie->id)->count();

        // Progi popularności
        $highPopularityThreshold = 20;
        $mediumPopularityThreshold = 10;

        // Bazowa cena filmu
        $basePrice = $movie->price_day;

        // Dynamiczna zmiana ceny w zależności od popularności
        if ($loansCount + $premiumCount >= $highPopularityThreshold) {
            return $basePrice * 1.5; // Podwyżka o 50%
        } elseif ($loansCount + $premiumCount >= $mediumPopularityThreshold) {
            return $basePrice * 1.25; // Podwyżka o 25%
        } else {
            return $basePrice * 0.75; // Obniżka o 25%
        }
    }
}
```


Walidacja Danych

Laravel oferuje prosty sposób walidacji danych za pomocą form requestów. Przykład walidacji dla dodawania nowego filmu:

```
class AddMovieRequest extends FormRequest
{
    public function authorize()
    {
        return true;
    }

    public function rules()
    {
        return [
            'title' => 'required|string|max:255',
            'description' => 'required|string',
            'category_id' => 'required|integer',
            'director' => 'required|string|max:255',
            'release_year' => 'required|integer|min:0',
            'duration' => 'required|string|min:0|regex:/^\d+(\.\d{1,2})?$/ ',
            'rate' => 'required|numeric|min:0',
            'video_path' => 'required|string',
            'price_day' => 'required|numeric|min:0',
            'available' => 'required|in:dostępny,niedostępny',
            'img_path' => 'nullable|image|mimes:jpeg,png,jpg,gif|max:2048',
        ];
    }
}
```

Seedowanie Danych

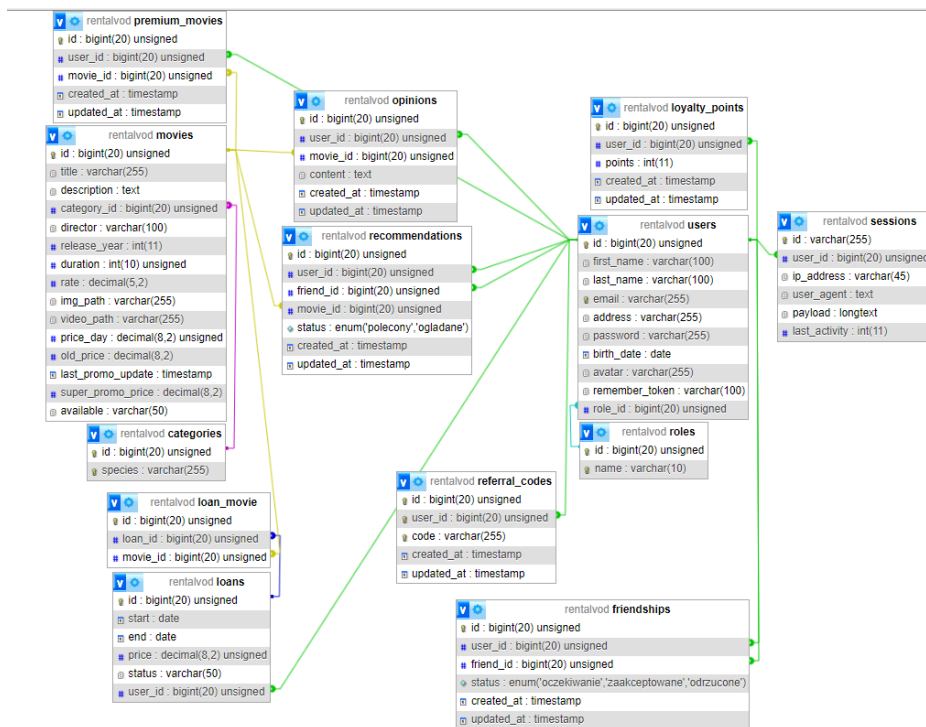
W celu załadowania przykładowych danych do bazy, używamy seederów. Przykład seedera:

```
class CategoriesSeeder extends Seeder
{
    public function run()
    {
        Category::insert([
            ['species' => 'Akcja'],
            ['species' => 'Komedia'],
            ['species' => 'Dramat'],
            ['species' => 'Fantasy'],
            ['species' => 'Horror'],
            ['species' => 'Romans'],
            ['species' => 'Przygodowy'],
            ['species' => 'Thriller'],
            ['species' => 'Naukowa-fikcja'],
            ['species' => 'Dokumentalny'],
            ['species' => 'Animacja'],
            ['species' => 'Biograficzny'],
            ['species' => 'Historyczny'],
            ['species' => 'Muzyczny'],
            ['species' => 'Wojenny'],
            ['species' => 'Western'],
            ['species' => 'Sportowy'],
            ['species' => 'Kryminał'],
            ['species' => 'Mystery'],
            ['species' => 'Rodzinny']
        ]);
    }
}
```

Każdy z powyższych elementów technologicznych i narzędzi jest starannie zintegrowany w aplikacji, co zapewnia jej sprawne działanie, łatwość utrzymania oraz możliwość dalszego rozwoju.

3. Baza danych

Diagram ERD



Opis rozwiązań zastosowanych w bazie

WYPOŻYCZALNIA FILMÓW VOD: Wypożyczalnia, dla której przygotowano bazę, nosi nazwę RentalVOD.

MOVIES: W tabeli movies przechowujemy wszystkie nasze filmy z najważniejszymi danymi o nich. Kolumna available wskazuje, czy film jest dostępny do wypożyczenia ("dostępny") czy jest obecnie wypożyczony ("niedostępny"). Dodatkowo tabela zawiera kolumny takie jak title, description, director, release_year, price_day, img_path, category_id.

USERS: W tabeli users przechowujemy informacje o użytkownikach, którzy założyli konto na naszej platformie. Tabela zawiera szczegółowe dane użytkowników, takie jak first_name, last_name, email, address, city, password, role_id (odpowiedzialna za uprawnienia użytkowników), avatar, loyalty_points.

LOANS: W tabeli loans przechowujemy informacje o wypożyczeniach, w tym datę rozpoczęcia (start_date), datę zakończenia (end_date), status wypożyczenia (status) oraz cenę wypożyczenia (price). Kolumna user_id przechowuje identyfikator użytkownika dokonującego wypożyczenia.

LOAN_MOVIE: Jest to tabela pośrednicząca łącząca tabelę loans i movies, co umożliwia przechowywanie informacji o tym, które filmy zostały wypożyczone w ramach danego wypożyczenia. Tabela zawiera kolumny loan_id i movie_id, które tworzą powiązania między tymi tabelami.

OPINIONS: Tabela opinions przechowuje opinie użytkowników o filmach, które wypożyczyli. Tabela zawiera kolumny takie jak content (treść opinii), movie_id i user_id, co pozwala na powiązanie opinii z odpowiednim filmem i użytkownikiem.

CATEGORIES: Tabela categories przechowuje informacje o kategoriach filmów. Kolumna species zawiera nazwę kategorii.

PREMIUM_MOVIES: Tabela w której są przechowywane informacje na temat filmów które zostały ulepszone do jakości premium przez użytkownika. Ma ona user_id i movie_id oraz timestampy, poprzez co można zobaczyć kiedy użytkownik zakupił premium.

REFERRAL_CODES: W tej tabeli są takie atrybuty jak user_id, code czy timestampy, poprzez co każdy użytkownik ma swój kod promocyjny dzięki któremu, jeżeli ktoś się zarejestruje na stronie otrzyma 20 punktów na konto.

LOYALTY_POINTS: Tabela w której są zapisywane punkty lojalnościowe konta, dzięki czemu użytkownik może wymienić je za jakość premium czy zakupić film. W tej tabeli są atrybuty typu: user_id, points czy timestampy.

RECOMMENDATIONS: Jeśli chodzi o tą tabelę to przedstawia ona polecane filmy poprzez znajomych. Ma ona atrybuty takie jak: user_id, friend_id, movie_id, status i timestampy.

FRIENDSHIPS: Tabela zawiera user_id, friend_id, status. Jest to tabela, która służy do dodawania znajomych.

Migracje

Migracje w Laravelu służą do zarządzania strukturą bazy danych. Pozwalają na tworzenie, modyfikowanie i usuwanie tabel oraz innych elementów bazy danych przy użyciu kodu PHP, co ułatwia zarządzanie bazą danych w trakcie rozwoju projektu.

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
use Illuminate\Support\Facades\DB;
return new class extends Migration
{
    public function up(): void
    {
        Schema::create('movies', function (Blueprint $table) {
            $table->id();
            $table->string('title', 255);
            $table->text('description');
            $table->unsignedBigInteger('category_id');
            $table->foreign('category_id')->references('id')->on('categories');
            $table->string('director', 100);
            $table->year('release_year');
            $table->integer('duration');
            $table->decimal('rate', 5, 2);
            $table->string('img_path', 255)->nullable();
            $table->string('video_path', 255);
            $table->decimal('price_day', 8, 2);
            $table->string('available', 50)->default('dostępny');
        });
    }

    public function down(): void
    {
        Schema::dropIfExists('movies');
    }
};
```

Każda migracja dziedziczy po klasie Migration i zawiera dwie metody: up() i down(). Metoda up() jest wywoływana podczas uruchamiania migracji i służy do utworzenia

struktury bazy danych. Metoda `down()` jest wywoływana podczas wycofywania migracji i służy do usunięcia utworzonych wcześniej elementów.

Seedery

Seedery w Laravelu służą do wypełniania tabel w bazie danych przykładowymi danymi. Są one przydatne podczas testowania i rozwijania aplikacji, pozwalając na szybkie załadowanie danych początkowych.

```
<?php

namespace Database\Seeders;

use App\Models\Movie;
use Illuminate\Database\Seeder;

class MoviesSeeder extends Seeder
{
    public function run(): void
    {
        Movie::insert([
            [
                'title' => 'Incepcja',
                'description' => 'Złodziej, który włamuje się do umysłów, ma za zadanie zasadzić ideę w umyśle osoby.',
                'category_id' => 8,
                'director' => 'Christopher Nolan',
                'release_year' => 2010,
                'duration' => 148,
                'rate' => 8.8,
                'img_path' => 'img/incepcja.jpg',
                'video_path' => 'V0HD9XEInc8',
                'price_day' => 15.00,
                'available' => 'dostępny'
            ],
            [
                'title' => 'Interstellar',
                'description' => 'Astronauta próbuje przetrwać na Marsie po zostaniu uznany za martwego.',
                'category_id' => 9,
                'director' => 'Christopher Nolan',
                'release_year' => 2014,
                'duration' => 169,
                'rate' => 8.6,
                'img_path' => 'img/interstellar.jpg',
                'video_path' => 'zShdZVxtX7E',
                'price_day' => 15.00,
                'available' => 'dostępny'
            ]
        ]);
    }
}
```

Seedery są przydatne w sytuacjach, gdy potrzebujemy przykładowych danych do testowania, inicjalizacji systemu lub tworzenia danych demonstracyjnych. Dzięki nim można łatwo wypełnić bazę danych wartościami początkowymi, co przyspiesza proces rozwoju aplikacji.

Opis Przykładowej Relacji w Modelach

Pomiędzy tabelami `movies` (filmy) i `loans` (wypożyczenia) w projekcie występuje relacja wiele do wielu (`many-to-many`) za pośrednictwem tabeli `loan_movie`. Taka relacja umożliwia, że jeden film może być wypożyczany w ramach wielu wypożyczeń, a jedno wypożyczenie może obejmować wiele filmów.

```
public function movies()
{
    return $this->belongsToMany(Movie::class, 'loan_movie', 'loan_id', 'movie_id');
}
```

W modelu `Movie` zdefiniowano metodę `loans()`, która ustanawia powiązanie wiele do wielu z modelem `Loan`. Wykorzystuje ona metodę `belongsToMany()` i określa nazwę tabeli pośredniczącej `loan_movie`. Dodatkowo, ustala nazwy kluczy obcych dla modelu `Movie` i `Loan`.

```
public function loans()
{
    return $this->belongsToMany(Loan::class, 'loan_movie', 'movie_id', 'loan_id');
}
```

W modelu Loan również zdefiniowano metodę movies(), która ustanawia powiązanie wiele do wielu z modelem Movie. Ta metoda także korzysta z metody belongsToMany() i określa nazwę tabeli pośredniczącej loan_movie. Podobnie jak w przypadku modelu Movie, określone są nazwy kluczy obcych dla modelu Loan i Movie.

Dzięki zdefiniowaniu tych relacji, można łatwo uzyskać powiązane filmy dla danego wypożyczenia poprzez dostęp do atrybutu movies na obiekcie Loan. Podobnie, można uzyskać wypożyczenia powiązane z danym filmem poprzez dostęp do atrybutu loans na obiekcie Movie.

W tabeli pośredniczącej loan_movie przechowywane są klucze obce dla wypożyczeń (loan_id) i filmów (movie_id), które tworzą powiązania między nimi.

```
return new class extends Migration
{
    public function up()
    {
        Schema::create('loan_movie', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('loan_id');
            $table->unsignedBigInteger('movie_id');
            $table->foreign('loan_id')->references('id')->on('loans')->onDelete('cascade');
            $table->foreign('movie_id')->references('id')->on('movies')->onDelete('cascade');
        });
    }

    public function down(): void
    {
        Schema::dropIfExists('loan_movie');
    }
};
```

Relacja wiele do wielu umożliwia elastyczne odwzorowanie przypadków, w których jeden obiekt może być powiązany z wieloma innymi obiektami, a także odwrotnie. Ta relacja jest szczególnie przydatna w przypadku modelowania związku między filmami a wypożyczeniami, gdzie film może być wypożyczany w ramach różnych wypożyczeń, a jedno wypożyczenie może dotyczyć wielu filmów.

4. GUI

Opis Interfejsu Użytkownika

Interfejs użytkownika (GUI) aplikacji "Wypożyczalnia Filmów VOD" został zaprojektowany z myślą o łatwości użytkowania i estetyce. Aplikacja wykorzystuje **Blade** jako silnik szablonów, co umożliwia dynamiczne generowanie widoków HTML i integrację z backendem Laravel. Całość interfejsu została zbudowana z użyciem nowoczesnych technologii front-endowych, takich jak Bootstrap do stylizacji oraz JavaScript do obsługi dynamicznych elementów interfejsu.

Główne Widoki Aplikacji

1. Strona Główna (home.blade.php)

Strona główna jest pierwszym widokiem, który użytkownik widzi po wejściu na stronę aplikacji. Zawiera ona:

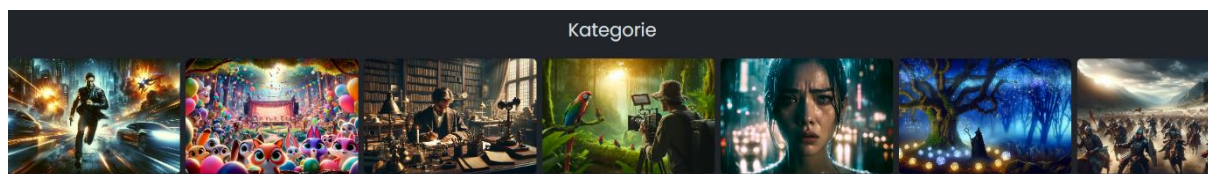
- **Nagłówek:** Zawiera logo aplikacji oraz nawigację umożliwiającą przejście do różnych sekcji strony.



- **Karuzela:** Dynamiczny slider wyświetlający promowane filmy.



- **Kategorie:** Sekcja wyświetlająca różne kategorie filmów dostępnych w wypożyczalni, z możliwością filtrowania filmów według kategorii.



- **Lista filmów:** Prezentacja wybranych filmów, które są aktualnie dostępne do wypożyczenia. Filmy są wyświetlane w postaci kart z obrazkiem, tytułem, kategorią, reżyserem, rokiem premiery i oceną.



- **Slider z 10 najpopularniejszymi filmami na stronie:** Sekcja przedstawiająca 10 filmów na tle zdjęć tych filmów z przedstawieniem miejsca filmu oraz jego dostępności czy nazwie.



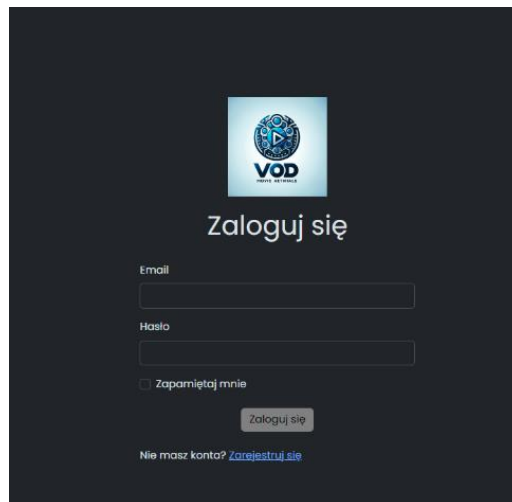
- **Sekcja filmów rekomendowanych:** Sekcja przedstawiająca filmy, które są najczęściej polecane przez użytkowników.



2. Strona Logowania (login.blade.php) i Rejestracji (register.blade.php)

Te widoki umożliwiają użytkownikom logowanie się do aplikacji oraz rejestrację nowych kont:

- **Formularz Logowania:** Umożliwia użytkownikom wprowadzenie adresu e-mail i hasła w celu zalogowania się do swojego konta.



The login form features the VOD logo at the top, followed by the title "Zaloguj się". It includes input fields for "Email" and "Hasło", a checkbox for "Zapamiętaj mnie", and a "Zaloguj się" button. A link "Nie masz konta? Zarejestruj się" is located at the bottom.

- **Formularz Rejestracji:** Umożliwia nowym użytkownikom utworzenie konta poprzez podanie takich informacji jak imię, nazwisko, adres e-mail, adres, miasto i hasło.



The registration form features the VOD logo at the top, followed by the title "Zarejestruj się". It includes input fields for "Imię", "Nazwisko", "Email", "Adres", "Hasło", "Potwierdź hasło", and "Kod polecający (opcjonalnie)". A "Zarejestruj" button is located at the bottom.

3. Panel Administratora (admin.blade.php)

Panel administratora jest dostępny tylko dla użytkowników z rolą administratora i zawiera narzędzia do zarządzania aplikacją:

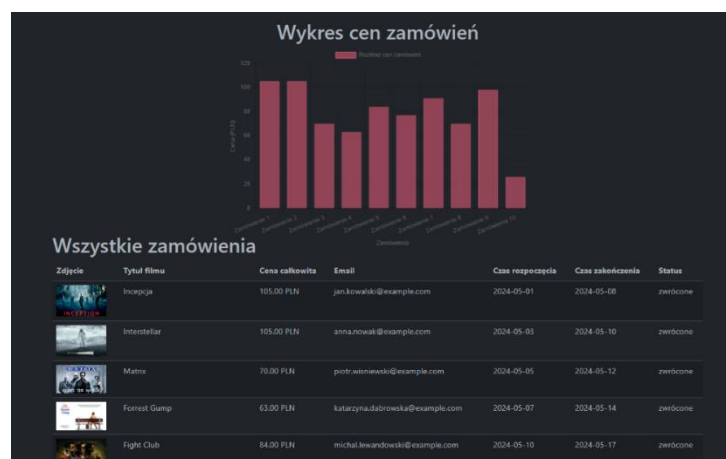
- **Zarządzanie użytkownikami:** Widok umożliwiający administratorowi przeglądanie, edytowanie i usuwanie kont użytkowników.

Wszyscy klienci							
ID	Avatar	Imię	Nazwisko	Email	Adres	Hasło	Akcje
1	Avatar	Jan	Kowalski	jan.kowalski@example.com	ul. Słoneczna 12, Warszawa	*****	Edytuj Usuń
2		Anna	Nowak	anna.nowak@example.com	ul. Główna 23, Kraków	*****	Edytuj Usuń
3		Piotr	Wiśniewski	piotr.wisniewski@example.com	ul. Krótka 5, Gdańsk	*****	Edytuj Usuń
4		Katarzyna	Dąbrowska	katarzyna.dabrowska@example.com	ul. Leśna 10, Wrocław	*****	Edytuj Usuń
5		Michał	Lewandowski	michal.lewandowski@example.com	ul. Morska 34, Szczecin	*****	Edytuj Usuń
6		Agnieszka	Kamińska	agnieszka.kaminska@example.com	ul. Wysoka 45, Łódź	*****	Edytuj Usuń
7		Tomasz	Zieliński	tomasz.zielinski@example.com	ul. Akacjowa 56, Poznań	*****	Edytuj Usuń
8		Marta	Kowalczyk	marta.kowalczyk@example.com	ul. Parkowa 67, Katowice	*****	Edytuj Usuń
9		Jakub1	Wojciechowski1	jakub.wojciechowski@example.com	ul. Ogrodowa 78, Gdynia1	*****	Edytuj Usuń
11		Marek	Woźniak	marek.wozniak@example.com	ul. Kwiatowa 90, Lublin	*****	Edytuj Usuń

- **Zarządzanie filmami:** Widok umożliwiający dodawanie nowych filmów, edytowanie istniejących oraz usuwanie filmów z bazy danych.

Wszystkie filmy											
ID	Zdjęcie	Tytuł	Opis	ID Kategorii	Reżyser	Rok Produkcji	Czas Filmu	Ocena	Średnia do filmu	Cena na Giełdzie	Dostępność
1		Incepcja	Złodziej, który włamuje się do umysłów, ma to zadanie zrealizować: idea w umyśle człowieka.	8	Christopher Nolan	2010	149	8.80	7.91	4.10	dostępny
2		Interstellar	Astronauta próbuje przetrwać na kosmicznej stacji, aby uratować umysł swojego zmarłego.	9	Christopher Nolan	2014	169	8.60	7.91	13.83	dostępny
3		Matriks	Historia komputerowego człowieka, który odkrywa prawdę o świecie i próbuje go zmienić.	9	Keanu Reeves	1999	136	8.70	7.91	10.00	dostępny
4		Forrest Gump	Historia człowieka, który przetrwał wiele trudnych wydarzeń i stał się wielkim sukcesem.	3	Robert Zemeckis	1994	142	8.80	7.91	9.00	dostępny

- **Zarządzanie zamówieniami:** Widok pozwalający na przeglądanie i zarządzanie zamówieniami złożonymi przez użytkowników.



- **Ustawienia admina:** Widok służący do włączania i wyłączania promocji, oraz ustawienia algorytmu filmów rekomendowanych w sekcji.

Ustawienia rekomendacji

Minimalna ocena filmu:

4

Minimalna liczba rekomendacji:

0

Zaktualizuj Zasadę

Ustawienia promocji

Włącz Promocje

4. Strona Profilu Użytkownika (profile.blade.php)

Strona profilu użytkownika pozwala użytkownikom na przeglądanie i edytowanie swoich danych oraz historii wypożyczeń:

- **Dane użytkownika:** Sekcja wyświetlająca podstawowe informacje o użytkowniku oraz umożliwiającą ich edytowanie.

Ustawienia konta

Ulica

ul. Słoneczna 12, Warszawa

Zapisz zmiany adresu

Zmień awatar

Wybierz plik Nie wybrano pliku

Zaktualizuj awatar

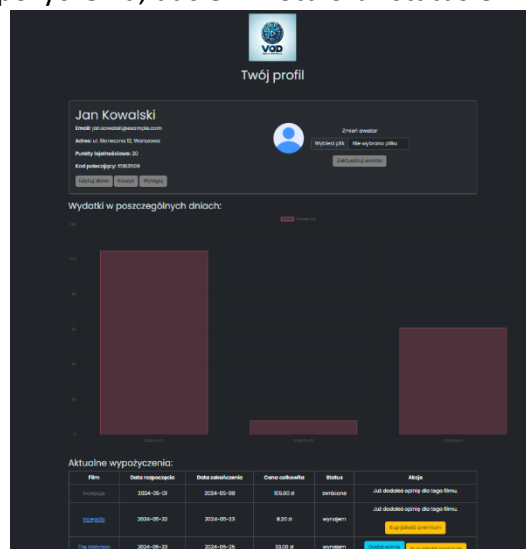
Obecne Hasło

Nowe Hasło

Potwierdzenie Nowego Hasła

Zmień hasło

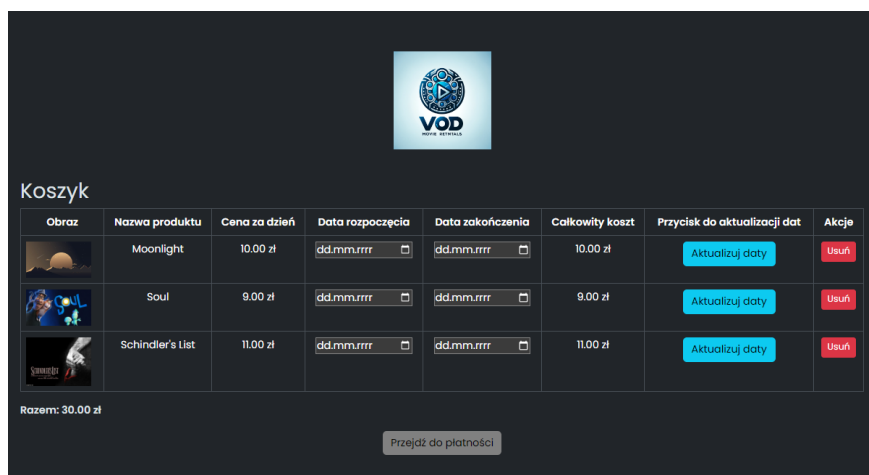
- **Historia wypożyczeń:** Lista filmów wypożyczonych przez użytkownika, z informacjami o dacie wypożyczenia, dacie zwrotu oraz statusie wypożyczenia.






5. Koszyk (cart.blade.php)

Widok koszyka umożliwia użytkownikom przeglądanie filmów dodanych do koszyka, zarządzanie datami wypożyczeń oraz finalizowanie zamówienia:

- **Lista filmów w koszyku:** Wyświetla wszystkie filmy dodane do koszyka wraz z ich ceną i wybranymi datami wypożyczenia.

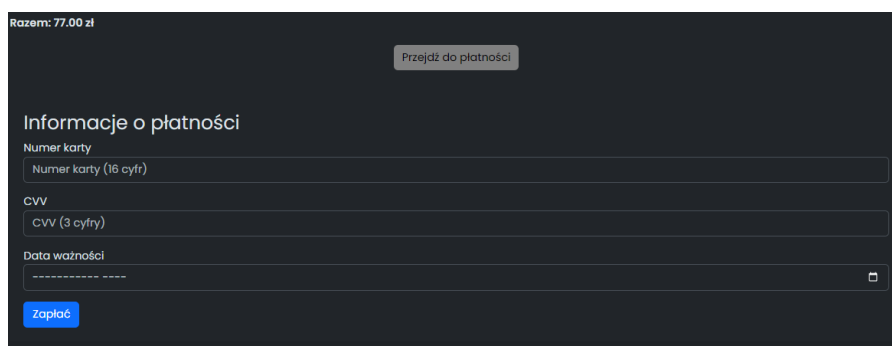


Obraz	Nazwa produktu	Cena za dzień	Data rozpoczęcia	Data zakończenia	Całkowity koszt	Przycisk do aktualizacji dat	Akcje
	Moonlight	10.00 zł	dd.mm.rrrr	dd.mm.rrrr	10.00 zł	Aktualizuj daty	Usuń
	Soul	9.00 zł	dd.mm.rrrr	dd.mm.rrrr	9.00 zł	Aktualizuj daty	Usuń
	Schindler's List	11.00 zł	dd.mm.rrrr	dd.mm.rrrr	11.00 zł	Aktualizuj daty	Usuń

Razem: 30.00 zł

Przejdź do płatności

- **Przycisk finalizacji zamówienia:** Umożliwia przejście do procesu zamawiania i opłacenia wypożyczenia.



Razem: 77.00 zł

Przejdź do płatności

Informacje o płatności

Numer karty
Numer karty (16 cyfr)

CVV
CVV (3 cyfry)

Data ważności

Zapłać

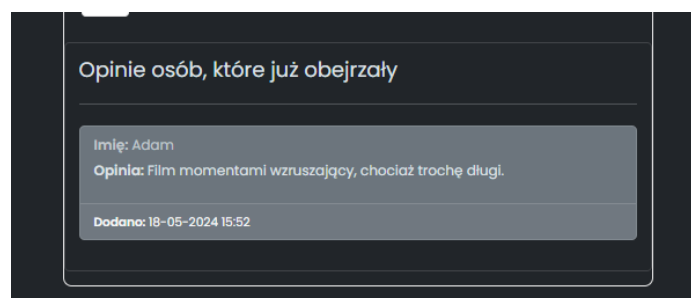
6. Strona Filmu (show.blade.php)

Widok szczegółowy filmu wyświetla pełne informacje o wybranym filmie:

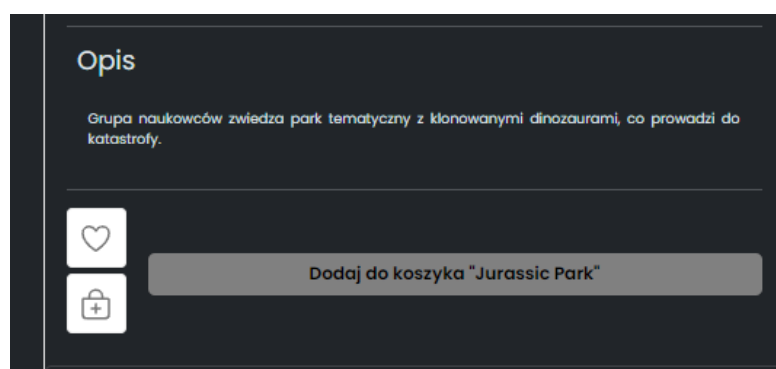
- **Informacje o filmie:** Zawiera tytuł, opis, reżysera, rok premiery, ocenę, kategorię oraz cenę wypożyczenia.



- **Opinie użytkowników:** Sekcja wyświetlająca opinie innych użytkowników, którzy wypożyczyli film, wraz z możliwością dodania własnej opinii.

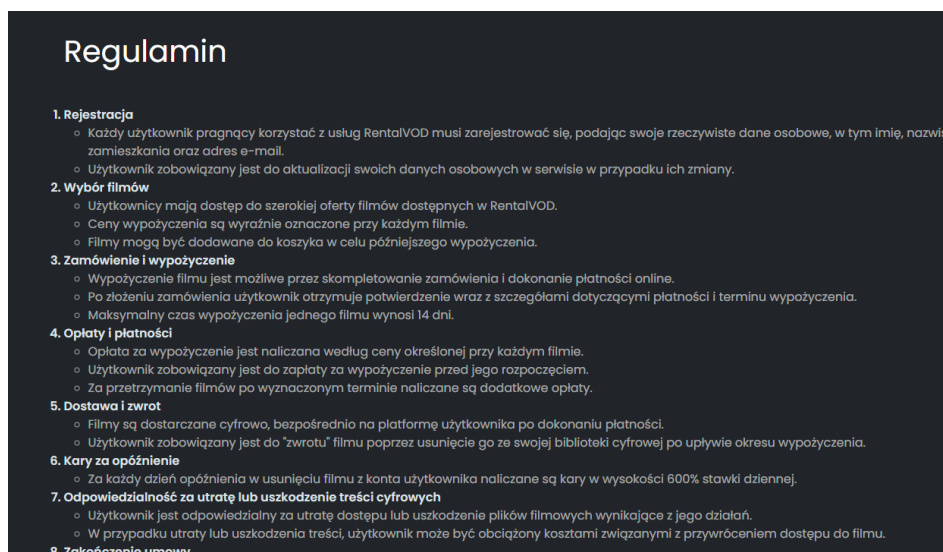


- **Przycisk dodania do koszyka:** Umożliwia dodanie filmu do koszyka w celu wypożyczenia.



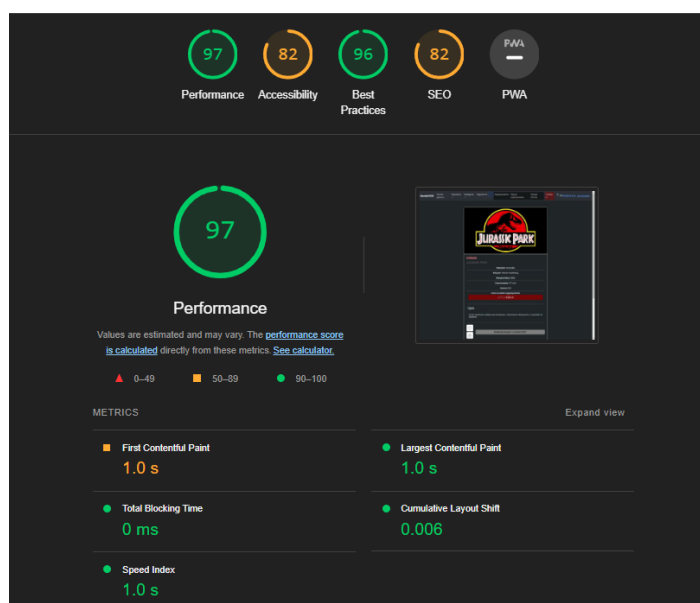
7. Strona regulaminu (regulamin.blade.php)

Widok strony na której jest zaprezentowany regulamin całej strony zawierający najważniejsze informacje.



Stylizacja i Responsywność

Aplikacja jest w pełni responsywna, co oznacza, że interfejs dostosowuje się do różnych rozmiarów ekranów, od komputerów stacjonarnych po urządzenia mobilne. Stylizacja została wykonana z użyciem **Bootstrap**, co zapewnia spójny i estetyczny wygląd wszystkich elementów interfejsu.



Dynamiczne Elementy

- **JavaScript:** Wykorzystany do obsługi dynamicznych elementów, takich jak karuzele, formularze, przyciski akcji i inne interaktywne komponenty.
- **Animacje:** Subtelne animacje dodają dynamiki do interfejsu, poprawiając doświadczenie użytkownika.

5. Uruchomienie aplikacji

Wymagania

1. **Pakiet XAMPP:** Pakiet XAMPP zawiera serwer Apache, bazę danych MySQL oraz interpreter PHP, które są niezbędne do uruchomienia aplikacji Laravel. Możesz pobrać XAMPP ze strony: [XAMPP Download](#).
2. **Composer:** Composer to narzędzie do zarządzania zależnościami w projekcie Laravel. Używane jest do instalacji wszystkich niezbędnych bibliotek i zależności frameworka. Możesz pobrać Composer ze strony: [Composer Download](#).
3. **Visual Studio Code:** Visual Studio Code to popularne środowisko programistyczne, które ułatwia edycję kodu. Możesz pobrać Visual Studio Code ze strony: [VS Code Download](#).

Kroki przy pierwszym uruchomieniu

1. Uruchom XAMPP:

- Po zainstalowaniu XAMPP, uruchom go i włącz moduły MySQL oraz Apache.
- XAMPP zapewnia graficzny interfejs, który pozwala na łatwe zarządzanie usługami serwera.

2. Przejdź do folderu projektu i uruchom plik start.bat:

- W folderze projektu znajduje się plik **start.bat**, który automatyzuje proces konfiguracji i uruchomienia aplikacji.
- Otwórz folder projektu i uruchom plik **start.bat** jako administrator, jeśli używasz innego systemu nie szkodzi jest plik start.sh dla macOS i linux (jeśli używasz tych systemów wejdź w drzewie katalogu na poprawną ścieżkę i wpisz w konsoli „bash start.sh”). Skrypt ten wykonuje następujące kroki:

```
%systemDrive%\xampp\mysql\bin\mysql -uroot -e "CREATE DATABASE IF NOT EXISTS rentalVOD;"

if %errorlevel% neq 0 msg %username% "Nie udało się utworzyć bazy danych." && exit /b %errorlevel%

php -r "copy('.env.example', '.env');"

call composer install

@REM call composer update

call composer require stripe/stripe-php

call composer require bensampo/laravel-enum

call composer require hoa/ruler

call composer require --dev barryvdh/laravel-ide-helper

call php artisan key:generate

call php artisan storage:link

@REM call php artisan migrate

@REM call php artisan db:seed

call php artisan migrate:fresh --seed

call php artisan ide-helper:generate

call php artisan serve

start http://127.0.0.1:8000

code .
```

- Skrypt ten wykonuje następujące operacje:
 - Tworzy bazę danych **rentalVOD**.
 - Kopiuje plik konfiguracyjny **.env.example** do **.env**.
 - Instaluje wszystkie zależności za pomocą **composer install**.
 - Wymaga dodatkowe pakiety takie jak **stripe/stripe-php** oraz **bensampo/laravel-enum**, również **hoa/ruler**
 - Generuje klucz aplikacji za pomocą **php artisan key:generate**.
 - Tworzy symbolic link dla przechowywania plików za pomocą **php artisan storage:link**.
 - Migruje bazę danych oraz seedy początkowe za pomocą **php artisan migrate:fresh --seed**.
 - Instaluje pakiet **barryvdh/laravel-ide-helper** dla ulepszenia doświadczenia programistycznego.
 - Uruchamia lokalny serwer deweloperski za pomocą **php artisan serve**.
 - Otwiera przeglądarkę na adresie <http://127.0.0.1:8000>.
 - Otwiera projekt w Visual Studio Code.

3. Uruchom wbudowany serwer deweloperski:

- Po uruchomieniu pliku **start.bat**, aplikacja powinna być dostępna pod adresem <http://127.0.0.1:8000>.

Kroki przy kolejnych uruchomieniach

1. Przejdź do katalogu projektu:

- Otwórz terminal lub wiersz poleceń.
- Przejdź do katalogu, w którym znajduje się projekt Laravel.

2. Uruchom wbudowany serwer deweloperski:

- Uruchom serwer deweloperski za pomocą komendy:

```
PS C:\Users\poczt\Desktop\OneDrive - Uniwersytet Rzeszowski\Studia Dawida\STUDIA\IV SEMESTR\Aplikacje internetowe 1 (AI)\PROJEKT\rentalVOD> php artisan serve
[INFO] Server running on [http://127.0.0.1:8000].
Press Ctrl+C to stop the server
```

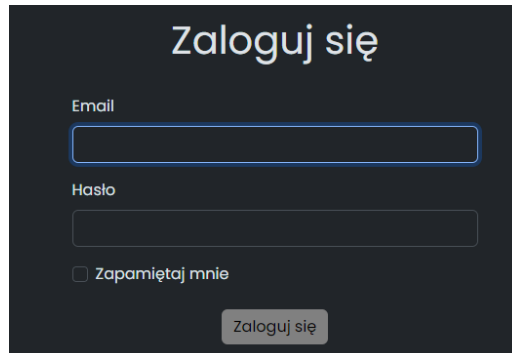
- Aplikacja powinna być dostępna pod adresem <http://127.0.0.1:8000>.

6. Funkcjonalności Aplikacji

Logowanie i Rejestracja

Logowanie:

- Na stronie logowania użytkownik zostaje poproszony o wprowadzenie swojego adresu e-mail oraz hasła.



Forma logowania z tytułem "Zaloguj się". Zawiera pola tekstowe dla "Email" i "Hasło", checkbox "Zapamiętaj mnie" oraz przycisk "Zaloguj się".

- Wprowadzone dane są walidowane pod kątem poprawności formatu i weryfikowane w bazie danych, aby sprawdzić, czy istnieje konto użytkownika o podanych danych uwierzytelniających.

```
public function login()
{
    if (Auth::check()) {
        logger('Użytkownik jest już zalogowany.');
```

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

```
    }
    return view('auth.login');
```

```
}

public function authenticate(AuthenticateRequest $request)
{
    $credentials = $request->validated();

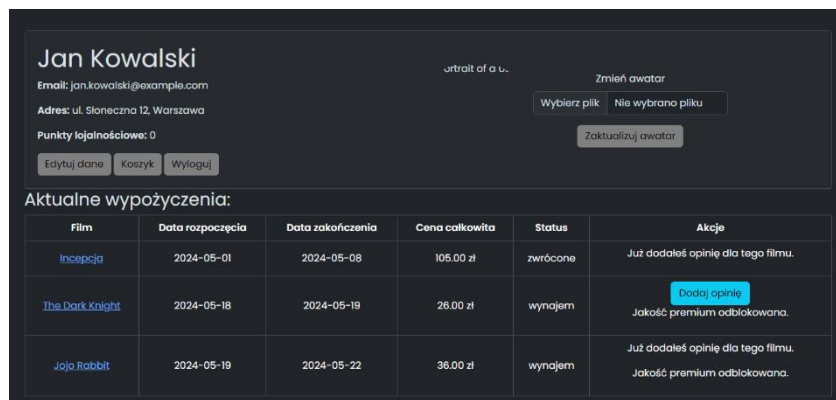
    $remember = $request->has('remember');
```

```
    if (Auth::attempt($credentials, $remember)) {
        $request->session()->regenerate();
        return redirect()->intended('home');
```

```
    } else {
        return back()->withErrors(['email' => 'Nieprawidłowe dane logowania.'])->onlyInput('email');
```

```
    }
}
```

- Jeśli podane dane są poprawne, użytkownik jest przekierowany do chronionej części aplikacji, gdzie ma dostęp do swojego konta.



Profil użytkownika Jan Kowalski. Zawiera informacje o użytkowniku, punkty lojalnościowe, przyciski do edycji danych i wylogowania, oraz tabelę aktualnych wypożyczeń.

Film	Data rozpoczęcia	Data zakończenia	Cena całkowita	Status	Akcje
Inocencja	2024-05-01	2024-05-08	105.00 zł	zwrócone	Już dodałeś opinię dla tego filmu.
The Dark Knight	2024-05-18	2024-05-19	26.00 zł	wynajem	Dodaj opinię Jakość premium odblokowana.
Jojo Rabbit	2024-05-19	2024-05-22	36.00 zł	wynajem	Już dodałeś opinię dla tego filmu. Jakość premium odblokowana.

- Jeśli podane dane są nieprawidłowe, użytkownik otrzymuje komunikat o błędzie i proszony jest o ponowne wprowadzenie danych logowania.

Rejestracja:

- Na stronie rejestracji użytkownik jest proszony o podanie swojego adresu e-mail, imienia, nazwiska, poprawnego adresu i hasła.

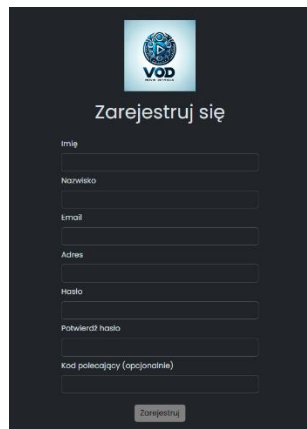
```
public function showRegistrationForm()
{
    return view('auth.register');
}

public function register(RegisterRequest $request)
{
    $validatedData = $request->validated();

    $user = User::create([
        'first_name' => $validatedData['first_name'],
        'last_name' => $validatedData['last_name'],
        'email' => $validatedData['email'],
        'password' => bcrypt($validatedData['password']),
        'address' => $validatedData['address'],
        'role_id' => 2 // Przypisuje nowym użytkownikom rolę klienta, ponieważ rola nr.1 to admin.
    ]);

    Auth::login($user);
    return redirect()->route('home');
}
```

- Podane dane są walidowane pod kątem poprawności formatu, a także sprawdzane pod kątem unikalności, np. czy wybrany adres e-mail jest już zajęty przez innego użytkownika.



- Podane informacje są zapisywane w bazie danych wraz z danymi logowania użytkownika.
- Po pomyślnej rejestracji, użytkownik jest automatycznie logowany do aplikacji i przekierowany na stronę główną.

Przykładowi użytkownicy:

- Administrator:**
 - E-mail: jan.kowalski@example.com
 - Hasło: Admin12345&
- Zwykły użytkownik:**
 - E-mail: anna.nowak@example.com
 - Hasło: password12345&A

```
User::insert([
    ['first_name' => 'Jan', 'last_name' => 'Kowalski', 'email' => 'jan.kowalski@example.com', 'address' => 'ul. Słoneczna 12, Warszawa', 'password' => Hash::make('Admin12345&'), 'role_id' => $adminRoleId],
    ['first_name' => 'Anna', 'last_name' => 'Nowak', 'email' => 'anna.nowak@example.com', 'address' => 'ul. Główna 23, Kraków', 'password' => Hash::make('password12345&A'), 'role_id' => $userRoleId],
]);
```

Przeglądanie Zasobów

Strona Główna:

- Strona główna wyświetla promowane filmy w formie karuzeli.



- Zawiera sekcje z popularnymi kategoriami filmów oraz losowo wybranymi filmami dostępnymi do wypożyczenia.



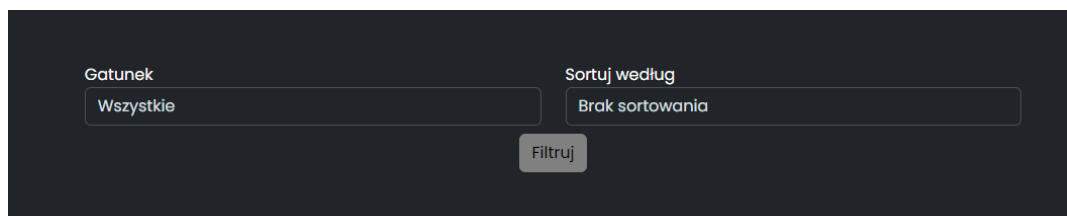
```
public function show($id)
{
    $movie = Movie::with(['category', 'opinions.user'])->where('id', $id)->firstOrFail();
    $promoPrice = $this->movieService->calculateDynamicPrice($movie);

    $movie->old_price = $promoPrice;
    $movie->save();

    return view('movies.show', compact('movie', 'promoPrice'));
}
```

Katalog Filmów:

- Użytkownicy mogą przeglądać pełny katalog filmów, korzystając z filtrów i opcji sortowania według kategorii, roku premiery, oceny itp.



```

public function index(Request $request)
{
    $query = Movie::with('category');

    if ($request->has('species') && $request->species != '') {
        $query->whereHas('category', function ($q) use ($request) {
            $q->where('species', $request->species);
        });
    }

    if ($request->has('category')) {
        $query->where('category_id', $request->category);
    }

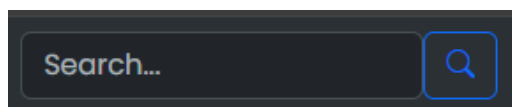
    switch ($request->sort_by) {
        case 'release1':
            $query->orderBy('release_year', 'asc');
            break;
        case 'release2':
            $query->orderBy('release_year', 'desc');
            break;
        case 'rate1':
            $query->orderBy('rate', 'asc');
            break;
        case 'rate2':
            $query->orderBy('rate', 'desc');
            break;
        case 'length1':
            $query->orderBy('duration', 'desc');
            break;
        case 'length2':
            $query->orderBy('duration', 'asc');
            break;
    }

    $movies = $query->paginate(6);

    foreach ($movies as $movie) {
        $promoPrice = $this->movieService->calculateDynamicPrice($movie);
        $movie->old_price = $promoPrice;
        $movie->save();
    }
}

```

- Możliwość przeszukiwania filmów na podstawie słów kluczowych.



```

public function search(Request $request)
{
    $query = $request->input('query');
    $movies = Movie::where('available', 'dostępny')
        ->where(function($queryBuilder) use ($query) {
            $queryBuilder->where('title', 'LIKE', '%' . $query . '%')
                ->orWhere('director', 'LIKE', '%' . $query . '%')
                ->orWhere('description', 'LIKE', '%' . $query . '%')
                ->orWhereHas('category', function($queryBuilder) use ($query) {
                    $queryBuilder->where('species', 'LIKE', '%' . $query . '%');
                });
        })
        ->paginate(10);
    return view('movies.index', compact('movies'));
}

```

Szczegóły Filmu:

- Każdy film ma stronę szczegółową, która wyświetla pełne informacje o filmie, takie jak opis, reżyser, rok premiery, ocena oraz dostępność.

THRILLER

Incepcja

Gatunek: Thriller

Reżyser: Christopher Nolan

Rok premiery: 2010

Czas trwania: 148 min

Ocena: 8.80

Cena za dzień wypożyczenia:

8,4375 zł 11,25 zł

Opis

Złodziej, który włamuje się do umysłów, ma za zadanie zasadzić ideę w umyśle osoby.

```
public function index()
{
    $movies = Movie::where('available', 'dostępny')
        ->inRandomOrder()
        ->limit(6)
        ->with('category')
        ->get();

    foreach ($movies as $movie) {
        $promoPrice = $this->movieService->calculateDynamicPrice($movie);
        $movie->old_price = $promoPrice;
        $movie->save();
    }

    return view('home', compact('movies'));
}
```

- Użytkownicy mogą dodawać opinie oraz przeglądać opinie innych użytkowników.

Opinie osób, które już obejrzały

Imię: Jan

Opinia: Bardzo dobry film, świetna obsada i fabuła.

Dodano: 09-05-2024 15:52

```

class OpinionController extends Controller
{
    public function store(StoreOpinionRequest $request)
    {
        $validatedData = $request->validated();

        $existingOpinion = Opinion::where('movie_id', $validatedData['movie_id'])
            ->where('user_id', auth()->id())
            ->first();

        if ($existingOpinion) {
            return back()->with('error', 'Możesz dodać tylko jedną opinię dla tego filmu.');
        }

        $opinion = new Opinion();
        $opinion->movie_id = $validatedData['movie_id'];
        $opinion->user_id = auth()->id();
        $opinion->content = $validatedData['content'];
        $opinion->save();

        return back()->with('success', 'Opinia została dodana.');
    }
}

```

Regulamin Strony:

- Strona z regulaminem, który określa zasady korzystania z serwisu oraz warunki wypożyczeń.

Regulamin

- Rejestracja**
 - Każdy użytkownik pragnący korzystać z usług RentalVOD musi zarejestrować się, podając swoje rzeczywiste dane osobowe, w tym imię, nazwisko, adres zamieszkania oraz adres e-mail.
 - Użytkownik zobowiązany jest do aktualizacji swoich danych osobowych w serwisie w przypadku ich zmiany.
- Wybór filmów**
 - Użytkownicy mają dostęp do szerokiej oferty filmów dostępnych w RentalVOD.
 - Ceny wypożyczenia są wyraźnie oznaczone przy każdym filmie.
 - Filmy mogą być dodawane do koszyka w celu późniejszego wypożyczenia.
- Zamówienie i wypożyczenie**
 - Wypożyczenie filmu jest możliwe przez skompletowanie zamówienia i dokonanie płatności online.
 - Po złożeniu zamówienia użytkownik otrzymuje potwierdzenie wraz z szczegółami dotyczącymi płatności i terminu wypożyczenia.
 - Maksymalny czas wypożyczenia jednego filmu wynosi 14 dni.
- Oplaty i płatności**
 - Opłata za wypożyczenie jest naliczana według ceny określonej przy każdym filmie.
 - Użytkownik zobowiązany jest do zapłaty za wypożyczenie przed jego rozpoczęciem.
 - Za przetrzymanie filmów po wyznaczonym terminie naliczane są dodatkowe opłaty.
- Dostawa i zwrot**
 - Filmy są dostarczane cyfrowo, bezpośrednio na platformę użytkownika po dokonaniu płatności.
 - Użytkownik zobowiązany jest do "zwrotu" filmu poprzez usunięcie go ze swojej biblioteki cyfrowej po upływie okresu wypożyczenia.
- Kary za opóźnienie**
 - Za każdy dzień opóźnienia w usunięciu filmu z konta użytkownika naliczane są kary w wysokości 600% stawki dziennej.
- Odpowiedzialność za utratę lub uszkodzenie treści cyfrowych**
 - Użytkownik jest odpowiedzialny za utratę dostępu lub uszkodzenie plików filmowych wynikające z jego działań.
 - W przypadku utraty lub uszkodzenia treści, użytkownik może być obciążony kosztami związanymi z przywróceniem dostępu do filmu.
- Zakończenie umowy**
 - Użytkownik może zakończyć umowę z RentalVOD w dowolnym momencie, z zastrzeżeniem uregulowania wszelkich zobowiązań finansowych.
- Zmiany w regulaminie**
 - RentalVOD zastrzega sobie prawo do wprowadzania zmian w regulaminie. Użytkownicy zostaną poinformowani o wszelkich zmianach przez aktualizacje na stronie internetowej.
- Postanowienia końcowe**
 - W przypadku sporów decydujące jest prawo obowiązujące w jurysdykcji siedziby firmy.
 - RentalVOD nie ponosi odpowiedzialności za szkody wynikłe z użytkowania serwisu poza zakresem gwarancji usługodawcy.

Zarządzanie przez Administratora

CRUD Filmów:

- **Dodawanie filmów:** Administrator może dodawać nowe filmy do bazy danych, wypełniając formularz z informacjami o filmie.
- **Edycja filmów:** Administrator może edytować istniejące filmy, aktualizując ich dane.
- **Usuwanie filmów:** Administrator może usuwać filmy z bazy danych.

Wszytkie filmy

Dodaj film

Dodaj kategorię

ID	Zdjęcie	Tytuł	Opis	ID Kategorii	Reżyser	Rok Produkcji	Czas Filmu	Ocena	Ścieżka do Filmu	Cena za Dzień	Dostępność	Akcje
1		Incepcja	Złodziej, który włamuje się do umysłów, ma za zadanie zasadzić ideę w umyśle osoby.	8	Christopher Nolan	2010	148	8.80	YoHD9XEinc0	11.25	dostępny	Edytuj Usuń
2		Interstellar	Astronauta próbuje przetrwać na Marsie po zostaniu uznany za martwego.	9	Christopher Nolan	2014	169	8.60	zSWdZVtXT7E	11.25	dostępny	Edytuj Usuń
3		Matrix	Haker komputerowy odkrywa prawdziwą naturę swojej rzeczywistości i swoją rolę w wojnie przeciwko jej kontrolerom.	9	Lana Wachowski	1999	136	8.70	vkQl3bBAly8	7.5	dostępny	Edytuj Usuń
4		Forrest Gump	Prosta historia mężczyzny, który przypadkiem bierze udział w ważnych wydarzeniach historycznych.	3	Robert Zemeckis	1994	142	8.80	b1vqoHBptjg	6.75	dostępny	Edytuj Usuń
5		Fight Club	Mężczyzna walczy przeciwko swojemu alter-ego w brutalnych walkach na pięści.	8	David Fincher	1999	139	8.80	SUXWAEX2jlg	9	dostępny	Edytuj Usuń

```

public function editUser($id) {
    $user = User::findOrFail($id);
    return view('admin.editUser', compact('user'));
}

public function updateUser(UpdateUserRequest $request, $id) {
    $validatedData = $request->validated();
    $user = User::findOrFail($id);
    $user->update($validatedData);

    $user->role_id = $request->has('admin') ? 1 : 2;
    $user->save();

    return redirect()->route('admin.users')->with('success', 'Dane użytkownika zostały zaktualizowane.');
```

```

}

public function deleteUser($id) {
    try {
        User::destroy($id);
        return redirect()->route('admin.users')->with('success', 'Użytkownik został usunięty.');
```

```

    } catch (\Exception $e) {
        return redirect()->route('admin.users')->with('error', 'Nie udało się usunąć użytkownika.');
```

```











    }
}

```

Zarządzanie Użytkownikami:

- **Dodawanie i edycja użytkowników:** Administrator może dodawać nowych użytkowników oraz edytować dane istniejących użytkowników.
- **Usuwanie użytkowników:** Administrator może usuwać konta użytkowników.

Wszyscy klienci

ID	Avatar	Imię	Nazwisko	Email	Adres	Hasło	Akcje
1		Jan	Kowalski	jan.kowalski@example.com	ul. Stoneczna 12, Warszawa	*****	Edytuj Usuń
2		Anna	Nowak	anna.nowak@example.com	ul. Główna 23, Kraków	*****	Edytuj Usuń
3		Piotr	Wiśniewski	piotr.wisniewski@example.com	ul. Krótka 5, Gdańsk	*****	Edytuj Usuń
4		Katarzyna	Dąbrowska	katarzyna.dabrowska@example.com	ul. Leśna 10, Wrocław	*****	Edytuj Usuń
5		Michał	Lewandowski	michal.lewandowski@example.com	ul. Morska 34, Szczecin	*****	Edytuj Usuń
6		Agnieszka	Kamińska	agnieszka.kaminska@example.com	ul. Wysoka 45, Łódź	*****	Edytuj Usuń
7		Tomasz	Zieliński	tomasz.zielinski@example.com	ul. Akacjowa 56, Poznań	*****	Edytuj Usuń
8		Marta	Kowalczyk	marta.kowalczyk@example.com	ul. Parkowa 67, Katowice	*****	Edytuj Usuń
9		Jakub1	Wojciechowski1	jakub.wojciechowski@example.com	ul. Ogrodowa 78, Gdynia1	*****	Edytuj Usuń
11		Marek	Wozniak	marek.wozniak@example.com	ul. Kwiatowa 90, Lublin	*****	Edytuj Usuń

< 1 2 >

```
public function addMovie(AddMovieRequest $request)
{
    $data = $request->all();

    if ($request->hasFile('img_path')) {
        $imagePath = $request->file('img_path')->store('public/img');
        $imagePath = $request->file('img_path')->store('img');
        $data['img_path'] = 'img/' . basename($imagePath);
    } else {
        $data['img_path'] = 'default.jpg';
    }

    $category = Category::find($request->category_id);
    if (!$category) {
        $category = new Category();
        $category->id = $request->category_id;
        $category->save();
    }

    Movie::create(array_merge($data, ['category_id' => $category->id]));

    return redirect()->route('admin.movies')->with('success', 'Film został dodany pomyślnie.');
```

```
public function deleteMovie($id)
{
    Movie::destroy($id);
    return redirect()->route('admin.movies')->with('success', 'Film został usunięty.');
```

```
public function updateMovie(UpdateMovieRequest $request, $id)
{
    $movie = Movie::findOrFail($id);
    $data = $request->all();

    if ($request->hasFile('img_path')) {
        Storage::delete($movie->img_path);
        $imagePath = $request->file('img_path')->store('public/img');
        $imagePath = $request->file('img_path')->store('img');
        $data['img_path'] = 'img/' . basename($imagePath);
    }

    $category = Category::find($request->category_id);
    if (!$category) {
        $category = new Category();
        $category->id = $request->category_id;
        $category->save();
    }

    $movie->update(array_merge($data, ['category_id' => $category->id]));

    return redirect()->route('admin.movies')->with('success', 'Film został zaktualizowany.');
```

Zarządzanie Wypożyczeniami:

- **Przeglądanie zamówień:** Administrator może przeglądać wszystkie zamówienia złożone przez użytkowników, zarządzać ich statusem oraz szczegółami wypożyczeń.
- **Obsługa zgłoszeń problemów:** Administrator ma dostęp do panelu zgłoszeń, gdzie może przeglądać i rozwiązywać problemy zgłoszone przez użytkowników.



```
public function userOrders($id) {  
    $user = User::with('loans.movies')->findOrFail($id);  
    return view('admin.userOrders', compact('user'));  
}  
  
public function orders() {  
    $loans = Loan::with(['user', 'movies'])->paginate(10);  
    return view('admin.orders', compact('loans'));  
}
```

Funkcjonalności dla Zalogowanych Użytkowników

Profil Użytkownika:

- Zalogowani użytkownicy mają dostęp do swojego profilu, gdzie mogą przeglądać swoje dane, historię wypożyczeń oraz zarządzać swoimi zamówieniami.
- Możliwość edycji danych osobowych i zmiany hasła.

Jan Kowalski

Email: jan.kowalski@example.com

Adres: ul. Słoneczna 12, Warszawa

Punkty lojalnościowe: 0

Edytuj dane Koszyk Wyloguj

Uprzejmie witamy

Zmień avatar

Wybierz plik Nie wybrano pliku

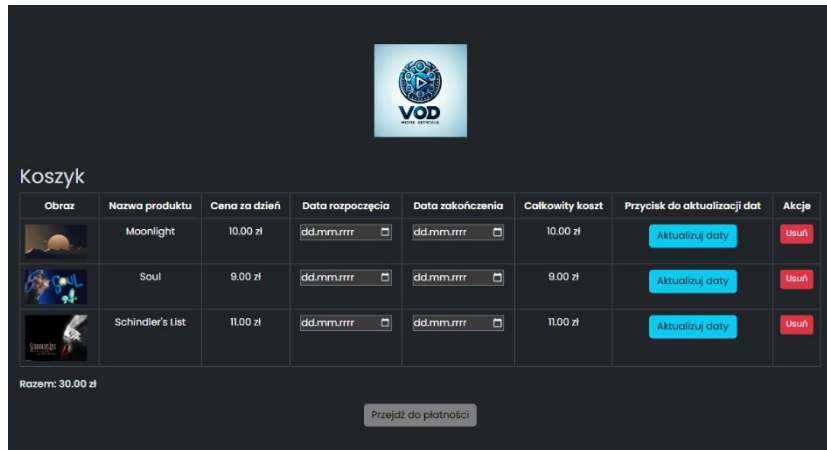
Zaktualizuj avatar




Aktualne wypożyczenia:

Film	Data rozpoczęcia	Data zakończenia	Cena całkowita	Status	Akcje
Incepcja	2024-05-01	2024-05-08	105.00 zł	zwrócone	Już dodałeś opinię dla tego filmu.
The Dark Knight	2024-05-18	2024-05-19	26.00 zł	wynajem	<button>Dodaj opinię</button> Jakość premium odblokowana.
Jojo Rabbit	2024-05-19	2024-05-22	36.00 zł	wynajem	Już dodałeś opinię dla tego filmu. Jakość premium odblokowana.

Koszyk:

- Użytkownicy mogą dodawać filmy do koszyka, zarządzać datami wypożyczeń oraz finalizować zamówienia.
- Koszyk wyświetla listę filmów, łączną cenę oraz daty wypożyczenia.



Obraz	Nazwa produktu	Cena za dzień	Data rozpoczęcia	Data zakończenia	Całkowity koszt	Przycisk do aktualizacji dat	Akcje
	Moonlight	10.00 zł	<input type="text" value="dd.mm.yyyy"/>	<input type="text" value="dd.mm.yyyy"/>	10.00 zł	<button>Aktualizuj daty</button>	<button>Usuń</button>
	Soul	9.00 zł	<input type="text" value="dd.mm.yyyy"/>	<input type="text" value="dd.mm.yyyy"/>	9.00 zł	<button>Aktualizuj daty</button>	<button>Usuń</button>
	Schindler's List	11.00 zł	<input type="text" value="dd.mm.yyyy"/>	<input type="text" value="dd.mm.yyyy"/>	11.00 zł	<button>Aktualizuj daty</button>	<button>Usuń</button>

Razem: 30.00 zł

Przejdź do płatności

```
public function addToCart(Request $request, $movie_id)
{
    $movie = Movie::findOrFail($movie_id);
    $cart = session()->get('cart', []);

    if (!isset($cart[$movie_id])) {
        $cart[$movie_id] = [
            "name" => $movie->title,
            "price" => $movie->price_day,
            "image" => asset('storage/' . $movie->img_path),
            "quantity" => 1,
            "totalCost" => $movie->price_day
        ];
    } else {
        $cart[$movie_id]['quantity'] += 1;
        $cart[$movie_id]['totalCost'] = $cart[$movie_id]['quantity'] * $movie->price_day;
    }

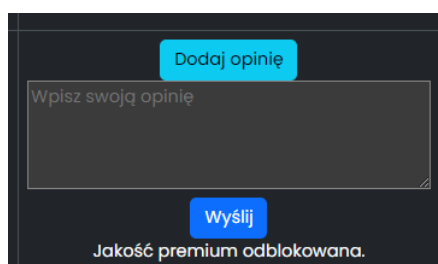
    session()->put('cart', $cart);
    return redirect()->route('cart.show')->with('success', 'Film dodany do koszyka.');
```

```
public function removeFromCart($movie_id)
{
    $cart = session('cart', []);
    if (isset($cart[$movie_id])) {
        unset($cart[$movie_id]);
        session(['cart' => $cart]);
    }

    return back()->with('success', 'Produkt został usunięty z koszyka.');
```

Opinie:

- Użytkownicy mogą dodawać opinie do wypożyczonych filmów, które będą widoczne dla innych użytkowników.
- Możliwość przeglądania i zarządzania własnymi opiniami.



Dodaj opinię

Wpisz swoją opinię

Wyślij

Jakość premium odblokowana.

Proces Wypożyczania Filmów

Dodawanie do Koszyka:

- Użytkownik dodaje filmy do koszyka, wybierając daty wypożyczenia.
- Koszyk wyświetla łączną cenę wypożyczenia na podstawie liczby dni i wybranych filmów.



```
public function checkout(Request $request)
{
    $cart = session('cart', []);
    if (empty($cart)) {
        return redirect()->route('cart.show')->with('error', 'Koszyk jest pusty.');
```

```
    }

    $user = Auth::user();
    $loyaltyPoints = $user->loyaltyPoints->points ?? 0;

    foreach ($cart as $id => $details) {
        if (empty($details['start']) || empty($details['end'])) {
            return back()->with('error', 'Niepoprawne daty wypożyczenia dla jednego z filmów.');
```

```
        }

        $startDate = new DateTime($details['start']);
        $endDate = new DateTime($details['end']);
        if ($endDate < $startDate) {
            return back()->with('error', 'Data zakończenia nie może być wcześniejsza niż data rozpoczęcia.');
```

```
        }

        $loan = new Loan();
        $loan->user_id = $user->id;
        $loan->start = $startDate->format('Y-m-d');
        $loan->end = $endDate->format('Y-m-d');
        $diff = $endDate->diff($startDate)->days + 1;

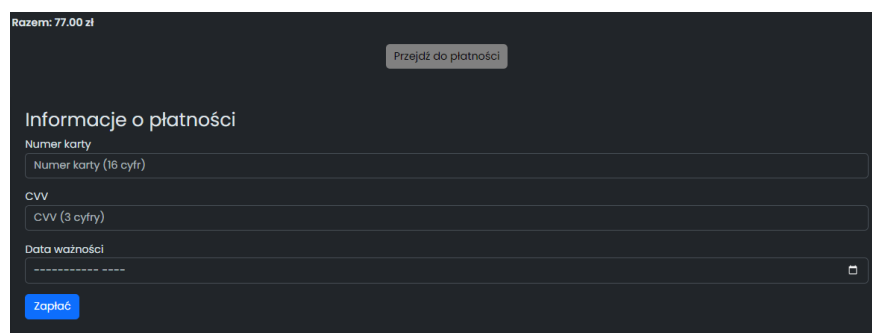
        $moviePrice = $details['price'];
        if ($loyaltyPoints >= 50) {
            $moviePrice = 0;
            $loyaltyPoints -= 50;
            $user->loyaltyPoints->points = $loyaltyPoints;
            $user->loyaltyPoints->save();
        }

        $loan->price = $moviePrice * $diff;
        $loan->status = StatusLoan::WYNAJEM;
        $loan->save();

        $loan->movies()->attach($id);
    }
}
```

Finalizacja Zamówienia:

- Po zatwierdzeniu koszyka użytkownik przechodzi do płatności.
- Płatność może być realizowana za pomocą punktów lojalnościowych lub kartą kredytową.



Historia Wypożyczeń:

- Użytkownik może przeglądać historię swoich wypożyczeń, sprawdzać status zamówień oraz szczegóły każdego wypożyczenia.

Jan Kowalski

Email: jan.kowalski@example.com

Adres: ul. Słoneczna 12, Warszawa

Punkty lojalnościowe: 0

Edytuj dane

Koszyk

Wyloguj

avatar of a u...

Zmień avatar

Wybierz plik

Nie wybrano pliku

Zaktualizuj avatar

Aktualne wypożyczenia:

Film	Data rozpoczęcia	Data zakończenia	Cena całkowita	Status	Akcje
Incepcja	2024-05-01	2024-05-08	105.00 zł	zwrócone	Już dodałeś opinię dla tego filmu.
The Dark Knight	2024-05-18	2024-05-19	28.00 zł	wynajem	Dodaj opinię Jakość premium odblokowana.
Jajo kabbitt	2024-05-19	2024-05-22	36.00 zł	wynajem	Już dodałeś opinię dla tego filmu. Jakość premium odblokowana.

```
public function showProfile()
{
    $user_id = Auth::id();
    $loans = Loan::with('movies')->where('user_id', $user_id)->paginate(3);
    return view('user.profile', compact('loans'));
}

public function showMovie($movie_id)
{
    $user_id = Auth::id();
    $movie = Movie::with(['loans' => function($query) use ($user_id) {
        $query->where('user_id', $user_id);
    }])->findOrFail($movie_id);

    $userHasAccess = $movie->loans->contains(function ($loan) {
        return $loan->status !== StatusLoan::ZWROCONO;
    });

    if ($userHasAccess) {
        return view('loans.show', compact('movie'));
    } else {
        return back()->with('error', 'Nie masz dostępu do tego filmu lub status wypożyczenia to "zwrócone.");
    }
}
```

7. Walidacja danych

W aplikacji "Wypożyczalnia Filmów VOD" walidacja danych jest kluczowym elementem zapewniającym poprawność i bezpieczeństwo operacji przeprowadzanych przez użytkowników. Laravel oferuje mechanizm walidacji, który pozwala na łatwe definiowanie reguł walidacji dla różnych żądań.

Walidacja dla różnych typów żądań

1. Dodawanie Kategorii (AddCategoryRequest)

- **Opis:** Żądanie dodania nowej kategorii filmowej.
- **Reguły walidacji:**
 - **genre:** wymagane, tekst, maksymalnie 255 znaków, unikalne w tabeli **categories** (kolumna **species**).
- **Kod:**

```
<?php
namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class AddCategoryRequest extends FormRequest
{
    public function authorize()
    {
        return true;
    }

    public function rules()
    {
        return [
            'genre' => 'required|string|max:255|unique:categories,species',
        ];
    }
}
```

2. Dodawanie Filmu (AddMovieRequest)

- **Opis:** Żądanie dodania nowego filmu do bazy danych.
- **Reguły walidacji:**
 - **title:** wymagane, tekst, maksymalnie 255 znaków.
 - **description:** wymagane, tekst.
 - **category_id:** wymagane, liczba całkowita.
 - **director:** wymagane, tekst, maksymalnie 255 znaków.
 - **release_year:** wymagane, liczba całkowita, minimalnie 0.
 - **duration:** wymagane, tekst, minimalnie 0, format liczbowy z opcjonalną częścią dziesiętną.
 - **rate:** wymagane, liczba, minimalnie 0.

- **video_path**: wymagane, tekst.
 - **price_day**: wymagane, liczba, minimalnie 0.
 - **available**: wymagane, jeden z wartości: **dostępny**, **niedostępny**.
 - **img_path**: opcjonalne, obrazek, formaty: jpeg, png, jpg, gif, maksymalnie 2048 kB.
- **Kod:**

```
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class AddMovieRequest extends FormRequest
{
    public function authorize()
    {
        return true;
    }

    public function rules()
    {
        return [
            'title' => 'required|string|max:255',
            'description' => 'required|string',
            'category_id' => 'required|integer',
            'director' => 'required|string|max:255',
            'release_year' => 'required|integer|min:0',
            'duration' => 'required|string|min:0|regex:/^\d+(\.\d{1,2})?$/ ',
            'rate' => 'required|numeric|min:0',
            'video_path' => 'required|string',
            'price_day' => 'required|numeric|min:0',
            'available' => 'required|in:dostępny,niedostępny',
            'img_path' => 'nullable|image|mimes:jpeg,png,jpg,gif|max:2048',
        ];
    }
}
```

3. Autoryzacja (AuthenticateRequest)

- **Opis**: Żądanie logowania użytkownika.
- **Reguły walidacji**:
 - **email**: wymagane, format email, maksymalnie 255 znaków.
 - **password**: wymagane, tekst, minimalnie 8 znaków, zawiera małe i duże litery, cyfry oraz znak specjalny.
- **Kod**:

```
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class AuthenticateRequest extends FormRequest
{
    public function authorize()
    {
        return true;
    }

    public function rules()
    {
        return [
            'email' => ['required', 'email', 'max:255'],
            'password' => ['required', 'string', 'min:8', 'regex:/[a-z]/', 'regex:/[A-Z]/', 'regex:/[0-9]/', 'regex:/[!@#$%&*?&#]/'],
        ];
    }
}
```

4. Zmiana Hasła (ChangePasswordRequest)

- **Opis:** Żądanie zmiany hasła użytkownika.
- **Reguły walidacji:**
 - **current_password:** wymagane.
 - **new_password:** wymagane, tekst, minimalnie 8 znaków, zawiera małe i duże litery, cyfry oraz znak specjalny, potwierdzenie hasła.
- **Komunikaty błędów:**
 - **new_password.min:** "Hasło musi zawierać co najmniej 8 znaków."
 - **new_password.confirmed:** "Potwierdzenie hasła nie zgadza się."
 - **new_password.regex:** "Hasło musi zawierać co najmniej jedną wielką literę, jedną małą literę, jedną cyfrę i jeden znak specjalny."
- **Kod:**

```
<?php
namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class ChangePasswordRequest extends FormRequest
{
    public function authorize()
    {
        return true;
    }

    public function rules()
    {
        return [
            'current_password' => 'required',
            'new_password' => 'required|string|min:8|confirmed|regex:/^(?=[a-z])(?=[A-Z])(?=\d)(?=.*[!%*?&])[A-Za-z\d@$!%*?&]{8,}$/',
        ];
    }

    public function messages()
    {
        return [
            'new_password.min' => 'Hasło musi zawierać co najmniej 8 znaków.',
            'new_password.confirmed' => 'Potwierdzenie hasła nie zgadza się.',
            'new_password.regex' => 'Hasło musi zawierać co najmniej jedną wielką literę, jedną małą literę, jedną cyfrę i jeden znak specjalny.'
        ];
    }
}
```

5. Rejestracja (RegisterRequest)

- **Opis:** Żądanie rejestracji nowego użytkownika.
- **Reguły walidacji:**
 - **first_name:** wymagane, tekst, maksymalnie 100 znaków, tylko litery.
 - **last_name:** wymagane, tekst, maksymalnie 100 znaków, tylko litery.
 - **email:** wymagane, format email, maksymalnie 255 znaków, unikalne w tabeli **users**.
 - **password:** wymagane, potwierdzenie hasła, minimalnie 8 znaków, zawiera małe i duże litery, cyfry oraz znak specjalny.
 - **address:** wymagane, tekst, maksymalnie 255 znaków.
- **Komunikaty błędów:**

- **first_name.required:** "Imię jest wymagane."
- **last_name.required:** "Nazwisko jest wymagane."
- **email.required:** "Adres email jest wymagany."
- **email.email:** "Proszę podać prawidłowy adres email."
- **email.unique:** "Podany adres email jest już używany."
- **password.required:** "Hasło jest wymagane."
- **password.confirmed:** "Hasła nie są identyczne."
- **password.min:** "Hasło musi zawierać co najmniej 8 znaków."
- **password.regex:** "Hasło musi zawierać małe i duże litery, cyfry oraz znak specjalny."
- **address.required:** "Adres jest wymagany."

- **Kod:**

```
<?php
namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class RegisterRequest extends FormRequest
{
    public function authorize()
    {
        return true;
    }

    public function rules()
    {
        return [
            'first_name' => 'required|string|max:100|alpha',
            'last_name' => 'required|string|max:100|alpha',
            'email' => 'required|email|max:255|unique:users,email',
            'password' => [
                'required',
                'confirmed',
                'min:8',
                'regex:/[a-z]/',
                'regex:/[A-Z]/',
                'regex:/[0-9]/',
                'regex:/[!@#$%&?&#&]/'
            ],
            'address' => 'required|string|max:255',
        ];
    }

    public function messages()
    {
        return [
            'first_name.required' => 'Imię jest wymagane.',
            'last_name.required' => 'Nazwisko jest wymagane.',
            'email.required' => 'Adres email jest wymagany.',
            'email.email' => 'Proszę podać prawidłowy adres email.',
            'email.unique' => 'Podany adres email jest już używany.',
            'password.required' => 'Hasło jest wymagane.',
            'password.confirmed' => 'Hasła nie są identyczne.',
            'password.min' => 'Hasło musi zawierać co najmniej 8 znaków.',
            'password.regex' => 'Hasło musi zawierać małe i duże litery, cyfry oraz znak specjalny.',
            'address.required' => 'Adres jest wymagany.',
        ];
    }
}
```

6. Dodawanie Opinii (StoreOpinionRequest)

- **Opis:** Żądanie dodania nowej opinii o filmie.
- **Reguły walidacji:**
 - **content:** wymagane, minimalnie 1 znak.
 - **movie_id:** wymagane, istnieje w tabeli **movies**.
- **Komunikaty błędów:**
 - **content.required:** "Treść opinii jest wymagana."
 - **content.min:** "Treść opinii musi mieć co najmniej 1 znak."
 - **movie_id.required:** "ID filmu jest wymagane."
 - **movie_id.exists:** "Podany film nie istnieje."
- **Kod:**

```
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class StoreOpinionRequest extends FormRequest
{
    public function authorize()
    {
        return true;
    }

    public function rules()
    {
        return [
            'content' => 'required|min:1',
            'movie_id' => 'required|exists:movies,id',
        ];
    }

    public function messages()
    {
        return [
            'content.required' => 'Treść opinii jest wymagana.',
            'content.min' => 'Treść opinii musi mieć co najmniej 1 znak.',
            'movie_id.required' => 'ID filmu jest wymagane.',
            'movie_id.exists' => 'Podany film nie istnieje.',
        ];
    }
}
```


7. Aktualizacja Adresu (UpdateAddressRequest)

- **Opis:** Żądanie aktualizacji adresu użytkownika.
- **Reguły walidacji:**
 - **address:** wymagane, tekst, maksymalnie 255 znaków, może zawierać tylko litery, cyfry, spacje oraz znaki ,.-.
- **Komunikaty błędów:**
 - **address.required:** "Adres jest wymagany."
 - **address.regex:** "Adres może zawierać tylko litery, cyfry, spacje oraz znaki ,.-."

- **Kod:**

```
<?php
namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class UpdateAddressRequest extends FormRequest
{
    public function authorize()
    {
        return true;
    }

    public function rules()
    {
        return [
            'address' => 'required|string|max:255|regex:/^[a-zA-Z0-9\s,.-]+$/',
        ];
    }

    public function messages()
    {
        return [
            'address.required' => 'Adres jest wymagany.',
            'address.regex' => 'Adres może zawierać tylko litery, cyfry, spacje oraz znaki ,.-',
        ];
    }
}
```

8. Aktualizacja Awatara (UpdateAvatarRequest)

- **Opis:** Żądanie aktualizacji awatara użytkownika.

- **Reguły walidacji:**

- **avatar:** wymagane, obrazek, formaty: jpeg, png, jpg, gif, svg, maksymalnie 2048 kB.

- **Komunikaty błędów:**

- **avatar.required:** "Musisz wybrać plik."
- **avatar.image:** "Plik musi być obrazem."
- **avatar.mimes:** "Dopuszczalne są tylko pliki w formatach: jpeg, png, jpg, gif, svg."
- **avatar.max:** "Maksymalny rozmiar pliku to 2048 kB."

- **Kod:**

```
<?php
namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class UpdateAvatarRequest extends FormRequest
{
    public function authorize()
    {
        return true;
    }

    public function rules()
    {
        return [
            'avatar' => 'required|image|mimes:jpeg,png,jpg,gif,svg|max:2048',
        ];
    }

    public function messages()
    {
        return [
            'avatar.required' => 'Musisz wybrać plik.',
            'avatar.image' => 'Plik musi być obrazem.',
            'avatar.mimes' => 'Dopuszczalne są tylko pliki w formatach: jpeg, png, jpg, gif, svg.',
            'avatar.max' => 'Maksymalny rozmiar pliku to 2048 kB.'
        ];
    }
}
```

9. Aktualizacja Koszyka (UpdateCartRequest)

- **Opis:** Żądanie aktualizacji dat wypożyczenia filmów w koszyku.
- **Reguły walidacji:**
 - **start:** wymagane, data.
 - **end:** wymagane, data, musi być późniejsza lub równa dacie **start**.
- **Komunikaty błędów:**
 - **start.required:** "Data rozpoczęcia jest wymagana."
 - **start.date:** "Data rozpoczęcia musi być prawidłową datą."
 - **end.required:** "Data zakończenia jest wymagana."
 - **end.date:** "Data zakończenia musi być prawidłową datą."
 - **end.after_or_equal:** "Data zakończenia nie może być wcześniejsza niż data rozpoczęcia."
- **Kod:**

```
<?php
namespace App\Http\Requests;
use Illuminate\Foundation\Http\FormRequest;
class UpdateCartRequest extends FormRequest
{
    public function authorize()
    {
        return true;
    }

    public function rules()
    {
        return [
            'start' => 'required|date',
            'end' => 'required|date|after_or_equal:start',
        ];
    }

    public function messages()
    {
        return [
            'start.required' => 'Data rozpoczęcia jest wymagana.',
            'start.date' => 'Data rozpoczęcia musi być prawidłową datą.',
            'end.required' => 'Data zakończenia jest wymagana.',
            'end.date' => 'Data zakończenia musi być prawidłową datą.',
            'end.after_or_equal' => 'Data zakończenia nie może być wcześniejsza niż data rozpoczęcia.',
        ];
    }
}
```

10. Aktualizacja Filmu (UpdateMovieRequest)

- **Opis:** Żądanie aktualizacji danych filmu.
- **Reguły walidacji:**
 - **title:** wymagane, tekst, maksymalnie 255 znaków.
 - **description:** wymagane, tekst.
 - **director:** wymagane, tekst, maksymalnie 255 znaków.

- **release_year**: wymagane, liczba całkowita, minimalnie 0.
- **duration**: wymagane, tekst, minimalnie 0, format liczbowy z opcjonalną częścią dziesiętną.
- **rate**: wymagane, liczba, minimalnie 0.
- **video_path**: wymagane, tekst.
- **price_day**: wymagane, liczba, minimalnie 0.
- **available**: wymagane, jeden z wartości: **dostępny**, **niedostępny**.
- **img_path**: opcjonalne, obrazek, formaty: jpeg, png, jpg, gif, maksymalnie 2048 kB.
- **Kod:**

```
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class UpdateMovieRequest extends FormRequest
{
    public function authorize()
    {
        return true;
    }

    public function rules()
    {
        return [
            'title' => 'required|string|max:255',
            'description' => 'required|string',
            'director' => 'required|string|max:255',
            'release_year' => 'required|integer|min:0',
            'duration' => 'required|string|min:0|regex:/^\d+(\.\d{1,2})?$/ ',
            'rate' => 'required|numeric|min:0',
            'video_path' => 'required|string',
            'price_day' => 'required|numeric|min:0',
            'available' => 'required|in:dostępny,niedostępny',
            'img_path' => 'nullable|image|mimes:jpeg,png,jpg,gif|max:2048',
        ];
    }
}
```

11. Aktualizacja Użytkownika (UpdateUserRequest)

- **Opis:** Żądanie aktualizacji danych użytkownika.
- **Reguły walidacji:**
 - **first_name:** wymagane, tekst, maksymalnie 100 znaków.
 - **last_name:** wymagane, tekst, maksymalnie 100 znaków.
 - **email:** wymagane, format email, maksymalnie 255 znaków, unikalne w tabeli **users** z wyjątkiem aktualizowanego użytkownika.
 - **address:** wymagane, tekst, maksymalnie 255 znaków.
- **Kod:**

```
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class UpdateUserRequest extends FormRequest
{
    public function authorize()
    {
        return true;
    }

    public function rules()
    {
        $userId = $this->route('id');

        return [
            'first_name' => 'required|string|max:100',
            'last_name' => 'required|string|max:100',
            'email' => [
                'required',
                'string',
                'email',
                'max:255',
                'unique:users,email,' . $userId,
                'regex:/^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/'
            ],
            'address' => 'required|string|max:255',
        ];
    }
}
```

Podsumowanie

Mechanizmy walidacji w aplikacji "Wypożyczalnia Filmów VOD" zapewniają poprawność danych wprowadzanych przez użytkowników na różnych etapach korzystania z aplikacji. Dzięki temu aplikacja działa sprawnie i bezpiecznie, minimalizując ryzyko błędów oraz zapewniając pozytywne doświadczenie użytkownikom.