|  |  |
| --- | --- |
| **AI1** | Dokumentacja projektu |
| **Autor** | KarolWojnar, 125148 |
| **Kierunek, rok** | Informatyka, II rok, st. stacjonarne (3,5-l) |
| **Temat  projektu** | *Platforma wypożyczania filmów VOD* |

# Spis treści

[Spis treści 1](#_Toc427987886)

[Wstęp 1](#_Toc90556347)

[Baza danych 1](#_Toc1837920307)

[Opis rozwiązań zastosowanych w bazie: 1](#_Toc727021494)

[Migracje, przykład: 3](#_Toc2003928655)

[Seedy do tabeli użytkowników(users): 5](#_Toc18891281)

[Narzędzia i technologie 6](#_Toc1183495116)

[Baza danych 6](#_Toc2032515914)

[Laravel 7](#_Toc792654277)

[Stripe 8](#_Toc1180417290)

[Uruchomienie aplikacji 9](#_Toc868329623)

[Wymagania 9](#_Toc595657374)

[Uruchomienie aplikacji 11](#_Toc598067982)

[Funkcjonalności 12](#_Toc219506285)

[Przeglądanie zasobów 13](#_Toc2001045219)

[Przykładowi użytkownicy: 14](#_Toc1892672257)

[Zasoby dostępne dla wszystkich 14](#_Toc99751506)

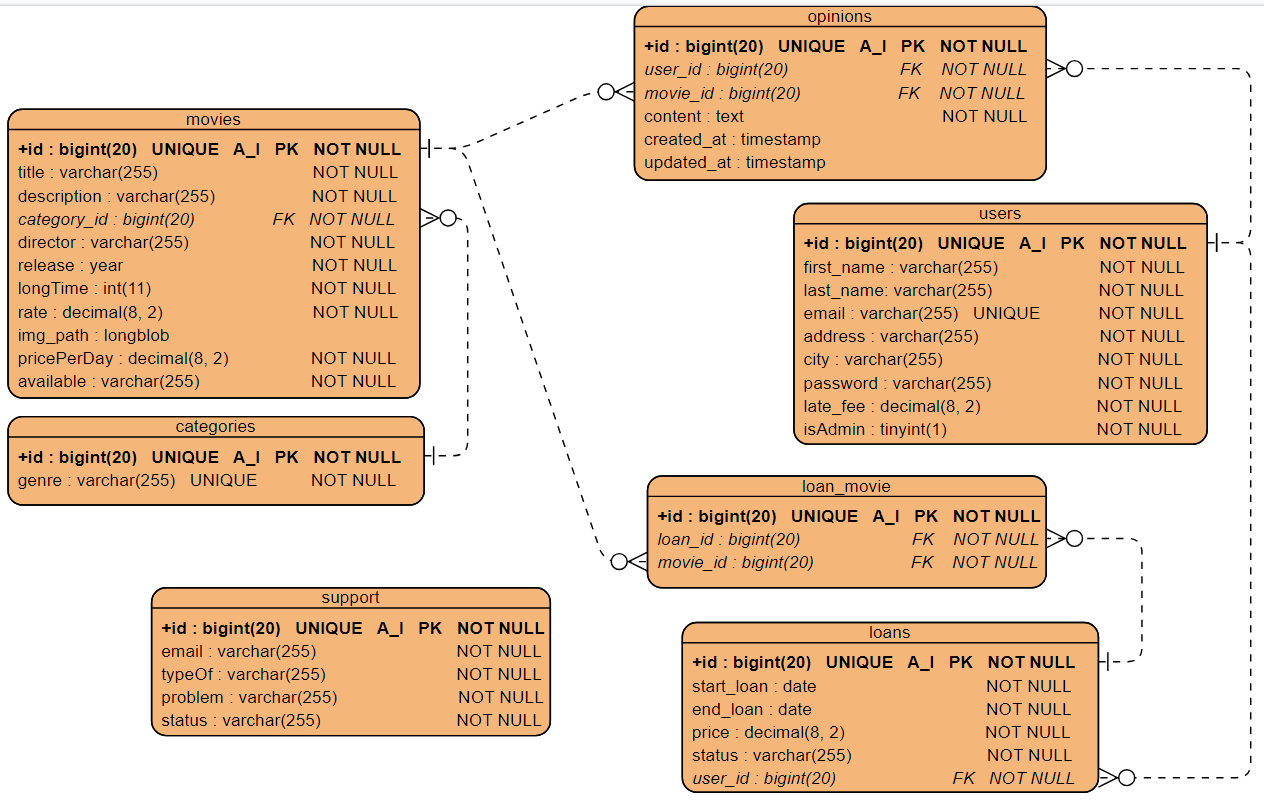
[Zasoby dostępne dla zalogowanego użytkownika 21](#_Toc144285187)

[CRUD administratora 29](#_Toc1548307190)

# Wstęp

Wypożyczalnia filmów blu ray opiera się na wysyłaniu fizycznych wersji filmów do klientów, którzy wcześniej opłacą wypożyczenie. Tylko zalogowani użytkownicy mogą dokonac wypożyczenia. Administrator będzie zarządzał statusami wysyłki, filmami oraz ma dostęp do kont uzytkowników i ich danych (poza hasłem), dodatkowo będzie zarządzał zgłoszeniami problemów przesłanych od odwiedzających stronę. Aplikacja mocno zautomatyzuje obsługę zamówień gdyż wiele rzeczy takich jak obliczanie kosztu wypożyczenia, koszt kary za spóźnione oddanie filmów czy zmiana dostępności filmów na stonie dzieje się automatycznie.

# Baza danych



## Opis rozwiązań zastosowanych w bazie:

WYPOŻYCZALNIA FILMÓW BLU RAY: wypożyczalnia dla której przygotowano bazę nosi nazwę CINEMA BLU-RAY.

MOVIES: w tabeli movies przechowujemy wszystkie nasze fimly z najważniejszymi danymi o nich, dodatkowo kolumna avalaible pokazuje czy film jest na wypożyczeniu ("niedostępny") czy jest dostępny do wypożyczenia ("dostępny").

USERS: w tabeli users przechowujemy informację o użytkownikach, którzy założyli konto na naszej platformie. W tabeli mam dokładne dane adresowe na potrzeby wysyłki filmów. Dodatkowo tabela przechowuje informacje o karach jakie został nałożone na danego użytkownika za spóźnione oddanie filmu("late\_fee").

LOANS: w tabeli loans przechowujemy informację o dacie startu i spodziewanego końca wypożyczenia na którego podstawie możemy dostosować odpowiednią karę dla użytkownika, którego id jest również przekazane do tej tabeli.

LOAN\_MOVIE: jest to tabela transferowa łącząca tabele loans i movies, jest to potrzebne ponieważ nie ma określonej liczby filmów którą użytkownik może na raz wypożyczyć.

OPINIONS: jest to tabela przechowująca opinie użytkowników o filmach które wypożyczyli. Opinie są widoczne później pod każdym filmem(o ile jakieś są). Tabela przechwouje id użytkownika który zostawił opinię oraz id filmu do jakiego zostałą napisana opinia.

SUPPORT: jest to tabela przechowująca problemy, jakie znaleźli odwiedzający stronę. Każdy kto wszedł na stronę i znalazł problem może przesłać nam zgłoszenie pozosatwiając po sobie maila.

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.

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".

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.

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.

## Migracje, przykład:



Ta migracja tworzy tabelę "users" w bazie danych. Tabela ta zawiera kolumny, takie jak "id" (identyfikator), "first\_name" (imię), "last\_name" (nazwisko), "email" (adres e-mail), "address" (adres), "city" (miasto), "password" (hasło), "late\_fee" (opłata za opóźnienie) i "isAdmin" (czy użytkownik ma uprawnienia administratora).

Migracje w Laravel służą do zarządzania strukturą bazy danych. Pozwalają na tworzenie, modyfikowanie i usuwanie tabel oraz innych elementów bazy danych poprzez wykorzystanie kodu PHP zamiast manualnych operacji na bazie danych.

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. Natomiast metoda "down()" jest wywoływana podczas wycofywania migracji i służy do usunięcia utworzonych wcześniej elementów.

Migracje są zapisywane w plikach PHP, które można uruchomić za pomocą narzędzia do migracji dostępnego w Laravel. Dzięki temu, struktura bazy danych może być łatwo zarządzana i kontrolowana wraz z postępem projektu.

W przypadku tej migracji, tworzona jest tabela "users", która będzie przechowywać informacje o użytkownikach aplikacji. Kolumny w tabeli są definiowane za pomocą obiektu Blueprint, który oferuje różne metody umożliwiające precyzyjne określanie typów kolumn, ich atrybutów, kluczy obcych itp.

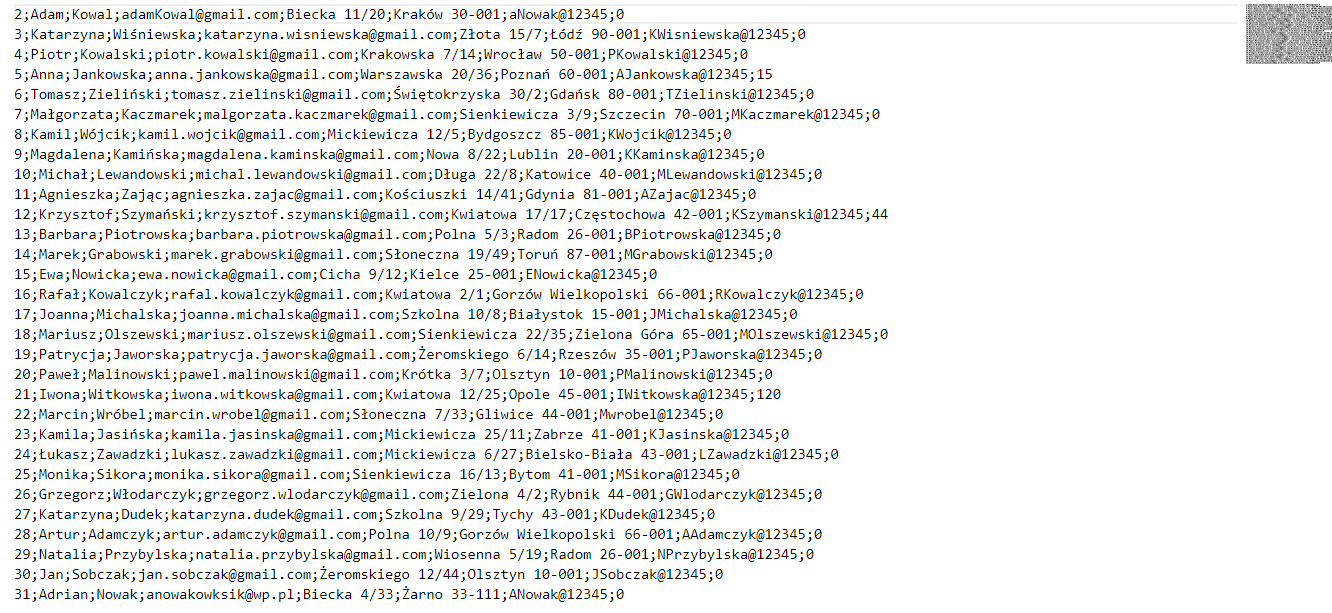
Migracje stanowią ważny element procesu wytwarzania oprogramowania w Laravel, umożliwiając deweloperom kontrolę nad strukturą bazy danych i zachowywanie jej w spójnym stanie podczas iteracji i rozwoju projektu.

## Seedy do tabeli użytkowników(users):

Ten Seeder (Seeder) służy do wypełniania tabeli "users" w bazie danych przykładowymi danymi. Seeder jest częścią mechanizmu Laravel, który umożliwia automatyczne wstawianie początkowych danych do bazy danych w celach testowych lub inicjalizacyjnych.

W przypadku tego seeder'a, najpierw jest dodawany rekord administratora do tabeli "users" za pomocą metody DB::table('users')->insert(). Wartości pól są podane bezpośrednio w kodzie, a hasło jest zabezpieczone za pomocą funkcji Hash::make().

Następnie seeder wczytuje dane z pliku CSV (users.csv):

i dodaje je do tabeli "users". Plik ten musi znajdować się w lokalizacji storage\_path('csv/users.csv'). Każdy wiersz w pliku jest podzielony na poszczególne wartości za pomocą funkcji str\_getcsv(). Odpowiednie wartości są wstawiane do tabeli "users" za pomocą metody DB::table('users')->insert().

Seedery w Laravelu służą do tworzenia danych początkowych w bazie danych. Pozwalają na automatyczne wypełnianie tabel danymi, co jest szczególnie przydatne podczas rozwijania aplikacji lub podczas testowania. Seedery mogą być uruchamiane za pomocą narzędzia do migracji lub specjalnej komendy w konsoli Laravel.

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.

# Narzędzia i technologie

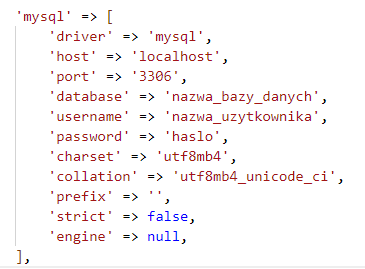
## Baza danych

Aplikacja wykorzystuje silnik bazodanowy MySQL w wersji 10.4.27-MariaDB. MySQL jest oprogramowaniem typu open source, dostępnym na licencji GNU General Public License (GPL). Możesz pobrać najnowszą wersję MySQL ze strony( <https://www.mysql.com/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).

Poniżej przedstawiono przykładową konfigurację połączenia z bazą danych MySQL:



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

Zmiana bazy danych

Jeśli chciałbyś zmienić silnik bazodanowy na inny niż MySQL, musisz zaktualizować ustawienia połączenia w pliku konfiguracyjnym config/database.php. W tym celu należy zmodyfikować sekcję connections i dostosować ją do nowego silnika bazodanowego.

Przy zmianie bazy danych zaleca się również sprawdzić dokumentację nowego silnika bazodanowego w celu uzyskania informacji dotyczących konkretnych ustawień i wymagań.

## Laravel

Laravel to popularny framework aplikacji webowych oparty na języku PHP. Jest on rozwijany przez Taylor Otwell i ma na celu zapewnienie wydajnego i eleganckiego środowiska do tworzenia aplikacji internetowych.

Laravel to zaawansowany framework PHP, który umożliwia szybkie i efektywne tworzenie skalowalnych aplikacji webowych. Wykorzystuje nowoczesne techniki programowania oraz wzorce projektowe, co ułatwia tworzenie czytelnego i łatwego w utrzymaniu kodu. Wersję Laravel, którą używam to: ^10.10.

Oficjalna dokumentacja Laravel jest dostępna na stronie <https://laravel.com/docs>. Możesz odwiedzić tę stronę, aby uzyskać pełną dokumentację Laravel, która zawiera szczegółowe informacje na temat funkcji, klas, interfejsów i sposobu użycia różnych komponentów frameworka.

Laravel jest udostępniany na podstawie licencji MIT. Oznacza to, że możesz swobodnie korzystać, modyfikować i rozpowszechniać framework. Licencja MIT pozwala również na integrację Laravela z innymi systemami i narzędziami open source.

Poniżej przedstawiam opis funkcji index() wraz z jej fragmentem kodu:



Funkcja index() jest częścią kontrolera i odpowiada za wyświetlenie listy filmów w widoku "movies.filter". Przed wyświetleniem listy, funkcja sprawdza, czy są jakieś nieopłacone wypożyczenia, których data początku wypożyczenia minęła. Jeśli takie wypożyczenia zostaną znalezione, filmy przypisane do tych wypożyczeń zostaną oznaczone jako "dostępne", a wypożyczenia zostaną usunięte. Następnie funkcja pobiera listę wszystkich filmów i przekazuje ją do widoku.

## Stripe

Stripe to narzędzie do obsługi płatności online, które zostało wykorzystane w projekcie. Pozwala ono na bezpieczne przetwarzanie płatności kartą kredytową. W aplikacji został zaimplementowany mechanizm integracji z API Stripe, który umożliwia użytkownikom dokonywanie płatności za usługi lub produkty. Poniżej przedstawiono przykładowy fragment kodu, w którym wykorzystano Stripe do przetwarzania płatności:



Ta funkcja przetwarza transakcję płatności za pomocą platformy Stripe. Oto krok po kroku opis jej działania:

1. Tworzony jest obiekt klienta Stripe poprzez wywołanie konstruktora klasy \Stripe\StripeClient i przekazanie do niego wartości tajnego klucza Stripe pobranego z pliku konfiguracyjnego (env('STRIPE\_SECRET')).
2. Pobierany jest token płatności (stripeToken) z przekazanego żądania ($request->input('stripeToken')). Token ten jest identyfikatorem karty płatniczej przesłanym przez formularz płatności.
3. Tworzony jest obiekt źródła płatności za pomocą metody sources->create() klienta Stripe. W tym przypadku jest to karta kredytowa (typ 'card') i token płatności jest przekazywany jako parametr.
4. Pobierana jest wartość całkowitej ceny (totalPrice) z przekazanego żądania ($request->input('totalPrice')), która reprezentuje kwotę do zapłaty za wypożyczenie filmu.
5. Wykonuje się blok try-catch, w którym rozpoczyna się tworzenie płatności przez utworzenie obiektu paymentIntent za pomocą metody paymentIntents->create() klienta Stripe. Przekazywane są w nim informacje o płatności, takie jak:
   1. amount: kwota płatności wyrażona w groszach (mnożona przez 100)
   2. currency: waluta płatności (w tym przypadku 'pln' oznacza polskie złote)
   3. description: opis płatności (w tym przypadku "Opłata za wypożyczenie filmu")
   4. source: identyfikator źródła płatności, który jest pobrany wcześniej z utworzonego obiektu źródła płatności ($source->id)
6. Zapisywane jest ID płatności (paymentIntentId) z obiektu paymentIntent.
7. Wywoływana jest metoda confirm() na obiekcie paymentIntents klienta Stripe, a jako argument przekazywane jest ID płatności ($paymentIntentId). Ta metoda potwierdza płatność.
8. Pobierany jest status płatności (status) z potwierdzonego obiektu paymentIntent.
9. Zwracany jest status płatności.
10. Jeśli wystąpi wyjątek (\Exception), to przekierowanie następuje do ścieżki o nazwie 'payment\_error' i do sesji dodawany jest komunikat o błędzie ('error'), który jest pobierany z wyjątku ($e->getMessage()).

Przy pierwszym uruchomieniu aplikacji jeśli klucze testowe używane do płatności na stronie wygasną skorzystaj z biblioteki Stripe Mock: Stripe udostępnia bibliotekę o nazwie "Stripe Mock", która pozwala symulować interakcje z API Stripe w środowisku testowym bez potrzeby rzeczywistych kluczy API Stripe. Możesz zainstalować tę bibliotekę i skonfigurować swoje testy, aby korzystały z niej jako zamiennika rzeczywistego API Stripe. Więcej informacji na temat biblioteki Stripe Mock znajdziesz w dokumentacji Stripe (<https://github.com/stripe/stripe-mock>).

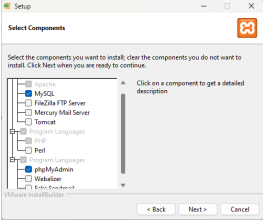
# Uruchomienie aplikacji

## Wymagania

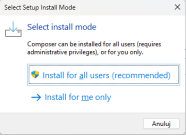
1. Pakiet XAMPP: https://www.apachefriends.org/download.html

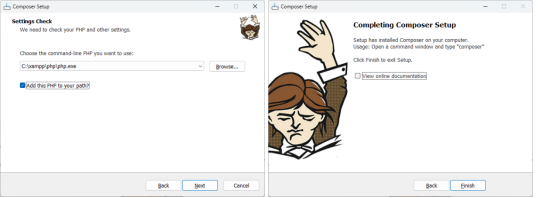


Odznaczenie niepotrzebnych składników:

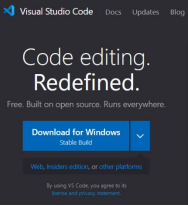


1. Composer - narzędzie do zarządzania zależnościami w projekcie Laravel. Composer jest używany do instalacji wszystkich zależności frameworka. Możesz pobrać Composer ze strony oficjalnej: <https://getcomposer.org/Composer-Setup.exe>



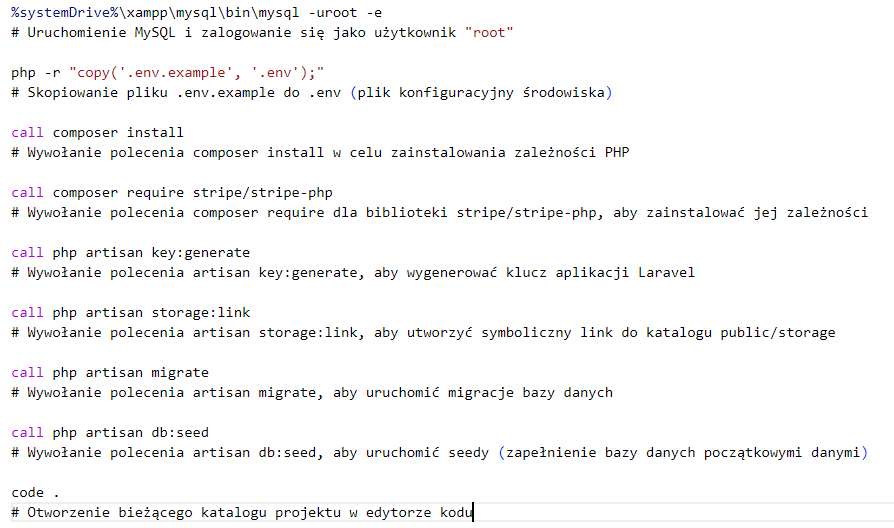


1. Visual Studio Code: <https://code.visualstudio.com/>



## Uruchomienie aplikacji

Kroki przy pierwszym uruchomieniu:

1. Uruchom Xampp i uruchom funkcje MySql oraz Apache.
2. Przejdź do folderu projektu i uruchom plik start.bat.  
     
   Na zrzucie ekranu jest wyjaśnione jakie po kolei kroki wykonuje plik start.bat
3. Uruchom wbudowany serwer deweloperski, aby uruchomić aplikację lokalnie:

Kroki przy kolejnych uruchomieniach:

1. Przejdź do katalogu projektu.
2. Uruchom wbudowany serwer deweloperski, aby uruchomić aplikację lokalnie:

# Funkcjonalności

Logowanie i rejestracja

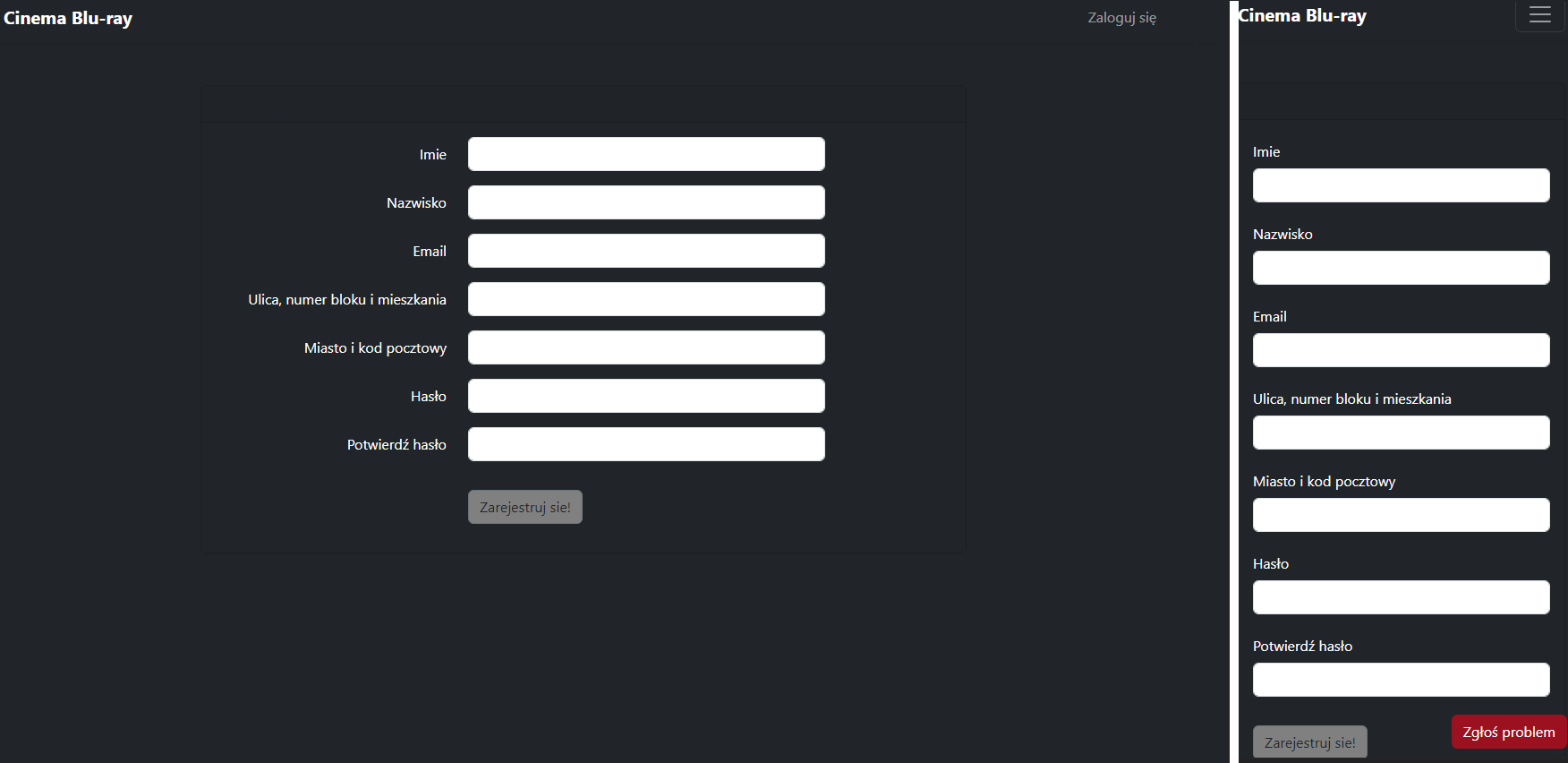
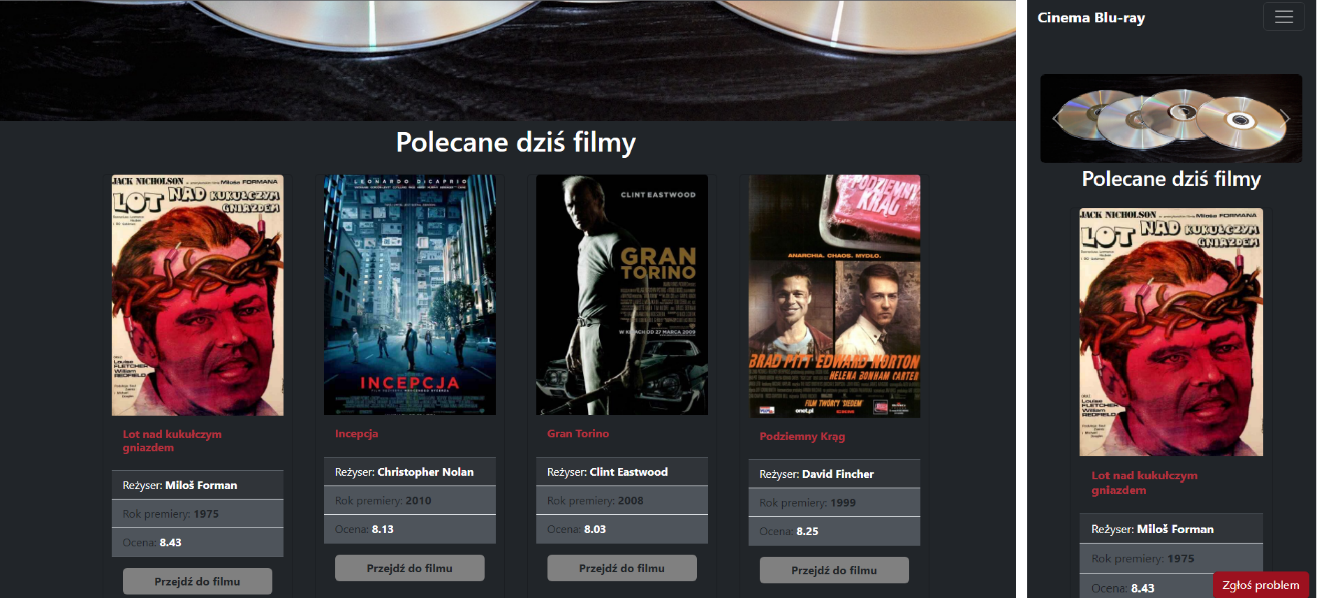
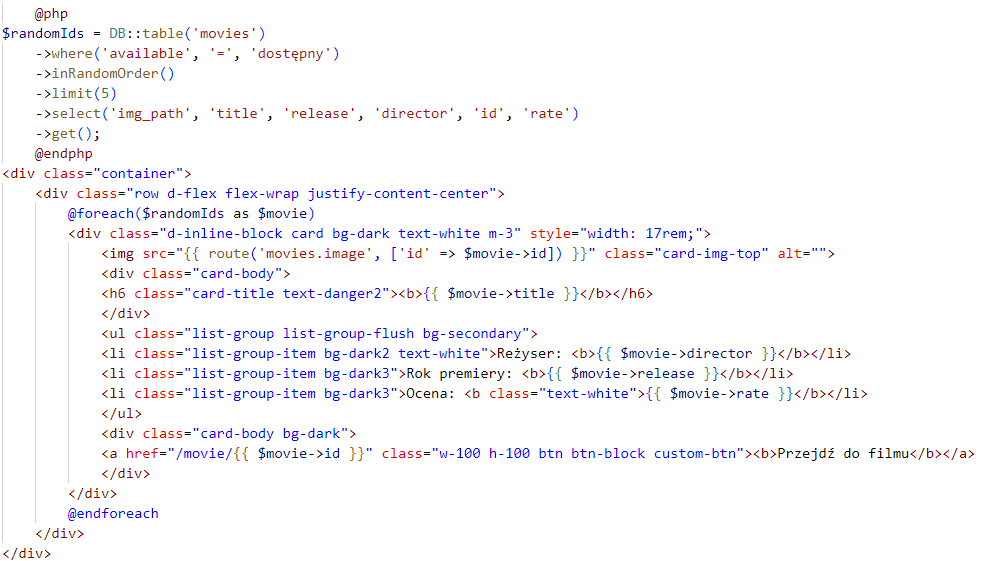
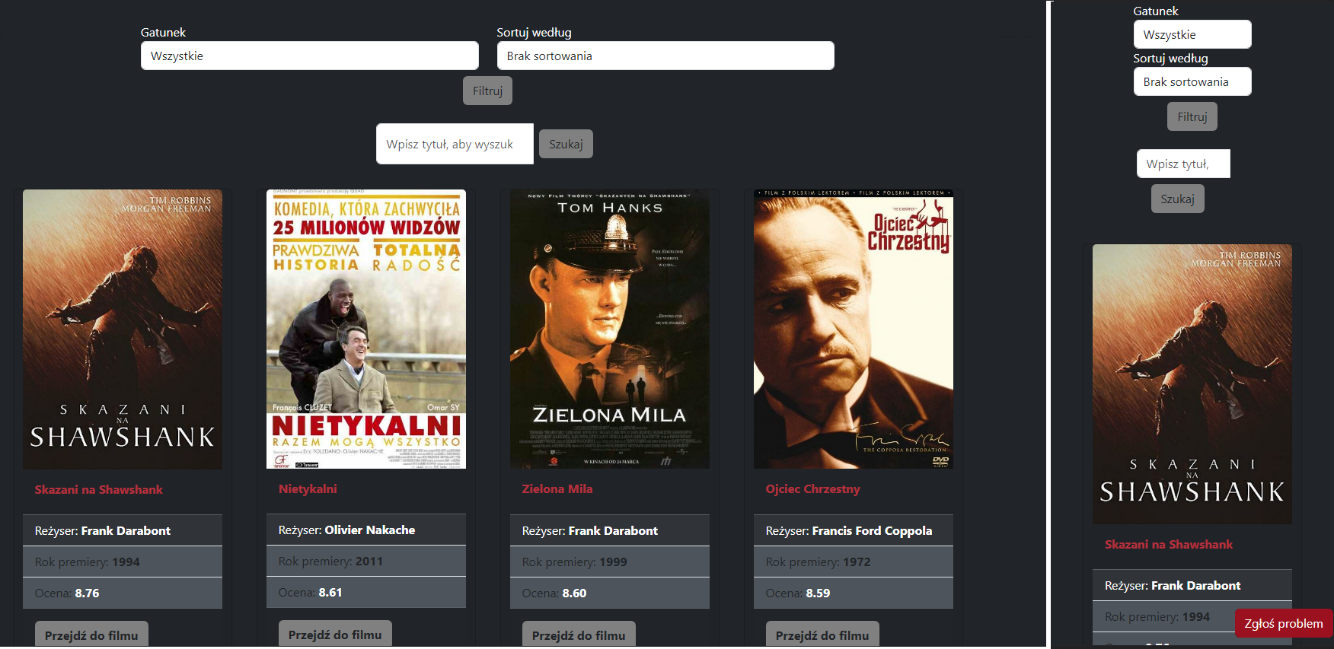
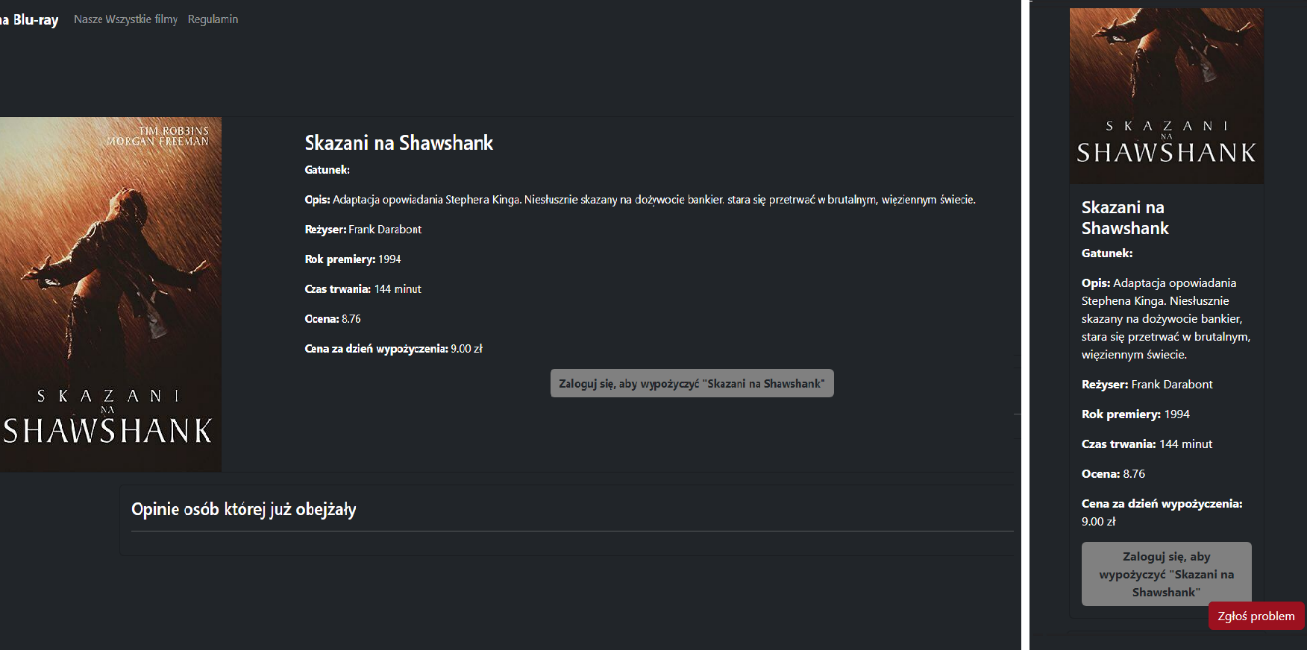
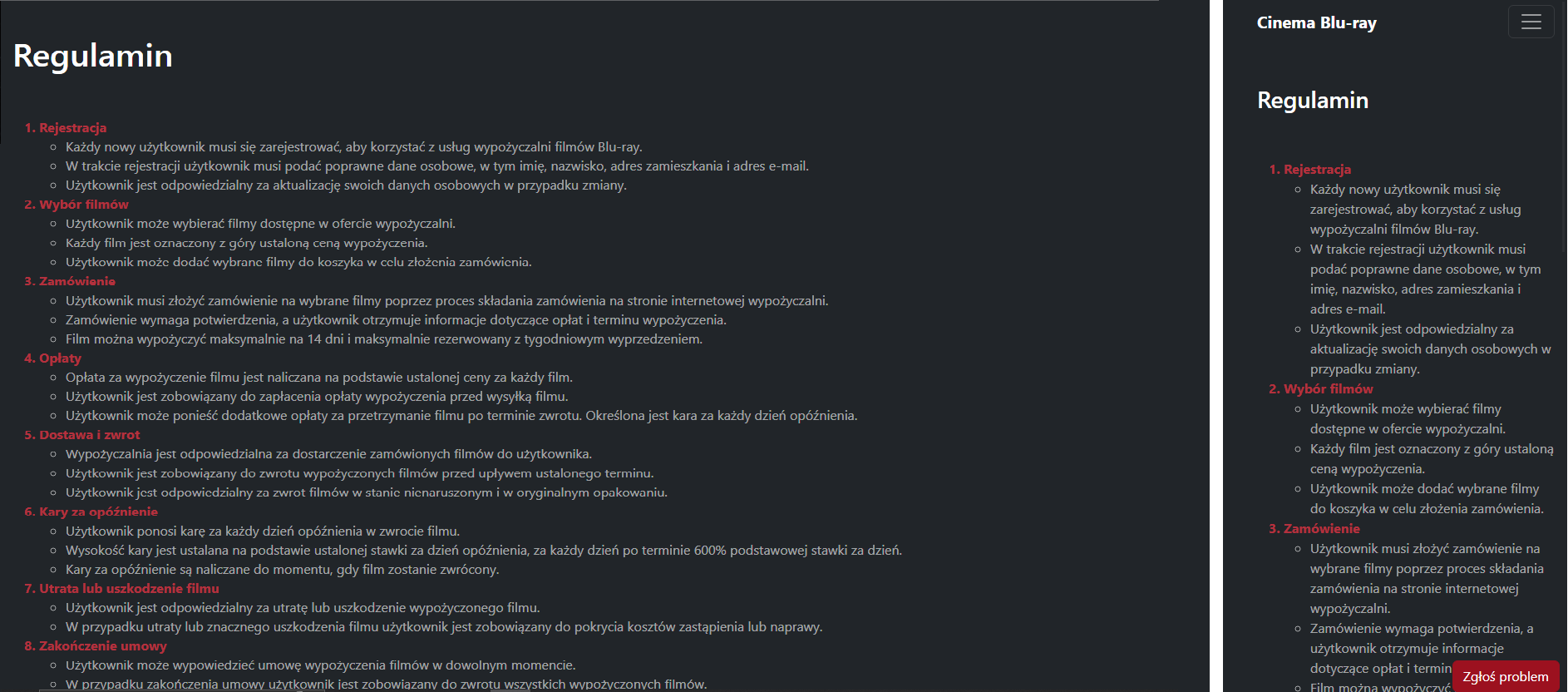
* Na stronie logowania użytkownik zostaje poproszony o wprowadzenie swojego adresu e-mail oraz hasła.
* Wprowadzone dane są walidowane pod kątem poprawności formatu i weryfikowane w bazie danych, aby sprawdzić, czy istnieje konto użytkownika o podanych danych uwierzytelniających.
* Jeśli podane dane są poprawne, użytkownik jest przekierowany do chronionej części aplikacji, gdzie ma dostęp do swojego konta.
* Jeśli podane dane są nieprawidłowe, użytkownik otrzymuje komunikat o błędzie i proszony jest o ponowne wprowadzenie danych logowania.
* Na stronie rejestracji użytkownik jest proszony o podanie swojego adresu e-mail, imiena, nazwiska, poprawnego adresu i hasła.
* Podane dane są walidowane pod kątem poprawności formatu, a także sprawdzane pod kątem unikalności, np. czy wybrany adresu e-mail jest już zajęty przez innego użytkownika.
* Podane informacje są walidowane, a następnie zapisywane w bazie danych wraz z danymi logowania użytkownika.
* Użytkownik zalogowany może zamawiać filmy na dany adres. Jeśli płatność nie jest pozytywna to można je ponownie opłacić na profilu. Filmy są automatycznie rezerwowane. Użytkownik na profilu widzi czy spóźnia się z oddaniem filmu. Widzi historię swoich filmów i może dodac do nich opinię, którą później wszyscy widzą.

## Przeglądanie zasobów

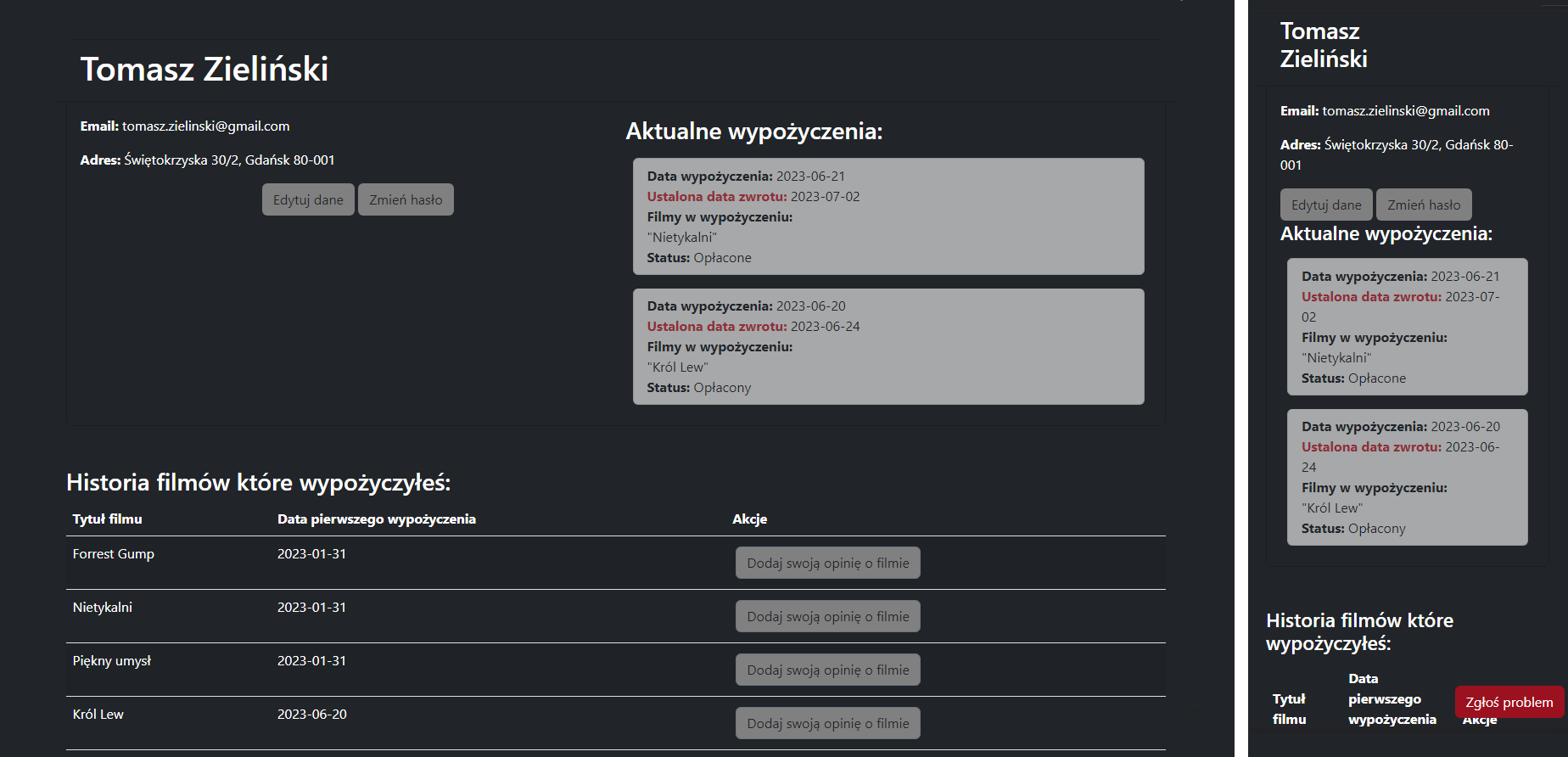
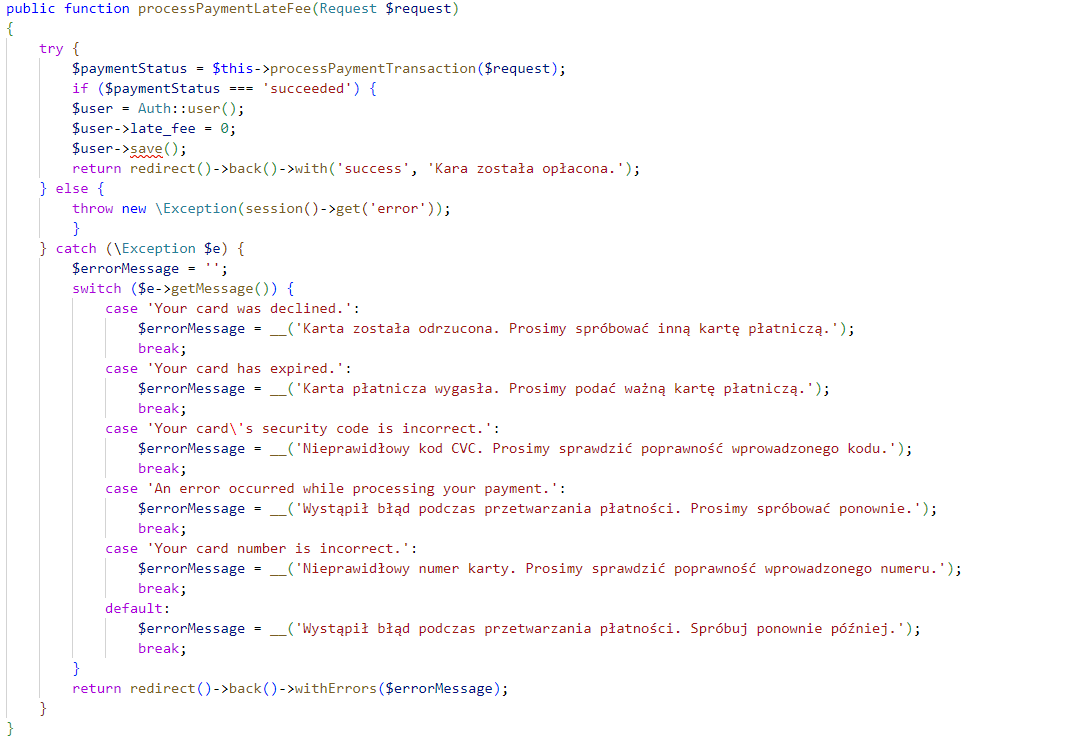
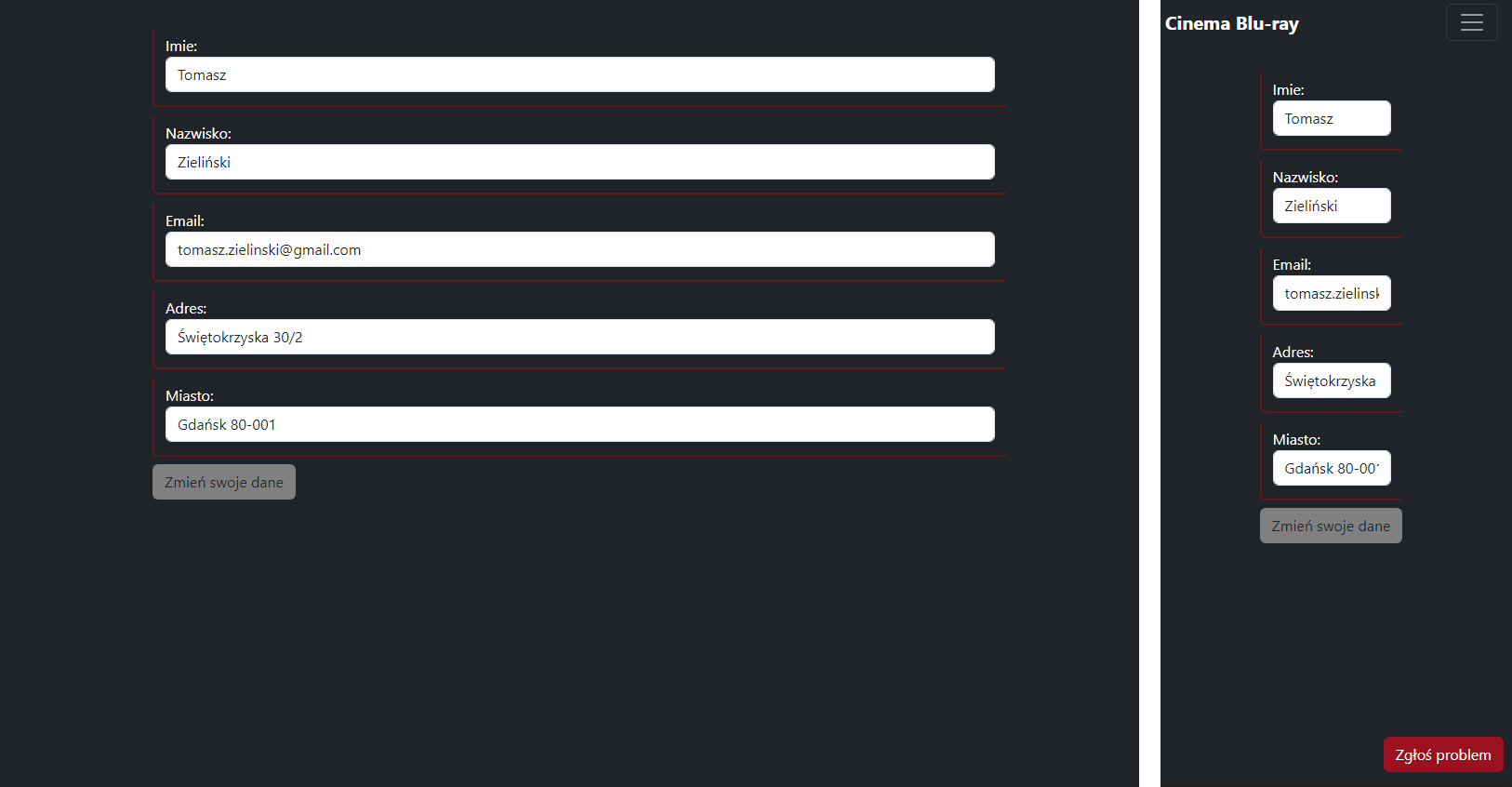
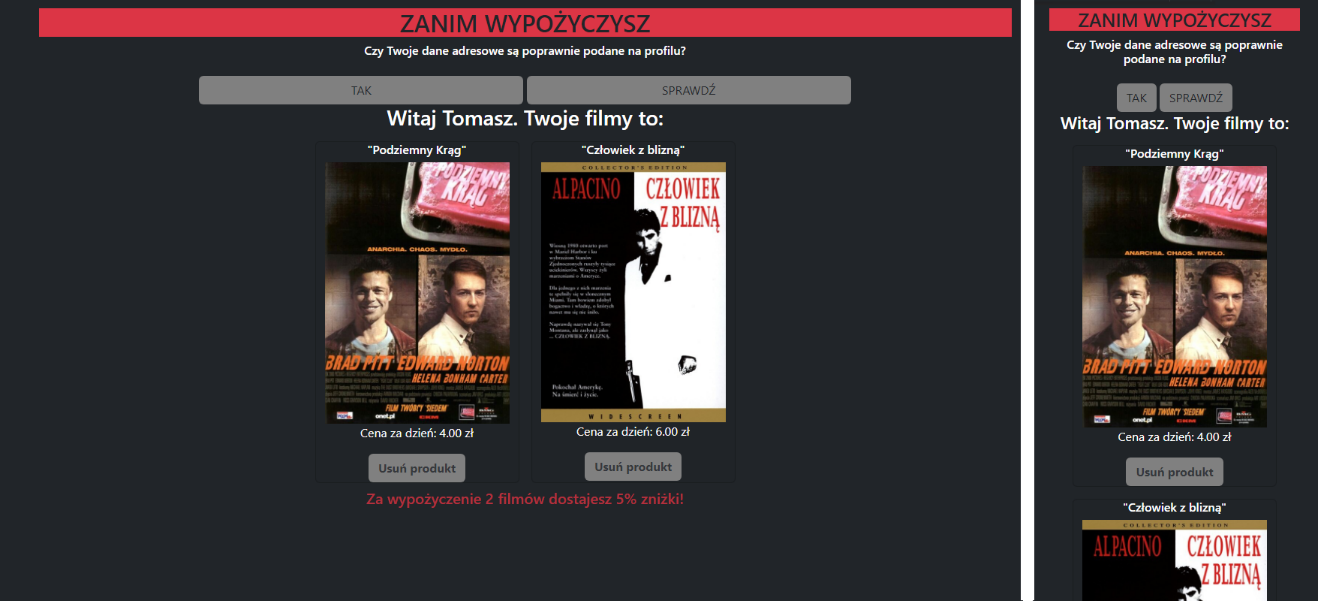
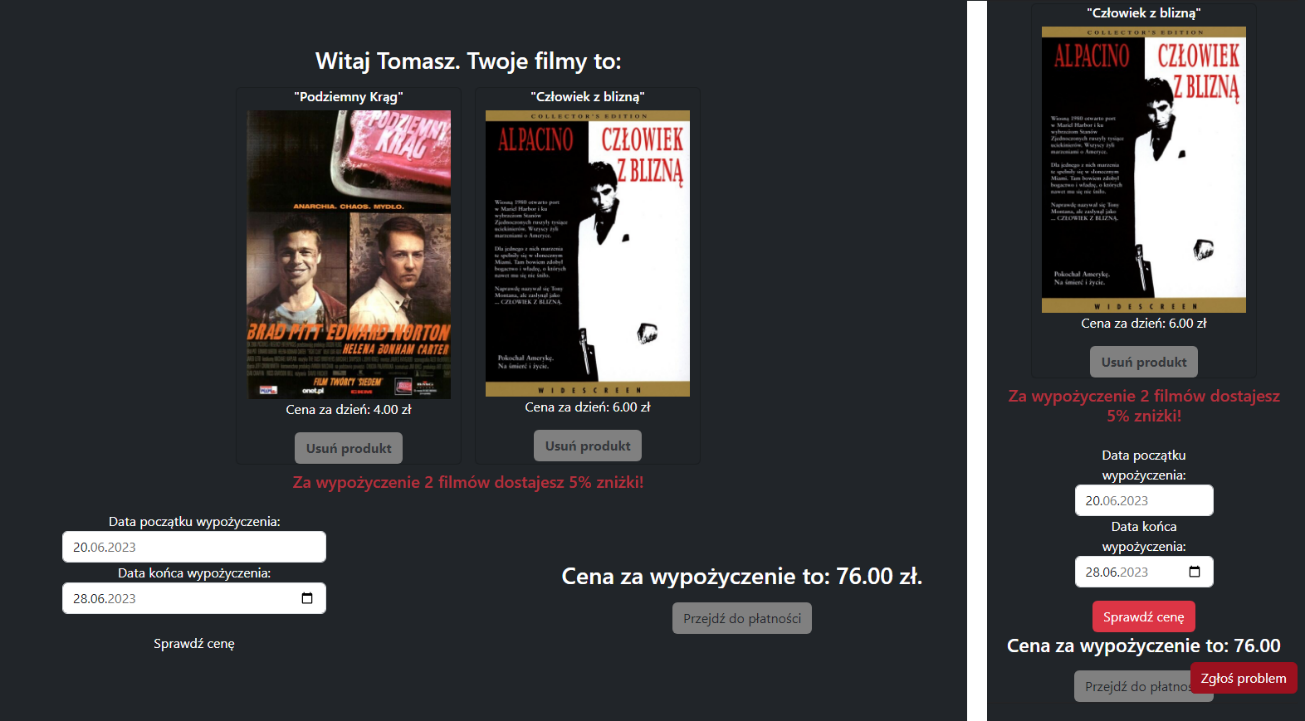
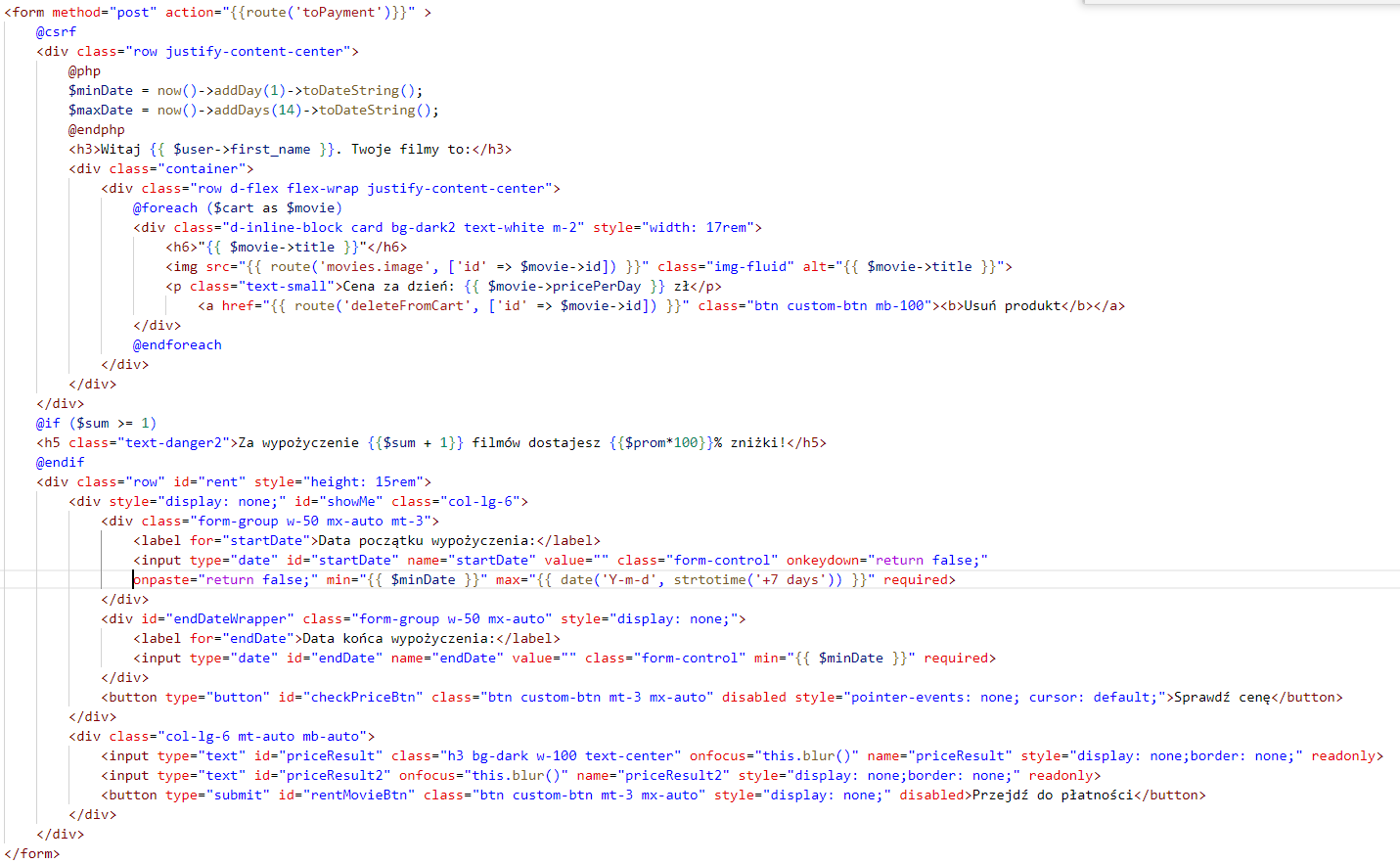
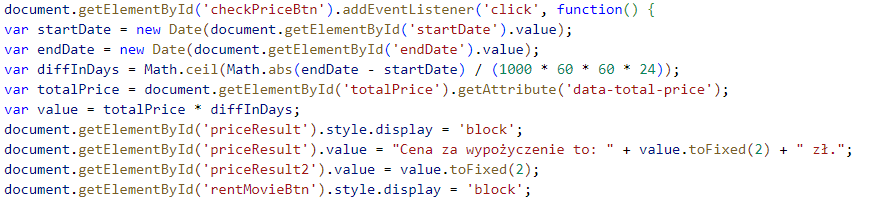
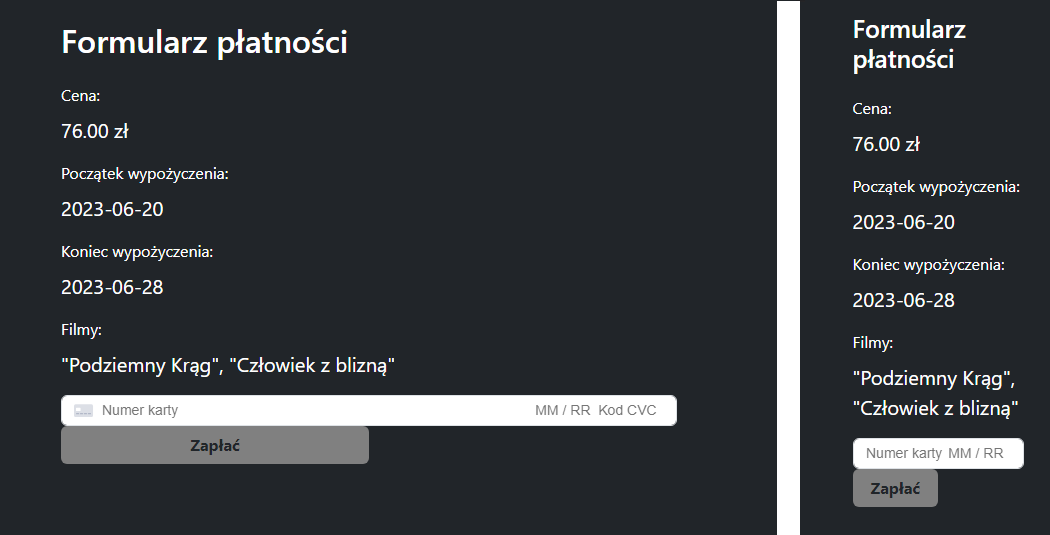
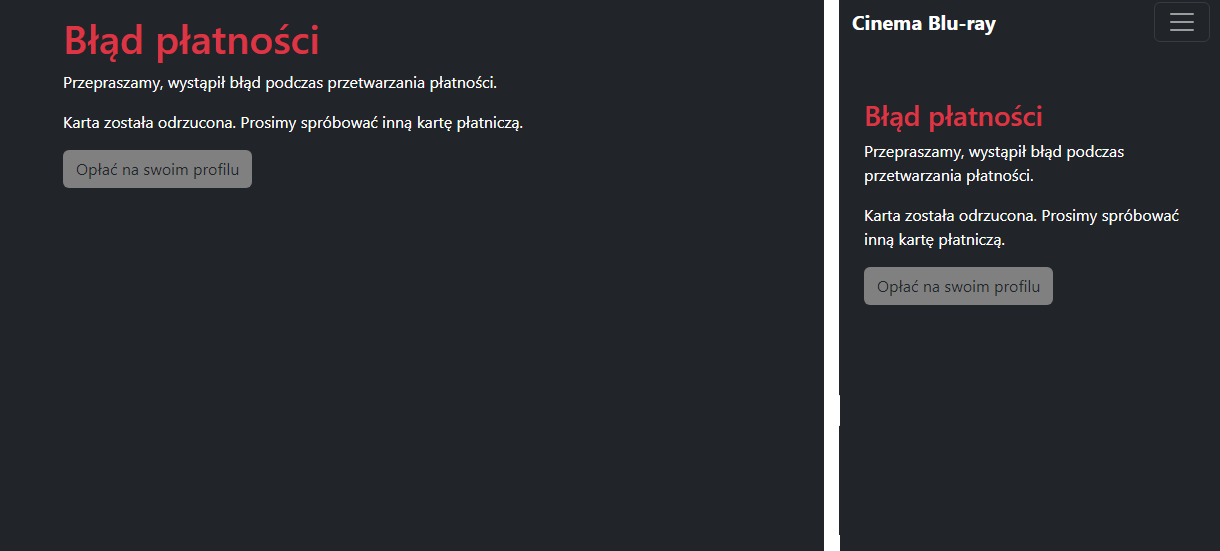
### Przykładowi użytkownicy:

* + Administrator:
    - Emial: admin@wp.pl
    - Hasło: Admin@12345
  + Zwykły użytkownik:
    - Emial: adamKowal@gmail.com
    - Hasło: aNowak@12345

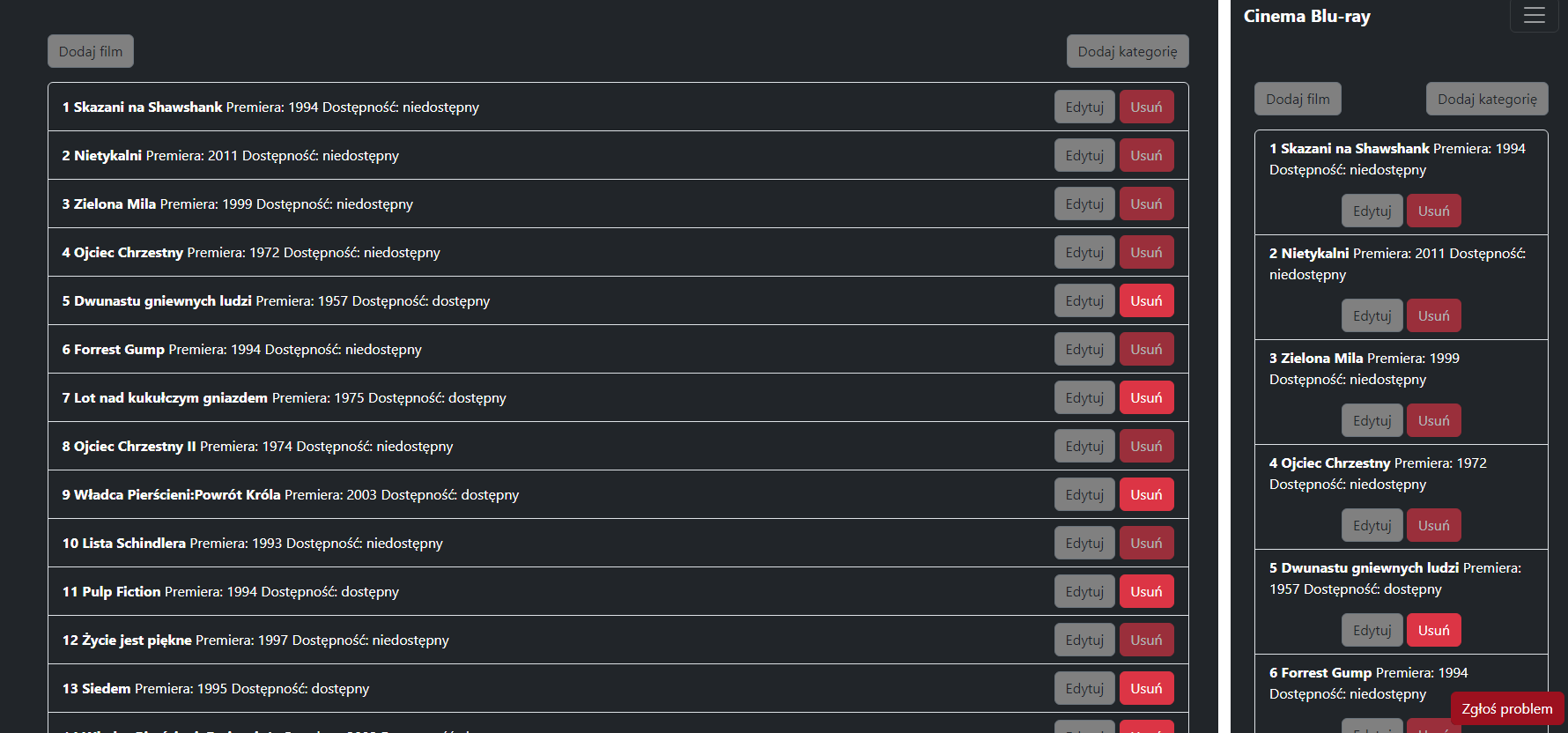
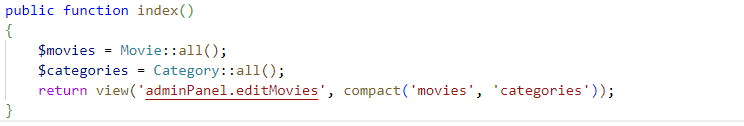
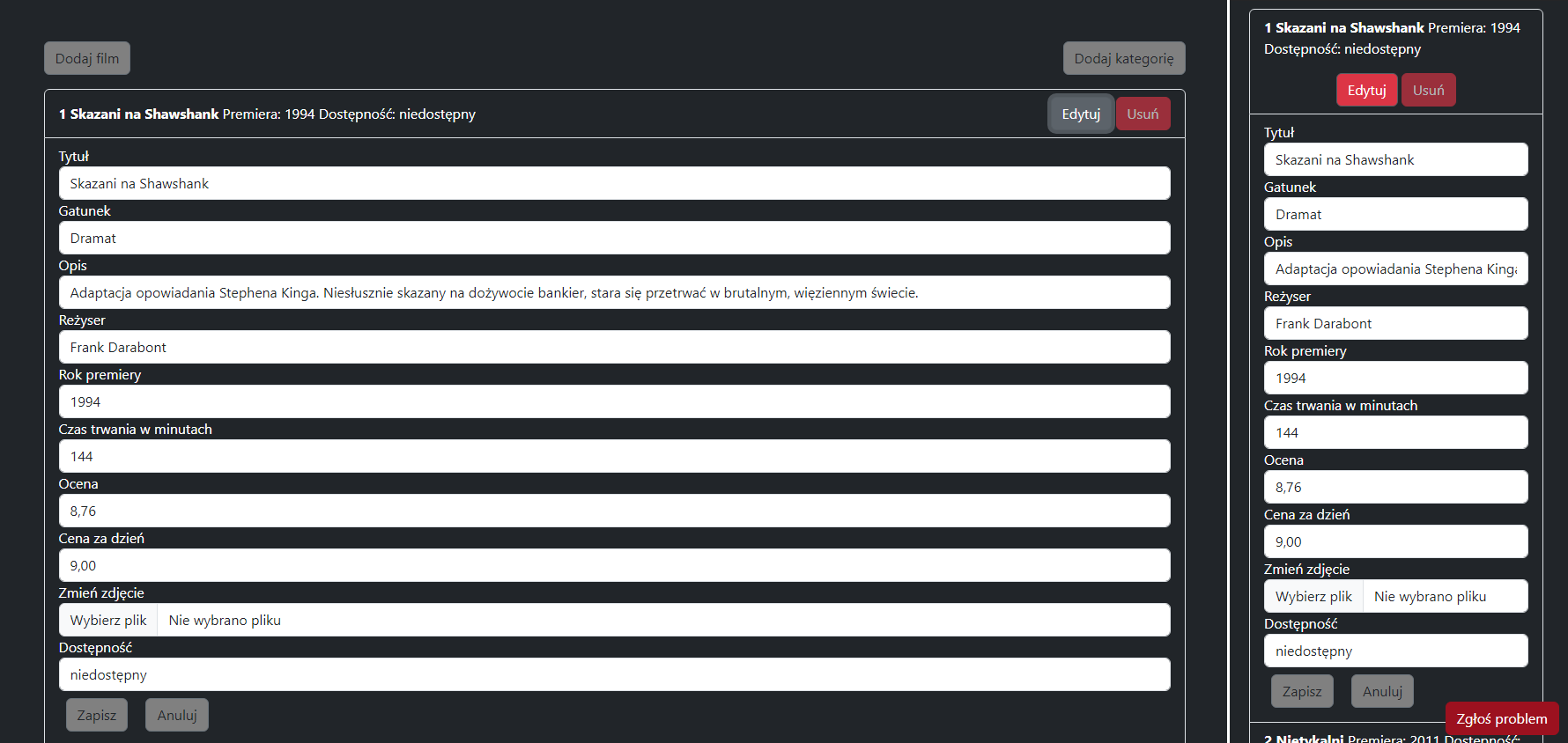
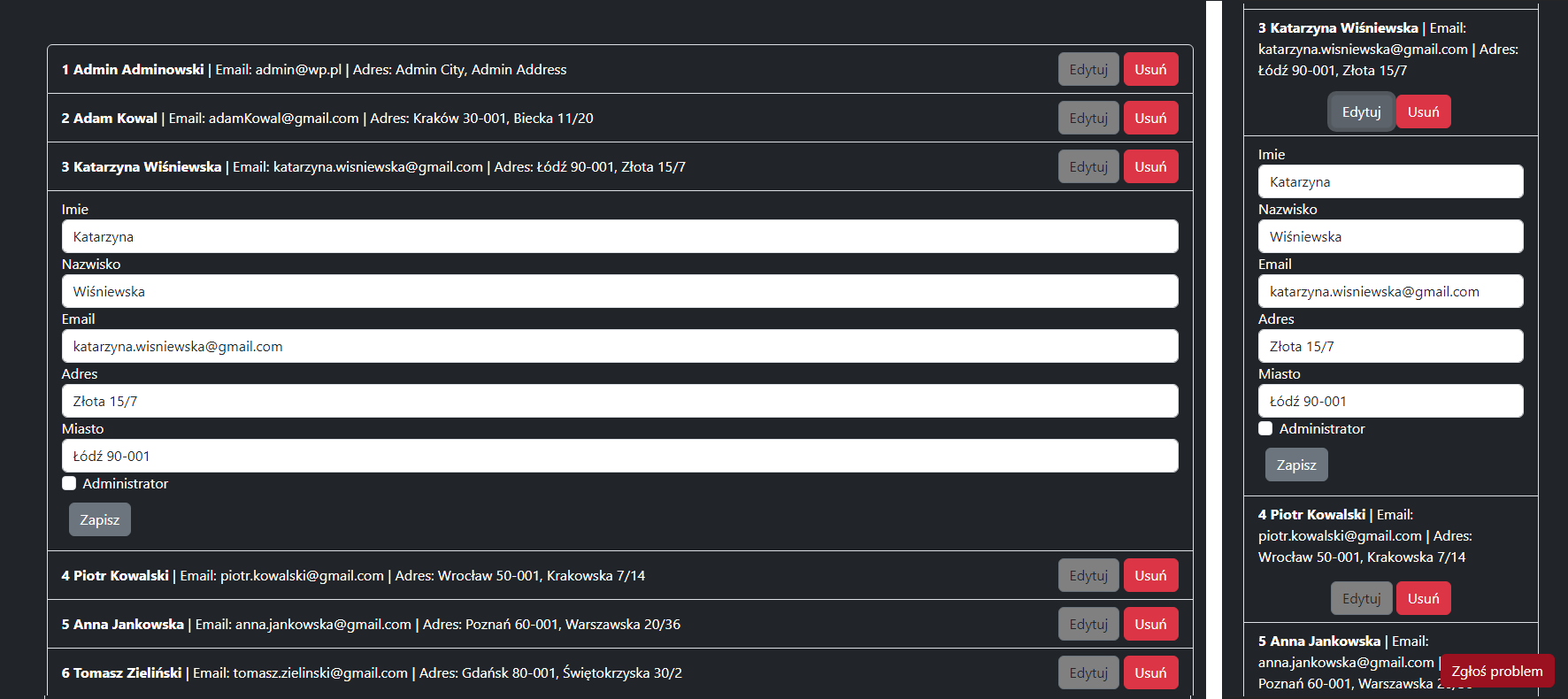
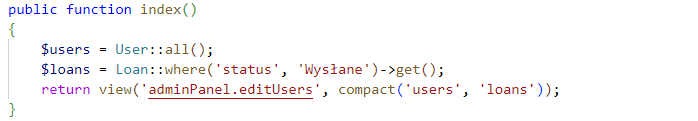
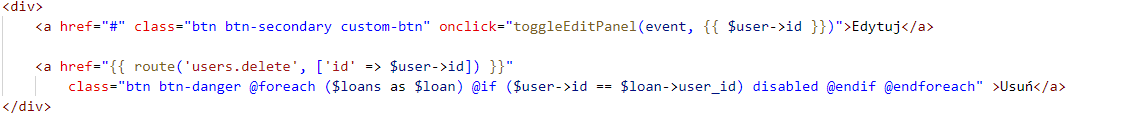
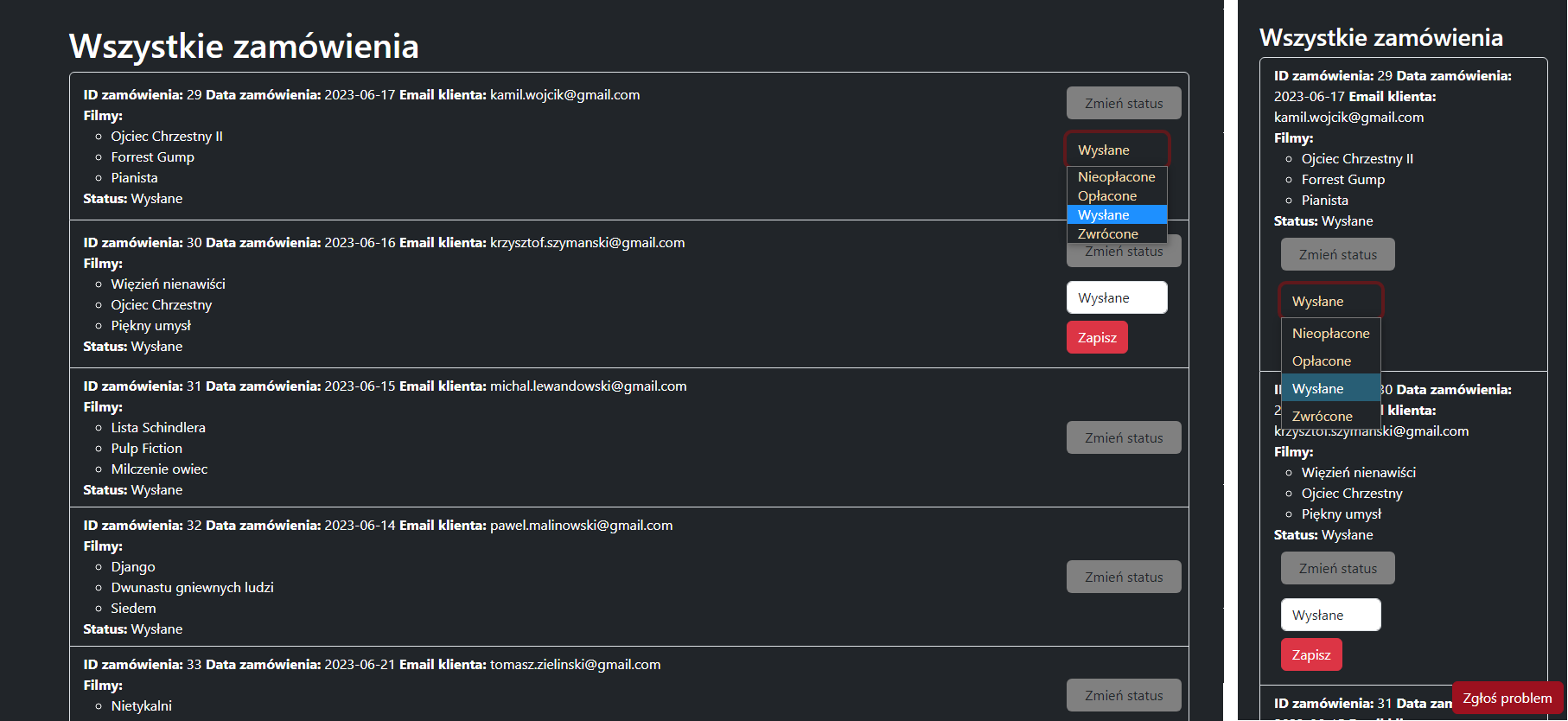
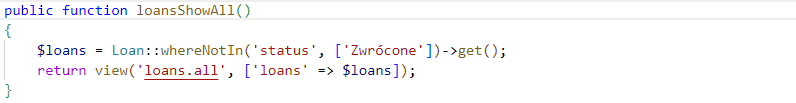
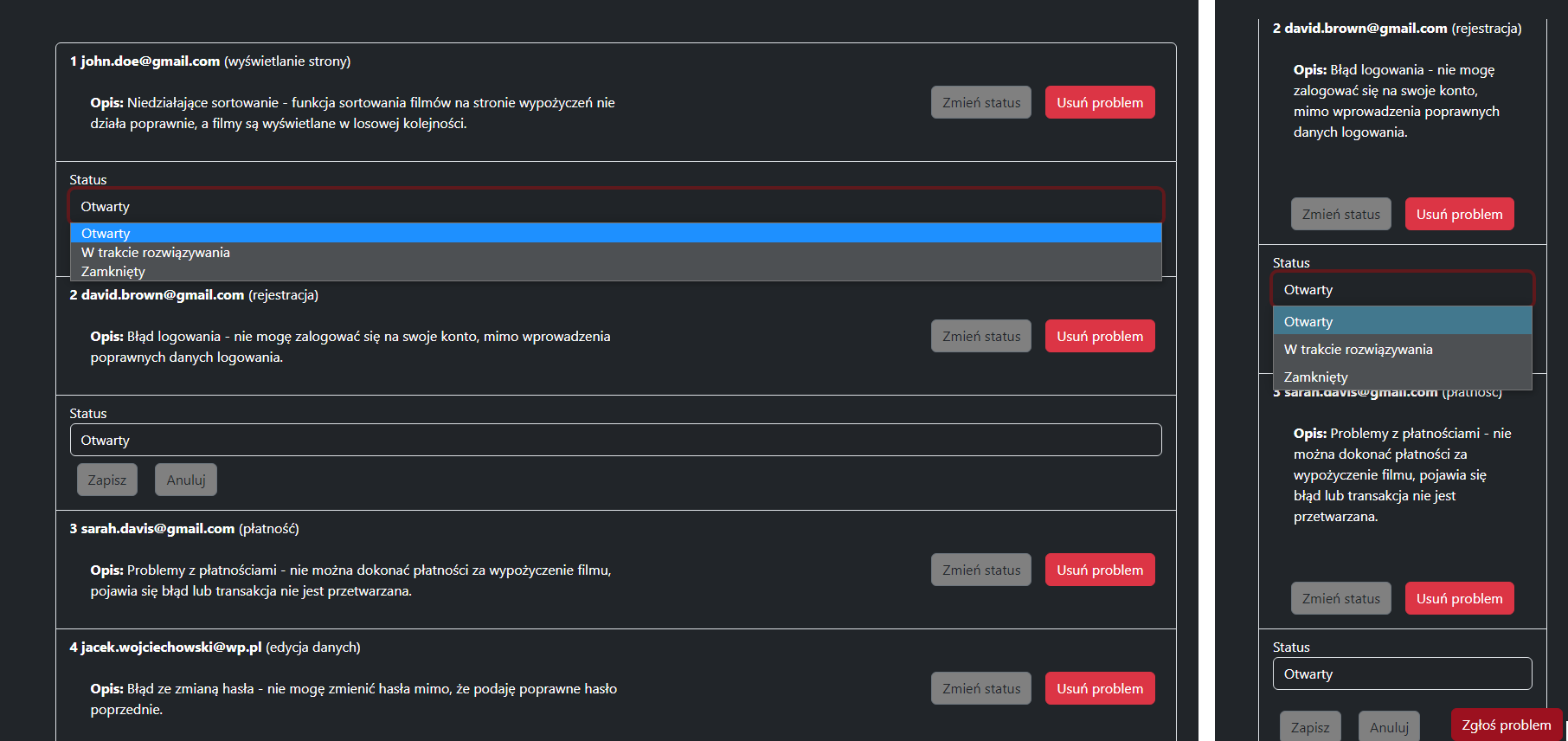
### Zasoby dostępne dla wszystkich

* Widok ekranu logowania.
* Powyższy zrzut ekranu przedstawia kod formularza logowania.
* Funkcja ta sprawdza poprawność danych logowania, obsługuje blokadę konta przy zbyt wielu nieudanych próbach logowania, wykonuje próbę logowania, obsługuje pomyślne i nieudane logowanie, a także odpowiednio zwraca odpowiedzi na podstawie wyniku procesu logowania.
* Widok ekranu rejestracji.
* Powyższy wycinek kodu odpowiada za widok formularza rejestracji.
* Funkcja ta przeprowadza proces rejestracji użytkownika, w tym walidację danych, utworzenie użytkownika, zalogowanie go do aplikacji i odpowiednie obsługiwanie wyniku rejestracji zgodnie z oczekiwaniami
* Widok strony głównej.
* Powyższy zrzut ekranu przedstawia kod losujący 5 dowolnych filmów, i wyświetla dane o nim.
* Katalog filmów
* Powyższy zrzut ekranu przedstawia fragment kodu, który wyświetla na stronie wszystkie filmy, które są przechowywane w zmiennej $movies.
* Szczegółowe dane filmu
* Funkcja przedstawiona powyżej odpowiada za wyświetlanie strony wybranego filmu dodatkowo zwracane są zmienne, które są potrzebne do sprawdzenia czy film może wylądować w koszyku.
* Widok regulaminu strony.
* Powyższy zrzut ekranu przestawia kod nagłówka strony, który w zależności od szerokości okna chowa swoje elementy listy do menu rozwijanego.
* Powyższy zrzut ekranu przedstawia kod jak i widok formularza wysyłki problemów znalezionych na stronie.
* Administrator może usuwać i edutować filmy i użytkowników, rozpatruje błędy zgłoszone na stronie, obsługuje zamówienia.

### Zasoby dostępne dla zalogowanego użytkownika

* Widok profilu zalogowane użytkownika, który ma aktualnie 2 wypożyczenia ze statusem Opłacone
* Powyższy zrzut ekranu przedstawia kod, który jest odpowiedzialny za sprawdzenie czy dany użytkownik ma naliczoną karę na swoje konto, jeśli tak to jest ona wyświetlana poniżej adresu. Jest wyświetlany od razu formularz pozwalający opłacić karę.
* Powyższy zrzut ekranu przedstawia funkcję która odpowiada za opłatę kary jaką miał użytkownik.
* Widok formularza zmiany danych dla zalogowanego użytkownika.
* Zrzut ekranu przedstawia kod, który po udanej zmianie danych pokazuje komunikat ze sukcesem i po 5 sekundach wraca na profil zalogowanego użytkownika.
* Funkcja przedstawiona powyżej odpowiada za sprawdzenie podanych danych podczas edycji, jeśli dane przejdą walidację funkcja aktualizuje dane użytkownika.
* Widok formularza zmiany hasła.
* Powyższy zrzut ekranu przedstawia walidację zrobioną w JavaScript po stronie klienta dla dla formularza zmiany hasła Sprawdza kolejne pola czy spełniają zadane warunki, jeśli nie, żądanie nie przejdzie do serwera.
* Widok koszyka z dodanymi do niego filmami.
* Widok koszyka z wybraną datą wypożyczenia, widoczną obliczoną na podstawie dni i wybranych filmów ceną wypożyczenia oraz z przyciskiem przenoszącym do płatności.
* Powyższy zrzut ekrany przedstawia formularz z widoku koszyka przesyłający do podstrony płatności najważniejsze rzeczy np. kwota do zapłaty.
* Powyższy zrzut ekranu przedstawia funkcję obliczającą cenę na podstawie samych filmów znajdujących się w koszyku.
* Powyższy zrzut ekranu przedstawia kod obliczający już całkowity koszt wypożyczenia na podstawie ceny przesłanej z funkcji(przedstawionej 2 zrzuty ekranu wyżej) i sumy dni wypożyczenia.
* Widok formularza płatności za wypożyczenie.
* Powyższy zrzut ekranu przedstawia kod formularza z podstrony płatności za wypożyczenie.
* Powyższy zrzut ekranu przedstawia kod funkcji odpowiadającej za tworzenie nowego wypożyczenia i w zależności od statusu płatności zamówienie dostaje status opłacony jeśli płatność zakończona sukcesem, status nieopłacony - płatność zakończona niepowodzeniem.
* Widok wyświetlany kiedy płatność nie zakończy się sukcesem.
* Widok kiedy płatność zakończona jest sukcesem.

## CRUD administratora

* Widok do zarządzania filmami(edycja, usuwanie, dodawanie). Dodatkowo jest tu opcja dodawania nowego gatunki filmowego.
* Powyższy zrzut ekranu przedstawia kod funkcji odpowiedzialnej za zwrócenie do widoku o nazwie ”adminPanel.editMovies” wszystkich rekordów z tabel movies i categories. Następnie przekierowuje nas do tego widoku.
* Powyższy zrzut ekranu przedstawia kod funkcji, które pozwalają podpiąć pod dany przycisk pokazywanie lub chowanie się formularza edycji, dodawania, usuwania filmu lub dodawania kategorii filmowej.
* Widok edycji filmu z rozwiniętym formularzem.
* Widok formularza edycji i usuwania użytkownika.
* Powyższy zrzut ekranu przedstawia kod funkcji, która przekierowuje mnie do widoku edycji użytkowników z danym o wszystkich użytkownikach i zwraca nam też wszystkie wypożyczenia które mają status ”Wysłane”.
* Powyższy zrzut ekranu przedstawia fragment kodu, który wykorzystuje wiedzę o statusie wypożyczeni do ustawienia odpowiedniego atrybutu dla odnośnika, który usuwa użytkownika. Dzięki temu nie można usunąć z bazy użytkownika, który jeszcze nie oddał wszystkich filmów.
* Powyższy zrzut ekranu przedstawia kod funkcji zapisującej nowy film dodany przez administartora.
* Widok do zarządzania wypożyczeniami.
* Powyższy zrzut ekranu przedstawia nam kod funkcji, która przekierowuje do widoku, w którym zarządza się wypożyczeniami. Przekazujemy w niej wypożyczenia tylko te gdzie status jest różny niż ”Zwrócone” dzięki czemu administrator widzi tylko potrzebne wypożyczenia i nie musi przeglądać już zakończonych wypożyczeń.
* Widok formularza zarządzającego zgłoszeniami o błędach na stronie.
* Powyższy zrzut ekranu przedstawia kod funkcji odpowiadającej za dodawanie problemów do bazy danych. Funkcja zanim prześle zgłoszenie najpierw sprawdza poprawność podanych danych.
* Powyższy zrzut ekranu przedstawia kod widoku podstrony do zarządzania problemami.