

**Projekt (Szkolenie techniczne 1)**

Autor: Anhelina Kryvaltsevich

Nr albumu: w70625

Kierunek: 4IID-P

**Prowadzący: Dr Marek Giebułtowski**

Rzeszów June 9, 2025

***Dokumentacja projektu — System zarządzania filmami***

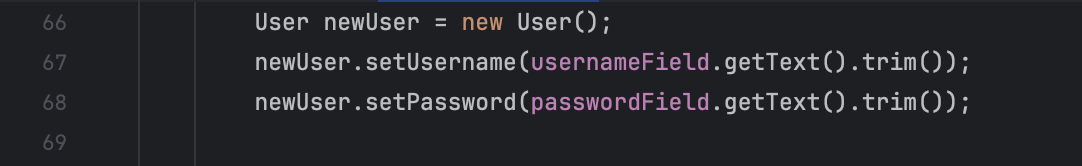
## ***1. Wstęp***

Projekt jest aplikacją desktopową napisaną w Javie z użyciem JavaFX, umożliwiającą użytkownikom rejestrację, logowanie oraz zarządzanie listą filmów. Aplikacja komunikuje się z bazą danych, w której przechowywane są dane użytkowników oraz filmy.

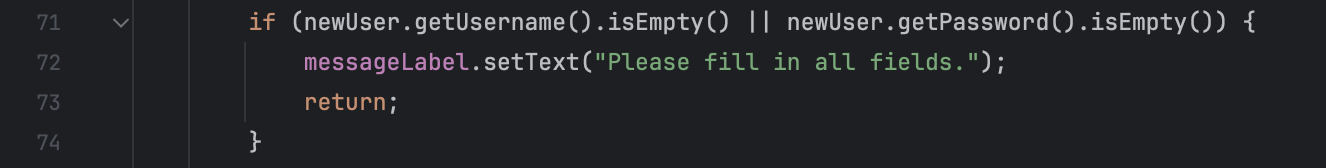
## ***2. Funkcjonalności***

### 1. Rejestracja nowego użytkownika

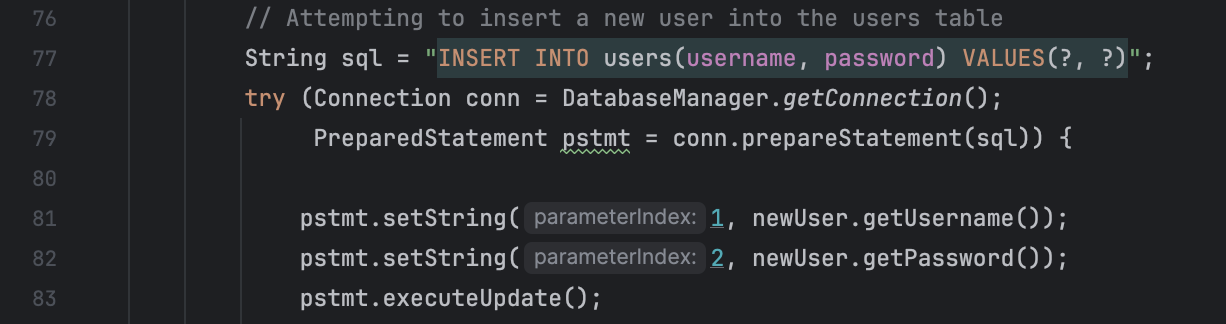
* **Opis:**  
  Użytkownik może utworzyć nowe konto, podając nazwę użytkownika i hasło. Dane są zapisywane do bazy danych w tabeli users.
* **Klasa:** LoginController.java
* **Metoda:** handleRegister()
* **Działanie metody:**
  + Pobiera dane z pól tekstowych (nazwa użytkownika i hasło):



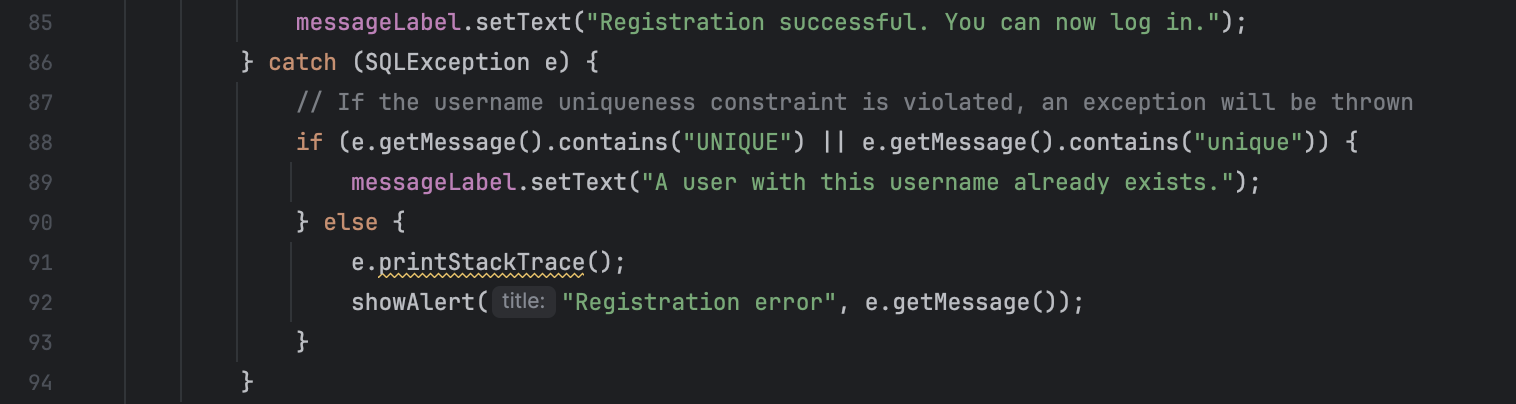
* + Sprawdza, czy pola nie są puste:



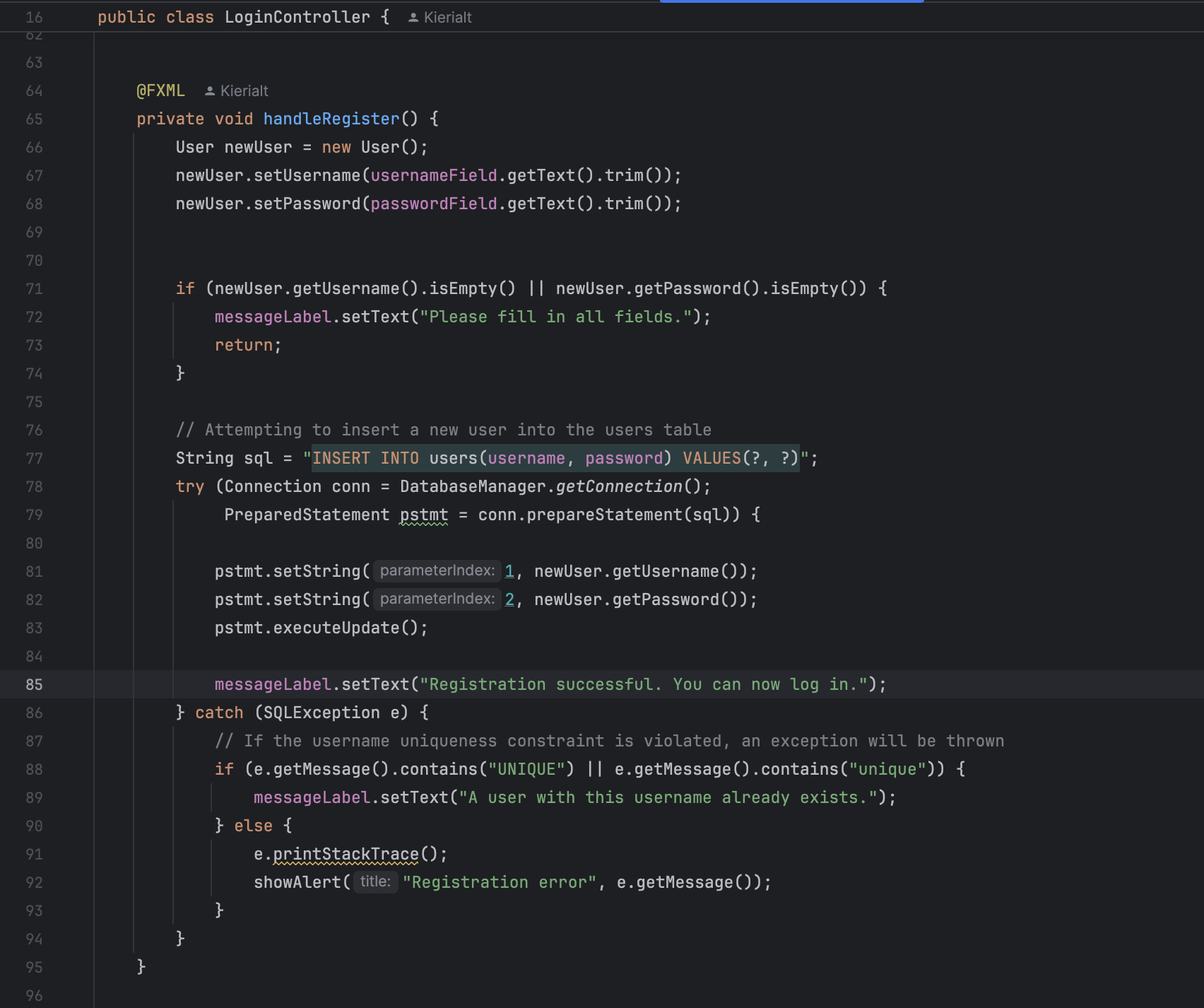
* + Wywołuje metodę getConnectoin() z klasy DatabaseManager, aby zapisać dane do bazy:



* + Wyświetla odpowiedni komunikat (sukces lub błąd):

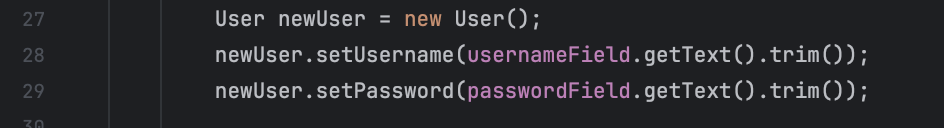


Pełna implementacja metody handleRegister():

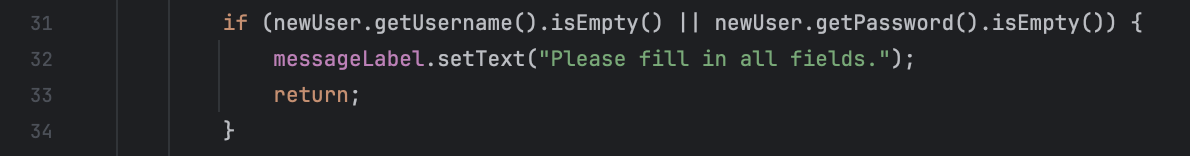


### ***2. Logowanie użytkownika***

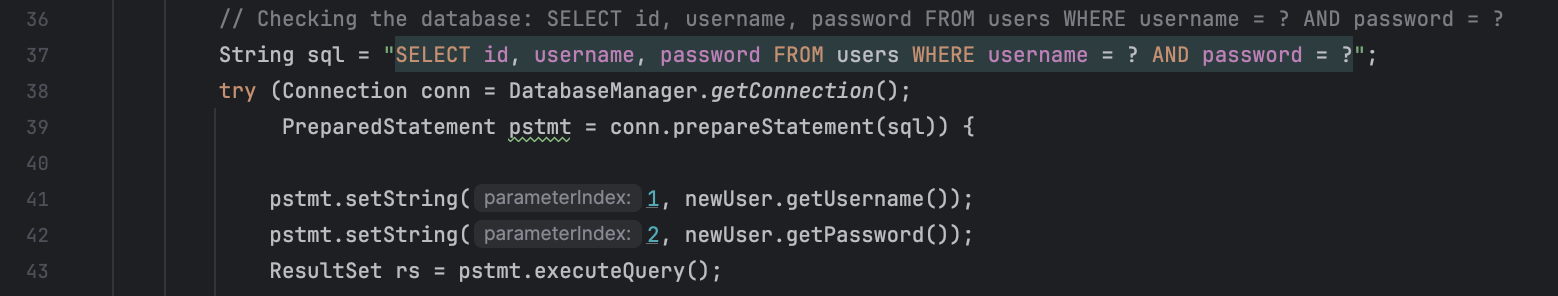
* **Opis:**  
  Użytkownik może się zalogować do systemu podając poprawne dane. Weryfikacja następuje w bazie danych.
* **Klasa:** LoginController.java
* **Metoda:** handleLogin()
* **Działanie metody:**
  + Pobiera dane z pól tekstowych (nazwa użytkownika i hasło):



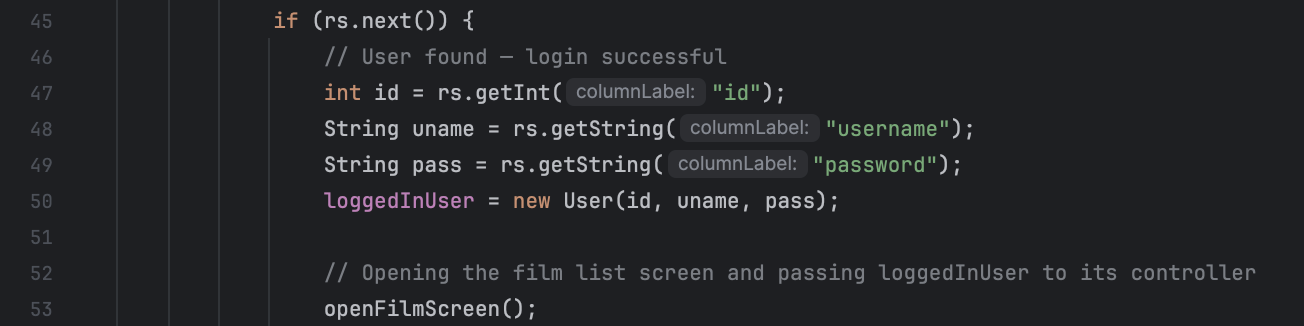
* + Sprawdza, czy pola nie są puste:



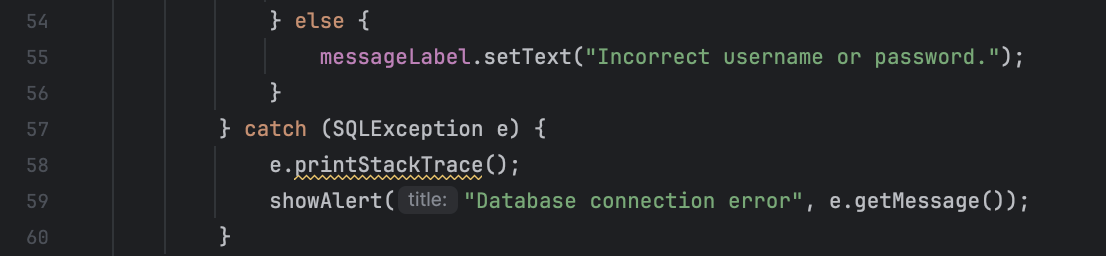
* + Wywołuje metodę getConnection() z klasy DatabaseManager:



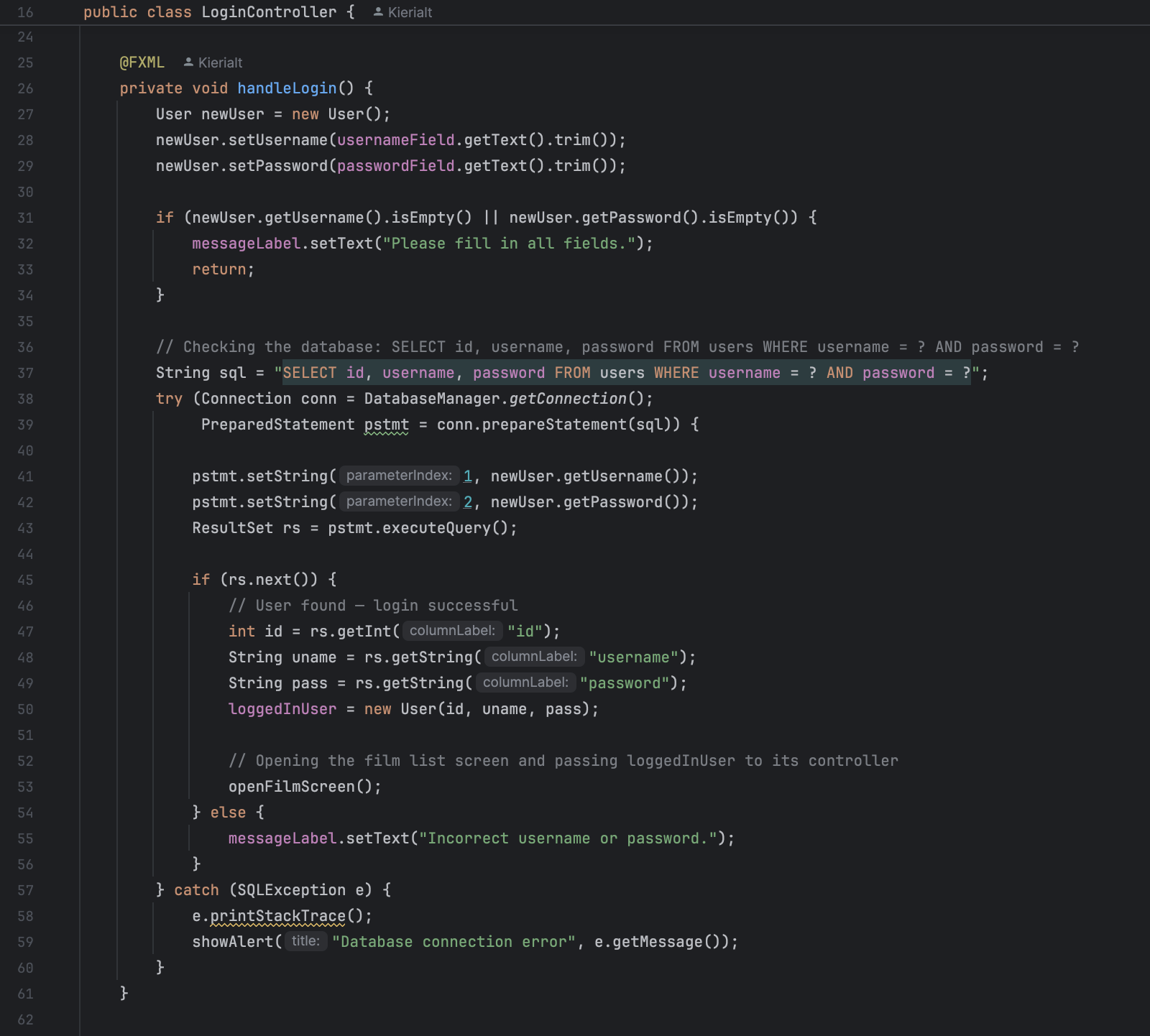
* + Jeśli dane są poprawne — otwiera główne okno z listą filmów (main.fxml):



* + W przeciwnym razie — wyświetla komunikat o błędzie logowania:



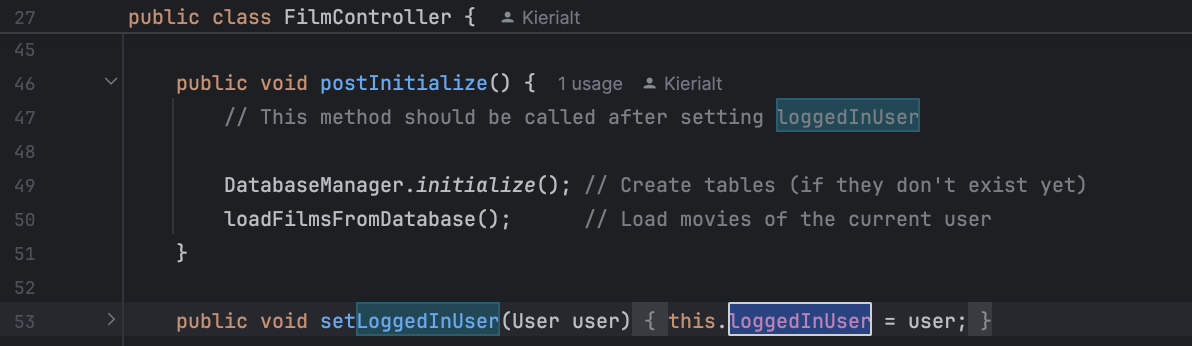
Pełna implementacja metody handleLogin():



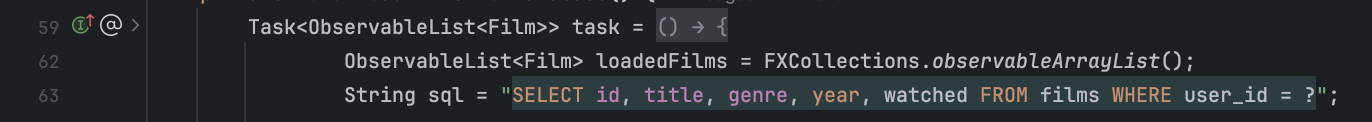
### ***3. Wyświetlanie listy filmów***

* **Opis:**  
  Po zalogowaniu użytkownik widzi listę filmów pobranych z bazy danych.
* **Klasa:** FilmController.java
* **Metoda:** initialize() oraz loadFilmsFromDatabase()
* **Działanie metody:**
  + Przy uruchomieniu main.fxml metoda initialize(), postinitialize() automatycznie wywołują loadFilmsFromDatabase():



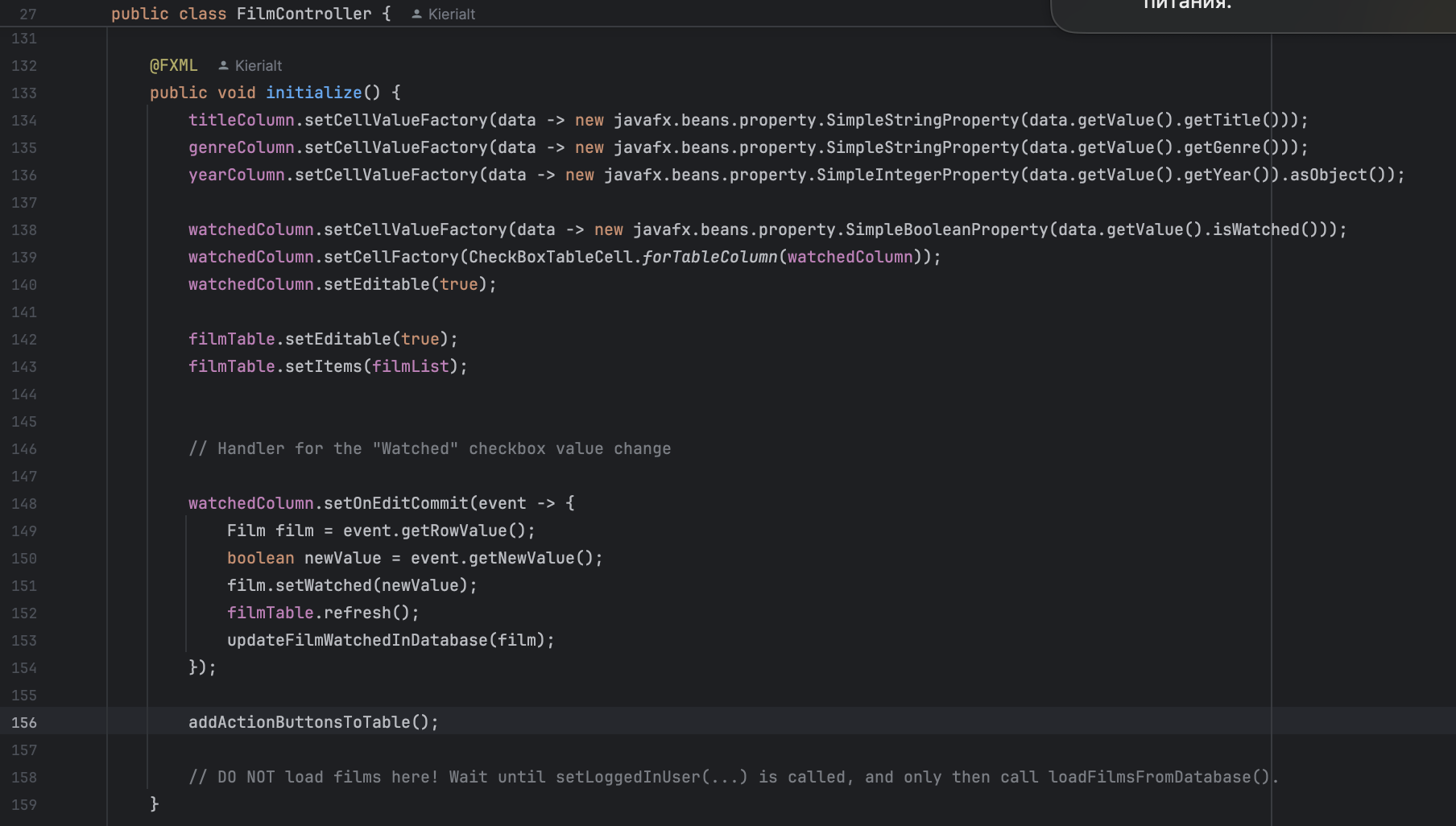


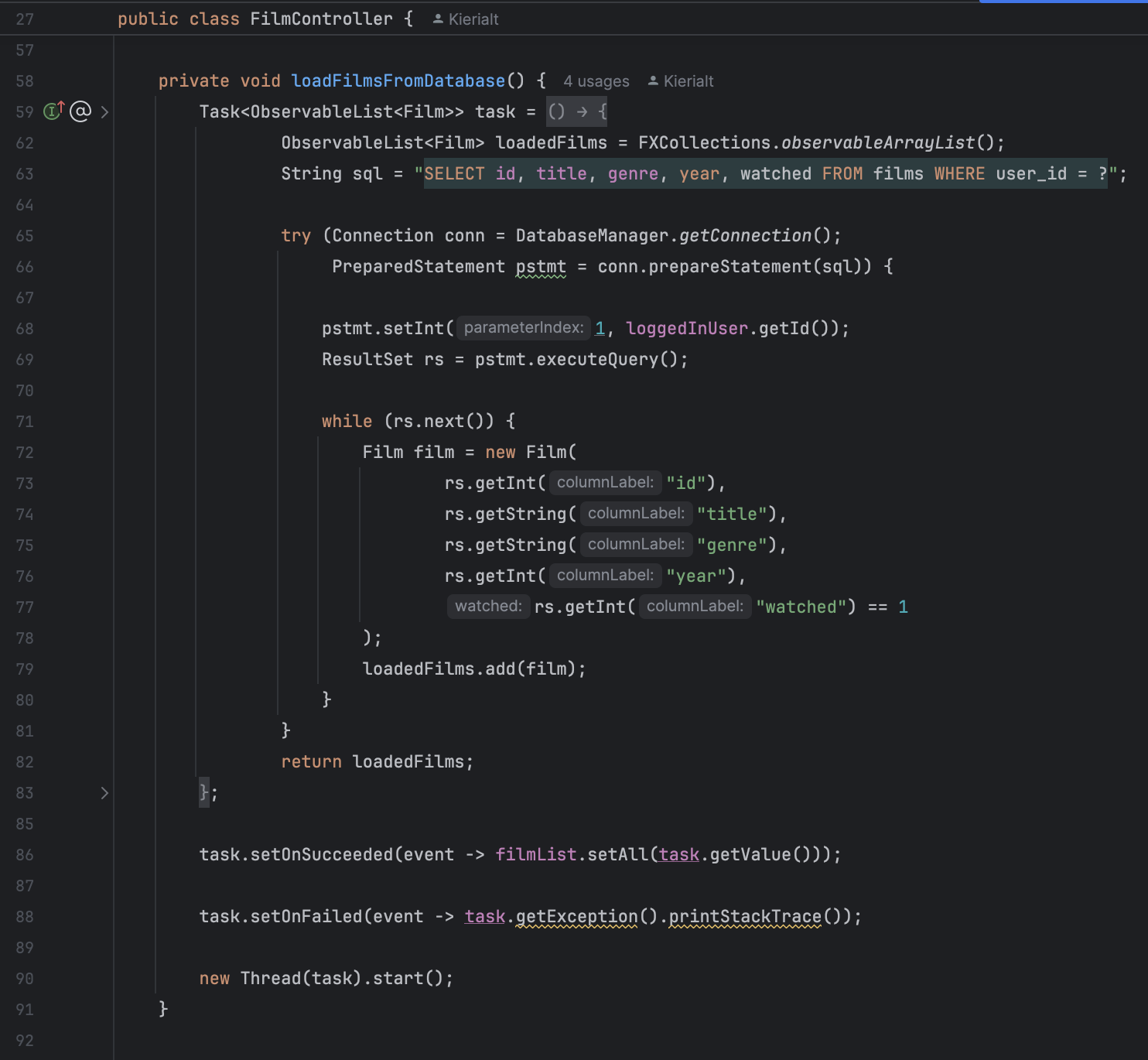
* + loadFilmsFromDatabase() pobiera dane filmów z bazy przy pomocy DatabaseManager:



* + Dane są wyświetlane w tabeli “films”:

Pełna implementacja metod initialize() oraz loadFilmsFromDatabase():



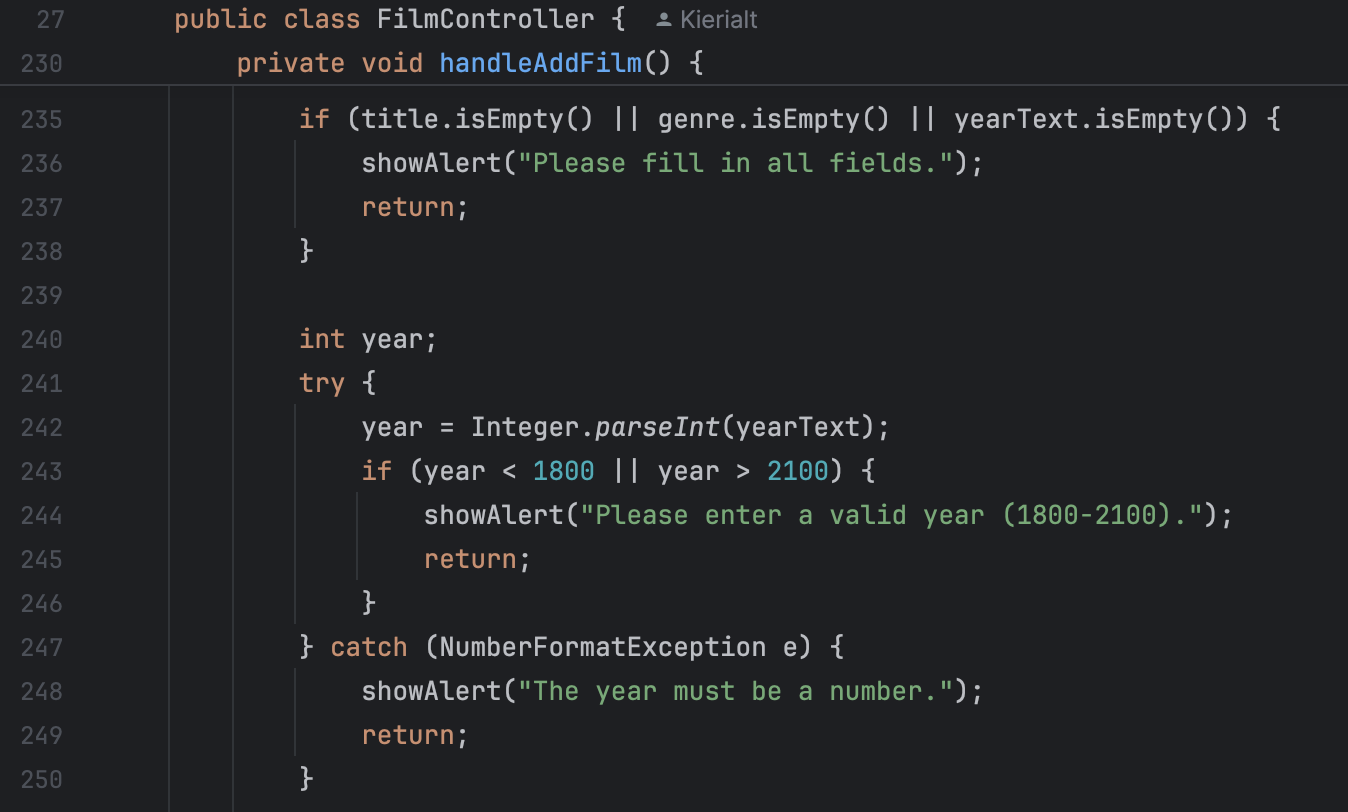


### ***4. Dodawanie nowego filmu***

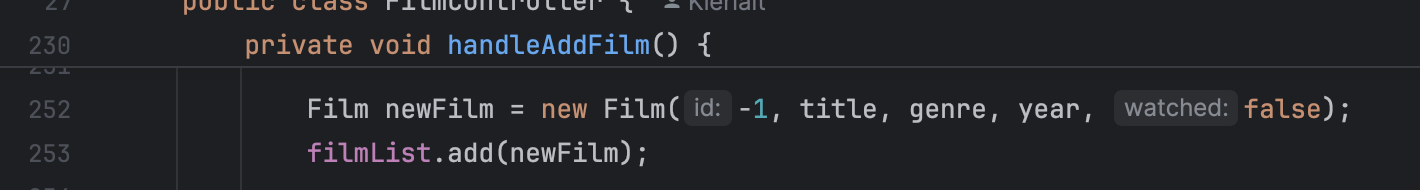
* **Opis:**  
  Użytkownik może dodać nowy film do listy, podając tytuł, gatunek i rok.
* **Klasa:** FilmController.java
* **Metoda:** handleAddFilm()
* **Działanie metody:**
  + Odczytuje dane z pól tekstowych (tytuł, gatunek, rok).



* + Sprawdza poprawność danych.



* + Odświeża listę filmów po dodaniu.





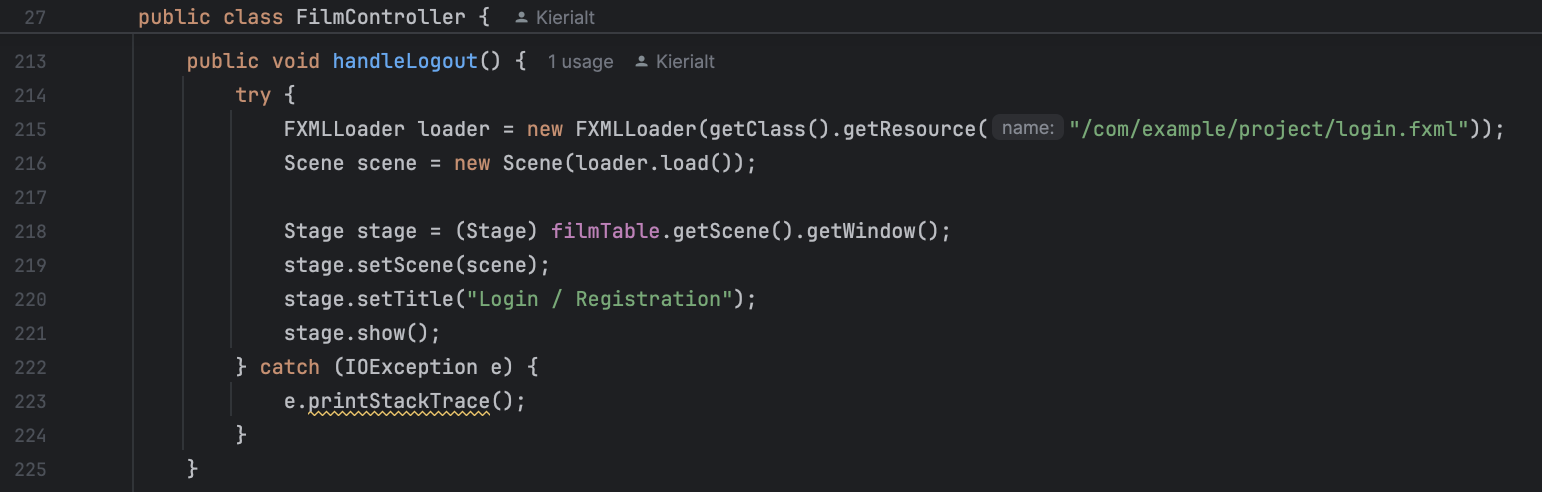
Pełna implementacja metody handleAddFilm():



### **5. Wylogowanie użytkownika**

* **Opis:**  
  Użytkownik może się wylogować i wrócić do ekranu logowania.
* **Klasa:** FilmController.java
* **Metoda:** handleLogout()
* **Działanie metody:**
  + Zamyka bieżące okno (Stage).
  + Ładuje ponownie login.fxml (ekran logowania).
  + Resetuje ewentualne dane użytkownika.

Pełna implementacja metody handleLogout():



***3. Architektura aplikacji***

Aplikacja została zaprojektowana w sposób modularny, zgodnie z zasadami separacji logiki, interfejsu i dostępu do danych. Każda klasa ma jasno określoną odpowiedzialność.

### ***1. Klasa Main.java***

* **Opis:**  
   Punkt wejścia aplikacji. Odpowiada za uruchomienie GUI i załadowanie początkowego widoku (login.fxml).
* **Główne zadania:**
  + Inicjalizacja JavaFX.
  + Ustawienie pierwszej sceny (ekranu logowania).
  + Konfiguracja podstawowego Stage.

### ***2. Klasa LoginController.java***

* **Opis:**  
   Kontroler dla ekranu logowania i rejestracji (login.fxml).
* **Główne zadania:**
  + Obsługa logowania użytkownika (handleLogin()).
  + Obsługa rejestracji nowego użytkownika (handleRegister()).
  + Przejście do głównego widoku (main.fxml) po udanym logowaniu.

### ***3. Klasa FilmController.java***

* **Opis:**  
   Kontroler dla głównego ekranu aplikacji (main.fxml), odpowiedzialny za zarządzanie filmami.
* **Główne zadania:**
  + Ładowanie listy filmów (loadFilms()).
  + Dodawanie nowych filmów (handleAddFilm()).
  + Wylogowanie użytkownika (handleLogout()).

### ***4. Klasa User.java***

* **Opis:**  
   Klasa modelu reprezentująca użytkownika systemu.
* **Pola:**
  + int id
  + String username
  + String password
* **Główne zadania:**
  + Przechowywanie informacji o użytkowniku po zalogowaniu.
  + Możliwość rozszerzenia o dodatkowe dane użytkownika.

### ***5. Klasa Film.java***

* **Opis:**  
   Klasa modelu reprezentująca pojedynczy film. Używana do przechowywania i przekazywania danych o filmach między warstwą bazy danych a interfejsem użytkownika.
* **Pola:**
  + int id – identyfikator filmu (klucz główny z bazy).
  + String title – tytuł filmu.
  + String genre – gatunek filmu.
  + int year – rok produkcji.
  + int watched - czy ten film został obejrzany.
* **Główne zadania:**
  + Przechowywanie danych filmów w formie obiektów.
  + Współpraca z TableView w main.fxml (JavaFX automatycznie odczytuje dane przez tzw. properties/gettery).

### ***6. Klasa DatabaseManager.java***

* **Opis:**  
   Warstwa pośrednicząca między aplikacją a bazą danych. Zarządza połączeniem oraz zapytaniami SQL.
* **Główne zadania:**
  + Tworzenie połączenia z bazą danych (connect()).
  + Rejestracja użytkownika.
  + Weryfikacja danych logowania.
  + Pobieranie listy filmów.
  + Dodawanie nowych filmów.

### ***7. Pliki FXML***

* **Opis:**  
   Pliki opisujące graficzny interfejs użytkownika w formacie XML.
* **Pliki:**
  + login.fxml — ekran logowania i rejestracji.
  + main.fxml — ekran główny z listą filmów i przyciskiem dodawania/wylogowania.
* **Główne zadania:**
  + Oddzielenie warstwy widoku od logiki.
  + Umożliwienie łatwej edycji interfejsu.

### ***8. Plik dialog-style.css***

* **Opis:**  
   Plik arkusza stylów CSS używany do personalizacji wyglądu okien dialogowych (Alert) i innych komponentów GUI.
* **Główne zadania:**
  + Nadanie aplikacji jednolitego i estetycznego wyglądu.
  + Stylowanie kolorów, czcionek, marginesów, krawędzi itp.
  + Może być przypięty do sceny w kodzie Java.

## ***4. Baza danych***

Aplikacja korzysta z lokalnej bazy danych (SQLite), w której przechowywane są dane użytkowników oraz filmy. W konfiguracji połączenia z bazą danych zastosowano ścieżkę względną zamiast bezwzględnej, co ułatwia uruchamianie aplikacji na różnych stanowiskach.

### ***1. Tabela users***

* **Opis:**  
   Przechowuje dane użytkowników zarejestrowanych w systemie.
* **Struktura:**

|  |  |  |
| --- | --- | --- |
| **Kolumna** | **Typ danych** | **Opis** |
| id | INTEGER | Klucz główny (autoincrement) |
| username | TEXT | Unikalna nazwa użytkownika |
| password | TEXT | Hasło użytkownika |

* **Uwaga:**  
   Hasła użytkowników nie są obecnie szyfrowane — planowana jest implementacja funkcji ich haszowania w przyszłości.

### **2. Tabela films**

* **Opis:**  
   Przechowuje dane filmów dodanych przez użytkowników.
* **Struktura:**

|  |  |  |
| --- | --- | --- |
| **Kolumna** | **Typ danych** | **Opis** |
| id | INTEGER | Klucz główny (AUTOINCREMENT) – identyfikator filmu |
| title | TEXT | Tytuł filmu |
| genre | TEXT | Gatunek filmu |
| year | INTEGER | Rok produkcji |
| watched | INTEGER | Czy film został obejrzany (0 – nie, 1 – tak) |
| user\_id | INTEGER | Klucz obcy powiązany z tabelą users |

* **Zarządzanie:**  
   Dodawanie i pobieranie filmów realizowane jest przez metody DatabaseManager (initialize(), getConnection()) oraz metody FilmController (saveFilmToDatabase(...)).

## **5. Instrukcja uruchomienia**

1. Sklonować repozytorium projektu lub rozpakować plik .zip.
2. Uruchomić projekt w środowisku obsługującym JavaFX (np. IntelliJIDEA).
3. Uruchomić klasę Main.
4. W oknie logowania można się zarejestrować lub zalogować na istniejące konto.
5. Po zalogowaniu otworzy się okno z listą filmów.