Casa de Filme

Documentatie

Poponet Tiberiu-Sergiu

Grupa 30231

Cuprins:

1. Enunt
2. Instrumente utilizate
3. Justificarea limbajului de programare ales
4. Descrierea diagramei UML
5. Descrierea aplicatiei
6. Descriere cod
7. Concluzie

# Enunt

(Problema 19):

***Problema 19***

Dezvoltați o aplicație care poate fi utilizată de către o **casă de producție filme**. Aplicația va avea 3 tipuri de utilizatori: angajat, manager și administrator.

Utilizatorii de tip **angajat** pot efectua următoarele operații după autentificare:

1. Vizualizarea listei filmelor sortată după tip și anul realizării (vizualizarea include și redarea unor imagini din film; între 1 și 3 imagini pentru fiecare film);
2. Filtrarea filmelor după anumite criterii: tip film (artistic sau serial), categorie film, anul realizării;
3. Operații CRUD în ceea ce privește persistența filmelor;
4. Căutarea unui film după titlu;
5. Salvare liste cu informații despre filme în mai multe formate: csv, json, xml, doc.

Utilizatorii de tip **manager** pot efectua următoarele operații după autentificare:

1. Vizualizarea listei filmelor sortată după tip și anul realizării (vizualizarea include și redarea unor imagini din film; între 1 și 3 imagini pentru fiecare film);
2. Filtrarea filmelor după anumite criterii: tip film (artistic sau serial), categorie film, anul realizării;
3. Căutarea unui film după titlu;
4. Salvare liste cu informații despre filme în mai multe formate: csv, json, xml, doc;
5. Vizualizarea unor statistici legate de filme utilizând grafice (structură radială, structură inelară, etc.).

Utilizatorii de tip **administrator** pot efectua următoarele operații după autentificare:

1. Operații CRUD pentru informațiile legate de utilizatorii aplicației;
2. Vizualizarea listei utilizatorilor și filtrarea acesteia după tipul utilizatorilor.

# Interfața grafică a aplicației va fi disponibilă în cel puțin 3 limbi de circulație internațională.

# Instrumente utilizate

* Java

Descriere: Java este un limbaj de programare și o platformă de dezvoltare utilizată pe scară largă pentru dezvoltarea diverselor tipuri de aplicații, de la aplicații web la aplicații de desktop și aplicații mobile.

Utilizare în proiect: Java a fost utilizat pentru a dezvolta aplicația în sine, inclusiv logica de afișare, manipularea datelor și interacțiunea cu baza de date.

* XAMPP

Descriere: XAMPP este un pachet software gratuit și open-source care conține un set de aplicații web populare, cum ar fi Apache, MySQL, PHP și Perl. XAMPP facilitează instalarea și configurarea unui mediu de dezvoltare local pentru dezvoltarea și testarea aplicațiilor web.

Utilizare în proiect: XAMPP este folosit pentru a configura un mediu de dezvoltare local pe computer. Acesta oferă un server web Apache pentru a rula aplicația, MySQL pentru baza de date, precum și PHP pentru logica aplicației.

* MySQL

Descriere: MySQL este un sistem de gestionare a bazelor de date relaționale (RDBMS) open-source și unul dintre cele mai populare sisteme de baze de date în lume. Este cunoscut pentru performanța sa, fiabilitate și ușurința de utilizare.

Utilizare în proiect: MySQL este utilizat pentru a stoca datele aplicației. Aici sunt păstrate informațiile despre filme, utilizatori, etc.

* phpMyAdmin

Descriere: phpMyAdmin este o aplicație web gratuită și open-source, care oferă o interfață grafică pentru administrarea bazelor de date MySQL. Cu ajutorul phpMyAdmin, poți crea, modifica și gestiona baze de date și tabele, puteți executa interogări SQL și multe altele.

phpMyAdmin este utilizat pentru a administra baza de date MySQL utilizată de aplicație. Folosind interfața sa grafică, poți face modificări în structura bazei de date, poți insera date, poți rula interogări SQL și multe altele.

* Java Swing

Descriere: Java Swing este un framework GUI pentru Java. A fost introdusă inițial ca o alternativă mai ușoară și mai flexibilă la AWT (Abstract Window Toolkit), care este framework-ul GUI original din Java.

Componente GUI: Java Swing oferă o gamă largă de componente GUI pe care le poți utiliza pentru a construi interfețe grafice pentru aplicațiile tale. Acestea includ butoane, casete de text, etichete, liste, tabele, ferestre, meniuri, etc.

Evenimente și acțiuni: Java Swing oferă un model de gestionare a evenimentelor care permite dezvoltatorilor să reacționeze la acțiunile utilizatorilor, cum ar fi clicurile de mouse sau tastatura. Aceste evenimente sunt utilizate pentru a controla interacțiunea utilizatorului cu aplicația.

Layout Managers: Pentru a organiza componentele GUI într-o manieră flexibilă și scalabilă, Java Swing utilizează layout managers. Aceștia gestionează poziționarea și dimensiunile componentelor în cadrul containerelor GUI, permițând o adaptare ușoară la diferite dimensiuni de ecran sau la schimbări de conținut.

În cadrul proiectului, Java Swing poate e utilizat pentru a crea interfața grafică a aplicației de administrare a filmelor. Poți construi ferestre pentru afișarea listei de filme, detalii despre filme, adăugarea, actualizarea sau ștergerea de filme, gestionarea categoriilor și multe altele.

- Intellij

IntelliJ IDEA este un mediu de dezvoltare integrat (IDE) puternic și popular pentru limbajul de programare Java și pentru alte limbaje, cum ar fi Kotlin, Scala, Groovy și multe altele. Este dezvoltat de compania JetBrains și este recunoscut pentru funcționalitățile sale avansate, ușurința de utilizare și performanța excelentă. Iată de ce l-am utilizat în dezvoltarea aplicației de gestionare a unei case de filme:

Interfață Utilizator Intuitivă: IntelliJ oferă o interfață utilizator prietenoasă și intuitivă, care facilitează navigarea și utilizarea IDE-ului.

Autocompletare Inteligentă: Este cunoscut pentru funcția sa de autocompletare inteligentă, care furnizează sugestii de cod, metode, variabile și clase în timp ce scrii cod. Acest lucru poate accelera semnificativ procesul de dezvoltare.

Verificare Statică și Corectare Automată a Erorilor: IntelliJ efectuează verificări statice ale codului în timp real și oferă sugestii pentru corectarea erorilor. Aceasta ajută la identificarea și remedierea problemelor de cod înainte de rularea aplicației.

Refactorizare Avansată: IDE-ul oferă o gamă largă de instrumente de refactorizare a codului, permițându-vă să reorganizați și să optimizați structura codului cu ușurință.

Suport Puternic pentru Java și Framework-uri: IntelliJ oferă suport excelent pentru Java și diverse framework-uri Java, cum ar fi Spring, Hibernate, Maven și altele. Aceasta înseamnă că poți lucra eficient cu tehnologii Java și frameworkuri asociate.

Integrare cu VCS (Version Control Systems): IntelliJ se integrează bine cu sistemele de control al versiunilor, cum ar fi Git, SVN și Mercurial. Acest lucru facilitează gestionarea codului sursă și colaborarea în echipe.

Plugin-uri Extensibile: IntelliJ oferă o gamă largă de plugin-uri extensibile, care pot fi instalate pentru a adăuga funcționalități suplimentare IDE-ului. De exemplu, poți adăuga suport pentru alte limbaje de programare sau frameworkuri prin instalarea plugin-urilor corespunzătoare.

Performanță și Stabilitate: IntelliJ este cunoscut pentru performanța sa excelentă și stabilitatea în timpul dezvoltării. Acest lucru este esențial pentru o experiență de dezvoltare fără probleme și eficientă.

# Justificarea limbajului ales

Java este un limbaj de programare de nivel înalt, orientat pe obiecte și cu o platformă de dezvoltare extrem de populară. Creat de către Sun Microsystems (acum deținut de Oracle Corporation), Java a fost lansat inițial în 1995 și de atunci a devenit unul dintre cele mai utilizate limbaje de programare din lume.

Orientat pe Obiect: Java este un limbaj de programare complet orientat pe obiect, ceea ce înseamnă că toate entitățile în Java sunt obiecte, iar operațiile se efectuează prin intermediul acestor obiecte.

Platformă Independentă: Java a fost conceput pentru a fi un limbaj portabil, ceea ce înseamnă că codul Java poate fi rulat pe orice platformă care are instalată mașina virtuală Java (JVM). Acest lucru face Java foarte popular pentru aplicații web și dezvoltarea de aplicații multi-platformă.

Sintaxă Simplă și Clară: Java a fost proiectat pentru a fi ușor de înțeles și de utilizat. Sintaxa sa este similară cu cea a altor limbaje de programare populare precum C++ și C#, ceea ce face tranzitia mai ușoară pentru dezvoltatori.

Java este cunoscut pentru fiabilitatea sa. Este un limbaj care pune accent pe gestionarea erorilor, ceea ce este esențial în aplicații critice cum ar fi cea de administrare a filmelor. Prin intermediul sistemului său de gestionare a memoriei și a tipurilor de date robuste, Java oferă o mai mare siguranță în ceea ce privește manipularea datelor și evitarea erorilor comune, cum ar fi pointerii greșit sau memoria coruptă.

Java este potrivită și pentru implementarea arhitecturii Model-View-ViewModel (MVVM) și pentru respectarea principiilor GRASP. Arhitectura MVVM îți permite să separi clar logica de afaceri (Model), interfața utilizator (View) și logica de prezentare (ViewModel), ceea ce poate facilita dezvoltarea unei aplicații mai ușor de întreținut și de testat. Principiile GRASP (General Responsibility Assignment Software Patterns) îți oferă un set de ghiduri pentru proiectarea arhitecturii software, facilitând dezvoltarea unui cod modular și ușor de înțeles.

Am ales Java deoarece am folosit limbajul si sunt obisnuit cu el din anul II de la materia T.P., unde am avut de facut 3 proiecte in Java:

Proiectul 1: Calculator de Polinoame

În cadrul acestui proiect, am fost provocat să dezvolt un calculator de polinoame folosind limbajul de programare Java. Scopul acestui calculator era să permit utilizatorului să efectueze operații aritmetice și algebrice cu polinoame, precum adunarea, scăderea, înmulțirea, împărțirea și derivarea. Acest proiect a implicat aplicarea conceptelor de analiză matematică și algebră, precum și gestionarea corectă a datelor de intrare și ieșire.

Proiectul 2: Sistem de Gestiune al Cozilor de la un Magazin

Pentru acest proiect, am dezvoltat un sistem de gestiune al cozilor de la un magazin folosind Java. Acest sistem a fost conceput pentru a gestiona intrările și ieșirile din cozi de la un magazin, păstrând evidența prețurilor, cantităților și vânzărilor, cât și timpul (durata) finalizării coșului de produce al persoanei și trimiterea sa la o casă cât mai potrivită, pentru un timp de gestionare mai bun. Aici, am lucrat cu concepte de programare orientată pe obiecte, cum ar fi crearea de clase pentru produse, cozi de așteptare pentru clienți, gestionarea coziloretc. Acest proiect a implicat și dezvoltarea unei interfețe grafice simple pentru utilizatori.

Proiectul 3: Sistem de Gestiune a unui Depozit Alimentar cu Baze de Date

MySQL

Ultimul proiect a fost un sistem de gestiune a unui depozit alimentar, unde am integrat utilizarea bazelor de date MySQL. Aici, am creat o bază de date pentru a stoca informații despre produse, stocuri, furnizori și comenzi. Scopul acestui proiect a fost de a gestiona eficient stocurile, de a plasa comenzi de la furnizori și de a urmări vânzările. Am folosit Java pentru a dezvolta logica aplicației și interacțiunea cu baza de date MySQL pentru a accesa și modifica datele. Acest proiect a implicat aplicarea cunoștințelor de bază de bază de date și SQL, precum și integrarea acestora într-o aplicație Java funcțională.

**Descrierea diagramelor UML**

Diagrama Use Case:

A diagram of a network

Description automatically generated

* Login: Toți utilizatorii trebuie să se autentifice în sistem pentru a accesa funcționalitățile.
* Vizualizare Filme: Tipurile de utilizatori angajat si manager pot vizualiza lista filmelor.Lista filmelor poate fi sortată după tip și anul realizării.
* Filtrare Filme: Tipurile de utilizatori angajat si manager pot filtra filmele după diferite criterii, cum ar fi tipul filmului (artistic sau serial), categoria filmului sau anul realizării.
* Operații CRUD Filme: Angajații pot efectua operații CRUD (Adaugare, Cautare, Actualizare, Ștergere) în ceea ce privește persistența filmelor. Aceste operații includ adăugarea unui nou film, vizualizarea detaliilor unui film, actualizarea informațiilor unui film și ștergerea unui film din sistem.
* Vizualizare Utilizatori: Administratorii pot vizualiza lista utilizatorilor din sistem. Această funcționalitate permite administratorilor să vadă informații despre ceilalți utilizatori ai aplicației.
* Operații CRUD Utilizatori: Administratorii au privilegii pentru a efectua operații CRUD (Adaugare, Cautare, Actualizare, Ștergere) legate de utilizatori. Aceste operații includ crearea unui nou utilizator, vizualizarea detaliilor unui utilizator, actualizarea informațiilor unui utilizator și ștergerea unui utilizator din sistem.
* Salvare Liste Filme sub tipuri CSV, DOC, XML, JSON de catre Angajat sau Manager
* Cautare Film dupa Tip de catre Angajat sau Manager
* Cautare User dupa tip, realizata de Administrator

In plus, angajatul poate vizualiza imagini ale filmelor, iar managerul poate vizualiza grafice referitore la statistici despre filme. Toti utilizatori pot selecta limba dorita dintre opriunile: engleza, romana, franceza.

A screenshot of a computer

Description automatically generated

(Grafice)

A person holding a rose

Description automatically generated

(Imagini)

Diagrama de clase:

A screenshot of a computer program

Description automatically generated

(Intellij)

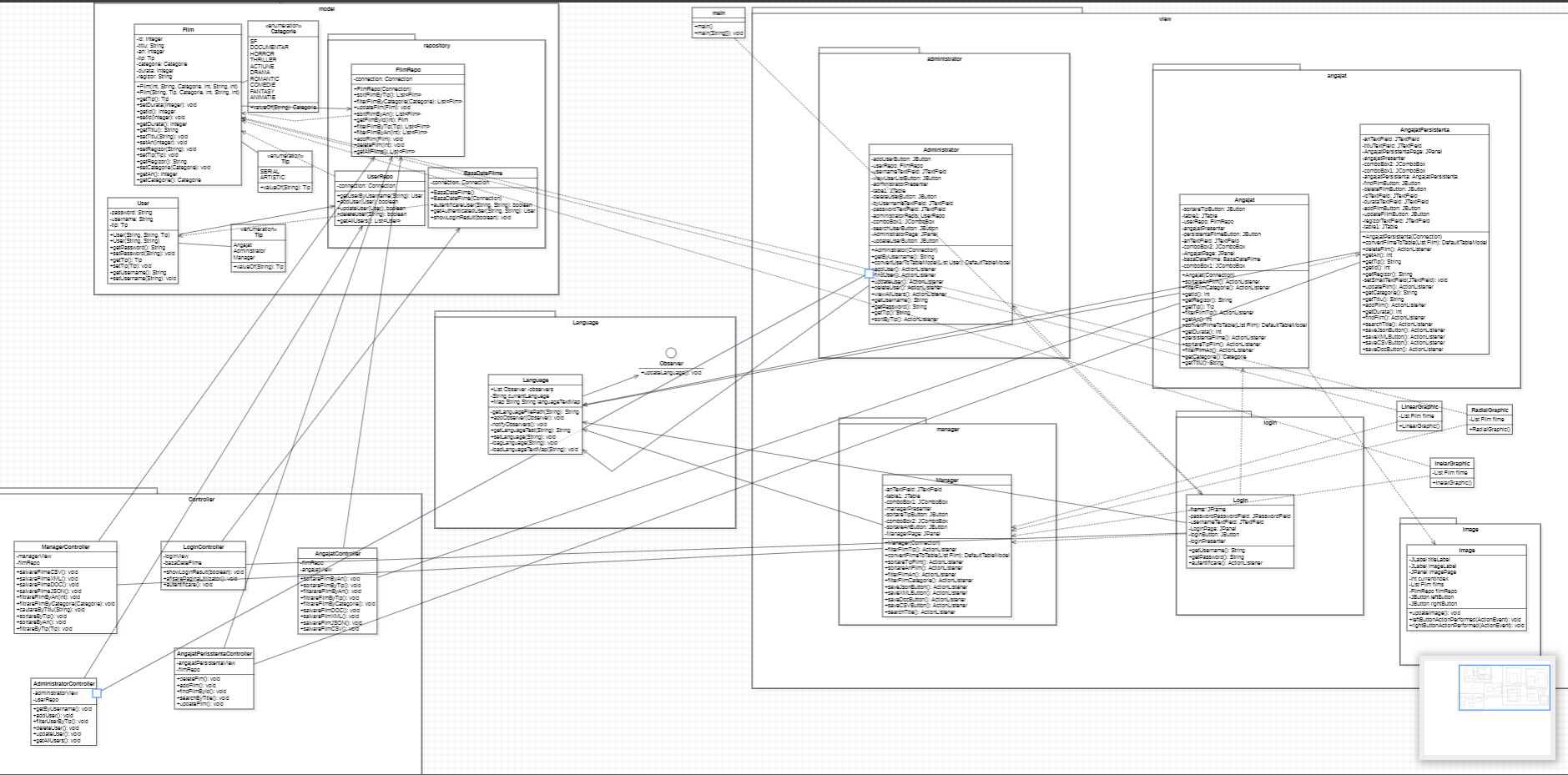
 (Clasele si implementarea lor se va prezenta mai jos, la sectiunea descrierea aplicatiei).

Diagrama Entitate-Relatie:

A screenshot of a computer

Description automatically generated

(tabelele din baza de date).

# Descrierea aplicatiei

Casa Film este o aplicație Java creată pentru gestionarea unei biblioteci de filme. Scopul principal al aplicației este de a permite utilizatorilor să gestioneze și să exploreze colecția de filme disponibile în bibliotecă. De asemenea, aplicația oferă funcționalități de adăugare, actualizare și ștergere a filmelor, gestionarea informațiilor despre fiecare film, precum și căutarea și filtrarea filmelor în funcție de diferite criterii. De asemenea, aplicatia permite gestiunea utilizatorilor și operații de adăugare, actualizare și ștergere a lor, gestionarea informațiilor despre fiecare utilizator.

În cadrul proiectului, am implementat un sistem care gestionează informațiile despre filme și utilizatori într-o bază de date. Am definit clasele Film și User, care reprezintă modelele de date pentru filme și utilizatori, respectiv. Aceste clase conțin atributele corespunzătoare și metode pentru a accesa și manipula aceste date.

Pentru a facilita interacțiunea cu baza de date, am creat clasele FilmRepo și UserRepo, care conțin operațiile CRUD (Create, Read, Update, Delete) necesare pentru gestionarea datelor din baza de date. Aceste clase sunt responsabile pentru adăugarea, recuperarea, actualizarea și ștergerea datelor din baza de date.

De asemenea, am definit o clasă DAOException pentru a gestiona excepțiile care pot apărea în timpul interacțiunii cu baza de date.

Pentru a salva datele despre filme în diferite formate (DOC, CSV, JSON, XML), am implementat metode specifice în cadrul clasei FilmRepo. Aceste metode permit exportul datelor din baza de date în diferite formate de fișiere.

În cadrul clasei UserRepo, am implementat operațiile CRUD pentru gestionarea utilizatorilor din baza de date. De asemenea, am adăugat o metodă pentru filtrarea utilizatorilor după tip.

Administrator: Această clasă definește interfața grafică pentru un utilizator cu rol de administrator. Ea conține butoane pentru diverse acțiuni precum adăugarea, actualizarea, căutarea și ștergerea utilizatorilor. De asemenea, oferă posibilitatea de a filtra utilizatorii după un anumit criteriu și de a schimba limba interfeței. Clasa Administrator utilizează un obiect AdministratorController pentru a gestiona acțiunile utilizatorului și a comunica cu baza de date.

Angajat: Această clasă definește interfața grafică pentru un utilizator cu rol de angajat. Acesta permite angajatului să sorteze și să filtreze filmele în funcție de an, categorie și tip. De asemenea, oferă posibilitatea de a salva filmele în diferite formate (CSV, XML, JSON, DOC) și de a schimba limba interfeței. Clasa Angajat utilizează un obiect AngajatController pentru a gestiona acțiunile utilizatorului și a comunica cu baza de date.

Image: Această clasă definește interfața grafică pentru vizualizarea imaginilor asociate filmelor. Utilizatorul poate naviga între imagini folosind butoanele și poate vedea titlul filmului asociat imaginii curente. Această clasă nu comunică direct cu baza de date, ci utilizează o instanță a clasei FilmRepo pentru a obține informații despre filme.

Manager: Această clasă definește interfața grafică pentru un utilizator cu rol de manager. Acesta permite managerului să sorteze și să filtreze filmele, să caute filme după titlu și să salveze filmele în diferite formate. De asemenea, oferă posibilitatea de a genera diferite tipuri de grafice (linie, inelar, radial) pentru a vizualiza datele despre filme. Clasa Manager utilizează un obiect ManagerController pentru a gestiona acțiunile utilizatorului și a comunica cu baza de date.

Pachetul com.example.casafilme.view.images conține o clasă separată pentru afișarea imaginilor asociate filmelor, iar pachetul com.example.casafilme.view conține clase pentru afișarea graficelor.

Controller: aceste clase gestionează logica de afaceri și interacționează cu modelele și vizualizările din aplicație. De exemplu, AdministratorController are metode pentru adăugarea, actualizarea, ștergerea și filtrarea utilizatorilor în funcție de rolul lor.

AngajatController gestionează operațiile legate de filme pe care un angajat le poate efectua, cum ar fi sortarea și filtrarea lor.

AngajatPersistentaController se ocupă de operațiile legate de persistența filmelor în baza de date sau în fișiere, cum ar fi adăugarea, actualizarea, ștergerea și căutarea filmelor.

LoginController este responsabil pentru autentificarea utilizatorilor și pentru redirecționarea lor către interfețele corespunzătoare, în funcție de rolul lor.

ManagerController gestionează operațiile pe filme pe care le poate efectua un manager, cum ar fi sortarea, filtrarea și căutarea lor, precum și salvarea lor în diferite formate de fișiere.

În ansamblu, această arhitectură urmează principiile MVC (Model-View-Controller), unde modelul reprezintă datele și logica de business, iar repositoriile acționează ca interfațe între model și baza de date. Această structură clară și modulară facilitează dezvoltarea, testarea și întreținerea proiectului.

**Descriere Cod**

Clasa UserRepo este responsabilă de interacțiunea cu baza de date pentru operațiile CRUD legate de utilizatori în cadrul aplicației de gestionare a casei de producție filme. Aceasta conține metode pentru adăugarea, citirea, actualizarea și ștergerea informațiilor despre utilizatori din baza de date. Utilizând obiecte de tip User, clasa manipulează datele asociate utilizatorilor, inclusiv numele de utilizator, parola și tipul de utilizator (administrator, angajat sau manager). Prin intermediul conexiunii la baza de date, clasa comunică și gestionează datele în mod eficient, asigurând consistența și securitatea informațiilor. Implementând aceste metode, clasa oferă un nivel de abstractizare și modularitate, facilitând administrarea și gestionarea utilizatorilor în cadrul aplicației.

Clasa BazaDateFilme reprezintă o componentă cheie a aplicației de gestionare a casei de producție filme, responsabilă de gestionarea conexiunii și a interacțiunii cu baza de date MySQL. Aceasta conține metode pentru autentificarea utilizatorilor în baza de date, verificând validitatea numelui de utilizator și a parolei. Prin intermediul acestei clase, se realizează conexiunea la baza de date utilizând JDBC și se implementează logica de verificare a autentificării. Metoda autentificareUser verifică autentificarea, iar getAuthenticatedUser obține detaliile utilizatorului autentificat, inclusiv numele de utilizator, parola și tipul de utilizator. Clasa oferă și un mesaj de feedback, indicând succesul sau eșecul procesului de autentificare. Utilizând aceste funcționalități, clasa facilitează securitatea și gestionarea utilizatorilor în cadrul aplicației.

Clasa FilmRepo este responsabilă pentru interacțiunea cu baza de date în ceea ce privește entitățile User și Film din aplicația de gestionare a casei de producție filme. Aceasta oferă metode pentru sortarea filmelor după tip și anul realizării, filtrarea filmelor în funcție de tip, categorie și an, precum și operațiile CRUD pentru filme. Utilizând PreparedStatement-uri, clasa realizează interogări adecvate către baza de date pentru a obține și manipula datele necesare. De asemenea, implementează metode pentru adăugarea, obținerea, actualizarea și ștergerea filmelor din baza de date, oferind un mecanism eficient de gestionare a datelor referitoare la filme în aplicație.

Clasa Film reprezintă o entitate în aplicația de gestionare a casei de producție filme, care conține informații despre un film specific. Ea are câteva atribute, precum id, titlu, tip, categorie, an, regizor și durata, ce reprezintă caracteristicile esențiale ale unui film. Folosind enum-urile Tip și Categorie, clasa definește categoriile și tipurile de filme disponibile, precum artistic sau serial, și categoriile de genuri precum dramă, comedie, acțiune etc. Constructorii clasei permit inițializarea obiectelor de tip Film cu informațiile necesare despre un film. De asemenea, clasa oferă metode pentru a accesa și actualiza atributele unui film, cum ar fi getTitlu(), getTip(), getCategorie(), getAn(), getRegizor(), getDurata(), setTitlu(), setTip(), setCategorie(), setAn(), setRegizor() și setDurata(), facilitând astfel manipularea datelor filmului în cadrul aplicației.

Clasa User reprezintă un utilizator al aplicației de gestionare a casei de producție filme, având atribute precum username, password și tip. Ea conține două constructori, unul pentru a inițializa un utilizator cu username, parolă și tip, și unul pentru a crea un utilizator cu username și parolă. Enum-ul Tip definește categoriile posibile de utilizatori: Angajat, Manager și Administrator. Clasa oferă metode pentru a accesa și modifica informațiile utilizatorului, cum ar fi getUsername(), getPassword(), getTip(), setUsername(), setPassword() și setTip().

AuthService furnizează funcționalitățile necesare pentru autentificarea utilizatorilor în aplicație. Aceasta inițializează o instanță a clasei BazaDateFilme în constructor, care gestionează conexiunea și interacțiunea cu baza de date MySQL. Metoda authenticateUser(username, password) verifică autentificarea utilizatorului pe baza credențialelor furnizate, în timp ce getAuthenticatedUser(username, password) returnează informațiile despre utilizatorul autentificat. Metoda showLoginResult(boolean bol) afișează rezultatul procesului de autentificare, iar getConnection() furnizează conexiunea la baza de date pentru alte operațiuni.

FilmService este responsabil pentru gestionarea datelor despre filme în aplicație. Constructorul său initializează o instanță a clasei AuthService pentru acces la conexiunea la baza de date. Metoda addFilm(Film film) permite adăugarea unui film în baza de date, iar getFilmById(int id) returnează informații despre un film specific pe baza ID-ului furnizat. updateFilm(Film film) actualizează detaliile despre un film existent în baza de date, în timp ce deleteFilm(int id) șterge un film în funcție de ID-ul dat. Aceste clase facilitează operațiile de autentificare și manipulare a datelor despre filme, respectând principiul responsabilității unice și asigurând un cod modular și ușor de întreținut.

In View avem:

Login: acest subpachet conține interfețe și clase GUI specializate pentru gestionarea paginii de conectare. Utilizatorii pot introduce datele de autentificare în câmpurile corespunzătoare și pot apăsa butonul "Conectare" pentru a accesa aplicația. Clasa LoginView este responsabilă pentru gestionarea interfeței de conectare și pentru comunicarea cu controller-ul asociat.

Administrator: subpachetul Administrator conține interfețe și clase GUI dedicate funcționalităților disponibile pentru administratori. Acestea includ operațiuni precum adăugarea, actualizarea și ștergerea utilizatorilor din sistem, precum și vizualizarea și căutarea acestora. Clasa AdministratorView se ocupă de gestionarea acestei interfețe și de comunicarea cu controller-ul corespunzător.

Angajat: aici sunt definite interfețe și clase GUI pentru afișarea funcționalităților disponibile pentru angajați. Acestea pot include operațiuni de filtrare și sortare a filmelor, adăugare sau actualizare a informațiilor despre filme și alte activități relevante. Clasa AngajatView este responsabilă pentru gestionarea acestei interfețe și pentru comunicarea cu controller-ul asociat.

Manager: subpachetul Manager conține interfețe și clase GUI specializate pentru interacțiunea cu funcționalitățile destinate managerilor. Acestea pot include gestionarea și vizualizarea datelor despre filme, generarea de rapoarte și alte operațiuni specifice acestui rol. Clasa ManagerView se ocupă de gestionarea interfeței de utilizator și de comunicarea cu controller-ul corespunzător.

Clasa Image furnizează o interfață grafică pentru afișarea imaginilor și titlurilor filmelor. Utilizând butoanele de navigare, utilizatorul poate trece prin lista de filme disponibile și vizualiza detaliile acestora. Această interfață oferă o experiență interactivă și intuitivă pentru explorarea colecției de filme vizual.

Clasele de controller sunt responsabile pentru gestionarea logicii de afaceri și comunicarea între interfețele grafice și baza de date a aplicației. Ele oferă funcționalități specifice pentru diferitele roluri de utilizatori, precum adăugarea, actualizarea și ștergerea de informații, sortarea și filtrarea datelor sau salvarea acestora în diferite formate. Astfel, fiecare controller își îndeplinește rolul în coordonarea acțiunilor utilizatorului și manipularea datelor în conformitate cu cerințele aplicației.

Fiecare clasă din aceste subpachete este proiectată pentru a oferi o interfață intuitivă și ușor de utilizat pentru utilizatorii cu privilegii specifice în cadrul aplicației de gestionare a casei de producție de filme.

**Concluzie**

Casa de Film este o aplicație Java robustă și bine structurată, creată pentru a gestiona eficient o bibliotecă de filme într-o casă de producție. Cu funcționalități diferite pentru utilizatorii de tip angajat, manager și administrator, aplicația oferă o experiență simplă și

Utilizarea instrumentelor precum Java, MySQL, Java Swing, XAMPP și Intellij a oferit o bază solidă pentru dezvoltarea și implementarea aplicației. Java a fost alegerea perfectă pentru limba de programare datorită orientării sale pe obiect, portabilității și fiabilității. De asemenea, Java Swing a oferit o interfață grafică flexibilă și ușor de utilizat pentru construirea paginilor aplicației.

În final, Casa de Film reprezintă un exemplu excelent de aplicare a conceptelor și tehnicilor de programare Java într-un proiect real. Prin intermediul acestei aplicații, utilizatorii pot gestiona cu ușurință colecția de filme, utilizând un sistem sigur, eficient și intuitiv.

Această aplicație nu numai că îndeplinește cerințele inițiale ale problemei propuse, dar și demonstrează un nivel ridicat de competență în dezvoltarea software și o înțelegere profundă a principiilor de proiectare și programare. Cu un cod bine structurat, o interfață prietenoasă și o funcționalitate extinsă, Casa de Film este un instrument valoros pentru orice casă de producție care își dorește o modalitate eficientă de gestionare a bibliotecii lor de filme.