



UNIVERSITATEA TEHNICĂ

DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
CATEDRA CALCULATOARE

MOVIE HOUSE

-documentație-
-proiect nr.17-

Student:

Pop Ruxandra Maria
Anul 3, grupa 30236

1. Enunțul problemei

Scopul proiectului este dezvoltarea unei aplicații desktop care poate fi utilizată de o casă de producție filme. Aceasta aplicație va dispune de 2 tipuri de utilizatori: angajat și administrator, fiecare putând efectua diferite operații în funcție de rolul lui.

Obiectivul principal al acestei teme este familiarizarea cu şablonul arhitectural Model-View-ViewModel, cu şablonul comportamental Command și cu un şablon de proiectare creațional ales de noi, eu am ales să implementez Builder.

2. Instrumente utilizate

Înainte să ne apucăm de proiectarea și implementarea propriu zisă a aplicației, este necesar să cunoaștem ce cerințe trebuie acesta să indeplinească, pentru a descoperi ce trebuie să facă sistemul, cum trebuie să funcționeze, pentru a stii ce obiective dorim să atingem la sfârșitul implementării.

Astfel ca și primă etapă în realizarea temei, am construit diagrama de use case, care cuprinde actorii aplicației și operațiile pe care le pot face. Diagrama de use case a fost realizată cu ajutorul instrumentului software **starUML**, tot prin intermediul acestui instrument am proiectat și diagrama de clase, care respectă arhitectura MVP.

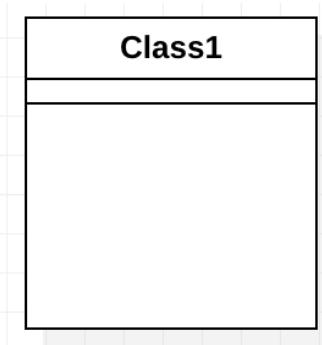


Fig2.1

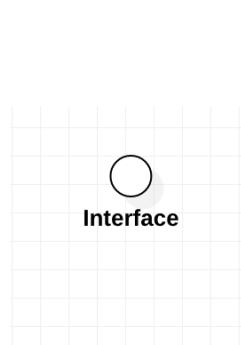


Fig2.2

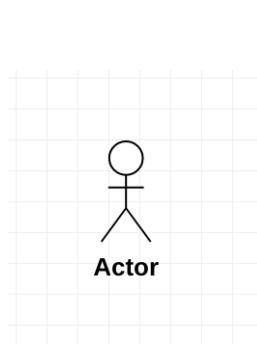


Fig2.3

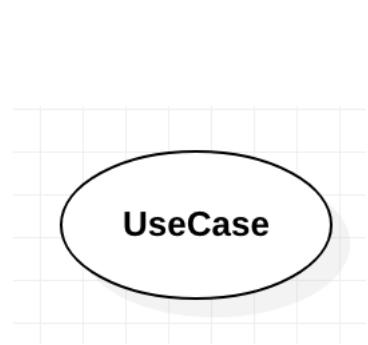


Fig2.4



Aplicația a fost implementată cu ajutorul mediulului de dezvoltare IDE IntelliJ IDEA 2021.2.2, iar ca limbaj de programare am ales să folosesc Java.

Fig2.5 IntelliJ icon

Java este un limbaj de programare orientat pe obiecte, a fost conceput de James Gosling la Sun Microsystems la începutul anilor 90 ,fiind lansat în 1995.

Am ales să folosesc acest limbaj ,pentru că mi se pare extrem de simplu să realizezi aplicații de tip GUI ,deoarece există framework-ul Java Swing ,care permite să realizezi interfețele intr-un mod mai prietenos,având o privire de asamblu asupra interfeței . Totodată , am vrut să îmi îmbunătățesc cunoștiințele despre acest limbaj de programare, deoarece în viitor îmi doresc să lucrez la proiecte de dimensiuni mai mari .

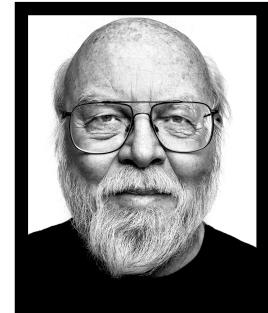


Fig2.6 James Gosling



Pentru persistență am utilizat o bază de date relațională și anume **phpMyAdmin**, unde am creat o bază de date numită **moviehouse** ,unde am creat două tabele : **movies** și **users**.

3. Descrierea diagramelor UML

3.1 Diagrama de use case

Diagrama de use case prezintă o colecție de cazuri de utilizare și actori.

În cazul acestei teme :

- actorul este utilizatorul(administratorul/angajatul)
- iar cazurile de utilizare sunt:

Pentru administrator

În caz de succes:

- Username-ul și parola sunt corecte și preluate din baza de date users
- Adaugă un nou utilizator completând toate field-urile necesare și apasă butonul Add
- Selectează utilizatorul a cărui date vrea să le modifice,completând field-urile necesare,toate detaliile despre utilizator înafară de username pot fi modificate.
- Totodată poate să steargă utilizatorul selectat prin apăsarea butonului Delete

- Poate să părasească contul prin apasarea butonului LogOut

În caz de nesucces:

- Username-ul sau/și parola nu sunt corecte, nu se găsesc în baza de date
- Nu selectează produsul pe care dorește să-l modifice
- Încearcă să modifice username-ul unui utilizator selectat
- Încearcă să adauge un nou utilizator care are același username ca un utilizator care există deja
- În toate situațiile se vor genera eroriile/avertizările necesare prin interfața grafică

Pentru angajat

În caz de succes:

- Username-ul și parola sunt corecte și preluate din baza de date users
- Adaugă un nou film completând toate field-urile necesare și apasă butonul Add,
- Selectează filmul a cărui date vrea să le modifice,completând field-urile necesare,toate detaliile despre film înfără de titlu pot fi modificate.
- Totodată poate să steargă filmul selectat prin apăsarea butonului Delete
- Poate să caute un film pe baza titlului acestuia
- Poate să filtreze filmele după numeroase criterii:tip film,categorie film,anul realizării
- Poate să genereze statistici în legătură cu numărul de filme dintr-o anumită categorie/tip
- Poate să salveze rapoarte cu informații despre filme în mai multe formate:csv,json,xml

În caz de nesucces:

- Username-ul și parola nu sunt corecte,nu se găsesc în baza de date
- Încearcă să adauge un nou film care are același titlu ca un alt film care există deja ,iar filmul nu este de tip serial.Dacă filmul are aceeași nume ca alt film ,dar ambele filme sunt de tip serial,dar anul realizări este același.
- Încearcă să modifice titlul unui film selectat
- În toate situațiile se vor genera eroriile/avertizările necesare prin interfața grafică

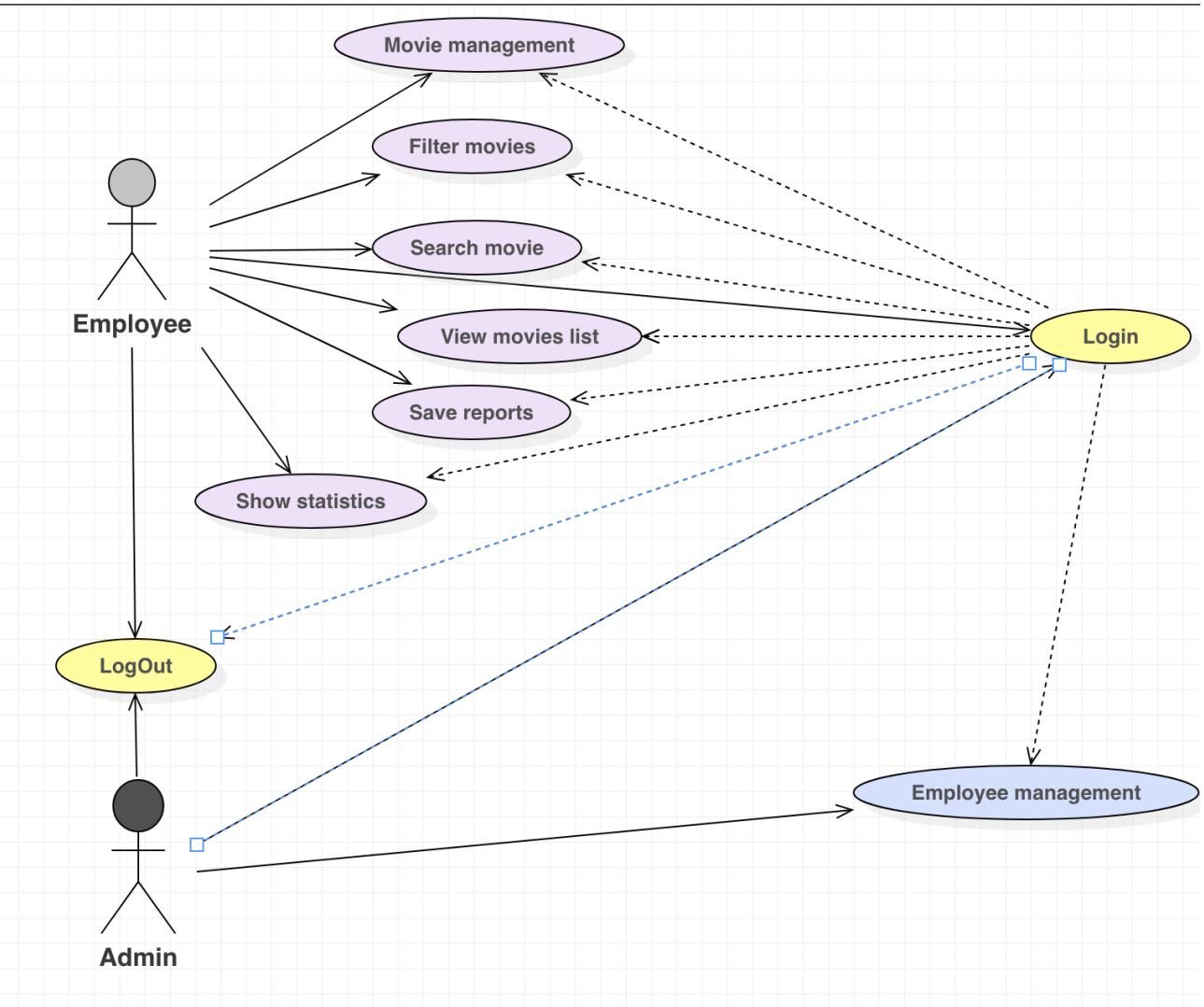


Figura 3.1.1.Diagrama de use case

3.2 Diagrama de clase

Reprezintă un set de clase, interfețe, colaborări și alte relații.

Diagrama de clase a fost realizată ,respectând modelul arhitectural Model-View-ViewModel și modelul comportamental Command.Astfel fiecare pachet ,clasa ,are un rol prestabilit.

MVP este un model arhitectural conceput pentru a facilita testarea automată a unitățiilor și pentru a îmbunătăți separarea preocupărilor în logica prezentării.

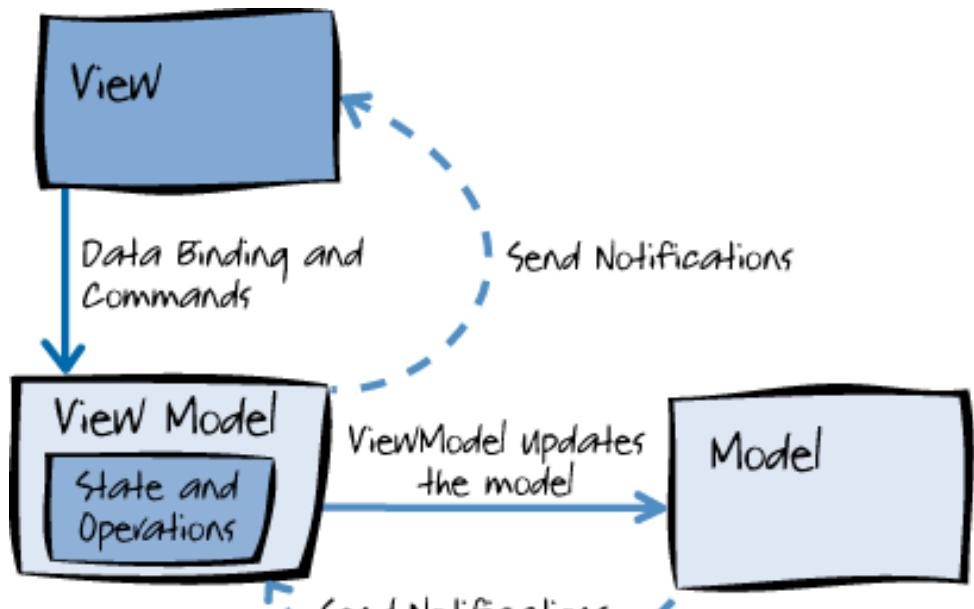
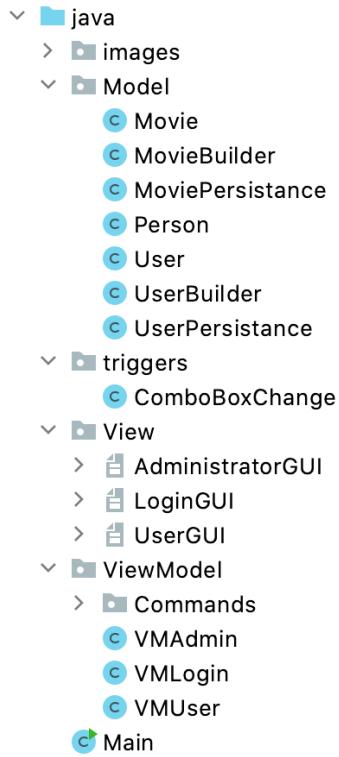


Fig 3.2.2 Modelul arhitectural MVVM

Fig 3.2.1 Pachetele cu clasele corespunzătoare

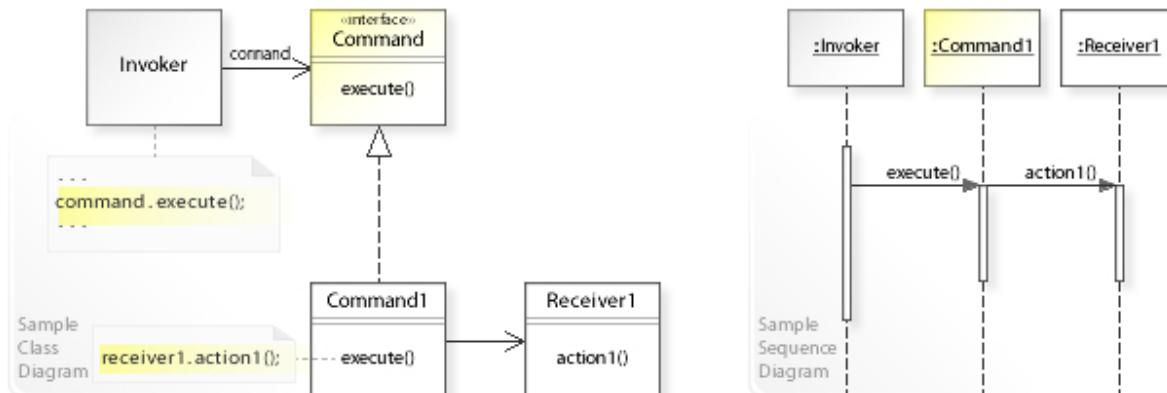


Fig 3.2.3 Modelul comportamental Command

- **Model** - se referă atât la un model de domeniu, care reprezintă conținutul real, cât și la nivelul de acces la baza de date

Cuprinde următoarele clase:

- Movie : cuprinde detalii despre film(titlu,categorie,tip,anul realizari)
- MoviePersistance : cuprinde o lista de filme asupra careia se vor realiza diferite metode care au ca scop scrierea/citirea/modificarea unei baze de date
- Person : cuprinde detaliiile despre o persoană din viața reală(nume și prenume)

- User : cuprinde detaliile despre utilizator(username,password,role)
- UserPersistance : cuprinde o lista de utilizatori asupra careia se vor realiza diferite metode care au ca scop scriere/citirea/modificarea unui fisier de tip xml
- UserBuilder și MovieBuilder sunt 2 clase folosite pentru implementarea sablonului creational Builder. Fiind o interfață pentru creare părțiilor unui obiect.

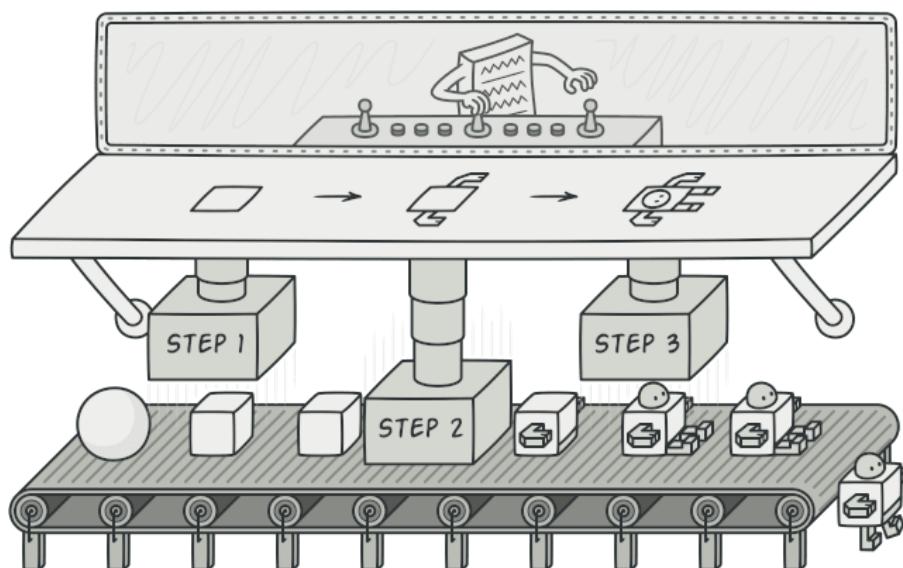


Fig 3.2.3 Modelul creational Builder

- **View** – reprezintă structura, aspectul a ceea ce un utilizator vede pe ecran. Afisează o reprezentare a modelului și primește interacțiunea utilizatorului cu interfața grafică și redirecționează manipularea acestora către modelul de vizualizare prin legarea de date care este definit pentru a lega view și.viewmodel

Cuprinde următoarele clase:

- AdministratorView: aici se proiectează interfață pt administrator prin numeroasele atrbute prezente, și se gasesc metodele pentru butoanele care realizează comunicarea cu componenta din presenter corespunzătoare administratorului
- UserView: aici se proiectează interfață pt angajat prin numeroasele atrbute prezente, și se gasesc metodele pentru butoanele care realizează comunicarea cu componenta din presenter corespunzătoare angajatului
- LoginView: aici se proiectează interfață pt pagina de login prin numeroasele atrbute prezente, și se gasesc metodele pentru butoanele care realizează comunicarea cu componenta din presenter corespunzătoare pagini de login

- **ViewModel**- reprezintă niște modele pentru view-uri, mai precis acestea se referă la o abstractizare a view-urilor care servesc și la binding-ul datelor între view și model. Pot fi privite și ca niște aspecte specializate ale Controalelor din design pattern-ul MVC (care acționează ca și data bindgins sau converters) în aşa fel încât să schimbe informația din formatul modelului în formatul view-ului și să paseze comenzi din view în model.

Cuprinde subpachetul **Command**:

Aici este definită interfața **ICommand**, interfață care cuprinde metoda `execute(int ok)` care va fi implementată de fiecare comandă în parte în funcție de necesitate. Cuprinde 9 clase, cu nume sugestive pentru comanda/comenzile pe care le realizează.

-`AddCommand`: reprezintă clasa care implementează comanda de adaugare a unui user , respectiv film , în funcție de valoarea parametrului `ok` din câmpul metodei `execute()`.

-`DeleteCommand`: reprezintă clasa care implementează comanda de stergere a unui user , respectiv film , în funcție de valoarea parametrului `ok` din câmpul metodei `execute()`.

-`FilterCommand`: reprezintă clasa care implementează comanda de filtrare a filmelor în funcție de tip,categorie,an

-`LoginCommand`: reprezintă clasa care implementează comanda de logare ca client / administrator în funcție de datele completeate în interfața GUI

-`SaveReportsCommand`: reprezintă clasa care implementează comanda de salvare a raportelor legate de filme, salvarea se poate face în 3 formate diferite:xml,json.,csv.

-`SearchMovieCommand`: reprezintă clasa care implementează comanda de căutare a unui film, în funcție de titlul inserat în campul `title` din interfața GUI

-`ShowCommand` : reprezintă clasa care implementează comanda de afisare a filmelor/utilizatorilor în tabelul din interfața GUI, în funcție de valoarea parametrului `ok` din interiorul metodei `execute()`.

-`ShowStatisticsCommand`: reprezintă clasa care implementează comanda de afisare a statisticilor legate de filme. Statisticile vor fi afisate în interfață grafică în 3 formate diferite.

-Update Command: reprezintă clasa care implementează comanda de actualizare a unui user ,respectiv film , în funcție de valoarea parametrului ok din câmpul metodei execute().Pot fi actualizate diferite campuri , înafara de titlu de la film, respectiv username de la user.

Cuprinde următoarele clase:

-VmUser : interfata intermediara dintre interfata utilizatorului si clasele model cu care are interacțiune utilizatorul.Se vor crea 8 obiecte de tip ICommand,pentru cele 8 comenzi disponibile.Pentru realizarea data binding-ului s-au definit campuri de tipul Property<Type>,astfel nu mai există metode de get si set pentru câmpurile din interfață.

-VmAdmin : interfata intermediara dintre interfata administratorului si clasele model cu care are interacțiune adminul.Se vor crea 4 obiecte de tip ICommand,pentru cele 4 comenzi disponibile.Pentru realizarea data binding-ului s-au definit campuri de tipul Property<Type>,astfel nu mai există metode de get si set pentru câmpurile din interfață.

-VmLogin : interfata intermediara dintre interfata pentru login si clasa model User cu care are interacțiune pagina de login.Se va crea un obiect de tipul ICommand,pentru comanda de logare.Pentru realizarea data binding-ului s-au definit campuri de tipul Property<Type>,astfel nu mai există metode de get si set pentru câmpurile din interfață.

3.3 Baza de date

Users	
PK	userID int NOT NULL
	firstName varchar(50) NOT NULL
	lastName varchar(50) NOT NULL
	username varchar(50) NOT NULL
	password1 varchar(50) NOT NULL
	role varchar(50) NOT NULL

Movies	
PK	movieID int NOT NULL
	title varchar(200) NOT NULL
	type varchar(200) NOT NULL
	category varchar(200) NOT NULL
	year int NOT NULL

Fig 3.3.1 Baza de date moviehouse

Pentru a realiza baza de date am folosit **phpMyAdmin** unde am creat o baza de date intitulată. **moviehouse** care cuprinde două tabele **users** și **movies** . Aceste tabele se află în relație directă cu clasele **user** și **movie** din pachetul **model**. Tabela **users** cuprinde câmpurile :userID, firstName,lastName,username,password1,role. Tabela **movies** cuprinde câmpurile :movieID,title,type,category,year. Între cele 2 tabele nu există relație ,deoarece nu au legatură una cu cealaltă.

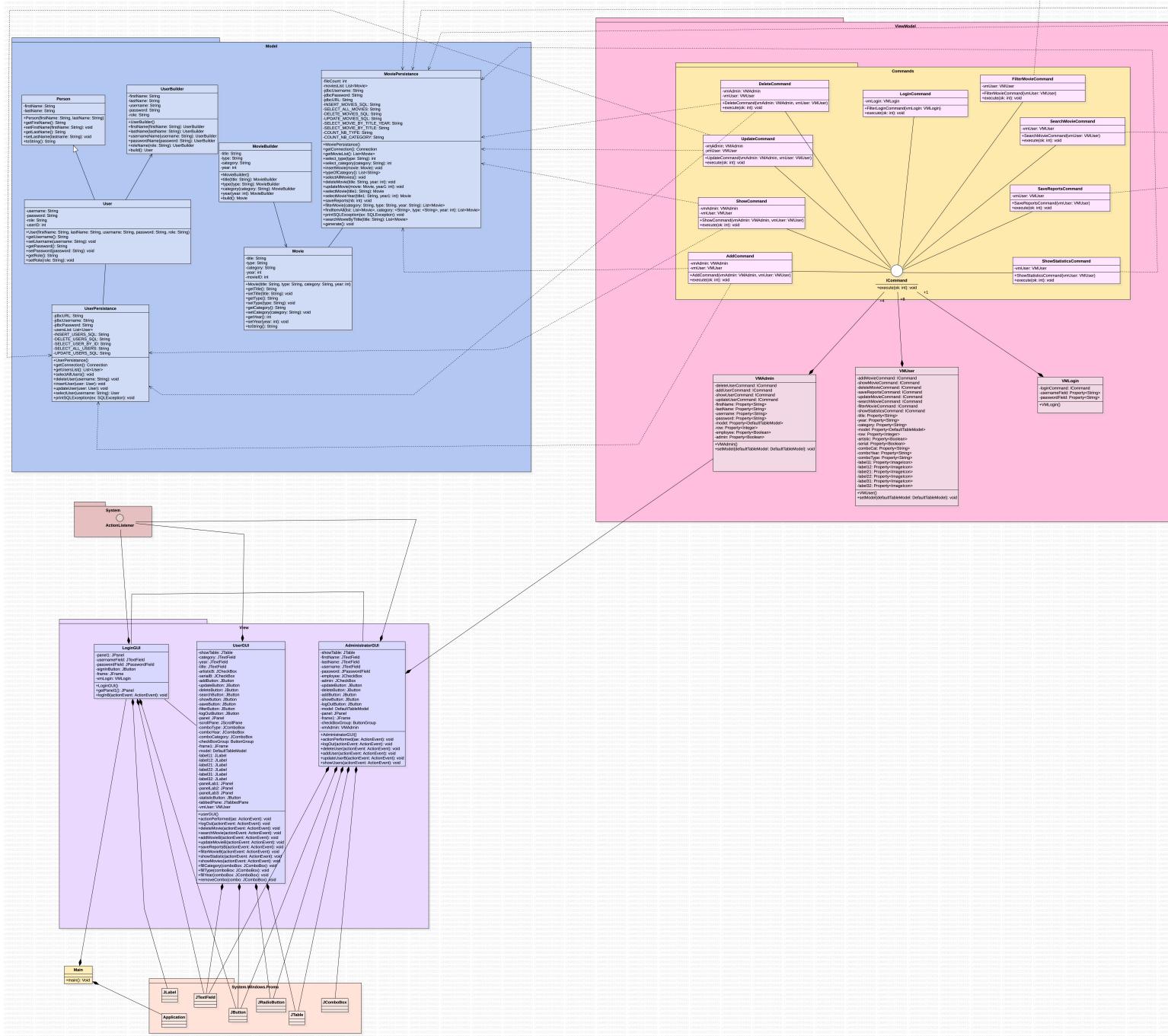


Fig. 3.2.4 Diagrama de clase

4. Descrierea aplicației

Primele filme au revoluționat viața culturală a lumii. Prin îmbunătățiri constante aduse camerei de filmat și tehniciilor de proiecție, filmul a devenit cel mai important mediu de comunicare al secolului al XXI-lea, ajungând să modeleze lumea aşa cum nu a mai făcut-o nicio altă artă până la apariția sa.

Tehnologia ,în ultimi anii , a evoluat drastic,datorită apariției noului "Coronavirus" ,iar marile case de producție filme ,au fost nevoie să își mute activitatea în online ,pentru a nu pierde interacțiunea dintre administratori și angajații.

Suntem actorii unei lumi aflate în continuă ascensiune , tocmai datorită acestui fapt și noi trebuie să ne menținem potrivit așteptărilor . Așadar , pentru a eficientiza munca diferitelor case de producție de filme ,propun următoarea aplicație denumită "Movie House", pentru angajați și administrator, care are următoarele beneficii:

- Vizualizarea listei care conține filmele
- Filtrarea filmelor după anumite criterii: tip film,categorie film,anul realizării
- Operații CRUD în ceea ce privește persistența filmelor
- Căutarea unui film după titlu
- Salvarea rapoarte/liste cu informații despre filme în mai multe formate
- Operații CRUD pentru informațiile legate de utilizatorii aplicației.
- Vizualizarea unor statistici legate de numărul de filme dintr-o anumite categorie,sau de un anumit tip

Codul care stă în spatele funcționalității acestei aplicații este unul destul de simplu și basic, am încercat să scriu cât mai clar și simplu toate operațiile menționate mai sus, am împărțit aplicația în clase specifice, am respectat modelul arhitectural MVP , respectiv modelul SOLID,pentru a fi mai simplu și eficient , a modifica sau a adăuga alte funcționalități , fără să se producă modificări majore liniilor de cod existente , ci doar să adăugăm alte linii.

De asemenea ,am încercat să creez o interfață unică, care să îmbine simplitatea cu stilul meu, și a ieșit ce puteți vedea mai jos,eu zic că am făcut o treabă destul de buna.:)

Pentru filtrările după anumite categorii am folosit expresiile lambda ,doarece mi s-a părut o modalitatea mai ușoara și mai eficientă decât a face numeroase comparații între diferite filme.

Aplicația a fost concepută să fie utilizată de orice categorie de persoană ,atât de una care are cunoștințe în tainele programării cât și de una care acum aude pentru prima dată cuvântul "programare". Prin urmare, folosesc o interfață User-Friendly pentru a permite utilizatorilor să se simtă mai familiarizați cu programul chiar înainte de a-l fi utilizat. Printr-o interfață utilizatorul poate înțelege mai bine cum funcționează programul, poate învăța mai repede modul de utilizare a acestei aplicații. Aceasta interfață cuprinde 3 frame-uri, unul principal pentru logare, iar altele două pentru angajat , respectiv administrator.

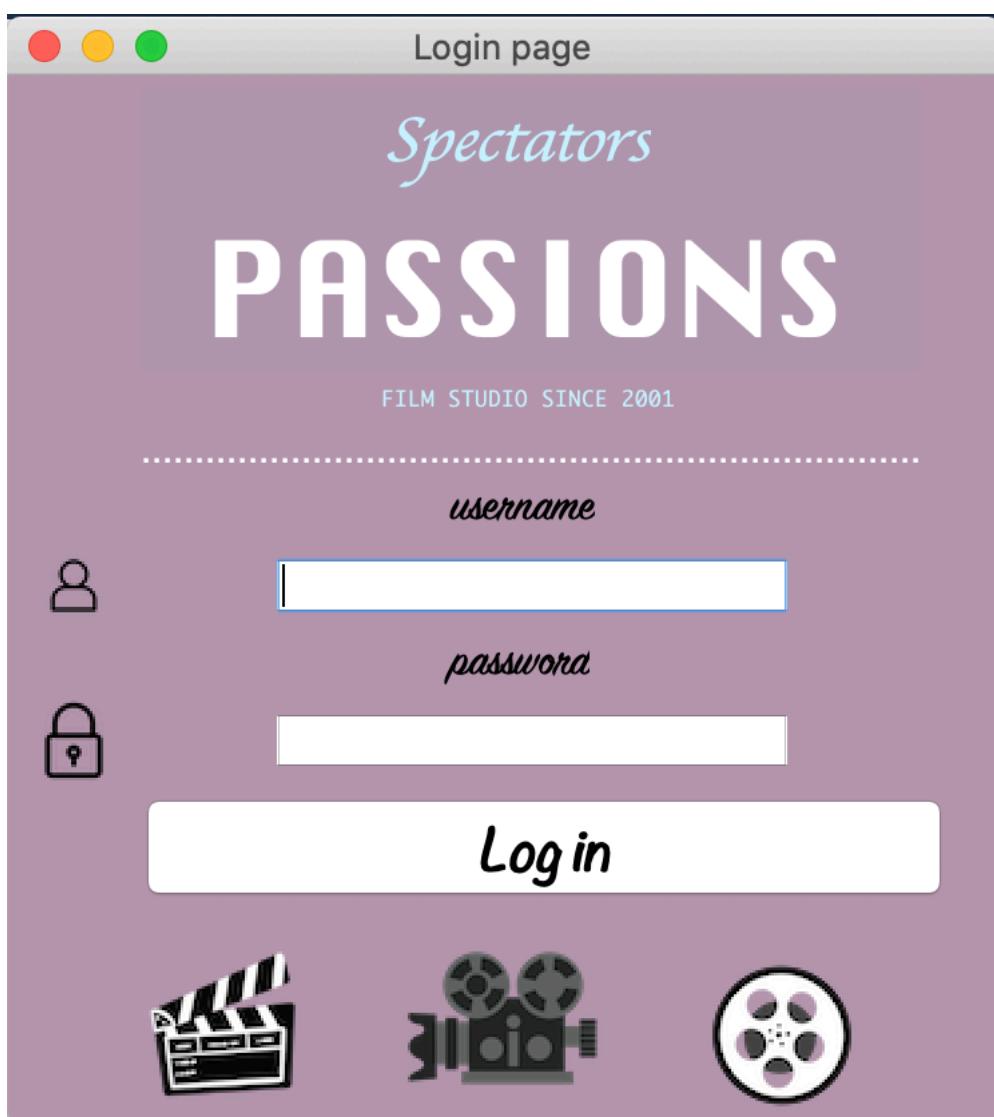


Fig4.1 Interfață pentru pagina de login

Admin page

USERS
-The best-

LogOut






Fist Name

Last Name

Username

Password

Admin **Employee**





Add **Update** **Delete** **Show**

Username:silvia
Password:cotan

Fig4.3 Interfață pentru pagina administratorului

Employee page

Movies
-Produced-

LogOut

Page Radiala Inelara Coloana





Select **Select** **Select**




Title

Category

Year

Artistic **Serial**

Filter **Search**

Username:iuliatimis
Password:timis

Add **Update** **Delete** **Show** **Save** **Statistic**

Fig4.2 Interfață pentru pagina angajatului

