Project -Databases

**Title**: Movie feedback and classification application

**Description**: I propose to create a database containing information about movies, their genres, as well as user feedback for certain films. Based on reviews received from users, the overall rating of the respective film is calculated.

**Technologies**:

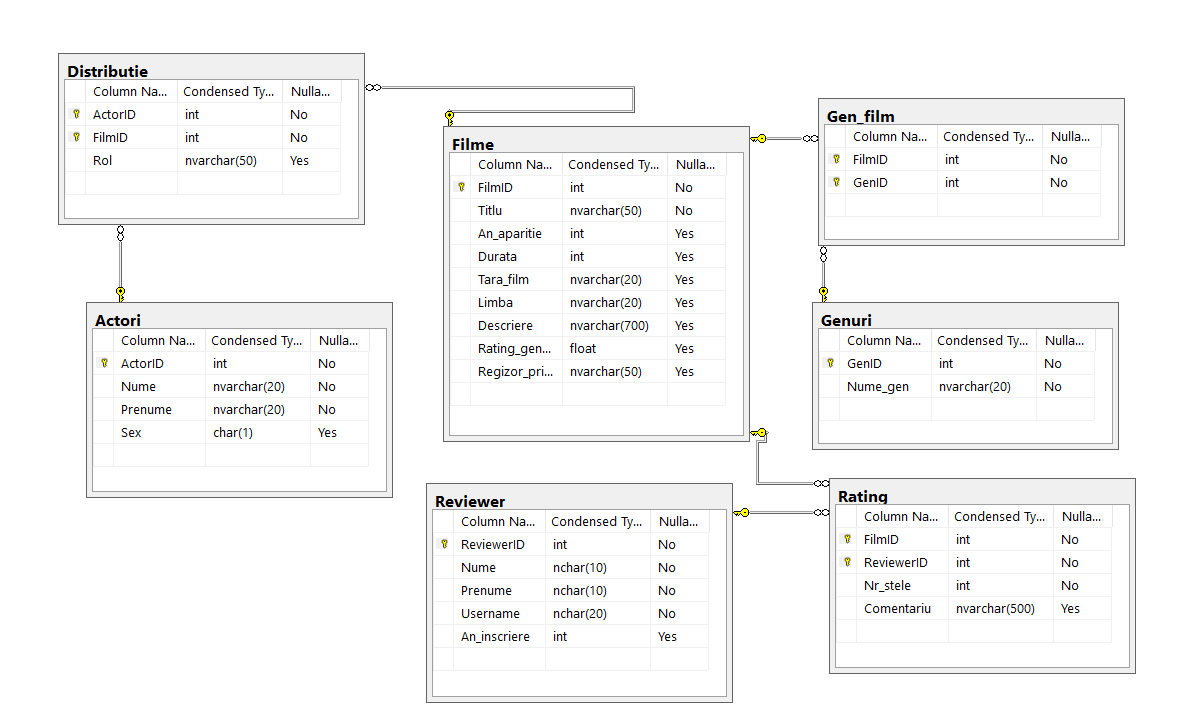
• Microsoft SQL Server 2014 for creating and adding the records to the database

• C# programming language for visual interface creation

**BD Architecture Presentation**

**Tables and relationships between them**

The database contains the tables :Filme, Actori, Distributie, Genuri, Gen\_film, Reviewer and Rating.



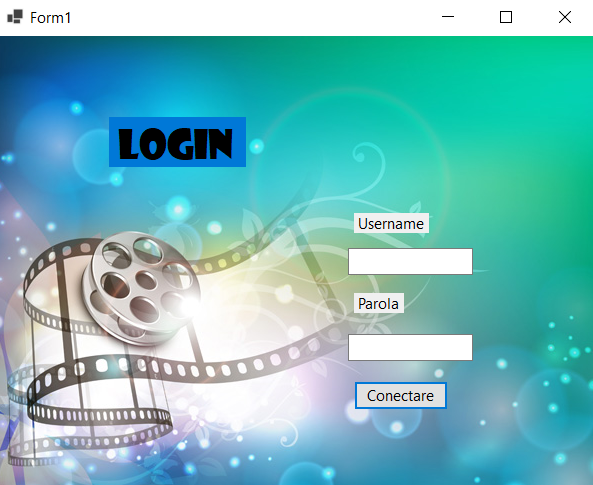
In the image above we can see both the type of data in each table, and the links between them.

**Integrity constraints imposed**

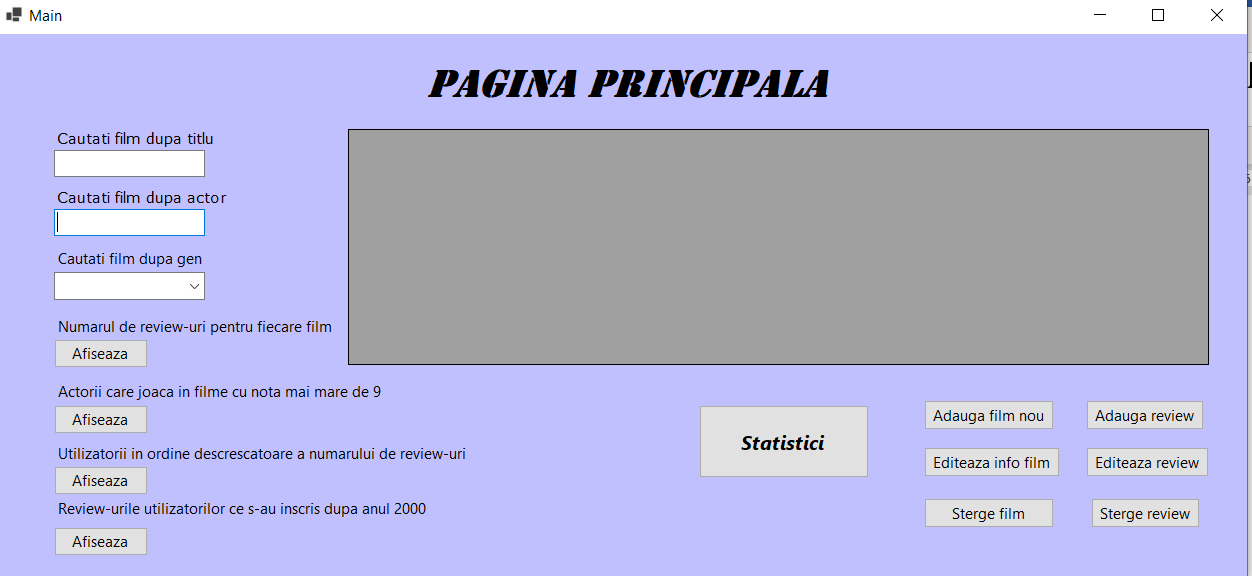
* For the Filme table: FilmID is primary key and uniquely identifies each record, and the Title field cannot be NULL
* Actori Table: ActorID is the primary key, and the name and first name fields cannot have NULL values
* Table Reviewer: ReviewerID is the primary key, and the name, first name and username are mandatory NOT NULL
* For the Genuri table, GenID is the primary key, and the gender name cannot be NULL.
* For the other 3 tables, the primary keys are made up of the composition of the 2 foreign keys.
* For the Rating table, the Nr\_stele cannot be null, because a review necessarily contains a score given to the film and an optional comment.

**Operation of the application**

When the application is run, the login page appears:



If the correct data is not entered, a warning message appears, otherwise this page sends us to the main page of the application.



From this page we have access through buttons to pages intended to add, modify or delete data from the Movies and Rating tables.

The first 3 boxes on the top left represent simple interrogations for searching for data about movies according to certain criteria(variable parameter queries).

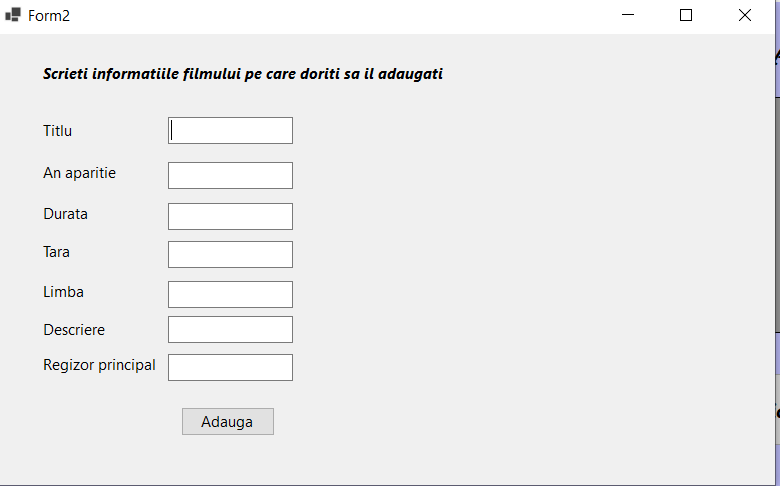
In the first box the search is made through portions of the title of the film, all films containing in the title a certain sequence of characters entered in the text box are displayed.

The second query is to search for movies by the first or last name of the actors who play in the film, and the third displays the films of a certain genre.

The following 4 buttons at the bottom represent 4 more simple queries, without variable parameter. They display directly the required information in the form of a table.

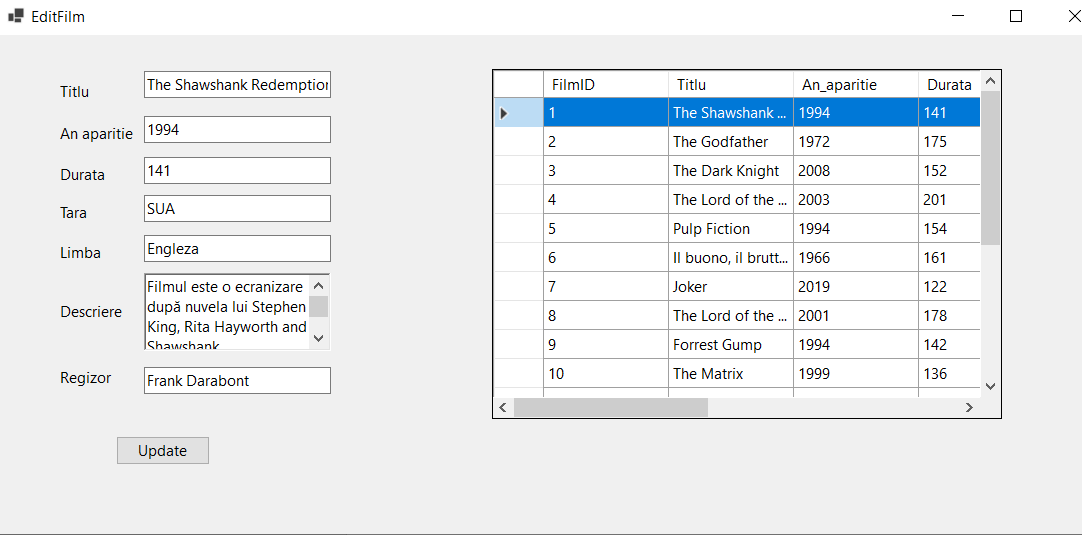
**Add new movie**

In this window you enter the data of the film that you want added to the database.



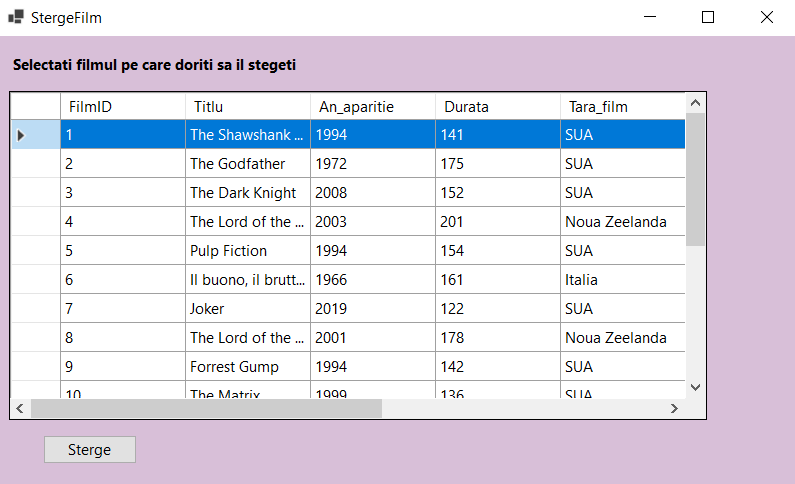
When the add button is pressed, if we enter a movie that already exists in the database, we will receive an error message and we will have to enter another movie. Otherwise, the movie is successfully added to the database.

**Edit movie information**



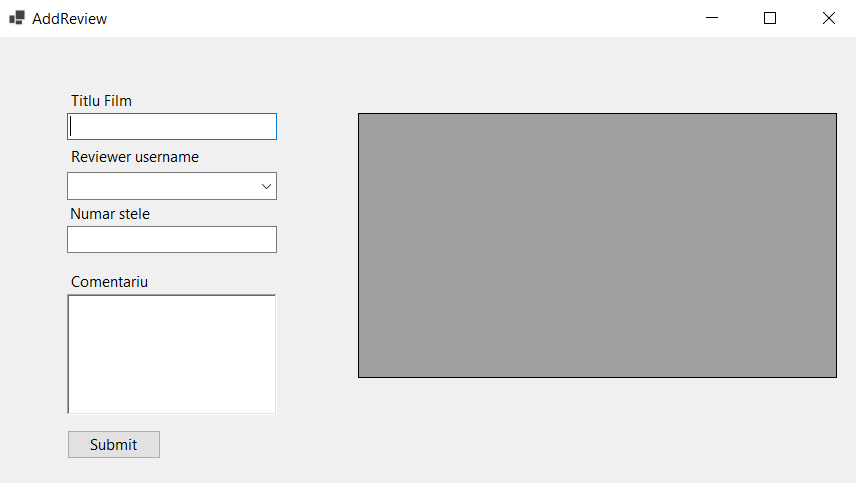
This page is intended to update data about a movie. Select a movie from the list, and in the text boxes appear the current data of the film, can also be modified and updated in the database by pressing the button "Update", after which the data is updated, the list is displayed.

**Delete Movie**



In this window you delete the selected movie, both from the Movies table and from the related tables that contain its primary key as a foreign key.

**Add review**

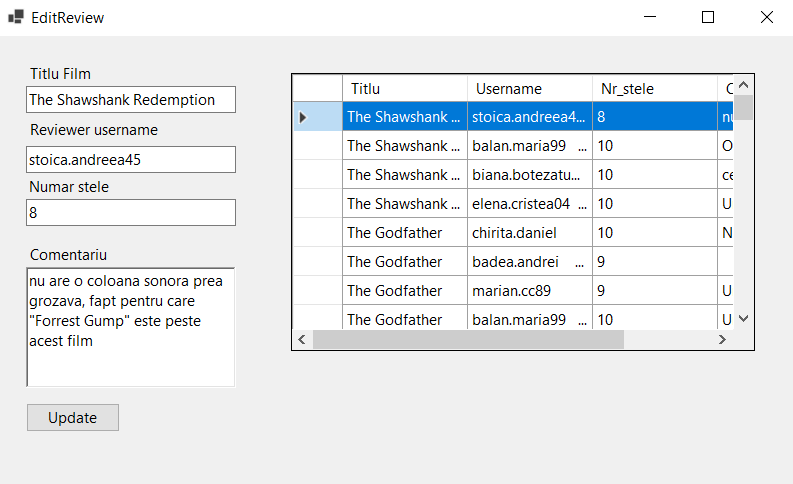


In this window you can add a new review. Write a portion of the title of the film, a list of the films that contain it is displayed, and if we click on one of the movie titles, the title is automatically filled in in the text box.

The next step is to select a username. Only those users who have not yet granted a review for that movie appear in the drop-down list, the others can only modify the existing one.

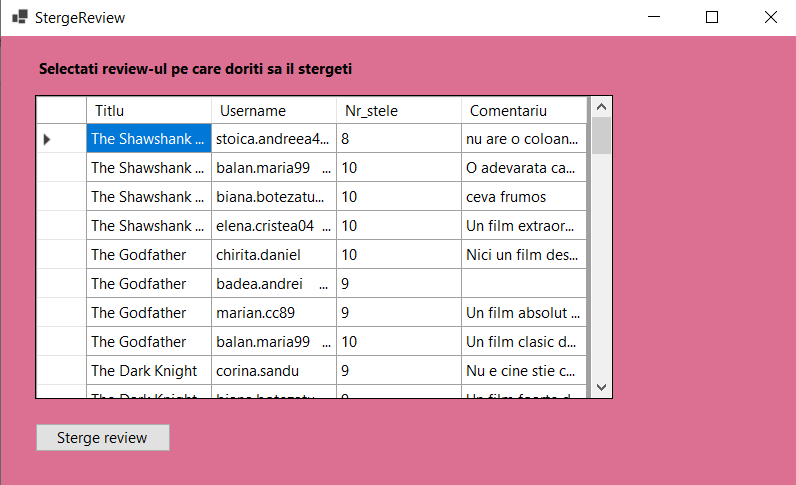
After a review has been introduced and the submit button has been pressed, update the note of the film for which the review was written.

**Edit review**



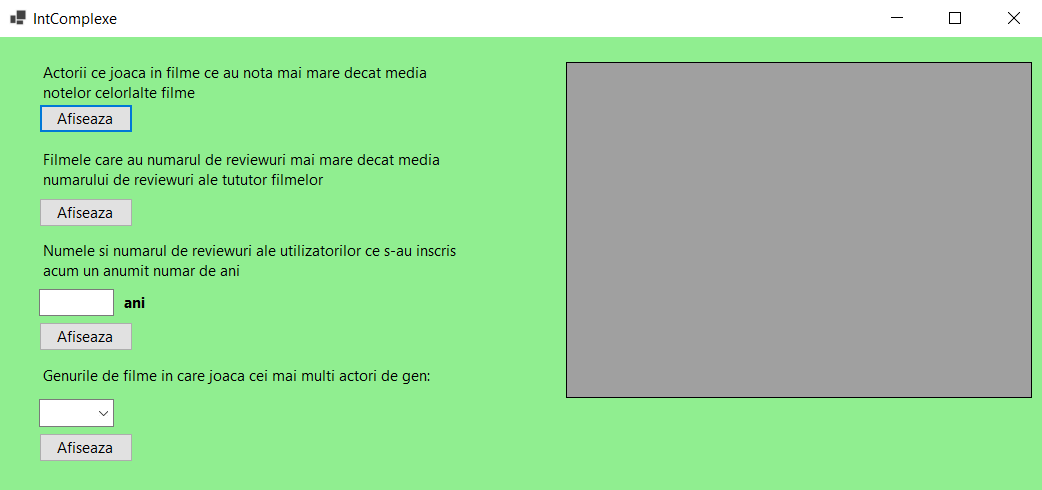
On this page select one of the existing reviews and change either the number of stars, or the comment, or both, after which the final note of the film is also updated, if the number of stars given has been changed.

**Delete review**



This page is intended to delete a selected review. After the deletion, the overall rating of the film for which the review was written is also updated.

**Statistics**



This page is intended for complex database queries. The first two have no variable parameter, while the last two have.

Simple queries

1. ***Search for movies by the names of the actors who play in them***

"Select F.Title, A.First Name, A.Name

from Movies F inner join Distribution D on F.FilmID=D.FilmID

inner join Actors A on A.ActorID=D.ActorID

where A.Name like '%' + textBox1.Text.Trim() + '%'

or A.First name like '%' + textBox1.Text.Trim() + '%'"

1. ***Search for movies by the genre they belong to***

"Select Title, An\_aparitie

from Films F inner join Gen\_film Gf on F.FilmID=Gf.FilmID

inner join Genres G on G.GenID=Gf.GenID

where G.Nume\_gen='" + comboBox1.Text+ "'"

1. ***Number of reviews for each film***

Select F.Title, Count(R.ReviewerID) as Numar\_Reviewuri

from Films F left join Rating R on F.FilmID=R.FilmID

Group by F.Title

1. ***Users in descending order of the number of reviews***

Select U.Name, U.First Name, Count(\*) as NoReviews

from Reviewer U inner join Rating R on U.ReviewerID=R.ReviewID

Group by U.Name, U.First Name

Order by Count(\*) DESC;

1. ***Actors who play in films with a score greater than 9***

Select Distinct A.First Name, A.Name

from Actors A inner join Cast D on A.ActorID=D.ActorID

inner join Movies F on F.FilmID=D.FilmID

Where F.Rating\_general>9;

1. ***Reviews of users who signed up after 2000***

Select U.Username, F.Title, R.Nr\_stele, R.Comment

from Reviewer U inner join Rating R on U.ReviewerID=R.ReviewID

inner join Movies F on F.FilmID=R.FilmID

Where U.An\_inscriere>2000;

Complex interrogations

1. ***Movies that have higher than the average number of reviews of all movies****.*

Select F.Title, NrRev.Nr

From Movies F, (Select FilmID, Count(\*) as No

From Rating

Group by FilmID) as NrRev

Where F.FilmID=NrRev.FilmID AND NrRev.Nr> (Select Avg(NrRev.Nr)

From (Select FilmID, Count(\*) as No From Rating

Group by FilmID) as NrRev )

1. ***Actors who play in films that have a higher rating than the average of the other films.***

Select Distinct A.First Name, A.Name

From Actors A inner join Cast D on A.ActorID=D.ActorID

inner join Movies F on F.FilmID=D.FilmID

Where F.Rating\_general>(Select Avg(Rating\_general)

From Movies

Where FilmID<>F.FilmID)

1. ***Name and number of user reviews that have signed up a certain number of years ago.***

Select U.Name, U.First Name,

(Select Count(\*)

From Rating

Where ReviewerID=U.ReviewerID) as NumberReviews

From Reviewer U

Where Year(Getdate())-U.An\_inscriere>?

1. ***Genres of films in which most actors of the genre (male or female) play***

Select G.Nume\_gen, Count(\*) as noAct

From Genres G inner join Gen\_film Gf on G.GenID=Gf.GenID

inner join Movies F on F.FilmID=Gf.FilmID

inner join Distribution D on F.FilmID=D.FilmID

inner join Actors A on A.ActorID =D.ActorID

Where A.Sex='F'

Group by G.Nume\_gen

Having Count(A.ActorID)=(Select TOP 1 Count(\*)

From Genres G1 inner join Gen\_film Gf1 on G1. GenID=Gf1.GenID

inner join Movies F1 on F1. FilmID=Gf1.FilmID

inner join Distribution D1 on F1. FilmID=D1. FilmID

inner join Actors A1 on A1. ActorID =D1. ActorID

Where A1. Sex='F'

Group by G1. Nume\_gen

Order by Count(\*) DESC)