**Proiect BD**

**Descrierea cerintelor**

Proiectul reprezinta crearea unei baze de date insotita de o interfata vizuala, care impreuna compun a aplicatie pentru feedback si clasificare a filmelor.

**Etape de proiectare**

Aplicatia este proiectata folosind HTML, CSS, PHP, iar baza de date este realizata in Microsoft SQL Server Management Studio. Pentru pornirea serverului am folosit modulul Apache de pe XAMPP Control Panel.

* **Tabelul Utilizatori**:  
  Acest tabel conține informații despre utilizatori. Fiecare utilizator este identificat unic prin coloana UtilizatorID. Sunt stocate numele utilizatorului, adresa de email, data nașterii, țara de origine și parola (ca hash binar). Tabelul include și un indicator pentru rolul de administrator, cu valori posibile 0 sau 1. Data înregistrării utilizatorului este obligatorie. Există constrângeri pentru unicitatea numelui utilizatorului, validarea rolului de administrator și o cheie primară pe UtilizatorID.
* **Tabelul Actori**:  
  Acest tabel conține detalii despre actori. Fiecare actor are un identificator unic, ActorID. Sunt stocate numele și prenumele actorului, data nașterii, naționalitatea, o biografie de maximum 2000 de caractere și un link către poza actorului. Cheia primară a tabelului este ActorID.
* **Tabelul CategoriiFilme**:  
  Acest tabel conține categoriile de filme. Fiecare categorie are un identificator unic, CategorieFilmID. Sunt stocate numele categoriei și o descriere a acesteia (de maximum 500 de caractere). Cheia primară a tabelului este CategorieFilmID.
* **Tabelul Regizori**:  
  Acest tabel conține informații despre regizori. Fiecare regizor are un identificator unic, RegizorID. Sunt stocate numele și prenumele regizorului, data nașterii, naționalitatea, o biografie de maximum 2000 de caractere și un link către poza regizorului. Cheia primară a tabelului este RegizorID.
* **Tabelul Filme**:  
  Acest tabel stochează informații despre filme. Fiecare film are un identificator unic, FilmID. Sunt incluse detalii despre regizor (RegizorID), categoria filmului (CategorieFilmID), titlul, data lansării, durata (în minute), o descriere (de maximum 2000 de caractere), un link către poster și un indicator dacă filmul a câștigat un premiu Oscar (0 sau 1). Cheia primară este FilmID. Există chei externe către Regizori și CategoriiFilme.
* **Tabelul FilmeActori**:  
  Acest tabel asociază actori cu filmele în care au jucat. Fiecare rând conține un FilmID, un ActorID, numele personajului interpretat și tipul rolului (principal sau secundar). Este inclus și un indicator pentru câștigarea unui premiu Oscar pentru acel rol (0 sau 1). Cheia primară este compusă din FilmID și ActorID. Există chei externe către tabelele Filme și Actori.
* **Tabelul UtilizatoriFilme**:  
  Acest tabel stochează filmele asociate utilizatorilor. Fiecare rând conține un UtilizatorID și un FilmID. Este inclus un indicator dacă filmul este favorit (0 sau 1), data adăugării și statusul vizionării (0 = nevizionat, 1 = vizionat). Cheia primară este compusă din UtilizatorID și FilmID. Există chei externe către tabelele Utilizatori și Filme.
* **Tabelul Reviewuri**:  
  Acest tabel conține recenziile pentru filme. Fiecare recenzie are un identificator unic, ReviewID. Sunt stocate ID-ul utilizatorului care a scris recenzia (UtilizatorID), ID-ul filmului la care se referă (FilmID), conținutul recenziei, un rating (cu o precizie de o zecimală) și data postării recenziei. Cheia primară este ReviewID. Există chei externe către tabelele Utilizatori și Filme.
* **Tabelul Comentarii**:  
  Acest tabel stochează comentarii asociate recenziilor. Fiecare comentariu are un identificator unic, ComentariuID. Sunt stocate ID-ul utilizatorului care a postat comentariul (UtilizatorID), ID-ul recenziei comentate (ReviewID), conținutul comentariului și data postării. Cheia primară este ComentariuID. Există chei externe către tabelele Utilizatori și Reviewuri.

**Structura tabelelor**:

O imagine care conține text, captură de ecran, diagramă, Paralel

Descriere generată automat

**Relatii intre tabele:**

* Filme-regizori:
  + Un film are un regizor. 1:1
  + Un regizor poate produce mai multe filme. 1:N
  + Rezulta 1:N
* Filme-actori
  + Un film poate avea mai multi actori. 1:N
  + Un actor poate juca in mai multe filme. 1:N
  + Rezulta N:N (Rezolvat prin tabelul FilmeActori)
* Utilizatori- FIlme
  + Un utilizator poate viziona mai multe filme. 1:N
  + Un film poate fi vizionat de mai multi utilizatori. 1:N
  + Rezulta N:N (Rezolvat prin tabelul UtilizatoriFilme)
* Utilizatori-Reviewuri
  + Un Utilizator poate scrie mai multe reviewri. 1:N
  + Un review apartine unui singur utilizator. 1:1
  + Rezulta 1:N
* Filme-Reviewuri
  + Un film poate avea mai multe reviewuri. 1:N
  + Un review este despre un singur film. 1:1
  + Rezulta 1:N
* Utilizatori-Comentarii
  + Un utilizator poate posta mai multe comentarii. 1:N
  + Comentariul apartine unui singur utilizator. 1:1
  + Rezulta 1:N
* Reviewuri-Comentarii
  + Un review poate avea mai multe comentarii 1:N
  + Comentariul se adreseaza unui singur review. 1:1
  + Rezulta 1:N
* Filme-CategoriiFilme
  + Un film apartine unei categorii de film. 1:1
  + O categorie de film cuprinde mai multe filme. 1:N
  + Rezulta 1:N

**Crearea tabelelor:**

CREATE DATABASE BDFilme COLLATE Latin1\_General\_100\_CI\_AS\_SC

CREATE TABLE Utilizatori

(

UtilizatorID int identity(1,1) NOT NULL,

Nume nvarchar(50) NOT NULL,

Email nvarchar(50) NOT NULL,

DataNasterii date,

Tara nvarchar(25),

Parola varbinary(32) NOT NULL,

RolAdmin binary(1) DEFAULT 0,

DataInregistrarii smalldatetime NOT NULL,

CONSTRAINT PK\_Utilizatori PRIMARY KEY (UtilizatorID),

CONSTRAINT UNQ\_Angajati\_Nume UNIQUE (Nume),

CONSTRAINT CHK\_Utilizator\_RolAdmin CHECK (RolAdmin=0 OR RolAdmin=1)

)

CREATE TABLE Actori

(

ActorID int identity(1,1) NOT NULL,

Nume nvarchar(50) NOT NULL,

Prenume nvarchar(50) NOT NULL,

DataNasterii date,

Nationalitate nvarchar(20),

Biografie nvarchar(2000),

Poza nvarchar(255), -- link poza

CONSTRAINT PK\_Actori PRIMARY KEY (ActorID)

)

CREATE TABLE CategoriiFilme

(

CategorieFilmID int identity(1,1) NOT NULL,

Nume nvarchar(50) NOT NULL,

DescriereCategorie nvarchar(500),

CONSTRAINT PK\_CatergoriiFilme PRIMARY KEY (CategorieFilmID)

)

CREATE TABLE Regizori

(

RegizorID int identity(1,1) NOT NULL,

Nume nvarchar(50) NOT NULL,

Prenume nvarchar(50) NOT NULL,

DataNasterii date,

Nationalitate nvarchar(20),

Biografie nvarchar(2000),

Poza nvarchar(255), -- link poza

CONSTRAINT PK\_Regizori PRIMARY KEY (RegizorID)

)

CREATE TABLE Filme

(

FilmID int identity(1,1) NOT NULL,

RegizorID int NOT NULL,

CategorieFilmID int NOT NULL,

Titlu nvarchar(50) NOT NULL,

DataLansarii date NOT NULL,

Durata int NOT NULL,

DescriereFilm nvarchar(2000),

Poster nvarchar(255), -- link poza

PremiuOscar binary(1) DEFAULT 0,

CONSTRAINT PK\_Filme PRIMARY KEY (FilmID),

CONSTRAINT CHK\_Filme\_PremiuOscar CHECK (PremiuOscar=0 OR PremiuOscar=1),

CONSTRAINT FK\_Filme\_Regizori FOREIGN KEY (RegizorID) REFERENCES Regizori(RegizorID),

CONSTRAINT FK\_Filme\_CategoriiFilme FOREIGN KEY (CategorieFilmID) REFERENCES CategoriiFilme(CategorieFilmID)

)

CREATE TABLE FilmeActori

(

FilmID int NOT NULL,

ActorID int NOT NULL,

Personaj nvarchar(50) NOT NULL,

TipRol char(10) NOT NULL, --principal, secundar

PremiuOscar binary(1) DEFAULT 0,

CONSTRAINT PK\_FilmeActori PRIMARY KEY (FilmID,ActorID),

CONSTRAINT CHK\_FilmeActori\_PremiuOscar CHECK (PremiuOscar=0 OR PremiuOscar=1),

CONSTRAINT FK\_FilmeActori\_Filme FOREIGN KEY (FilmID) REFERENCES Filme(FilmID),

CONSTRAINT FK\_FilmeActori\_Actori FOREIGN KEY (ActorID) REFERENCES Actori(ActorID)

)

CREATE TABLE UtilizatoriFilme

(

UtilizatorID int NOT NULL,

FilmID int NOT NULL,

Favorit binary(1) DEFAULT 0,

DataAdaugarii date NOT NULL,

StatusVizionare binary(1) DEFAULT 0,

CONSTRAINT PK\_UtilizatoriFilme PRIMARY KEY (UtilizatorID,FilmID),

CONSTRAINT CHK\_UtilizatoriFilme\_Favorit CHECK (Favorit=0 OR Favorit=1),

CONSTRAINT CHK\_UtilizatoriFilme\_StatusVizionare CHECK (StatusVizionare=0 OR StatusVizionare=1),

CONSTRAINT FK\_UtilizatoriFilme\_Utilizatori FOREIGN KEY (UtilizatorID) REFERENCES Utilizatori(UtilizatorID),

CONSTRAINT FK\_UtilizatoriFilme\_Filme FOREIGN KEY (FilmID) REFERENCES Filme(FilmID)

)

CREATE TABLE Reviewuri

(

ReviewID int identity(1,1) NOT NULL,

UtilizatorID int NOT NULL,

FilmID int NOT NULL,

ContinutReview nvarchar(2000) NOT NULL,

Rating DECIMAL(3,1) NOT NULL,

DataPostarii smalldatetime NOT NULL,

CONSTRAINT PK\_Reviuwuri PRIMARY KEY (ReviewID),

CONSTRAINT FK\_Reviewuri\_Utilizatori FOREIGN KEY (UtilizatorID) REFERENCES Utilizatori(UtilizatorID),

CONSTRAINT FK\_Reviewuri\_Filme FOREIGN KEY (FilmID) REFERENCES Filme(FilmID)

)

CREATE TABLE Comentarii

(

ComentariuID int identity(1,1) NOT NULL,

UtilizatorID int NOT NULL,

ReviewID int NOT NULL,

ContinutComentariu nvarchar(2000) NOT NULL,

DataPostarii smalldatetime NOT NULL,

CONSTRAINT PK\_Comentarii PRIMARY KEY (ComentariuID),

CONSTRAINT FK\_Comentarii\_Utilizatori FOREIGN KEY (UtilizatorID) REFERENCES Utilizatori(UtilizatorID),

CONSTRAINT FK\_Comentarii\_Reviewuri FOREIGN KEY (ReviewID) REFERENCES Reviewuri(ReviewID)

)

**Conectarea la baza de date:**

$servername = "DESKTOP-OEU6GV8\SQLEXPRESS";

$database = "BDFilme";

$uid="";

$password="";

$connection=[

"Database" => $database,

"Uid"=>$uid,

"PWD"=>$password,

"Encrypt" => true,

"TrustServerCertificate" => true

];

$conn = sqlsrv\_connect($servername,$connection);

if(!$conn){

    die(print\_r(sqlsrv\_errors(),true));

}

**Insert in baza de date:**

Tabel utilizatori (Sign-Up):

$hashedPassword = hash('sha256', $parola, true); // true returnează raw binary data

// Pregătire interogare SQL

$sql = "INSERT INTO Utilizatori (Nume, Email, DataNasterii, Tara, Parola, DataInregistrarii)

        VALUES (?, ?, ?, ?, CONVERT(VARBINARY(32), ?), GETDATE())";

$params = array($nume, $email, $dataNasterii, $tara, $hashedPassword);

// Executare interogare

$stmt = sqlsrv\_query($conn, $sql, $params);

if ($stmt === false) {

    die(print\_r(sqlsrv\_errors(), true));

    header("Location: form.php?error=Eroare.") ;

} else {

    echo "Cont creat cu succes!";

    header("Location: home.php");

Tabel UtilizatoriFilme ( Watchlist):

$utilizatorID = $row['UtilizatorID'];

// Preluarea filmului introdus de utilizator

$film = isset($\_POST['film']) ? trim(htmlspecialchars($\_POST['film'])) : '';

if ($film) {

    // Cod pentru căutare și inserare a filmului

    $sql = "SELECT FilmID FROM Filme WHERE Titlu = ?";

    $params = array($film);

    $stmtAdd = sqlsrv\_query($conn, $sql, $params);

    if ($stmtAdd === false) {

        die(print\_r(sqlsrv\_errors(), true));

    }

    $row = sqlsrv\_fetch\_array($stmtAdd, SQLSRV\_FETCH\_ASSOC);

    if (!$row) {

        echo "Eroare: filmul nu a fost găsit.";

    } else {

        $filmID = $row['FilmID'];

        // Verifică dacă filmul este deja în lista utilizatorului

        $sql = "SELECT \* FROM UtilizatoriFilme WHERE UtilizatorID = ? AND FilmID = ?";

        $params = array($utilizatorID, $filmID);

        $stmtAdd = sqlsrv\_query($conn, $sql, $params);

        if ($stmtAdd === false) {

            die(print\_r(sqlsrv\_errors(), true));

        }

        if (sqlsrv\_fetch\_array($stmtAdd, SQLSRV\_FETCH\_ASSOC)) {

            echo "Film deja adăugat în lista ta!";

        } else {

            // Inserează filmul în lista de vizionare

            $sql = "INSERT INTO UtilizatoriFilme (UtilizatorID, FilmID, DataAdaugarii)

                    VALUES (?, ?, GETDATE())";

            $params = array($utilizatorID, $filmID);

            $stmtAdd = sqlsrv\_query($conn, $sql, $params);

            if ($stmtAdd === false) {

                die(print\_r(sqlsrv\_errors(), true));

            }

            echo "Film adăugat cu succes în lista ta!";

        }

    }

}

Tabel filme (necesita Admin):

$sql = "INSERT INTO Filme (RegizorID, CategorieFilmID, Titlu, DataLansarii, Durata, DescriereFilm, Poster, PremiuOscar)

            VALUES (?, ?, ?, ?, ?, ?, ?, ?)";

    $params = array($regizorID, $categorieFilmID, $titlu, $dataLansarii, $durata, $descriereFilm, $poster, $premiuOscar);

    $stmt = sqlsrv\_query($conn, $sql, $params);

    if ($stmt === false) {

        die(print\_r(sqlsrv\_errors(), true));  // Dacă apare o eroare la inserare

    } else {

        echo "Filmul a fost adăugat cu succes!";

    }

**Update in baza de date:**

In tabelul de UtilizatoriFilme:

<?php

$film = isset($\_POST['filmUp']) ? trim(htmlspecialchars($\_POST['filmUp'])) : '';

$status = isset($\_POST['status']) ? $\_POST['status'] : null;

$favorit= isset($\_POST['preferat'])? $\_POST['preferat'] : null;

if ($film) {

    // Cod pentru căutare și updatare a filmului

    $sql = "SELECT FilmID FROM Filme WHERE Titlu = ?";

    $params = array($film);

    $stmtUp = sqlsrv\_query($conn, $sql, $params);

    if ($stmtUp === false) {

        die(print\_r(sqlsrv\_errors(), true));

    }

    $row = sqlsrv\_fetch\_array($stmtUp, SQLSRV\_FETCH\_ASSOC);

    if (!$row) {

        echo "Eroare: filmul nu a fost găsit.";

    } else {

        $filmID = $row['FilmID'];

            // Updateaza

            $sql = "UPDATE UtilizatoriFilme SET StatusVizionare = ?, Favorit = ? WHERE UtilizatorID = ? AND FilmID = ?";

            $binaryStatus = ($status == "1") ? 0x01 : 0x00;

            $binaryFavorite= ($favorit == "1") ? 0x01 : 0x00;

            $params = array($binaryStatus,$binaryFavorite,$utilizatorID, $filmID);

            $stmtUp = sqlsrv\_query($conn, $sql, $params);

            if ($stmtUp === false) {

                die(print\_r(sqlsrv\_errors(), true));

            }

            echo "Film updatat cu succes din lista ta!";

        }

    }

?>

In tabelul cu Filme (necesita Admin):

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

    // Preia numele filmului din formular

    $titlu = $\_POST['titlu'];

    // Căutăm filmul în baza de date folosind titlul

    $sqlFilm = "SELECT FilmID, Titlu, RegizorID, CategorieFilmID, DataLansarii, Durata, DescriereFilm, Poster, PremiuOscar

                FROM Filme WHERE Titlu = ?";

    $stmtFilm = sqlsrv\_query($conn, $sqlFilm, array($titlu));

    if ($stmtFilm === false || sqlsrv\_has\_rows($stmtFilm) === false) {

        die("Eroare: Filmul nu a fost găsit.");

    }

    // Dacă găsim filmul, îl preluăm

    $filmData = sqlsrv\_fetch\_array($stmtFilm, SQLSRV\_FETCH\_ASSOC);

    $filmID = $filmData['FilmID'];  // Salvează ID-ul filmului pentru actualizare

}

if ($\_SERVER['REQUEST\_METHOD'] == 'POST' && isset($filmID)) {

    // Preia datele din formular pentru actualizare

    $numeRegizor = $\_POST['numeRegizor'];

    $numeCategorie = $\_POST['numeCategorie'];

    $dataLansarii = $\_POST['dataLansarii'];

    $durata = $\_POST['durata'];

    $descriereFilm = $\_POST['descriereFilm'];

    $poster = $\_POST['poster'];

    $premiuOscar = isset($\_POST['premiuOscar']) ? 1 : 0; // Verifică dacă premiul Oscar este selectat

    // Căutăm ID-ul regizorului în baza de date

    $sqlRegizor = "SELECT RegizorID FROM Regizori WHERE Nume = ?";

    $stmtRegizor = sqlsrv\_query($conn, $sqlRegizor, array($numeRegizor));

    if ($stmtRegizor === false || sqlsrv\_has\_rows($stmtRegizor) === false) {

        die("Eroare: Regizorul nu a fost găsit.");

    }

    $rowRegizor = sqlsrv\_fetch\_array($stmtRegizor, SQLSRV\_FETCH\_ASSOC);

    $regizorID = $rowRegizor['RegizorID'];

    // Căutăm ID-ul categoriei în baza de date

    $sqlCategorie = "SELECT CategorieFilmID FROM CategoriiFilme WHERE Nume = ?";

    $stmtCategorie = sqlsrv\_query($conn, $sqlCategorie, array($numeCategorie));

    if ($stmtCategorie === false || sqlsrv\_has\_rows($stmtCategorie) === false) {

        die("Eroare: Categoria nu a fost găsită.");

    }

    $rowCategorie = sqlsrv\_fetch\_array($stmtCategorie, SQLSRV\_FETCH\_ASSOC);

    $categorieFilmID = $rowCategorie['CategorieFilmID'];

    // Actualizăm filmul în baza de date

    $sql = "UPDATE Filme

            SET RegizorID = ?, CategorieFilmID = ?, DataLansarii = ?, Durata = ?, DescriereFilm = ?, Poster = ?, PremiuOscar = ?

            WHERE FilmID = ?";

    $params = array($regizorID, $categorieFilmID, $dataLansarii, $durata, $descriereFilm, $poster, $premiuOscar, $filmID);

    $stmt = sqlsrv\_query($conn, $sql, $params);

    if ($stmt === false) {

        die(print\_r(sqlsrv\_errors(), true));  // Dacă apare o eroare la actualizare

    } else {

        echo "Filmul a fost actualizat cu succes!";

    }

}

**Delete in baza de date:**

Delete in UtilizatoriFilme:

   <?php $username = $\_SESSION['username'];

$sql = "SELECT UtilizatorID FROM Utilizatori WHERE Nume = ?";

$params = array($username);

$stmt = sqlsrv\_query($conn, $sql, $params);

if ($stmt === false) {

    die(print\_r(sqlsrv\_errors(), true));

}

$row = sqlsrv\_fetch\_array($stmt, SQLSRV\_FETCH\_ASSOC);

if (!$row) {

    echo "Eroare: utilizatorul nu a fost găsit.";

    exit;

}

$utilizatorID = $row['UtilizatorID'];

// Preluarea filmului introdus de utilizator

$film = isset($\_POST['filmSters']) ? trim(htmlspecialchars($\_POST['filmSters'])) : '';

if ($film) {

    // Cod pentru căutare și stergere a filmului

    $sql = "SELECT FilmID FROM Filme WHERE Titlu = ?";

    $params = array($film);

    $stmtDel = sqlsrv\_query($conn, $sql, $params);

    if ($stmtDel === false) {

        die(print\_r(sqlsrv\_errors(), true));

    }

    $row = sqlsrv\_fetch\_array($stmtDel, SQLSRV\_FETCH\_ASSOC);

    if (!$row) {

        echo "Eroare: filmul nu a fost găsit.";

    } else {

        $filmID = $row['FilmID'];

            // Sterge

            $sql = "DELETE FROM UtilizatoriFilme WHERE UtilizatorID = ? AND FilmID = ?";

            $params = array($utilizatorID, $filmID);

            $stmtDel = sqlsrv\_query($conn, $sql, $params);

            if ($stmtDel === false) {

                die(print\_r(sqlsrv\_errors(), true));

            }

            echo "Film sters cu succes din lista ta!";

        }

    }

?>

Delete in tabelul Filme (necesita Admin):

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

    // Preia titlul filmului din formular

    $titlu = $\_POST['titlu'];

    // Căutăm filmul în baza de date folosind titlul

    // Ștergem filmul din baza de date

    $sqlDelete = "DELETE FROM Filme WHERE Titlu = ?";

    $stmtDelete = sqlsrv\_query($conn, $sqlDelete, array($titlu));

    if ($stmtDelete === false) {

        die(print\_r(sqlsrv\_errors(), true));  // Dacă apare o eroare la ștergere

    } else {

        echo "Filmul a fost șters cu succes!";

    }

}

**Interogari simple:**

Selecteaza lista de filme a unui utilizator:

$sql = "SELECT F.Titlu AS Titlu,

               UF.Favorit AS Favorit,

               UF.DataAdaugarii AS DataAdaugarii,

               UF.StatusVizionare AS StatusVizionare,

               R.Rating AS Rating

        FROM Utilizatori U

        JOIN UtilizatoriFilme UF ON U.UtilizatorID = UF.UtilizatorID

        JOIN Filme F ON UF.FilmID = F.FilmID

        LEFT JOIN Reviewuri R ON F.FilmID = R.FilmID AND U.UtilizatorID = R.UtilizatorID

        WHERE U.Nume = ?";

Selecteaza filmele alaturi de categoriile lor, cu posibilitatea de filtrare dupa categorie:

if ($categorie) {

    $sql = "SELECT F.Titlu AS Titlu , F.DataLansarii AS DataLansarii, F.Durata AS Durata, CF.Nume AS CategorieFilm, F.DescriereFilm AS DescriereFilm, F.Poster AS Poster

        FROM Filme F JOIN CategoriiFilme CF ON (F.CategorieFilmID=CF.CategorieFilmID)

         WHERE CF.Nume = ?";

    $params[] = $categorie;

}

Selecteaza regizorii alaturi de nr de filme realizate, posibilitate de filtrare dupa nume:

if ($regizor) {

    $sql = "SELECT R.Nume AS Nume, R.Prenume AS Prenume, R.DataNasterii AS DataNasterii, R.Nationalitate AS Nationalitate, R.Biografie AS Biografie, R.Poza AS Poza, COUNT(F.FilmID) AS NrFilme

 FROM Regizori R JOIN Filme F ON (R.RegizorID=F.RegizorID)

         WHERE Nume LIKE ?

          GROUP BY Nume, Prenume, DataNasterii, Nationalitate,

                 Biografie, Poza";

    $params[] = $regizor . '%';

}

Selecteaza actorii alaturi de nr de filme jucate, posibilitate de filtrare dupa nume:

if ($actor) {

    $sql = "SELECT A.Nume AS Nume, A.Prenume AS Prenume, A.DataNasterii AS DataNasterii, A.Nationalitate AS Nationalitate, A.Biografie AS Biografie, A.Poza AS Poza, COUNT(FA.ActorID) AS NrFilme

 FROM Actori A JOIN FilmeActori FA ON (A.ActorID=FA.ActorID)

 WHERE Nume LIKE ?

 GROUP BY Nume, Prenume, DataNasterii, Nationalitate,

                 Biografie, Poza";

    $params[] = $actor . '%';

}

Selecteaza reviewurile alaturi de filmul corespunzator, posibilitate de filtrare dupa numele filmului:

if ($film ) {

    $sql = "SELECT U.Nume AS Nume, F.Titlu AS Titlu, R.ContinutReview AS Review, R.Rating AS Rating, R.DataPostarii AS DataPostarii

FROM Reviewuri R JOIN Utilizatori U ON (R.UtilizatorID=U.UtilizatorID)

                JOIN Filme F ON (F.FilmID=R.FilmID)

                WHERE F.Titlu LIKE ? ";

    $params[] = '%'.$film.'%';

}

Selectarea comentariilor unui user:

if ($user ) {

    $sql = "SELECT U.Nume AS Nume, C.ContinutComentariu AS ContinutComentariu, C.DataPostarii AS DataPostarii

FROM Utilizatori U JOIN Comentarii C ON (U.UtilizatorID=C.UtilizatorID)

WHERE Nume LIKE ?";

    $params[] = '%'.$userAles.'%';

}

**Interogari complexe:**

Top filme în funcție de rating în ultimele 6 luni:

$sql= "SELECT TOP 3 Titlu, RatingMediu

FROM (

    SELECT F.Titlu, AVG(R.Rating) AS RatingMediu

    FROM Filme F

    JOIN Reviewuri R ON F.FilmID = R.FilmID

    WHERE R.DataPostarii > DATEADD(MONTH, -6, GETDATE())

    GROUP BY F.Titlu

) AS Subquery

WHERE RatingMediu > 8.0

ORDER BY RatingMediu DESC;";

Filmele vizionate de mai mult de un anumit număr de utilizatori, se selecteaza nr de utilizatori.

$sql2= "SELECT Titlu, Durata, DataLansarii,Poster

FROM Filme F

WHERE F.FilmID IN (

    SELECT UF.FilmID

    FROM UtilizatoriFilme UF

    GROUP BY UF.FilmID

    HAVING COUNT(UF.UtilizatorID) > ?

)

ORDER BY Titlu";

$params[]=isset($\_GET['numar']  ) ? $\_GET['numar']   : 1;

Filmele care au fost vizionate și care au primit recenzii pozitive de la utilizatori, cu alegerea valorii de prag a recenziilor pozitive:

$sql3= "SELECT F.Titlu, Durata, DataLansarii, AVG(R.Rating) AS MedieRating,Poster

FROM Filme F JOIN Reviewuri R ON (F.FilmID=R.FilmID)

WHERE F.FilmID IN (

    SELECT UF.FilmID

    FROM UtilizatoriFilme UF

    WHERE UF.StatusVizionare = 1

)

AND F.FilmID IN (

    SELECT R.FilmID

    FROM Reviewuri R

    GROUP BY R.FilmID

    HAVING AVG(R.Rating) > ?

)

GROUP BY F.Titlu, Durata, DataLansarii, Poster

ORDER BY AVG(R.Rating) DESC;";

$params[]=isset($\_GET['rating\_ales']  ) ? $\_GET['rating\_ales']  : 5;

Actorii care au jucat in cele mai multe filme de o anumita categorie, cu selectarea categoriei:

$sql4="SELECT TOP 5 A.Prenume, A.Nume, COUNT(FA.FilmID) AS NumarFilme, A.Poza

FROM Actori A

JOIN FilmeActori FA ON A.ActorID = FA.ActorID

WHERE FA.FilmID IN (

    SELECT F.FilmID

    FROM Filme F

    WHERE F.CategorieFilmID = (SELECT CategorieFilmID FROM CategoriiFilme WHERE Nume = ?)

)

GROUP BY A.Nume,A.Prenume,A.Poza

ORDER BY COUNT(FA.FilmID) DESC ";

$categorie = isset($\_GET['categorie']) ? $\_GET['categorie'] : '';

$params[]=$categorie;

**Aplicatia**

Aplicatia consta intr-o interfata grafica pentru feedback si clasificare de filme, care cuprinde si o parte administrativa, adminul avand capacitatea de a adauga, modifica si sterge informatii din tabelele din baza de date. Pentru admin este necesar un rol special.

De asemenea, interogările simple și complexe realizate pe baza de date sunt integrate în interfața grafică și pot fi accesate din meniu, de pe diferite pagini, prin intermediul butoanelor, introducând un cuvânt cheie sau făcând clic pe un text sugestiv. Fiecare pagină include un scurt text care indică sugestiv conținutul.

Proiectul, implementat în PHP și SQL Server, respectă recomandările specificate în cerințele proiectului și se bazează, de asemenea, pe documentație suplimentară disponibilă online.