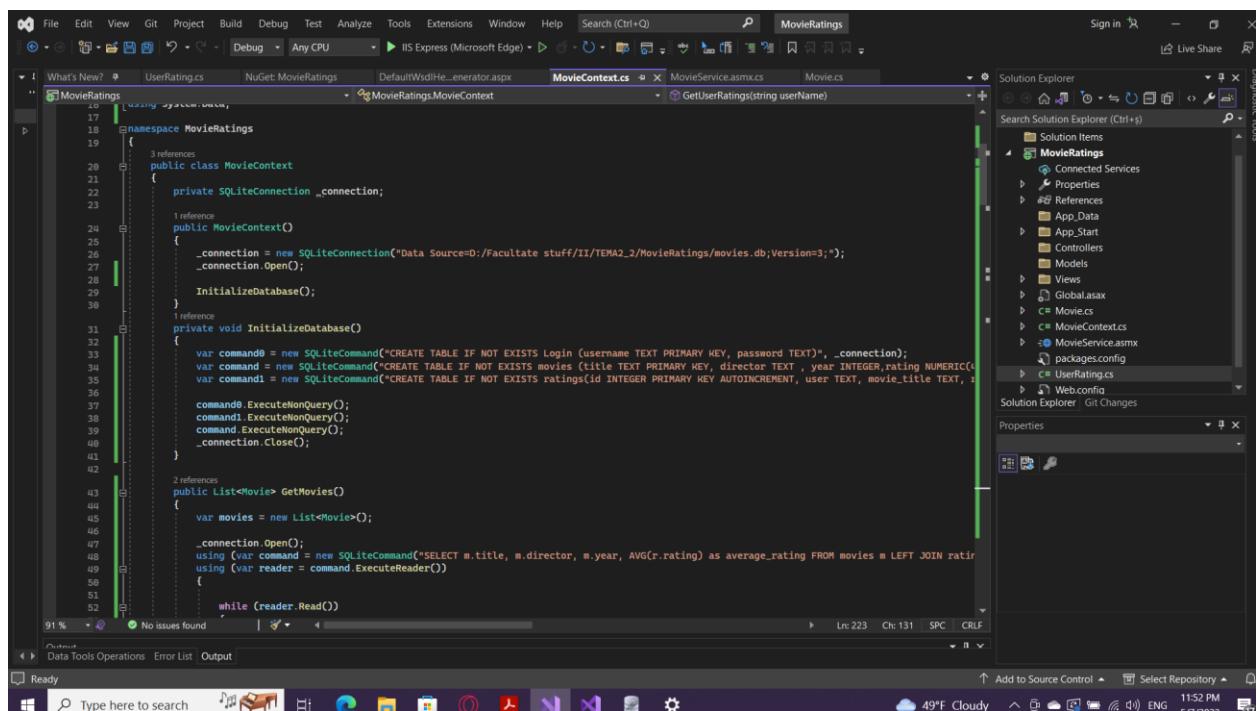


## TEMA 2 – Poran Tudor -Gr.30131

Această aplicație permite utilizatorilor să acceseze și să interacționeze cu o bază de date care conține informații despre filme, printre care și un rating mediu al celorlalți utilizatori. Fiecare utilizator în parte poate să își vadă propriile rating-uri și să adauge filme noi în baza de date, pentru a fi notate/evaluate de restul utilizatorilor.



The screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the `MovieContext.cs` file, which contains C# code for managing a SQLite database. The code includes methods for creating tables, initializing the database, and retrieving movie data. The Solution Explorer on the right shows the project structure for "MovieRatings" with files like `UserRating.cs`, `Movie.cs`, and `MovieService.asmx.cs`. The status bar at the bottom indicates the system is at 49°F Cloudy, the time is 11:52 PM, and the user has 5 notifications.

```
using System;
using System.Data;
using System.Data.SQLite;
namespace MovieRatings
{
    public class MovieContext
    {
        private SQLiteConnection _connection;
        public MovieContext()
        {
            _connection = new SQLiteConnection("Data Source=D:/Facultate stuff/II/TEMA2_3/MovieRatings/movies.db;Version=3;");
            _connection.Open();
            InitializeDatabase();
        }
        private void InitializeDatabase()
        {
            var command0 = new SQLiteCommand("CREATE TABLE IF NOT EXISTS Login (username TEXT PRIMARY KEY, password TEXT)", _connection);
            var command1 = new SQLiteCommand("CREATE TABLE IF NOT EXISTS movies (title TEXT PRIMARY KEY, director TEXT, year INTEGER, rating NUMERIC(4,2))");
            var command2 = new SQLiteCommand("CREATE TABLE IF NOT EXISTS ratings(id INTEGER PRIMARY KEY AUTOINCREMENT, user TEXT, movie_title TEXT, rating NUMERIC(4,2))");
            command0.ExecuteNonQuery();
            command1.ExecuteNonQuery();
            command2.ExecuteNonQuery();
            _connection.Close();
        }
        public List<Movie> GetMovies()
        {
            var movies = new List<Movie>();
            _connection.Open();
            using (var command = new SQLiteCommand("SELECT m.title, m.director, m.year, AVG(r.rating) as average_rating FROM movies m LEFT JOIN ratings r ON m.title = r.movie_title GROUP BY m.title"))
            {
                using (var reader = command.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        movies.Add(new Movie()
                        {
                            Title = reader.GetString(0),
                            Director = reader.GetString(1),
                            Year = reader.GetInt32(2),
                            AverageRating = reader.GetDouble(3)
                        });
                    }
                }
            }
            return movies;
        }
    }
}
```

Screenshot of Microsoft Visual Studio showing the MovieRatings project in the Solution Explorer. The MovieContext.cs file is open in the code editor, displaying C# code for a SQLite database context. A tooltip is visible over the `_connection` field, indicating it is of type `SQLiteConnection`. The code handles user login, rating addition, and connection management.

```

    public bool Login(string username, string password, bool valid)
    {
        _connection.Open();
        //string username = textBoxUsername.Text;
        //string password = textBoxPassword.Text;
        valid = false;

        string query = "SELECT COUNT(*) FROM Login WHERE username=@username AND password=@password";
        using (SQLiteCommand command = new SQLiteCommand(query, _connection))
        {
            command.Parameters.AddWithValue("@username", username);
            command.Parameters.AddWithValue("@password", password);

            int count = Convert.ToInt32(command.ExecuteScalar());
            if (count > 0)
            {
                valid = true;
                _connection.Close();
                return valid;
            }
            else
            {
                valid = false;
                _connection.Close();
                return valid;
            }
        }
    }

    public void AddRating(string movieTitle, string user, float rating)
    {
        _connection.Open();
        var command = new SQLiteCommand("INSERT OR REPLACE INTO ratings (movie_title, user, rating) VALUES (@movieTitle, @user, @rating)", _connection);
        command.Parameters.AddWithValue("@movieTitle", movieTitle);
        command.Parameters.AddWithValue("@user", user);

        float roundedRating = (float)Math.Round(rating, 2);
        command.Parameters.AddWithValue("@rating", roundedRating);
        command.ExecuteNonQuery();
        _connection.Close();
    }

```

The screenshot displays two instances of the MovieContext.cs file from a .NET application named MovieRatings. The left instance shows the original code, while the right instance shows the modified code. Both instances are displayed in the Visual Studio code editor.

**Original Code (Left):**

```
        public List<Movie> GetMovies()
        {
            var movies = new List<Movie>();

            _connection.Open();
            var command = new SQLiteCommand("SELECT m.title, m.director, m.year, AVG(r.rating) as average_rating FROM movies m LEFT JOIN ratings r ON m.title = r.movie_title WHERE NOT EXISTS (SELECT 1 FROM movies WHERE title = m.title AND director = m.director AND year = m.year)", _connection);
            using (var reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    float averageRating = reader.IsDBNull(0) ? 0 : reader.GetFloat(0);
                    float roundedAverageRating = (float)Math.Round(averageRating, 2);

                    var movie = new Movie(reader.GetString(0), reader.GetString(1), reader.GetInt32(2), averageRating);
                    movies.Add(movie);
                }
            }
            _connection.Close();
            return movies;
        }

        public void AddMovie(Movie movie)
        {
            // Movie movie = new Movie(title, Director, Year);
            _connection.Open();
            var command = new SQLiteCommand("INSERT INTO movies (title, director, year) SELECT @title, @director, @year WHERE NOT EXISTS (SELECT 1 FROM movies WHERE title = @title AND director = @director AND year = @year)", _connection);
            command.Parameters.AddWithValue("@title", movie.Title);
            command.Parameters.AddWithValue("@director", movie.Director);
            command.Parameters.AddWithValue("@year", movie.Year);
            command.ExecuteNonQuery();
            _connection.Close();
        }

        public void AddRating(string movieTitle, string user, float rating)
        {
            _connection.Open();
            var command = new SQLiteCommand("UPDATE ratings SET rating = @rating WHERE movie_title = @movieTitle AND user = @user", _connection);
            command.Parameters.AddWithValue("@rating", rating);
            command.Parameters.AddWithValue("@movieTitle", movieTitle);
            command.Parameters.AddWithValue("@user", user);
            command.ExecuteNonQuery();
            _connection.Close();
        }
    }
```

**Modified Code (Right):**

```
        public List<Movie> GetMovies()
        {
            var movies = new List<Movie>();

            _connection.Open();
            var command = new SQLiteCommand("SELECT m.title, m.director, m.year, AVG(r.rating) as average_rating FROM movies m LEFT JOIN (SELECT movie_title, rating FROM ratings GROUP BY movie_title) r ON m.title = r.movie_title WHERE NOT EXISTS (SELECT 1 FROM movies WHERE title = m.title AND director = m.director AND year = m.year)", _connection);
            using (var reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    float averageRating = reader.IsDBNull(0) ? 0 : reader.GetFloat(0);
                    float roundedAverageRating = (float)Math.Round(averageRating, 2);

                    var movie = new Movie(reader.GetString(0), reader.GetString(1), reader.GetInt32(2), averageRating);
                    movies.Add(movie);
                }
            }
            _connection.Close();
            return movies;
        }

        public void AddMovie(Movie movie)
        {
            // Movie movie = new Movie(title, Director, Year);
            _connection.Open();
            var command = new SQLiteCommand("INSERT INTO movies (title, director, year) SELECT @title, @director, @year WHERE NOT EXISTS (SELECT 1 FROM movies WHERE title = @title AND director = @director AND year = @year)", _connection);
            command.Parameters.AddWithValue("@title", movie.Title);
            command.Parameters.AddWithValue("@director", movie.Director);
            command.Parameters.AddWithValue("@year", movie.Year);
            command.ExecuteNonQuery();
            _connection.Close();
        }

        public void AddRating(string movieTitle, string user, float rating)
        {
            _connection.Open();
            var command = new SQLiteCommand("UPDATE ratings SET rating = @rating WHERE movie_title = @movieTitle AND user = @user", _connection);
            command.Parameters.AddWithValue("@rating", rating);
            command.Parameters.AddWithValue("@movieTitle", movieTitle);
            command.Parameters.AddWithValue("@user", user);
            command.ExecuteNonQuery();
            _connection.Close();
        }
    }
```

The screenshot displays two instances of Microsoft Visual Studio running side-by-side. Both instances show the same project structure and code files.

**Top Instance:**

- Solution Explorer:** Shows the project "MovieRatings" with files like MovieService.asmx, MovieContext.cs, Movie.cs, and UserRating.cs.
- Properties:** Shows standard file properties.
- Task List:** Shows "No issues found".
- Code Editor:** Displays the content of `MovieService.asmx.cs`. The code includes several WebMethod annotations and database logic using Entity Framework.

```
31 [WebMethod]
32 public List<Movie> GetMovies()
33 {
34     return _context.GetMovies().ToList();
35 }
36
37 [WebMethod]
38 public void AddMovie(string Title, string Director, int Year)
39 {
40     Movie movie = new Movie{Title, Director, Year, 0};
41     _context.AddMovie(movie);
42 }
43
44 [WebMethod]
45 public List<Movie> SearchMovies(string title)
46 {
47     var movies = _context.GetMovies() Where(m => m.Title.Contains(title)).ToList();
48     return movies;
49 }
50
51 [WebMethod]
52 public void AddRating(string movieTitle, string user, float rating)
53 {
54     float roundedRating = (float) Math.Round(rating, 2);
55     _context.AddRating(movieTitle, user, roundedRating);
56 }
57
58 [WebMethod]
59 public bool Login(string username, string password)
60 {
61     return _context.Login(username, password, false);
62 }
63
64 [WebMethod]
65 public bool AddUser(string username, string password)
66 {
67     return _context.AddUser(username, password, true);
68 }
69
70 [WebMethod]
71 public void SetMovieAverageRating(string movieTitle)
72 {
73 }
```

**Bottom Instance:**

- Solution Explorer:** Shows the project "MovieRatings" with files like MovieService.asmx, MovieContext.cs, Movie.cs, and UserRating.cs.
- Properties:** Shows standard file properties.
- Task List:** Shows "No issues found".
- Code Editor:** Displays the content of `MovieContext.cs`. The code includes database logic using SQLite and Entity Framework.

```
206     averageRating;
207 }
208
209 [reference]
210 public void SetMovieAverageRating(string movieTitle)
211 {
212     var averageRating = GetMovieAverageRating(movieTitle);
213
214     _connection.Open();
215     var command = new SQLiteCommand("UPDATE movies SET rating = @averageRating WHERE title = @movieTitle", _connection);
216     command.Parameters.AddWithValue("@averageRating", averageRating);
217     command.Parameters.AddWithValue("@movieTitle", movieTitle);
218     command.ExecuteNonQuery();
219     _connection.Close();
220 }
221
222 [reference]
223 public List<UserRating> GetUserRatings(string userName)
224 {
225     var ratings = new List<UserRating>();
226
227     _connection.Open();
228     using (var command = new SQLiteCommand("SELECT movie_title, rating FROM ratings WHERE user = @userName", _connection))
229     {
230         command.Parameters.AddWithValue("@userName", userName);
231         using (var reader = command.ExecuteReader())
232         {
233             while (reader.Read())
234             {
235                 var rating = new UserRating
236                 {
237                     MovieTitle = reader.GetString(0),
238                     Username = userName,
239                     Rating = reader.GetFloat(1)
240                 };
241                 ratings.Add(rating);
242             }
243         }
244     }
245 }
```

Screenshot of Microsoft Visual Studio showing two code editors side-by-side, both displaying C# files for a Movie Ratings application.

**Top Window (MovieContext.cs):**

```

105     return valid;
106 }
107 }
108 }
109 }
110 }
111 }
112 }
113 }
114 }
115 }
116 }
117 }
118 }
119 }
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }

1 reference
public void SetMovieAverageRating(string movieTitle)
{
    float averageRating = 0;
    int ratingCount = 0;

    if (_connection.State == ConnectionState.Closed)
        _connection.Open();

    using (SQLiteCommand command = new SQLiteCommand("SELECT rating FROM ratings WHERE movie_title = @title", _connection))
    {
        command.Parameters.AddWithValue("@title", movieTitle);

        using (SQLiteDataReader reader = command.ExecuteReader())
        {
            while (reader.Read())
            {
                float rating = reader.GetFloat(0);
                averageRating += rating;
                ratingCount++;
            }
        }
    }

    if (ratingCount > 0)
    {
        averageRating /= ratingCount;
    }

    // Round to 2 decimal places
    averageRating = (float)Math.Round(averageRating, 2);
    if (_connection.State == ConnectionState.Open)
        _connection.Close();
    return averageRating;
}

```

**Bottom Window (UserRating.cs):**

```

121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }

1 reference
public bool AddUser(string username, string password, bool valid)
{
    _connection.Open();
    valid = true;
    string query = "SELECT COUNT(*) FROM Login WHERE username=@username AND password=@password";
    using (SQLiteCommand command = new SQLiteCommand(query, _connection))
    {
        command.Parameters.AddWithValue("@username", username);
        command.Parameters.AddWithValue("@password", password);
        username = Regex.Replace(username, @"\s+", ""); // removes whitespaces
        if (!Regex.IsMatch(username, "[a-zA-Z0-9-]+"))
        {
            // Nu este de utilizator nu respectă formatul dorit, afișăm un mesaj de eroare

            valid = false;
            _connection.Close();
            return valid;
        }
    }

    int count = Convert.ToInt32(command.ExecuteScalar());
    if (count > 0)
    {
        // MessageBox.Show("Login successful.");
    }

    valid = false;
    _connection.Close();
    return valid;
}

if (valid)
{
    SQLiteCommand cmd = new SQLiteCommand("INSERT INTO Login (Username, Password) VALUES ('" + username + "','" + password + "')", _connection);
    cmd.ExecuteNonQuery();
    _connection.Close();
    return valid;
}

_connection.Close();

```

Screenshot of Microsoft Visual Studio showing two code editors side-by-side, both displaying the same C# file: MovieContext.cs.

**Top Window (MovieContext.cs):**

```

171     public void AddUser(string username, string password, bool valid)
172     {
173         _connection.Open();
174         valid = true;
175         string query = "SELECT COUNT(*) FROM Login WHERE username=@username AND password=@password";
176         using (SQLiteCommand command = new SQLiteCommand(query, _connection))
177         {
178             command.Parameters.AddWithValue("@username", username);
179             command.Parameters.AddWithValue("@password", password);
180             username = Regex.Replace(username, @"\s+", ""); // removes whitespaces
181             if (!Regex.IsMatch(username, @"[a-zA-Z0-9_-]+"))
182             {
183                 // Numele de utilizator nu respectă formatul dorit, afișăm un mesaj de eroare
184
185                 valid = false;
186                 _connection.Close();
187                 return valid;
188             }
189
190             int count = Convert.ToInt32(command.ExecuteScalar());
191             if (count > 0)
192             {
193                 // MessageBox.Show("Login successful.");
194
195                 valid = false;
196                 _connection.Close();
197                 return valid;
198             }
199             if (valid)
200             {
201                 SQLiteCommand cmd = new SQLiteCommand("INSERT INTO Login (Username, Password) VALUES ('" + username + "', '" + password + "')", _connection);
202                 cmd.ExecuteNonQuery();
203                 _connection.Close();
204                 return valid;
205             }
206         }
207         _connection.Close();

```

**Bottom Window (MovieContext.cs):**

```

166     public float GetMovieAverageRating(string movieTitle)
167     {
168         float averageRating = 0;
169         int ratingCount = 0;
170
171         if (_connection.State == ConnectionState.Closed)
172             _connection.Open();
173
174         using (SQLiteCommand command = new SQLiteCommand("SELECT rating FROM ratings WHERE movie_title = @title", _connection))
175         {
176             command.Parameters.AddWithValue("@title", movieTitle);
177             using (SQLiteDataReader reader = command.ExecuteReader())
178             {
179                 while (reader.Read())
180                 {
181                     float rating = reader.GetFloat(0);
182                     averageRating += rating;
183                     ratingCount++;
184                 }
185             }
186
187             if (ratingCount > 0)
188                 averageRating /= ratingCount;
189
190             // Round to 2 decimal places
191             averageRating = (float)Math.Round(averageRating, 2);
192             if (_connection.State == ConnectionState.Open)
193                 _connection.Close();
194             return averageRating;
195         }
196     }
197
198     public void SetMovieAverageRating(string movieTitle)
199     {
200
201
202
203
204
205
206
207

```

The Solution Explorer on the right shows the project structure for "MovieRatings" with files like MovieContext.cs, MovieService.asmx, and UserRating.cs.

Screenshot of Microsoft Visual Studio showing two code editors side-by-side.

**Left Editor:** Displays the `MovieContext.cs` file. The code implements a `MovieContext` class with methods for setting movie average ratings and getting user ratings. It uses SQLite and Entity Framework. The code editor shows lines 207 to 404.

```

207     public void SetMovieAverageRating(string movieTitle)
208     {
209         var averageRating = GetMovieAverageRating(movieTitle);
210
211         _connection.Open();
212         var command = new SQLiteCommand("UPDATE movies SET rating = @averageRating WHERE title = @movieTitle", _connection);
213         command.Parameters.AddWithValue("@averageRating", averageRating);
214         command.Parameters.AddWithValue("@movieTitle", movieTitle);
215         command.ExecuteNonQuery();
216         _connection.Close();
217     }
218
219     public List<UserRating> GetUserRatings(string userName)
220     {
221         var ratings = new List<UserRating>();
222
223         _connection.Open();
224         using (var command = new SQLiteCommand("SELECT movie_title, rating FROM ratings WHERE user = @userName", _connection))
225         {
226             command.Parameters.AddWithValue("@userName", userName);
227
228             using (var reader = command.ExecuteReader())
229             {
230                 while (reader.Read())
231                 {
232                     var rating = new UserRating()
233                     {
234                         MovieTitle = reader.GetString(0),
235                         Username = userName,
236                         Rating = reader.GetFloat(1)
237                     };
238                     ratings.Add(rating);
239                 }
240             }
241             _connection.Close();
242         }
243
244         return ratings;
245     }

```

**Right Editor:** Displays the `MovieService.asmx.cs` file. This is a Web Service implementation. It includes methods for getting movies, adding movies, searching movies, adding ratings, logging in, and adding users. The code editor shows lines 31 to 73.

```

31     [WebMethod]
32     public List<Movie> GetMovies()
33     {
34         return _context.GetMovies().ToList();
35     }
36
37     [WebMethod]
38     public void AddMovie(string Title, string Director, int Year)
39     {
40         Movie movie = new Movie(Title, Director, Year, 0);
41         _context.AddMovie(movie);
42     }
43
44     [WebMethod]
45     public List<Movie> SearchMovies(string title)
46     {
47         var movies = _context.GetMovies().Where(m => m.Title.Contains(title)).ToList();
48         return movies;
49     }
50
51     [WebMethod]
52     public void AddRating(string movieTitle, string user, float rating)
53     {
54         float roundedRating = (float) Math.Round(rating, 2);
55         _context.AddRating(movieTitle, user, roundedRating);
56     }
57
58     [WebMethod]
59     public bool Login(string username, string password)
60     {
61         return _context.Login(username, password, false);
62     }
63
64     [WebMethod]
65     public bool AddUser(string username, string password)
66     {
67         return _context.AddUser(username, password, true);
68     }
69
70     [WebMethod]
71     public void AddUser(string username, string password)
72     {
73     }

```

The Solution Explorer on the right shows the project structure for "MovieRatings" with files like `MovieContext.cs`, `MovieService.asmx`, and `UserRating.cs`.

**Screenshot 1 (Top): MovieRatings Service**

```

    public void AddRating(string movieTitle, string user, float rating)
    {
        float roundedRating = (float)Math.Round(rating, 2);
        _context.AddRating(movieTitle, user, roundedRating);
    }

    [WebMethod]
    public bool Login(string username, string password)
    {
        return _context.Login(username, password, false);
    }

    [WebMethod]
    public bool AddUser(string username, string password)
    {
        return _context.AddUser(username, password, true);
    }

    [WebMethod]
    public float GetMovieAverageRating(string movieTitle)
    {
        return _context.GetMovieAverageRating(movieTitle);
    }

    [WebMethod]
    public void SetMovieAverageRating(string movieTitle)
    {
        _context.SetMovieAverageRating(movieTitle);
    }

    [WebMethod]
    public List<UserRating> GetUserRatings(string userName)
    {
        return _context.GetUserRatings(userName);
    }
}

```

**Screenshot 2 (Bottom): MovieServiceUser Application**

```

    public partial class Form1 : Form
    {
        MovieServiceUser.MovieServiceReference.MovieServiceSoapClient service =
            new MovieServiceUser.MovieServiceReference.MovieServiceSoapClient();

        // DataSet dsLogin;
        int reference;
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
        }

        private void buttonCreateUser_Click(object sender, EventArgs e)
        {
            Form createAdmin = new CreateUser();
            createAdmin.Show();
        }

        private void buttonLogin_Click(object sender, EventArgs e)
        {
            string username = textBoxUsername.Text;
            string password = textBoxPassword.Text;

            bool valid = service.Login(username, password);

            if (valid)
            {
                // MessageBox.Show("User loged");
            }
        }
    }

```

The image displays three vertically stacked screenshots of the Microsoft Visual Studio IDE interface, showing different parts of a Windows application project.

**Screenshot 1 (Top):** Shows the `Form1.cs [Design]` tab selected. The code editor contains the following C# code:

```
private void buttonCreateUser_Click(object sender, EventArgs e)
{
    CreateAdmin createAdmin = new CreateAdmin();
    createAdmin.Show();
}

private void buttonLogin_Click(object sender, EventArgs e)
{
    string username = textboxUsername.Text;
    string password = textboxPassword.Text;

    bool valid = service.Login(username, password);

    if (valid)
    {
        MovieApp movieApp = new MovieApp(username);
        movieApp.Show();
        this.Hide();
    }
    else
        MessageBox.Show("Parola sau user invalid");
}
```

**Screenshot 2 (Middle):** Shows the `CreateUser.cs [Design]` tab selected. The code editor contains the following C# code:

```
public partial class CreateUser : Form
{
    MovieServiceUser.MovieServiceReference.MovieServiceSoapClient service =
        new MovieServiceUser.MovieServiceReference.MovieServiceSoapClient();

    public CreateUser()
    {
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        string username = textboxUsername.Text;
        string password = textboxPassword.Text;
        bool valid = service.AddUser(username, password);

        if (valid)
            MessageBox.Show("User-ul a fost adaugat cu succes");
        else
            MessageBox.Show("User invalid");
    }
}
```

**Screenshot 3 (Bottom):** Shows the `MovieApp.cs [Design]` tab selected. The code editor contains the following C# code:

```
namespace MovieServiceUser
{
    public partial class Form1 : Form
    {
        MovieServiceUser.MovieServiceReference.MovieServiceSoapClient service =
            new MovieServiceUser.MovieServiceReference.MovieServiceSoapClient();

        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string username = textboxUsername.Text;
            string password = textboxPassword.Text;
            bool valid = service.Login(username, password);

            if (valid)
                MovieApp movieApp = new MovieApp(username);
                movieApp.Show();
                this.Hide();
            else
                MessageBox.Show("Parola sau user invalid");
        }
    }
}
```

MovieApp.cs

```

13  namespace MovieServiceUser
14  {
15      public partial class MovieApp : Form
16      {
17          MovieServiceUser.MovieServiceReference.MovieServiceSoapClient service =
18              new MovieServiceUser.MovieServiceReference.MovieServiceSoapClient();
19          private List<MovieServiceReference.Movie> movies;
20          private string currentUser;
21          private string username;
22
23          public MovieApp(string username)
24          {
25              InitializeComponent();
26              currentUser = username;
27              label1.Text = username + "'s Ratings ";
28
29              MovieServiceReference.Movie[] movieArray = service.GetMovies();
30
31              // Convert the array to a list
32              movies = movieArray.ToList();
33
34              dataGridView1.AutoGenerateColumns = false;
35              dataGridView1.Columns.Add(new DataGridViewTextBoxColumn() { HeaderText = "Title", DataPropertyName = "Title" });
36              dataGridView1.Columns.Add(new DataGridViewTextBoxColumn() { HeaderText = "Director", DataPropertyName = "Director" });
37              dataGridView1.Columns.Add(new DataGridViewTextBoxColumn() { HeaderText = "Year", DataPropertyName = "Year" });
38              dataGridView1.Columns.Add(new DataGridViewTextBoxColumn() { HeaderText = "Rating", DataPropertyName = "Rating" });
39              dataGridView1.DataSource = movies;
40              dataGridView1.Refresh();
41
42              dataGridView1.ReadOnly = false;
43              foreach (DataGridViewColumn column in dataGridView1.Columns)
44              {
45                  column.ReadOnly = false;
46              }
47              dataGridView1.AllowUserToAddRows = true;
48              dataGridView1.Anchor = AnchorStyles.Top | AnchorStyles.Left | AnchorStyles.Bottom | AnchorStyles.Right;
49              //dataGridView1.Dock = DockStyle.Fill;
50
51
52              var userRatings = service.GetUserRatings(currentUser);
53
54              dataGridView2.AutoGenerateColumns = false;
55              dataGridView2.Columns.Add(new DataGridViewTextBoxColumn() { HeaderText = "Movie Title", DataPropertyName = "MovieTitle" });
56              dataGridView2.Columns.Add(new DataGridViewTextBoxColumn() { HeaderText = "Rating", DataPropertyName = "Rating" });
57              dataGridView2.DataSource = userRatings;
58              dataGridView2.Anchor = AnchorStyles.Top | AnchorStyles.Left | AnchorStyles.Bottom | AnchorStyles.Right;
59
60              dataGridView2.Refresh();
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83

```

No issues found

Form1.cs [Design] CreateUser.cs [Design] CreateUser.cs Form1.cs NuGet: MovieServiceUser MovieApp.cs [Design]\*

button1\_Click(object sender, EventArgs e)

Solution Explorer

Search Solution Explorer (Ctrl+Shift+F)

- Settings.settings
- References
- App.config
- CreateUser.cs
- CreateUser.Designer.cs
- CreateUser.resx
- Database1.mdf
- Database1\_log.ldf
- Form1.cs
- Form1.Designer.cs
- Form1.resx
- MovieAdd.cs
- MovieApp.cs
- MovieApp.Designer.cs
- MovieApp.resx
- packages.config
- Program.cs

Solution Explorer Git Changes

Properties

Add to Source Control

Ready

Type here to search

49°F Cloudy 11:56 PM 5/27/2022

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) MovieServiceUser

MovieApp.cs

```

44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83

```

No issues found

Form1.cs [Design] CreateUser.cs [Design] CreateUser.cs Form1.cs NuGet: MovieServiceUser MovieApp.cs [Design]\*

button1\_Click(object sender, EventArgs e)

Solution Explorer

Search Solution Explorer (Ctrl+Shift+F)

- Settings.settings
- References
- App.config
- CreateUser.cs
- CreateUser.Designer.cs
- CreateUser.resx
- Database1.mdf
- Database1\_log.ldf
- Form1.cs
- Form1.Designer.cs
- Form1.resx
- MovieAdd.cs
- MovieApp.cs
- MovieApp.Designer.cs
- MovieApp.resx
- packages.config
- Program.cs

Solution Explorer Git Changes

Properties

Add to Source Control

Ready

Type here to search

49°F Cloudy 11:56 PM 5/27/2022

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) MovieServiceUser

MovieApp.cs

```

85     public void UpdateDataGridView(List<MovieServiceReference.Movie> movieList)
86     {
87         dataGridView1.DataSource = movieList;
88         dataGridView1.Refresh();
89     }
90
91     private void label1_Click(object sender, EventArgs e)
92     {
93     }
94
95
96     private void dataGridView1_CellContentClick(object sender, DataGridViewCellEventArgs e)
97     {
98     }
99
100    private void button1_Click_1(object sender, EventArgs e)
101    {
102    }
103
104    private void buttonRating_Click(object sender, EventArgs e)
105    {
106        float rating = float.Parse(textBoxRating.Text);
107        float roundedRating = (float)Math.Round(rating, 2);
108        string selectedMovieTitle = dataGridView1.SelectedRows[0].Cells[0].Value.ToString();
109        service.AddRating(selectedMovieTitle, currentUser, roundedRating);
110
111        MessageBox.Show("Rating adaugat cu succes");
112        // Call the GetMovieRatings method to retrieve the updated movie rating
113        float updatedRating = service.GetMovieAverageRating(selectedMovieTitle);
114        service.SetMovieAverageRating(selectedMovieTitle);
115        // Update the movie rating in the dataGridView
116        dataGridView1.SelectedRows[0].Cells[3].Value = updatedRating;
117        UpdateDataGridView(movies);
118        List<UserRating> userRatings = service.GetUserRatings(currentUser).ToList();
119        dataGridView2.DataSource = userRatings;
120        dataGridView2.Refresh();
121    }
122
123 }

```

No issues found

Ready

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) MovieServiceUser

Server Explorer Toolbox

Solution Explorer

Search Solution Explorer (Ctrl+F)

- Settings.settings
- References
- App.config
- CreateUser.cs
- CreateUser.Designer.cs
- CreateUser.resx
- Database1.mdf
- Database1\_log.ldf
- Form1.cs
- Form1.Designer.cs
- Form1.resx
- MovieAdd.cs
- MovieApp.cs
- MovieApp.Designer.cs
- MovieApp.resx
- packages.config
- Program.cs

Solution Explorer Git Changes Properties

Add to Source Control

49°F Cloudy 11:56 PM ENG

MovieApp.cs

```

98
99
100
101    private void button1_Click_1(object sender, EventArgs e)
102    {
103    }
104
105    private void buttonRating_Click(object sender, EventArgs e)
106    {
107        float rating = float.Parse(textBoxRating.Text);
108        float roundedRating = (float)Math.Round(rating, 2);
109        string selectedMovieTitle = dataGridView1.SelectedRows[0].Cells[0].Value.ToString();
110        service.AddRating(selectedMovieTitle, currentUser, roundedRating);
111
112        MessageBox.Show("Rating adaugat cu succes");
113        // Call the GetMovieRatings method to retrieve the updated movie rating
114        float updatedRating = service.GetMovieAverageRating(selectedMovieTitle);
115        service.SetMovieAverageRating(selectedMovieTitle);
116        // Update the movie rating in the dataGridView
117        dataGridView1.SelectedRows[0].Cells[3].Value = updatedRating;
118        UpdateDataGridView(movies);
119        List<UserRating> userRatings = service.GetUserRatings(currentUser).ToList();
120        dataGridView2.DataSource = userRatings;
121        dataGridView2.Refresh();
122    }
123
124 }

```

No issues found

Ready

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) MovieServiceUser

Server Explorer Toolbox

Solution Explorer

Search Solution Explorer (Ctrl+F)

- Settings.settings
- References
- App.config
- CreateUser.cs
- CreateUser.Designer.cs
- CreateUser.resx
- Database1.mdf
- Database1\_log.ldf
- Form1.cs
- Form1.Designer.cs
- Form1.resx
- MovieAdd.cs
- MovieApp.cs
- MovieApp.Designer.cs
- MovieApp.resx
- packages.config
- Program.cs

Solution Explorer Git Changes Properties

Add to Source Control

49°F Cloudy 11:56 PM ENG

MovieApp.cs\* CreateUser.cs [Design] CreateUser.cs Form1.cs MovieApp.cs [Design]\* MovieAdd.cs [Design] MovieAdd.cs

```

12  {
13      public partial class MovieAdd : Form
14      {
15          private List<MovieServiceReference.Movie> movieList;
16          private DataGridView dataGridView;
17          MovieServiceUser.MovieServiceSoapClient service =
18              new MovieServiceUser.MovieServiceReference.MovieServiceSoapClient();
19
20          reference
21          public MovieAdd(List<MovieServiceReference.Movie> movieList, DataGridView dataGridView)
22          {
23              InitializeComponent();
24              this.movieList = movieList;
25              this.dataGridView = dataGridView;
26          }
27
28          reference
29          private void buttonAdd_Click(object sender, EventArgs e)
30          {
31              // Get the text from the three textboxes
32              string title = textBoxTitle.Text;
33              string director = textBoxDirector.Text;
34              int year = int.Parse(textBoxYear.Text);
35
36              // Create a new movie object
37              MovieServiceReference.Movie newMovie = new MovieServiceReference.Movie
38              {
39                  Title = title,
40                  Director = director,
41                  Year = year
42              };
43
44              // Add the new movie object to the list of movies
45              movieList.Add(newMovie);
46              try
47              {
48                  // Call the AddMovie method in the web service
49                  service.AddMovie(newMovie.Title, newMovie.Director, newMovie.Year);
50                  Console.WriteLine("Movie added to database.");
51              }
52          }
53
54
55          // Set the DataSource of the DataGridView to the updated BindingList object
56          dataGridView.DataSource = movieList;
57
58          MovieApp parentForm = (MovieApp)this.Owner;
59          parentForm.UpdateDataGridView(movieList);
60
61          // Close the MovieAdd form
62          this.Close();
63      }
64  }

```

No issues found | Data Tools Operations Error List Output

Ready Type here to search 49°F Cloudy Add to Source Control Live Share

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) MovieServiceUser

Solution Explorer

- Settings.settings
- References
- App.config
- CreateUser.cs
  - CreateUser.Designer.cs
  - CreateUser.resx
- Database1.mdf
  - Database1\_log.ldf
- Form1.cs
  - Form1.Designer.cs
  - Form1.resx
- MovieAdd.cs
- MovieApps
  - MovieApp.cs
    - MovieApp.Designer.cs
    - MovieApp.resx
    - packages.config
  - Program.cs

Solution Explorer Git Changes

Properties

Add to Source Control

Ready Type here to search 49°F Cloudy Add to Source Control Live Share

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) MovieServiceUser

Solution Explorer

- Settings.settings
- References
- App.config
- CreateUser.cs
  - CreateUser.Designer.cs
  - CreateUser.resx
- Database1.mdf
  - Database1\_log.ldf
- Form1.cs
  - Form1.Designer.cs
  - Form1.resx
- MovieAdd.cs
- MovieApps
  - MovieApp.cs
    - MovieApp.Designer.cs
    - MovieApp.resx
    - packages.config
  - Program.cs

Solution Explorer Git Changes

Properties

Add to Source Control

MovieApp.cs\* MovieServiceUser

```
30     string title = textBoxTitle.Text;
31     string director = textBoxDirector.Text;
32     int year = int.Parse(textBoxYear.Text);
33
34     // Create a new movie object
35     MovieServiceReference.Movie newMovie = new MovieServiceReference.Movie()
36     {
37         Title = title,
38         Director = director,
39         Year = year
40     };
41
42     // Add the new movie object to the list of movies
43     movieList.Add(newMovie);
44     try
45     {
46         // Call the AddMovie method in the web service
47         service.AddMovie(newMovie.Title, newMovie.Director, newMovie.Year);
48         Console.WriteLine("Movie added to database.");
49     }
50     catch (Exception ex)
51     {
52         Console.WriteLine("Error adding movie to database: " + ex.Message);
53     }
54
55     // Set the DataSource of the DataGridView to the updated BindingList object
56     dataGridView.DataSource = movieList;
57
58     MovieApp parentForm = (MovieApp)this.Owner;
59     parentForm.UpdateDataGridView(movieList);
60
61     // Close the MovieAdd form
62     this.Close();
63 }
64
65 }
66
67 }
```

90% No issues found Ln: 29 Ch: 13 SPC CRLF

Data Tools Operations Error List Output

Ready Type here to search File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) MovieServiceUser

MovieApp.cs\* MovieServiceUser

```
30     string title = textBoxTitle.Text;
31     string director = textBoxDirector.Text;
32     int year = int.Parse(textBoxYear.Text);
33
34     // Create a new movie object
35     MovieServiceReference.Movie newMovie = new MovieServiceReference.Movie()
36     {
37         Title = title,
38         Director = director,
39         Year = year
40     };
41
42     // Add the new movie object
43     movieList.Add(newMovie);
44     try
45     {
46         // Call the AddMovie method in the web service
47         service.AddMovie(newMovie.Title, newMovie.Director, newMovie.Year);
48         Console.WriteLine("Movie added to database.");
49     }
50     catch (Exception ex)
51     {
52         Console.WriteLine("Error adding movie to database: " + ex.Message);
53     }
54
55     // Set the DataSource of
56     dataGridView.DataSource =
57
58     MovieApp parentForm = (MovieApp)this.Owner;
59     parentForm.UpdateDataGridView(movieList);
60
61     // Close the MovieAdd form
62     this.Close();
63 }
64
65 }
66
67 }
```

SQLiteStudio (3.4.4) - (movies (movies))

Databases

| title               | director          | year | rating            |
|---------------------|-------------------|------|-------------------|
| The Matrix Reloaded | Lana Wachowski    | 2003 | 5.17000007629395  |
| The Matrix          | Lana Wachowski    | 1999 | 7.099999900463257 |
| Inception           | Christopher Nolan | 2010 | 9.8699998855908   |
| Fight Club          | David Fincher     | 1999 | 8.31999969482422  |
| Barbie              | Greta Gerwig      | 2024 | 8.7600002288184   |
| Shrek               | Vicky Jenson      | 2001 | 10                |

Structure Data Constraints Indexes Triggers DDL

Status

- [22:29:52] Database passed in command line parameters (D:\Facultate stuff\II\TEMA2\_2\MovieRatings\movies.db) was already on the list under name: movies
- [22:33:08] Committed changes for table 'ratings' successfully.

90% No issues found Ln: 29 Ch: 13 SPC CRLF

Data Tools Operations Error List Output

MovieApp.cs\* CreateUser.cs [Design] CreateUser.cs [Design] MovieApp.cs [Design] MovieAdd.cs [Design]

```

30     string title = textBoxTitle.Text;
31     string director = textBoxDirector.Text;
32     int year = int.Parse(textBoxYear.Text);
33
34     // Create a new movie object
35     MovieServiceReference.Movie newMovie =
36     {
37         Title = title,
38         Director = director,
39         Year = year
40     };
41
42     // Add the new movie object
43     movieList.Add(newMovie);
44     try
45     {
46         // Call the AddMovie
47         service.AddMovie(newMovie);
48         Console.WriteLine("Movie added successfully!");
49     }
50     catch (Exception ex)
51     {
52         Console.WriteLine("Error: " + ex.Message);
53     }
54
55     // Set the DataSource of
56     // dataGridView1
57     dataGridView1.DataSource =
58     MovieApp.parentForm = (MovieAdd)parentForm;
59     parentForm.UpdateDataGridView();
60
61     // Close the MovieAdd form
62     this.Close();
63 }
64
65 }
66
67 }
68

```

90 % No issues found Data Tools Operations Error List Output

Ready

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) MovieServiceUser TP Live Share

SQLiteStudio (3.4.4) - [Login (movies)] Database Structure View Tools Help

Databases

Filter by name

- movies (SQLite 3)
  - Tables (3)
    - Login
    - movies
    - ratings
  - Columns (4)
  - Indexes
  - Triggers
  - Views

Structure Data Constraints Indexes Triggers DDL

Grid view Form view Filter data

|   | username | password |
|---|----------|----------|
| 1 | tudor    | 12345    |
| 2 | poran    | tudor    |
| 3 | ruxibuxi | ruxi24   |

Status

[22:29:52] Database passed in command line parameters (D:\Facultate stuff\II\TEMA2\_2\MovieRatings\movies.db) was already on the list under name: movies

[22:33:08] Committed changes for table 'ratings' successfully.

movies (movies) Login (movies) ratings (movies)

Ln: 29 Ch: 13 SPC CRLF

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) MovieServiceUser TP Live Share

SQLiteStudio (3.4.4) - [ratings (movies)] Database Structure View Tools Help

Databases

Filter by name

- movies (SQLite 3)
  - Tables (3)
    - Login
    - movies
    - ratings
  - Columns (4)
  - Indexes
  - Triggers
  - Views

Structure Data Constraints Indexes Triggers DDL

Grid view Form view Filter data

|   | id   | user  | movie title         | rating           |
|---|------|-------|---------------------|------------------|
| 1 | NULL | poran | Shrek               | 9.80000019073486 |
| 2 | NULL | tudor | Fight Club          | 8.64999961853027 |
| 3 | NULL | poran | The Matrix Reloaded | 8.10000038146973 |
| 4 | NULL | tudor | Shrek               | 10               |
| 5 | NULL | tudor | Barbie              | 7.76999998092651 |
| 6 | NULL | tudor | The Matrix          | 8.30000019073486 |
| 7 | NULL | tudor | The Matrix Reloaded | 6.5              |
| 8 | NULL | poran | Barbie              | 3                |
| 9 | NULL | tudor | Inception           | 10               |

Status

[22:29:52] Database passed in command line parameters (D:\Facultate stuff\II\TEMA2\_2\MovieRatings\movies.db) was already on the list under name: movies

[22:33:08] Committed changes for table 'ratings' successfully.

movies (movies) Login (movies) ratings (movies)

Ln: 29 Ch: 13 SPC CRLF

Type here to search 49°F Cloudy 1157 PM 5/7/2023

Screenshot of the Microsoft Visual Studio IDE showing a multi-monitor setup. The top monitor displays the SQL Server Object Explorer and the bottom monitor displays the Windows Taskbar.

**Top Monitor (Visual Studio):**

- SQL Server Object Explorer:** Shows a database named "movies" with three tables: "Login", "movies", and "ratings".
- Code Editor:** Displays "MovieApp.cs" with C# code for creating a movie object and adding it to a list.
- Toolbars:** Standard Visual Studio toolbars for File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help.
- Status Bar:** Shows "MovieServiceUser" and various status indicators.

**Bottom Monitor (Windows Taskbar):**

- Taskbar:** Shows the Start button, search bar, pinned icons for File Explorer, Edge, and File Explorer, and the system tray with battery, signal, and volume icons.
- System Icons:** Shows the date and time as "5/7/2023 11:57 PM" and the weather as "49°F Cloudy".

