UNIVERSITATEA TEHNICĂ GHEORGHE ASACHI, IAȘI

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

Ingineria Programării

**Mp3Player**

1309B

Apostol Roxana-Maria

Baciu Raluca-Daniela

Nistor Dumitrița-Florina

Pîrvan Ines-Iuliana

2023

1. **Documentul specificațiilor cerințelor (SRS)**
2. **Introducere**
   1. Scop

Documentația SRS (Software Requirements Specification) pentru Mp3Player are ca scop definirea cerințelor sistemului, descrierea funcționalităților și a caracteristicilor utilizatorilor, precum și stabilirea constrângerilor sistemului. Această documentație servește drept bază pentru dezvoltarea și implementarea Mp3Player-ului în C#.

* 1. Definiții, acronime, abrevieri
* SRS: Software Requirements Specification
* MP3: Formatul de fișier audio MPEG-1 Audio Layer 3
* C#: Un limbaj de programare orientat pe obiecte dezvoltat de Microsoft
  1. Referințe
     1. Ingineria Programării – laboratoare
     2. Ingineria Programării – cursuri
     3. https: //app.diagrams.net/
     4. https://www.dofactory.com/net/command-design-pattern
     5. https://refactoring.guru/design-patterns/command/csharp/example
     6. https://refactoring.guru/design-patterns/command
     7. https://www.c-sharpcorner.com/UploadFile/8ea152/mp3-media-player-in-C-Sharp-4-0/
     8. https://refactoring.guru/design-patterns/facade
     9. https://stackoverflow.com/questions/466458/c-sharp-audio-library

1. **Descriere de ansamblu**
   1. Perspectiva produsului

Mp3Player-ul este un program de redare a fișierelor audio în format MP3. Acesta permite utilizatorilor să adauge, să șteargă și să redea melodii, să creeze playlist-uri personalizate și să acceseze versurile melodiilor.

* 1. Funcțiile produsului

Mp3Player-ul oferă următoarele funcționalități:

* Adăugare și ștergere de melodii



2.2.1 Utilizatorul poate adăuga melodii în biblioteca Mp3Player-ului.

2.2.2 Utilizatorul poate șterge melodii din biblioteca Mp3Player-ului.

* Redare și control al melodiei



2.2.3 Utilizatorul poate reda o melodie selectată din biblioteca Mp3Player-ului.

2.2.4 Utilizatorul poate opri redarea unei melodii.

2.2.5 Utilizatorul poate pune în pauză/reda o melodie.

2.2.6 Utilizatorul poate regla volumul redării melodiei.

* Creare, editare și ștergere playlist-uri



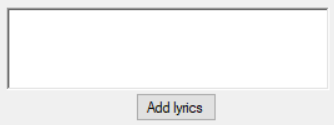
2.2.7 Utilizatorul poate crea playlist-uri personalizate.

2.2.8 Utilizatorul poate adăuga melodii în playlist-uri.

2.2.9 Utilizatorul poate șterge playlist-uri.

2.2.10 Utilizatorul poate reda playlist-uri.

* Accesare versuri pentru melodii



2.2.11 Utilizatorul poate accesa versurile pentru melodii în timpul redării.

* 1. Caracteristicile utilizatorului

Mp3Player-ul este destinat utilizatorilor obișnuiți care doresc să redea fișiere audio în format MP3, să creeze playlist-uri personalizate și să acceseze versurile melodiilor.

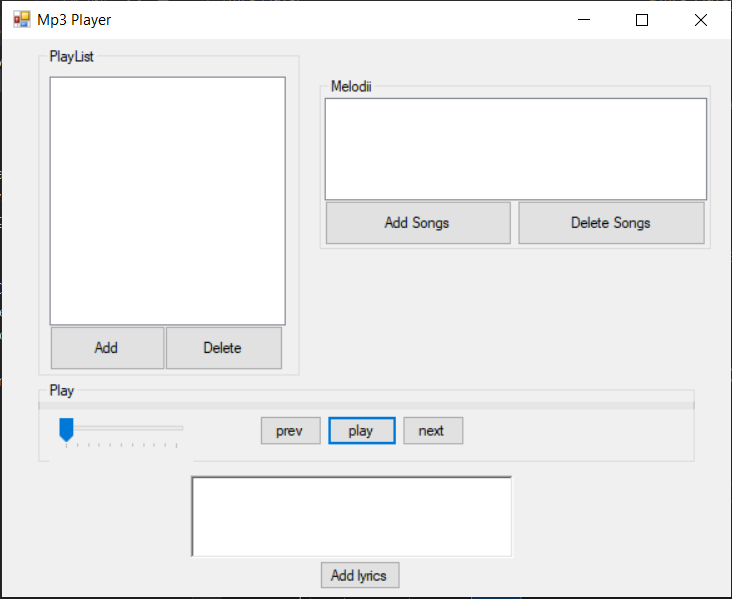
* 1. Constrângeri generale
* Mp3Player-ul este dezvoltat în limbajul de programare C#.
* Sistemul este limitat la redarea fișierelor audio în format MP3.
* Interacțiunea cu interfața grafică a Mp3Player-ului se face prin intermediul mouse-ului.
* Versurile se preiau din txt.

1. **Cerințe specifice**
   1. Cerințe non-funcționale

* Performanță
* Ușurința în utilizare
  1. Cerințe funcționale
* Adăugare și gestionare melodii
* Redare și control al melodiilor
* Interfața cu utilizatorul

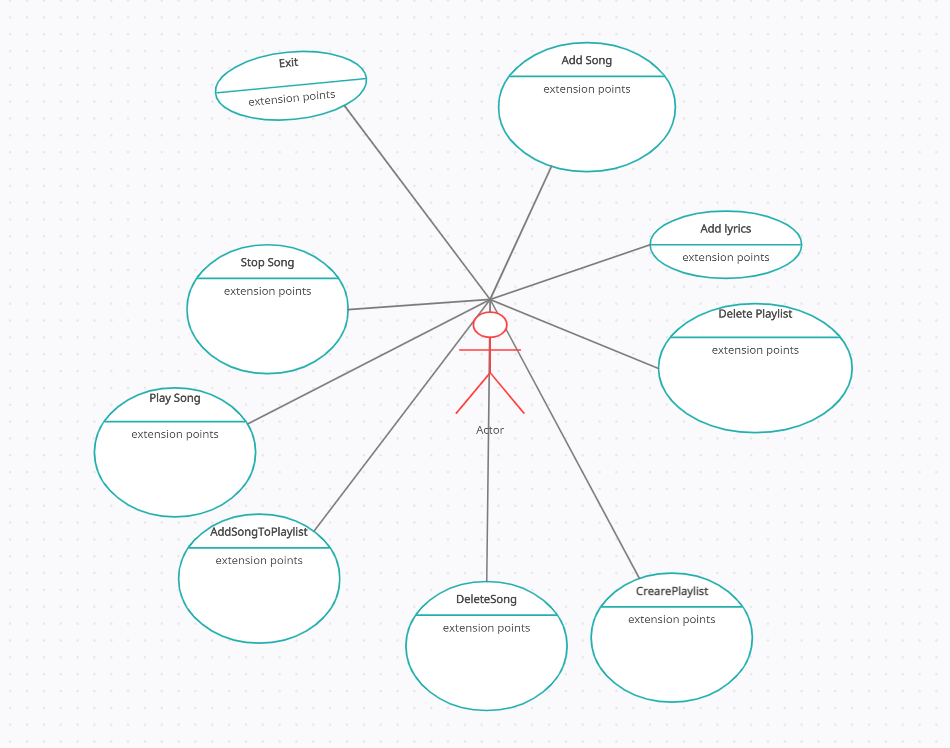
1. **Arhitectură sistem**

* Interfața grafică cu utilizatorul

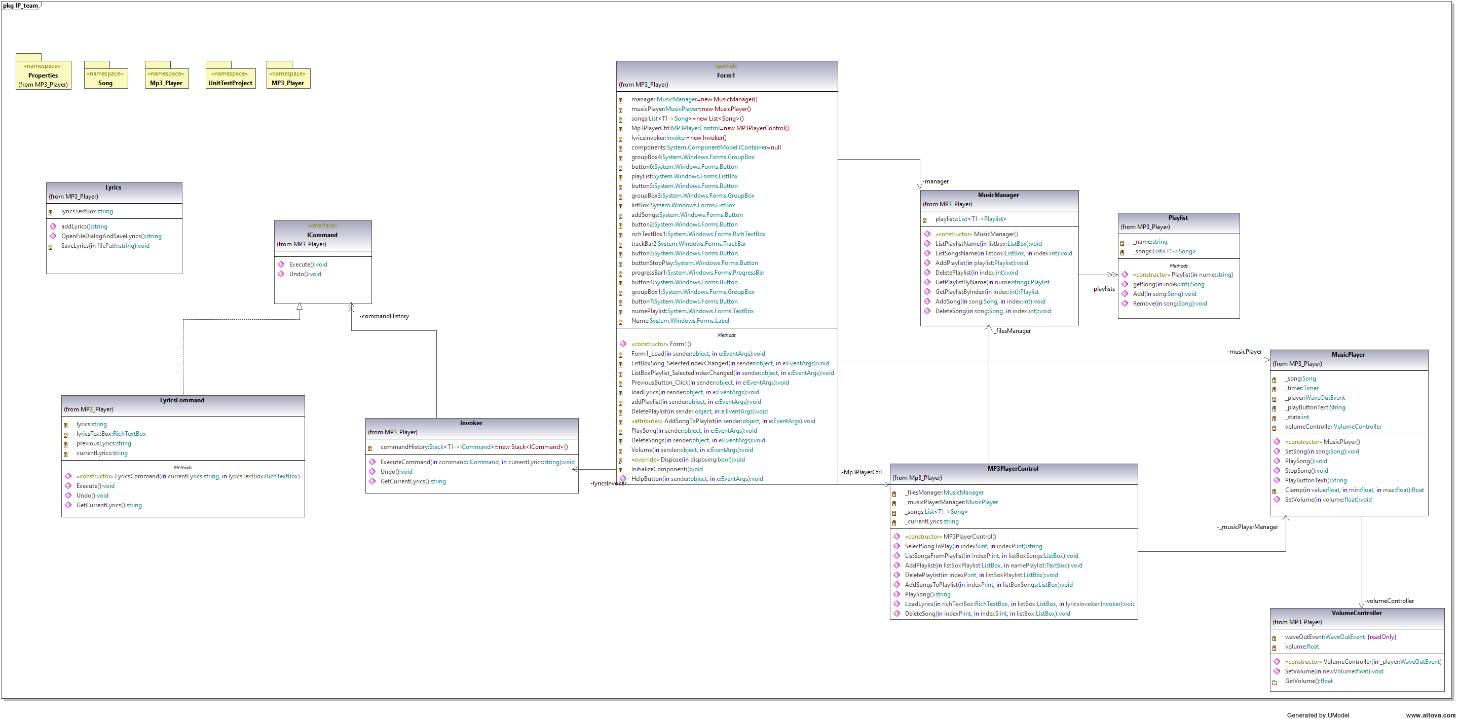


* Logica aplicației
* Gestionarea datelor (informații despre melodii, versuri)

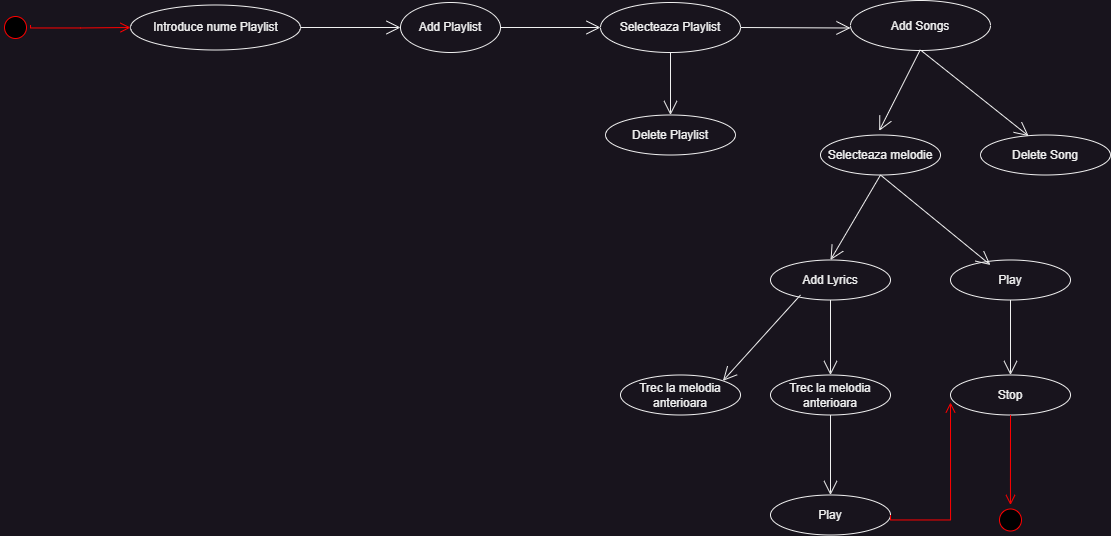
1. **Diagrame UML**
2. Diagrama cazurilor de utilizare



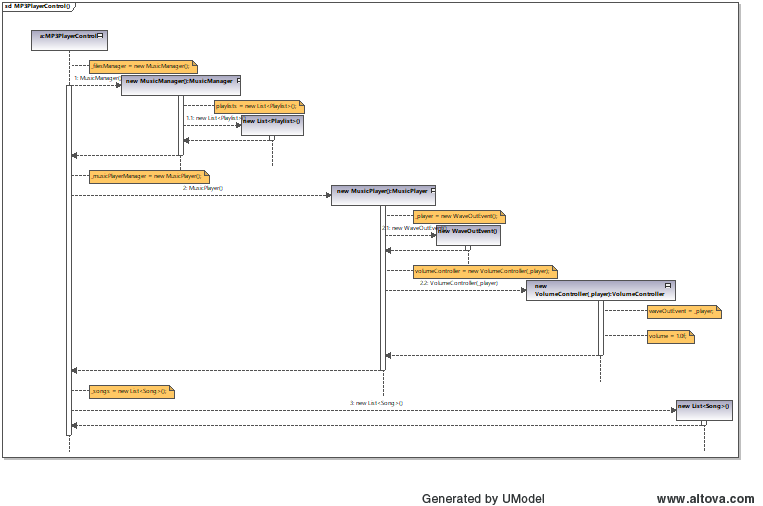
1. Diagrama de clase



1. Diagrama de activități



1. Diagrama de secvențe

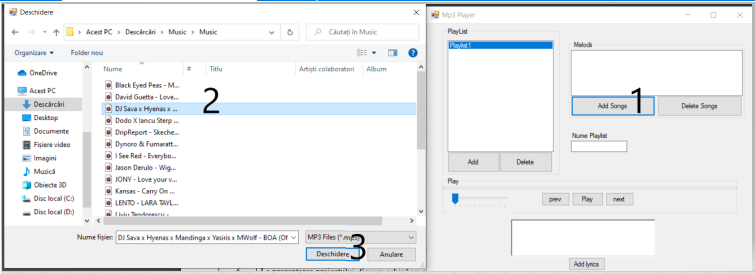


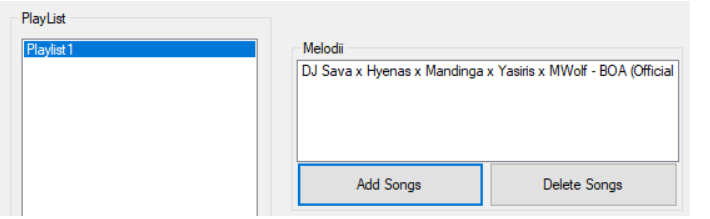
1. **Modul de utilizare al programului**

Mp3Player-ul este un program de redare a fișierelor audio în format MP3. Acesta permite utilizatorilor să adauge, să șteargă și să redea melodii, să creeze playlist-uri personalizate și să acceseze versurile melodiilor.

**Add+Delete Songs**

Butonul este inițial dezactivat. Se activează la crearea unui playlist. Prin click, se poate adăuga o melodie dintr-un director într-unul dintre playlist-urile create.

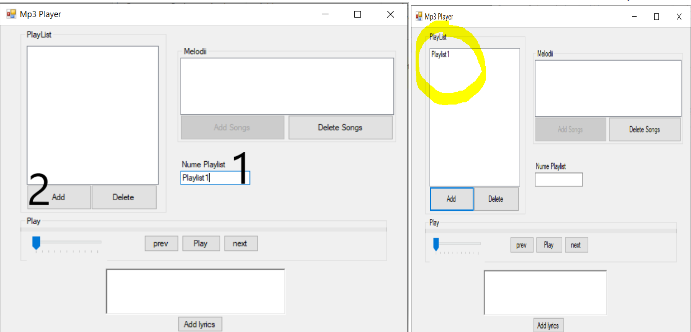




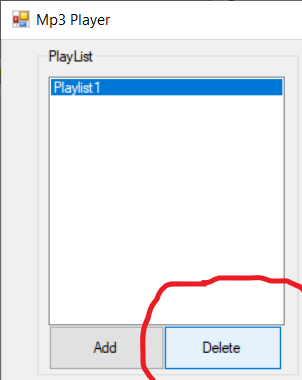
realizează ștergerea unei melodii dintr-un playlist creat.

**Add+Delete Playlist**

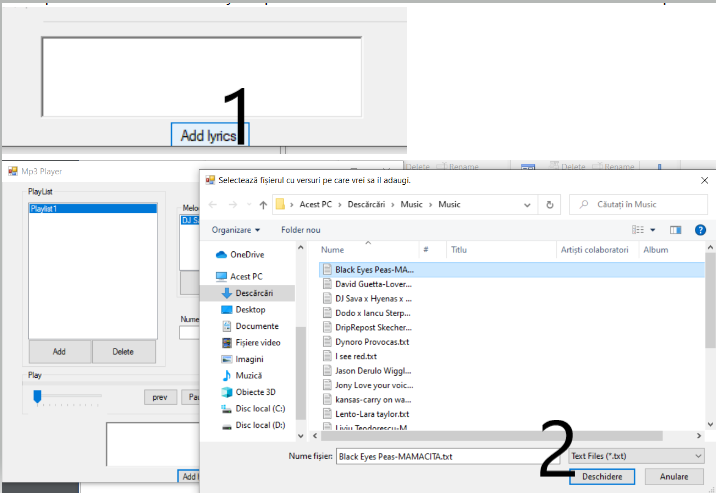
Pentru a adăuga un Playlist nou se introduce numele pe care dorim să îl dăm în căsuța ”Nume Playlist” și se apasă butonul **Add**.



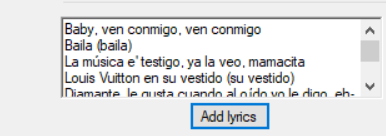
Butonul **Delete** realizează ștergerea playlist-ului selectat.



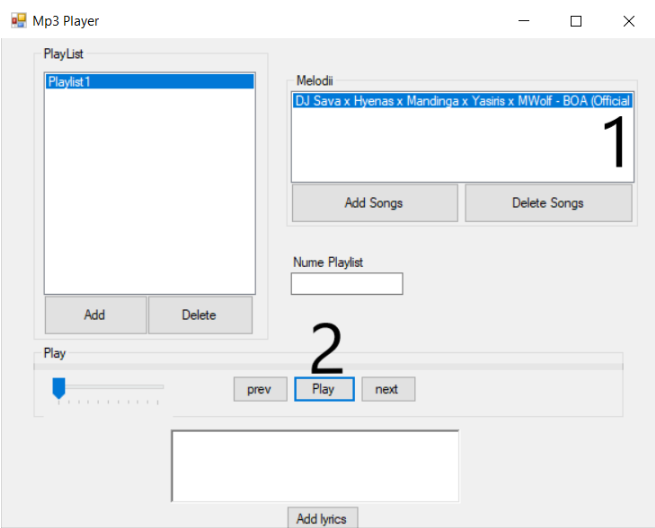
Prin apăsarea butonului **Add Lyrics** putem selecta dintr-o listă de txt-uri versurile melodiei pe care le dorim.



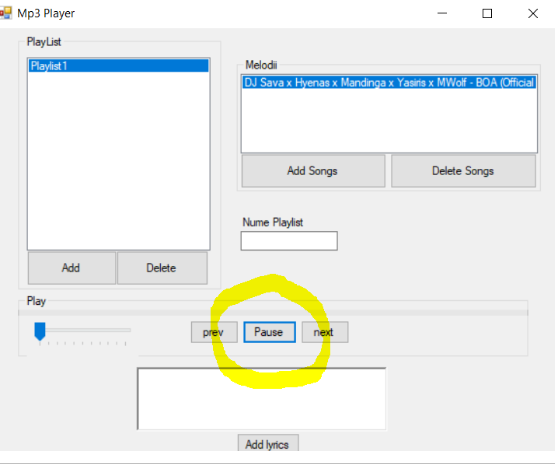
Versurile apar în căsuța text superioară butonului.



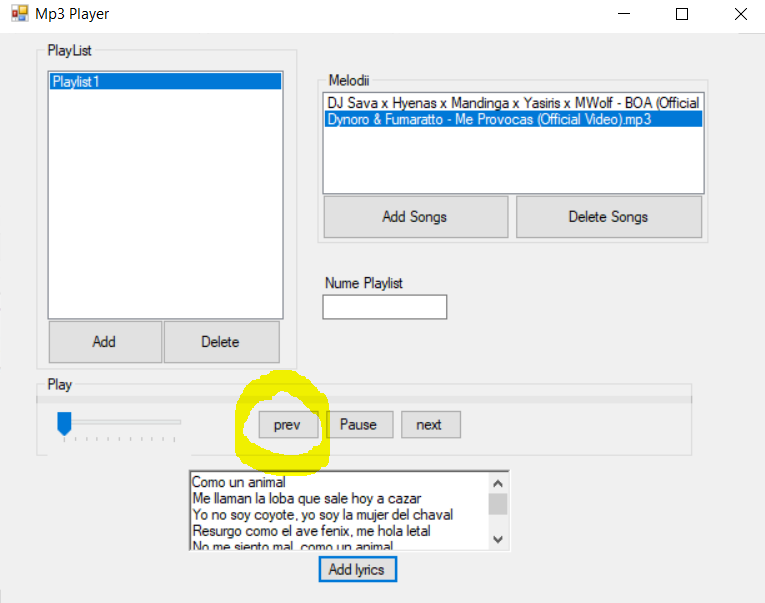
Pentru a porni melodia o selectăm și apăsăm butonul **Play**.



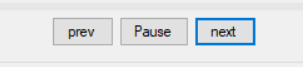
Pentru a pune pauză, se apasă butonul **Pause**.



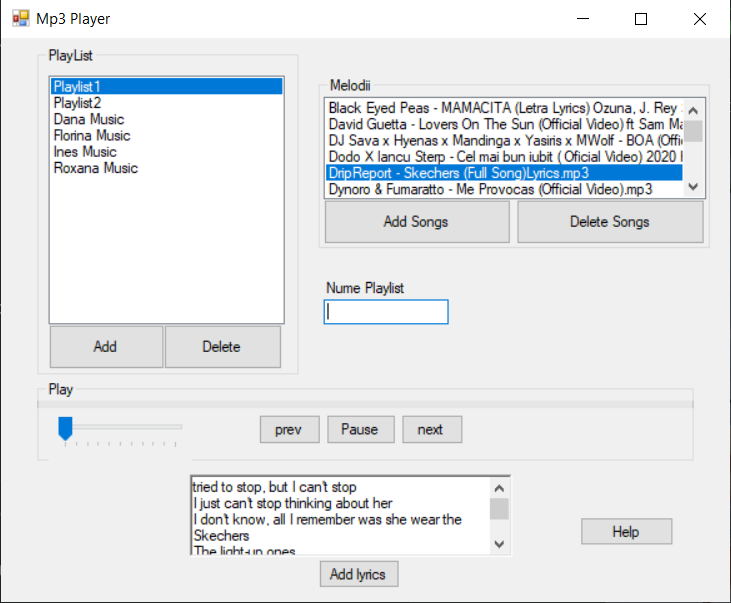
Butonul **Prev** permite întoarcerea la versurile melodiei anterioare. Când se dă play la o nouă melodie, versurile anterioare dispar.



Butonul **Next** trece la următoarea melodie din Playlist.



1. **Programul în execuție (playlist-uri+lista melodii+lyrics)**



1. **Anexe**

**Detalii implementare (clasele AddPlaylist, DeletePlaylist, AddSongsToPlaylist, NextSong)**

// La apasarea butonului Add Playlist se creaza un nou obiect de tip

// playlist cu numele pe care il primeste din TextBox-ul "Nume Playlist"

// care va fi inserat in lista de redare

public void AddPlaylist(ListBox listBoxPlaylist, TextBox namePlaylist)

{

try

{

// Verificați dacă numele playlist-ului nu este gol

if (!string.IsNullOrWhiteSpace(namePlaylist.Text))

{

// Creați un nou playlist și adăugați-l în MusicManager

Playlist newPlaylist = new Playlist(namePlaylist.Text);

\_filesManager.AddPlaylist(newPlaylist);

// Adăugați numele playlist-ului în listbox

listBoxPlaylist.Items.Add(namePlaylist.Text);

// Goliți caseta de text pentru numele playlist-ului

namePlaylist.Text = string.Empty;

}

else

{

throw new Exception("Introduceți un nume valid pentru playlist.");

}

}

catch (Exception ex)

{

MessageBox.Show("A apărut o excepție: " + ex.Message, "Eroare");

}

}

// Daca exista un playlist selectat atunci la apasarea butonului Delete

// se sterge din lista de redare elementul cu indexul aferent

public void DeletePlaylist(int indexP, ListBox listBoxPlaylist)

{

try

{

if (indexP >= 0)

{

// Ștergeți playlist-ul din MusicManager

\_filesManager.DeletePlaylist(indexP);

// Ștergeți numele playlist-ului din listbox

listBoxPlaylist.Items.RemoveAt(indexP);

}

else

{

throw new Exception("Selectați un playlist pentru a șterge.");

}

}

catch (Exception ex)

{

MessageBox.Show("A apărut o excepție: " + ex.Message, "Eroare");

}

}

// Adaugarea unor fisiere de tip .mp3 in playlistul selectat curent

public void AddSongsToPlaylist(int indexP, ListBox listBoxSongs)

{

// Se deschide o fereastra in care putem naviga pentru a selecta fisierele

// pe care vrem sa le importam

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "MP3 Files (\*.mp3)|\*.mp3";

openFileDialog.Multiselect = true;

// Adaugam noi obiecte <Song> in playlist, obiecte initializate cu

// campurile file si path ale fisierelor selectate din fereastra

// proaspat deschisa; daca fereastra s-a deschis:

if (openFileDialog.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

// Initializam campul file al obiectelor de tip <Song>

foreach (String file in openFileDialog.SafeFileNames)

{

Song song = new Song();

song.Name = file;

\_songs.Add(song);

}

// Initializam campul path al obiectelor de tip <Song>

int j = 0;

foreach (String path in openFileDialog.FileNames)

{

\_songs[j].FilePath = path;

j++;

}

// Adaugam obiectele <Song> create anterior in playlistul curent

foreach (Song song in \_songs)

{

\_filesManager.AddSong(song, indexP);

}

// Afisam obiectele <Song> in ListBox si curatam lista pentru

// urmatoarea incarcare

for (int i = 0; i < \_songs.Count; i++)

{

listBoxSongs.Items.Add(\_songs[i].Name);

}

\_songs.Clear();

}

}

public int NextSong(int indexP, int currentIndex)

{

// Incrementăm indexul pentru a obține următorul index

int nextIndex = currentIndex + 1;

// Obținem numărul total de melodii în playlist

int noSongs = \_filesManager.GetPlaylistByIndex(indexP).SongsNumber();

// Verificăm dacă suntem la ultima melodie din playlist

if (nextIndex >= noSongs)

{

nextIndex = 0; // Dacă suntem la ultima melodie, trecem la prima melodie

}

// Obținem melodia următoare din playlist

Song song = \_filesManager.GetPlaylistByIndex(indexP).getSong(nextIndex);

// Setăm melodia în MusicPlayerManager

\_musicPlayerManager.SetSong(song);

// Returnăm indexul melodiei următoare

return nextIndex;

}

public void LoadLyrics(RichTextBox richTextBox, ListBox listBox, Invoker lyricsInvoker)

{

try

{

if (string.IsNullOrEmpty(richTextBox.Text) && (listBox.SelectedIndex != -1))

{

// Se creează un obiect de tip Lyrics

Lyrics lyrics = new Lyrics();

// Se deschide un dialog de selectare a fișierului și se salvează versurile selectate

string selectedLyrics = lyrics.OpenFileDialogAndSaveLyrics();

// Se actualizează conținutul controlului RichTextBox cu versurile selectate

richTextBox.Text = selectedLyrics;

\_currentLyrics = richTextBox.Text;

// Se creează o comandă de tip LyricsCommand cu versurile și controlul RichTextBox

ICommand lyricsCommand = new LyricsCommand(\_currentLyrics, richTextBox);

// Se execută comanda folosind obiectul Invoker

lyricsInvoker.ExecuteCommand(lyricsCommand, \_currentLyrics);

}

else if (!string.IsNullOrEmpty(richTextBox.Text))

{

// Se aruncă o excepție dacă conținutul controlului RichTextBox nu este gol

throw new Exception("Ați adăugat deja versuri pentru melodie.");

}

else if (listBox.SelectedIndex == -1)

{

// Se aruncă o excepție dacă nu este selectată nicio melodie în controlul ListBox

throw new Exception("Pentru a adăuga versuri trebuie să selectați mai întâi o melodie.");

}

}

catch (Exception ex)

{

// Se afișează un mesaj de eroare într-un MessageBox în caz de excepție

MessageBox.Show("A apărut o excepție: " + ex.Message);

}

}

**Testare și validare**

// Nu se sterge si lista daca am un singur element

[TestMethod]

public void Test\_MusicManager\_7\_DeleteSongFrom\_Playlist()

{

Song song = new Song(name, path);

Playlist playlist1 = new Playlist("Playlist de test 1");

MusicManager playlists = new MusicManager();

playlist1.Add(song);

playlists.AddPlaylist(playlist1);

playlists.DeleteSong(song,0);

Assert.IsTrue((playlists.GetPlaylistByName("Playlist de test 1") == null));

}

}

**Gestionarea erorilor și excepțiilor**

public void DeleteSong(int indexP, int indexS, ListBox listBox)

{

try

{

if (indexP >= 0 && indexS >= 0)

{

Playlist selectedPlaylist = \_filesManager.GetPlaylistByIndex(indexP);

Song selectedSong = selectedPlaylist.getSong(indexS);

if (selectedSong != null)

{

\_filesManager.DeleteSong(selectedSong, indexP);

listBox.Items.RemoveAt(indexS);

}

else

{

throw new Exception("Selectați un cântec pentru a șterge.");

}

}

else

{

throw new Exception("Selectați un playlist și un cântec pentru a șterge.");

}

}

catch (Exception ex)

{

MessageBox.Show("A apărut o excepție: " + ex.Message, "Eroare");

}

}

**Task-uri**

* Florina

◦ clasa MP3PlayerControl care cuprinde un șablon de proiectare numit Fațadă;

◦ Help

◦ Documentația

◦ Diagrama cazuri de utilizare

* Roxana

◦ Șablon de proiectare numit Command care cuprinde interfața ICommand, LyricsCommand, Invoker

◦ Clasa lyrics și legătura cu butonul Add lyrics și cu butonul Prev

◦ Diagrame UML

◦ VolumeController

* Dana

◦ Clasa MusicManager,clasa Playlist și clasa Song

◦ Legătura cu butoanele Add și Delete Playlist și Legătura cu butoanele add și delete Songs

◦ Tratarea excepțiilor

◦ Antete surse + interfața

◦ Pentru Next la melodie

* Ines

◦ Clasa MusicPlayer care face legătura cu butonul play/pause

◦ Interfața grafica

◦ DLL-urile

◦ Testarea unităților