

UNIVERSIDAD DE MARGARITA

ALMA MATER DEL CARIBE

VICERRECTORADO ACADÉMICO

DECANATO DE INGENIERIA

UNIDAD CURRICULAR: PROGRAMACION II

**LYRA**

**REPRODUCTOR**

Profesor:

Cesar Requena

Integrantes:

Guillermo García

Los reproductores de audio y video son herramientas fundamentales en nuestro día a día, permitiéndonos disfrutar y compartir contenido multimedia de manera fácil y conveniente. Desde su surgimiento, han evolucionado de manera impresionante, adaptándose a las necesidades y demandas tecnológicas de cada época.

En sus inicios, los reproductores se limitaban únicamente a la reproducción de música. Uno de los primeros dispositivos de este tipo fue el fonógrafo, patentado por Thomas Edison en 1877. Este invento revolucionó la manera en que la música podía ser escuchada y replicada, utilizando un cilindro que giraba y una aguja que recorría sus ranuras y emitía el sonido.

Posteriormente, en la década de 1960, se introdujo una nueva forma de reproducción de audio: el casete. Este formato permitió a las personas crear sus propias mezclas de canciones, grabar y escuchar música de manera portátil. A mediados de los años 80, surgió el compact disc (CD), que ofrecía una calidad de audio superior y se convirtió en un estándar para la reproducción musical.

Con la llegada de la era digital, aparecieron los reproductores de audio portátiles más populares como el reproductor de MP3. El MP3 revolucionó la forma en que almacenamos y escuchamos música, permitiéndonos llevar miles de canciones en un dispositivo pequeño y fácil de usar. Desde entonces, los reproductores de audio digital han continuado evolucionando, ofreciendo características como pantalla táctil, conectividad inalámbrica y capacidades de reproducción de múltiples formatos.

En cuanto a los reproductores de video, su historia también ha sido notable. Durante muchos años, la reproducción de películas y programas de televisión se realizaba a través de reproductores de video analógicos, como el famoso VHS. Estos reproductores utilizaban cintas magnéticas para almacenar y reproducir videos.

Sin embargo, con el avance de la tecnología digital, surgieron los reproductores de DVD, que ofrecían una calidad de video superior y permitían una navegación más fácil entre diferentes capítulos o escenas. Con el tiempo, estos fueron reemplazados por el Blu-Ray, que brindaba una resolución aún mayor y capacidad de almacenamiento.

Actualmente, los reproductores de audio y video se han fusionado en dispositivos multifuncionales, como los smartphones y las PC. Estos dispositivos permiten reproducir música, ver videos en alta definición, acceder a servicios de streaming y mucho más.

A través del presente informe se señala las ideas y principios sobre los cuales he hecho el reproductor LYRA, haciendo uso de la plantilla proporcionada por el profesor Cesar Requena.

FASE I

REPRODUCCIÓN DE AUDIO Y VIDEO

El reto del reproductor realmente no es hacer que reproduzca, ya que para ello tenemos distintas librerías y herramientas que hacen esta parte del proyecto mas amena, como son la librería VLC y el WMP. De esta forma, he decidido utilizar WMP, ya que es algo que podemos integrar directamente en el editor de interfaz y nos hace más fácil y visual la edición de este. Culminada esta parte del proyecto lo siguiente sería poder manejar el WMP desde los controles externos proporcionados por la plantilla (para no tener que usar los ya implementados por el WMP).

FASE II

CONTROLES

Ninguno de los controles de la plantilla es funcional en un inicio (de ser así el proyecto perdería el chiste), por lo que fue necesario pero la mar toda su funcionalidad desde cero (así como cambiar su imagen por una propia). De esta forma, utilizamos distintas funciones y sentencias de este WMP que nos permiten controlar diferentes aspectos como el volumen, pausar, iniciar, detener.

A la hora de controlar el volumen, he decidido implementar un track bar con un máximo de 100, usando su evento scroll para que al mover la barra el volumen se adapte al estado en el que el usuario la deja. Además, he implementado botones para repetir, mezclar y detener la reproducción.

FASE III

BARRA DE PROGRESO

La barra de progreso que viene por defecto con la plantilla realmente no es una barra de progreso, sino un panel; por lo que fue completamente inútil y debió ser reemplazado. De esta manera he decidido reemplazarlo por un track bar, el cual, en conjunto con un timer, hacen posible que la barra se mueva en sincronía con el vídeo o audio en reproducción. Esta barra se adapta a la duración del archivo en reproducción, de forma que sin importar qué estés reproduciendo, la barra será capaz de seguirle el ritmo a la duración (por ejemplo, para un archivo de corta duración no te sobrará barra, para un archivo de larga duración no te faltará barra). Así mismo, el track bar es capaz de, al ser modificado por el usuario, establecer la posición del vídeo a la nueva posición de la barra, permitiendo así que el usuario maneje el vídeo de forma más cómoda.

FASE IV

MANEJO DE ARCHIVOS

De nada sirve un reproductor con botones y funciones bien definidas si al final no puedes reproducir un archivo, es por esto por lo que, mediante el botón open file puedes elegir un archivo en tu computadora para reproducirlo; con el botón open folder eres capaz de introducir la ruta de una carpeta y ver todos los archivos reproducibles que se encuentran en ella; con el botón recent media puedes ver los archivos anteriormente reproducidos y volverlos a insertar en el reproductor.

FASE V

MANEJO DE PLAYLIST

Para las playlist he decidido utilizar un data grip view, el cual nos permitirá seleccionar qué canciones queremos agregar a una nueva playlist, para posteriormente guardarla en nuestro dispositivo y reproducirla en cualquier momento (guardándola en un archivo .LYRA), el cual podrá ser reconocido perfectamente por el programa en cualquier otro momento.

Una vez creadas las playlist, con el botón manage playlist podemos ver aquellas playlist que ya han sido creadas, reproducirlas o eliminarlas. Así mismo con el botón export podemos exportar una playlist, guardando todos los archivos de ella en la carpeta que queramos (ideal para cuando queremos pasar una playlist de nuestra pc a un pen drive u otro dispositivo).

FASE VI

EXTRAS AGREGADOS

No hay nada mejor para un programa que (aparte de ser funcional), tenga algo que lo caracterice. Es por esto por lo que he decidido agregar distintas funciones a mi reproductor que permitan marcar la diferencia, como lo son: la opción de descargar vídeo, que nos permite tomar el ID de un video de YouTube y descargarlo en la calidad que queramos (eligiendo dónde queremos que se guarde); la opción de convertir vídeo, la cual nos permite cambiar el formato de un archivo vídeo a otro; la opción de convertir audio, la cual nos permite cambiar el formato de archivo un audio a otro; el extractor de audio, que nos permite separar el audio de un vídeo y guardarlo como un .mp3 (en caso de querer otro formato podemos utilizar el convertidor de audio); la opción de screenshot, que nos permite capturar un momento del vídeo en reproducción y guardarlo en la ruta que queramos.

FASE VII

FINAL

LYRA es un programa que busca brindar una solución a un problema sin mucho esfuerzo para el usuario (reproducir archivos de audio y vídeo), ofreciéndole distintas opciones que puedan ayudarlo que otras aplicaciones de reproducción normalmente no ofrecerían. Como son el poder descargar vídeos de YouTube sin máximo esfuerzo, convertir tus archivos de audio y vídeo a otros formatos o extraer el audio de un vídeo en tan solo un par de clics.

PSEUDOCÓDIGO

Clase Interfaz

Clase Interfaz

Declarar random como Random;

Declarar RecentURLs como arreglo de cadenas;

Declarar ExternalInput como booleano;

Declarar PlayList como arreglo de cadenas;

Declarar MixxedPlaylist como arreglo de cadenas;

Declarar ExternalPlaylist como booleano;

Declarar CurrentElement como entero;

Declarar RepeatMode como booleano;

Declarar RepeatSMode como booleano;

Declarar ShuffleMode como booleano;

Declarar play como booleano;

Declarar Second, Minute, Hour como enteros;

Declarar position como entero;

Declarar TempVol como entero;

Declarar ExternalURL como cadena;

Declarar Ruta como cadena;

Método Interfaz

InicializarComponentes();

OcultarSubMenu();

Asignar evento KeyDown a Form1\_KeyDown\_1;

Asignar evento MouseMove a Form1\_MouseMove;

Fin Método

Método SidePanelHider

Declarar CursorPosition como posición del cursor en la ventana;

Si CursorPosition.X > 233 entonces

panelSideMenu.Visible = falso;

sino

panelSideMenu.Visible = verdadero;

Fin Si

Fin Método

Método SidePanelHider2

Declarar CursorPosition como posición del cursor en la ventana;

Si CursorPosition.X > 20 entonces

panelSideMenu.Visible = falso;

sino

panelSideMenu.Visible = verdadero;

Fin Si

Fin Método

Método CleanPlaylist

ExternalPlaylist = falso;

Declarar temp como arreglo de cadenas;

PlayList = temp;

Fin Método

Método StartTimer

Iniciar timer1;

Fin Método

Método SetVisible

Player.Visible = verdadero;

Traer Player al frente;

Fin Método

Método SetExternalURL(URL como cadena)

ExternalURL = URL;

Mostrar mensaje con ExternalURL;

Reproducir Externamente;

Fin Método

Método Reset

Pausar();

Second = 0;

Minute = 0;

Hour = 0;

position = 0;

MTBPosition.Value = 0;

LblDuration.Text = "00:00:00";

LblPosition.Text = "00:00:00";

Fin Método

Método ExternalPlay

Reset();

Player.URL = ExternalURL;

Fin Método

Método Mixxer(mix como arreglo de cadenas, n como entero)

Si n < longitud de mix - 1 entonces

Incrementar n;

Declarar x como número aleatorio entre 0 y longitud de mix;

Declarar temp como mix[x];

Declarar y como número aleatorio entre 0 y longitud de mix;

mix[x] = mix[y];

mix[y] = temp;

Llamar Mixxer con mix y n;

sino

MixxedPlaylist = mix;

Fin Si

Fin Método

Método PlaylistPlay

CurrentElement = 0;

Reset();

Player.URL = PlayList[CurrentElement];

SetVisible();

Reproducir();

Fin Método

Método NextElement(PL como arreglo de cadenas)

Incrementar CurrentElement;

Reset();

SetTimeStuff();

Player.URL = PL[CurrentElement];

Reproducir();

Si CurrentElement > longitud de PlayList - 2 entonces

Si RepeatMode entonces

CurrentElement = 0;

Reset();

Player.URL = PL[CurrentElement];

Reproducir();

sino

ExternalPlaylist = falso;

Fin Si

Fin Si

Fin Método

Método PreviusElement(PL como arreglo de cadenas)

Decrementar CurrentElement;

Si CurrentElement > -1 entonces

Reset();

Player.URL = PL[CurrentElement];

Reproducir();

sino

Mostrar mensaje "No hay elementos previos";

Fin Si

Fin Método

Método Pausar

Pausar Player.Ctlcontrols;

play = falso;

PBPlay.Image = Propiedades.Resources.play2;

Detener timer1;

Fin Método

Método Play

Reproducir Player.Ctlcontrols;

play = verdadero;

PBPlay.Image = Propiedades.Resources.Pause;

MTBPosition.Enabled = verdadero;

Iniciar timer1;

Fin Método

Método hideSubMenu

panelMediaSubMenu.Visible = falso;

panelPlaylistSubMenu.Visible = falso;

panelToolsSubMenu.Visible = falso;

Fin Método

Método showSubMenu(subMenu como Panel)

Si subMenu.Visible == falso entonces

hideSubMenu();

subMenu.Visible = verdadero;

sino

subMenu.Visible = falso;

Fin Si

Fin Método

Método btnMedia\_Click(sender como objeto, e como EventArgs)

showSubMenu(panelMediaSubMenu);

Fin Método

Método CleanRecentMedia

Declarar ClearURLs como arreglo de cadenas;

RecentURLs = ClearURLs;

openChildForm(new RecentMedia(this));

Fin Método

Método SetRecentMedia(RecentURL como cadena)

Declarar tempURLs como arreglo de cadenas de longitud RecentURLs.Length+1;

Copiar RecentURLs a tempURLs;

tempURLs[tempURLs.Length - 1] = RecentURL;

RecentURLs = tempURLs;

Fin Método

Método SetTimeStuff

MTBPosition.Maximum = (int)Player.currentMedia.duration-1;

MTBPosition.Value = (int)Player.Ctlcontrols.currentPosition;

Declarar HourDuration como MTBPosition.Maximum / 3600;

Declarar MinuteDuration como MTBPosition.Maximum / 60;

Declarar SecondDuration como (MTBPosition.Maximum % 60)+1;

Incrementar position;

Second = position;

Si Second > 59 entonces

Hour = Second / 3600;

Minute = Second / 60;

Second = Second % 60;

Fin Si

Si MinuteDuration > 60 entonces

LblPosition.Text = string.Format("{0:D2}:{1:D2}:{2:D2}", Hour, (Minute - (Hour \* 60)), Second-1);

sino

LblPosition.Text = string.Format("{0:D2}:{1:D2}", Minute, Second-1);

Fin Si

Fin Método

Método button2\_Click(sender, EventArgs e)

Llamar CleanPlaylist;

Si play es verdadero entonces

Llamar Pausar;

Llamar Reset;

FinSi

Usar OpenFileDialog openFileDialog;

openFileDialog.Filter = "Archivos multimedia|\*.asf;\*.wma;\*.wmv;\*.wm;\*.asx;\*.wax;\*.wvx;\*.wmx;\*.wpl;\*.dvr-ms;\*.wmd;\*.avi;\*.mpg;\*.mpeg;\*.m1v;\*.mp2;\*.mp3;\*.mpa;\*.mpe;\*.m3u;\*.mid;\*.midi;\*.rmi;\*.aif;\*.aifc;\*.aiff;\*.au;\*.snd;\*.wav;\*.cda;\*.ivf;\*.wmz;\*.wms;\*.mov;\*.m4a;\*.mp4;\*.m4v;\*.mp4v;\*.3g2;\*.3gp2;\*.3gp;\*.3gpp;\*.aac;\*.adt;\*.adts;\*.m2ts;\*.mkv";

Si openFileDialog.ShowDialog es DialogResult.OK entonces

Llamar Reset;

Ruta = openFileDialog.FileName;

Player.URL = Ruta;

Player.Visible = verdadero;

Llamar Play;

Llamar SetRecentMedia(Ruta);

BtnPlay.Select;

FinSi

FinUsar

Llamar hideSubMenu;

FinMétodo

Método button3\_Click(sender, EventArgs e)

Llamar CleanPlaylist;

Si play es verdadero entonces

Llamar Pausar;

Llamar Reset;

FinSi

Llamar openChildForm(new Folder(this))

Llamar hideSubMenu;

FinMétodo

Método button4\_Click(sender, EventArgs e)

Llamar hideSubMenu;

FinMétodo

Método button5\_Click(sender, EventArgs e)

Llamar CleanPlaylist;

Si play es verdadero entonces

Llamar Pausar;

position = 0;

MTBPosition.Value = 0;

LblDuration.Text = "00:00:00";

LblPosition.Text = "00:00:00";

FinSi

Llamar openChildForm(new RecentMedia(this))

Llamar hideSubMenu;

FinMétodo

Método btnPlaylist\_Click(sender, EventArgs e)

Llamar showSubMenu(panelPlaylistSubMenu);

FinMétodo

Método button8\_Click(sender, EventArgs e)

Llamar openChildForm(new NewPL(this));

Llamar hideSubMenu;

FinMétodo

Método button7\_Click(sender, EventArgs e)

Llamar openChildForm(new ManagePL(this));

Llamar hideSubMenu;

FinMétodo

Método button6\_Click(sender, EventArgs e)

Llamar hideSubMenu;

FinMétodo

Método button1\_Click(sender, EventArgs e)

Llamar hideSubMenu;

FinMétodo

Método btnTools\_Click(sender, EventArgs e)

Llamar showSubMenu(panelToolsSubMenu);

FinMétodo

Método button13\_Click(sender, EventArgs e)

Llamar openChildForm(new VideoDL(this));

Llamar hideSubMenu;

FinMétodo

Método button12\_Click(sender, EventArgs e)

Llamar openChildForm(new VideoConverter(this));

Llamar hideSubMenu;

FinMétodo

Método button10\_Click(sender, EventArgs e)

Llamar openChildForm(new AudioConverter(this));

Llamar hideSubMenu;

FinMétodo

Método button9\_Click(sender, EventArgs e)

Llamar hideSubMenu;

FinMétodo

Método btnHelp\_Click(sender, EventArgs e)

Llamar openChildForm(new Help());

Llamar hideSubMenu();

FinMétodo

Método btnExit\_Click(sender, EventArgs e)

Llamar Application.Exit();

FinMétodo

Método openChildForm(Form childForm)

Si activeForm no es nulo entonces

Llamar activeForm.Close();

FinSi

activeForm = childForm;

childForm.TopLevel = falso;

childForm.FormBorderStyle = FormBorderStyle.None;

childForm.Dock = DockStyle.Fill;

panelChildForm.Controls.Add(childForm);

panelChildForm.Tag = childForm;

childForm.BringToFront();

childForm.Show();

FinMétodo

Método Form1\_Load(sender, EventArgs e);

Player.Ctlcontrols.stop();

panelSideMenu.Visible = falso;

FinMétodo

Método pictureBox2\_Click(sender, EventArgs e)

MTBPosition.Enabled = verdadero;

Si ExternalInput es verdadero entonces

Llamar Reset();

Llamar ExternalPlay();

ExternalInput = falso;

FinSi

Si Player.URL es "" entonces

MessageBox.Show("There is no file to play");

Sino

Player.Visible = verdadero;

MTBPosition.Maximum = (int)Player.currentMedia.duration;

Si play es falso entonces

Llamar Play();

SinoSi play es verdadero entonces

Llamar Pausar();

FinSi

FinSi

BtnPlay.Select();

FinMétodo

Método BtnFullScreen\_Click(sender, EventArgs e)

Si ExternalPlaylist entonces

Si ShuffleMode es falso entonces

Llamar Mixxer(PlayList, 0);

PBShuffle.Image = Properties.Resources.Shuffle;

ShuffleMode = verdadero;

Sino

PBShuffle.Image = Properties.Resources.ShuffleOff;

ShuffleMode = falso;

FinSi

FinSi

FinMétodo

Método pictureBox8\_Click(sender, EventArgs e)

Si MTBVolume.Value no es 0 entonces

TempVol = MTBVolume.Value;

MTBVolume.Value = 0;

Player.settings.mute = verdadero;

LblVolumen.Text = MTBVolume.Value.ToString() + "%";

BtnVolumen.Image = Properties.Resources.SpeakerOff;

Sino

MTBVolume.Value = TempVol;

LblVolumen.Text = MTBVolume.Value.ToString() + "%";

Player.settings.mute = falso;

BtnVolumen.Image = Properties.Resources.Speaker;

FinSi

FinMétodo

Método BtnPlay\_Click(sender, EventArgs e)

MTBPosition.Enabled = verdadero;

Si ExternalInput es verdadero entonces

Llamar Reset();

Llamar ExternalPlay();

ExternalInput = falso;

FinSi

Si Player.URL es "" entonces

MessageBox.Show("No hay archivo a reproducir");

Sino

Player.Visible = verdadero;

MTBPosition.Maximum = (int)Player.currentMedia.duration;

Si play es falso entonces

Llamar Play();

SinoSi play es verdadero entonces

Llamar Pausar();

FinSi

FinSi

FinMétodo

Método Player\_ClickEvent(sender, AxWMPLib.\_WMPOCXEvents\_ClickEvent e)

BtnPlay.Select();

FinMétodo

Método PBAdelante\_Click(sender, EventArgs e)

position += 10;

Player.Ctlcontrols.currentPosition += 10;

Llamar SetTimeStuff();

FinMétodo

Método PBAtras\_Click(sender, EventArgs e)

position -= 10;

Player.Ctlcontrols.currentPosition -= 10;

Llamar SetTimeStuff();

FinMétodo

Método PBStop\_Click(sender, EventArgs e)

Player.Ctlcontrols.stop();

Llamar Reset();

FinMétodo

Método PBAdelante\_DoubleClick(sender, EventArgs e)

Si ExternalPlaylist es falso entonces

Player.Ctlcontrols.stop();

Llamar Reset();

Sino

Si ShuffleMode es verdadero entonces

Llamar NextElement(MixxedPlaylist);

Sino

Llamar NextElement(PlayList);

FinSi

FinSi

FinMétodo

Método PBAtras\_DoubleClick(sender, EventArgs e)

Si ExternalPlaylist es falso entonces

string x = Player.URL;

Player.Ctlcontrols.stop();

Reset();

Player.URL = x;

Sino

Si ShuffleMode es verdadero entonces

Llamar PreviusElement(MixxedPlaylist);

Sino

Llamar PreviusElement(PlayList);

FinSi

FinSi

FinMétodo

Método MTBPosition\_Scroll(sender, EventArgs e)

LblVolumen.Text = (MTBVolume.Value).ToString() + "%";

Player.settings.volume = MTBVolume.Value;

Si MTBVolume.Value no es 0 entonces

BtnVolumen.Image = Properties.Resources.Speaker;

Sino

BtnVolumen.Image = Properties.Resources.SpeakerOff;

FinSi

FinMétodo

Método MTBPosition\_MouseDown(sender, MouseEventArgs e)

Llamar Pausar();

FinMétodo

Método PBRepeat\_Click(sender, EventArgs e)

Si ExternalPlaylist es verdadero entonces

Si RepeatMode es verdadero entonces

RepeatMode = falso;

RepeatSMode = falso;

PBRepeat.Image = Properties.Resources.RepeatOff;

Sino

RepeatMode = verdadero;

RepeatSMode = falso;

PBRepeat.Image = Properties.Resources.Repeat;

FinSi

Sino

Si RepeatSMode es verdadero entonces

RepeatSMode = falso;

RepeatMode = falso;

PBRepeat.Image = Properties.Resources.RepeatOff;

Sino

RepeatSMode = verdadero;

RepeatMode = falso;

PBRepeat.Image = Properties.Resources.RepeatS;

FinSi

FinSi

FinMétodo

Método PBRepeat\_DoubleClick(sender, EventArgs e)

Si ExternalPlaylist es verdadero entonces

Si RepeatSMode es verdadero entonces

RepeatSMode = falso;

RepeatMode = falso;

PBRepeat.Image = Properties.Resources.RepeatOff;

Sino

RepeatSMode = verdadero;

RepeatMode = falso;

PBRepeat.Image = Properties.Resources.RepeatS;

FinSi

FinSi

FinMétodo

Método MTBPosition\_Scroll\_1(sender, EventArgs e)

FinMétodo

Método Form1\_MouseMove(sender, MouseEventArgs e)

Llamar SidePanelHider();

FinMétodo

Método panelChildForm\_MouseMove(sender, MouseEventArgs e)

Llamar SidePanelHider();

FinMétodo

Método Player\_MouseMoveEvent(sender, AxWMPLib.\_WMPOCXEvents\_MouseMoveEvent e)

Llamar SidePanelHider();

FinMétodo

Método panelPlayer\_MouseMove(sender, MouseEventArgs e)

var CursorPosition = this.PointToClient(Cursor.Position);

Si CursorPosition.X > 7 entonces

panelSideMenu.Visible = falso;

Sino

panelSideMenu.Visible = verdadero;

FinSi

FinMétodo

Método BtnCapture\_Click(sender, e)

intentar

si Path.GetExtension(Player.URL) es igual a ".mp4" entonces

Pausar();

declarar videoWidth, videoHeight, videoX, videoY como enteros;

declarar VideoRatioAspect como decimal igual a Player.currentMedia.imageSourceWidth / Player.currentMedia.imageSourceHeight;

declarar PlayerRatioAspect como decimal igual a Player.Width / Player.Height;

si PlayerRatioAspect > VideoRatioAspect entonces

videoHeight = Player.Height;

videoWidth = videoHeight \* VideoRatioAspect;

videoX = Player.PointToScreen(new Point()).X + (Player.Width - videoWidth) / 2;

videoY = Player.PointToScreen(new Point()).Y;

sino

videoWidth = Player.Width;

videoHeight = videoWidth / VideoRatioAspect;

videoX = Player.PointToScreen(new Point()).X;

videoY = Player.PointToScreen(new Point()).Y + (Player.Height - videoHeight) / 2;

fin del si

declarar bitmap como nuevo Bitmap(videoWidth, videoHeight);

declarar graphic como Graphics.FromImage(bitmap);

graphic.CopyFromScreen(videoX, videoY, 0, 0, bitmap.Size);

declarar folderBrowserDialog como nuevo FolderBrowserDialog();

si folderBrowserDialog.ShowDialog() es igual a DialogResult.OK entonces

declarar FolderSelected como cadena igual a folderBrowserDialog.SelectedPath;

declarar FullURL como cadena igual a Path.Combine(FolderSelected, "Capture.png");

declarar name como cadena igual a "Capture";

declarar ext como cadena igual a ".png";

declarar x como entero igual a 1;

mientras File.Exists(FullURL) hacer

declarar tempFileName como cadena igual a string.Format("{0}({1})", name , x++);

FullURL = Path.Combine(FolderSelected, tempFileName + ext);

fin del mientras

bitmap.Save(FullURL, System.Drawing.Imaging.ImageFormat.Png);

fin del si

fin del si

capturar ex como Exception

MessageBox.Show("There has been a problem saving the capture. Error: "+ex);

fin del capturar

fin del Método

procedimiento BtnExtractor\_Click(sender, e)

openChildForm(new AudioExtractor(this));

hideSubMenu();

fin del Método

Método MTBPosition\_MouseUp(sender, e)

Play();

Player.Ctlcontrols.currentPosition = MTBPosition.Value;

position = (int)Player.Ctlcontrols.currentPosition;

timer1.Start();

BtnPlay.Select();

SetTimeStuff();

fin del Método

Método timer1\_Tick(sender, e)

SetTimeStuff();

si position > (int)Player.currentMedia.duration y ExternalPlaylist entonces

si ShuffleMode entonces

si RepeatSMode entonces

Player.Ctlcontrols.stop();

Reset();

Play();

sino

NextElement(MixxedPlaylist);

fin del si

sino

si RepeatSMode entonces

Player.Ctlcontrols.stop();

Reset();

Play();

sino

NextElement(PlayList);

fin del si

fin del si

sino si position > (int)Player.currentMedia.duration entonces

si RepeatSMode entonces

Player.Ctlcontrols.stop();

Reset();

Play();

sino

timer1.Stop();

LblPosition.Text = "00:00:00";

position = 0;

MTBPosition.Value = 0;

play = false;

MTBPosition.Enabled = false;

PBPlay.Image = Properties.Resources.play2;

fin del si

fin del si

fin del Método

fin de la clase

Clase Folder

clase Folder extiende Form

declarar Principal como Interfaz;

declarar ElementSelected como cadena;

declarar SupportedFormats como arreglo de cadenas igual a { ".asf", "mkv", ".wma", ".wmv", ".wm", ".wmx", ".wpl", ".avi", ".mpg", ".mpeg", ".m1v", ".mp2", ".mp3", ".mpa", ".mpe", ".m3u", ".mid", ".midi", ".rmi", ".aif", ".aifc", ".aiff", ".au", ".snd", ".wav", ".cda", ".ivf", ".wmz", ".wms", ".mov", ".m4a", ".mp4", ".m4v", ".mp4v", ".3g2", ".3gp2", ".3gp", ".3gpp", ".aac", ".adt", ".adts", ".m2ts", ".flac" };

Método Folder(Principal)

InitializeComponent();

este.Principal = Principal;

DGVFolder.SelectionChanged += new EventHandler(DGVFolder\_SelectionChanged);

fin del Método

Método SendMedia()

Principal.SetRecentMedia(TBRuta.Text + "\\" + ElementSelected);

Principal.ExternalURL = TBRuta.Text + "\\" + ElementSelected;

Principal.ExternalInput = true;

este.Close();

fin del Método

Método button5\_Click(sender, e)

este.Close();

fin del Método

Método Files()

declarar FilesArray como arreglo de cadenas igual a Directory.GetFiles(TBRuta.Text);

para cada File en FilesArray hacer

si SupportedFormats contiene alguna cadena x tal que File termina con x entonces

declarar n como entero igual a DGVFolder.Rows.Add();

DGVFolder.Rows[n].Cells[0].Value = System.IO.Path.GetFileName(File);

fin del si

fin del para

fin del Método

Método BtnSearch\_Click(sender, e)

DGVFolder.Rows.Clear();

fileSystemWatcher1.Path = TBRuta.Text;

Files();

fin del Método

Método dataGridView1\_CellClick(sender, e)

BtnPlay.Enabled = true;

declarar n como entero igual a e.RowIndex;

si n no es igual a -1 entonces

ElementSelected = (cadena)DGVFolder.Rows[n].Cells[0].Value;

fin del si

fin del Método

Método BtnPlay\_Click(sender, e)

SendMedia();

fin del Método

Método TBRuta\_KeyDown(sender, e)

si e.KeyCode es igual a Keys.Enter entonces

DGVFolder.Rows.Clear();

fileSystemWatcher1.Path = TBRuta.Text;

Files();

fin del si

fin del Método

Método DGVFolder\_SelectionChanged(sender, e)

si DGVFolder.SelectedCells.Count es mayor a 0 entonces

BtnPlay.Enabled = true;

declarar n como entero igual a DGVFolder.SelectedCells[0].RowIndex;

si n no es igual a -1 entonces

ElementSelected = (cadena)DGVFolder.Rows[n].Cells[0].Value;

fin del si

fin del si

fin del Método

Método DGVFolder\_KeyDown(sender, e)

si e.KeyCode es igual a Keys.Enter entonces

SendMedia();

fin del si

fin del Método

Método Form2\_MouseMove(sender, e)

Principal.SidePanelHider2();

fin del si

fin del Método

Clase RecentMedia

clase RecentMedia extiende Form

declarar Principal como Interfaz;

declarar RecentMediaArray como arreglo de cadenas igual a new string [0];

declarar URLsArray como arreglo de cadenas igual a new string [0];

declarar ElementSelected como cadena;

método RecentMedia(Principal)

InitializeComponent();

este.Principal = Principal;

DGVRecentMedia.SelectionChanged += new EventHandler(DGVRecentMedia\_SelectionChanged);

fin del método

método SendMedia()

Principal.Reset();

Principal.ExternalURL = ElementSelected;

Principal.ExternalInput = true;

este.Close();

fin del método

método button5\_Click(sender, e)

este.Close();

fin del método

método AddMedia(NewRecentMedia, NewURL)

declarar TempElementsArray como arreglo de cadenas igual a new string[RecentMediaArray.Length+1];

Array.Copy(RecentMediaArray, TempElementsArray, RecentMediaArray.Length);

TempElementsArray[RecentMediaArray.Length] = NewRecentMedia;

RecentMediaArray = TempElementsArray;

declarar TempURLsArray como arreglo de cadenas igual a new string[URLsArray.Length + 1];

Array.Copy(URLsArray, TempURLsArray, URLsArray.Length);

TempURLsArray[URLsArray.Length] = NewURL;

URLsArray = TempURLsArray;

DGVRecentMedia.Rows.Clear();

para cada File en RecentMediaArray hacer

declarar n como entero igual a DGVRecentMedia.Rows.Add();

DGVRecentMedia.Rows[n].Cells[0].Value = File;

fin del para

fin del método

método Form3\_Load(sender, e)

para cada File en Principal.RecentURLs.Reverse() hacer

declarar n como entero igual a DGVRecentMedia.Rows.Add();

DGVRecentMedia.Rows[n].Cells[0].Value = System.IO.Path.GetFileName(File);

fin del para

fin del método

método DGVRecentMedia\_CellClick(sender, e)

fin del método

método BtnPlay\_Click(sender, e)

SendMedia();

fin del método

método BtnClean\_Click(sender, e)

este.Close();

Principal.CleanRecentMedia();

fin del método

método DGVRecentMedia\_SelectionChanged(sender, e)

si DGVRecentMedia.SelectedCells.Count > 0 entonces

BtnPlay.Enabled = true;

declarar n como entero igual a DGVRecentMedia.SelectedCells[0].RowIndex;

si n no es igual a -1 entonces

ElementSelected = Principal.RecentURLs[(Principal.RecentURLs.Length - 1) - n];

fin del si

fin del si

fin del método

método DGVRecentMedia\_KeyDown(sender, e)

si e.KeyCode es igual a Keys.Enter entonces

SendMedia();

fin del si

fin del método

método Form3\_MouseMove(sender, e)

Principal.SidePanelHider2();

fin del método

fin de la clase

Clase NewPL

Clase NewPL extiende Form

Principal es Interfaz;

NewPlaylist es arreglo de cadena[0];

PlaylistsFolder es cadena = Path.Combine(Application.StartupPath, "LYRA-PlayLists");

Método NewPL(Principal es Interfaz)

InitializeComponent();

this.Principal = Principal;

fin del método

Método CleanNewPlaylist()

ClearPlayList es arreglo de cadena[0];

NewPlaylist = ClearPlayList;

DGVNewPlaylist.Rows.Clear();

fin del método

Método BtnAdd\_Click(sender, e)

openFileDialog es OpenFileDialog;

openFileDialog.Filter = "Archivos multimedia|\*.asf;\*.wma;\*.wmv;\*.wm;\*.wmx;\*.wpl;\*.avi;\*.mpg;\*.mpeg;\*.m1v;\*.mp2;\*.mp3;\*.mpa;\*.mpe;\*.m3u;\*.mid;\*.midi;\*.rmi;\*.aif;\*.aifc;\*.aiff;\*.au;\*.snd;\*.wav;\*.cda;\*.ivf;\*.wmz;\*.wms;\*.mov;\*.m4a;\*.mp4;\*.m4v;\*.mp4v;\*.3g2;\*.3gp2;\*.3gp;\*.3gpp;\*.aac;\*.adt;\*.adts;\*.m2ts;\*.mkv;\*.flac";

openFileDialog.Multiselect = true;

si openFileDialog.ShowDialog() == DialogResult.OK

archivosSeleccionados es arreglo de cadena = openFileDialog.FileNames;

para cada File en archivosSeleccionados

si !NewPlaylist.Contains(File)

tempURLs es arreglo de cadena[NewPlaylist.Length + 1];

Array.Copy(NewPlaylist, tempURLs, NewPlaylist.Length);

tempURLs[tempURLs.Length - 1] = File;

NewPlaylist = tempURLs;

fin del si

fin del para

DGVNewPlaylist.Rows.Clear();

para cada File en NewPlaylist

n es entero = DGVNewPlaylist.Rows.Add();

DGVNewPlaylist.Rows[n].Cells[0].Value = System.IO.Path.GetFileName(File);

fin del para

fin del si

fin del método

Método BtnClean\_Click(sender, e)

CleanNewPlaylist();

fin del método

Método BtnSave\_Click(sender, e)

PlayList es cadena = Path.Combine(PlaylistsFolder, TBName.Text+".LYRA");

si TBName.Text.Equals(" ") o TBName.Text.Equals("")

PlayList = Path.Combine(PlaylistsFolder, "PlayList.LYRA");

fin del si

si !Directory.Exists(PlaylistsFolder)

Directory.CreateDirectory(PlaylistsFolder);

fin del si

para cada Element en NewPlaylist

OldPlayList es cadena = File.Exists(PlayList) ? File.ReadAllText(PlayList) : "";

File.WriteAllText(PlayList, OldPlayList + Element + "\n");

fin del para

MessageBox.Show(PlayList);

CleanNewPlaylist();

this.Close();

fin del método

Método button5\_Click(sender, e)

this.Close();

fin del método

Método TBName\_KeyPress(sender, e)

InvalidTyping es cadena = @"<>:""/\|?\*";

si InvalidTyping.Contains(e.KeyChar.ToString())

e.Handled = true;

fin del si

si TBName.Text.Equals(" ") o TBName.Text.Equals("")

BtnSave.Enabled = false;

sino

BtnSave.Enabled = true;

fin del si

fin del método

Método TBName\_Click(sender, e)

si TBName.Text.Equals("PlaylistName") o TBName.Text.Equals(" ") o TBName.Text.Equals("")

TBName.Text = "";

fin del si

fin del método

Método Form4\_MouseMove(sender, e)

Principal.SidePanelHider2();

fin del método

fin de la clase

Clase ManagePL

Clase ManagePL extiende Form

Principal es Interfaz;

ElementSelected es entero;

PlaylistToExport es cadena;

Playlists es arreglo de cadenas;

PlaylistSelected es arreglo de cadenas;

Método ManagePL(Principal es Interfaz)

Principal es Principal;

DGVPlaylist.SelectionChanged += DGVPlaylist\_SelectionChanged;

fin del Método

Método PrintDGV

para cada Archivo en PlaylistSelected hacer

n es DGVPlaylist.Rows.Add();

DGVPlaylist.Rows[n].Cells[0].Value es System.IO.Path.GetFileName(Archivo);

fin del para

fin del Método

Método OpenList

DGVPlaylist.Rows.Clear();

PlaylistToExport es System.IO.Path.GetFileNameWithoutExtension(Playlists[ElementSelected]);

Temp es File.ReadAllLines(Playlists[ElementSelected]);

PlaylistSelected es Temp;

PrintDGV();

BtnBack.Enabled es verdadero;

BtnOpen.Enabled es falso;

BtnDelete.Enabled es falso;

BtnPlay.Enabled es verdadero;

BtnExport.Enabled es verdadero;

fin del Método

Método RefreshDGV

DGVPlaylist.Rows.Clear();

si no Directory.Exists(NewPL.PlaylistsFolder) entonces

Directory.CreateDirectory(NewPL.PlaylistsFolder);

fin del si

Temp es Directory.GetFiles(NewPL.PlaylistsFolder);

Playlists es Temp;

para cada Archivo en Playlists hacer

si Archivo.EndsWith(".LYRA") entonces

n es DGVPlaylist.Rows.Add();

DGVPlaylist.Rows[n].Cells[0].Value es System.IO.Path.GetFileNameWithoutExtension(Archivo);

fin del si

fin del para

fin del Método

Método SendMedia

Principal.PlayList es PlaylistSelected;

Principal.ExternalPlaylist es verdadero;

Principal.PlaylistPlay();

Cerrar();

fin del Método

Método button5\_Click

Cerrar();

fin del Método

Método Form5\_Load

RefreshDGV();

fin del Método

Método BtnOpen\_Click

si Playlists.Length > 0 entonces

OpenList();

sino

MessageBox.Show("There is no PlayLists to show");

fin del si

fin del Método

Método BtnBack\_Click

RefreshDGV();

BtnOpen.Enabled es verdadero;

BtnDelete.Enabled es verdadero;

BtnBack.Enabled es falso;

BtnExport.Enabled es falso;

fin del Método

fin del Método

Método DGVPlaylist\_SelectionChanged

si DGVPlaylist.SelectedCells.Count > 0 entonces

si BtnBack.Enabled es falso entonces

BtnOpen.Enabled es verdadero;

BtnDelete.Enabled es verdadero;

sino

BtnOpen.Enabled es falso;

BtnDelete.Enabled es falso;

fin del si

ElementeSelected es DGVPlaylist.SelectedCells[0].RowIndex;

fin del si

fin del Método

Método DGVPlaylist\_KeyDown

si e.KeyCode es Keys.Enter entonces

si BtnOpen.Enabled es verdadero entonces

OpenList();

fin del si

fin del si

fin del Método

Método BtnPlay\_Click

SendMedia();

fin del Método

Método BtnDelete\_Click

si DGVPlaylist.SelectedCells.Count > 0 entonces

x es DGVPlaylist.SelectedCells[0].RowIndex;

si x no es -1 entonces

File.Delete(Playlists[x]);

DGVPlaylist.Rows.Clear();

RefreshDGV();

BtnOpen.Enabled es falso;

BtnDelete.Enabled es falso;

BtnBack.Enabled es falso;

fin del si

fin del si

fin del Método

Método BtnClearAll\_Click

para i desde 0 hasta Playlists.Length hacer

File.Delete(Playlists[i]);

fin del para

RefreshDGV();

fin del Método

Método BtnExport\_Click

Temp es File.ReadAllLines(Playlists[ElementSelected]);

PlaylistSelected es Temp;

FolderBrowserDialog folderBrowserDialog es nuevo FolderBrowserDialog();

si folderBrowserDialog.ShowDialog() es DialogResult.OK entonces

intentar

FolderSelected es folderBrowserDialog.SelectedPath;

DestinyFolder es Path.Combine(FolderSelected, PlaylistToExport);

si no Directory.Exists(DestinyFolder) entonces

Directory.CreateDirectory(DestinyFolder);

fin del si

para cada canción en PlaylistSelected hacer

file es System.IO.Path.GetFileName(canción);

Destiny es Path.Combine(DestinyFolder, file);

File.Copy(canción, Destiny);

fin del para

MessageBox.Show("Your Playlist has been exported succesfully");

capturar ex es Exception

MessageBox.Show("There has been a problem with the export, try again or restart the program. You also can check if it is all right with your folders. " + "Error: " + ex);

fin del intentar

fin del si

fin del Método

Método Form5\_MouseMove

Principal.SidePanelHider2();

fin del Método

fin de la clase