PROYECTO PRÁCTICO NO.1

PARTY MIX

24 OCTUBRE 2021

UNIVERSIDAD RAFAEL LANDÍVAR

Creado por: JULIO ANTHONY ENGELS RUIZ COTO - 1284719

Listen on Spotify

I. INTRODUCCIÓN

Debido a la pandemia del COVID 19 realizó un cambio drástico en la metodología de cursos en línea como presenciales, se ha visto que muchos estudiantes han empezado a perder interés en las mismas. Sin embargo, se ha percatado que la música ha ayudado con el desinterés, permitiendo acompañar a los estudiantes en el transcurso de cada curso, reproducir música que ayuda a su estado de ánimo. Por lo anterior, como futuros profesionales, se les ha proporcionado la tarea de crear un programa que permita a los estudiantes crear una playlist de música para reproducirla durante sus trabajos en grupo, individuales y principalmente en los proyectos de programación.

Las pilas y colas son herramientas que se utilizan día con día, se ven presentes en los supermercados, en el tráfico y finalmente en las aplicaciones de uso diario. El concepto de pila es organizar elementos encima de otro, para remover un elemento solo es permitido sacar el último elemento ingresado (en la cima). El concepto de cola es colocar elementos en forma consecutiva, para remover un elemento es permitido sacar solamente el primer elemento ingresado (en el fondo).

II. ANÁLISIS

II.I ENTRADAS

La primera entrada es el archivo CSV que a continuacion se muestra el codigo:

```
#include "CsvReader.h"

CsvReader::CsvReader(std::string path)
{
    //para la lectura del csv
    setlocale(LC_ALL, "spanish"); // ANSI Codification
    content = "";
    cursor = 0;

    std::ifstream file(path);

    std::string data;
    while(std::getline(file, data))
        content += data + ",";
    file.close();
}
// content = "cancion1, cancion2, cancion3, cancion4"
bool CsvReader::Read(std::string& data)
{
    data = "";//vacia la variable de entrada

    if (cursor >= content.length())
        return false;//por si no encuentra nada en content

    while (true)
    {
        if (cursor >= content.length())//ha llegado al final de la cadena
            return true;
        if (content[cursor] == ',') {//si se encuentra una coma
            cursor++;
            return true;
        }
        data += content[cursor];//agrega la letra a la data actual
        cursor++;//recorre cada letra del arreglo de nombres
    }
}
```

La segunda entrada son los cuadros de textos donde se ingresa el nombre y canción en la fila de reproducción a continuación se muestra un pedazo del codigo implementado:

```
Void btnAgregar_Click(Object^ sender, EventArgs^ e)
{
    try
    {
        string nombre = msclr::interop::marshal_as<std::string>(txtNombre->Text);//para que el usuario agregue el nombre de la cancion
        string artista = msclr::interop::marshal_as<std::string>(txtArtista->Text);//para que el usuario agregue el artista de la cancion

        nombre = (nombre == "") ? "Cancion sin nombre" : nombre;//por si no tiene nombre se coloca como cancion sin nombre
        artista = (artista == "") ? "desconocido" : artista;//si el artista es desconocido

        nombre[0] = toupper(nombre[0]);//para validar que el usuario ingrese la primera letra en mayuscula ya que el ordenamiento se genera con
          mayusculas
        artista[0] = (artista != "desconocido") ? toupper(artista[0]) : artista[0];

        Cancion* cancion = new Cancion(nombre, artista);
        reproduccion->enqueue(cancion);//para poner en la cola de reproduccion

        txtNombre->Clear();
        txtArtista->Clear();

        LlenarDatos();
    }
    catch (Exception^ e)
    {
        MessageBox::Show("AGREGAR" + e->Message, "ERROR DETECTADO", MessageBoxButtons::OK, MessageBoxIcon::Error);
    }
}
```

II.II SALIDAS

Como salidas se tiene que cuando el usuario a terminado de usar la aplicación se exporta su playlist en un archivo CSV o en un archivo TXT.

```
Void btnGuardar_Click(Object^ sender, EventArgs^ e)
{
    try
    {
        //para poder exportar el archivo con la playlist actual
        saveFileDialog1->Filter = "CSV Files | *.csv | TXT Files| *.txt";
        if (saveFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)
        {
            CsvWriter* csv = new CsvWriter();
            Pila* temp = new Pila();
            while (playlist->stackempty() == false)
            {
                Cancion* cancion = playlist->pop();
                string info = cancion->ToString();
                csv->Write(info);
                temp->push(cancion);
            }

            string file = msclr::interop::marshal_as<std::string>(saveFileDialog1->FileName);
            csv->Save(file);

            while (temp->stackempty() == false)
            {
                playlist->push(temp->pop());
            }
        }
    }
    catch (Exception^ e)
    {
        MessageBox::Show("EXPORTACION DEFECTUOSA" + e->Message, "ERROR DETECTADO", MessageBoxButtons::OK, MessageBoxIcon::Error);
    }
}
```

```
#include "CsvWriter.h"

CsvWriter::CsvWriter()
{
    setlocale(LC_ALL, "spanish");
    data = "";
    newLine = true;
}
void CsvWriter::Write(std::string info)
{
    if (newLine == false)
        data += ",";
    data += info;
    newLine = false;
}
void CsvWriter::NextLine()
{
    data += "\n";
    newLine = true;
}
void CsvWriter::Save(std::string path)
{
    std::ofstream file(path);
    file << data;
    file.close();
}
```

Como otra salida del programa se tiene a la funcion llenar datos.

```
void LlenarDatos()
{
    lbxPlayList->Items->Clear();
    for (int i = 0; i < playlist->size(); i++)
        lbxPlayList->Items->Add(gcnew String(playlist->GetValue(i)->ToString().c_str()));//para acceder a la cancion se utiliza el ->getvalue(i)

    lbxReproduccion->Items->Clear();
    for (int i = 0; i < reproduccion->size(); i++)//para que recorra toda la reproduccion
        lbxReproduccion->Items->Add(gcnew String(reproduccion->GetValue(i)->ToString().c_str()));//para acceder a la cancion se utiliza el -
          >getvalue(i)

    if (actual == nullptr)
        lblCancion->Text = "ninguno";//a un inicio no hay cancion que se este escuchando
    else
        lblCancion->Text = gcnew String(actual->ToString().c_str());//para que la cancion que se este escuchando se imprima en el lblcancion
}
```

II.III PROCESOS

Como primer proceso se tiene el de reproducir una canción de la pila playlist que es el de hacerle pop a la pila playlist a continuacion dejo una pedazo del codigo utilizado.

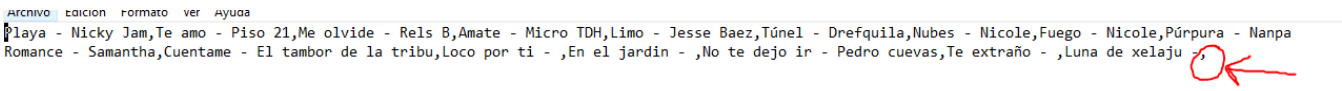
```
Void btnReproducir_Click(Object^ sender, EventArgs^ e)
{
    try
    {
        //se va quitando la cancion que el usuario le da a reproducir
        actual = playlist->pop();

        //para que la cola de reproduccion se coloque en la pila playlist cuando el usuario saque todas de la playlist
        if (playlist->size() == 0)
        {
            btnSincronizar_Click(sender, e);//mande a llamar a al funcion de sincronizar para no volverla a escribir ;3
        }

        LlenarDatos();
    }
    catch (Exception^ e)
    {
        MessageBox::Show("REPRODUCCION" + e->Message, "ERROR DETECTADO", MessageBoxButtons::OK, MessageBoxIcon::Error);
    }

}
```

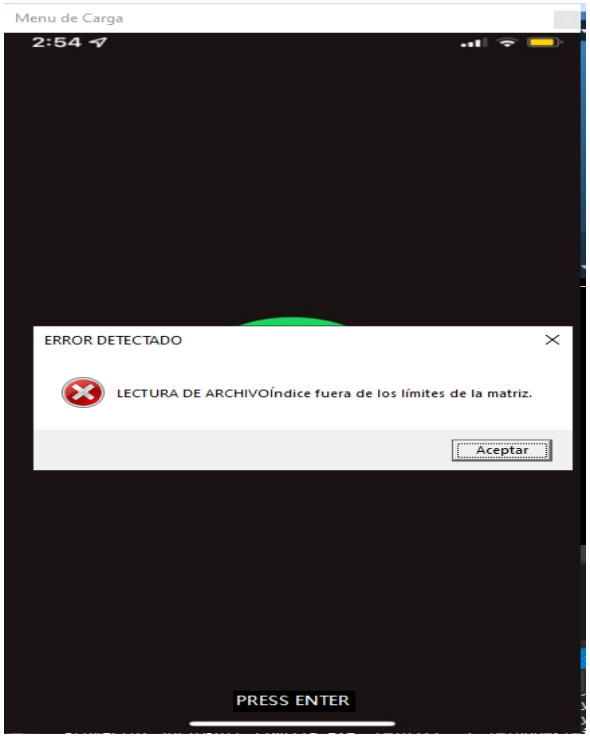
Como segundo proceso se tiene el de agregar una canción que es hacerle un encolar a la fila de reproducción que es la cola.

```
Void btnAgregar_Click(Object^ sender, EventArgs^ e)
{
    try
    {
        string nombre = msclr::interop::marshal_as<std::string>(txtNombre->Text);//para que el usuario agregue el nombre de la cancion
        string artista = msclr::interop::marshal_as<std::string>(txtArtista->Text);//para que el usuario agregue el artista de la cancion

        nombre = (nombre == "") ? "Cancion sin nombre" : nombre;//por si no tiene nombre se coloca como cancion sin nombre
        artista = (artista == "") ? "desconocido" : artista;//si el artista es desconocido

        nombre[0] = toupper(nombre[0]);//para validar que el usuario ingrese la primera letra en mayuscula ya que el ordenamiento se
          genera con mayusculas
        artista[0] = (artista != "desconocido") ? toupper(artista[0]) : artista[0];

        Cancion* cancion = new Cancion(nombre, artista);
        reproduccion->enqueue(cancion);//para poner en la cola de reproduccion

        txtNombre->Clear();
        txtArtista->Clear();

        LlenarDatos();
    }
    catch (Exception^ e)
    {
        MessageBox::Show("AGREGAR" + e->Message, "ERROR DETECTADO", MessageBoxButtons::OK, MessageBoxIcon::Error);
    }

}
```

II.IV RESTRICCIONES

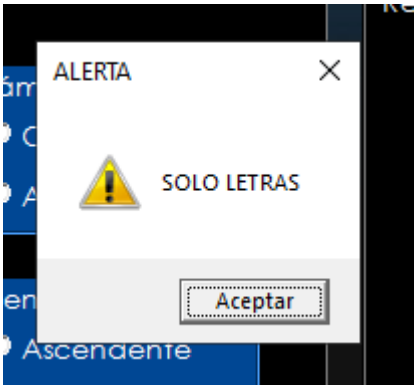
Como restricciones se tiene que si el nombre de las canciones al final de la lista en el archivo CSV se deja con una coma se rompera el formato de ingreso ya que lo recomendable es dejarlo sin coma de lo contrario le aparecera una ventana emergente.





Como otra restricción es que al momento de ingresar una nueva canción con su respectivo artista en al fila de reproducción el usuario debe de ingresar texto para que logre funcionar de lo contrario le aparecera una ventana emergente el cual no lo dejara continuar se muestra a continuación el codigo utilizado para validar esta restricción y con su respectiva alerta.

```
private: Void txtNombre_KeyPress(Object^ sender,KeyPressEventArgs^ e) {
    //validacion de solo letras segun la tabla ascii del 65 al 122 se tomo como letra sea mayuscula o minuscula
    if ((e->KeyChar >= 33 && e->KeyChar <= 64) || (e->KeyChar >= 91 && e->KeyChar <= 96) || (e->KeyChar >= 123 && e->KeyChar <= 255))
    {
        MessageBox::Show("SOLO LETRAS","ALERTA", MessageBoxButtons::OK, MessageBoxIcon::Exclamation);
        e->Handled = true;
        return;
    }
}
private: Void txtArtista_KeyPress(Object^ sender, :KeyPressEventArgs^ e) {
    if ((e->KeyChar >= 33 && e->KeyChar <= 64) || (e->KeyChar >= 91 && e->KeyChar <= 96) || (e->KeyChar >= 123 && e->KeyChar <= 255))
    {
        MessageBox::Show("SOLO LETRAS", "ALERTA", MessageBoxButtons::OK, MessageBoxIcon::Exclamation);
        e->Handled = true;
        return;
    }
```

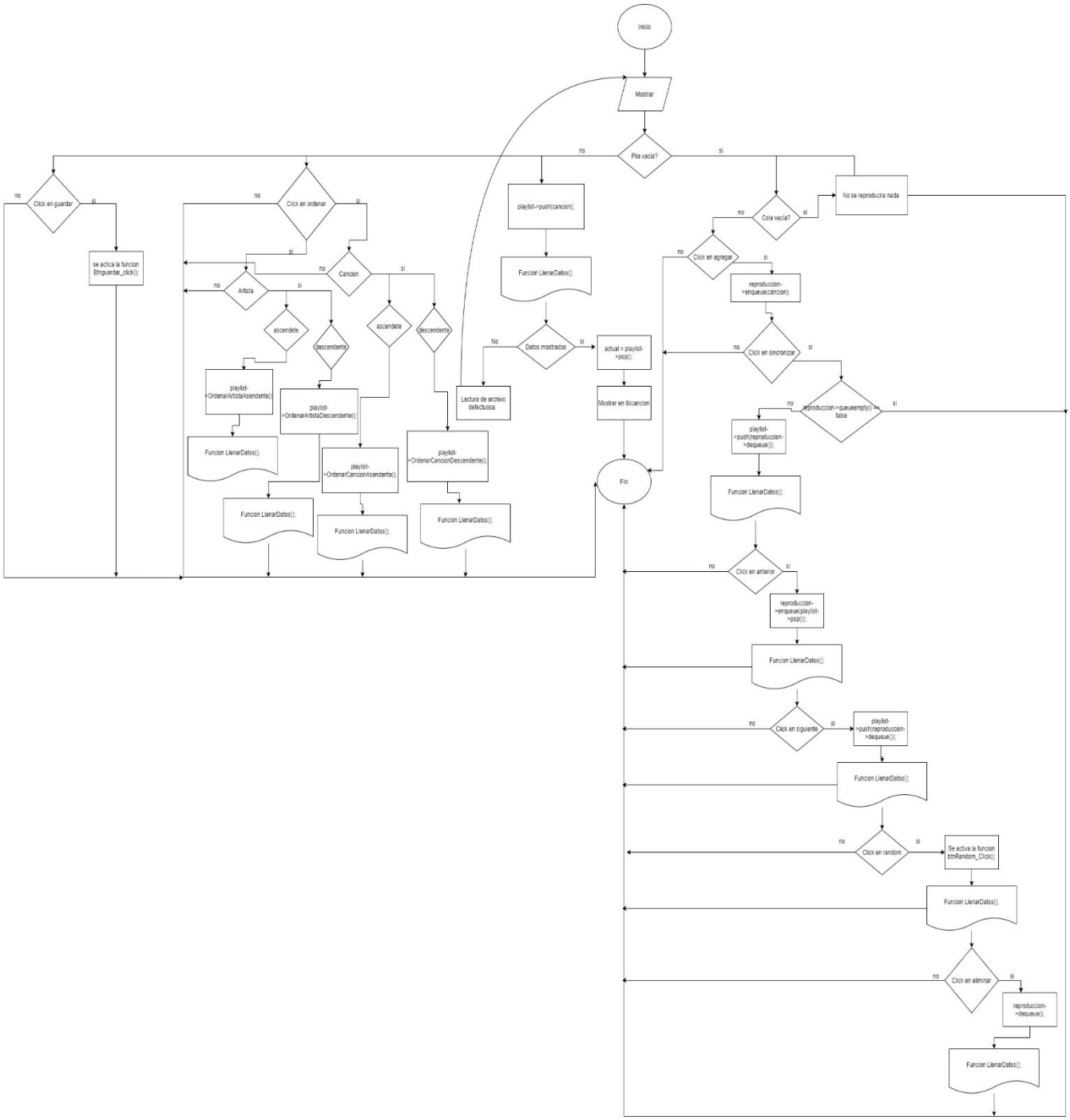


Como otro restricción se nos pidio validaramos para que todas las canciones posean un artista y si en dado caso no existira un artista definido se debe de mostrar la palabra desconocido, a continuación se muestra el codigo utilizado.

```
try
{
    CsvReader* csv = new CsvReader("songs.csv");
    playlist = new Pila();
    reproduccion = new Cola();
    actual = nullptr;

    string data;
    while (csv->Read(data)) //   "Playa - Nicky Jam"
    {
        String^ line = gcnew String(data.c_str());// c_str() extrae del string la ruta pero con formato const char
        //String recive ese tipo de dato que es más neutro para crear el objeto de Visual C++

        String^ Nombre = line->Split('-')[0]->Trim();//se obtiene el nombre
        String^ Artista = line->Split('-')[1]->Trim();//se obstiene el artista y con trim es una operación de cadena en la que se
          eliminan espacios en blanco adicionales del inicio y el final de una cadena

        //De (String^) a (string)
        //stackoverflowhttps://stackoverflow.com/questions/946813/c-cli-converting-from-systemstring-to-stdstring
        string nombre = msclr::interop::marshal_as<std::string>(Nombre);
        string artista = msclr::interop::marshal_as<std::string>(Artista);

        artista = (artista == "") ? "desconocido" : artista;
        //operador ternario es, si artista tiene un espacio en blanco devuelva desconocido si no
        //que devuelva el artista

        Cancion* cancion = new Cancion(nombre, artista);
        playlist->push(cancion);
    }

    LlenarDatos();
}
catch (Exception^ e)
{
    MessageBox::Show("LECTURA DE ARCHIVO" + e->Message, "ERROR DETECTADO", MessageBoxButtons::OK, MessageBoxIcon::Error);
}
```

III. DISEÑO

III.I DIAGRAMA DE FLUJO



IV. CONCLUSIONES

- La implementación de pilas y colas mediante listas enlazadas posibilita la representación eficiente de los datos en situaciones donde es necesario indicar el orden de procesamiento de los mismos y no es posible prever la cantidad de elementos a procesar por cuanto este tipo de representación permite crear y destruir variables dinámicamente.
- La solución del proyecto mostrado se realizo de una manera satisfactoria resolviendo el problema de una manera eficiente en mi punto de vista.
- Se logró ejecutar el programa mostrando en todo momento tanto el estado de la cola como la de la pila.
- Se validaron errores usando técnicas de programación adecuadas para esta resolución.
- Según como se vea, los reproductores de música pueden trabajar con pilas o colas agregando canciones al final de la lista y reproduciendo la primera.

V. RECOMENDACIONES

- Se recomienda tener los archivos CSV a la mano con la playlist ya cargada para tener una facilidad al ingresarla al programa.
- Se recomienda que el usuario lea el manual de usuario para poder agregar una canción a la cola de reproducción ya que se consideraron ingresar canciones solo de texto por lo que al ingresar números le saldra una excepción.
- Se recomienda manipular el programa relajado par no generar fallos del mismo.
- Una vez terminado de escuchar sus canciones si la cola de reproducción esta con datos esa cola pasara a la pila playlist para poder seguir escuchando sus canciones que agrego anteriormente.

VI. REFERENCIAS

VI.I BIBLIOGRÁFICAS

- *https://www.itsa.edu.co/docs/25-E-Arrieta-Manual-de-Estructura-de-Datos.pdf*

- *JOYANES Aguilar, Luis. Fundamentos de Programación: Algoritmos, estructuras de datos y objetos (4ta. Edición). Mc-Graw Hill. España, 2008.*

VI.II E-GRAFÍA

- *IOSTREAM: https://docs.microsoft.com/en-us/cpp/standard-library/iostream?view=msvc-160*

- *STRING: https://docs.microsoft.com/en-us/cpp/standard-library/string?view=msvc-160*

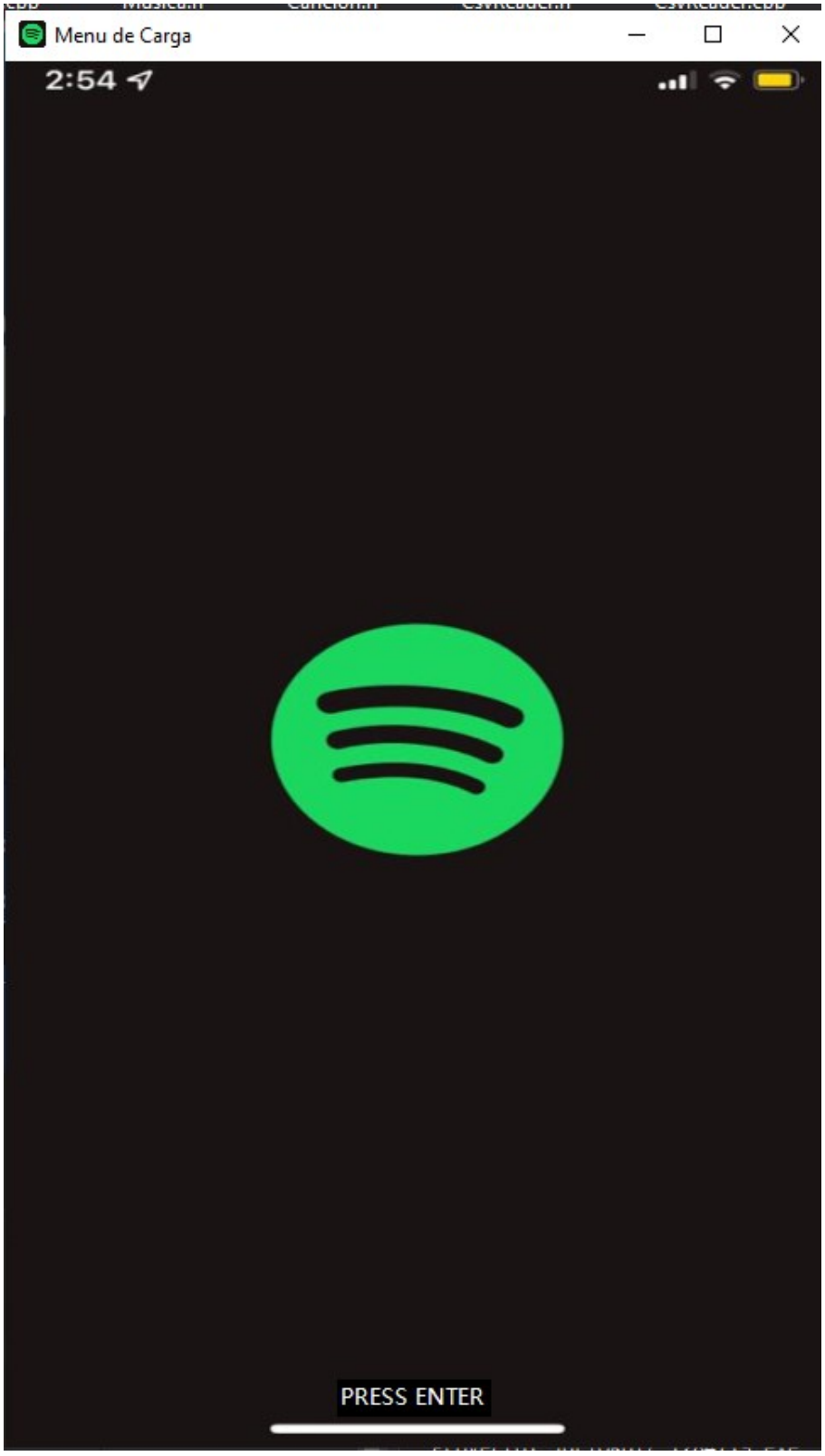
- *LOCALE: https://docs.microsoft.com/en-us/cpp/standard-library/locale?view=msvc-160*

- *FSTREAM: https://docs.microsoft.com/en-us/cpp/standard-library/fstream?view=msvc-160*

- *MSCLR\MARSHAL_CPPSTD.H: https://docs.microsoft.com/en-us/cpp/dotnet/overview-of-marshaling-in-cpp?view=msvc-160*

- *Conversión de C ++ / CLI de System :: String ^ a std :: string: https://stackoverflow.com/questions/946813/c-cli-converting-from-systemstring-to-stdstring*

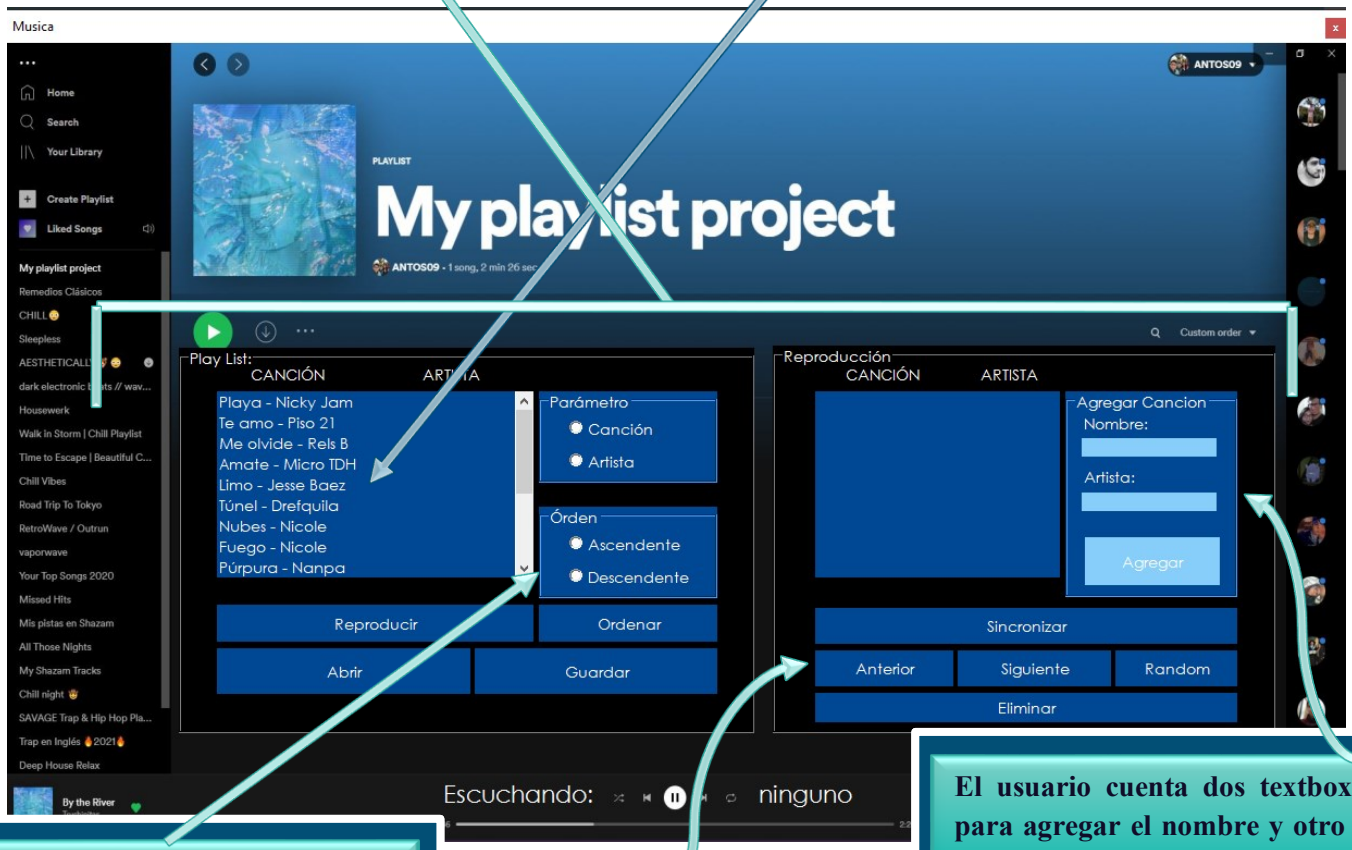
VII. ANEXOS



El usuario al momento de ejecutar el programa se encontrará con el siguiente formulario de inicio, el cual el usuario tiene que presionar la tecla ENTER para poder acceder a la aplicación principal.

El usuario luego de presionar la tecla ENTER se le desplegara el siguiente formulario.

El usuario ingresa su playlist canción y artista, por medio de un archivo CSV separado por comas el cual luego de ser leído el archivo se ingresa en la pila playlist. También cuenta con un boton de abrir para futuros archivos.



El usuario despues de importar su archivo se muestra una serie de grupos de cuadros en donde debe de seleccionar que parámetro quiere ordenar y que tipo de orden quiere si ascendente o descendente luego de escoger dicha selección cuenta con un boton llamado ordenar. También cuenta con la opción de reproducir que saca la primera de la pila playlist, se muestra abajo a la par de donde dice escuchando. Cuenta con un boton de guardar el cual guarda su playlist en un archivo CSV o TXT según escoja el usuario. También cuenta con un boton llamado abrir su función es acceder a futuros archivos que quiera visualizar el usuario en su playlist.

El usuario cuenta dos textbox uno para agregar el nombre y otro para agregar el artista (se valido para que solo ingrese texto) y si no ingresa nada se colocara canción a sin nombre y artista a desconocido.

El usuario cuenta con una sección de botones los cuales el botón de sincronizar pasa la cola de reproducción a la pila playlist, el botón anterior hace un pop a la playlist y se encola en la cola de reproducción, el botón siguiente hace que se desencole la primera de la cola de reproducción y se le de un push a la pila plylist, en el botón que dice random saca una canción aleatoria de la cola de reproduccióny se coloca en la pila playlist, y por ultimo cuenta con un boton de eliminar que elimina la primera de la cola.