

YouTify

En este documento se brindará una breve explicación del propósito y funcionamiento de la aplicación YouTify, desarrollada como segundo ejercicio integrador de la materia Laboratorio II.

La misma cumple el propósito de integrar diversas plataformas de streaming de música (en este caso, Youtube Music y Spotify, aunque podría extenderse) y le permite al usuario llevar a cabo operaciones dentro de cada plataforma como crear playlists, agregar canciones a playlists existentes, y también habilita la transferencia de una playlist, y sus canciones, a la otra plataforma.


Para comenzar a utilizar la aplicación, es necesario que el usuario se identifique en la pantalla de Login con un usuario y una clave.





Ingrese su cuenta

Login
Sign Up

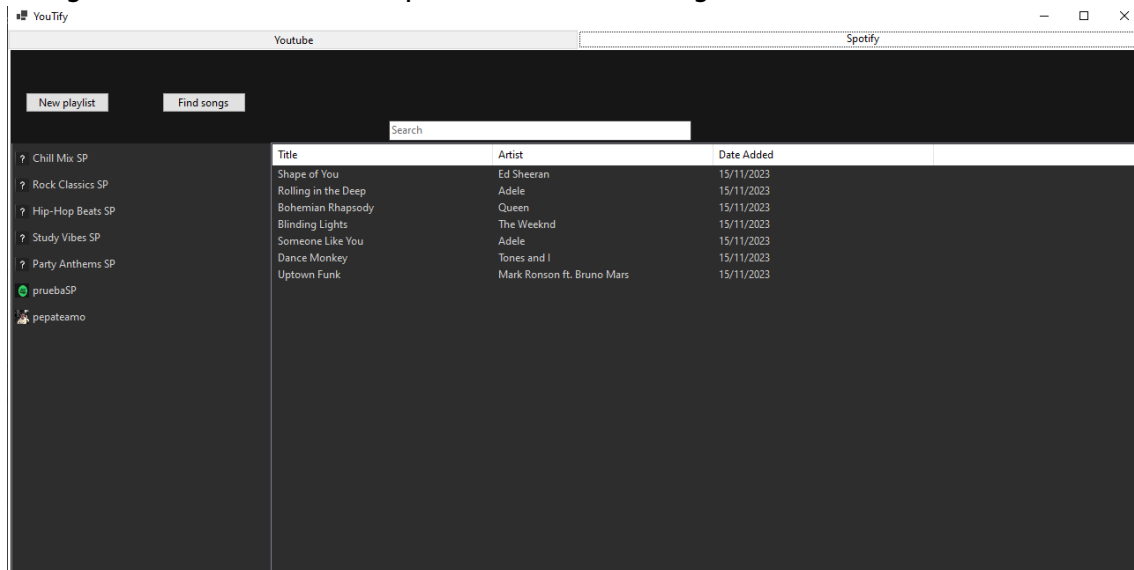


Ingrese su cuenta

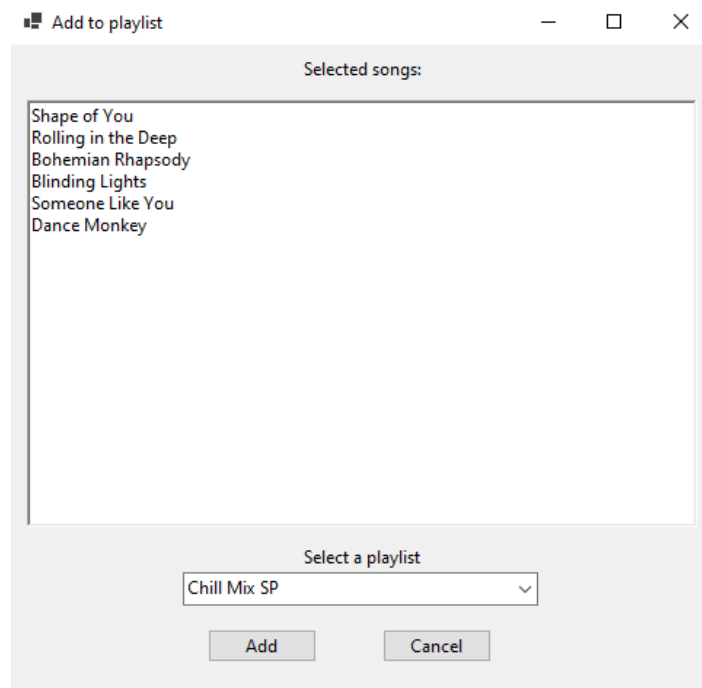
Login
Sign Up

Una vez iniciada la sesión en una plataforma, el usuario podrá llevar a cabo todas las operaciones que no involucren conexión entre plataformas (es decir, crear y modificar playlists, etc).

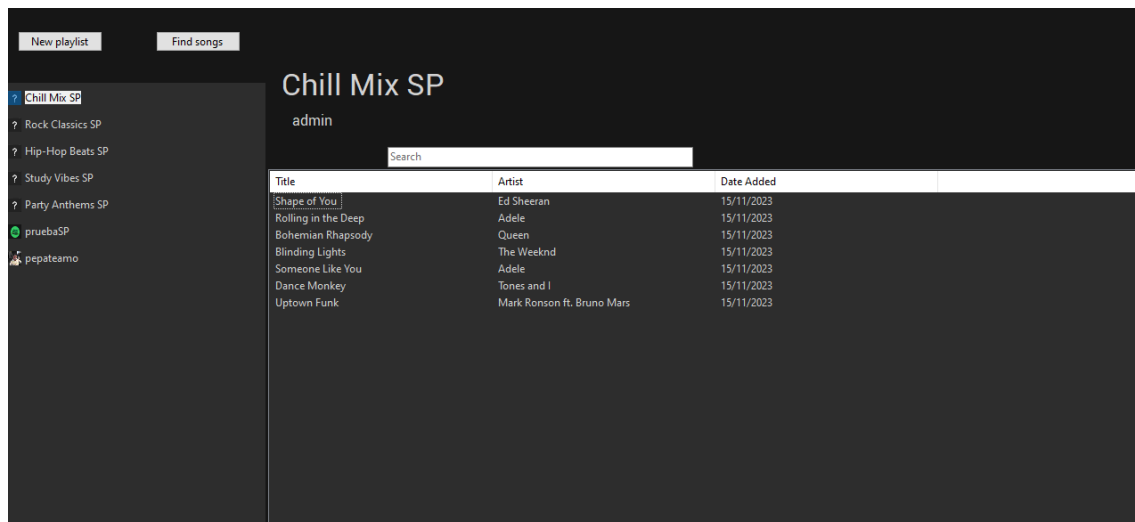
Al ingresar, se visualizará una pantalla similar a la siguiente:



En el panel izquierdo, se visualizarán las playlists creadas para el usuario actual. De no existir ninguna, se mostrará un mensaje indicando la situación. En el panel central de la pantalla, se podrán explorar todas las canciones disponibles para la plataforma actual. A través de selección multilínea y utilización del menú contextual (clic derecho), se podrán agregar las canciones seleccionadas a una playlist deseada a través de un nuevo formulario:



Al ejecutar un doble clic sobre una playlist específica, la misma se expandirá tomando el lugar del explorador de canciones, y mostrará una vista detallada de la playlist seleccionada, la cual contiene las canciones correspondientes para dicha lista:



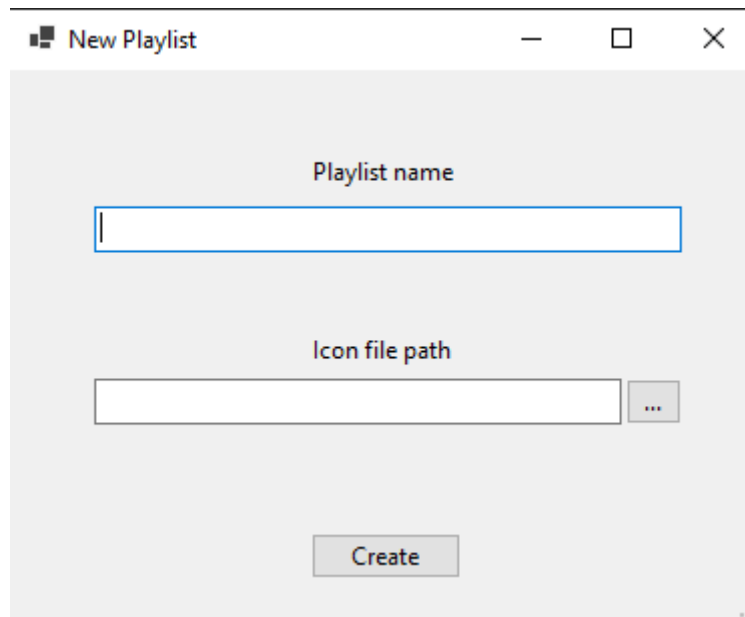
(doble clic sobre "Chill Mix SP")

Para transferir las canciones de una playlist a otra de la plataforma opuesta, se puede utilizar el menú contextual sobre las playlist y ejecutar clic sobre "Transfer to". Esto mostrará un nuevo formulario que permite seleccionar la playlist destino, perteneciente a otra plataforma.

Tenga en cuenta que para que se permita la transferencia de playlists, el usuario debe haber iniciado sesión en ambas plataformas, y tener por lo menos una playlist en la plataforma destino.

Si se quisiera volver al explorador global de canciones, basta con dar clic en el botón "Find songs".

Si se quiere agregar una nueva playlist, esto se puede lograr a través del botón "New playlist". El mismo disparará un nuevo formulario tipo modal donde se podrá ingresar información sobre la playlist a generar:



El campo "Playlist name" es obligatorio no se permitirá la creación de la playlist sin información ingresada. El campo "Icon file path", por otro lado, es opcional y se puede utilizar para especificar una imagen con la cual se mostrará la playlist al visualizarla en modo lista. Para seleccionar la imagen se puede pegar la ruta completa en la caja de texto o también hacer click en el botón de 3 puntos, el cual disparará un selector de archivos sobre el explorador de Windows y obtendrá automáticamente la ruta de la imagen seleccionada.

Una vez finalizado, se hace clic sobre el botón "Create" y la playlist aparecerá en la lista.

Índice de temas requeridos:

- Excepciones:

Durante la ejecución del programa, múltiples errores pueden surgir los cuales provocarán excepciones e interrupciones en el flujo del código. Todas las excepciones son atrapadas y serán, en última instancia, manejadas por o a través del formulario principal para informar al usuario de la situación sin cortar la ejecución de la aplicación. Un ejemplo de esto se encuentra en la clase `LoginPanel`, en los métodos de `Login` y `Signup`, los cuales atraparán potenciales excepciones de ingreso de datos, y las manejarán para que sean visualizadas por el usuario.

- Métodos de extensión:

Se implementan métodos de extensión en el proyecto de Windows Forms, para las clases "Song" y "Playlist". Estos métodos se utilizan para transformar un

objeto de estos tipos a un tipo "ListViewItem", utilizado por el control ListView, para mostrar dichos objetos en formato lista. Dichos métodos están implementados en la clase "Extensions"

- Pruebas unitarias:

Se implementó una clase de unit test para verificar el funcionamiento de operaciones como creación de playlists en ambas plataformas, así como verificación de error al intentar crear una playlist duplicada. Se encuentran en "Tests.PlaylistTests"

```
[TestMethod]
public async void Test_CreateYoutubePlaylist_OK()
{
    // Arrange
    User user = new User("testUser");
    Playlist playlist = new Playlist("1", "playlist1", "", user);

    // Act
    await PlaylistRepository.CreatePlaylist(playlist, EPlatform.Youtube);
    var playlists = await PlaylistRepository.GetPlaylists(user, EPlatform.Youtube);

    // Assert
    Assert.IsTrue(playlists.Count > 0);
}

[TestMethod]
public async void Test_DuplicateYoutubePlaylists_Error()
{
    // Arrange
    User user = new User("testUser");
    Playlist playlist = new Playlist("1", "playlist1", "", user);

    // Act
    await PlaylistRepository.CreatePlaylist(playlist, EPlatform.Youtube);

    // Assert
    await Assert.ThrowsExceptionAsync<EDatabaseInsertError>(async () => await PlaylistRepository.CreatePlaylist(playlist, EPlatform.Youtube));
}
```

- Generics:

Utilicé generics en el proyecto Entities, más específicamente en el método "DatabaseHelper.ExecuteSelectQuery()". El propósito de los genéricos en esta funcionalidad es para generalizar la lógica de obtención de datos a partir de una base de datos, especificando que la función puede ser utilizada para cualquier tipo siempre y cuando implemente la interfaz "IEntity". Con esto, logro instanciar todos los registros obtenidos de la base de datos a su tipo adecuado para utilización en el sistema, sin importar la lógica de instanciación de cada clase.

- Interfaces:

Como fue indicado anteriormente, se utiliza la interfaz "IEntity" dentro del proyecto Entities para especificar que funcionalidades deberían cumplir todas las entidades que forman parte del modelo del sistema. Nótese que con entidades, me refiero a clases cuya información es dinámica y persistente, y se obtendrán de algún medio de persistencia al iniciar la ejecución del programa.

Dichas entidades serán capaces de instanciarse a sí mismas a través de un objeto "SqlDataReader", ya que es su responsabilidad conocer los campos que deben obtener y donde deberían asignarlos. También obliga a las entidades a implementar la obtención de campos y valores necesarios para ejecutar una consulta de inserción en una base de datos SQL.

- Archivos y serialización:

Se implementó serialización de objetos tipo Song dentro del control personalizado "SongSearch", a través de un botón de exportación que serializará y guardará en formato json las canciones presentes en dicho SongSearch. El archivo será guardado en el directorio de ejecución de la aplicación.

También se utiliza la lectura de archivos para obtener el connection string de la base de datos, guardado en formato txt en el mismo directorio. Esta lógica puede visualizarse en "Entities.FileHelper"

- SQL y bases de datos:

El modo principal de persistencia de datos en esta aplicación es a través de una base de datos SQL, la cual es accedida únicamente por la clase "DatabaseHelper", la cual es consumida por diferentes servicios y entidades para obtener y generar la información deseada. Cuenta con operaciones de inserción, lectura, edición y eliminación.

- Delegados y expresiones lambda:

Los delegados y expresiones lambda son utilizados múltiples veces a lo largo del sistema, para el caso de los delegados desarrollé algunos tipos propios para manejar las situaciones de Login y Signup, así como en el control "SongSearch" para definir la firma del evento lanzado al querer agregar una selección de canciones a una playlist.

Para las expresiones lambda, estas son utilizadas en diversos lugares, principalmente en operaciones LINQ sobre colecciones y listas de datos. Un ejemplo de esto se puede observar en la clase "SpotifyPlaylistService". Dentro de esta clase observamos la utilización de expresiones lambda en funciones "Select" ejecutadas sobre una lista.

Otro ejemplo de estas expresiones, fuera de operaciones LINQ, es la utilización de funciones anónimas dentro de propiedades (Ej:

"frmAddToPlaylist.SelectedPlaylist"). Sin embargo, intenté no cargar dichas propiedades con lógica dentro de su definición y, en cambio, opté por ejecutar llamadas a funciones específicas para cumplir la funcionalidad requerida.

- Concurrencia:

A pesar de que no implementé programación multi-hilo en el sentido de llevar a cabo una funcionalidad ejecutándola específicamente en un hilo diferente al principal, la concurrencia puede verse presente en múltiples ocasiones, principalmente en todo lo que refiere a la conexión con la base de datos. Por ejemplo, al finalizar el ingreso de datos en el formulario

"frmAddNewPlaylist", el formulario padre utilizará la información ingresada para crear una nueva playlist en la base de datos. Este proceso se lleva a cabo asincrónicamente, es decir en un hilo paralelo al hilo de ejecución del formulario. El usuario puede continuar operando el formulario a su disposición, y una vez finalizada la creación de la playlist, se obtendrán nuevamente todas

las listas disponibles y se visualizarán en la lista de playlists del usuario.
(evento "MainPage.btnNewPlaylist_Click")

- Eventos:

Como se indica anteriormente, múltiples eventos son utilizados para el correcto funcionamiento del formulario y la conexión con los datos. Además de eso, implemente algunos eventos propios para cubrir ciertas necesidades referidas al login, por ejemplo.

En el control de usuario "LoginPanel" se definen 2 propiedades de tipo EventHandler las cuales serán asignadas por el formulario principal, y serán utilizadas para dar aviso del intento de login del usuario.

Esto se puede ver en esa misma clase, en los eventos de controles nativos "btnLogin_Click" y "btnSignUp_Click". Estos eventos llamarán a sus propiedades asignadas y a sus propiedades asignadas y se ejecutarán funciones pertenecientes al formulario padre, el cual tomará las medidas necesarias para comenzar el proceso de iniciado de sesión.

La asignación de los EventHandlers para los LoginPanel's se lleva a cabo en el archivo "frmPrincipal.Designer.cs", en la asignación de propiedades de "lpSpotify" y "lpYoutube"