Universidad de los Andes > FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

ENTREGA 2 PROYECTO
Programación Orientada a Objetos

Equipo 13

Integrantes:

Carmen Benavente

Roberto Vergara

Luis Zeballos

Profesor:

Carlos Díaz

Andrés Howard

12 de mayo de 2020

CONCEPTUALIZACIÓN

Para esta entrega nos vimos complicados con muchas partes en la implementación del código. Tres días antes de la entrega preguntamos sobre un problema que estábamos teniendo con un Form de Windows y nos dijeron que no se podían utilizar, por lo que tuvimos que reiniciar gran parte del código e investigación pasando la reproducción de multimedia de un Form de Windows a la consola directo por medio de WindowsMediaPlayer y Process para abrir los archivos. Es por esta razón por la que hay varias partes del código que nos encantaría implementar para la siguiente entrega pero que nos encontramos en contra del tiempo en este minuto y no alcanzamos a implementarlas. Algunas de estas cosas es por ejemplo la implementación de la excepción “Exeption System.FormatException” que es cuando se recibe un formato que no es válido. Es por la falta de esa excepción y por una limpieza de código que se asume que el usuario a lo largo de todo el código es un usuario amigable y que cumple con los formatos pedidos en el console o menú. Dentro de las implementaciones que queremos realizar para la próxima entrega es el mostrar el archivo imagen de los álbumes, así como también la capacidad de seguir a personas de la clase Person y User. Dentro de las cosas que se asumen para nuestra plataforma es que el admin que agrega las canciones y películas ya conoce el código y por ende, la única manera de agregar archivos multimedia y objetos de clase Movies y Songs es solo por medio del código. Los factores innovadores que tenemos en nuestro programa será una lista de las canciones de las películas y por otro lado un atributo con el link a una página web sobre las distintas personas del programa. Este último no está implementado por un tema de falta de tiempo, se espera implementarlo para la próxima entrega. Lo otro que nos gustaría implementar y tenemos todo listo para hacerlo es darle un atributo Trailer a las Movies que sea distinto del archivo de Video, pero por un tema de peso del programa decidimos dejar simplemente un video de mp4 por cada película. Nuestro código es capaz de abrir los archivos multimedia de tipo .wav, .mpg, .mp3, .mp4, .avi, .wmp, etc. Pero utilizamos los más fáciles de descargar que son los .mp3 y .mp4 para la prueba del código. Como nuevo factor innovador de nuestro programa agregamos en cada una de las personas un link a una página web donde se muestra información importante de dicha persona, pero no logramos implementar el funcionamiento correcto de la apertura de la página web sin utilizar un Form de Windows por lo que lo eliminamos y pretendemos lograr implementarlo para la siguiente entrega. Por último, no pusimos el archivo de texto .gitignore dado que necesitábamos que se subiera al repositorio la carpeta .bin donde se encuentran todos los archivos multimedia de nuestra plataforma. A continuación se enlistan las distintas clases utilizadas con sus respectivos atributos y métodos con una pequeña descripción de cada uno.

Movies:

Corresponde a la clase que tendrá de objetos a las películas, con todas las especificaciones necesarias para que se pueda desarrollar el programa. Esta clase es Serializable.

Se puede notar del Uml que tanto Computer cómo Playlist necesitan información de películas, lógicamente para utilizarlas en sus propios métodos, los cuales necesitan los objetos movies..

Otro aspecto importante es que las personas relacionadas con la película se llaman de una clase Person, la cual tendrá la información básica de estás.

Atributos:

Title : string → Título de la película.

Director : Person → Director de la película, corresponde a un objeto de tipo Person.

Actors :List<Person> → Corresponde a una lista con todos los actores involucrados (lista de tipo Person).

Writer : Person → Escritor de la película, objeto de tipo Person.

Lenght : integer → Duración de la película en minutos.

Categories : List<string> → Lista que contiene todas las categorías a las que pertenece la película.

Studio : string → Estudio que realizó la película.

Description : string → Descripción de la película.

Year : string → Año publicación.

Resolution : string → Resolución disponible.

Memory : string → Corresponde al peso de la película (gb).

numReproductions : integer → Cantidad de reproducciones.

Rating : List<double> → Puntuaciones que ha recibido de la película.

RatingProm : double⟶ Promedio de puntuaciones de la película .

Trailer : string ⟶ Nombre del archivo de trailer.

Video : string ⟶ Nombre del archivo con la película .

MovieSongs : List<Songs> ⟶ Canciones involucradas en la película.

Min: integer ⟶ Corresponde al minuto en que queda la película, por defecto será 0 si nunca ha sido reproducida.

Métodos:

Play() ⟶ Método para reproducir la película.

PlayTrailer() ⟶ Método para reproducir el trailer.

MovieInformation() ⟶ Método para mostrar la información de la película.

Songs:

La clase Songs es bastante parecida a movies, básicamente sin un atributo video. Songs al igual que movies, crea objetos de tipo canción con sus respectivos atributos correspondientes. Songs es llamado por Computer, básicamente para el funcionamiento y por Playlist, para crear Playlist de tipo Songs. Esta clase es Serializable.

Al mismo tiempo, Songs llama a Artist (clase hija de Person), para identificar quien canta la canción, y también llama a la clase Album, la cual lógicamente indica el álbum de la canción.

Atributos:

player : WindowsMediaPlayer ⟶ clase utilizada para reproducir la canción.

Title : string ⟶ Título de la Canción.

Lenght : integer → Largo de la canción en minutos.

numReproduction : integer → número de reproducciones en la plataforma.

Rating : List<double> → Puntuaciones que recibe la Canción (de 1 a 5).

RatingProm : double → Promedio de puntuaciones.

Genre : List<strings> → Lista que contiene los géneros de la canción.

Lyrics : string → Letra.

Composer : Person → Compositor de la Canción de tipo Person.

Resolution : string → Calidad de la canción, del sonido.

Memory : string → Cantidad de memoria que utiliza la canción en el programa.

Artist : Artist → artista de la canción de tipo Artists.

Album : Album → album de la canción de tipo álbum.

Music : string → Nombre del archivo de música.

Writer : Person → Escritor de la canción de tipo Person.

Type : string → tipo de archivo de música.

Min: integer → Corresponde al min en qué queda la canción, por defecto será 0 a no ser que se reproduzca y no se termine.

Métodos:

Play() → Método para iniciar la canción.

SongsInformation() → Método para mostrar la información de la canción.

Files:

Files corresponderá a una clase estática que guarde listas de todas las películas, canciones, playlists y usuarios, básicamente corresponderá a nuestra Base de Datos.

Por lo tanto al ser nuestra "Base de Datos", entrega datos al admin, (que puede agregar canciones y películas), a Computer que contiene todos los métodos para el funcionamiento del programa, a Search que corresponde al buscador, y a Playlists que le entrega la lista de canciones y películas. Esta clase será Serializable por lo que será la encargada de guardar la información cuando el programa se cierre.

Atributos:

AllMovies : List<Movies> → Lista de Movies.

AllSongs : List<Songs> → Lista de Songs.

AllPlaylistsSongs : List<Playlists> → Playlists de canciones.

AllPlaylistsMovies : List<Playlists> → Playlists de películas.

AllSmartPlaylistsSongs : List<SmartPlaylist> → SmartPlaylist de canciones.

AllSmartPlaylistsMovies : List<SmartPlaylist> → SmartPlaylist de películas.

AllUsers : Dictionary<int,List<string>> → Base de datos de User en el programa.

Métodos:

Get_Lista_Usuarios() → Método para mostrar los usuarios.

ChangePasword() → Método para cambiar la contraseña.

AddUser() → Método para agregar a un usuario.

GetData() → Método para obtener los datos de un usuario.

LogIn() → Método para realizar el LogIn.

Search:

Search corresponde a una clase estática que nos funcionará como un buscador, es decir, que desde palabras claves, personas, características, etc.. nos encontrará las películas/canciones relacionadas.

Por lo mismo, Computer recibe la información que Search encuentra, y además como ya se mencionó anteriormente, busca la información que necesita de la clase Files.

Métodos:

Searching() → Método que muestra el menú del buscador y llama a las otras funciones de search.

QueryTitle() → Método que busca las películas con un título específico.

QueryPerson() → Método que busca las películas con una persona específica.

QueryCategories() → Método que busca las películas con una categoría específica.

QueryStudio() → Método que busca las películas con un estudio específico.

QueryYear() → Método que busca las películas con un año específico.

QueryRating() → Método que busca las películas con un rating mayor o igual a uno específico.

QueryNumRep() → Método que busca las películas con un número de reproducciones mayor o igual a uno específico.

QueryTitleS() → Método que busca las canciones con un título específico.

QueryPersonS() → Método que busca las canciones con una persona específica.

QueryGenreS() → Método que busca las canciones con un género específico.

QueryNumRepS() → Método que busca las canciones con un número de reproducciones mayor o igual a uno específico.

QueryRatingS() → Método que busca las canciones con un rating mayor o igual a uno específico.

QueryAlbumS() → Método que busca las canciones con un álbum específico.

QueryMovieSong() → Método que busca las canciones de una película en específico.

SeeNamesMovies() → Método que muestra las películas encontradas.

SeeNamesSongs() → Método que muestra las canciones encontradas.

User:

Corresponde a la clase del Usuario el cual guarda su información personal como email,contraseña.

Atributos:

Type : string → Tipo de usuario, puede ser Free, Premium o Admin.

UserName : string → ID del usuario.

Email: string → Email con el que se registro.

Password: string → Contraseña.

Privacy : boolean→Si es o no su cuenta privada.

Métodos:

AddSong() → Método que solo puede acceder un usuario de tipo admin que agrega canciones a la base de datos.

AddMovie() → Método que solo puede acceder un usuario de tipo admin que agrega películas a la base de datos.

OnEmailVerified() → Método que recibe el evento para verificar el mail.

OnEmailSent() → Método que recibe el evento para enviar mail.

Person:

Clase del tipo persona, representa a una persona con sus atributos. También es padre de la clase Artist. Esta clase es Serializable.

Atributos:

Name : string → Nombre de la persona.

Birthday : date → Fecha de nacimiento de la persona.

Genre : char → Género de la persona.

Link : string → Link a wikipedia o imdb con la info de la persona.

Artist:

Clase hija de Person, es la clase que representa al artista.

Atributos:

numReproduction : integer → numero de personas que escuchan el artista.

Álbum:

Clase que representa álbumes.

Atributos:

Name : string→Nombre del álbum.

Year : Date→Año de lanzamiento del álbum.

Image : string → Nombre de la imagen del álbum.

Artist : Artist→Artista del álbum.

Computer:

Corresponde a la clase que trabajará como sistema operativo de nuestro programa, será el encargado de hacer posibles todas las interacciones y será el controlador de todo. Es el simil de Espotifai del laboratorio 2. Todos los usuarios llaman a los métodos del computador, mientras que el computador llama a los métodos de las clases Files, Search, Movies, Songs, Playlist, SmartPlaylist y Console.

Métodos:

Searcher() → Método encargado de realizar una búsqueda, llama al método Search.Searching().

SearchTypeSongs() → Método encargado de buscar una canción por un criterio específico.

SearchTypeMovies() → Método encargado de buscar una película por un criterio específico.

Rate() → Método encargado de votar la calidad de dicho objeto, actúa sobre Rating de objeto Songs/Movies.

CreatePlaylist() → Método encargado de crear un objeto Playlist.

CreateSmartPlaylist() → Método encargado de crear un objeto SmartPlaylist.

AutomaticRep()à Método que reproduce todas las canciones de una lista.

Register() → Método que realiza el registro de un usuario.

ChangePassword() → Método encargado de editar la constraseña de un objeto User.

GenerateLink() → M+etodo que genera un link de verificación.

UpgradeFree() → Método que pasa un usuario de Free a Premium.

AddQueue() → Método encargado de agregar a la lista de cola de reproducción.

Playlist:

Esta clase es la encargada de generar listas con las canciones o películas que se le entregan. Recibe información de Computer y utiliza a Songs, Movies y Files.

Atributos:

List<Songs> → lista de canciones que puede o no estar vacía.

List<Movies> → lista de películas que puede o no estar vacía.

Name : string → nombre de la lista.

Pravicy : boolean → Ver si se agrega o no la playlist a las playlist públicas.

Type : string → Ve si es una playlist de películas o de canciones.

Métodos;

VerPlaylist() → Método entrega la información de la playlist.

Console1:

Esta clase estática es la encargada de llamar a la consola y recibir información desde ella. Recibe información de Computer y de Search.

Métodos:

InitialMessage() → Método que imprime un mensaje de bienvenida.

SecondMessage() → Método que muestra el menú.

SeeMovies() → Método que interactúa con las películas, las puede reproducir, valorar o agregar a una playlist.

SeeSongs() → Método que interactúa con las canciones, las puede reproducir, valorar o agregar a una playlist.

CreatePlaylist() → Método que crea una playlist, puede ser normal o smartplaylist.

ModifiedPlaylist() → Método encargado de modificar una playlist, como cambiar el nombre, sacar elementos o agregar elementos.

SeePlaylist() → Método que reproduce playlist.

SeeProgramPlaylist() à Método que ve todas las playlist que son públicas.

Search() → Método que llama al buscador de computer.

SeeMoviesSearch() → Método encargado de interactuar con los resultados de una búsqueda de películas.

SeeSongsSearch() → Método encargado de interactuar con los resultados de una búsqueda de canciones.

SmartPlaylist

Esta clase es la que agrega a la SmartPlaylist todas las películas o canciones que cumplan con el criterio especificado. Clase que se hereda de la clase Playlist.

Atributos:

Criterio : string → Criterio de la smartplaylist.

NameCriterio : string → Valor del criterio de la smarplaylist.

ChangePasswordEventArgs

Clase de base para enviar los eventos de ChangePasword.

MailSender

Clase que envía el correo, con los métodos para poder manejar los eventos.

RegisterEventArgs

Clase de base para enviar los eventos de Register.