**Consideraciones / Supuestos**

**Clases / Objetivos:**

**App**: Es la clase base del programa, posee la mayoría de los métodos para cumplir las funcionalidades del programa, así como también la mayoría de listas involucrada a este.

En primer lugar esta clase contiene todas las bases de datos necesarias para realizar diferentes acciones en la aplicación como las siguientes: DataBaseSongs (guarda objetos de la clase Songs), DataBaseMovies (guarda objetos de la clase Movies),DataBaseUsers (guarda objetos de la clase Perfil) y las smartlist , todas estas base de datos guardan objetos en una lista , a excepción de las smart list la cual las razones de por qué no usamos listas para estos lo explicaremos en las siguientes líneas.También para solucionar el tema de las Listas inteligentes, utilizaremos dos tipos de menu de busqueda: uno con todo lo relacionado con canciones, como lo es en el caso para los artistas y otro menú el cual será para la búsqueda de todo lo relacionado con películas (Trabajadores películas y películas).

Específicamente utilizaremos 2 menús solo para así acelerar el proceso de búsqueda con dos SmartList distintas para cada menú. Para definir las smartlist vimos que podría ser de utilidad la utilización de diccionarios, dado que con estos podemos recibir un string del criterio o criterios que ingrese el usuario, y además de una lista la cual contendrá lo mostrado anteriormente por otros usuarios (las canciones o películas según criterio si es que los hubo), y en caso de que exista el primer usuario de la aplicacion , no sera necesario estas Smartlist.Para almacenar estas smartlist utilizaremos 2, para el caso de las canciones el nombre de variable es DictionarySmartS , el cual como dijimos anteriormente recibirá un string del criterio y listas con las canciones que antes fueron mostradas por otros usuarios, y para el caso del menú de las películas utilizaremos una variable DictionarySmartM.

El primer método es SearchKeyword, recibe una palabra clave (Key) que es buscado entre la información de canciones (y videos) para filtrar las distintas canciones o películas en su busqueda, el metodo SearchPerson tambien recibe una key para filtrar cuáles personas deben estar (autores/actores) en cierta canción o video. El método SearchPersonChar recibe una Key para filtrar las características de las distintas personas que deben estar en al canción o video ( como por ej sexo, edad , nacionalidad,etc). El método Searchres filtra según el tamaño de la cancion o pelicula por mayor, menor o igual algun tamaño ( aun no tenemos la información de los distintos tipos de tamaños de los archivos). SearchCategory tambien como todas las anteriores recibe una Key(palabra clave) y se ocupa de filtrar las canciones y películas según sus distintas categorías.Searcheval filtra según la nota definida a esta. SearchSong sirve para, desde la DataBaseSongs, buscar una canción coincidente, a una palabra clave recibida por el método, así mismo, el método SearchPlaylistS busca una playlist en la DB, de acuerdo a una palabra clave recibida, y entrega su información.. Todos estos métodos anteriormente descritos retornan valores de tipo string para luego mostrarlo en consola para el usuario. También la clase APP posee un método llamado Addimage, el cual es una imagen para representar un disco o película, le pusimos que recibía un archivo de tipo string, ya que no sabemos aún qué tipo de variable es una imagen en el programa (por el momento recibirá el nombre del archivo, por ejemplo Nombre\_cancion.mp3). El método AddplaylistM sirve para agregar películas a la playlist de peliculas, recibe como parámetro de una película de tipo Movies, agrega a una lista este parámetro y no retorna nada, similar a este le método está AddplaylistS (agregar canciones a la playlists de canciones), recibe como parámetro una canción de tipo Songs, agrega a una lista este parámetro y no retorna nada, además el método EliminatePlaylist, que recibe una playlist de canciones y la elimina, no retorna nada. El método SongsDownloaded sirve para almacenar las canciones descargadas por el usuario, recibe como parámetro una canción de tipo Songs, y retorna un string para luego ser mostrado en consola. No existe un metodo para descargar peliculas. SongTime es el método para guardar el tiempo de reproducción, para que cuando se vuelva a abrir la aplicación, el archivo esté en el mismo tiempo que quedó el usuario, recibe una cancion tipo Songs y retorna un Int (entero) con el tiempo.

Y finalmente tenemos 2 métodos de Smarlitst , el cual guarda en un diccionario el criterio y una lista de películas o canciones , dependiendo de qué busqueda haya realizado el usuario (una smartlist para Canciones y otro para películas), este método retorna una lista la cual después puede ser usada a través del método ShowSmartlistConsole , el cual recibirá el criterio correspondiente a la búsqueda realizada por el usuario y el tipo de lista (que puede ser smartlist de movies o de canciones) , y luego retorna un string para mostrar en consola.Se agregaron las listas KeyWordSongs, KeyWordPlaylistS y KeyWordSingers, correlacionadas con SearchKeyword, creadas para almacenar en ellas, canciones, playlist de música y cantantes respectivamente debido a la búsqueda de una palabra clave, se van agregando objetos a sus listas y luego se entrega en forma de string toda la información encontrada en base a lo buscado.

Los atributos son llamados por distintos métodos de distintas clases, para poder organizarlos de acuerdo a distintas peticiones del usuario.

**Songs**: La clase songs posee todos los atributos informacionales de canciones a utilizar y mostrar en el programa, estos van de atributos triviales utilizados comúnmente:Name (nombre de la canción),álbum (nombre del álbum),género,duración,cantante,año de publicación, compositor, líricas, todos estos nombrados anteriormente son de tipo string .La clase además posee un atributos como RankingS el cual graba un promedio de las valorizaciones del usuario en un tipo de valor float, también tiene un atributo generado por la plataforma como número de reproducciones de la canciones(Reproductions), cantidad de usuarios que les gusta la canción (LikeS), estos dos últimos son de tipo int, para mostrarlos en números enteros(no puede haber media reproducción ni medio like), consideramos también un atributo Albumimage el cual graba el nombre del archivo ingresado por el administrador para luego almacenarlo en un objeto de la clase canción adecuado (por el momento no sabemos cómo almacenar un archivo de tipo mp3, por esto lo consideramos un string para luego llamar el archivo adecuado), pasa esto mismo con los atributos AudioQuality (calidad del audio), songSize(tamaño de la cancion), TypeFileS (tipo de archivo). También posee los atributos de Downloads y Candownload, el primero indica la cantidad de descargas que tiene la canción en tipo int (valores enteros), el segundo indica si la canción se puede descargar o no, el cual es tipo booleano (true or false, verdadero o falso).Los atributos son llamados por distintos métodos de distintas clases, para poder organizarlos de acuerdo a distintas peticiones del usuario. La clase posee un método InfoSong que entrega información importante de la canción, ya sea álbum a la que pertenece, Nombre de la canción, reproducciones, descargas, etc.

**Movies**:La clase movies posee todos los atributos informacionales de películas a mostrar y utilizar en el programa, estos van de atributos triviales utilizados comúnmente titulo,categoria,duracion,YearPublishM(año de publicación), Description(descripción de la pelicula),director, starring(protagonista),Studio(estudio).La clase además posee otros atributos extras para grabar los rating de las películas, para este atributo lo nombramos con el nombre de RankingM el cual graba un decimal del promedio de las valorizaciones de los usuarios hacia las películas en tipo de valor float para mostrar los decimales, .La clase también tiene atributos generado por la plataforma como número de reproducciones de la película (Reproductions), cantidad de usuarios que les gusta la película (LikeS), estos dos últimos son de tipo int, para mostrarlos en números enteros (no puede haber media reproducción ni medio like).También consideramos agregar un atributo MovieImage el cual grabará el nombre del archivo ingresado por el administrador para luego almacenarlo en la película adecuada (por el momento no sabemos cómo almacenar un archivo del tipo mp3 en una clase , es por esto que por el momento lo consideramos como un string para luego llamar el archivo adecuado), pasa esto mismo nombrado anteriormente con los atributos MovieQuality (calida de pelicula), MovieSize(tamaño pelicula), TyfileM(tipo de archivo de la pelicula), ademas posee el metodo InfoMovie, el cual retorna un string de forma coherente con la informacion de pelicula

**Profile**: En la clase Profile consideramos que para ingresar a un perfil común del consumidor (tanto para usuarios premium como gratuitos) y perfil del administrador, necesitamos como atributos el UserName(nombre de usuario), de tipo string, Password (contraseña), de tipo string también. Y para registrarse es necesario los atributos Email(mail), de tipo string , Password (contraseña), de tipo string y UserName (UserName será visto en la aplicación por otros usuarios , si es que este es un perfil público). Para solucionar la privacidad de los perfiles utilizaremos un atributo Privacy el cual es un atributo del tipo bool, consideraremos true como público y false como private (Para el perfil del administrador, será siempre privado).Para definir el concepto de si un usuario es premium utilizaremos un atributo llamado Premium, de tipo bool, el cual nosotros identificamos que un perfil es premium si es que es true, y false si no lo es. También tiene como atributo una lista de todas las playlist tanto de películas como canciones(PlaylistM O PlayslistS), que contienen valores de tipo (Playlist M o S dependiendo si es de películas o canciones), posee también una lista de las playlist favoritas (PlaylistFavM o PlaylistFavS) de películas y canciones de el usuario del tipo playslist que le corresponde. Además tiene como atributo UserFollowers y UserFollowing (seguidores del usuario y seguidores que sigue el usuario) los cuales son una lista con variables de tipo profile (porqué lo realizan usuarios), la cantidad de seguidores es un atributo derivado de solo utilizar un comando de mostrar el largo de la lista,por esto no le pusimos un atributo mas que guarde un entero con al cantidad. Posee atributos PlaylistFollowingS y PlaylistFollowingM ( playlist que sigue el usuario de canciones y películas respectivamente), estas tienen variables de tipo Playlist.Tambien posee como atributos SingerFollowing y workerMovieFollowing (cantantes seguidos y trabajadores de películas seguidos), los cuales son una lista de variables tipo Singer y workerMovie respectivamente.

Como métodos primero tiene el metodo PlayINQueue (agregar cola de reproduccion), recibe como parámetros el nombre de la playlist( para buscarla) y el nombre del perfil que identifica al usuario.

ChangePrivacyPlaylist con retorno vacío, este método sirve para cambiar la privacidad de la playlist, si un usuario es privado no puedo tener listas públicas. Por último posee el método VerifyLogin que sirve para verificar si la contraseña y nombre del usuario ingresada son correctas, esta recibe como parámetros Username y Password, y retorna True si el usuario es verificado y falso si no lo es.

Consideramos por el momento de agregar solo un administrador , el cual tendrá un email de ingreso: AdminSpotiflix y password: 123. Con estos datos se puede ingresar al menú del administrador , con el cual el admin tendrá el acceso para agregar canciones y películas.

**PlaylistS**: La clase PlaylistS posee como atributos el OwnerUser(nombre del usuario creador de la playlist) de tipo string, Name\_Playslist(nombre de la playlist de canciones) también de tipo string, la privacidad de la Playlist (PrivacyS) con tipo bool (true si la playlist es privada, en caso contrario false) ,FollowersPS el cual almacena los usuarios que siguen a esta lista de canciones, la cantidad de seguidores es un atributo derivado de solo utilizar un comando de mostrar el largo de la lista,por esto no le pusimos un atributo mas que guarde un entero con al cantidad. A su vez posee métodos para accionar con ella, como eliminar canciones de ésta (EliminateSong(Songs song)), InfoPlaylistS que retorna un string con el creador y el nombre de la playlist. E indudablemente la lista de canciones de la playlist del usuario, que contiene valores tipo Profile. Resumiendo, el objetivo principal de esta clase es almacenar al gusto del usuario sus canciones favoritas. En otras clases detallamos los métodos asociadas a esta.

**PlaylistM**: La clase PlaylistS posee como atributos OwnerUser(nombre del usuario creador de la playlist) de tipo string, Name\_Playslist(nombre de la playlist de canciones) también de tipo string, la privacidad de la Playlist (PrivacyM) con tipo bool (true si la playlist es privada, en caso contrario false), FollowersPM (la lista con los usuario de tipo profile, el cual contiene los seguidores de la Playlist) , la cantidad de seguidores es atributo derivado de solo utilizar un comando de mostrar el largo de la lista,por esto no le pusimos un atributo mas que guarde un entero con al cantidad. Y también indudablemente la lista de películas de la playlist del usuario, que contiene valores tipo Profile. Resumiendo, el objetivo principal de esta clase es almacenar al gusto del usuario sus películas favoritas, como también almacenar sus seguidores. En otras clases detallamos los métodos asociadas a esta.

**Singer**: La clase singer posee como atributos, toda la información del cantante, tal como su nombre, apellido y nombre artístico. También, posee un atributo para grabar los rating de los cantantes, este atributo lo nombramos RankingSinger, el cual graba un decimal del promedio de las valorizaciones de los usuarios hacia los cantantes en tipo de valor float para mostrar los decimales y permite el seguimiento de los usuarios hacia los cantantes .

**workerMovie**: La clase workerMovie posee como atributo, toda la información de la persona que trabaja en alguna película, tal como su nombre, apellido, nombre artístico y el rol que cumplen dentro de la misma. En adición, posee un atributo para grabar los rating de los trabajadores, este atributo lo nombramos RankingWM, el cual graba un decimal del promedio de las valorizaciones de los usuarios hacia los cantantes en tipo de valor float para mostrar los decimales. Además permite el seguimiento de los usuarios.

**Suposiciones:**

1. Existe sólo un administrador , y para ingresar a esta cuenta se necesita del Email: “AdminSpotiflix” y Contraseña: “123”
2. La calidad de las canciones son todas de 320kps (sabiendo se puede aproximar un poco la duración de la canción con el tamaño de este).
3. La calidad de las peliculas son todas 720p (sabiendo se puede aproximar un poco la duración de la película con el tamaño de este).
4. Existen 2 menús separado para el usuario normal (premium y no) , en un menú va estar todo lo relacionado con canciones, y en el otro todo lo relacionado con películas.
5. Las canciones solo considera un cantante (en otras palabras no cuenta cuando la canción es una banda y va agregando a cada uno de los integrantes en Singers, solo considera un cantante ).