Consideraciones / Supuestos

Clases / Objetivos:

Archivo de Proyecto [Models(Proyecto 2)]:

Este proyecto de nuestro programa contiene todas las definiciones de nuestros objetos necesarios para crear objetos de estas clases (Estas clases más que nada definen los atributos de estas clases, y si contienen métodos, son solo para ver la información entera del objeto). Las clases que conforman esta solución son las siguientes:

- ➔ **Movies:**

Atributos:

```
private string title;
private string category;
private float rankingM;
private string DatePublish;
private string description;
private string studio;
private double movieSize;
private string typeFileM;
private TimeSpan durationM;
private string FilePath;

//Candownload: Define si la película puede ser descargada o no
public bool Candownload;
public int Downloads;

// cast: Contiene una lista de la clase WorkerMovie en donde van a estar los
trabajadores de la película especifica.
private List<WorkerMovie> cast = new List<WorkerMovie>();
private int likesM;
private int reproductionsM;
```

Cambios c/r Entrega 2 [Atributos Movies]:

Lo cambiado con respecto a lo del proyecto 2 corresponde a los siguientes atributos:

```
private TimeSpan durationM;
private string FilePath;
```

Para durationM se cambio el tipo de variable de `int` → A `TimeSpan` para reservar de manera correcta la duración de la película.
Y también se agrego un nuevo atributo de FilePath , el cual almacena la nueva ubicación del archivo c/r a película (mp4 o avi) a un directorio ubicado en la carpeta debug.

Basta decir que para este atributo falta establecer un método de actualización del FilePath para que funcione para mas de una persona la aplicación, pero para esta entrega bastaba que funcionara para un computador.

➔ **PlaylistM**

Atributos:

```
// OwnerUser: Dueño de la playlist
o   public string OwnerUser;
o   public string Name_PlaylistM


//PrivacyM: Establece la privacidad de la lista, este atributo es necesario
para poder buscar listas (solo se pueden buscar las que están públicas,
basta en clarecer que, si el usuario dueño de la lista es público, esto
convierte a todas sus playlist a publicas también)

o   public bool PrivacyM

//TypeM: Establece de que tipo de archivo se está almacenando la playlist
Solo de este tipo de archivo se va a aceptar agregar a esta playlist.
o   public string TypeM;

//ListM : Almacena las películas en una playlistM (playlist de peliculas)
o   public List<Movies > ListM = new List<Movies > { };
```

Cambios c/r Entrega 2 [Atributos PlaylistM]:

No hubo cambios.

➔ **PlaylistS**

Atributos:

```
//Contiene las mismas definiciones que PlaylistM
o   public string OwnerUser;
o   public string Name_PlaylistS
o   public bool PrivacyS
o   public string type;
o   public List<Songs> ListS = new List<Songs> { };
```

Cambios c/r Entrega 2 [Atributos PlaylistS]:

No hubo cambios.

➔ **User**

Atributos:

```
o   public string Email;
o   public string UserName;
o   public string Password;
```

```
//Privacy: Privacidad del usuario (True es privado y False es publico)
se puede cambiar en el menú de configuración del usuario
  public bool Privacy;

// PlaylistS: Guarda las listas de canciones del usuario del usuario
(puede contener tanto las listas creadas por le mimos o creadas por otros
usuarios)
  private List<PlaylistS> PlaylistS = new List<PlaylistS> { };

// FavSong: Lista que guarda las canciones favoritas del usuario
  private List<Songs> FavSong = new List<Songs> { };

// PlaylistM: Cumple la misma funcion que PlaylistS pero para películas
  private List<PlaylistM> PlaylistM = new List<PlaylistM> { };

// FavMovies: Cumple la misma funcion que FavSong pero para películas
  private List<Movies> FavMovies = new List<Movies> { };
```

Cambios c/r Entrega 2 [Atributos User]:

No hubo cambios (a parte del cambio del nombre de la clase Profile ⟶ Users)

➔ **Songs**

Atributos:

```
  private string name_Song;
  private string album;
  private string songGenre;
  private float rankingS;
  private string lyrics;
  private string datePublish;

// WorkersSong: Contiene los trabajadores pertenecientes a la canción
  private List<WorkerSong> WorkersSong = new List<WorkerSong>();
  private string filePath;
  private string typeFileS;
  private double songSize;
  private TimeSpan durationS;

  private int likesS;
  private int reproductionsS;
  private int downloadsS;
```

Cambios c/r Entrega 2 [Atributos Songs]:

Lo cambiado con respecto a lo del proyecto 2 corresponde a los siguientes atributos:

```
//La misma explicación del cambio realizado en películas.
  private TimeSpan durationS;
  private string FilePath;
```

WorkerMovie

Atributos:

```
public string Name;
public int Age;

// Gender: Genero de la persona
public string Gender;

// WorkerMovieRol: Rol del trabajador (solo acepta una cantidad limitada de
roles,los cuales estan establecidos al agregar trabajadores en el programa)
public string WorkerMovieRol;
public float RankingWM;
```

Cambios c/r Entrega 2 [Atributos WorkerMovie]:

Se elimino MoviesIn que era una lista que contenía las películas en las cuales había participado el trabajador. La razón por la cual se elimino fue mas que nada por que se generaba una recursividad entre workermovie y movies , y tampoco de por si era muy necesario tener que grabar dos veces lo mismo.

WorkerSong

Atributos:

```
private string name;
private string Gender;
private int age;
private string rol;
private float rankingWorkerS;
```

Cambios c/r Entrega 2 [Atributos WorkerSong]:

Se cambio el nombre de Singers a WorkerSong, dado que el nombre no representaba bien a los otros trabajadores que podrían participar en la canción (como es el caso de un compositor). También hubo un cambio parecido a lo realizados en WorkerMovie pero para canciones donde se eliminó SongsIn.

Contenido nuevo:

Archivo de Proyecto [Interfaz y Controllers]:

Este nuevo proyecto de Interfaz y controllers fue agregado para cumplir el modelo controllers , Interfaz y Models , para así evitar acoplamiento entre la interfaz y los modelos.

Este proyecto contiene la interfaz visual y los controllers tal como dice el título. Específicamente los controllers tienen métodos tanto de búsquedas, de almacenamiento y como para obtener información de las respectivas clases de models que están almacenadas

en controllers. Las siguientes clases y appforms que conforman a este archivo de proyecto son las siguientes:

AppForms [Interfaz]:

➔ AppForm: Contiene todas las vistas necesarias para el desarrollo del programa.

Clases [Controllers]:

➔ **MovieControlller:**

Atributos:

```
// DataBaseMovies: Guarda todas las películas agregadas por el usuario
administrador
o   private List<Movies> DataBaseMovies = new List<Movies>();

// TemporaListMovies: Guarda una sola película temporal, la cual va a servir
para poder agregar los trabajadores temporales a esa película (como es temporal
se va a borrar su contenido después de agregar una película).
o   public List<Movies> TemporaListMovies = new List<Movies>();

//Listas que recorren la base de datos de películas y graban el contenido que
cumplen con cada filtro de búsqueda (por categoría, titulo, ranking, etc.)
o   protected List<Movies> KeyWordMovies = new List<Movies> { };
o   protected List<Movies> KeyWordRankingigualmovies = new List<Movies> { };
o   protected List<Movies> KeyWordRankingmenormovies = new List<Movies> { };
o   protected List<Movies> KeyWordRankingmayormovies = new List<Movies> { };
o   protected List<Movies> KeyWordCategoria = new List<Movies> { };
```

➔ **PlaylistMControlller:**

Atributos:

```
// DataBasePlaylistM: Guarda todas las playlist de películas creadas por la
aplicación
o   private List<List<PlaylistM>> DataBasePlaylistM =
    new List<List<PlaylistM>>();
//Faltaron más atributos para completar este controlador (falta de tiempo y
problemas de grupo)
```

➔ **PlaylitsSControlller:**

Atributos:

```
// DataBasePlaylistS: Cumple la misma funcionalidad que DataBasePlaylistM pero
para canciones.
o   private List<List<PlaylistS>> DataBasePlaylistS = new
    List<List<PlaylistS>>();
//Faltaron más atributos
```

➔ **SongsControlller:**

Atributos:

```
// DataBaseSongs: Cumple la misma funcionalidad de MovieController pero para
canciones
```

```
    private List<Songs> DataBaseSongs = new List<Songs>();

    // TemporaryListSongs: Guarda una lista temporal en la cual contiene una
    canción y luego de agregar dicha canción se borra ese contenido y se sigue con
    la siguiente canción.
    private List<Songs> TemporaryListSongs = new List<Songs>();

    //Listas que recorren la base de datos de canciones y graban el contenido que
    cumplen con cada filtro de búsqueda (por categoría, titulo, ranking, etc.)
    protected List<Songs> KeyWordRankingigualsongs = new List<Songs> { };
    protected List<Songs> KeyWordRankingmayorsongs = new List<Songs> { };
    protected List<Songs> KeyWordRankingmenorsongs = new List<Songs> { };
    protected List<Songs> KeyWordSongs = new List<Songs> { };
```

➔ WorkerMovieController

Atributos:

```
//DataBaseWorkersM: Guarda todos los trabajadores de películas
private List<WorkerMovie> DataBaseWorkersM = new List<WorkerMovie>();

// TemporaryListWorkersM: Guarda los trabajadores que se van agregando a una
película, y luego de que se haya agregado la película se borra todo el contenido
de esta y se sigue con la siguiente película si es que el administrador lo desea
(Solo toma los trabajadores nuevos).
private List<WorkerMovie> TemporaryListWorkersM = new List<WorkerMovie>();

// OldWorkersTemporaryList: Lo mismo que para TemporaryListWorkersM pero para
trabajadores antiguos.
private List<WorkerMovie> OldWorkersTemporaryList = new List<WorkerMovie>();

// NotEmptyWorkers: Lista de trabajadores que copia el contenido de las listas
temporales , para que no se pierda la información de las bases de datos una vez se
borran las listas temporales.
private List<WorkerMovie> NotEmptyWorkers = new List<WorkerMovie>();
```

➔ WorkerSongControlller

Atributos:

```
// Estos atributos cumplen la misma funcionalidad que los atributos de
WorkersMovies pero para canciones.
private List<WorkerSong> DataBaseWorkersS = new List<WorkerSong>();

private List<WorkerSong> TemporaryWorkerSongs = new List<WorkerSong>();

private List<WorkerSong> TemporaryOldWorkerS = new List<WorkerSong>();

private List<WorkerSong> NotEmptyWorkersS = new List<WorkerSong>();
```

➔ UserController

Atributos:

```
    // DataBaseUsers: Contiene todos los usuarios de la app (se cargan mediante un
metodo de serialización que graba y carga los usuarios registrados anteriormente)
    private List<User> DataBaseUsers = new List<User>();
```