METODOLOGIAS DE PROGRAMACION II

INFORME FINAL
PLATAFORMA DE STREAMING

DOCENTE: Claudia Cappelleti
INTEGRANTES: Persoglia Rodrigo, Marcos Sosa y Lucas Muñoz

Contenido

1) Descripción del problema planteado.

El cliente solicita la creación de una plataforma de streaming de películas y series. La misma constara de 3 tipos de planes que brindan acceso a distintos tipos de contenido, siendo 'Plan Oro' quien brinda acceso a todo el contenido internacional de la plataforma, 'Plan Plata' donde solo se puede acceder al contenido disponible para su país y 'Plan Básico' donde el contenido es mas limitado.

Los usuarios tendrán una suscripción a uno de los planes por 30 días y el pago se realiza mediante debito automático en su tarjeta de crédito. La suscripción se renueva automáticamente finalizado ese plazo salvo que el cliente solicite la baja del mismo o falle el medio de pago.

Los usuarios dentro de la plataforma podrán buscar el contenido en una barra de búsqueda filtrando por nombre, por director o por genero del contenido. También la plataforma debe ordenar si el cliente lo quisiera por las mayores valoraciones y sugerir un contenido exclusivo de acuerdo a su historial de búsquedas.

2) Especificación e implementación de las clases utilizadas.

Las clases utilizadas son las siguientes:

1. **Plataforma:** Tiene los datos principales de la propia plataforma (nombre, razón social, teléfono, mail y legales). Además, tiene una lista de los usuarios de la plataforma, una lista de los planes disponibles y una lista del contenido visual el cual se explicará mas en detalle posteriormente. También la clase Plataforma tiene los métodos de búsqueda de contenido.
2. **Plan:** Tiene los precios y la descripción de cada plan al que un usuario puede suscribirse.
3. **Usuario:** Tiene los datos principales del usuario (nombre, mail, usuario, clave, tarjeta de crédito). Además, cada usuario tiene una suscripción a un Plan. La clase usuario valida que el usuario tenga una suscripción activa para consumir el contenido que ofrece Plataforma.
4. **Suscripción:** Esta clase funciona de nexo entre el Usuario y el Plan al que esta suscripto actualmente. Usuario valida la suscripción mediante la fecha de vencimiento. También nos indica el tipo de plan y por consecuencia el contenido al que el Usuario puede acceder.
5. **ContenidoVisual:** Es una clase abstracta la cual heredaran las clases Película y Serie. Agrupa las principales características de un contenido (nombre, fecha de estreno, director, genero, calificación, valoración, sinopsis y tipo de plan). También tiene un listado del reparto de ese contenido y un método para consultar por el mismo.
6. **Elenco:** Tiene el dato del rol de cada actor/actriz para un determinado contenido visual.
7. **Película:** Sub clase de ContenidoVisual. Además de los atributos de su clase padre, Película nos brinda la url y urlTrailer de la misma y el nombre de la saga en caso de pertenecer a alguna.
8. **Serie:** Sub clase de ContenidoVisual. Además de los atributos de su clase padre, Serie nos indica la cantidad de temporadas que la propia Serie tiene. Además, una serie esta compuesta por Episodios.
9. **Episodio:** Esta clase tiene nos detalla nombre, numero de temporada y numero de episodio. Al igual que Película, Episodio nos proporciona url y urlTrailer que son los datos concretos que la plataforma necesita para la reproducción del contenido solicitado.

3) Implementación de la aplicación desarrollada.

La aplicación fue escrita en Smalltalk usando Dolphin Smalltalk v.7.0.5 como ide. La misma podríamos dividirla en 2 partes. La implementación de las clases descriptas en el punto anterior y la aplicación propiamente dicha. Dentro de la aplicación también tenemos 2 secciones. Por un lado, tenemos una sección usuarios, donde hay algunos usuarios precargados y también la opción de hacer una carga masiva de dicha entidad. La sección usuarios es una aplicación símil consola donde la entrada de datos se hace mediante Prompter prompt. El menú de esta sección lo logramos usado whileTrue y la aplicación finaliza cuando el usuario ingresa 'fin'. Dentro del menú podemos acceder a las siguientes funcionalidades:

1. Ingresar Usuario.
2. Ver suscripciones activas.
3. Contar Suscripciones Activas/Inactivas.
4. Buscar Usuarios por mail.
5. Mostrar contraseñas no seguras.
6. Buscar usuario por su Nick.

La segunda parte de la aplicación es la búsqueda de contenido. Dentro de la misma instanciamos la Plataforma con el contenido visual precargado. Al igual que la sección usuarios es una aplicación símil consola donde la entrada de datos se hace mediante Prompter prompt. El menú de esta sección lo logramos usado whileTrue y la aplicación finaliza cuando el usuario ingresa la opción '6'. Dentro del menú podemos realizar búsquedas de contenido por los siguientes criterios:

1. Por Nombre del contenido (serie o película)
2. Por Director del contenido.
3. Por Genero del contenido.
4. Por Valoración.
5. Sugerencia (sugiere un contenido en particular de los mas valorados)

4) Descripción de la metodología de trabajo utilizada por los integrantes del grupo, dónde deberá indicarse si adoptó algunas de las prácticas ágiles vistas en la materia y a qué metodología corresponde.

Para el desarrollo de la aplicación se usó la metodología XP. La elección de esta metodología se fundamenta en la adaptabilidad a cambios que proporciona la misma. Primero definimos las pruebas a realizar y luego la codificación. La carga horaria semanal no fue definida estrictamente sino que fue algo acorde a los tiempos que manejó cada programados. Se hizo una reunion en la cual concretamos la fase de exploración y planificación. En la misma definimos las historias de usuarios para posteriormente iniciar con las iteraciones. Tambien realizamos reuniones donde se programó en parejas. En cuanto a los Roles, los alumnos cumplimos con el rol de desarrolladores y la docente con el rol de cliente. Las pruebas y el seguimiento se hizo en conjunto (tester y tracker) y en cuanto a los roles de coach y big boss no fueron requeridos.

5) Breve descripción de los Frameworks que se podrían utilizar para realizar un desarrollo orientado a objetos.

Los framework que se podrían usar en un desarrollo orientado a objetos dependerían de la naturaleza del proyecto. Por ejemplo, si necesitáramos que nuestro desarrollo sea una aplicación de escritorio una buena elección seria usar .Net (framework para lenguaje C#) que cuenta con una opción para el desarrollo de este tipo de aplicaciones con una amplia gama de herramientas que facilitan el desarrollo de las mismas.

Si el desarrollo fuera una API podríamos optar también por usar .NET o Spring Boot (Java), ambos casos son lenguajes orientados a objetos fuertemente tipados lo que nos da mayor control sobre el flujo de información que manejamos.

Si necesitáramos que la aplicación fuera una pagina web además de las APIs antes mencionadas deberíamos agregar la capa de front, para lo cual nos podemos valer de frameworks web como Angular o similares.

6) Especificación de al menos dos casos de uso de Patrones en el contexto del problema planteado.

Para el desarrollo de nuestra aplicación podriamos usar los siguientes patrones:

- **Strategy**: Debido que el contenido visual puede ser una película o una serie (la cual a su vez tiene episodios) necesitamos 2 estrategias distintas para la reproducción del contenido.
- **Iterator**: La búsqueda de contenido puede arrojar mas de 1 resultado por lo que tener una interface de iteración uniforme es una necesidad. Usando iterator podemos lograr este objetivo.

7) La descripción de qué repositorios se podrían utilizar para el desarrollo de software colaborativo.

Para el desarrollo de esta aplicación utilizamos un repositorio remoto en Github (https://github.com/markitos28/Streaming). Las clases en formato .cls y las aplicaciones en formato .st creadas durante el desarrollo las exportamos usando el comando 'File Out' para el commit inicial. Posteriormente cada programador antes de empezar a trabajar debe hace un pull del repositorio remoto al repositorio local, importar las clases que tengan modificaciones usando el comando 'File In' y proceder a la codificación. Finalizada la codificación, luego de guardar los cambios, usamos el comando 'File Out' nuevamente para exportar los archivos que tengan modificaciones, las reemplazamos en el repositorio local, se hace un commit con una descripción acorde a los cambios realizados y por último un push.