**ANEXO EXAMEN DE CERTIFICACIÓN**

|  |  |
| --- | --- |
| Plan de Estudio | Desarrollo Aplicaciones Fullstack Java Trainee |
| Anexo | Caso Home Flix Prime |

# Caso “Home Flix Prime”

La compañía “Home Flix Prime” es una idea creada por un grupo de emprendedores locales que quieren mejorar la forma en que las cadenas de cable local distribuyen sus contenidos on-demand. La idea de fondo, es brindar una plataforma que permita la administración de los títulos que están disponibles bajo este tipo de plataformas on-demand y cobrarles un fee mensual por el uso.

El grupo de emprendedores necesita buscar financiamiento, razón por la cual decide construir un prototipo rápido con el objeto de ponerlo prontamente a pruebas con un grupo de usuarios y sacar feedback temprano. Su plan es tener en 2 meses un prototipo de producto que le permita conseguir inversionistas que financien el desarrollo del producto final y les ayude con la comercialización.

Para este propósito, se ha formado un equipo de desarrollo de primera línea del cual usted forma parte como desarrollador full-stack Java. El equipo de proyectos también lo conforma un Scrum Master, un Product Owner, un Diseñador UX/UI, un Diseñador Web, un Desarrollador Mobile, y un Arquitecto de Software Cloud.

El proyecto busca, dentro de otras cosas, organizar los títulos de films que estarán disponibles en una primera instancia. A continuación, se listan los requisitos funcionales de alto nivel del sistema:

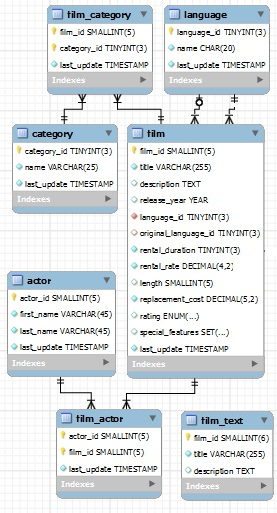
* El sistema debe permitir visualizar el catálogo de títulos
* El sistema debe permitir realizar búsquedas de títulos
* El sistema debe permitir analizar si los títulos son atractivos o no

A la fecha, ha transcurrido gran parte del proyecto y se tiene el siguiente avance:

* Ya se cuenta con un prototipo funcional del aplicativo
* Existe un modelo de datos diseñado
* Existe una base de datos con datos de prueba
* Existe una aplicación web desarrollada con Spring Framework que desarrolla algunas de las funcionalidades requeridas

## Modelo de Datos

A continuación, se presenta el modelo de datos diseñado por el arquitecto en conjunto con el analista funcional.



La tabla *film* contiene todos los títulos de películas que se quieren administrar en la plataforma, destacándose atributos como el lenguaje original (*original\_language*\_id), el lenguaje de doblaje (*languague\_id*), el año de lanzamiento (*release\_year*), la duración (*length*), la clasificación parental (*rating*), y las características especiales (*special\_features*), entre otros. Asimismo, información del valor del servicio on-demand, en los atributos *rental\_rate* (valor del arriendo) y *rental\_duration* (tiempo disponible para ver el título).

La tabla *language* contiene el listado de lenguajes que son referenciados en cada film. Nótese que tiene una doble relación con la tabla *film*, tanto con el idioma original como con el idioma del doblaje.

Por otra parte, existe una tabla de categorías (*category*) y una tabla que relaciona las categorías con los films (*film\_category*), esto significa que un film podría estar en varias categorías.

De la misma forma, existe una tabla de actores (*actor*), que contiene el listado de todos los actores y que se relaciona a través de la tabla *film\_actor*, que contiene la relación entre actores y películas ya que obviamente una película tiene muchos actores y un actor puede estar en varias películas.

## Requerimientos a Desarrollar

El jefe de proyectos, quien lleva un control meticuloso de las actividades del proyecto, le ha solicitado a Usted que realice las siguientes tareas:

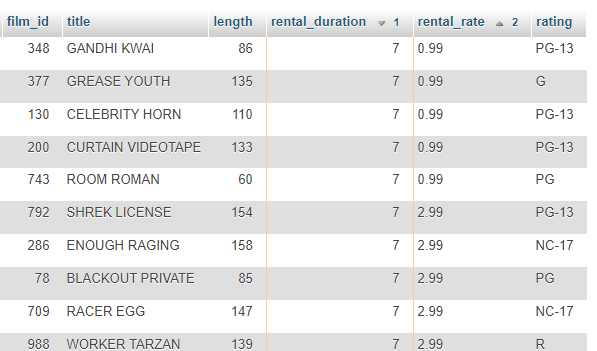
1. Realizar consultas a la base de datos
2. Construir un algoritmo de análisis de títulos
3. Construir unidades de prueba para verificar el analizador de títulos
4. Crear un buscador de películas
5. Crear una API REST que disponibilice la información del buscador de películas

A continuación, se especifica con mayor detalle cada uno de los requerimientos:

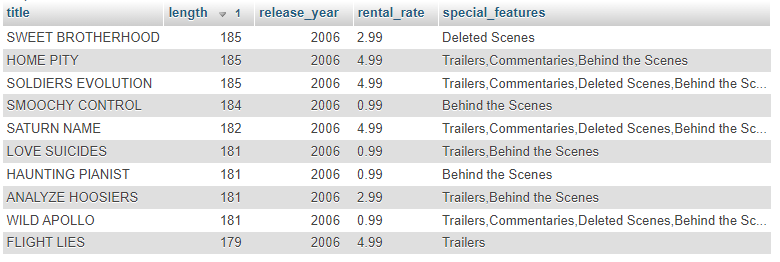
### Realizar consultas a la base de datos

Antes de comenzar con el desarrollo, se decidió hacer un análisis exploratorio de la base de datos con el objeto de familiarizarse con el esquema y con los datos. A continuación, se le solicita que construya las siguientes consultas SQL. **Para esto, cree un package en su proyecto Java con nombre “consultas”, cree un archivo de texto plano por cada una de ellas, identificando claramente de qué consulta se trata** (ejm: Consulta-A.sql, Consulta-B.sql, …etc).

1. Se requiere conocer el listado de películas de agentes secretos (en su descripción contiene “Secret Agent”) y que su duración de arriendo es de 5 o más días, ordenar por duración descendente y precio de renta ascendente. Desplegar sólo los campos señalados en la imagen referencial:

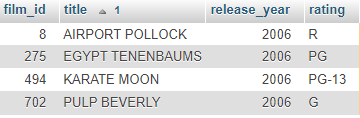


1. Se requiere conocer las 10 películas de mayor duración que tienen rating “R”, de forma descendente. Debe desplegar el título, la duración, el año, el precio de renta y las características especiales. El reporte debe tener la siguiente forma:

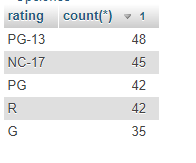


Nota: si no está familiarizado con cómo limitar el resultado de una consulta, puede buscar en google “*how do I limit a SQL query?*”.

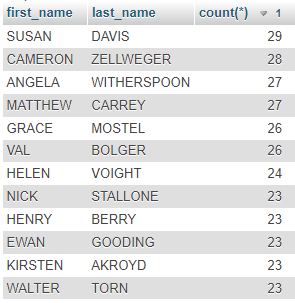
1. Se requiere tener un listado con las películas de la categoría ‘Horror’ en las que ha participado SUSAN DAVIS. Ordene el resultado por título ascendente. El reporte debe tener la siguiente forma:



1. Se requiere un listado con la cantidad de películas que hay para cada rating, pero sólo considerando aquellas cuyo lenguaje original es Francés, ordene el resultado por cantidad descendiente. El reporte debe tener la siguiente forma:



1. Se requiere el listado de artistas con la cantidad de películas en la cual aparece, pero que tienen rating PG, PG13 ó G, ordenar por cantidad descendente. El reporte debe tener la siguiente forma:

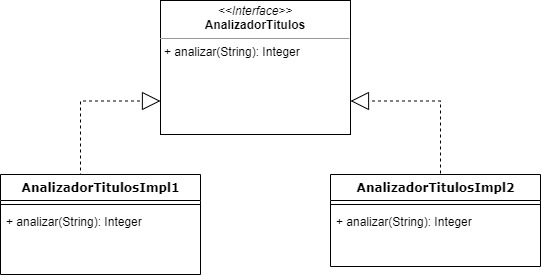


### Implementar algoritmo para el análisis de nombres de películas

Una funcionalidad importante en este sistema es la recomendación de títulos, la idea es que el usuario elija las películas recomendadas la mayor parte de las veces porque de esta forma se genera un enganche con la plataforma.

Para lograr este cometido, se está analizando la hipótesis de cómo el nombre de la película influye en la elección. En la actualidad, la opinión está dividida entre los product managers razón por la cual se solicita crear dos algoritmos para decidir si el nombre es atractivo o no, pero tratando que el cambio entre uno u otro algoritmo impacte lo menos posible al software.

El arquitecto propuso definir una interfaz de forma que se puedan llevar a cabo las dos implementaciones propuestas. El siguiente diagrama refleja el diseño propuesto por el arquitecto:



En el diseño propuesto por el arquitecto, hay un método que se llama analizar y que recibe como parámetro un String con el título de la película. El método retornará un número entero de acuerdo a las siguientes condiciones:

* Si el título es atractivo, entonces retornará el valor 1
* Si el título es indiferente, entonces retornará el valor 0
* SI el título es poco atractivo, entonces retornará el valor -1

Cada implementación debe cumplir con la definición anterior. Para este efecto, se especifica el algoritmo que debe tener cada una de las implementaciones solicitadas:

Las reglas del **analizador de títulos implementación 1** son las siguientes:

* El algoritmo deberá recibir el título de la película y deberá contar cuántas palabras tiene.
* Si tiene una palabra, entonces se considera un título poco atractivo
* Si tiene dos palabras, se considera un título indiferente
* Si tiene más de dos palabras, se considera un título atractivo

Las reglas del **analizador de títulos implementación 2** son las siguientes:

* El algoritmo deberá recibir el título de la película y deberá contar cuántas vocales tiene en total
* Si tiene menos de 8 vocales, entonces se considera un título poco atractivo
* Si tiene entre 9 y 12 vocales, se considera un título indiferente
* Si tiene más de 12 vocales, se considera un título atractivo

Para probar el buen funcionamiento de ambos algoritmos, podrá crear una aplicación de consola o bien lo podrá hacer directamente en las unidades de pruebas respectivas que se solicitan en la siguiente sección.

### Construir unidades de prueba para verificar funcionamiento de ambas implementaciones

Construya clases de prueba en Java que permita verificar el correcto funcionamiento **de ambas implementaciones**, considerando al menos los siguientes tests:

* Tests que considere casos normales
* Tests que considere condiciones de borde

### Crear Buscador de películas

Para mejorar la gestión de los administradores, se requiere crear una página web dinámica que despliegue el buscador de películas, tal como se detalla en la siguiente imagen mock-up.



Se pide:

1. Desplegar el listado de categorías en el primer combobox, ordenado alfabéticamente, con valores que provengan de la base de datos
2. Desplegar un cuadro de texto para ingresar el título o parte del título a buscar.
3. Desplegar el listado de películas y su información principal (de acuerdo al detalle especificado en la imagen mock-up), de acuerdo a los filtros seleccionados (categoría y título), a partir de la información que se encuentra en la base de datos.

Para realizar el requerimiento, el arquitecto le señala lo siguiente:

* Utilizar jsp y taglibs jstl para el despliegue de la vista (u otra tecnología de vista)
* Utilizar bootstrap para los componentes y controles
* Que los elementos se ajusten a distintos tamaños de pantalla

### Crear una API REST que disponibilice el buscador de películas

Disponibilice un servicio REST que permita obtener la misma información del buscador de preguntas del ejercicio anterior. Recuerde que el servicio podría recibir como parámetro la categoría y el título (o parte del título) para realizar la búsqueda.

Como recomendación, se sugiere la reutilización de capas de su arquitectura MVC.

# Consideraciones generales

A continuación, algunas consideraciones generales para la ejecución de este caso:

* Recuerde que el trabajo es individual
* Puede navegar internet si lo requiere
* Recuerde que debe entregar el código fuente del proyecto, no sirve si se entrega el ejecutable (.class, .war)