**DOCUMENTO DE DISEÑO DE ERRORES Y EXCEPCIONES**

**DANIEL ARANGO CRUZ**

**LAURA DANIELA ARIAS FLOREZ**

**DIEGO ALEJANDRO GONZALEZ VARGAS**

**INTRODUCCION**

El presente trabajo tiene como objetivo la consolidación de un diseño y documentación del manejo de errores y excepciones propuestas para un proyecto de programación orientada a objetos. Este proyecto tiene como finalidad principal la consolidación de una aplicación que permita las búsquedas y manipulación sencilla e intuitiva de unas bases de datos NO relacionales provenientes de una biblioteca. Dicha base de datos es pública por lo que su información no se encuentra sujeta a derechos de autor. Así mismo, vale la pena mencionar también los requerimientos funcionales de la aplicación, entre los cuales encontramos los siguientes:

* Buscar un libro en la base de datos a partir de su título.
* Buscar los libros de un autor a partir del nombre o parte del nombre de este.
* Buscar las categorías en las que participa un autor a partir del nombre del autor.
* Mostrar la calificación promedio de todos los libros.
* Mostrar la categoría con más libros.
* Mostrar la categoría mejor calificada.
* Mostrar el número de libros sin su portada cargada.
* Mostrar autores que participan en más de una categoría.
* Renombrar una categoría
* Borrar los libros de un(os) autores específicos.

Dicho todo lo anterior, las últimas dos funcionalidades, así como la carga de datos, tienen múltiples escenarios que se considera que pueden generar errores en la ejecución del programa debido a circunstancias específicas que se pueden presentar y de las cuales se debe tener cuidado.

En este sentido, se consideró pertinente la generación de excepciones para manejar los casos particulares que pueden presentar. No obstante, la generación de estas excepciones esta fuertemente ligada a la estructura general del proyecto. En consecuencia, lejos de modificarse la estructura, se propone la agregación de métodos a las clases y sus excepciones correspondientes, por lo que se concluye con la presentación del gráfico UML de la aplicación original:

Diagrama

Descripción generada automáticamente

**EXCEPCION 1: CARGA DE DATOS**

**Motivación/ Contexto:**

La aplicación funciona inicialmente sin ningún tipo de dato. Así pues, es necesario la carga individual de los documentos correspondientes a las categorías y a los libros de la biblioteca. No obstante, dentro de este proceso no hay garantías de que todos los libros pertenezcan a las categorías del otro documento. Vale la pena aclara que la carga de categorías se hace de manera previa a la carga de libros. Así las cosas, se hace necesaria la creación de un mecanismo de contención de un posible error en la carga de libros sin categoría preexistente. En concreto, se propone el manejo de una excepción, aplicándose como procede:

**Explicación (Métodos creados/ Clases modificadas):**

Para poder manejar la excepción de una categoría inexistente, hizo falta la modificación del método que realizaba la búsqueda de la categoría en el modelo lógico de la librería, de modo que expulsase la excepción si no hallaba la categoría como procede:

Texto

Descripción generada automáticamente

Posteriormente, esta excepción es atrapada por el método que la invoca dentro de la misma librería lógica en la carga de los libros, e invoca la creación de una categoría si es necesario, como procede:

Texto

Descripción generada automáticamente

Finalmente, hizo falta la creación del método para agregar la categoría:

Captura de pantalla de computadora

Descripción generada automáticamente

Adicionalmente, en cuanto a las modificaciones de interfaz, se tuvo que imprimir las categorías nuevas creadas, así como el nombre y la cantidad de libros de cada una de ellas. Para poder alcanzar este objetivo se propuso un atributo adicional para la biblioteca que mostrase las categorías nuevas no contempladas en los documentos, y de allí sacar toda la información necesaria para la impresión, como se muestra:

Imagen que contiene Texto

Descripción generada automáticamente

**Diagrama de secuencia:**

Diagrama

Descripción generada automáticamente

**EXCEPCION 2: RENOMBRAR CATEGORIAS REPETIDAS**

**Motivación/ Contexto:**

Una de las nuevas funcionalidades propuestas era la de renombrar las categorías a las que pertenecían los libros. Sin embargo, esta funcionalidad tenía una restricción: NO se podía tener dos categorías llamadas igual ya que esto afectaría el correcto funcionamiento de las búsquedas. Dicho lo anterior, para la implementación de esta nueva funcionalidad también se propuso una excepción que le diera manejo a este escenario de error, como sigue:

**Explicación (Métodos creados/ Clases modificadas):**

Para poder hacer la implementación de esta funcionalidad se creó un botón adicional en la interfaz gráfica que desplegaba un diálogo que solicitaba al usuario la categoría a modificar.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media

Posteriormente, se llamaba un método dentro del modelo lógico que hallase la categoría, y en caso de encontrarla, desplegaba un nuevo diálogo que solicitaba el nuevo nombre de la categoría. No obstante, es aquí donde se utiliza el método que detecta la creación de clases dobles e impide la creación de la misma, por lo que se encierra este método dentro de un try-catch que permite el despliegue de un dialogo final que indique al usuario si se presentó la excepción

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media

Para la implementación lógica del método cambiarCate dentro de la librería del modelo, lo primero que se tuvo en cuenta fue la preexistencia del nombre nuevo de la categoría, a partir de un segundo método que retornaba un booleano que indicaba la necesidad o no de la expulsión de la excepción:

Captura de pantalla de computadora

Descripción generada automáticamente

Posteriormente, en caso de descartar la duplicidad del nombre, se procede a cambiar el nombre de la categoría a partir de un setter común dentro de la clase categoría:

Captura de pantalla de computadora

Descripción generada automáticamente

Para evitar ambigüedades, se procede a mostrar el método auxiliar usado para la excepción:

Captura de pantalla de computadora

Descripción generada automáticamente

**Diagrama de secuencia:**

Diagrama

Descripción generada automáticamente

**EXCEPCION 3: BORRADO DE AUTORES INEXISTENTES:**

**Motivación/ Contexto:**

Otra de las nuevas funcionalidades propuestas para la implementación de esta aplicación era la creación de un opción que permitiese la eliminación de todos los libros de la base de datos escritos por uno o más autores especificados por parámetro. Sin embargo, al igual que en los casos anteriores, esat funcionalidad tenía un escenario de error de bastante probabilidad que era la escritura de nombres de autores inexistentes, y este escenario debía parar el proceso de borrado de todos los libros. Así las cosas, una vez más, lejos de fabricar métodos de prevención, se manejaron excepciones para el control de este tipo de escenarios, y se implementaron como procede:

**Explicación (Métodos creados/ Clases modificadas):**

Siguiendo una metodología muy similar a la utilizada en la excepción anterior, para la implementación de esta funcionalidad se tuvo en cuenta la creación de un nuevo botón que invocara un método dentro de la interfaz. Este método es el de mayor importancia en el análisis de la interfaz, pues es este método el que invoca la librería y su borrado de libros, que finalmente es la que genera la excepción. Así pues, el método dentro de la interfaz atrapa la excepción e imprime un mensaje de acuerdo con el éxito o fracaso del proceso dentro de la librería:

Captura de pantalla de computadora

Descripción generada automáticamente

Ahora bien, la implementación de esta funcionalidad y la generación de la excepción se encuentran en el método borrarLibros del modelo. Este método inicia buscando cada uno de los nombres en el catálogo para saber cuantos y cuales libros tiene cada autor, de modo que a partir de eso se saca una lista de booleanos en función de la existencia o no de libros para cada uno de los autores (Se conserva el orden de la lista de string, de modo que el boolean de la posición 1 indica si el autor de la posición uno de String[] tiene o no libros).

Captura de pantalla de computadora

Descripción generada automáticamente

Posteriormente, si todos los autores tienen libros se procede a borrar del catálogo general y de la lista de libros de las categorías.

Captura de pantalla de computadora

Descripción generada automáticamente

La implementación del borrado en categoría es el siguiente:

Captura de pantalla de computadora

Descripción generada automáticamente

Por otra parte, si se tiene tan solo un false en la lista de libros por autor, se hace necesaria la expulsión de la excepción, por lo que se realiza esa comprobación. Para la impresión, se configura toda la información de los nombres que son o no autores reales en el mensaje de la excepción.

Captura de pantalla de computadora

Descripción generada automáticamente

**Diagrama de secuencia:**

Diagrama

Descripción generada automáticamente