Programa “misLibros” en Java

Algoritmos y Programas (COM-11101)

José Ulises Quevedo Llergo
000189442

Instituto Tecnológico Autónomo de México

Ciudad de México a 6 de diciembre de 2019

1. Introducción

a) Contexto

Desde que soy pequeño me ha gustado leer. Comencé con unos cuantos libros: la colección de Harry Potter, libros de cuentos para pequeños y uno que otro libro de chistes. A medida que yo crecía, mi colección comenzaba a crecer un poco más. Nuevos géneros se iban añadiendo a la colección, sagas enteras llenaban mis estantes. Unos años más tarde, y ahora era yo quién compraba mis propios libros. No solo eso, si no que me comencé a comprar libros más rápido de lo que los leía. De repente mi librero se llenó de libros sin abrir, sagas sin uno o dos libros, libros de géneros extraños que no sabía donde poner. La clase de algoritmos y programas me dio la oportunidad de poner un poco de orden en el caos que mi colección de libros se ha convertido.

b) Identificación del problema

Soy extremadamente meticuloso con mis libros. Me gusta saber exactamente cuántos libros tengo, de qué genero, de qué autores, qué libros ya leí y cuáles aún no. He intentado organizar la información de mis libros de mil maneras diferentes: a mano, anotando todos los datos en papel; con archivos de Excel, donde se crean tablas enormes difíciles de filtrar y modificar; incluso con plataformas en línea que están diseñadas específicamente para eso. Ninguna funcionaba para mí. Ya sea porque eran difíciles de actualizar, utilizar o mantener, o porque tenían funcionalidad que no necesitaba, y le faltaba alguna funcionalidad que para mí era indispensable. Es clave para mí, por ejemplo, saber qué libros tengo prestados y a quién se los presté, para poder así recuperarlos después. También necesito saber qué libros ya leí, pues tengo muchos libros en mi librero que compré y jamás leí. Era momento, entonces, de diseñar un sistema de mí y para mí, que combinara la funcionalidad que quería con la facilidad de uso y mantenimiento que mi librero requiere.

c) Objetivos

Con mi proyecto, entonces, busco crear un sistema que pueda utilizar para organizar mi librero. Debe ser capaz de adaptarse a mi librero: registrar los libros en varios géneros diferentes y mantener cada uno separado. Permitirme ver qué libros de mi librero he leído y cuáles no, filtrar mis libros por calificación, por autor, y por genero. No debe ser complicado de usar, y debe ayudarme, en suma, a mantener orden en mi librero. De esta forma, el problema de orden en mi librero se verá resuelto y podré recuperar el control sobre mis libros.

d) Organización del documento

En este documento se presenta el proceso de diseño e implementación de mi proyecto. Primero, se analizan los requisitos funcionales y restricciones del mismo. A continuación, se describen las partes que forman al proyecto. Se describen las clases usadas y a grandes rasgos, cómo funcionan y que papel juegan en el proyecto. Se incluyen también los diagramas UML de las clases, en los que se incluyen todos sus métodos y atributos. En tercer lugar, se describe la implementación como un todo: el funcionamiento del sistema, su interfaz gráfica y cómo utilizar la aplicación. Después, se comentan las pruebas que se realizaron durante el proyecto, además de las modificaciones que se realizaron como consecuencia de estas pruebas. Finalmente, se concluye y se presenta el código de todas las clases utilizadas, al igual que el código de las pruebas realizadas.

2. Análisis

a) Requisitos funcionales

Existen una serie de requisitos que el programa debe cumplir. En primer lugar, debe trabajar con arreglos y matrices. Debe poder trabajar sobre estos, realizando las siguientes operaciones: dar de alta un objeto dentro del arreglo o matriz; eliminar un elemento de la colección; buscar elementos

en la matriz, o filtrar los elementos a través de una búsqueda que recopile todos los miembros que cumplan con alguna condición. El programa debe también, en algún momento, realizar una lectura de datos desde un archivo. Se deben utilizar métodos genéricos para trabajar con los arreglos. Por último, el programa debe cumplir con los criterios de código limpio y ordenado que se aprendieron durante el curso (sangrías, nomenclaturas, etc.).

b) Restricciones

El programa debe estar programado en Java, utilizando el paradigma de programación orientada a objetos (POO) para las clases, y el modelo Vista-Controlador para la interfaz gráfica. Por lo tanto, se debe respetar el principio de encapsulamiento, protegiendo las clases internas y accediendo a ellas a través de las clases externas. El resto de las restricciones se presentan por las limitaciones de mis habilidades, y son limitaciones del programa que se comentarán más adelante.

3. Diseño

a) Descripción de cada clase

- Clase Libro

La clase Libro es la clase base del proyecto. Esta clase modela un libro de la vida real. Por lo tanto, todo libro tiene título, autor, género y formato (digital o físico). El libro puede estar marcado como prestado a través de una variable booleana. Además, cada libro puede estar leído o no leído; dependiendo de esto cada libro puede también tener una calificación y una opinión. Cada libro contiene un arreglo de tipo Cita, que contiene citas del libro utilizando la clase que se describe en el siguiente apartado. La clase libro tiene una serie de métodos para acceder a sus atributos (get's), y algunos para modificarlos (por ejemplo, para cambiar el estado de un libro a leído, prestado, cambiar su calificación o agregarle una cita).

- Clase Cita

La clase Cita es una clase pequeña que se utiliza para guardar los datos de una cita de un libro. Por lo tanto, sus únicos atributos son el texto de la cita y la página del libro donde se encuentra. La clase tiene la habilidad de regresar el texto, la página, o una representación de la cita que contiene tanto la página como el texto. Las citas solamente existen dentro de los libros, en un arreglo. La clase Libro agrega citas a su arreglo especificando el texto y la página, y creando ahí mismo la cita. También puede imprimir todas las citas de un libro, llamando al toString() de cada cita del arreglo.

- Clase Librero

La clase Librero es la clase externa del proyecto, a través de la cual se maneja todo el programa. El librero tiene una matriz de Libros, que representa un librero donde cada renglón es un género diferente. Por lo tanto, tiene también un arreglo de los géneros permitidos. Tiene también un arreglo donde se guardan los libros prestados, junto con un arreglo paralelo donde se guarda el nombre de la persona a la que se le prestó. Respetando el principio de encapsulamiento, es a través del librero que se accede a los libros, y a sus citas. Por lo tanto, para cada método de la clase Libro (prestar, leer, agregarCita, etc.) existe un método en la clase Librero que recibe el titulo del libro (y cualquier otro parámetro necesario) y a través de este se accede al método del Libro. Así, se evita trabajar directamente con el objeto de la clase Libro. La clase librero tiene otras dos clases de métodos: los primeros son métodos que se necesitan para poder trabajar con la interfaz gráfica (de la cuál se hablará más adelante). Estos regresan algunos atributos del Librero que se necesitan para poder montar la interfaz de manera correcta. El resto de los métodos del librero son los métodos que resuelven el problema: búsquedas de libros leídos o no leídos, filtrado de libros por autor, inserción y eliminación de libros del librero, transferencia del librero a la colección de libros

prestados, entre otras. Todos estos métodos trabajan con los libros del librero, y representan la parte más significativa del proyecto.

- Clase LibreroVista

Esta clase es la clase responsable de crear la interfaz gráfica principal. Trabaja con la librería Java Swing, y utiliza diversos componentes para crear el interfaz principal del programa. Los atributos de esta clase son los diferentes componentes de la interfaz. Esta interfaz es la correspondiente al librero, y por lo tanto contiene campos para trabajar con los métodos de búsqueda, filtrado, alta y baja de libros. La clase tiene como único método a su constructor, en el cual se construyen todos los componentes y se van agregando a los paneles, para así construir la vista.

- Clase LibreroControlador:

La clase LibreroControlador es la encargada de conectar a la clase LibreroVista con la clase Librero. Esta le da funcionalidad a todos los botones de la interfaz, e invoca a los diversos métodos del librero. Esta clase contiene, como atributo estático, un objeto de la clase Librero, que es a partir del cual trabaja el programa. Contiene también, un constructor donde se asocian los botones de la Vista con su escuchador correspondiente. La clase contiene una serie de clases privadas internas, que son los escuchadores: estas implementan la interfaz ActionListener y contienen el código que debe correr cuando los botones son detonados. En una de estas clases privadas se crea un LibroControlador (del que se hablará a continuación) y desde el cual se manejan las vistas adicionales. Esta clase, por último, contiene un método main desde el que se corre el programa, creando el objeto de la clase LibreroControlador y comenzando la ejecución.

- Clase LibroVista

La clase LibroVista es la encargada de crear la interfaz particular para trabajar con libros individuales. Esta vista presenta los datos de un libro, al igual que las opciones para prestarlo y modificar su calificación. También muestra las citas del libro, y contiene un área para agregar nuevas citas.

- Clase LibroControlador

Esta clase es la que le da funcionalidad a la clase LibroVista. Aunque trabaja con un libro particular, en ningún momento tiene contacto con un objeto de tipo libro. Más bien, trabaja con el mismo objeto de la clase Librero que utiliza el LibreroControlador, pues la clase Librero tiene métodos para modificar el estado de los Libros del librero. Este controlador no tiene un main, pues el objeto de la clase LibroControlador se instancia a través del LibreroControlador, por lo que con el main del LibreroControlador basta.

- Clase ManejadorArreglosGenerico

Esta clase contiene métodos estáticos que trabajan con arreglos. Los métodos son genéricos ("Tipo" T) por lo que funcionan con arreglos de cualquier tipo. Asumen que los objetos de los arreglos implementarán la interfaz *Comparable,* y por lo tanto que podrán utilizar su método compareTo() para ordenar, buscar, etc. La clase se usa frecuentemente en la clase Librero para buscar libros, eliminar e insertar en géneros, y trabajar con el arreglo de libros prestados.

- Clase ManejadorMatricesGenerico

Similar a la clase anterior, esta clase contiene una colección de métodos estáticos genéricos para trabajar con matrices. Debido a que el atributo principal del librero es una matriz de tipo Libro, esta clase se uso especialmente para las búsquedas por título.

b) Estándares Utilizados

Se trabajó con un enfoque de programación orientada a objetos, por lo que para cada clase que se diseño se consideraron sus atributos y métodos. Estos se organizaron y se presentan en diagramas del Unified Modeling Language (UML), estándar en la industria del desarrollo de software. A continuación, se incluyen los diagramas UML de las clases mencionadas anteriormente.[1]

Libro
– titulo : String – genero: String – autor: String – formato: Char – citas: Cita[] – citasRegistradas: int – MAX_CITAS: int = 20 – leido: boolean – prestado: boolean – calificacion: double – opinion: String
+ Libro() + Libro(String) + Libro(String, String, String, char) + Libro(String, String, String, char, double, String) + getTitulo() : String + getGenero() : String + getAutor() : String + getFormato() : char + isLeido() : boolean + leer() + isPrestado() : boolean + prestar() + regresar() + getCalificacion() : double + calificar(double)

[1] No se incluyen diagramas UML para las clases LibreroControlador y LibroControlador, sin embargo, el código de ambas se puede encontrar al final del documento.

+ setOpinion(String) + getOpinion() : String + agregarCita(String, int) : boolean + getCitasRegistradas(): int + getCitas() : String + toString() : String + compareTo(Libro) : int + equals(Object) : boolean + hashCode() : int

Cita
– titulo: String – pag: int
+ Cita(String, int) + getCita() : String + getPag() : int + toString() : String + compareTo(Cita) : int + equals(Object) : Boolean + hashCode() : int

Librero
– dueno: String – libros: Libro[][] – generos: String[] – librosXGenero: int[] ~ MAX_GENEROS : int = 4 ~ MAX_LIBROS: int = 40 – librosPrestados: Libro[] – cuantosPrestados: int – personasPrestadas: String[] – MAX_PRESTADOS : int = 20 – librosTotales : int
+ Librero() + Librero(String) + getDueno() : String + getLibrosTotales() : int

+ getGeneros() : String[]
+ getUnLibro(String) : String
+ getUnLibroDePrestados(String) : String
+ getUnLibro(int, int) : String
+ getLibrosXGenero(int) : int
+ altaLibro(String, String, String, char) : boolean
+ altaLibro(String, String, String, char, double, String) : Boolean
- altaLibro(Libro) : bolean
+ eliminarLibro(String) : boolean
+ agregarCita(String, String, int) : boolean
+ isPrestado(String) : boolean
+ leer(String)
+ calificar(String, double)
+ setOpinion(String, String)
+ getCitas(String) : String
+ prestar(String, String)
+ regresar(String)
+ getUnLibroObjeto(String) : Libro
+ librosUnAutor(String) : String
+ librosConXCali(double): String
+ librosConMasDeXCalif(double) : String
+ buscarEnPrestados(String) : Libro
+ librosLeidos() : String
+ librosNoLeidos() : String
+ librosPrestados()String
+ libroFavorito() : String
+ librosEnCadaGenero() : String
+ librosLeidos(String) : String
+ librosNoLeidos(String) : String
+ libroFavorito(String) : String
+ librosPrestados(String) : String
+ cuantosLibrosGenero(String) : int
+ toString() : String
+ compareTo(Librero) : int
+ equals(Object) : boolean
+ hashCode(): int

Notemos que MAX_GENEROS y MAX_LIBROS tienen accesibilidad de paquete, pues son necesarios para construir la vista.

LibreroVista
serialVersionUID : long = 1L ~ librero : Librero ~ screenSize : Dimension ~ labelTitulo : JLabel ~ labelAutor : JLabel ~ labelGenero : JLabel ~ labelFormato : JLabel ~ labelLeido : JLabel ~ labelCalif : JLabel ~ labelOpinion : JLabel ~ txtTitulo : JTextField ~ txtAutor : JTextField ~ txtCalif : JTextField ~ boxGenero : JComboBox ~ radioLeido : JRadioButton ~ radioNoLeido : JRadioButton ~ radioDigital: JRadioButton ~ radioFisico : JRadioButton ~ grupoLeidos: ButtonGroup ~ grupoFormato: ButtonGroup ~ botonAltaLibro: JButton ~ botonEliminaLibro: JButton ~ botonAltaDesdeArchivo : JButton ~ txtOpinion: JTextArea ~ txtLibrero: JTextArea ~ libritos: JButton[][] ~ scrollLibrero : JScrollPane ~ labelBuscaPorTitulo: JLabel ~ labelLibrosUnAutor : JLabel ~ labelLibrosXEstrellas : JLabel ~ labelBuscaPrestados : JLabel ~ txtBuscaPorTitulo : JTextField ~ txtBuscaPorAutor: JTextField ~ txtBuscaXEstrellas: JTextField ~ txtPrestados : JTextField ~ botonBuscaTitulo : JButton

~ botonBuscaEnPrestados : JButton ~ botonBuscaAutor : JButton ~ botonBuscaXEstrellas : JButton ~ botonBuscaMasEstrellas : JButton ~ botonBuscaPrestados : JButton ~ botonArchivo : JButton ~ botonLeidos : JButton ~ botonNoLeidos : JButton ~ botonPrestados : JButton ~ botonLibroFavorito : JButton ~ botonLibrosXGenero : JButton ~ botonTodoMiLibrero: : JButton ~ areaInformacion: JTextArea ~ scroll: JScrollPane ~ boxGeneroMetodos : JComboBox ~ botonLeidosG : JButton ~ botonNoLeidosG : JButton ~ botonPrestadosG : JButton ~ botonLibroFavoritoG : JButton ~ botonCuantosLibros: JButton
+ LibreroVista(String, Librero)

LibroVista
– serialVersionUID : long. = 1L ~ infoLibro : JTextArea ~ panelInfo : JPanel ~ botonPrestar : JButton ~ botonLeer : JButton ~ botonImprimeCitas : JButton ~ citas : JTextArea ~ citasScroll : JScrollPane ~ labelPag : JLabel ~ labelCita : JLabel ~ txtPag : JTextField ~ txtCita : JTextArea ~ botonAgregarCita : JButton ~ libreroDeTrabajo : Librero
+ LibroVista(String, Librero)

ManejadorArreglosGenerico
+ posMayor(T[], int) : int + posMenor(T[], int) : int + posMenor(T[], int, int) : int + cuantosMayX(T[], int, T): int + cuantosMenX(T[], int, T) : int + cuantosIgualX(T[], int, T): int + cualesMayX(T[], int, T) : ArrayList<Integer> + cualesMenX(T[], int, T) : ArrayList < Integer> + swap(T[], int, int) + invierte(T[], int) + unCorrimientoDer(T[], int, int) + unCorrimientoIzq(T[], int, int) + busSecDes(T[], int, T) : int + busSecOrd(T[], int, T) ; int + seleccionDirecta(T[], int) + busquedaBinaria(T[], int, T) : int + esCapicuo(T[], int) : boolean + union(T[], int, T[], int) : ArrayList<T> + interseccion(T[], int, T[], int) : ArrayList<T> + abrirEspacio(T[], int, int) : int + agregaDatos(T[], int, ArrayList<T>) : int + insertaEnOrden(T[], int, T) : int + eliminaEnOrden(T[], int, T) : T + eliminaEnDesorden(T[], int, T) : T + eliminaPos(T[], int, int) : T + insertaAlFinal(T[], int, T) : int + eliminaRepetidos(T[], int) : int

ManejadorMatricesGenerico
+ matrizString(T[][], int, int) : String + maxFila(T[][], int, int) : int + minFila(T[][], int, int) : int + maxColumna(T[][], int, int) : int + minColumna(T[][], int, int) : int + posMayor(T[][], int, int) : int[]

+ posMenor(T[][], int, int) : int[] + traspuesta(T[][], int) + esSimetrica(T[][], int) : boolean + buscaPorRenglon(T[][], int, int, T) : int + buscaPorColumna(T[][], int, int, T) : int + buscaMatriz(T[][], int, int, T) : int[] + buscaMatriz(T[][], int, int[], T): int[] + sonIguales(T[][], int, int, T[][], int, int) : boolean

4. Implementación

a) Descripción completa de la implementación

Para implementar el programa se utilizó Java, trabajando con Eclipse de IDE. La mayoría de los métodos son modificaciones a los algoritmos estándares de búsqueda y filtro. Dos de los algoritmos más importantes utilizados son la búsqueda en la matriz de libros y las búsquedas por arreglos. Para estos se utilizó el tanto el ManejadorArreglosGenerico como el ManejadorMatricesGenerico. A continuación, se presenta el código de las búsquedas que estas clases implementan:

```
public static <T> int busSecDes(T[] a, int n, T x) {
        int i = 0;
        int pos;

        while(i<n && !a[i].equals(x))
                i++;

        if(i<n) //if (i ==n)
                pos = i;
        else
```

```
                pos = -1;
            return pos;
        }
```

Para el método de búsqueda del ManejadorMatricesGenerico, se realizó una pequeña modificación. Debido a que cada género (representado por un renglón de la matriz de libros del librero) podía tener un número diferente de libros, el método no podía recibir una única "n" que le indicara cuantas columnas recorrer. El método modificado recibe un arreglo de enteros que le indican, por renglón, hasta que columna recorrer. Esto porque en el librero existe el arreglo librosXGenero que funciona como un arreglo de contadores de libros por género. A continuación, el código:

```
public static <T> int[] buscaMatriz(T[][] mat, int m, int[] nPorFila, T x) {
        int j = 0;
        int pos = -1;
        int[] res = {-1,-1};

        while(j < m && pos==-1) {
            pos = ManejadorArreglosGenerico.busSecDes(mat[j],
nPorFila[j], x);
            j++;
        }

        if(pos != -1) {
            res[0] = j-1;
            res[1] = pos;
```

```
        }
        return res;
    }
```

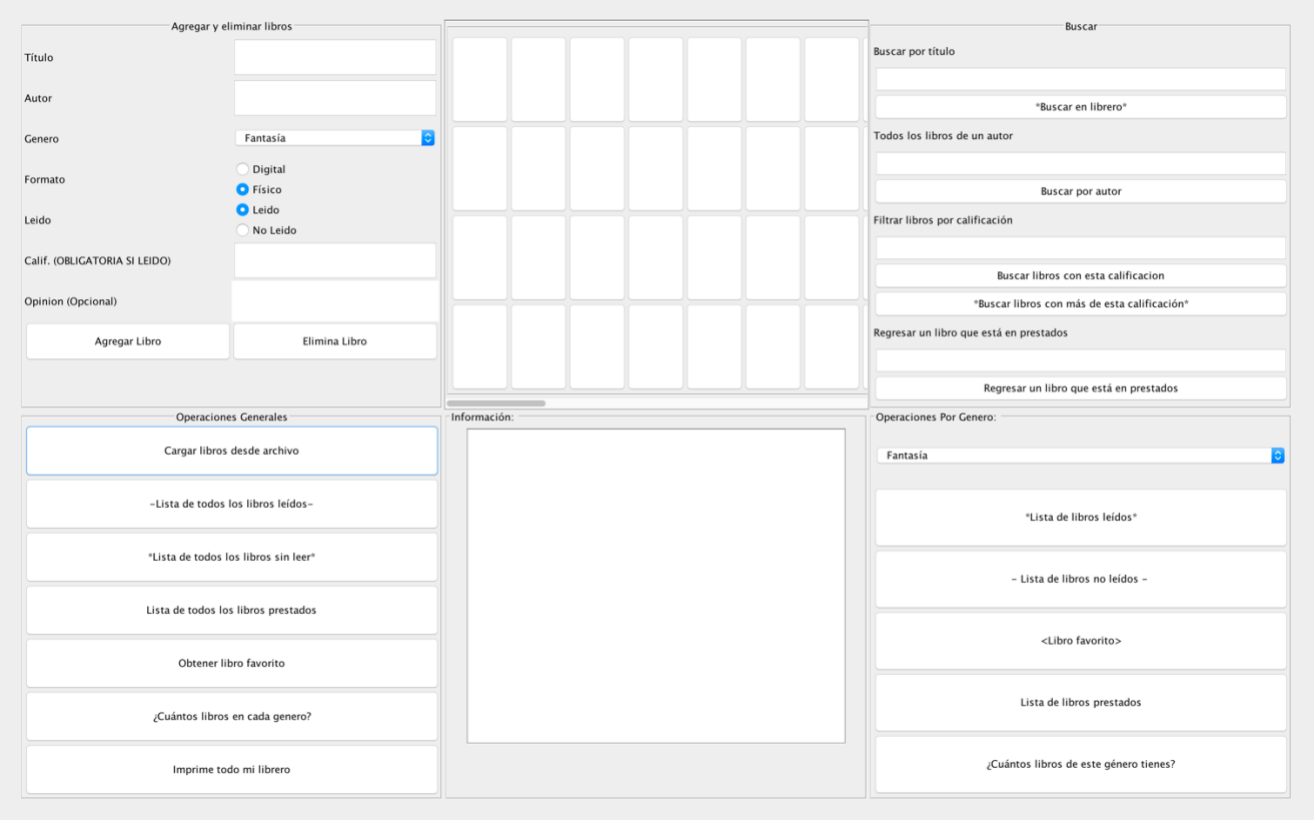
La mayor parte del programa funcionaba a través de los métodos del librero. La mayoría de los métodos de esta clase siguen el mismo proceso: buscan un libro en la matriz o en el arreglo, o bien recorren la matriz seleccionando libros que cumplen con algunas características. Todos los métodos se incluyen en el código anexado, pero aquí haremos hincapié al método privado altaLibro(Libro). Este método solamente se utiliza para regresar un libro del arreglo de libros prestados al librero original. Dado que el libro ya existe y ya fue creado, no es necesario obtener los datos, si no que solamente se transfiere el objeto de una dirección a otra. El método se llama dentro del método regresar(String) del librero. A continuación, se presenta el código de este método:

```
private boolean altaLibro(Libro libro) {
        //Alta que solo se usa para mover un libro de Prestados al
Librero, porque el libro ya está creado y viviendo en prestados
        boolean res = false;
        int indiceGen;

        indiceGen = ManejadorArreglosGenerico.busSecDes(generos,
MAX_GENEROS, libro.getGenero());

        if(indiceGen != -1 && librosXGenero[indiceGen] < MAX_LIBROS)
{
            libro.regresar();
            libros[indiceGen][librosXGenero[indiceGen]] = libro;
            librosXGenero[indiceGen]++;
```

```
                res = true;
            }

            return res;
        }
```

b) GUI

El programa tiene dos interfaces gráficas. La primera, que es la interfaz inicial, muestra el librero junto con todas las posibles operaciones que se pueden realizar sobre el librero. Aquí se presenta una imagen de la interfaz y una breve descripción de cada zona. En la sección “Manual del usuario” se explicará como utilizar cada sección.

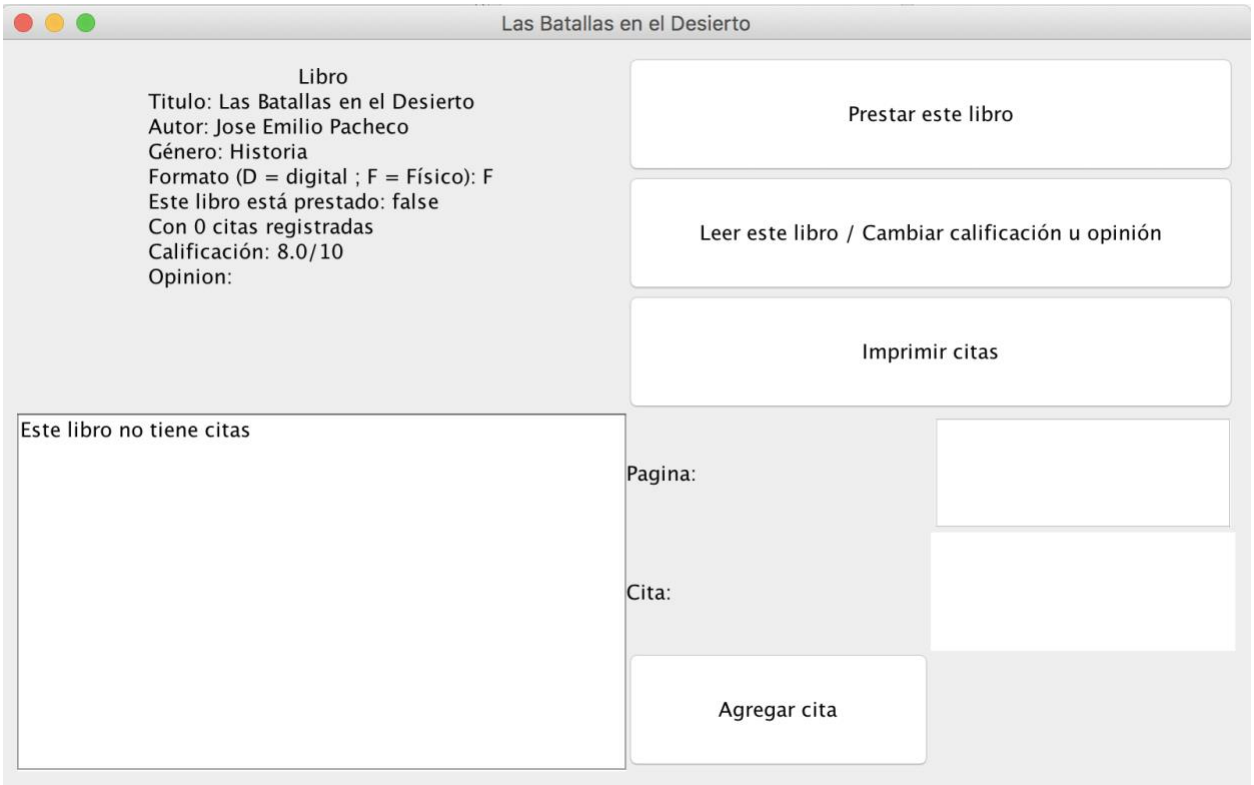


Explicación de cada sección (de izquierda a derecha, por filas):

- Agrega y elimina libros: En esta sección se pueden dar de alta nuevos libros, o eliminar libros existentes en la biblioteca
- Librero: Esta zona muestra todos los libros en el librero, cada renglón de botones representa un género. Se puede hacer click sobre el libro para desplegar una nueva ventana.
- Búsquedas: En esta sección se pueden buscar libros por título, por autor, filtrar por alguna calificación o regresar un libro prestado
- Operaciones generales: Aquí se pueden aplicar filtros sobre todo el librero, por ejemplo, obtener todos los libros leídos, o los libros prestados.
- Ventana de información: Aquí se despliega la información que se solicita

– Operaciones por género: Se llevan a cabo las mismas operaciones que en las operaciones generales, pero trabajando únicamente con un solo género.

La segunda interfaz que se presenta es la que se despliega cuando se accede a un libro, ya sea presionando el botón del librero o buscándolo a través de las búsquedas:



En esta interfaz se incluyen los datos del libro seleccionado. Se presentan las opciones para prestar el libro, leerlo y asignarle una calificación o cambiar la existente, e imprimir las citas del libro. También se presenta una sección para agregar una cita al libro.

c) Manual de usuario

Una vez iniciado el programa, se pueden **dar de alta libros** de manera manual o a través de un archivo. A continuación se explica como hacer ambos:

- A mano:
    - Ingrese los datos del libro que desea dar de alta. Es indispensable para cualquier libro proporcionar autor, título, género y seleccionar el formato.
    - Si ya leyó el libro, seleccione la opción “Leído”. Debe ingresar obligatoriamente una calificación para el libro
    - Si ya leyó el libro, puede añadir una opinión sobre el mismo. Esta es opcional, si no desea agregar una, deje esta casilla en blanco.
- A través de un archivo:
    - Seleccione la opción “Cargar libros desde archivo”.
    - Seleccione el archivo que contiene sus libros. Debe ser de tipo TXT y debe estar acomodado de la misma forma que el archivo muestra que se incluye en el anexo.
- Nota: Está permitido dar de alta un libro del mismo título más de una vez, se asume que esto significa que se tienen dos copias del mismo.

Una vez dada de alta sus libros, puede realizar las siguientes operaciones:

- **Eliminar un libro:** Ingrese el título del libro que desea eliminar en la sección de “Agrega y elimina libros”. Presione “Eliminar libro”.
- **Buscar un libro por título:** En la sección de “Búsquedas” escriba el libro que desea buscar. Si está en su librero, se abrirá una nueva ventana con la información del libro
- **Buscar libros por autor:** En la sección “Búsquedas”, escriba el nombre del autor que desea buscar. La información se desplegará en la ventanilla de información.

- **Filtrar libros por calificación:** Ingrese una calificación numérica en la ventana correspondiente de “Búsquedas”. Puede buscar libros con esa calificación, o buscar libros que tengan más de esa calificación.
- **Prestar un libro:** Seleccione, ya sea desde su librero o a través de una búsqueda, el libro que desea prestar. Seleccione la opción “Prestar este libro” en la ventanilla que se despliega. Ingrese el nombre de la persona a la que se le presta el libro.
- **Regresar un libro:** Para regresar un libro de sus libros prestados a su librero, primero verifique que esté en sus libros prestados con la opción “Lista de todos los libros prestados”. Introduzca el título en la ventana de “Regresar un libro prestado” y presione el botón.
- **Agregar una cita a un libro:** Seleccione el libro al que desea agregar una cita. Se abrirá la ventana del libro. Inserte la página de la cita (solo el número) y el texto y presione “Agregar cita”. Puede agregar hasta 20 citas por libro.
- **Marcar un libro como leído / cambiar su calificación:** Una vez haya leído un libro, puede añadirle una calificación y opinión, marcándolo como leído. Acceda a la ventana del libro a través de su librero, y escoja la opción “Leer / cambiar calificación”. Ingrese la nueva calificación (obligatoria) y opinión (opcional).
- **Filtrar su librero:** Puede filtrar su librero, ya sean todos los libros o por genero, utilizando los botones de la sección “Operaciones generales” y “Operaciones por género”.

5. Pruebas y resultados

a) Descripción de las pruebas realizadas (casos de pruebas)

Para probar mi proyecto, realicé una serie de pruebas de los diferentes casos posibles. Comencé probando la clase más básica, el libro. Probé el obtener sus datos, agregar citas, y obtener estas citas. Me aseguré que esta clase funcionara perfectamente, pues era la base de mi proyecto. De ahí proseguí a probar la clase Librero. Esta clase fue más complicada de probar, pues tenía que probar tanto los métodos "externos" (búsquedas, filtros), como los internos, que trabajaban con los objetos de libros. Probé varios a través de la consola, y me aseguré que funcionaran. Otros, los probé una vez construido el GUI, pues son métodos que trabajaban con el GUI para generar los botones o para crear las ventanas individuales de los libros. En la sección de código se incluyen los *mains* con los que se probó la clase libro y la clase librero.

b) Resultados y ajustes

Las pruebas me sirvieron principalmente para darme cuenta de errores de dedos que tenía en mis métodos. Sin embargo, hubo algunas que me obligaron a cambiar mi diseño o a crear nuevos métodos para que el programa funcionara. Un ejemplo es el método modificado de búsqueda en matriz que se incluye en una sección anterior. Éste lo cree debido a que me di cuenta que mis búsquedas no estaban funcionando, porque mis parámetros de columna eran diferentes para cada renglón de la matriz.

6. Conclusiones

En suma, logré tener un proyecto que resolviera mi problemática inicial. El sistema "misLibros" funciona como yo quiero que funcione. Tiene funcionalidad que me es útil, y cumple con los requisitos que yo considero un sistema de organización de libros debe tener. Realizar este proyecto desde cero me enseñó todo lo que implica construir un producto de software. Desde la etapa de

diseño, donde definí qué iba a hacer y que funcionalidad tendría, aprendí cosas nuevas. Durante la implementación, tuve que hacer cambios sobre la marcha para ajustarme a los contratiempos que surgían, y aprendí el valor de escribir buen código para hacer los cambios fáciles y rápidos. Usar programación orientada a objetos me permitió aprender sobre la distribución de responsabilidades, encapsulamiento y comunicación entre clases. En conclusión, creo que este proyecto fue exitoso en utilizar todos los conocimientos que obtuve durante el semestre para construir mi primera pieza de software que va más allá de las tareas, y llega a ser un producto que puedo utilizar en mi vida diaria.

7. Código