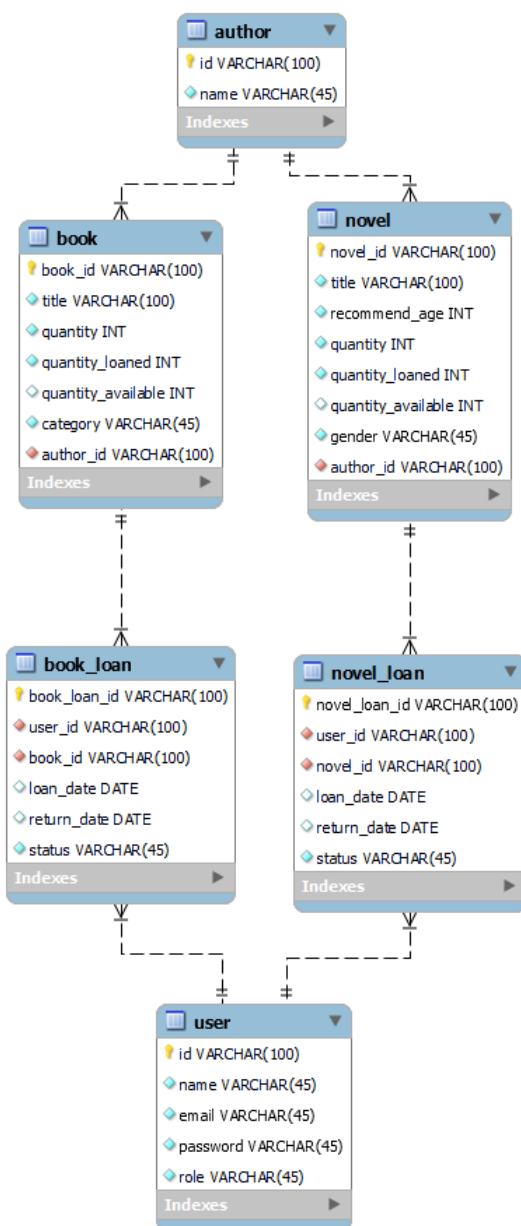


Diseño base de datos

Se identificaron las siguientes entidades:

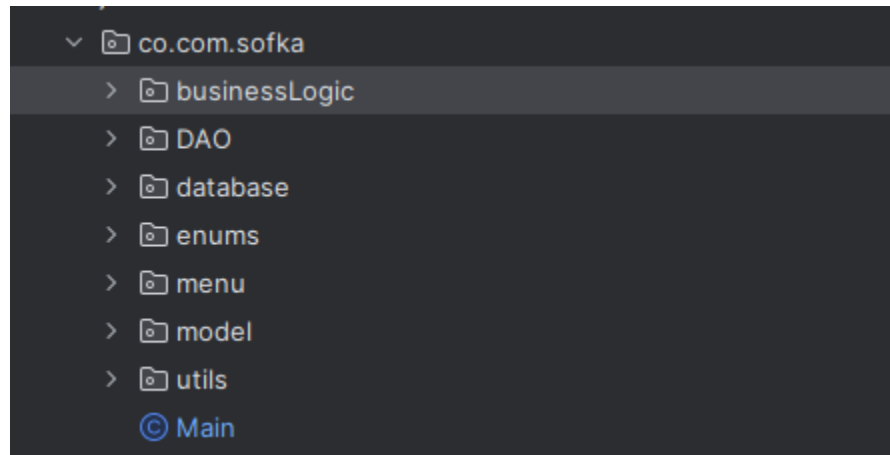
- Novela
- Libro
- Autor
- Usuario (administrador/asistente/lector)
- Prestamo_novela
- Prestamo_libro














Y se realizó el siguiente modelo:



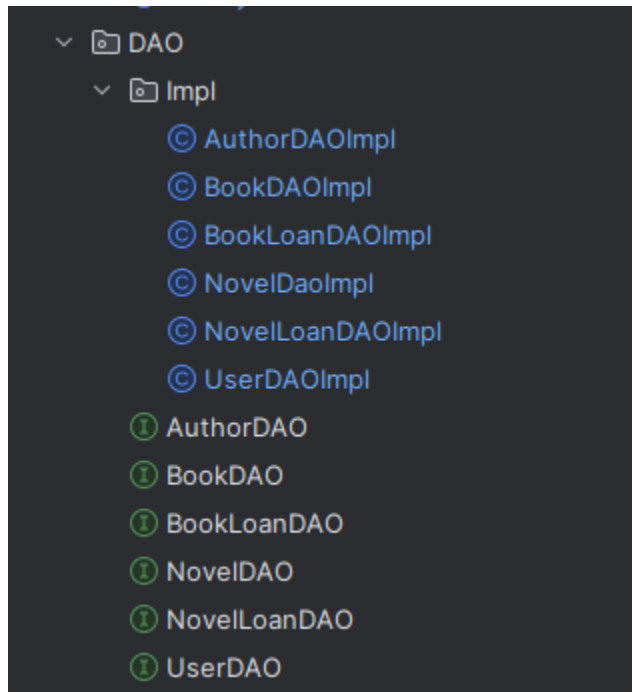
Estructura del proyecto

- businessLogic
 - Logica del administrador
 - Logica del asistente.
 - Logica del lector
- DAO (Objetos de transferencia de datos y sus interfaces)
 - BookDAO
 - NovelDAO, etc.
- Database (conexión con la base de datos)
- Enums (enumeradores para los estados del préstamo y los roles)
- Menu (mensajes de interfaz de usuario)
- Model (modelado de las entidades del sistema)
 - Book
 - Novel
 - User, etc
- Utils (clases que serán útiles independientemente del contexto)



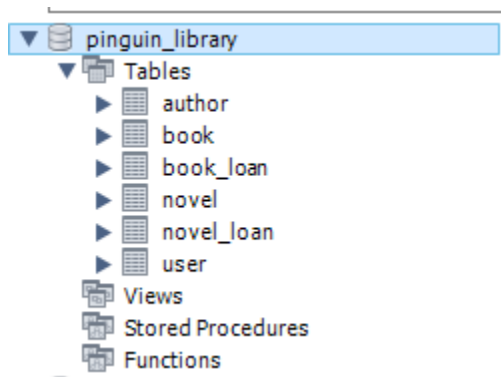
- ▼  businessLogic
 - >  administrator
 - >  assistant
 - >  generalAdmin
 - >  reader
 - © Authentication
 - © Library
- ▼  DAO
 - >  Impl
 - ① AuthorDAO
 - ① BookDAO
 - ① BookLoanDAO
 - ① NovelDAO
 - ① NovelLoanDAO
 - ① UserDAO
- ▼  database
 - >  mysql
 - ① DataBase
 - © DatabaseConfigConstants
- ▼  enums
 - © LoanStatus
 - © UserType
- ▼  menu
 - © MenuConstant
 - © MenuMessage
- ▼  model
 - © Author
 - © Book
 - © BookLoan
 - © Item
 - © Manuscript
 - © Novel
 - © NovelLoan
 - © User
- ▼  utils
 - © CsvAuthorManagement
 - © CsvBookManagement

Patrón de diseño utilizado:
DAO (Data Access Object)



Conexión a la base de datos:

Base de datos creada:



Conexión desde java usando variables de entorno:

Variables de entorno

Variables del sistema	
Variable	Valor
GRADLE_DB	penguin_library
GRADLE_PASS	johan2233Z.
GRADLE_SERVER	localhost
GRADLE_USER	root

Llamado desde Java usando System.getenv();

```
package co.com.sofka.database;
2 usages  👤 johan +1
public class DatabaseConfigConstants {
    1 usage
    public static final String SERVER = System.getenv( name: "GRADLE_SERVER");
    1 usage
    public static final String DATA_BASE_NAME = System.getenv( name: "GRADLE_DB");
    1 usage
    public static final String USER = System.getenv( name: "GRADLE_USER");
    2 usages
    public static final String PASSWORD = System.getenv( name: "GRADLE_PASS");
}
```

Ejemplo de usos:

Se implementa login y registro de usuario funcional:

El login tiene la finalidad de filtrar el tipo de acceso e información que tendrá cada tipo de usuario (administrador/asistente/lector)

```
Welcome to Pinguin Library
1. Login
2. Register
3. Exit
Enter your option: 1
```

Manejo de reconocimiento de correo registrado:

Se filtra primero buscando en la base de datos algun correo asociado

```
Welcome to Pinguin Library
1. Login
2. Register
3. Exit
Enter your option: 1
Enter your email: correoIncorrecto no registrado
Enter your password: abcde
Incorrect email. Please try again.
Enter your email:
```

Manejo de 3 intentos de contraseña incorrecta máximo:

Después de encontrar el correo y que se pruebe que el usuario está registrado en el sistema, se hará verificación de la contraseña

```
1. Login
2. Register
3. Exit
Enter your option: 1
Enter your email: correoIncorrecto
Enter your password: abcde
Incorrect email. Please try again.
Enter your email: prueba@example.com correcto
Enter your password: contrasenaIncorrecta incorrecto
Incorrect password. Please try again.
You have: 2 attempts left 2 intentos mas
Enter your email:
```

Redirección a menu personalizado dependiendo del tipo de usuario:

```

switch (option){
    case 1:
        User user = login();
        if (user != null) redirectPersonalizedMenu(user);
        else System.exit( status: 0);
        break;

```

Lleva a su respectivo menu dependiendo del rol:

```

private static void redirectPersonalizedMenu(User user) {
    switch (user.getRole()){
        case READER:
            readerMenu(user);
            break;
        case ADMINISTRATOR:
            administratorMenu(user);
            break;
        case ASSISTANT:
            assistantMenu(user);
            break;
    }
}

```

Menú personalizado para (Lector)

Tienes las opciones de acceder a

- Libro
 - Mirar todos los libros
 - Mirar libro por nombre
 - Reservar libro
 - Devolver reserva
- Novela
 - Métodos similares al del libro, se omitirá por simplicidad
- Author
 - Mirar autores y sus obras

```
Login successful. Welcome, prueba!  
Welcome to Main Menu, prueba  
=====
```

1. Book options
2. Novel options
3. Author options
4. Back

Menú de libros para el usuario:

```
1. Book options  
2. Novel options  
3. Author options  
4. Back  
Enter your option: 1
```

```
Welcome to Book Menu, prueba  
=====
```

1. See all books
2. See book by name
3. Loan book
4. See my book loans
5. Return book loan
6. Back

```
Enter your option:
```

opción(1):

Retorna una lista descriptiva de todos los libros cuya cantidad disponible es mayor que 0


```
1. See all books
2. See book by name
3. Loan book
4. See my book loans
5. Return book loan
6. Back
Enter your option: 1
Book title: Cien annos de soledad
Book author: Gabriel García Márquez
available quantity: 6

Book title: Harry Potter y la piedra filosofal
Book author: J.K. Rowling
available quantity: 7

Book title: El resplandor
Book author: Stephen King
available quantity: 14

Book title: Tokio blues
Book author: Haruki Murakami
available quantity: 19

Book title: Asesinato en el Orient Express
Book author: Agatha Christie
available quantity: 11
```

opción(2):
Buscar libro disponible por nombre

```
1. See all books
2. See book by name
3. Loan book
4. See my book loans
5. Return book loan
6. Back
Enter your option: 2
Enter the title: Cien annos de soledad
Book{category='Ficcion', title='Cien annos de soledad', author=Author{id='1', name='Gabriel García Márquez'}, id='1', quantity=10}
```

opción(3) : reservar libro:

1. See all books
2. See book by name
3. Loan book
4. See my book loans
5. Return book loan
6. Back

Enter your option: 3

Enter book name:

Cien annos de soledad

Al momento de reservar un libro, este descuenta su stock disponible automáticamente a través de la siguiente lógica:

Aunque en la base de datos se actualiza automáticamente la cantidad disponible, esto se refuerza a través del uso de Java

```
public void loanBook(User user, Book book){
    book.setQuantityLoaned(book.getQuantityLoaned() + 1);
    book.setQuantityAvailable(book.getQuantityAvailable() - 1);
    bookDAO.updateBook(book);
    BookLoan bookLoan = new BookLoan(UUID.randomUUID().toString(),
        user,
        book,
        loanDate: null,
        returnDate: null,
        LoanStatus.REQUESTED);
    bookLoanDAO.insertBookLoan(bookLoan);
}
```

El registro aparecerá en la opción (4)

La fecha de reserva y de retorno se generan cuando un asistente o administrador acepte la petición de libro

```
=====
1. See all books
2. See book by name
3. Loan book
4. See my book loans
5. Return book loan
6. Back
Enter your option: 4
Book loan id: 63ddc511-ab23-49fe-9f86-b5052dfbaaf1
Book id: Cien annos de soledad
Book loan user: prueba@example.com
Book loan date: null
Book return date: null
Book loan status: REQUESTED
```

opción(5): devolver el préstamo

– se puede implementar una vez haya sido aceptado el préstamo por un asistente o administrador

Después de que un asistente o administrador acepta nuestra solicitud (más adelante se acepta con el ejemplo de administrador)

Nuestra solicitud se ve así:

(El nombre del libro también cambio ya que más adelante en las operaciones crud se modificó este libro desde el perfil del administrador)

```
1. See all books
2. See book by name
3. Loan book
4. See my book loans
5. Return book loan
6. Back
Enter your option: 4
Book loan id: 63ddc511-ab23-49fe-9f86-b5052dfbaaf1
Book id: Cien annos de soledad prueba
Book loan user: prueba@example.com
Book loan date: 2024-04-16
Book return date: 2024-05-01
Book loan status: COMPLETED
```

Ahora, vamos a devolverlo:

```
Welcome to Book Menu, prueba
=====
1. See all books
2. See book by name
3. Loan book
4. See my book loans
5. Return book loan
6. Back
Enter your option: 5
Enter book loan id:
63ddc511-ab23-49fe-9f86-b5052dfbaaf1
```

Y ahora el estado del préstamo es:

```
Enter your option: 4
Book loan id: 63ddc511-ab23-49fe-9f86-b5052dfbaaf1
Book id: Cien annos de soledad prueba
Book loan user: prueba@example.com
Book loan date: 2024-04-16
Book return date: 2024-05-01
Book loan status: FINISHED
```

– Las opciones con novela son similares, se omitirá por simplicidad

```
1. Book options
2. Novel options
3. Author options
4. Back
Enter your option: 2
Welcome to Novel Menu, prueba
=====
1. See all novels
2. See novel by name
3. Loan novel
4. See my novel loans
5. Return novel loan
6. Back
```

Opciones de autor:

Menú personalizado acciones relacionadas con el autor:

```
=====
```

```
1. Book options
```

```
2. Novel options
```

```
3. Author options
```

```
4. Back
```

```
Enter your option: 3
```

```
Welcome to Author Menu, prueba
```

```
=====
```

```
1. See all authors
```

```
2. See author by name
```

```
3. Back
```

```
Enter your option:
```

```
1
```

opción(1): mostrar todos los autores y sus libros relacionados, esto se logra gracias al uso de collections y funciones de agrupación

```
Welcome to Author Menu, prueba
```

```
=====
```

```
1. See all authors
```

```
2. See author by name
```

```
3. Back
```

```
Enter your option:
```

```
1
```

```
Author: Stephen King
```

```
Books:
```

```
- El resplandor
```

```
Author: Gabriel García Márquez
```

```
Books:
```

```
- Cien años de soledad
```

```
Author: J.K. Rowling
```

```
Books:
```

```
- Harry Potter y la piedra filosofal
```

```
Author: Haruki Murakami
```

```
Books:
```

```
- Tokio blues
```

Se agrupa los datos por autor y se listan los libros que fueron escritos por este autor (code):

```
private static void seeAuthorsAndTheirBooks() {
    Map<Author, List<Book>> authorsAndTheirBooks = readerManagement.getAllAvailableBooks() List<Book>
        .stream() Stream<Book>
        .collect(Collectors.groupingBy(Book::getAuthor));
    authorsAndTheirBooks.forEach((author, books) -> {
        System.out.println("Author: " + author.getName());
        System.out.println("Books: ");
        books.forEach(book -> System.out.println("- " + book.getTitle()));
        System.out.println();
    });
}
```

– También se puede buscar autor nombre, se omite por simplicidad

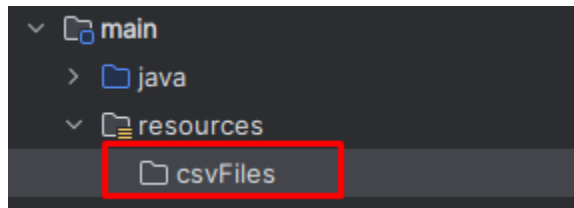
MENÚ DEL ADMINISTRADOR (el de asistente es igual, menos por la parte de manejo de usuarios)

```
> Task :Main.main()
Welcome to Pinguin Library
1. Login
2. Register
3. Exit
Enter your option: 1
Enter your email:
pinguino@example.com
Enter your password:
pinguino123
Login successful. Welcome, pinguino!
Welcome to Main Menu, pinguino
=====
1. Book options
2. Book loan options
3. Novel options
4. Novel loan options
5. Author options
6. User options
7. Back
Enter your option:
```

credenciales administrador

Opciones administrador

opción(1): cuenta con todas las operaciones crud con respecto a un libro y además, se puede exportar la información de los libros a un archivo csv que estará en la siguiente carpeta:



```
Welcome to Main Menu, pinguino
=====
1. Book options
2. Book loan options
3. Novel options
4. Novel loan options
5. Author options
6. User options
7. Back
Enter your option: 1

Welcome to Book Menu, pinguino
=====
1. Create book
2. See all books
3. See book by name
4. Update book
5. Delete book
6. Export book data in csv file
7. Back
Enter your option: |
```

opcion(1): crear libro

Libro creado:

```

1. Create book
2. See all books
3. See book by name
4. Update book
5. Delete book
6. Export book data in csv file
7. Back
Enter your option: 1
Enter book title: Prueba
Enter author's name: Stephen King
Enter category: suspense
Enter quantity: 80
Enter quantity loaned: 10

```

Opción(2) se ven todos los libros:

```

=====
1. Create book
2. See all books
3. See book by name
4. Update book
5. Delete book
6. Export book data in csv file
7. Back
Enter your option: 2
Book{category='Ficción', title='Cien annos de soledad', author=Author{id='1', name='Gabriel Garcoa Morquez'}, id='1', quantity=10, quantityLoan
Book{category='horror', title='prueba', author=Author{id='2', name='J.K. Rowling'}, id='10', quantity=10, quantityLoaned=0, quantityAvailable=1
Book{category='horror', title='prueba2', author=Author{id='2', name='J.K. Rowling'}, id='11', quantity=1, quantityLoaned=0, quantityAvailable=1
Book{category='Fantasia', title='Harry Potter y la piedra filosofal', author=Author{id='2', name='J.K. Rowling'}, id='2', quantity=8, quantityL
Book{category='Terror', title='El resplandor', author=Author{id='3', name='Stephen King'}, id='3', quantity=15, quantityLoaned=1, quantityAvail
Book{category='Drama', title='Tokio blues', author=Author{id='4', name='Haruki Murakami'}, id='4', quantity=20, quantityLoaned=1, quantityAvail
Book{category='Misterio', title='Asesinato en el Orient Express', author=Author{id='5', name='Agatha Christie'}, id='5', quantity=12, quantityL
Book{category='horror', title='1886', author=Author{id='6', name='Ernest Hemingway'}, id='c15cac91-06b0-4135-a004-8927d58bb457', quantity=10, q
Book{category='suspense', title='Prueba', author=Author{id='3', name='Stephen King'}, id='d0e8d0b2-5019-409a-b918-e251f451512d', quantity=80, q

```

libro creado

Opción (3): buscar libro por nombre


```

=====
1. Create book
2. See all books
3. See book by name
4. Update book
5. Delete book
6. Export book data in csv file
7. Back
Enter your option: 3
Enter book title: Cien annos de soledad
Book{category='Ficción', title='Cien annos de soledad', author=Author{id='1', name='Gabriel García Márquez'}, id='1', quantity=10, quantityLoaned=5, quantityAvailable=5}
Welcome to Book Menu, pinguino
=====

```

Opción(4) editar libro:

Se busca el libro por id, seguido a eso se ingresan los nuevos datos del libro, el id no cambia

```

Welcome to Book Menu, pinguino
=====
1. Create book
2. See all books
3. See book by name
4. Update book
5. Delete book
6. Export book data in csv file
7. Back
Enter your option: 4
Enter book id: 1
Enter book title: Cien annos de soledad prueba
Enter author's name: Stephen King
Enter category: horror
Enter quantity: 100
Enter quantity loaned: 3
Welcome to Book Menu, pinguino

```

Resultado:

```

Enter your option: 4
Book{category='horror', title='Cien annos de soledad prueba', author=Author{id='3', name='Stephen King'}, id='1', quantity=100, quantityLoaned=3, quantityAvailable=5}
Welcome to Book Menu, pinguino

```

opción(6) exporta la información de los libros a un csv:

Usando apache commons csv:

Code: se crea el archivo usando FileWriter y se escribe sobre él usando CSVPrinter

```

1 usage  ~johan *
public class CsvBookManagement {
    1 usage
    private static final BookDAO bookDAO = new BookDAOImpl();

    2 usages  ~johan *
    public static void exportBooksData() {

```

```

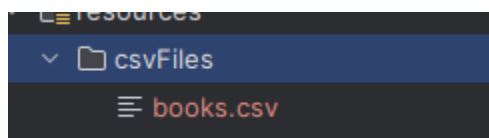
1 usage  👤 johan
private static void writeBooksToCsv(List<Book> books, CSVPrinter csvPrinter) {
    for (Book book : books) {
        try {
            csvPrinter.printRecord(
                book.getId(),
                book.getTitle(),
                book.getQuantity(),
                book.getQuantityLoaned(),
                book.getQuantityAvailable(),
                book.getCategory(),
                book.getAuthor().getName());
        } catch (IOException e) {
            UserLogger.error(e.getMessage());
        }
    }
}
}
}

```

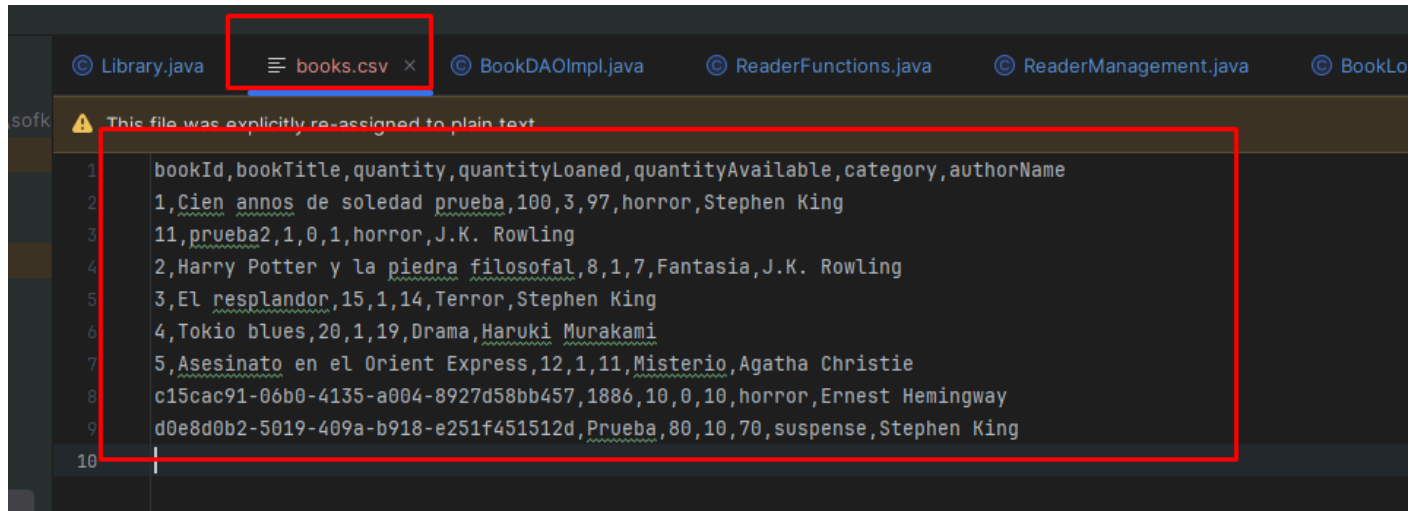
```

1. Create book
2. See all books
3. See book by name
4. Update book
5. Delete book
6. Export book data in csv file
7. Back
Enter your option: 6
Welcome to Book Menu, pinguino

```



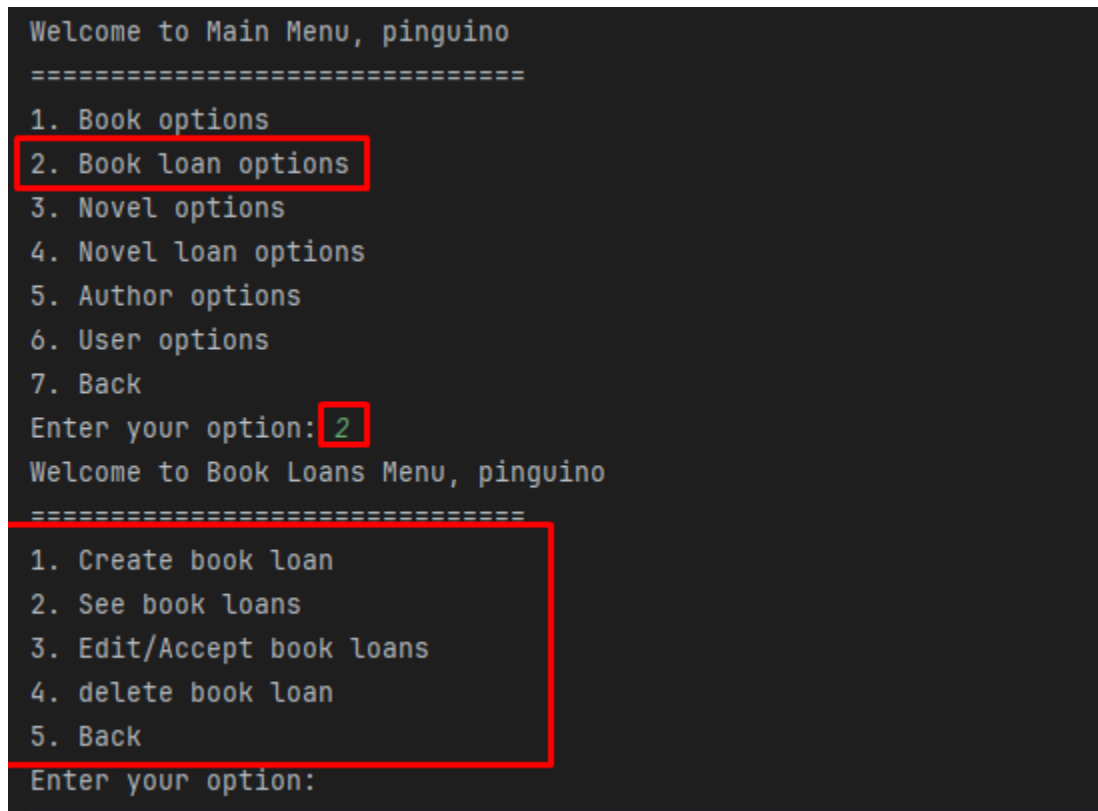
Archivo csv:



```
bookId,bookTitle,quantity,quantityLoaned,quantityAvailable,category,authorName
1,Cien annos de soledad prueba,100,3,97,horror,Stephen King
11,prueba2,1,0,1,horror,J.K. Rowling
2,Harry Potter y la piedra filosofal,8,1,7,Fantasia,J.K. Rowling
3,El resplandor,15,1,14,Terror,Stephen King
4,Tokio blues,20,1,19,Drama,Haruki Murakami
5,Asesinato en el Orient Express,12,1,11,Misterio,Agatha Christie
c15cac91-06b0-4135-a004-8927d58bb457,1886,10,0,10,horror,Ernest Hemingway
d0e8d0b2-5019-409a-b918-e251f451512d,Prueba,80,10,70,suspense,Stephen King
```

– las funciones de novela son bastantes similares, se omitiran por simplicidad

Opciones de préstamos de libros: (operaciones crud clásicas, para el ejemplo se documentara la opción 2 y 3)



```
Welcome to Main Menu, pinguino
=====
1. Book options
2. Book loan options
3. Novel options
4. Novel loan options
5. Author options
6. User options
7. Back
Enter your option: 2
Welcome to Book Loans Menu, pinguino
=====
1. Create book loan
2. See book loans
3. Edit/Accept book loans
4. delete book loan
5. Back
Enter your option:
```

Opcion 2:

Se listan las solicitudes de libros hechas por los lectores, en orden de inserción

```
=====
1. Create book loan
2. See book loans
3. Edit/Accept book loans
4. delete book loan
5. Back
Enter your option: 2
Book loan id: 03155198-4aa4-4551-b2f9-4b2ab69b64e8
Book title: Cien annos de soledad prueba
Book loan date: 2024-04-16
Book loan return date: 2024-05-01
Book loan status: COMPLETED

Book loan id: 1
Book title: 1886
Book loan date: 2021-10-24
Book loan return date: 2021-11-05
Book loan status: FINISHED

Book loan id: 2
Book title: Harry Potter y la piedra filosofal
Book loan date: 2024-04-16
Book loan return date: 2024-05-01
Book loan status: COMPLETED

Book loan id: 3
Book title: El resplandor
Book loan date: 2024-04-17
Book loan return date: 2024-05-02
Book loan status: FINISHED

Book loan id: 4
Book title: Tokio blues
Book loan date: 2024-04-18
Book loan return date: 2024-05-03
Book loan status: REQUESTED
```

Aquí se encuentra la que se creó en los primeros ejemplos usando al lector:
El id es creado aleatoriamente y las fechas no han sido definidas por que no se ha aceptado,

El nombre del libro es diferente porque en los ejemplos anteriores se modificó el libro cien annos de soledad

```
Book loan id: 63ddc511-ab23-49fe-9f86-b5052dfbaaf1
Book title: Cien annos de soledad prueba
Book loan date: null
Book loan return date: null
Book loan status: REQUESTED
```

Opción 3, aceptamos la solicitud anterior:

```
Welcome to Book Loans Menu, pinguino
=====
1. Create book loan
2. See book loans
3. Edit/Accept book loans
4. delete book loan
5. Back
Enter your option: 3
Enter book loan id:
63ddc511-ab23-49fe-9f86-b5052dfbaaf1
Edit/Accept
Accept
Welcome to Book Loans Menu, pinguino
```

Y este sería su resultado:

```
Book loan id: 63ddc511-ab23-49fe-9f86-b5052dfbaaf1
Book title: Cien annos de soledad prueba
Book loan date: 2024-04-16 fecha de aceptacion
Book loan return date: 2024-05-01 fecha 15 dias despues de aceptacion
Book loan status: COMPLETED cambio de estado a completado
```

Operaciones crud para el autor basicas:

```
Welcome to Main Menu, pinguino
=====
1. Book options
2. Book loan options
3. Novel options
4. Novel loan options
5. Author options
6. User options
7. Back
Enter your option: 5
Welcome to Author Menu, pinguino
=====
1. Create author
2. See all authors
3. See author by name
4. Update author
5. Delete author
6. Back
```

Y operaciones crud para los usuarios (SOLO LOS ADMINISTRADORES TIENEN ESTA FUNCIONALIDAD)

```
Welcome to Main Menu, pinguino
=====
1. Book options
2. Book loan options
3. Novel options
4. Novel loan options
5. Author options
6. User options
7. Back
Enter your option: 6
Welcome to User Menu, pinguino
=====
1. Create user
2. See all users
3. See user by email
4. Update user
5. Delete user
6. Back
Enter your option:
```

Se aplicó el valor agregado:

- Conexión base de datos usando variables del sistema (véase sección conexión base de datos)
- Exportar datos a archivo csv (véase manejo de libros en el menú del administrador, que también es posible hacerlo desde el menú del asistente)
- Uso de Logger (no se implementó en gran medida, solo en algunas excepciones, adjunto ejemplo:)

```

String filePath = "/src/main/resources/csv/2000/books.csv";
try (FileWriter fileWriter = new FileWriter(filePath)) {
    CSVFormat csvFormat = CSVFormat.DEFAULT
        .builder()
        .setHeader("bookId", "bookTitle", "quantity", "quantityLoa
        .build();
    CSVPrinter csvPrinter = new CSVPrinter(fileWriter, csvFormat);

    writeBooksToCsv(books, csvPrinter);
} catch (IOException e) {
    UserLogger.error(e.getMessage());
}
}

```

```

1 usage  johan
private static void writeBooksToCsv(List<Book> books, CSVPrinter csvPrinter) {
    for (Book book : books) {
        try {
            csvPrinter.printRecord(
                book.getId(),
                book.getTitle(),
                book.getQuantity(),
                book.getQuantityLoaned(),
                book.getQuantityAvailable(),
                book.getCategory(),
                book.getAuthor().getName());
        } catch (IOException e) {
            UserLogger.error(e.getMessage());
        }
    }
}

```

Y la clase es:

```

public class UserLogger {

    private static final Logger logger = Logger.getLogger(UserLogger.class.getName());

    > public static void info(String message) { logger.info(message); }

    > public static void error(String message) { logger.severe(message); }
}

```


Aspectos a mejorar:

- **Aumentar manejo de excepciones**
- **Mejorar los menu**
- **Refactorizar aun mas el codigo**

Fue un reto bastante complicado y retador, muchas gracias.