

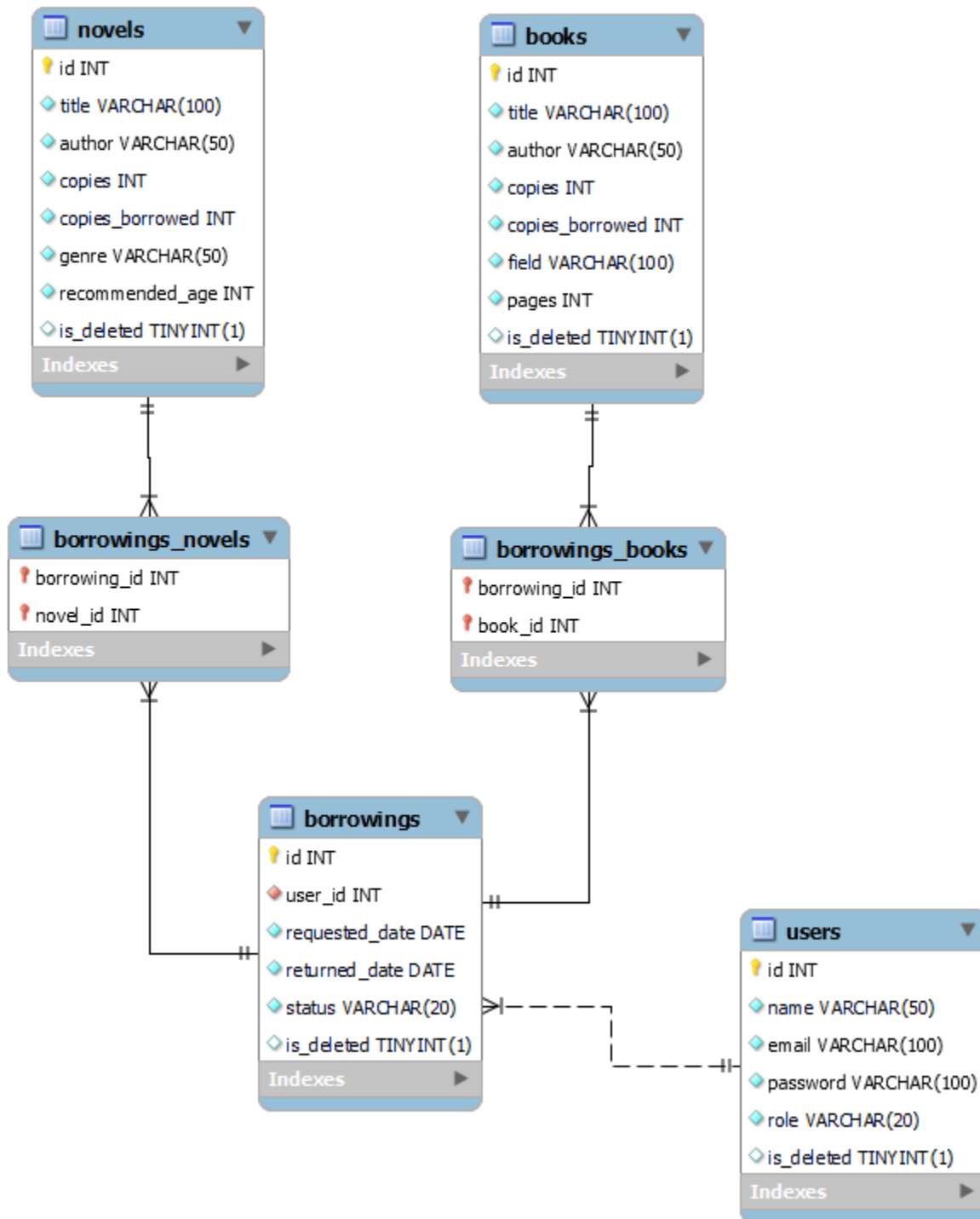
Taller Java 2:
Johan Cifuentes QA Engineer

Aspectos técnicos de la expansión

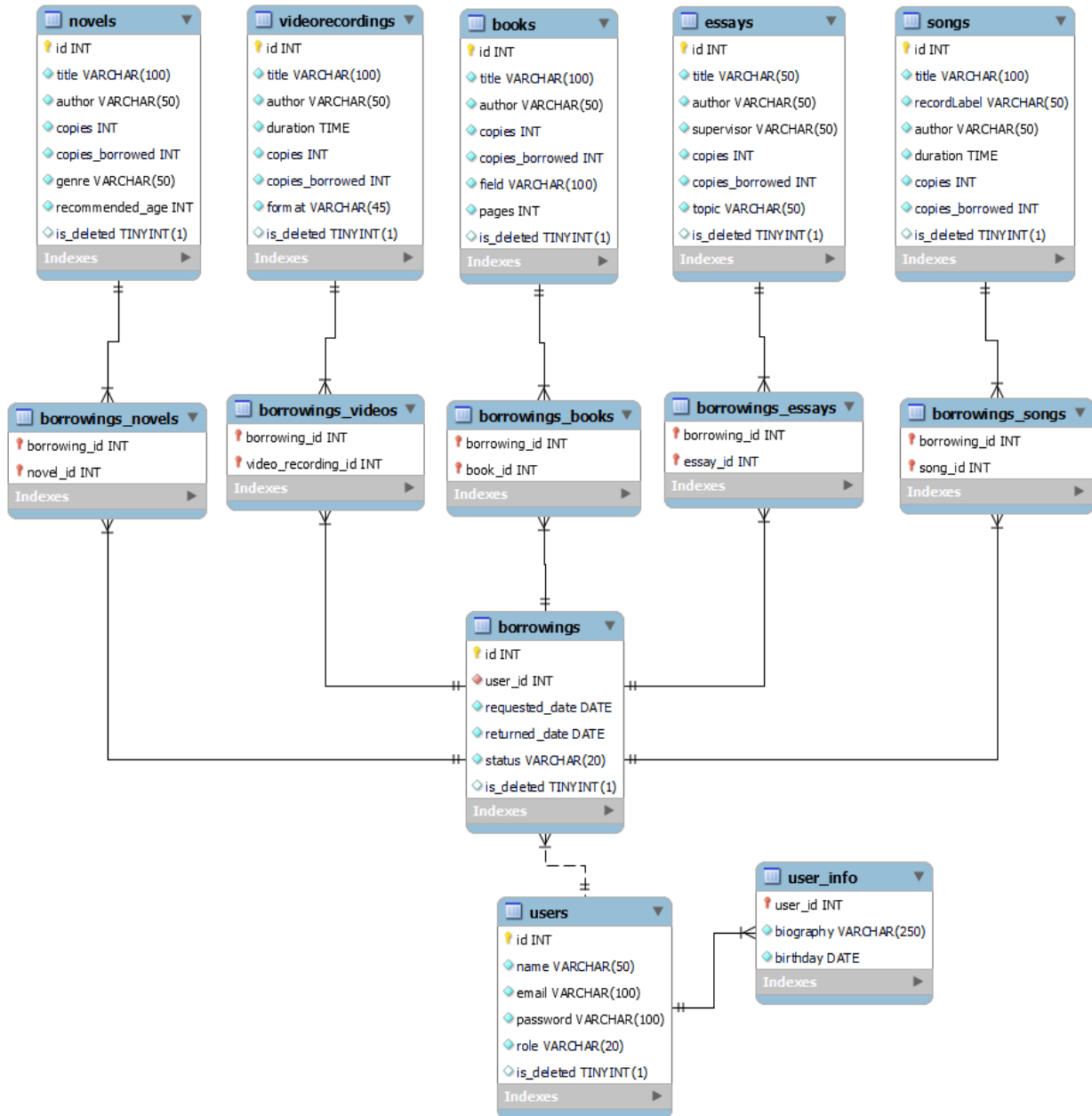
Expansion modelo:

(modelo original)

Se añadieron las entidades de video, ensayo y cancion con sus respectivas tablas intermedias, tambien se agrego la tabla de información extra para el usuario:



Modelo con extensión de las nuevas entidades:



Json and XML serialization y deserializacion

```
public class JsonUtil {  
    2 usages  
    private static final Gson gson = new Gson();  
  
    5 usages  👤 johan  
    public static <T> void writeJson(List<T> objects, String fileName) {  
        String filePath = "./src/main/resources/JsonOutFiles/";  
        try (Writer writer = new FileWriter(fileName: filePath + fileName)) {  
            gson.toJson(objects, writer);  
        } catch (IOException e) {  
            System.out.println(e.getMessage());  
        }  
    }  
  
    1 usage  👤 johan  
    public static List<Book> readBookJsonFile(String fileName) {  
        String filePath = "./src/main/resources/JsonInputFiles/";  
        try (Reader reader = new FileReader(fileName: filePath + fileName)) {  
            👤 johan  
            Type type = new TypeToken<List<Book>>(){}.getType();  
            return gson.fromJson(reader, type);  
        } catch (IOException e) {  
            System.out.println(e.getMessage());  
            return null;  
        }  
    }  
}
```

```

7 usages  👤 johan
public class XmlUtil {
    2 usages
    private static final ObjectMapper xmlMapper = new XmlMapper();

    5 usages  👤 johan
    public static <T> void writeXml(List<T> objects, String fileName) {
        String filePath = "./src/main/resources/XMLOutFiles/";
        try (Writer writer = new FileWriter( fileName: filePath + fileName)) {
            xmlMapper.writeValue(writer, objects);
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }
    }

    1 usage  👤 johan
    public static List<Book> readXmlFile(String fileName) {
        String filePath = "./src/main/resources/XMLInputFiles/";
        try (Reader reader = new FileReader( fileName: filePath + fileName)) {
            👤 johan
            TypeReference<List<Book>> typeReference = new TypeReference<>() {};
            return xmlMapper.readValue(reader, typeReference);
        } catch (IOException e) {
            System.out.println(e.getMessage());
            return null;
        }
    }
}

```

Manual de Usuario:

Al empezar el programa tienes la capacidad de loguearte o mirar la información disponible de la librería como novelas, videos, canciones y ensayos

```

> Task :Main.main()
Bienvenido a la biblioteca La Pingüinera!!
Menú principal: 1. Login | 2. Libros | 3. Novelas | 4. Video grabaciones | 5. Canciones | 6. Ensayos | 0. Exit
> |

```

Puedes entrar directamente o loguearte como lector en el sistema:

```
Login menu: 1. Login with email | 2. Register | 0. Back
```

Como lector tendras todas estas funcionalidades, podras interactuar con los objetos que tiene la libreria

```
Successful login as: Johan Cifuentes : johan@example.com
```

```
Menú principal: 2. Libros | 3. Novelas | 4. Video grabaciones | 5. Canciones | 6. Ensayos | 7. Prestamos | 8. Personal info | 12. Log out | 0. Exit  
> |
```

Haremos el ejemplo completo con libros, pero toda la funcionalidad es igual para las demas entidades

Podrás listar los libros disponibles que te ofrece el sistema:

```
Menú Books: 1. List Books | 2. List authors | 3. Search by author | 4. Borrow a Book | 0. Back
```

```
> 1
```

```
Book{id=1, title='Cien años de soledad', author='Gabriel García Márquez', copies=6, copiesBorrowed=1, field='Ficción', pages=432}  
Book{id=3, title='Crónica de una muerte anunciada', author='Gabriel García Márquez', copies=2, copiesBorrowed=0, field='Ficción', pages=128}  
Book{id=4, title='La sombra del viento', author='Carlos Ruiz Zafón', copies=4, copiesBorrowed=0, field='Ficción', pages=576}  
Book{id=5, title='Cien años de soledad', author='Gabriel García Márquez', copies=10, copiesBorrowed=0, field='Ficción', pages=434}  
Book{id=6, title='Rayuela', author='Julio Cortázar', copies=5, copiesBorrowed=1, field='Ficción', pages=700}  
Book{id=7, title='Neuromancer', author='William Gibson', copies=3, copiesBorrowed=0, field='Ciencia ficción', pages=320}  
Book{id=8, title='Los pilares de la Tierra', author='Ken Follett', copies=2, copiesBorrowed=1, field='Novela histórica', pages=528}  
Book{id=9, title='El código Da Vinci', author='Dan Brown', copies=5, copiesBorrowed=2, field='Thriller', pages=416}  
Book{id=10, title='El nombre del viento', author='Patrick Rothfuss', copies=4, copiesBorrowed=0, field='Fantasía', pages=672}  
Book{id=11, title='El fin de la eternidad', author='Isaac Asimov', copies=3, copiesBorrowed=1, field='Ficción científica', pages=384}  
Book{id=12, title='La isla del tesoro', author='Robert Louis Stevenson', copies=2, copiesBorrowed=0, field='Novela de aventuras', pages=336}  
Book{id=13, title='1984', author='George Orwell', copies=5, copiesBorrowed=2, field='Distopía', pages=352}
```

```
Menú Books: 1. List Books | 2. List authors | 3. Search by author | 4. Borrow a Book | 0. Back
```

Y tambien podras todos los autores de libros que tiene el sistema

```
Menú Books: 1. List Books | 2. List authors | 3. Search by author | 4. Borrow a Book | 0. Back
```

```
> 2
```

```
Patrick Rothfuss  
William Gibson  
Carlos Ruiz Zafón  
Gabriel García Márquez  
Robert Louis Stevenson  
Isaac Asimov  
Dan Brown  
Julio Cortázar  
Ken Follett  
George Orwell
```

```
Menú Books: 1. List Books | 2. List authors | 3. Search by author | 4. Borrow a Book | 0. Back
```

```
> |
```

Podras solicitar varios libros:

```

Men♦ Books: 1. List Books | 2. List authors | 3. Search by author | 4. Borrow a Book | 0. Back
> 4
Type the id of the book you want to borrow:
> 1 id libro de interes

Men♦ Books: 1. List Books | 2. List authors | 3. Search by author | 4. Borrow a Book | 0. Back
> 4
Type the id of the book you want to borrow:
> 7
Book added, go to the Borrowings menu to complete the request

```

Seguidamente a eso, tendras salir e ir a la seccion de prestamos para mirar tus libros solicitados:

```

Men♦ principal: 2. Libros | 3. Novelas | 4. Video grabaciones | 5. Canciones | 6. Ensayos | 7. Prestamos |
>

```

Una vez dentro, tendrás la posibilidad de ver los libros que vas a pedir prestados:

```

Men♦ Borrowings: 1. List selected items | 2. List all borrowings | 3. Show borrowing details | 4. Confirm request | 0. Back
> 1
Selected items pending to confirm request: {0}2
Book{id=3, title='Cronica de una muerte anunciada', author='Gabriel Garcia Marquez', copies=2, copiesBorrowed=0, field='Ficcion', pages=128}
Book{id=1, title='Cien años de soledad', author='Gabriel Garcia Marquez', copies=6, copiesBorrowed=1, field='Ficcion', pages=432}
Men♦ Borrowings: 1. List selected items | 2. List all borrowings | 3. Show borrowing details | 4. Confirm request | 0. Back
> 1

```

Con la opción 4 confirmas tu solicitud de préstamo e ingresas la fecha de cuando lo vas a devolver:

```

Men♦ Borrowings: 1. List selected items | 2. List all borrowings | 3. Show borrowing details | 4. Confirm request | 0. Back
> 4 lo confirmas
Requesting selected items
Type the due date before all the items must be returned: (YYYY-MM-DD)
> 2024-10-24 ingresas la fecha de devolucion
Successful request, find an employee to borrow the items from
Men♦ Borrowings: 1. List selected items | 2. List all borrowings | 3. Show borrowing details | 4. Confirm request | 0. Back
> 2 miras tus prestamos
Borrowings list:
Borrowing{ id=6, borrowerId=7, borrower=Johan Cifuentes, requestedDate=2024-04-19, returnDate=2024-10-24, status=REQUESTED}
Men♦ Borrowings: 1. List selected items | 2. List all borrowings | 3. Show borrowing details | 4. Confirm request | 0. Back
> |

```

solicitado


se genera tu prestamo

Ahora un Empleado o administrador tendra que aceptar tu solicitud, lo veremos ahora mismo

Empleado Administrador

(como administrador podrás hacer todo lo de empleado y usuario)

```
Bienvenido a la Biblioteca La Pingüinera!!
Men principal: 1. Login | 2. Libros | 3. Novelas | 4. Video grabaciones | 5. Canciones | 6. Ensayos | 0. Exit
> 1
Login menu: 1. Login with email | 2. Register | 0. Back
> 1
Email:
> juan@example.com
password:
> juanpass
Successful login as: Juan Pérez : juan@example.com
Men principal: 2. Libros | 3. Novelas | 4. Video grabaciones | 5. Canciones | 6. Ensayos | 7. Prestamos | 8. Personal info | 9. Menu admin | 12. Log out | 0. Exit
> |
```



Vamos a probar el ejemplo de administrador, ya que de ahí descenden las funcionalidades de empleado, lo único que cambia es el menú de administrador

Vamos a probar confirmar los préstamos generados en el sistema:
Vamos a la función 7

```
Successful login as: Juan Pérez : juan@example.com
Men principal: 2. Libros | 3. Novelas | 4. Video grabaciones | 5. Canciones | 6. Ensayos | 7. Prestamos | 8. Personal info | 9. Menu admin | 12. Log out | 0. Exit
> 7
Men Borrowings: 1. List selected items | 2. List all borrowings | 3. Show borrowing details | 4. Confirm request |
5. Search by email | 6. Confirm borrowing | 7. Finalize borrowing | 8. Delete borrowing | 0. Back
>
```

Como podemos evidenciar, como empleado y administrador tenemos las funcionalidades de la 5 a la 8, permitiendo hacer mucha mas cosas que como usuario

Vamos a listar las solicitudes de prestamos que han hecho los lectores:

```
Men Borrowings: 1. List selected items | 2. List all borrowings | 3. Show borrowing details | 4. Confirm request |
5. Search by email | 6. Confirm borrowing | 7. Finalize borrowing | 8. Delete borrowing | 0. Back
> 2
Borrowings list:
Borrowing{ id=1, borrowerId=2, borrower=Juan Pérez, requestedDate=2024-04-03, returnDate=2024-04-17, status=BORROWED}
Borrowing{ id=2, borrowerId=3, borrower=María Gómez, requestedDate=2024-04-05, returnDate=2024-04-19, status=REQUESTED}
Borrowing{ id=3, borrowerId=4, borrower=Pedro Ramirez, requestedDate=2024-04-07, returnDate=2024-04-21, status=REQUESTED}
Borrowing{ id=4, borrowerId=5, borrower=Luisa Martínez, requestedDate=2024-04-09, returnDate=2024-04-23, status=FINALIZED}
Borrowing{ id=5, borrowerId=6, borrower=Andrés López, requestedDate=2024-04-01, returnDate=2024-04-11, status=BORROWED}
```

Como empleado o administrador vamos a aceptar el préstamo 2

Entramos a la función 6 del sistema que me permite confirmar préstamos, seguido a eso ingresamos el id del préstamo y ya quedaría confirmado, para verificar utilizamos la opción de listar la información y se evidencia que cambio de solicitado a prestado

```
Men Borrowings: 1. List selected items | 2. List all borrowings | 3. Show borrowing details | 4. Confirm request |
5. Search by email | 6. Confirm borrowing | 7. Finalize borrowing | 8. Delete borrowing | 0. Back
> 6
Type the id of the borrowing you want to confirm as delivered
> 2
Borrowing confirmed successfully
```


También puedo finalizar un préstamo, ya una vez un cliente me ha devuelto los libros, usaremos la opción 7, y luego ingresamos el id del préstamo 3 que se encuentra en estado prestado, seguido a esto veremos la lista y confirmaremos que ha sido finalizado

```
Menu Borrowings: 1. List selected items | 2. List all borrowings | 3. Show borrowing details | 4. Confirm request |
5. Search by email | 6. Confirm borrowing | 7. Finalize borrowing | 8. Delete borrowing | 0. Back
> 7
Type the id of the borrowing you want to finalize
> 2
Borrowing finalized successfully
Menu Borrowings: 1. List selected items | 2. List all borrowings | 3. Show borrowing details | 4. Confirm request |
5. Search by email | 6. Confirm borrowing | 7. Finalize borrowing | 8. Delete borrowing | 0. Back
> 1
Selected items pending to confirm request: {0}0
Menu Borrowings: 1. List selected items | 2. List all borrowings | 3. Show borrowing details | 4. Confirm request |
5. Search by email | 6. Confirm borrowing | 7. Finalize borrowing | 8. Delete borrowing | 0. Back
> 2
Borrowings list:
Borrowing{ id=1, borrowerId=2, borrower=Juan Perez, requestedDate=2024-04-03, returnDate=2024-04-17, status=BORROWED}
Borrowing{ id=2, borrowerId=3, borrower=María Gomez, requestedDate=2024-04-05, returnDate=2024-04-19, status=FINALIZED}
Borrowing{ id=3, borrowerId=4, borrower=Pedro Ramirez, requestedDate=2024-04-07, returnDate=2024-04-21, status=REQUESTED}
Borrowing{ id=4, borrowerId=5, borrower=Luisa Martinez, requestedDate=2024-04-09, returnDate=2024-04-23, status=FINALIZED}
Borrowing{ id=5, borrowerId=6, borrower=Andrés Lopez, requestedDate=2024-04-01, returnDate=2024-04-11, status=BORROWED}
Menu Borrowings: 1. List selected items | 2. List all borrowings | 3. Show borrowing details | 4. Confirm request |
5. Search by email | 6. Confirm borrowing | 7. Finalize borrowing | 8. Delete borrowing | 0. Back
> |
```

Ahora, como cualquier usuario editaremos nuestra información personal y agregaremos información extra si queremos, para esto salimos nuevamente al menú principal:

```
Menu principal: 2. Libros | 3. Novelas | 4. Video grabaciones | 5. Canciones | 6. Ensayos | 7. Prestamos | 8. Personal info | 9. Menu admin | 12. Log out | 0. Exit
>
```

Podremos ver nuestra información personal:

```
Men♦ PersonalInfo: 1. See personal info | 2. Edit personal info | 3. Add personal info | 0. Back
> 1
User{id=2, name='Juan Perez', email='juan@example.com', role=ADMINISTRATOR, biography='null', birthday=null}
Men♦ PersonalInfo: 1. See personal info | 2. Edit personal info | 3. Add personal info | 0. Back
>
```

Ahora, agregaremos información personal con la función 3 :

Podremos evidenciar que luego de agregar la información extra y la volvemos a consultar, nuestra información se habrá actualizado

```
Men♦ PersonalInfo: 1. See personal info | 2. Edit personal info | 3. Add personal info | 0. Back
> 3
Enter biography: my guapo
Enter birthday (yyyy-mm-dd): 2023-10-24
Men♦ PersonalInfo: 1. See personal info | 2. Edit personal info | 3. Add personal info | 0. Back
> 1
User{id=2, name='Juan Perez', email='juan@example.com', role=ADMINISTRATOR, biography='my guapo', birthday=2023-10-24}
Men♦ PersonalInfo: 1. See personal info | 2. Edit personal info | 3. Add personal info | 0. Back
> |
```

Ahora, vamos a editar nuestra información personal donde seremos de cambiar toda nuestra información incluyendo contraseña y correo, seguido a eso la volveremos a consultar y veremos la actualización:

```
Men♦ PersonalInfo: 1. See personal info | 2. Edit personal info | 3. Add personal info | 0. Back
> 2
Enter name: david
Enter email: david@example.com
Enter password: davidpass
Enter biography: muy sexy
Enter birthday (yyyy-mm-dd): 2003-02-02
Men♦ PersonalInfo: 1. See personal info | 2. Edit personal info | 3. Add personal info | 0. Back
> 1
User{id=2, name='david', email='david@example.com', role=ADMINISTRATOR, biography='muy sexy', birthday=2003-02-02}
Men♦ PersonalInfo: 1. See personal info | 2. Edit personal info | 3. Add personal info | 0. Back
> |
```

← nueva data

Ahora, pasaremos a funciones más avanzadas de administrador:

Estas son las funcionalidades que tengo como administrador del sistema:

```
Men♦ principal: 2. Libros | 3. Novelas | 4. Video grabaciones | 5. Canciones | 6. Ensayos | 7. Prestamos | 8. Personal info | 9. Menu admin | 12. Log out | 0. Exit
> 9
Men♦ Admin: 1. List users | 2. Create employee user | 3. Update user | 4. Delete user | 5. Export Csv inventory data | 6. Import Csv inventory data |
7. Export Json inventory data | 8. Import Json inventory data | 9. Export XML inventory data | 10. Import XML inventory data | 0. Back
>
```

Podemos ver la lista de usuarios actual:

```
Men Admin: 1. List users | 2. Create employee user | 3. Update user | 4. Delete user | 5. Export Csv inventory data | 6. Import Csv inventory data | 7. Export Json inventory data | 8. Import Json inventory data | 9. Export XML inventory data | 10. Import XML inventory data | 0. Back
> 1
List of users:
User{id=1, name='John Doe', email='administrador@pingu.com.co', role=ADMINISTRATOR, biography='null', birthday=null}
User{id=2, name='david', email='david@example.com', role=ADMINISTRATOR, biography='null', birthday=null}
User{id=3, name='María Gómez', email='maria@example.com', role=READER, biography='null', birthday=null}
User{id=4, name='Pedro Ramirez', email='pedro@example.com', role=READER, biography='null', birthday=null}
User{id=5, name='Luisa Martínez', email='luisa@example.com', role=EMPLOYEE, biography='null', birthday=null}
User{id=6, name='Andrés López', email='andres@example.com', role=READER, biography='null', birthday=null}
User{id=7, name='Johan Cifuentes', email='johan@example.com', role=READER, biography='null', birthday=null}
User{id=8, name='super', email='super@example.com', role=SUPER, biography='null', birthday=null}
Men Admin: 1. List users | 2. Create employee user | 3. Update user | 4. Delete user | 5. Export Csv inventory data | 6. Import Csv inventory data | 7. Export Json inventory data | 8. Import Json inventory data | 9. Export XML inventory data | 10. Import XML inventory data | 0. Back
>
```

Podremos crear un usuario como empleado:

```
Men Admin: 1. List users | 2. Create employee user | 3. Update user | 4. Delete user | 5. Export Csv inventory data | 6. Import Csv inventory data | 7. Export Json inventory data | 8. Import Json inventory data | 9. Export XML inventory data | 10. Import XML inventory data | 0. Back
> 2
Enter the employee user data:
Name:
> example
Email:
> example
password:
> example
```

Y seguidamente listarlo:

```
Men Admin: 1. List users | 2. Create employee user | 3. Update user | 4. Delete user | 5. Export Csv inventory data | 6. Import Csv inventory data | 7. Export Json inventory data | 8. Import Json inventory data | 9. Export XML inventory data | 10. Import XML inventory data | 0. Back
> 1
List of users:
User{id=1, name='John Doe', email='administrador@pingu.com.co', role=ADMINISTRATOR, biography='null', birthday=null}
User{id=2, name='david', email='david@example.com', role=ADMINISTRATOR, biography='null', birthday=null}
User{id=3, name='María Gómez', email='maria@example.com', role=READER, biography='null', birthday=null}
User{id=4, name='Pedro Ramirez', email='pedro@example.com', role=READER, biography='null', birthday=null}
User{id=5, name='Luisa Martínez', email='luisa@example.com', role=EMPLOYEE, biography='null', birthday=null}
User{id=6, name='Andrés López', email='andres@example.com', role=READER, biography='null', birthday=null}
User{id=7, name='Johan Cifuentes', email='johan@example.com', role=READER, biography='null', birthday=null}
User{id=8, name='super', email='super@example.com', role=SUPER, biography='null', birthday=null}
User{id=9, name='example', email='example', role=EMPLOYEE, biography='null', birthday=null}
```

Podremos editarlo y asignarle un rol .

```

Men♦ Admin: 1. List users | 2. Create employee user | 3. Update user | 4. D
7. Export Json inventory data | 8. Import Json inventory data | 9. Export
> 3
Type the id of the user you want to update:
> 9
Name:
> example2
Email:
> example2
Choose a role: 1. READER | 2. EMPLOYEE | 3. ADMIN
> READER
Dato invalido. Por favor ingrese un número entero.
Choose a role: 1. READER | 2. EMPLOYEE | 3. ADMIN
> 1

```

Y listarlo nuevamente:

```

List of users:
User{id=1, name='John Doe', email='administrador@pingu.com.co', role=ADMINISTRATOR, biography='null', birthday=null}
User{id=2, name='david', email='david@example.com', role=ADMINISTRATOR, biography='null', birthday=null}
User{id=3, name='Mar♦a G♦mez', email='maria@example.com', role=READER, biography='null', birthday=null}
User{id=4, name='Pedro Ramirez', email='pedro@example.com', role=READER, biography='null', birthday=null}
User{id=5, name='Luisa Mart♦nez', email='luisa@example.com', role=EMPLOYEE, biography='null', birthday=null}
User{id=6, name='Andr♦s L♦pez', email='andres@example.com', role=READER, biography='null', birthday=null}
User{id=7, name='Johan Cifuentes', email='johan@example.com', role=READER, biography='null', birthday=null}
User{id=8, name='super', email='super@example.com', role=SUPER, biography='null', birthday=null}
User{id=9, name='example2', email='example2', role=READER, biography='null', birthday=null}

```

Podremos eliminarlos :

Y ya no aparecerá en la lista de empleados:

```

Men Admin: 1. List users | 2. Create employee user | 3. Update user | 4. Delete user | 5. Export Csv inventory data | 6. Import Csv inventory data | 7. Export Json inventory data | 8. Import Json inventory data | 9. Export XML inventory data | 10. Import XML inventory data | 0. Back
> 4
Type the id of the user you want to delete:
> 9
User deleted successfully
Men Admin: 1. List users | 2. Create employee user | 3. Update user | 4. Delete user | 5. Export Csv inventory data | 6. Import Csv inventory data | 7. Export Json inventory data | 8. Import Json inventory data | 9. Export XML inventory data | 10. Import XML inventory data | 0. Back
> 1
List of users:
User{id=1, name='John Doe', email='administrador@pingu.com.co', role=ADMINISTRATOR, biography='null', birthday=null}
User{id=2, name='david', email='david@example.com', role=ADMINISTRATOR, biography='null', birthday=null}
User{id=3, name='María Gómez', email='maria@example.com', role=READER, biography='null', birthday=null}
User{id=4, name='Pedro Ramirez', email='pedro@example.com', role=READER, biography='null', birthday=null}
User{id=5, name='Luisa Martínez', email='luisa@example.com', role=EMPLOYEE, biography='null', birthday=null}
User{id=6, name='Andrés López', email='andres@example.com', role=READER, biography='null', birthday=null}
User{id=7, name='Johan Cifuentes', email='johan@example.com', role=READER, biography='null', birthday=null}
User{id=8, name='super', email='super@example.com', role=SUPER, biography='null', birthday=null}

```

Ahora, exportaremos la información en distintos formatos (csv, json, xml):

```

7. Export Json inventory data | 8. Import Json inventory data | 9. Export XML inventory data | 10. Import XML inventory data | 0. Back
> 5
Exporting library items data to file
Enter the name you want for the export file:
> data
Successful export to: data.csv.csv
Men Admin: 1. List users | 2. Create employee user | 3. Update user | 4. Delete user | 5. Export Csv inventory data | 6. Import Csv inventory data | 7. Export Json inventory data | 8. Import Json inventory data | 9. Export XML inventory data | 10. Import XML inventory data | 0. Back
> 7
Json data successfully exported
Men Admin: 1. List users | 2. Create employee user | 3. Update user | 4. Delete user | 5. Export Csv inventory data | 6. Import Csv inventory data | 7. Export Json inventory data | 8. Import Json inventory data | 9. Export XML inventory data | 10. Import XML inventory data | 0. Back
> 9
XML data successfully exported

```

```

| 5. Export Csv inventory data |
ory data | 10. Import XML invento

```

```

7. Export Json inventory data

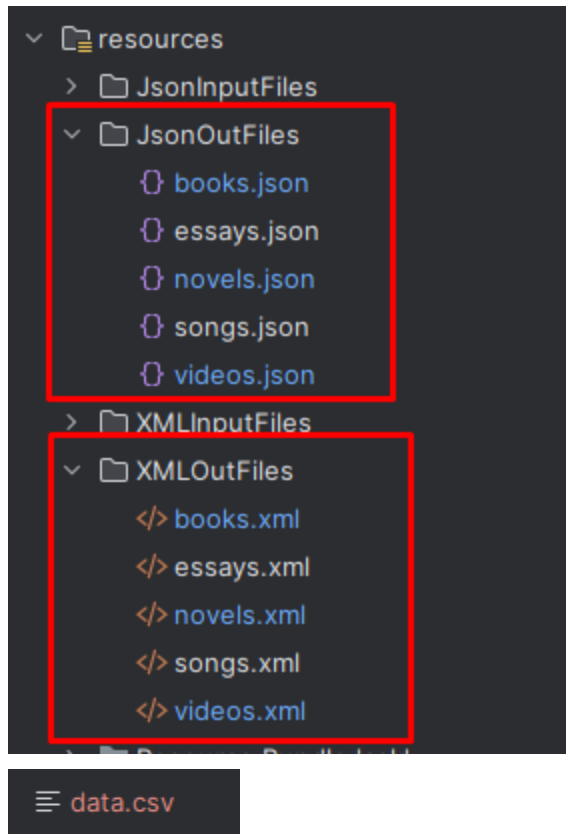
```

```

| 9. Export XML inventory data

```

Teniendo como resultado los siguientes archivos:

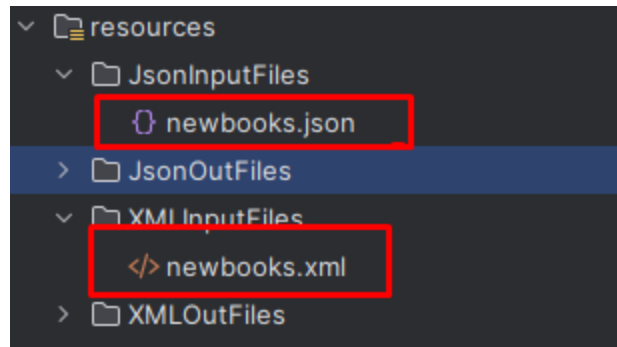


```
newbooks.json  </> newbooks.xml  books.json x  essays.json  novels.json  </> essays.xml

1 [{"field": "Ficcion", "pages": 432, "id": 1, "title": "Cien años de soledad", "author": "Gabriel Garc\u00eda M\u00f3rquez", "copies": 6, "copiesBorrowed": 1},
2   {"field": "Ficcion", "pages": 368, "id": 2, "title": "El amor en los tiempos del c\u00f3lera", "author": "Gabriel Garc\u00eda M\u00f3rquez", "copies": 1, "copiesBorrowed": 0},
3   {"field": "Ficcion", "pages": 128, "id": 3, "title": "Cronica de una muerte anunciada", "author": "Gabriel Garc\u00eda M\u00f3rquez", "copies": 2, "copiesBorrowed": 0},
4   {"field": "Ficcion", "pages": 576, "id": 4, "title": "La sombra del viento", "author": "Carlos Ruiz Zaf\u00f3n", "copies": 4, "copiesBorrowed": 0},
5   {"field": "Ficcion", "pages": 434, "id": 5, "title": "Cien a\u00f1os de soledad", "author": "Gabriel Garc\u00eda M\u00f3rquez", "copies": 10, "copiesBorrowed": 0},
6   {"field": "Ficcion", "pages": 700, "id": 6, "title": "Rayuela", "author": "Julio Cort\u00e1zar", "copies": 5, "copiesBorrowed": 1}]
```

Cada uno funcionando correctamente

Ahora las funcionalidades de importar me traeran los recursos que traen estos archivos:



Cada uno con diferente formato pero en listas.

```
newbooks.json x </> newbooks.xml books.json essays.json novels.json </> essays.xml
1  [
2    {
3      "field": "Ciencia ficción",
4      "pages": 320,
5      "id": 7,
6      "title": "Neuromancer",
7      "author": "William Gibson",
8      "copies": 3,
9      "copiesBorrowed": 0
10   },
11   {
12     "field": "Novela histórica",
13     "pages": 528,
14     "id": 8,
15     "title": "Los pilares de la Tierra",
16     "author": "Ken Follett",
17     "copies": 2,
18     "copiesBorrowed": 1
19   },
20   {
21     "field": "Thriller",
22     "pages": 416,
23     "id": 9,
24     "title": "El código Da Vinci",
25     "author": "Dan Brown",
26     "copies": 5,
27     "copiesBorrowed": 2
28   },
29   {
30     "field": "Fantasía",
31     "pages": 672,
32     "id": 10,
33     "title": "El nombre del viento",
34     "author": "Patrick Rothfuss",
35     "copies": 4,
36     "copiesBorrowed": 0
37   }
38 ]
```

Y:


```
newbooks.json  </> newbooks.xml ×  books.json  essays.json

1  <libros>
2    <libro>
3      <field>Ficción científica</field>
4      <pages>384</pages>
5      <id>11</id>
6      <title>El fin de la eternidad</title>
7      <author>Isaac Asimov</author>
8      <copies>3</copies>
9      <copiesBorrowed>1</copiesBorrowed>
10    </libro>
11    <libro>
12      <field>Novela de aventuras</field>
13      <pages>336</pages>
14      <id>12</id>
15      <title>La isla del tesoro</title>
16      <author>Robert Louis Stevenson</author>
17      <copies>2</copies>
18      <copiesBorrowed>0</copiesBorrowed>
19    </libro>
20    <libro>
21      <field>Distopía</field>
22      <pages>352</pages>
23      <id>13</id>
24      <title>1984</title>
25      <author>George Orwell</author>
26      <copies>5</copies>
27      <copiesBorrowed>2</copiesBorrowed>
28    </libro>
29  </libros>
30
```

Y con las siguientes opciones llenamos la tablas de books utilizando deserializadores:

```
Xml data successfully exported
Men Admin: 1. List users | 2. Create employee user | 3. Update user | 4. Delete user | 5. Export Csv inventory data | 6. Import Csv inventory data |
7. Export Json inventory data | 8. Import Json inventory data | 9. Export XML inventory data | 10. Import XML inventory data | 0. Back
>

Men Admin: 1. List users | 2. Create employee user | 3. Update user | 4. Delete user | 5. Export Csv inventory data | 6. Import Csv inventory data |
7. Export Json inventory data | 8. Import Json inventory data | 9. Export XML inventory data | 10. Import XML inventory data | 0. Back
> 8

Men Admin: 1. List users | 2. Create employee user | 3. Update user | 4. Delete user | 5. Export Csv inventory data | 6. Import Csv inventory data |
7. Export Json inventory data | 8. Import Json inventory data | 9. Export XML inventory data | 10. Import XML inventory data | 0. Back
> 10

Books imported

Men Admin: 1. List users | 2. Create employee user | 3. Update user | 4. Delete user | 5. Export Csv inventory data | 6. Import Csv inventory data |
7. Export Json inventory data | 8. Import Json inventory data | 9. Export XML inventory data | 10. Import XML inventory data | 0. Back
> 1
```

Y ahora la información quedara cargada en la base de datos:

	id	title	author	copies	copies_borrowed	field	pages	is_deleted
1	1	Cien años de soledad	Gabriel García Márquez	6	1	Ficción	432	0
2	2	El amor en los tiempos del cólera	Gabriel García Márquez	1	1	Ficción	368	0
3	3	Crónica de una muerte anunciada	Gabriel García Márquez	2	-1 original	Ficción	128	0
4	4	La sombra del viento	Carlos Ruiz Zafón	4	0	Ficción	576	0
5	5	Cien años de soledad	Gabriel García Márquez	10	0	Ficción	434	0
6	6	Rayuela	Julio Cortázar	5	1	Ficción	700	0
7	7	Neuromancer	William Gibson	3	0	Ciencia ficción	320	0
8	8	Los pilares de la Tierra	Ken Follett	2	1	Novela histórica	528	0
9	9	El código Da Vinci	Dan Brown	5	2 json	Thriller	416	0
10	10	El nombre del viento	Patrick Rothfuss	4	0	Fantasa	672	0
11	11	El fin de la eternidad	Isaac Asimov	3	1	Ficción científica...	384	0
12	12	La isla del tesoro	Robert Louis Stevenson	2	0	Novela de aventuras	336	0
13	13	1984	George Orwell	5	2 xml	Distopía	352	0
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Insertando correctamente la información, se hace el ejemplo con libros ya que falta aumentar las funcionalidades

Valor agregado:

Definición constantes:

```
2 usages new *
public class MongoClientConnectionConstant {
    2 usages
    protected static final String CONNECTION_STRING = "mongodb://localhost:27017";
    2 usages
    protected static final String DATABASE_NAME = "pingu";
}
1
```

Establecer conexión:

```
5 usages new *
10 public class MongoDBConnector {
    5 usages
    11 private static MongoClient mongoClient;
    2 usages
    12 private static MongoDBDatabase mongoDatabase;
    13
    1 usage new *
    14 public MongoDBConnector(){
    15     try{
    16         mongoClient = MongoClient.create(CONNECTION_STRING);
    17         mongoDatabase = mongoClient.getDatabase(DATABASE_NAME);
    18         System.out.println("Successfully connected");
    19     }catch (Exception e){
    20         System.out.println(e.getMessage());
    21     }
    22 }
    23
    no usages new *
    24 > public MongoClient getMongoClient() { return mongoClient; }
    27
    1 usage new *
    28 > public MongoDBDatabase getMongoDatabase() { return mongoDatabase; }
    31
    no usages new *
    32 public void closeConnection() {
    33     if (mongoClient != null) {
    34         mongoClient.close();
    35         System.out.println("Closed connection");
    36     }
    37 }
    38 }
```

Datos de ejemplo:

The screenshot shows the MongoDB Compass interface. At the top, the 'users' collection is selected. Below the navigation bar, there are tabs for Documents (3), Aggregations, Schema, Indexes (1), and Validation. A query bar is present with the text 'Type a query: { field: 'value' } or [Generate query](#)'. Below the query bar, there are buttons for ADD DATA, EXPORT DATA, UPDATE, and DELETE. The main area displays three documents from the 'users' collection:

```
{ "_id": ObjectId('6623e696e9e306f122117b7b'), "id": 1000, "name": "mongo user", "email": "mongo@example.com", "role": "ADMINISTRATOR", "biography": "muy guapo", "birthday": "2003-10-24T00:00:00.000+00:00" }
{ "_id": ObjectId('6623e696e9e306f122117b7c'), "id": 1001, "name": "otro usuario", "email": "otro@example.com", "role": "READER", "biography": "interesante", "birthday": "1995-05-15T00:00:00.000+00:00" }
{ "_id": ObjectId('6623e696e9e306f122117b7d'), "id": 1002, "name": "tercer usuario", "email": "tercero@example.com", "role": "READER", "biography": "informativo", "birthday": "1988-12-30T00:00:00.000+00:00" }
```

The text 'Example data' is written in red to the right of the second document.

Resultado de un tipo 'find()':

```
Successfully connected
example data
{"_id": {"$oid": "6623e696e9e306f122117b7b"}, "id": 1000, "name": "mongo user", "email": "mongo@example.com", "role": "ADMINISTRATOR", "biography": "muy guapo", "birthday": "2003-10-24T00:00:00.000+00:00"},
{"_id": {"$oid": "6623e696e9e306f122117b7c"}, "id": 1001, "name": "otro usuario", "email": "otro@example.com", "role": "READER", "biography": "interesante", "birthday": "1995-05-15T00:00:00.000+00:00"},
{"_id": {"$oid": "6623e696e9e306f122117b7d"}, "id": 1002, "name": "tercer usuario", "email": "tercero@example.com", "role": "READER", "biography": "informativo", "birthday": "1988-12-30T00:00:00.000+00:00"}
```

CONCLUSIONES DEL TALLER:

- Un taller bastante retador, principalmente si tienes un proyecto muy cerrado y difícil de expandir
- Me da de enseñanza a mejorar la legibilidad de mi código y a crear componentes más independientes, refactorizar aun mas el código y mantenerlo simple pero funcional

OPORTUNIDADES DE MEJORA

- Practicar y practicar más a profundidad
- En cuanto al proyecto, reestructurar el menú principal, ya que al hacerlo dinámico está muy cerrado a cambios y tiene una complejidad muy alta
- Expandir las funcionalidades del superusuario, ya que solo puede hacer tareas fantasma usando libros y reservas

Lista de tareas completadas

1. Debe haber un superusuario que puede verificar todas las funcionalidades del sistema y debe poder crear los administradores, incluso debe tener la opción de restaurar una vez realice las pruebas de préstamo.
2. Todas las personas incluyendo los usuarios del sistema pueden cambiar su contraseña y modificar o agregar 2 campos (cada una de las tablas que contiene información de personas) adicionales en la base de datos, recuerde no llenar tablas con datos nulos.
3. La librería quiere agregar Videograbaciones, canciones y también ensayos tipo tesis. Es importante que aumente todas las capacidades de su sistema.
4. La librería desea integrarse con Archivos XML y JSON razón por la cual debe generar y recibir archivos de estos dos formatos para el manejo del inventario, adicional a los archivos de tipo Excel(CSV o XLS).
5. Cada vez que el super usuario registre un nuevo usuario se debe almacenar esta acción en una tabla llamada registros creados y debe registrarse de forma automática.

1: Si, logra crear, editar, eliminar préstamos y libros sin afectar la base de datos,
Y al momento de crear administradores si quedan guardados

2: Si, se logró hacer.

3: Si, se lograron incluir las nuevas entidades y su funcionamiento igual que novelas y libros

4: Si, se pueden importar y exportar archivos XML Y JSON, se puede exportar toda la info de la librería pero al momento de importar solo se integró con libros

5. Si, se creo una tabla de auditoría que registra cada vez que se inserta un usuario en la tabla de usuarios