

Solución Taller Java 2 La Pingüinera 2.0

Realizado por: Jesús David Bonelo Cuellar

Con base en el trabajo del compañero: Johan Steven Cifuentes:

<https://github.com/Training2024SU/2024-C1-QA-JAVA-T01/pull/5>

Nuevos elementos para prestar

- **La librería quiere agregar Videograbaciones, canciones y también ensayos tipo tesis. Es importante que aumente todas las capacidades de su sistema.**

Identifiqué que el diseño de la base de datos y el flujo de la aplicación era poco flexible para introducir nuevos tipos de ítems de la biblioteca de forma fácil, ya que cada tipo se encontraba separado en la base de datos y tiene su propia tabla de préstamos, del mismo modo la lógica programada tampoco fue generalizada, es decir, las opciones comunes de los ítems fue reimplementada por cada tipo.

Siguiendo el enunciado original, el cual según mi interpretación indica que se debe permitir realizar préstamos de múltiples ítems y de diferente tipo dentro de un único préstamo, las nuevas funcionalidades se implementaron bajo esta premisa.

La nueva actualización no modifica la implementación existente, sino que añade las funcionalidades nuevas de forma extra a la aplicación.

Elaboré los nuevos menús utilizando la misma estrategia de mensajes, opciones y permisos de acceso, y además extendí los ya existentes.

Para la lógica de base de datos y acciones realizables, para las nuevas funciones la hice de forma genérica cuando fue posible, facilitando la futura extensibilidad del sistema o una refactorización de la lógica de las funcionalidades anteriores para integrarlas con las nuevas.

Actualización modelo de base de datos

- Se altera la tabla de **user** con 2 nuevas columnas.

- Se crea una nueva tabla llamada **Resources** para almacenar los datos comunes de los nuevos tipos de recursos disponibles para préstamo.
- Se crean nuevas tablas para almacenar la información particular de los nuevos recursos: **Songs**, **Video_recordings**, **Essays**. Estas se relacionan con **Resources** por medio del id.
- Se crea una nueva tabla para los préstamos de los nuevos recursos **Loans** conteniendo toda la información de un préstamo.
- Los nuevos recursos pueden ser prestados varios al mismo tiempo dentro de un solo id de préstamo, para ello se crea la tabla intermedia utilizada en la relación muchos a muchos.

Scripts sql adjuntos en el repositorio:

- **updatePenguinDatabaseV2.sql** para actualizar el diseño de la base de datos existente
- **InsertDataExampleV2.sql** para actualizar la información en las tablas anteriores y llenar las nuevas

```
USE `Penguin_Library`;
```

```
ALTER TABLE user ADD birth_date DATE;
ALTER TABLE user ADD phone VARCHAR(50);
```

```
DROP TABLE IF EXISTS Resources;
CREATE TABLE IF NOT EXISTS Resources (
    resource_id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    type VARCHAR(30) NOT NULL,
    title VARCHAR(100) NOT NULL,
    quantity INT NOT NULL,
    quantity_loaned INT NOT NULL,
    author_id VARCHAR(100) NOT NULL,
    is_deleted BOOLEAN DEFAULT 0,
    FOREIGN KEY (author_id)
        REFERENCES author (id)
) ENGINE=INNODB;
```

```
DROP TABLE IF EXISTS Songs;
CREATE TABLE IF NOT EXISTS Songs (
```

```
    resource_id INT NOT NULL,  
    duration FLOAT NOT NULL,  
    FOREIGN KEY (resource_id)  
    REFERENCES Resources (resource_id)  
) ENGINE=INNODB;
```

```
DROP TABLE IF EXISTS Video_recordings;  
CREATE TABLE IF NOT EXISTS Video_recordings (  
    resource_id INT NOT NULL,  
    resolution varchar(100) NOT NULL,  
    FOREIGN KEY (resource_id)  
    REFERENCES Resources (resource_id)  
) ENGINE=INNODB;
```

```
DROP TABLE IF EXISTS Essays;  
CREATE TABLE IF NOT EXISTS Essays (  
    resource_id INT NOT NULL,  
    academic_level varchar(50) NOT NULL,  
    FOREIGN KEY (resource_id)  
    REFERENCES Resources (resource_id)  
) ENGINE=INNODB;
```

```
DROP TABLE IF EXISTS Loans;  
CREATE TABLE IF NOT EXISTS Loans (  
    loan_id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,  
    request_date DATE NOT NULL,  
    return_date DATE NOT NULL,  
    status VARCHAR(45) NOT NULL DEFAULT 'REQUESTED',  
    user_id VARCHAR(100) NOT NULL,  
    is_deleted BOOLEAN DEFAULT 0,  
    FOREIGN KEY (user_id)  
    REFERENCES user (id)  
) ENGINE=INNODB;
```

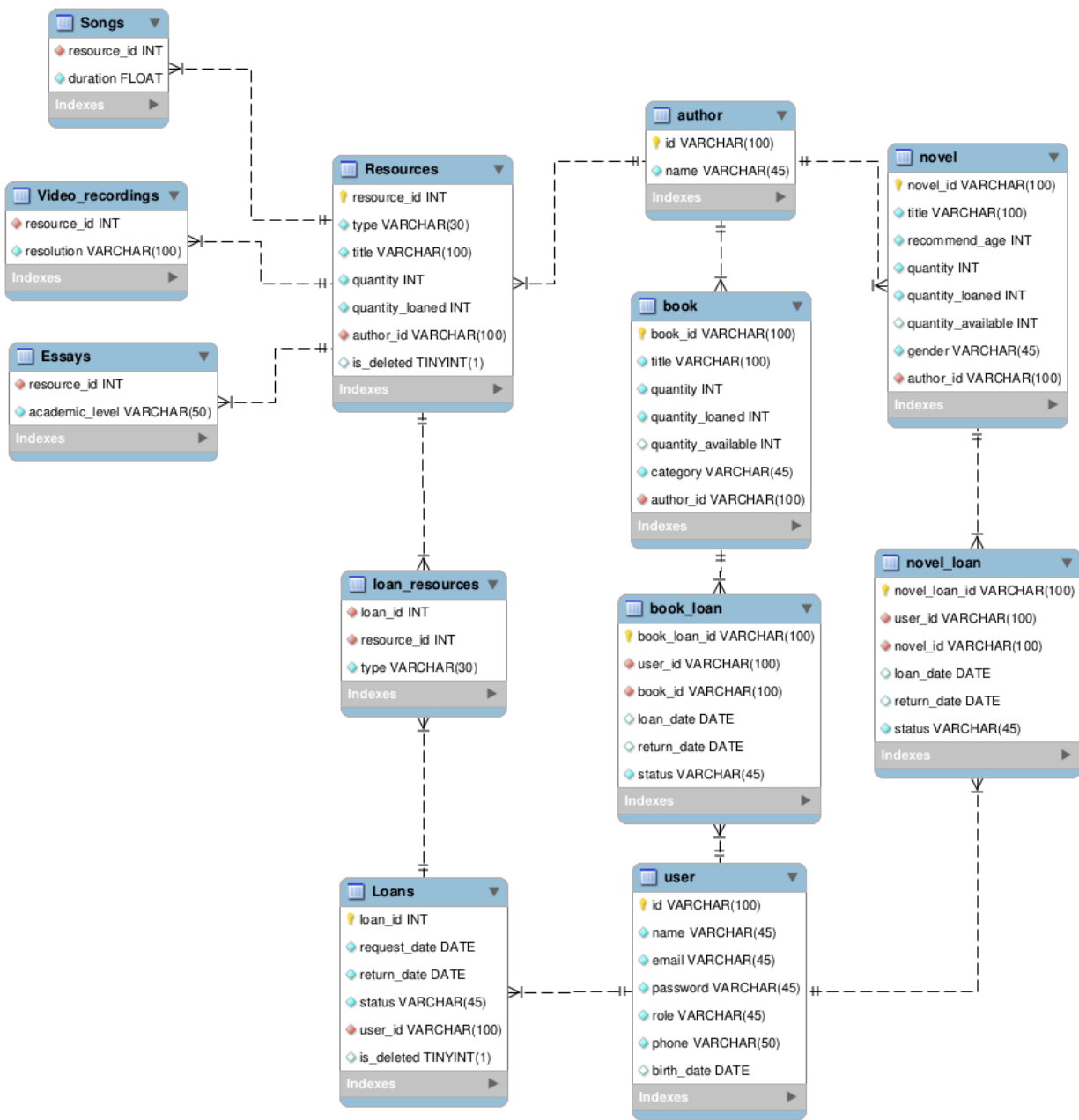
```
DROP TABLE IF EXISTS loan_resources;  
CREATE TABLE IF NOT EXISTS loan_resources (  
    loan_id INT NOT NULL,  
    resource_id INT NOT NULL,  
    type VARCHAR(30) NOT NULL,
```

```

FOREIGN KEY (loan_id)
REFERENCES Loans (loan_id),
FOREIGN KEY (resource_id)
REFERENCES Resources (resource_id)
) ENGINE=INNODB;

```

Nuevo diagrama EER resultante



Implementación en Java

- Creo los modelos, DAOs, funciones y menús de los nuevos tipos de elementos para prestar.

Las agregué bajo una nueva categoría de “otros recursos” dando a entender que los ítems principales son los libros y novelas

Las opciones para lectores:

Al realizar las nuevas funcionalidades de los recursos para préstamos, lo hice de la forma más genérica posible ya que todos son utilizados y se comportan de la misma manera, la única excepción es el imprimir sus detalles lo cual para permitir la interoperabilidad, sobrescriben la definición heredada.

Ejemplo en el DAO:

```
public Resource buildResourceFromResultSet(ResultSet resultSet)
throws SQLException {
    ResourceType type =
ResourceType.valueOf(resultSet.getString("type"));
    int id = resultSet.getInt("resource_id");
    String title = resultSet.getString("title");
    int quantity = resultSet.getInt("quantity");
    int quantityLoaned = resultSet.getInt("quantity_loaned");
    String authorId = resultSet.getString("author_id");
    String authorName = resultSet.getString("name");
    Author author = new Author(authorId, authorName);
    return switch (type) {
        case SONG -> getSong(type, id, title, quantity,
quantityLoaned, author);
        case VIDEO_RECORDING -> getVideo(type, id, title,
quantity, quantityLoaned, author);
        case ESSAY -> getEssay(type, id, title, quantity,
quantityLoaned, author);
    };
}
```

Ejemplo en las clases de Management:

```

public void exportResourcesToFile(String fileName, String extension)
throws IOException {
    // get all resources with details, and group them by their type
    Map<ResourceType, List<Resource>> resourcesByType =
        resourceDAO.getAllResources().stream().map(r ->
getResourceDetails(r.getId())).collect(Collectors.groupingBy(Resource
::getType));

    // save a file for every type of resource
    for (Map.Entry<ResourceType, List<Resource>> entry :
resourcesByType.entrySet()) {
        ResourceType resourceType = entry.getKey();
        List<Resource> resourceList = entry.getValue();
        String name = "";
        switch (resourceType) {
            case SONG -> name = fileName + "_songs" + extension;
            case VIDEO_RECORDING -> name = fileName + "_videos"
+ extension;
            case ESSAY -> name = fileName + "_essays" +
extension;
        }
        switch (extension) {
            case ".json" ->
DataManagement.exportResourcesToJson(name, resourceList);
            case ".xml" ->
DataManagement.exportResourcesToXML(name, resourceList);
        }
    }
}

```

Super usuario

- Debe haber un super usuario 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 prestamo.

- Actualizo el enum de tipo de usuario con el nuevo rol.

```
public enum UserType {
    SUPERUSER,
    ADMINISTRATOR,
    ASSISTANT,
    READER
}
```

- Creo un usuario con este rol desde la base de datos puesto que no debe poder crearse desde la aplicación

#	id	name	email	password	role	phone	birth_date
1	7	David	dav@id	1234	SUPERUSER	11	2024-11-01

#	Time	Action	Message
1	22:40:44	SELECT * FROM Pinguin_Library.user LIMIT 0, 1000	9 row(s) returned

- Creo el menú particular para el super usuario utilizando la misma arquitectura de los demás menús y en este aprovecho el colocar todos los menús anteriores como opciones para que con este rol se puedan probar todas las funcionalidades de los demás.

```
private static void redirectPersonalizedMenu(User user) {
    switch (user.getRole()) {
        case READER -> readerMenu(user);
        case ADMINISTRATOR -> administratorMenu(user);
        case ASSISTANT -> assistantMenu(user);
        case SUPERUSER -> superuserMenu(user);
    }
}
```

```

public static void superuserMenu(User user) {
    while (true) {
        superuserMenuMessage(user);
        int option = askInt(enterYourOptionMessage);
        switch (option) {
            case 1 -> readerMenu(user);
            case 2 -> assistantMenu(user);
            case 3 -> administratorMenu(user);
            case 4 -> createAdminUser();
            case 5 -> restoreLoans();
            case 6 -> {
                System.out.println(exitingMessage);
                return;
            }
            default -> System.out.println(incorrectOptionMessage);
        }
    }
}

```

- Para la opción de crear usuarios administradores reutilizo la lógica de la opción para crear usuarios asistentes (omitida por brevedad)


Restaurado de los préstamos de prueba:

Creo en el menú de superusuario la opción de restaurar los préstamos realizados.

```

public void restoreSuperUserLoans() {
    List<Loan> superUserLoans =
        loanDAO.getAllLoans().stream().filter(l ->
l.getUser().getRole() == UserType.SUPERUSER).toList();
    for (Loan loan : superUserLoans) {
        finishResourceLoan(loan.getId());
        deleteResourceLoan(loan.getId());
    }
}

```


Welcome to Resources Menu, David  **superuser**

=====

1. See all songs
2. See all video recordings
3. See all essays
4. Select a resource to loan
5. List selected resources for loan
6. Request resource loan
7. See my resource loans
8. See my loan details
9. Back

Enter your option:

> 7

List of your loans:

Loan{id=6, requestedDate=2024-04-19, returnDate=2024-04-20, status=REQUESTED, user=David}
Loan{id=7, requestedDate=2024-04-19, returnDate=2024-11-11, status=REQUESTED, user=David}
Loan{id=8, requestedDate=2024-04-19, returnDate=2024-11-11, status=REQUESTED, user=David}
Loan{id=9, requestedDate=2024-04-19, returnDate=2024-11-11, status=REQUESTED, user=David}
Loan{id=10, requestedDate=2024-04-19, returnDate=2024-11-11, status=REQUESTED, user=David}
Loan{id=12, requestedDate=2024-04-19, returnDate=2024-11-11, status=REQUESTED, user=David}
Loan{id=13, requestedDate=2024-04-19, returnDate=2024-11-11, status=REQUESTED, user=David}
Loan{id=14, requestedDate=2024-04-19, returnDate=2024-11-11, status=REQUESTED, user=David}
Loan{id=16, requestedDate=2024-04-19, returnDate=2024-04-19, status=FINISHED, user=David}

Welcome to Resources Menu, David

=====

Welcome to Superuser Menu, David

=====

1. Reader menu
2. Assistant menu
3. Administrator menu
4. Create user
5. Restore super user loans
6. Back

Enter your option:

> 5

All loans by superuser were restored

Welcome to Superuser Menu, David

=====

```
3. See all essays
4. Select a resource to loan
5. List selected resources for loan
6. Request resource loan
7. See my resource loans
8. See my loan details
9. Back
Enter your option:
> 7
List of your loans: 0 resultados
Welcome to Resources Menu, David
=====
```

Se borran todos los préstamos del superusuario y se restaura el inventario de los recursos prestados.

Perfil de usuario

- **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.**
- Actualizo la tabla de usuarios existente agregando las columnas `phone` y `birth_date`, además actualizo los registros ya existentes rellenando los nuevos datos. (Evidencia en el archivo `InsertDataExampleV2.sql`)
- Creo un submenú de perfil de usuario con las opciones de cambiar contraseña y de actualizar todos sus datos del perfil.
- Lo agrego en cada uno de los menús principales por rol

```
Welcome to Main Menu, David
=====
1. Book options
2. Novel options
3. Other resources options
4. Author options
5. Profile options
6. Back
Enter your option:
> 5
Welcome to Profile Menu, David
=====
1. Update profile
2. Change password
3. Back
Enter your option:
>
```

La lógica de este menú se puede evidenciar en el archivo `UserProfileFunctions` el cual reutiliza los métodos definidos para las opciones de administrar usuarios

Registro de creación de usuarios administradores

- 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.
- Mi solución (debido a que la indicación es realizarlo utilizando triggers en la base de datos) fue colocar una condicional que compruebe si al insertar un usuario, este tiene el rol de administrador, el cual solo puede dar el superusuario y entonces registra en la tabla indicada, cuál fue el usuario agregado y la fecha.

Archivo `admins_log_trigger.sql` adjunto en el repositorio

```

CREATE TABLE IF NOT EXISTS admins_log (
    action VARCHAR(100) NOT NULL,
    user_id VARCHAR(100) NOT NULL,
    user_name VARCHAR(45),
    date DATE NOT NULL
) ENGINE=INNODB;

DELIMITER $$
CREATE TRIGGER log_new_admin_user AFTER INSERT ON user
FOR EACH ROW
BEGIN
    IF NEW.role = 'ADMINISTRATOR' THEN
        INSERT INTO admins_log (action, user_id, user_name, date)
        VALUES ('A superuser created a new admin user',
        LAST_INSERT_ID(), NEW.name, NOW());
    END IF;
END$$

DELIMITER ;

```

#	action	user_id	user_name	date
1	A superuser created a new admin user	123	pablito	2024-04-21
2	A superuser created a new admin user	12356	pablito	2024-04-21

#	Time	Action	Message
1	21:16:26	select * from admins_log LIMIT 0, 1000	2 row(s) returned

Video demostración

<https://youtu.be/je1ejRcQTkU>

Extras:

- Refactoricé todas las interacciones con el usuario, para manejar la entrada de datos incorrecta y mostrar el debido mensaje indicativo.

Listado de funcionalidades no completadas:

Actualización 2.0:

- Algunas de las funcionalidades de la actualización se agregaron solo para los recursos de préstamo nuevos.
- Mejorar el manual de usuario

Versión 1.0:

- Borrado lógico
- Manejo de datos ingresados erroneamente por el usuario
- Importado de csv (si exporta)