Modelo primer parcial

JDBC & Programación Funcional

Se nos solicita el desarrollo de un sistema de gestión para una biblioteca pública que presta libros a sus usuarios de forma gratuita. Este sistema permitirá registrar los libros disponibles, administrar los usuarios y gestionar los préstamos de libros entre los usuarios y la institución.

El objetivo principal del sistema es permitir llevar control de la disponibilidad de los libros, así como los préstamos activos y las devoluciones pendientes. La aplicación también busca ofrecer algunas estadísticas útiles para la administración de la biblioteca, como los libros más solicitados o los clientes más frecuentes.

Debido al funcionamiento de la biblioteca, cada préstamo comprende sólo un libro. Cada usuario puede tener hasta un máximo de cinco libros solicitados en cualquier momento. Al realizar los préstamos, se deberá manejar correctamente las unidades disponibles del libro y se deberá siempre comprobar la disponibilidad del libro antes de permitir un préstamo.

Base de Datos

```
CREATE TABLE usuarios (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      nombre TEXT NOT NULL,
      email TEXT UNIQUE NOT NULL
);

CREATE TABLE libros (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      titulo TEXT NOT NULL,
      autor TEXT NOT NULL,
      anio_publicacion INTEGER,
      unidades_disponibles INTEGER
);

CREATE TABLE prestamos (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      libro_id INTEGER NOT NULL,
      usuario_id INTEGER NOT NULL,
      fecha_prestamo DATE NOT NULL DEFAULT (DATE('now')),
      fecha_devolucion DATE,
      FOREIGN KEY (libro_id) REFERENCES Libros(id) ON DELETE CASCADE,
      FOREIGN KEY (usuario_id) REFERENCES Usuarios(id) ON DELETE CASCADE
);
```

Se nos provee también la siguiente clase DatabaseConnection.

```java
public class DatabaseConnection {

	private static final String URL = "jdbc:sqlite:biblioteca.db";
	private static final DataSource datasource;

	static {

		HikariConfig config = new HikariConfig();
		config.setJdbcUrl(URL);
		config.setConnectionInitSql("PRAGMA foreign_keys = ON;");
		datasource = new HikariDataSource(config);
	}

	public static Connection getConnection() throws SQLException {
	return datasource.getConnection();
	}
}
```

Recordar incluir las dependencias de HikariPool y el driver de SQLite

```xml
<!-- https://mvnrepository.com/artifact/com.zaxxer/HikariCP -->
<dependency>
   <groupId>com.zaxxer</groupId>
   <artifactId>HikariCP</artifactId>
   <version>6.3.0</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.xerial/sqlite-jdbc -->
<dependency>
   <groupId>org.xerial</groupId>
   <artifactId>sqlite-jdbc</artifactId>
   <version>3.49.1.0</version>
</dependency>

<dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <version>1.18.38</version>
            <scope>provided</scope>
</dependency>
```

Actividades

Al realizar las operaciones, prestar mucha atención a las constraints de la base de datos y manejar las excepciones que puedan ocurrir. Se nos solicita que el sistema implemente patrón singleton en los repositorios, para optimizar su funcionamiento. El código debe ser prolijo y estar correctamente separado en capas y en carpetas. Se nos solicitan los siguientes requisitos funcionales. Los filtrados y manipulación de datos deben ser realizados mediante programación funcional.

Requisitos Funcionales

RF01 - El sistema deberá permitirle al Bibliotecario realizar alta y baja de usuarios.
RF02 - El sistema deberá permitirle al Bibliotecario listar todos los usuarios
RF03 - El sistema deberá permitirle al Bibliotecario listar todos los usuarios con préstamos activos.
RF04 - El sistema deberá permitirle al Bibliotecario generar un préstamo nuevo.
RF05 - El sistema deberá permitirle al Bibliotecario marcar un préstamo como devuelto.
RF06 - El sistema deberá permitirle al Bibliotecario visualizar todos los préstamos.
RF07 - El sistema deberá permitirle al Bibliotecario visualizar todos los préstamos activos.
RF08 - El sistema deberá permitirle al Bibliotecario visualizar el libro más prestado.
RF09 - El sistema deberá permitirle al Bibliotecario visualizar el total de libros disponibles.
RF10 - El sistema deberá permitirle al Bibliotecario visualizar todos los libros.
RF11 - El sistema deberá permitirle al Bibliotecario visualizar el usuario con mayor número de préstamos históricos en el sistema.
RF12 - El sistema deberá permitirle al Bibliotecario visualizar el promedio de préstamos por usuario, pero solo teniendo en cuenta a los usuarios que tienen por lo menos un préstamo.

RNF01 - El sistema debe estar correctamente separado en capas Repositorio y Servicios.

RNF02 - El sistema debe separar los comportamientos de sus implementaciones. (Segregación de interfaces)