Universidad Nacional de Colombia - sede Bogotá
Facultad de Ingeniería
Departamento de Sistemas e Industrial
Curso: Ingeniería de Software 1 (2016701)

🧩 Patrón de Diseño: DAO (Data Access Object)

🔷 1. ¿Qué es el patrón DAO?

El patrón DAO (Data Access Object) es un patrón de diseño estructural que proporciona una abstracción sobre las operaciones de persistencia en una base de datos. En lugar de que múltiples clases accedan directamente a la base de datos con SQL embebido, el patrón DAO centraliza estas operaciones en clases específicas, que se encargan de comunicar el dominio del sistema con la fuente de datos.

🎯 Propósito:

- Separar la lógica de negocio del acceso a datos.
- Aislar el código SQL, facilitando cambios o migraciones de base de datos.
- Fomentar una arquitectura limpia, mantenible y testeable.

🛠️ 2. ¿Cómo fue implementado en el proyecto?

En el proyecto Scalia, el patrón DAO fue implementado para gestionar el acceso a datos relacionados con los usuarios (por ahora). A continuación, se presentan fragmentos relevantes del código.

📁 Clase: `UserDAO.java`

```java
public class UserDAO {

    public boolean createUser(User user) {
```

```
        String sql = "INSERT INTO users (username, email, password,
first_name, last_name, created_at, updated_at) VALUES (?, ?, ?, ?, ?, ?,
?)";
        try {
            LocalDateTime now = LocalDateTime.now();
            int userId = DatabaseConnection.executeInsert(sql,
                user.getUsername(),
                user.getEmail(),
                user.getPassword(),
                user.getFirstName(),
                user.getLastName(),
                now,
                now
            );

            return userId != -1;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }
}
```

📁 Clase de soporte: `DatabaseConnection.java`

Contiene métodos estáticos reutilizables para ejecutar queries, como:

Java

```
public static int executeInsert(String sql, Object... params) throws
SQLException {
    Connection conn = getConnection();
    try (PreparedStatement stmt = conn.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS)) {
        for (int i = 0; i < params.length; i++) {
            stmt.setObject(i + 1, params[i]);
        }
        int affectedRows = stmt.executeUpdate();
        if (affectedRows > 0) {
            ResultSet keys = stmt.getGeneratedKeys();
            if (keys.next()) return keys.getInt(1);
        }
        return -1;
    }
```

```
}
```

📌 3. Justificación del patrón en el contexto del proyecto

✅ Separación de responsabilidades

El patrón DAO permite que los **controladores JavaFX** (`LoginController`, `RegisterController`, etc.) estén enfocados únicamente en manejar eventos de la interfaz gráfica. Toda la lógica relacionada con base de datos está encapsulada en `UserDAO`.

✅ Facilidad de mantenimiento y testing

Al concentrar las consultas SQL en una sola clase, es más sencillo realizar cambios en la estructura de la base de datos, o escribir pruebas unitarias sobre métodos como `createUser()`.

✅ Escalabilidad

El uso de DAO facilita agregar nuevas entidades (Instrumento, Categoría, Usuario, etc.), cada una con su propio DAO, sin mezclar la lógica de persistencia entre capas del sistema.