Padrões de Projeto na Camada de Persistência

Este documento descreve os principais padrões de projeto utilizados na camada de persistência do sistema de aluguel de salas. O objetivo é garantir boas práticas de engenharia de software, como separação de responsabilidades, desacoplamento e maior testabilidade do código.

1. Repository Pattern

O padrão **Repository** foi utilizado para definir a interface de acesso a dados, permitindo que a lógica de persistência fique separada da lógica de negócio. Isso facilita a manutenção e possibilita a troca da implementação sem impactar o restante do sistema.

```
Exemplo: UsuarioRepository.java public interface UsuarioRepository { void
salvar(Usuario usuario); Usuario buscarPorId(Long id); List listarTodos(); void
atualizar(Usuario usuario); void deletar(Long id); }
```

2. DAO (Data Access Object) Pattern

O padrão **DAO** foi aplicado para implementar a lógica de acesso ao banco de dados utilizando JDBC. Dessa forma, o código SQL fica isolado em uma classe específica, facilitando o reuso e a manutenção.

```
Exemplo: UsuarioRepositoryImpl.java public class UsuarioRepositoryImpl implements
UsuarioRepository { private Connection conn; public UsuarioRepositoryImpl(Connection
conn) { this.conn = conn; } @Override public void salvar(Usuario usuario) { try {
PreparedStatement stmt = conn.prepareStatement( "INSERT INTO usuario (nome, email)
VALUES (?, ?)"); stmt.setString(1, usuario.getNome()); stmt.setString(2,
usuario.getEmail()); stmt.executeUpdate(); } catch (SQLException e) {
e.printStackTrace(); } } }
```

3. Singleton + Factory Pattern

O padrão **Singleton** foi combinado com o **Factory** para gerenciar a criação da conexão com o banco de dados. Assim, o sistema garante que apenas uma instância de conexão seja utilizada, reduzindo o consumo de recursos e centralizando a lógica de configuração de acesso ao banco.

```
Exemplo: ConnectionFactory.java public class ConnectionFactory { private static
Connection conn; private ConnectionFactory() {} public static Connection
getConnection() { if (conn == null) { try { conn = DriverManager.getConnection(
"jdbc:mysql://localhost:3306/aluguel_salas", "root", "senha"); } catch (SQLException
e) { e.printStackTrace(); } } return conn; } }
```

Conclusão:
A aplicação dos padrões Repository, DAO, Singleton e Factory trouxe maior organização à camada de persistência, permitindo desacoplamento entre regras de negócio e persistência de dados, além de facilitar testes e manutenção futura.