Relatório de Refatoração ETAPA 6

Princípios SOLID Aplicados

1. Single Responsibility Principle (SRP)

Definição: Cada classe deve ter uma única responsabilidade, ou seja, deve existir apenas uma razão para ela mudar.

Aplicação: - Classes Afetadas:
- UserService, CustomerService, SaleService, ExpenseService, ServiceOrderService, LogService, ReportService, SaleItemService - DAOs: UserDAO, CustomerDAO, SaleDAO, ExpenseDAO, ExpenseCategoryDAO, ServiceOrderDAO, SystemLogDAO, SaleItemDAO - Modelos: User, Customer, Sale, SaleItem, Expense, ExpenseCategory, ServiceOrder, SystemLog

Justificativa:
Cada classe de serviço (Service) é responsável por regras de negócio de uma entidade específica, enquanto cada DAO é responsável apenas pela persistência de dados dessa entidade. Os modelos representam apenas os dados e suas regras internas. Isso facilita manutenção, testes e evolução do sistema.

---

2. Open/Closed Principle (OCP)

Definição: Classes devem estar abertas para extensão, mas fechadas para modificação.

Aplicação: - Classes Afetadas:
- DAOs implementam a interface genérica DAO<T>, permitindo que novas entidades sejam facilmente adicionadas sem modificar as existentes. - Serviços podem ser estendidos para adicionar novas regras de negócio sem alterar o funcionamento básico.

Justificativa:
A interface DAO<T> permite que novas implementações sejam criadas para outras entidades sem alterar o contrato das operações básicas (CRUD). Os métodos dos serviços podem ser estendidos ou sobrescritos conforme novas necessidades surgem.

---

3. Liskov Substitution Principle (LSP)

Definição: Subtipos devem ser substituíveis por seus tipos base sem alterar o funcionamento do programa.

Aplicação: - Classes Afetadas:
- DAOs que implementam DAO<T> podem ser usados polimorficamente. - Enumerações como UserType, PaymentMethod, ServiceStatus, RecurrenceType são usadas para garantir tipos seguros e previsíveis.

Justificativa:
O uso da interface DAO<T> garante que qualquer DAO pode ser utilizado onde se espera um DAO genérico, sem causar erros de comportamento. O uso de enums para status e tipos reforça a previsibilidade e substituição segura.

---

4. Interface Segregation Principle (ISP)

Definição: Os clientes não devem ser forçados a depender de interfaces que não utilizam.

Aplicação: - Classes Afetadas:
- A interface DAO<T> define apenas métodos essenciais para persistência, evitando métodos desnecessários para entidades que não precisam deles.

Justificativa:
A interface é enxuta e específica para operações de persistência, evitando que classes sejam obrigadas a implementar métodos irrelevantes.

---

5. Dependency Inversion Principle (DIP)

Definição: Dependa de abstrações, não de implementações concretas.

Aplicação: - Classes Afetadas:
- Serviços dependem de DAOs via interface (DAO<T>), e não diretamente de implementações concretas. - O uso de HibernateUtil para obter o EntityManager abstrai o mecanismo de persistência.

Justificativa:
Os serviços podem receber DAOs diferentes (inclusive mocks para testes) sem alterar seu funcionamento. O acesso ao banco é abstraído pelo utilitário, facilitando mudanças futuras na tecnologia de persistência.

Padrão de Projeto Utilizado

DAO (Data Access Object)

Definição: O padrão DAO separa a lógica de acesso a dados da lógica de negócio, encapsulando operações de persistência em classes específicas.

Aplicação: - **Classes Afetadas:**
- Todas as classes no pacote dao (UserDAO, CustomerDAO, etc.) implementam o padrão DAO. - A interface DAO<T> define o contrato para operações CRUD.

Justificativa:
O uso do padrão DAO facilita a manutenção, testes e evolução do sistema, pois separa claramente as responsabilidades. Permite trocar o mecanismo de persistência (por exemplo, de Hibernate para JDBC) sem afetar a lógica de negócio. Também facilita a reutilização e o reaproveitamento de código.

Service Layer

Definição: O padrão Service Layer organiza a lógica de negócio em serviços, separando-a da camada de persistência e da apresentação.

Aplicação: - **Classes Afetadas:**
- Todas as classes no pacote service (UserService, SaleService, etc.)

Justificativa:
Centraliza regras de negócio, validações e operações complexas, facilitando testes, manutenção e reutilização. Permite que a camada de apresentação (ex: controllers) interaja com a lógica de negócio de forma consistente.

Resumo

- **SOLID:** Todos os princípios foram aplicados, principalmente SRP, OCP, LSP, ISP e DIP, garantindo um código modular, extensível e fácil de manter.
- **Padrões de Projeto:** DAO e Service Layer são utilizados para separar responsabilidades, facilitar testes e promover reutilização.
- **Benefícios:** O sistema está preparado para crescer, ser testado e sofrer alterações tecnológicas com mínimo impacto nas regras de negócio.