Banco NEB - Documentação Técnica CRUD

1. Visão Geral

Objetivo do Projeto:

O objetivo deste projeto é simular dois sistemas bancários integrados em uma única plataforma: um sistema voltado para **funcionários** e um **aplicativo móvel para clientes**. A proposta foi solicitada pelo professor Moak, da disciplina de Banco de Dados, na Instituição de Ensino SENAI Bahia.

Devido a limitações de tempo, não foi possível desenvolver os dois sistemas completos conforme o planejado. Embora a estrutura dos dois sistemas tenha sido idealizada, o foco principal foi na implementação do banco de dados, abrangendo as funcionalidades tanto para **funcionários** quanto para **clientes**. Contudo, no back-end, o CRUD (Create, Read, Update, Delete) foi implementado exclusivamente para a parte de **clientes**, utilizando Java. O banco de dados foi projetado e configurado para suportar as operações para ambos os sistemas, mas com a parte do cliente sendo priorizada.

Devido à natureza da disciplina e ao uso do **MySQL**, algumas funcionalidades que normalmente seriam implementadas no back-end foram realizadas diretamente no banco de dados. Isso demonstra o domínio na utilização da ferramenta de gerenciamento de banco de dados, aplicando conceitos de **SQL**, **procedures** e **triggers**.

---

2. Arquitetura do Projeto

Este trabalho tem como objetivo praticar a organização e a modelagem de projetos através da implementação de uma arquitetura em camadas. As camadas são subgrupos dentro do projeto que agrupam classes com funções semelhantes, proporcionando maior controle organizacional e facilitando a manutenção do código.

Abaixo, descrevemos as camadas do sistema e suas respectivas responsabilidades:

2.1 Camada Main

- **Função:** Responsável por iniciar a aplicação e interagir diretamente com o usuário. A camada *Main* se comunica com a camada *Controller*, solicitando funções e informações de acordo com as interações do usuário.
- **Nomenclatura:** Não possui convenção específica de nomenclatura e serve como ponto de entrada do sistema.

2.2 Camada Controller

- **Função:** Gerencia as interações entre o usuário e o sistema, atuando como intermediária entre a camada *Service* e a aplicação principal (Main). Realiza verificações iniciais, como validação de CPF ou senha, garantindo que os dados estejam no formato adequado antes de serem enviados para processamento na camada *Service*.
- **Nomenclatura:** As classes nesta camada começam com "Controller", por exemplo, ControllerCliente.

2.3 Camada Service

- **Função:** Serve como intermediária entre as requisições vindas do *Controller* e as camadas *DAO* e *DTO*, onde a lógica de negócios é aplicada e processada antes de interagir com o banco de dados.
- **Nomenclatura:** As classes nesta camada começam com "Service", por exemplo, ServiceCliente.

2.4 Camada DTO (Data Transfer Object)

- **Função:** Armazena dados temporariamente antes de serem utilizados em conjunto com o banco de dados, facilitando a transferência de dados entre camadas.
- **Nomenclatura:** As classes nesta camada terminam com "DTO", por exemplo, ClienteDTO.

2.5 Camada DAO (Data Access Object)

- **Função:** Contém as classes responsáveis por gerar requisições ao banco de dados, retornando valores ou realizando verificações de forma organizada e segura.
- **Nomenclatura:** As classes nesta camada terminam com "DAO", por exemplo, ClienteDAO.

2.6 Camada Util

- **Função:** Contém classes utilitárias que podem ser usadas em qualquer parte do programa para melhorar a eficiência do código, agregando funções de uso comum.
- **Nomenclatura:** As classes desta camada começam com "Util", por exemplo, UtilValidador.

Diagrama do Fluxo de Camadas