PROJETO PRÁTICO ESTÁGIO

Descrição do projeto:

O sistema de cadastro de clientes é uma aplicação em **Java** que interage com um banco de dados **MySQL**. Seu objetivo principal é permitir que os usuários registrem informações detalhadas sobre clientes, como nome, e-mail, senha, CPF, tipo de endereço, endereço e CEP. Além disso, o sistema oferece a funcionalidade de busca por CPF, que também impossibilita a criação de mais de uma conta por CPF.

A aplicação foi desenvolvida sobre um sistema de terminal como parte de um protótipo, com a estrutura de código atual é possível adicionar novas funções facilmente além de poder escalar para uma aplicação web ou uma interface gráfica (GUI).

Funcionalidades do sistema:

Cadastro de Clientes: Os usuários podem registrar clientes com campos obrigatórios e opcionais. A validação dos dados é realizada para garantir a integridade das informações, verificando a validade do CPF, a força da senha e etc.

Busca por CPF: O sistema permite que os usuários consultem informações de clientes cadastrados utilizando o CPF. Isso proporciona uma maneira rápida e eficiente de acessar dados, essencial para a gestão de informações.

Prevenção de Duplicidade: A aplicação garante que cada CPF seja único, evitando a duplicação de registros no banco de dados.

Estrutura do código:

Main: Esta classe é o ponto principal da aplicação e controla a interação com o usuário. Ela apresenta um menu com opções de cadastro e busca, coleta as entradas do usuário e controla o fluxo da aplicação. A

simplicidade desta classe permite que novos desenvolvedores facilmente entendam como a aplicação é estruturada.

ClienteDTO: Define a estrutura de dados dos clientes. Com métodos getters e setters, ela encapsula as informações do cliente: nome, e-mail, senha, CPF, tipo de endereço, endereço e CEP. Essa abordagem de encapsulamento garante que os dados sejam manipulados de maneira controlada, evitando acessos diretos e indevidos.

ClienteDAO: Gerencia a comunicação com o banco de dados. Ele contém métodos para cadastrar clientes e buscar informações específicas. A centralização das operações de banco de dados nesta classe não apenas melhora a organização do código, mas também simplifica a manutenção, já que qualquer alteração na lógica de acesso a dados pode ser feita em um único lugar, ela tem 3 métodos, um para o cadastro do cliente, um para a consulta de CPF, e outro para verificar se o CPF já existe no banco de dados.

HashUtil: Implementa a segurança das senhas, utilizando hashing para armazená-las de forma segura no banco.

ValidadorCliente: A classe ValidadorCliente lida com a validação dos dados inseridos pelo usuário. Ela verifica se os campos estão preenchidos corretamente, validando CPF, e-mail, senha e outros critérios. Essa estrutura permite que a lógica de validação seja facilmente ajustada ou expandida conforme novas regras de negócio forem necessárias, sem impactar as outras partes do sistema. Por enquanto ela é simples, apenas verificando o tamanho dos dados, e formatando eles como o CPF e CEP, para a legibilidade e normalização no banco de dados.

***DatabaseConnection*:** Esta classe centraliza a lógica de conexão com o banco de dados MySQL, ao encapsular a configuração de conexão, ela simplifica o acesso ao banco e facilita ajustes futuros, como alterações de credenciais ou mudança de banco.

Tecnologias usadas:

As tecnologias escolhidas para o desenvolvimento do sistema são robustas e amplamente reconhecidas por sua eficácia e confiabilidade. O uso da linguagem de programação Java, por exemplo, é uma decisão estratégica, pois ela é frequentemente utilizada em aplicações empresariais, oferecendo alta performance e um ecossistema rico em bibliotecas e frameworks.

O banco de dados MySQL, utilizado para o armazenamento e manipulação de dados, é conhecido por sua eficiência e capacidade de lidar com grandes volumes de informações de forma estruturada. Essa confiabilidade é fundamental, especialmente em sistemas que gerenciam dados sensíveis, como informações pessoais de clientes.

Possíveis implementações:

O sistema de cadastro de clientes demonstra um considerável potencial de evolução, principalmente devido à sua arquitetura modular e bem estruturada. Essa estrutura permite a adição de novas funcionalidades de maneira fluida e eficiente, sem comprometer a integridade do sistema existente. Uma das direções mais promissoras para o futuro da aplicação é a integração com APIs externas. Por meio dessa integração, seria possível validar dados inseridos no sistema, como CPF e endereços, utilizando serviços confiáveis. Isso não apenas aumentaria a precisão das informações cadastradas, mas também elevaria a confiabilidade do sistema como um todo.

Uma transição de uma aplicação de terminal para uma interface gráfica (GUI) poderia ser um grande passo na melhoria da experiência do usuário. Uma interface visual não apenas tornaria a interação mais intuitiva, mas também tornaria o sistema mais acessível a um público mais amplo, incluindo aqueles que não estão familiarizados com comandos de texto.