

PROJETO CRUD MAVEN, HIBERNATE E POSTGRES

O presente projeto consiste em um CRUD na linguagem de programação Java com uma Tela de Cadastro de Clientes, sendo acessado através de um menu específico na Tela Principal do JForm com auxílio da IDE Netbeans, para este projeto de CRUD foi utilizado o padrão de projetos Maven, o banco de dados Postgres e o framework de persistência Hibernate.

1 O Projeto CRUD com Hibernate

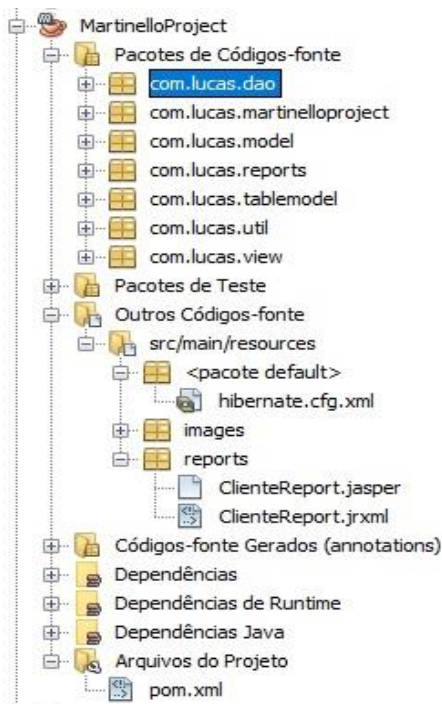
O projeto foi desenvolvido na linguagem de programação Java, através da IDE Netbeans 8.2, com Java JDK 8.

O padrão adotado no desenvolvimento foi o MVC (Model View Controller), pois é um padrão em que o desenvolvimento fica melhor organizado, e posteriormente uma melhor legibilidade das classes e códigos.

O presente projeto utiliza o Maven para a sua organização, esta escolha se deve uma melhor portabilidade do projeto, pois através do arquivo pom.xml podemos configurar e carregar todas as dependências necessárias para o projeto rodar, isso evita de ficarmos procurando manualmente as libs necessárias, basta apenas verificar no MVN Repository (repositório oficial Maven) a lib necessária e acrescentar uma nova dependência no arquivo pom.xml, depois compilamos o projeto para que seja feito o download das dependências citadas .

2 Estrutura do Projeto

O presente projeto tem o nome de MartinelloProject e foi criado através do padrão Maven.



O projeto conta com aproximadamente 10 pacotes, onde destacamos:

- *com.lucas.dao*
- *com.lucas.martinelloproject*
- *com.lucas.model*
- *com.lucas.reports*
- *com.lucas.tablemodel*
- *com.lucas.util*
- *com.lucas.view*

E na organização Outros Código-fonte está o pacote do Maven `src/main/resources` que contém os recursos a serem integrados ao projeto, tais como: o arquivo `hibernate.cfg.xml`, o pacote de imagens e o pacote com os arquivos de relatórios.

2.1 Pacote *com.lucas.dao*

O pacote *com.lucas.dao* contém a classe `ClienteDAO.class` que é responsável por fazer a persistência com o banco de dados, esta classe contém 03 atributos de conexão: `factory`, `transaction` e `session` em que respectivamente instanciam das classes `Factory`, `Transaction` e `Session` que são importantes para a transação com o banco de dados.

O objeto `factory` da classe `Factory` faz a conexão com a classe `HibernateUtil.class` que está no pacote *com.lucas.uti*, onde esta classe é a

responsável pela conexão com o arquivo hibernate.cfg.xml que contém os detalhes da conexão com o banco de dados.

Assim em cada método da classe ClienteDAO, é feita a abertura da sessão de conexão através do objeto session, que posteriormente o objeto transaction faz o início de uma transação com o banco, que se estiver tudo “ok” ele faz o commit no banco de dados, senão ele lança uma exceção e faz o rollback.

A classe contém um método construtor ClienteDAO() em que faz a operação da conexão com o banco através da classe HibernateUtil.

Além disso, contém os métodos CRUD: create, update, delete, select e findCliente.

O método create(Cliente cliente) do tipo void é responsável pela inserção do cadastro do cliente no banco de dados, este método contém um parâmetro do tipo Objeto Cliente, em que conecta com a classe Cliente que está no pacote com.lucas.model que é responsável por captar os dados digitados pelo usuário através dos métodos setters de seus atributos e faz o retorno através dos getters para que então possa ser “commitado” no banco de dados.

O método update(Cliente cliente) do tipo void é responsável pela alteração dos dados retornados através de uma consulta em um table, o seu comportamento é semelhante ao método create(), no entanto, diverge através da linha session.merge(cliente) em que este faz o update no banco de dados, ao contrário da linha session.save(cliente) que está presente no método create(), em que o mesmo é responsável pela inserção.

O método delete(Cliente cliente) do tipo void é responsável por deletar um dado através do Objeto Cliente.

O método select(long id) do tipo Cliente é responsável por pesquisar um determinado cliente através do atributo id, filtrando assim, especificadamente um determinado cliente.

O método findCliente() do tipo List é responsável por listar todos os cadastros salvos no banco de dados, ele retorna os dados do cliente através de um ArrayList.

2.2 Pacote com.lucas.martinelloproject

O pacote com.lucas.martinelloproject contém uma classe chamada Principal.class, esta classe é responsável por definir qual JForm poderá ser aberto quando rodar o projeto, tornando assim, uma maneira dinâmica para controlar qual tela do Java Swing poderá ser aberto primeiro ao carregar o

sistema, isso é possível, pois esta classe contém um método principal public static void main() para realizar esta operação.

2.3 Pacote com.lucas.model

O pacote com.lucas.model contém uma das principais classes do sistema, em nosso projeto, temos apenas a classe Cliente, por que ela é uma das mais importantes?, - pois nela definimos as diretrizes, as regras de negócio através de seus atributos, ou seja, é esta classe que irá criar uma nova tabela no banco de dados através das @Annotations do Hibernate.

Desse modo, não precisamos mais criar e configurar as nossas tabelas através de comandos SQL na Query do banco de dados, pois a própria classe se torna uma table (Entidade) e seus atributos se tornam fields (colunas), isto graças ao poder do framework Hibernate que através do Mapping da classe, torna esta dinâmica possível.

2.4 Pacote com.lucas.reports

O pacote com.lucas.reports contém a classe ClienteRelatorio que tem métodos de acesso e visualização dos relatórios gerados através do JasperReport, os relatórios estão salvos no caminho "src/main/resources/reports ", onde em nosso projeto, o acesso se dá ao arquivo ClienteReport.jrxml.

2.5 Pacote com.lucas.tablemodel

O pacote com.lucas.tablemodel contém uma classe ClienteTableModel, esta classe é responsável por configurar a visualização da consulta dos clientes através uma table presente na Tela de Cadastro do Cliente, ou seja, quando é realizado um cadastro de um cliente, logo após, a sua inserção através do botão Salvar, os dados do cliente são carregados automaticamente na table de pesquisa, que se localiza logo abaixo dos fields de cadastro.

Cadastro de clientes

Novo Salvar Excluir Cancelar Imprimir

Formulário

Categoria: ☒ Física ☐ Jurídica

Nome: Data de Cadastro: 26/10/2020

CPF/CNPJ: CEP:

Endereço: Número:

Complemento:

Bairro: Cidade: UF: MT

Telefone:

Email: Situação: Ativo

Nome	CPF/CNPJ
Lucas de Souza Oliveira	019.065.781-25

2.6 Pacote com.lucas.util

O pacote com.lucas.util contém duas classes: HibernateUtil e ValidaCpfCnpj.

A classe HibernateUtil é uma classe que faz a ponte com a conexão do banco de dados, esta classe tem acesso ao arquivo hibernate.cfg.xml, que contém as configurações de conexão com o banco de dados, portanto é uma classe controladora de conexão com as classes DAO do sistema.

A classe ValidaCpfCnpj é responsável pela validação do campo CPF/CNPJ presente na tela de cadastro do cliente, sendo assim, só é possível fazer um cadastro de maneira legítima com estes documentos.

2.7 Pacote com.lucas.view

O pacote com.lucas.view é responsável pela organização das classes JForm, as telas do sistema, este pacote contém 5 classes JForm: ViewBuscaCliente, ViewCliente, ViewPrincipal, ViewRelatorioCliente, ViewSobre.

O JForm ViewBuscaCliente é utilizado quando vamos pesquisar algum cliente através de um botão de pesquisa, assim ele é acionado e possui um campo de pesquisa e table para exibir as informações pesquisadas, sendo assim, ao selecionar um cliente da table de resultados, após um duplo clique ele transfere as informações do cliente selecionado ao field de destino.

O JForm ViewCliente é responsável pelo cadastro de todas as informações do cliente, é a principal tela de cadastro do projeto, nela contém os campos de cadastro, os botões para iniciar um novo cadastro, salvar, excluir, cancelar e imprimir os dados dos clientes, além de conter uma table para a exibição dos cadastros armazenados no banco de dados.

O JForm ViewPrincipal é a tela de início do sistema, nela contém os menus de acesso aos forms e alguns atalhos, é a tela principal tela de exibição.

O JForm ViewRelatorioCliente é responsável pela consulta individualizada de um determinado cliente e por fim ao clicar no botão consultar gera um relatório específico do cliente selecionado.

O JForm ViewSobre contém os dados do desenvolvedor deste projeto.

2.8 Pasta src/main/resources

Esta é a pasta de recursos do projeto do Maven, em nosso projeto utilizamos ela para armazenarmos o arquivo de conexão do hibernate com o banco de dados, imagens e relatórios do sistema.

3 Telas do Sistema

3.1 Tela ViewCliente

Este JFrame pode ser acessado via ViewPrincipal através do menu Cadastros – Cadastrar Cliente, pela tecla de atalho F3 ou através da imagem de um Cliente no atalho rápido do sistema.

O projeto contém a tela ViewCliente como responsável pelo CRUD da aplicação. Ao entrar nesta página, automaticamente uma JTable é carregada com as informações de clientes cadastrados no banco de dados, isso deve, através do método carregarDados() que lista todos os clientes.


Cadastro de clientes

Novo Salvar Excluir Cancelar Imprimir

Formulário

Categoria: ☐ Física ☐ Jurídica

Nome: Data de Cadastro:

CPF/CNPJ: CEP: 

Endereço: Número:

Complemento:

Bairro: Cidade: UF:

Telefone: Situação:

Email:

Nome	CPF/CNPJ
Lucas de Souza Oliveira	019.065.781-25

De imediato todos os fields estão desabilitados, os mesmos só serão habilitados se clicar no botão Novo para incluir um novo cadastro ou com um duplo clique no JTable em um cliente selecionado para fazer alguma alteração.

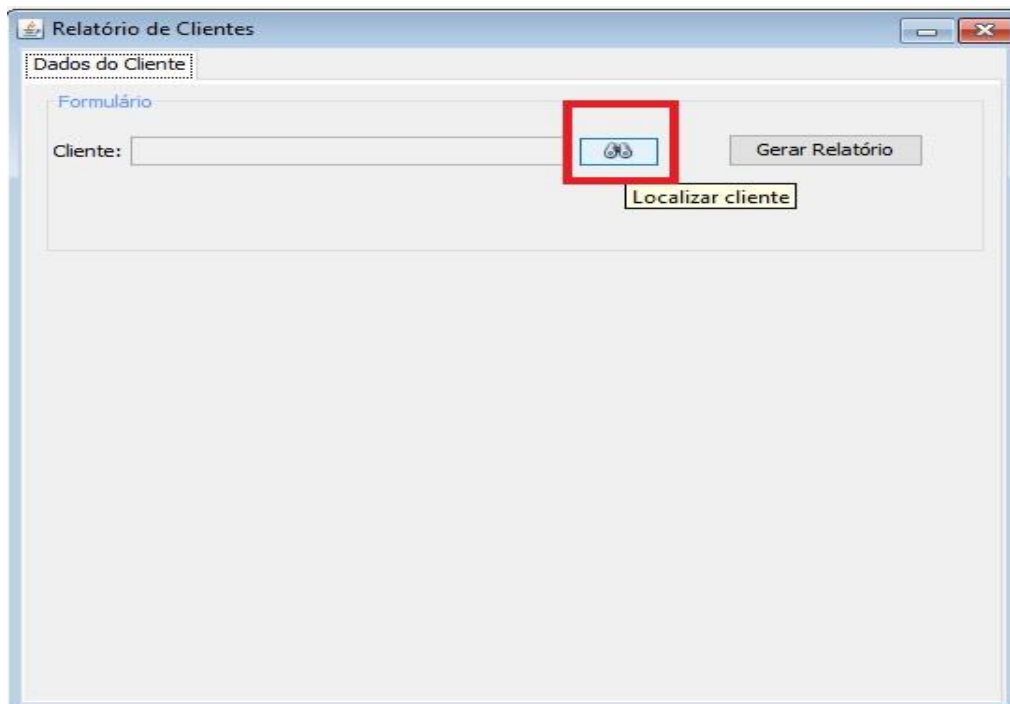
Alguns campos do formulário possuem máscaras e validações:

- O campo CPF/CNPJ contém máscaras que alternam de acordo com a escolha de Pessoa Física ou Jurídica, além disso, contém validação de CPF e CNPJ, onde só pode ser inserido CPF e CNPJ legítimos.
- O campo CEP utiliza API do ViaCEP, assim, basta digitar o CEP de seu endereço e clicar no botão de pesquisa (representado pela imagem de um binóculo), que automaticamente irá carregar os campos de Endereço, Complemento, Cidade e UF.
- O campo Email contém validação, assim o email deverá ter pelo menos @, . e .com.
- O campo Telefone tem máscara para DDD e celular com 9 dígitos.

Com isso o cadastro de um Cliente se dá de maneira legítima, assim após o cadastro, ao clicar no botão Imprimir, será impresso um relatório com todos os dados dos clientes armazenados no banco de dados.

3.2 Tela ViewRelatorioCliente

Este JFrame pode ser acessado via ViewPrincipal através do menu Relatórios – Relatórios por Cliente, pela tecla de atalho F4 ou através da imagem de um Relatório no atalho rápido do sistema.



Este JFrame é responsável por gerar um relatório específico de um cliente selecionado, sendo assim, o usuário ao entrar no sistema terá de primeiramente clicar no botão que tem a imagem de um binóculo com o texto de Localizar Cliente, e será direcionado para tela de BuscaCliente, onde serão listados todos os clientes armazenados no banco de dados, o mesmo pode fazer uma busca pelo campo Filtro ou já selecionar diretamente em um resultado obtido, assim ao escolher um cliente através de um duplo clique na tabela de resultados, será direcionado de volta a tela ViewRelatorioCliente e depois basta clicar no botão Gerar Relatório.

4 Conclusão

O presente projeto demonstra que é possível desenvolver um programa Desktop com Java seguindo os melhores padrões de desenvolvimento, principalmente com o uso do framework Hibernate a fim de agilizar na criação e configuração da persistência com o banco de dados.

