**Programação Orientada a Objetos**

**Henrique dos Santos de Oliveira**

**Professor-José Eduardo Soares de Lima**

Escola Ulbra São Lucas

10/09/2024

RESUMO

Este trabalho aborda conceitos fundamentais no desenvolvimento de software com Java, focando em operações de banco de dados com o padrão CRUD, integração com SQL usando JDBC, criação de interfaces gráficas com Swing e a aplicação do padrão MVC para a organização em camadas. Exploramos como conectar um aplicativo Java a um banco de dados SQL, como construir interfaces gráficas eficientes utilizando o NetBeans e, por fim, como implementar o padrão MVC, facilitando a manutenção e o desenvolvimento modular das aplicações.

**Palavras-chave:** CRUD, JDBC, SQL, Swing, JavaFX, MVC, Padrão de arquitetura, NetBeans

* **INTRODUÇÃO**

O desenvolvimento de aplicações Java envolve uma série de técnicas que permitem a criação de soluções robustas e escaláveis. Desde a interação com bancos de dados até a implementação de interfaces gráficas, o uso de boas práticas e padrões, como o MVC, é fundamental para garantir a qualidade do software. Este trabalho visa explorar esses conceitos de maneira prática, demonstrando como realizar operações CRUD com SQL, criar interfaces gráficas com Swing e organizar o código utilizando o padrão de arquitetura MVC.

* **DEFINIÇÃO DE CRUD.**

"CRUD" é um acrônimo que representa as quatro operações básicas que podem ser realizadas em um banco de dados ou em uma aplicação que manipula dados. A sigla CRUD vem das iniciais das operações em inglês:

Create (Criar): Esta operação adiciona novos dados ao sistema. Em um banco de dados, isso geralmente envolve a inserção de novos registros em uma tabela. Por exemplo, adicionar um novo usuário a um sistema.

Read (Ler): Refere-se à operação de recuperação de dados existentes. Isso pode envolver a consulta e visualização de informações armazenadas no banco de dados. Por exemplo, exibir uma lista de todos os usuários cadastrados.

Update (Atualizar): Envolve a modificação dos dados existentes. Isso pode incluir a alteração de informações de registros já existentes no banco de dados. Por exemplo, atualizar o endereço de um usuário.

Delete (Excluir): Esta operação remove dados do sistema. Isso pode envolver a exclusão de registros de uma tabela no banco de dados. Por exemplo, deletar um usuário que não faz mais parte do sistema.

Essas operações são fundamentais para a manipulação de dados em muitas aplicações e sistemas de gerenciamento de banco de dados (DBMS). A implementação adequada do CRUD garante que os usuários possam gerenciar informações de forma eficaz e eficiente.

Em um contexto mais amplo, CRUD pode ser aplicado a diferentes tipos de armazenamento de dados, não se limitando a bancos de dados relacionais. Por exemplo, em aplicações web, os princípios CRUD são frequentemente usados para gerenciar informações de forma estruturada e acessível.

* **COMO FAÇO A RELAÇÃO DO MEU APLICATIVO EM JAVA COM UM BANCO DE DADOS SQL.**

Para estabelecer a relação entre um aplicativo Java e um banco de dados SQL, você precisará usar a JDBC (Java Database Connectivity), que é a API padrão para conectar e executar consultas em bancos de dados a partir de aplicativos Java.

Baixar o driver JDBC do banco de dados:

Dependendo do banco de dados que você está usando (MySQL, PostgreSQL, SQL Server, etc.), será necessário baixar o driver JDBC correspondente (um arquivo .jar). Este driver permite a comunicação entre sua aplicação Java e o banco de dados SQL.

Configurar o driver no NetBeans:

Adicione o driver JDBC ao seu projeto no NetBeans, seguindo estes passos:

Clique com o botão direito no projeto.

Selecione "Propriedades".

Vá até "Bibliotecas" e clique em "Adicionar JAR/Folder".

Selecione o arquivo .jar do driver JDBC que você baixou.

Estabelecer conexão com o banco de dados

Você precisará fornecer a URL de conexão do banco, o nome de usuário e a senha.

Executar comandos SQL (Consultas/Atualizações)

A partir da conexão, você pode usar objetos Statement ou PreparedStatement para executar comandos SQL (SELECT, INSERT, UPDATE, DELETE).

Exemplo prático no NetBeans:

Passo 1: Preparando o banco de dados

Suponha que você tenha um banco de dados chamado meu\_banco com uma tabela chamada usuarios.



Passo 2: Código Java no NetBeans

Adicione o driver JDBC do MySQL (ex: mysql-connector-java.jar) ao projeto.

Crie a classe Java que vai estabelecer a conexão com o banco de dados e realizar uma consulta:





Explicação do código:

Registrar o Driver

O método Class.forName("com.mysql.cj.jdbc.Driver") carrega o driver JDBC do MySQL.

Estabelecer a conexão

O método DriverManager.getConnection() cria a conexão com o banco de dados usando a URL, nome de usuário e senha.

Executar uma consulta

Usamos um objeto Statement para executar o comando SQL (SELECT id, nome, email FROM usuarios), e o resultado é armazenado em um ResultSet.

Processar os resultados

O while (rs.next()) percorre os resultados da consulta e exibe as informações de cada linha (usuário).

Fechar recursos

O ResultSet, Statement e Connection são fechados para liberar os recursos.

* **COMO FAZER A CRIAÇÃO DE UM AMBIENTE GRÁFICO EM UM APLICATIVO JAVA UTILIZANDO NETBEANS**

Para criar uma interface gráfica em um aplicativo Java, usamos bibliotecas como Swing ou JavaFX. Aqui, vou usar o Swing

Swing é uma biblioteca do Java que fornece componentes gráficos para a criação de interfaces de usuário (botões, janelas, caixas de texto, etc.). Os componentes Swing são usados para construir interfaces gráficas em aplicativos de desktop.

Os principais componentes de Swing são:

JFrame: a janela principal do aplicativo.

JPanel: um contêiner que pode conter outros componentes.

JButton: um botão.

JLabel: um rótulo de texto.

JTextField: um campo de texto.

Passos para criar uma interface gráfica no NetBeans

Criar um novo projeto Java no NetBeans:

No NetBeans, vá até Arquivo > Novo Projeto.

Selecione "Aplicativo Java" e dê um nome ao projeto.

Criar um novo JFrame Form:

Clique com o botão direito na pasta "Pacotes de código-fonte" no seu projeto.

Selecione Novo > JFrame Form. Isso cria uma nova classe com uma interface gráfica.

Adicionar componentes gráficos:

Ao abrir o JFrame Form, você verá o GUI Builder, onde pode arrastar e soltar componentes como botões, campos de texto, etc.

No painel "Paleta", arraste componentes (como JButton, JLabel, JTextField) para o JFrame.

Adicionar funcionalidades aos componentes:

Clique duas vezes em um botão ou outro componente para abrir o código Java associado a ele.

No método gerado, adicione as funcionalidades que deseja, como exibir uma mensagem ao clicar no botão.

Exemplo prático

Passo 1: Criando a janela com Swing no NetBeans

Aqui está um exemplo simples de um aplicativo com uma janela que contém um rótulo, um campo de texto e um botão. Quando o botão é clicado, o texto digitado no campo de texto é exibido em um JLabel.

Código Java:

import javax.swing.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class ExemploGUI extends JFrame {

private JLabel label;

private JTextField textField;

private JButton button;

public ExemploGUI() {

setTitle("Exemplo de Interface Gráfica");

setSize(300, 200);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

setLayout(null);

label = new JLabel("Digite algo:");

label.setBounds(10, 20, 100, 30);

add(label);

textField = new JTextField();

textField.setBounds(120, 20, 150, 30);

add(textField);

button = new JButton("Exibir");

button.setBounds(10, 70, 100, 30);

add(button);

button.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String texto = textField.getText();

JOptionPane.showMessageDialog(null, "Você digitou: " + texto);

}

});

}

public static void main(String[] args) {

ExemploGUI gui = new ExemploGUI();

gui.setVisible(true);

}

}

* **DEFINIR PROGRAMAÇÃO NO CONCEITO MVC (PROGRAMAÇÃO EM CAMADAS)**

MVC (Model-View-Controller) é um padrão de arquitetura de software muito utilizado para organizar e estruturar aplicações em camadas. Ele divide uma aplicação em três componentes principais: Model (Modelo), View (Visão) e Controller (Controlador), separando a lógica de negócios, a interface com o usuário e o controle das interações.

1. Model (Modelo):

Representa os dados e a lógica de negócios da aplicação.

Manipula diretamente os dados, seja acessando um banco de dados ou manipulando arquivos.

O Model notifica as Views sobre qualquer mudança nos dados.

2. View (Visão):

É a interface com o usuário.

Exibe os dados do Model e envia as interações do usuário para o Controller.

A View não tem lógica de negócios, ela apenas exibe informações.

3. Controller (Controlador):

O Controller é o intermediário entre a View e o Model.

Recebe as requisições do usuário feitas na View, processa essas requisições (por exemplo, validando dados) e informa o Model sobre o que precisa ser feito.

Após manipular os dados, o Controller atualiza a View.

Exemplo prático: Aplicação MVC em Java

Aqui está um exemplo básico em Java de como implementar o padrão MVC. A aplicação simula a gestão de um usuário.

Estrutura do exemplo:

Model: Representa um usuário (dados).

View: Exibe o nome do usuário e um botão para atualizar o nome.

Controller: Faz a interação entre a View e o Model, atualizando os dados conforme necessário.

Model:

public class User {

private String name;

public User(String name) {

this.name = name;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

View:

import javax.swing.\*;

import java.awt.event.ActionListener;

public class UserView extends JFrame {

private JTextField nameField = new JTextField(20);

private JButton updateButton = new JButton("Atualizar Nome");

public UserView() {

setTitle("Gestão de Usuário");

setSize(300, 150);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(null);

JLabel label = new JLabel("Nome do usuário:");

label.setBounds(10, 10, 150, 25);

add(label);

nameField.setBounds(120, 10, 150, 25);

add(nameField);

updateButton.setBounds(120, 50, 150, 25);

add(updateButton);

}

public String getUserName() {

return nameField.getText();

}

public void setUserName(String name) {

nameField.setText(name);

}

public void addUpdateListener(ActionListener listener) {

updateButton.addActionListener(listener);

}

}

Crontoller:

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class UserController {

private User model;

private UserView view;

public UserController(User model, UserView view) {

this.model = model;

this.view = view;

this.view.setUserName(model.getName());

this.view.addUpdateListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

model.setName(view.getUserName());

JOptionPane.showMessageDialog(null, "Nome atualizado para: " + model.getName());

}

});

}

}

Classe Principal:

public class Main {

public static void main(String[] args) {

User model = new User("Henrique");

UserView view = new UserView();

UserController controller = new UserController(model, view);

view.setVisible(true);

}

}

Explicação do exemplo:

Model (User): Representa o objeto usuário, que tem um nome e métodos para obter e alterar esse nome.

View (UserView): É a interface gráfica que permite ao usuário ver e atualizar o nome. Ele contém um campo de texto para exibir e modificar o nome do usuário e um botão para submeter a atualização.

Controller (UserController): Faz a ligação entre a interface gráfica (View) e os dados (Model). Quando o botão de atualizar é clicado, o nome do usuário é atualizado no User (Model), e uma mensagem de confirmação é exibida.

* **CONSIDERAÇÕES FINAIS**

O desenvolvimento de aplicativos em Java oferece uma ampla gama de ferramentas e padrões para construção de software robusto e modular. Desde a conexão a bancos de dados SQL até a criação de interfaces gráficas e organização de código com MVC, é possível criar sistemas eficientes e fáceis de manter. O uso de bibliotecas como Swing e APIs como JDBC permite que os desenvolvedores façam uso de boas práticas e padrões amplamente aceitos na indústria.

**REFERÊNCIAS**

[**https://www.devmedia.com.br/padrao-mvc-java-magazine/21995**](https://www.devmedia.com.br/padrao-mvc-java-magazine/21995)

[**https://www.treinaweb.com.br/blog/o-que-e-mvc**](https://www.treinaweb.com.br/blog/o-que-e-mvc)

<https://www.alura.com.br/artigos/como-criar-interface-grafica-swing-java?srsltid=AfmBOooqJ5lLGneEU_NhVPBivk89GTdgX9-lhRDpHfUGu_qbFrc-GXTe>

<https://www.dio.me/articles/conectando-um-banco-de-dados-sql-com-java-um-guia-passo-a-passo>

<https://www.devmedia.com.br/java-crie-uma-conexao-com-banco-de-dados/5698>

<https://coodesh.com/blog/dicionario/o-que-e-crud/>

<https://www.escoladnc.com.br/blog/crud-o-que-e-e-como-funciona/>