







Guia Passo a Passo: Aplicação Swing com SQLite para Cadastro de Pessoas

Este tutorial ensinará a criar uma aplicação **Swing** no **NetBeans 18** para gerenciar uma tabela pessoa no **SQLite**, seguindo a mesma estrutura do código original fornecido.

ATENÇÃO: Em programação Java, **DAO** significa **Data Access Object**. É um **padrão de projeto (design pattern)** que tem como objetivo **isolar a lógica de acesso a dados** (por exemplo, operações com banco de dados) da lógica de negócios da aplicação.

Estrutura do Projeto

 ProjetoPessoas/

- └─  ConexaoSQLite.java → Gerencia a conexão com o banco
- └─  CriarBanco.java → Cria a tabela `pessoa`
- └─  Pessoa.java → Modelo (classe entidade)
- └─  PessoaDAO.java → Operações CRUD no banco
- └─  TelaPrincipal.java → Interface gráfica (Swing)

Passo a Passo para Implementação

1. Criar o Projeto no NetBeans

1. Abra o NetBeans 18
2. **File → New Project → Java with Ant → Java Application**
3. Nomeie como CadastroPessoas, desmarque “Create Main Class” e clique em **Finish**

2. Adicionar o Driver SQLite

1. Baixe o driver JDBC do SQLite: [sqlite-jdbc](#)
2. No NetBeans:
 - Clique com o botão direito em **Libraries**
 - **Add JAR/Folder** → Selecione o arquivo sqlite-jdbc-X.X.X.jar

3. Criar a Classe ConexaoSQLite

```
// ConexaoSQLite.java
import java.sql.Connection;
```

```

import java.sql.DriverManager;
import java.sql.SQLException;

public class ConexaoSQLite {
    public static Connection conectar() {
        Connection conn = null;
        try {
            String url = "jdbc:sqlite:peessoas.db"; // Arquivo do banco
            conn = DriverManager.getConnection(url);
        } catch (SQLException e) {
            System.out.println("Erro na conexão: " + e.getMessage());
        }
        return conn;
    }
}

```

Explicação:

- Cria uma conexão com o banco pessoas.db (será gerado automaticamente).

4. Criar a Classe CriarBanco

```

// CriarBanco.java
import java.sql.Connection;
import java.sql.Statement;

public class CriarBanco {
    public static void criarTabela() {
        String sql = "CREATE TABLE IF NOT EXISTS pessoa ("
            + "id INTEGER PRIMARY KEY AUTOINCREMENT, "
            + "nome TEXT NOT NULL, "
            + "idade INTEGER NOT NULL, "
            + "salario REAL NOT NULL);";

        try (Connection conn = ConexaoSQLite.conectar();
            Statement stmt = conn.createStatement()) {
            stmt.execute(sql);
            System.out.println("Tabela criada com sucesso!");
        } catch (Exception e) {
            System.out.println("Erro ao criar tabela: " + e.getMessage());
        }
    }

    public static void main(String[] args) {
        criarTabela(); // Rode uma vez para criar o banco
    }
}

```

Explicação:

- Executa o SQL para criar a tabela pessoa.

- **Execute esta classe uma vez** para criar o banco.

5. Criar a Classe Pessoa (Modelo)

```
// Pessoa.java
public class Pessoa {
    private int id;
    private String nome;
    private int idade;
    private double salario;

    // Construtor
    public Pessoa(int id, String nome, int idade, double salario) {
        this.id = id;
        this.nome = nome;
        this.idade = idade;
        this.salario = salario;
    }

    // Getters e Setters (gerados automaticamente no NetBeans: Alt+Insert)
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
    public String getNome() { return nome; }
    public void setNome(String nome) { this.nome = nome; }
    public int getIdade() { return idade; }
    public void setIdade(int idade) { this.idade = idade; }
    public double getSalario() { return salario; }
    public void setSalario(double salario) { this.salario = salario; }
}
```

6. Criar a Classe PessoaDAO (CRUD)

```
// PessoaDAO.java
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class PessoaDAO {
    public static void inserir(Pessoa p) {
        String sql = "INSERT INTO pessoa (nome, idade, salario) VALUES (?, ?, ?)";
        try (Connection conn = ConexaoSQLite.conectar();
            PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, p.getNome());
            pstmt.setInt(2, p.getIdade());
            pstmt.setDouble(3, p.getSalario());
            pstmt.executeUpdate();
        } catch (SQLException e) {
            System.out.println("Erro ao inserir: " + e.getMessage());
        }
    }
}
```

```

public static List<Pessoa> listar() {
    List<Pessoa> pessoas = new ArrayList<>();
    String sql = "SELECT * FROM pessoa ORDER BY id";
    try (Connection conn = ConexaoSQLite.conectar();
        PreparedStatement stmt = conn.prepareStatement(sql);
        ResultSet rs = stmt.executeQuery()) {
        while (rs.next()) {
            Pessoa p = new Pessoa(
                rs.getInt("id"),
                rs.getString("nome"),
                rs.getInt("idade"),
                rs.getDouble("salario")
            );
            pessoas.add(p);
        }
    } catch (SQLException e) {
        System.out.println("Erro ao listar: " + e.getMessage());
    }
    return pessoas;
}

public static void atualizar(Pessoa p) {
    String sql = "UPDATE pessoa SET nome = ?, idade = ?, salario = ? WHERE id = ?";
    try (Connection conn = ConexaoSQLite.conectar();
        PreparedStatement stmt = conn.prepareStatement(sql)) {
        stmt.setString(1, p.getNome());
        stmt.setInt(2, p.getIdade());
        stmt.setDouble(3, p.getSalario());
        stmt.setInt(4, p.getId());
        stmt.executeUpdate();
    } catch (SQLException e) {
        System.out.println("Erro ao atualizar: " + e.getMessage());
    }
}

public static void excluir(int id) {
    String sql = "DELETE FROM pessoa WHERE id = ?";
    try (Connection conn = ConexaoSQLite.conectar();
        PreparedStatement stmt = conn.prepareStatement(sql)) {
        stmt.setInt(1, id);
        stmt.executeUpdate();
    } catch (SQLException e) {
        System.out.println("Erro ao excluir: " + e.getMessage());
    }
}
}

```

7. Criar a Interface TelaPrincipal (Swing)

1. Clique direito no projeto → New → JFrame Form

2. Nomeie como TelaPrincipal

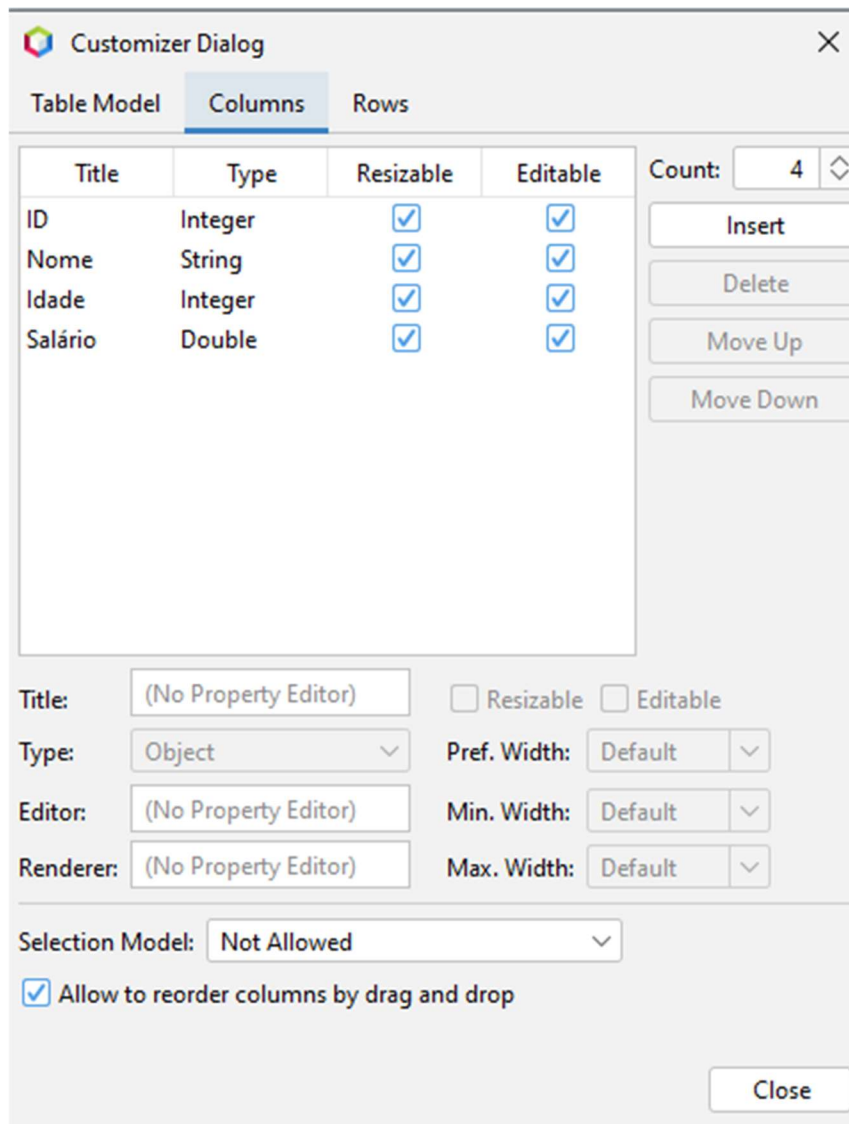
3. Projete a interface com os componentes:

- **3 JTextField** → txtNome, txtIdade, txtSalario
- **1 JTable** → tabelaPessoas
- **3 JButton** → btnInserir, btnAtualizar, btnExcluir
- **3 JLabel** → não precisa nomear variáveis para estes objetos.

The screenshot shows a Java Swing window with a light gray background. At the top, there are three text input fields labeled "Nome:", "Idade:", and "Salário:". Below these fields are three buttons: "Inserir", "Atualizar", and "Excluir". Below the buttons is a JTable with four columns: "ID", "Nome", "Idade", and "Salário". The table is currently empty. The window has a standard Mac OS X-style title bar with a red, yellow, and green button on the left.

OBS.: Configuração do **JTable**

- 1- Clicar com botão direito sobre o objeto e selecionar "Table Contents ...";
- 2- Dentro do "Customizer Dialog" selecionar aba "Columns";
- 3- Depois alterar "Title" e "Type" conforme tela abaixo:



Variáveis necessárias apresentadas no código:

```
private javax.swing.JButton btnAtualizar;
private javax.swing.JButton btnExcluir;
private javax.swing.JButton btnInserir;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTable tabelaPessoas;
private javax.swing.JTextField txtIdade;
private javax.swing.JTextField txtNome;
private javax.swing.JTextField txtSalario;
```

Código da Tela Principal (Preencher somente com o código em cor azul)

```
// TelaPrincipal.java
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.util.List;
```

```
public class TelaPrincipal extends javax.swing.JFrame {
```

```
public TelaPrincipal() {
```

```
    initComponents();
```

```
    atualizarTabela();
```

```
    // Adiciona o listener de seleção à tabela
```

```
    tabelaPessoas.getSelectionModel().addListSelectionListener(e -> {
```

```
        if (!e.getValueIsAdjusting()) { // Evita duplicação de eventos
```

```
            carregarDadosSelecionados();
```

```
        }
```

```
    });
```

```
}
```

```
//Adicione este método para preencher os campos quando uma linha é selecionada:
```

```
private void carregarDadosSelecionados() {
```

```
    int linhaSelecionada = tabelaPessoas.getSelectedRow();
```

```
    if (linhaSelecionada >= 0) { // Se uma linha foi selecionada
```

```
        // Obtém os dados da linha selecionada
```

```
        int id = (int) tabelaPessoas.getValueAt(linhaSelecionada, 0);
```

```
        String nome = (String) tabelaPessoas.getValueAt(linhaSelecionada, 1);
```

```
        int idade = (int) tabelaPessoas.getValueAt(linhaSelecionada, 2);
```

```
        double salario = (double) tabelaPessoas.getValueAt(linhaSelecionada, 3);
```

```
        // Preenche os campos
```

```
        txtNome.setText(nome);
```

```
        txtIdade.setText(String.valueOf(idade));
```

```
        txtSalario.setText(String.valueOf(salario));
```

```
    }
```

```
}
```

```
private void atualizarTabela() {
```

```
    DefaultTableModel model = (DefaultTableModel) tabelaPessoas.getModel();
```

```
    model.setRowCount(0); // Limpa a tabela
```

```
    List<Pessoa> pessoas = PessoaDAO.listar();
```

```
    for (Pessoa p : pessoas) {
```

```
        model.addRow(new Object[]{
```

```
            p.getId(),
```

```
            p.getNome(),
```

```
            p.getIdade(),
```

```
            p.getSalario()
```

```
        });
```

```
    }
```

```
}
```

```
private void limparCampos() {  
    txtNome.setText("");  
    txtIdade.setText("");  
    txtSalario.setText("");  
}
```

//Evento actionPerformed do botão "Inserir"

```
private void btnInserirActionPerformed(java.awt.event.ActionEvent evt) {  
    String nome = txtNome.getText();  
    int idade = Integer.parseInt(txtIdade.getText());  
    double salario = Double.parseDouble(txtSalario.getText());  
  
    PessoaDAO.inserir(new Pessoa(0, nome, idade, salario));  
    atualizarTabela();  
    limparCampos();  
}
```

//Evento actionPerformed do botão "Atualizar"

```
private void btnAtualizarActionPerformed(java.awt.event.ActionEvent evt) {  
    int linhaSelecionada = tabelaPessoas.getSelectedRow();  
  
    if (linhaSelecionada == -1) {  
        JOptionPane.showMessageDialog(this, "Selecione uma pessoa para atualizar!");  
        return;  
    }  
  
    try {  
        int id = (int) tabelaPessoas.getValueAt(linhaSelecionada, 0);  
        String nome = txtNome.getText();  
        int idade = Integer.parseInt(txtIdade.getText());  
        double salario = Double.parseDouble(txtSalario.getText());  
  
        PessoaDAO.atualizar(new Pessoa(id, nome, idade, salario));  
        atualizarTabela();  
        limparCampos();  
    } catch (NumberFormatException ex) {  
        JOptionPane.showMessageDialog(this, "Idade e salário devem ser números válidos!");  
    }  
}
```

//Evento actionPerformed do botão "Excluir"

```
private void btnExcluirActionPerformed(java.awt.event.ActionEvent evt) {  
    int linha = tabelaPessoas.getSelectedRow();  
    if (linha >= 0) {  
        int id = (int) tabelaPessoas.getValueAt(linha, 0);  
        PessoaDAO.excluir(id);  
        atualizarTabela();  
    }  
}
```



```
// Método main (gerado automaticamente)
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(() -> {
        new TelaPrincipal().setVisible(true);
    });
}

// Variables declaration - do not modify
private javax.swing.JButton btnAtualizar;
private javax.swing.JButton btnExcluir;
private javax.swing.JButton btnInserir;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTable tabelaPessoas;
private javax.swing.JTextField txtIdade;
private javax.swing.JTextField txtNome;
private javax.swing.JTextField txtSalario;
// End of variables declaration
```

◆ Como Testar a Aplicação

1. Execute CriarBanco.java **uma vez** para criar o banco.
 2. Execute TelaPrincipal.java para abrir a interface.
 3. Teste:
 - **Inserir:** Preencha os campos e clique em "Inserir".
 - **Atualizar:** Selecione uma pessoa, edite os campos e clique em "Atualizar".
 - **Excluir:** Selecione uma pessoa e clique em "Excluir".
-

Conclusão

Agora você tem um **CRUD completo** em **Java Swing + SQLite**.

Implementando Seleção Automática na Tabela Swing

Para carregar automaticamente os dados da linha selecionada da JTable nos JTextFields, você precisa adicionar um ListSelectionListener à tabela. Veja como fazer isso passo a passo:

◆ Como Funciona?

1. **Quando o usuário seleciona uma linha na tabela**, o ListSelectionListener é acionado.

2. O método `carregarDadosSelecionados()` pega os valores da linha selecionada e os coloca nos `JTextFields`.
 3. **Se o usuário editar os campos e clicar em "Atualizar"**, a linha selecionada será modificada no banco de dados.
-