

EducaCiência FastCode

Fala Galera,

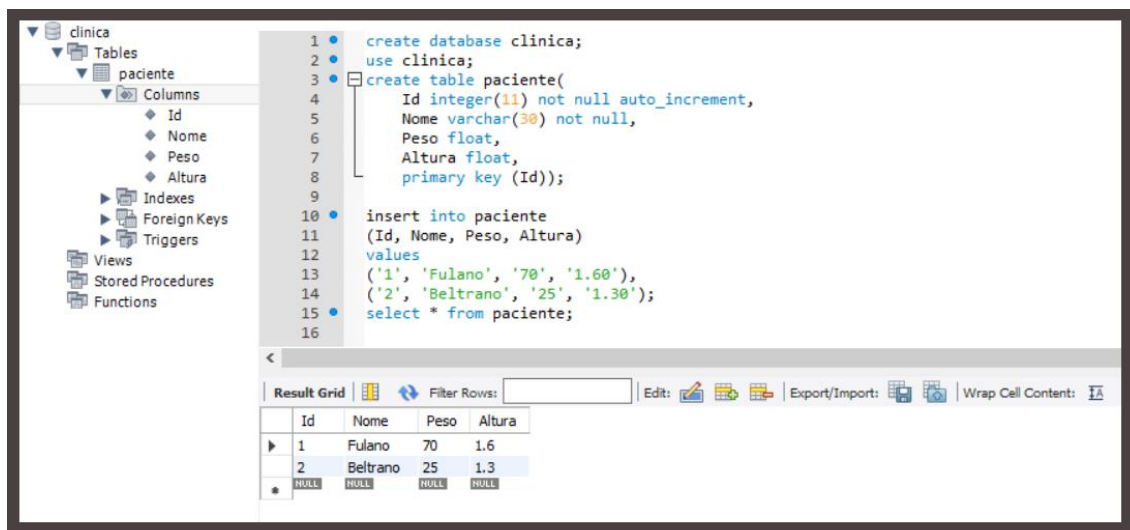
- Artigo: 25/2020 Data: Junho/2020
- Público Alvo: Desenvolvedores – Iniciantes
- Tecnologia: Java
- Tema: CRUD – criando Método Deletar
- Link: <https://github.com/perucello/DevFP>

Criaremos uma série de artigos para explanarmos o CRUD em um projeto Desktop. Neste artigo 25/2020 daremos continuidade no nosso propósito, criando o MÉTODO DELETAR.

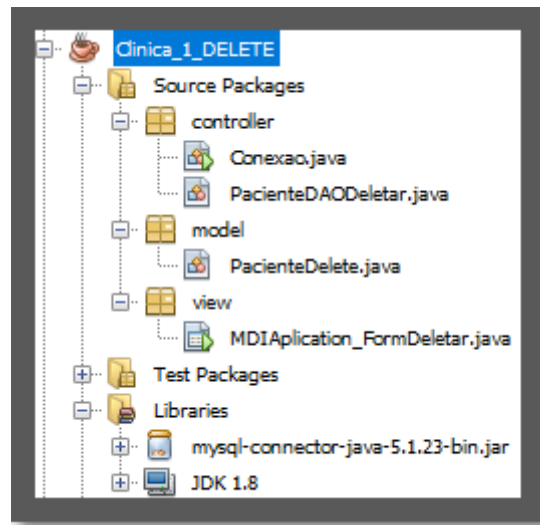
Para este ambiente , já temos nosso Banco de Dados criado e chamado “clinica”.

Nosso ambiente consiste em:

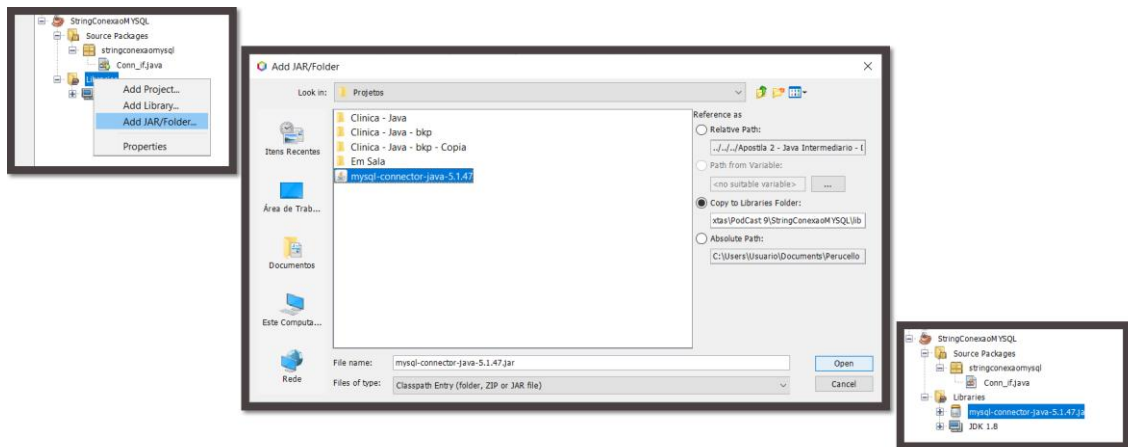
⇒ Banco de Dados MySql



⇒ Padrão MVC de arquitetura



Vale lembrar que temos que adicionar nossa biblioteca “jar” de Conexão. Vamos lá.



CRUD

Deletar - 01

Vou tentar detalhar o máximo possível os passos para que não fique nenhuma dúvida , ou que possa ser replicado sem maiores problemas o nosso Projeto.

Pacote Controller – neste pacote, teremos duas classes, a nossa Classe Conexão e nossa Classe PacienteDAOInserir.

a) Classe Conexão – nesta classe, teremos nossa String de Conexão com o Banco de Dados

```
1 package controller;
2 import java.sql.Connection;
3 import java.sql.DriverManager;
4 import java.sql.SQLException;
5 import javax.swing.JOptionPane;
6 //Classe Conexão
7 public class Conexao {
8     //Detalhes da Conexão
9     private static final String DATABASE="clinica"; // Nome do Banco de Dados
10    private static final String HOST="jdbc:mysql://localhost:3306/clinica"; //Host
11    private static final String DRIVER="com.mysql.jdbc.Driver"; //Driver
12    private static final String URL="jdbc:mysql://localhost:3306/clinica?useTimezone=true&serverTimezone=UTC&useSSL=false"; //Para tirar erro de SSL em alguns casos em que tem varios Bancos e Certificados
13    private static final String USER="root"; //Nome do usuario de acesso ao Banco de Dados
14    private static final String PWD=""; // senha definida no Banco de Dados
15
16    //Método conectar
17    public static Connection Conectar(){
18        try{
19            Class.forName(DRIVER);
20            return DriverManager.getConnection(URL, USER, PWD);
21        }
22        catch (ClassNotFoundException | SQLException e) {
23            System.out.println("ERRO: " + e.getMessage());
24            JOptionPane.showMessageDialog(null, "Falha de comunicação com BD", "ERRO", JOptionPane.WARNING_MESSAGE);
25            return null;
26        }
27    }
28
29    //Método Desconectar
30    public static void Desconectar (Connection con){
31        try{
32            if (con != null){
33                con.close();
34            }
35        }
36        catch (SQLException e){
37            System.out.println("ERRO: " + e.getMessage());
38        }
39    }
40
41    //PARA TESTAR VIA SCRIPT
42    //Método Main
43    public static void main(String[] args){
44        if (Conectar() != null){
45            System.out.println("Conexão realizada com sucesso!");
46        }
47    }
48 }
```

- b) Classe PacienteDAODeletar – nesta classe, adicionaremos as Strings Sql que será executada junto ao nosso Banco de Dados, vale ressaltar que temos alguns passos pré definidos no nosso código, onde estou utilizando do recurso *//comentário* para explicar melhor.

Abordaremos neste Método Deletar também o Listar, para que tenha um pouco mais de sentido qual o dado que realmente iremos deletar.

Metodo Deletar

```

1  package controller;
2
3  import java.awt.HeadlessException;
4  import java.sql.Connection;
5  import java.sql.PreparedStatement;
6  import java.sql.ResultSet;
7  import java.sql.SQLException;
8  import java.sql.Statement;
9  import java.util.ArrayList;
10 import java.util.List;
11 import javax.swing.JOptionPane;
12 import model.PacienteDelete;
13
14 public class PacienteDAODeletar {
15
16     //Abrindo nossa conexão com Banco de Dados
17     private final Connection con;
18     private PreparedStatement cmd;
19
20     public PacienteDAODeletar(){
21         this.con = Conexao.Conectar();
22     }
23
24     //Criando método Deletar
25     public int deletar(PacienteDelete p){
26         try{
27             String sql = "delete from paciente where id = ?";
28
29             cmd = con.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS);
30             cmd.setInt(1, p.getId());
31             cmd.execute();
32             JOptionPane.showMessageDialog(null, "Excluido com sucesso");
33         }
34         catch (HeadlessException | SQLException e){
35             System.out.println("ERRO: " + e.getMessage());
36             return -1;
37         }
38         finally{
39             Conexao.Desconectar(con);
40         }
41         return 0;
42     }
43
44     //Criando método Listar => trazemos ja do Projeto Listar já criado anteriormente
45     //Listar
46     public List<PacienteDelete> listar(){...28 lines }
47
48 }

```

Metodo Listar

```

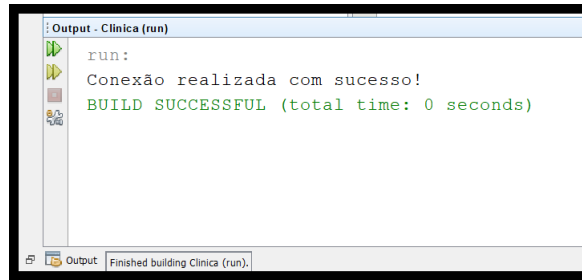
14 public class PacienteDAODeletar {
15
16     //Abrindo nossa conexão com Banco de Dados
17     private final Connection con;
18     private PreparedStatement cmd;
19
20     public PacienteDAODeletar() {
21         this.con = Conexao.Conectar();
22     }
23
24     //Criando método Deletar
25     public int deletar(PacienteDelete p) {...18 lines }
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43     //Criando método Listar => trazemos ja do Projeto Listar já criado anteriormente
44     //Listar
45     public List<PacienteDelete> listar() {
46         try {
47             String sql = "select * from paciente order by id";
48             cmd = con.prepareStatement(sql);
49             ResultSet rs = cmd.executeQuery();
50
51             List<PacienteDelete> lista = new ArrayList<>();
52             while(rs.next()) {
53
54                 PacienteDelete p = new PacienteDelete();
55                 p.setId(rs.getInt("id"));
56                 p.setNome(rs.getString("nome"));
57                 p.setPeso(rs.getFloat("peso"));
58                 p.setAltura(rs.getFloat("altura"));
59
60                 lista.add(p);
61             }
62             return lista;
63         }
64         catch (SQLException e) {
65             System.out.println("ERRO: " + e.getMessage());
66             return null;
67         }
68         finally {
69             Conexao.Desconectar(con);
70         }
71     }
72 }
73
74
75

```

Com isso, temos nosso Pacote Controller já pronto, temos nossa String de Conexão com o Banco de Dados OK e temos nossa Classe que será responsável por deter a manipulação com o Banco de Dados Ok.

Vamos apenas testar nossa conexão com o Banco de Dados.

- Subir WampServer
- Clicar com Botão na classe Conexão e pedir para executar (run)



Pacote Model – nesta classe, teremos nossa Regra de Negócio, porém, neste Projeto em questão, vamos ter nossos Métodos Getters e Setters – lembrando que no Netbeans, temos um recurso para gerar estes Métodos automáticos, onde estamos informando o momento em nosso Script.

```

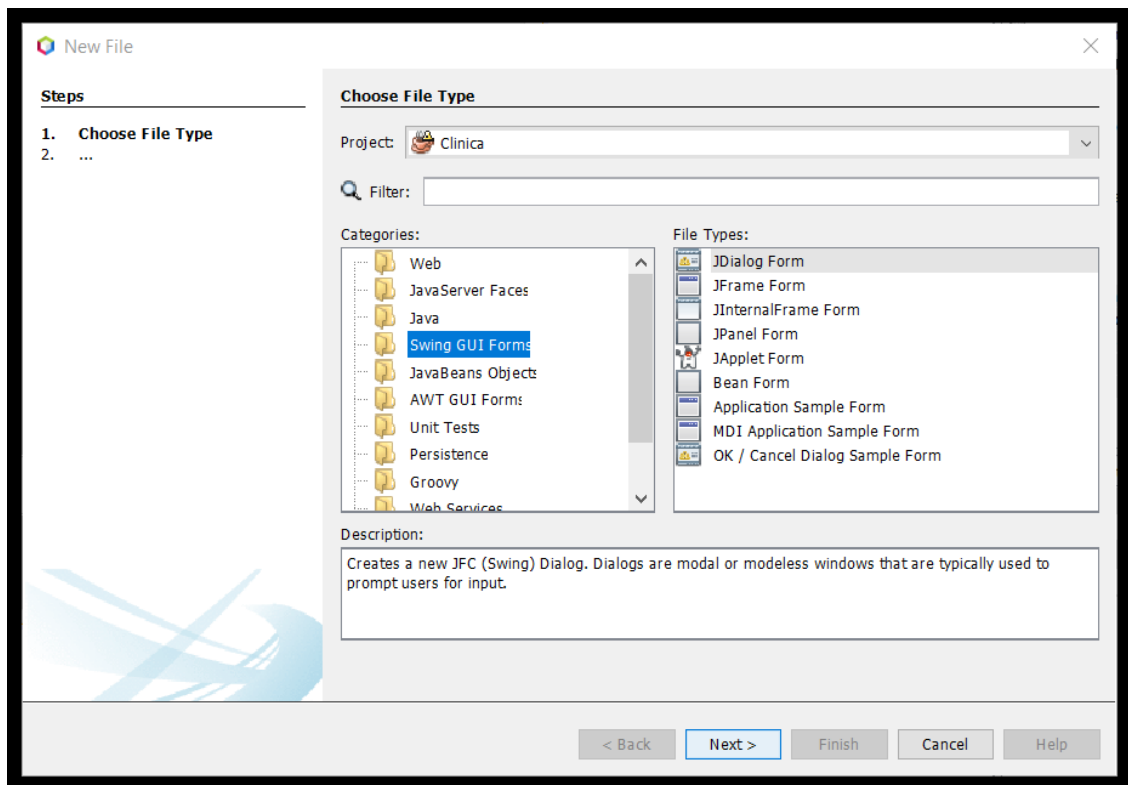
3      public class PacienteDelete {
4          private int id;
5          private String nome;
6          private float peso;
7          private float altura;
8
9          // InsertCode - Contrutor -> NAO SELECIONAR ATRIBUTOS
10         public PacienteDelete() {
11             }
12         // InsertCode - Contrutor selecionando todos os atributos
13         public PacienteDelete(int id, String nome, float peso, float altura) {
14             this.id = id;
15             this.nome = nome;
16             this.peso = peso;
17             this.altura = altura;
18         }
19         // InsertCode - Mtodos Getters/Setters
20         public int getId() {
21             return id;
22         }
23
24         public void setId(int id) {
25             this.id = id;
26         }
27
28         public String getNome() {
29             return nome;
30         }
31
32         public void setNome(String nome) {
33             this.nome = nome;
34         }
35
36         public float getPeso() {
37             return peso;
38         }
39
40         public void setPeso(float peso) {
41             this.peso = peso;
42         }
43
44         public float getAltura() {
45             return altura;
46         }
47
48         public void setAltura(float altura) {
49             this.altura = altura;
50         }
51         // Insert Code Assinatura Override To String ( referenciar Nome )
52         @Override
53         public String toString() {
54             //return super.toString(); //To change body of generated methods, choose Tools | Templates.
55             return nome;
56         }
57     }

```

Pacote View – Nesse momento, após criarmos as classes nos Pacotes Controller e Model vamos criar a iteração do Projeto com o Usuario, ou seja, a Tela em que o usuário irá ver e manipular os dados para que seja inserido no Banco de Dados as informações como é nossa proposta neste artigo – CRUD Método Deletar.

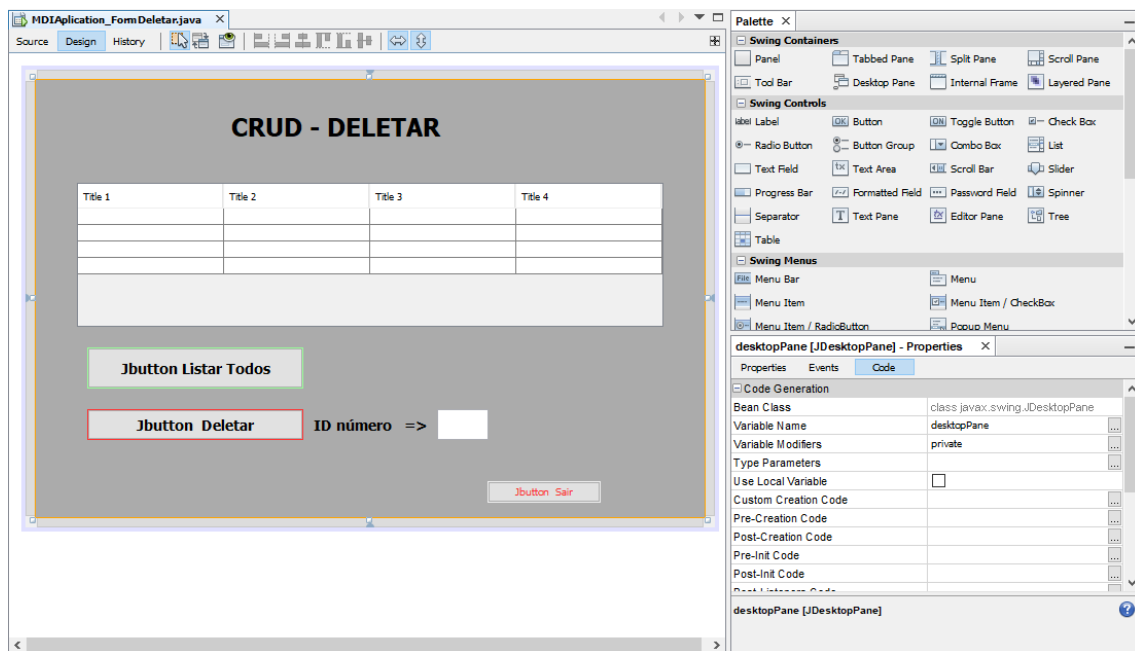
Não é minha intenção deixar a Tela com o visual bem apresentado, mas, sim explorarmos os códigos do lado do BackEnd, sendo assim, vamos utilizar de um artefato visual que o próprio netbeans nos fornece.

Vamos clicar com Botão direito no pacote View irmos em **New/Other** e selecionar **Swing Gui Forms**, nesta categoria vamos selecionar **MDI Application Sampe Form**.



Vamos nomear nosso Form criado MDI e vamos manipulá-lo agora.
Visto que temos nossa tabela no Banco de Dados já criada, vamos definir a tela do nosso Form MDI e criarmos a interface.

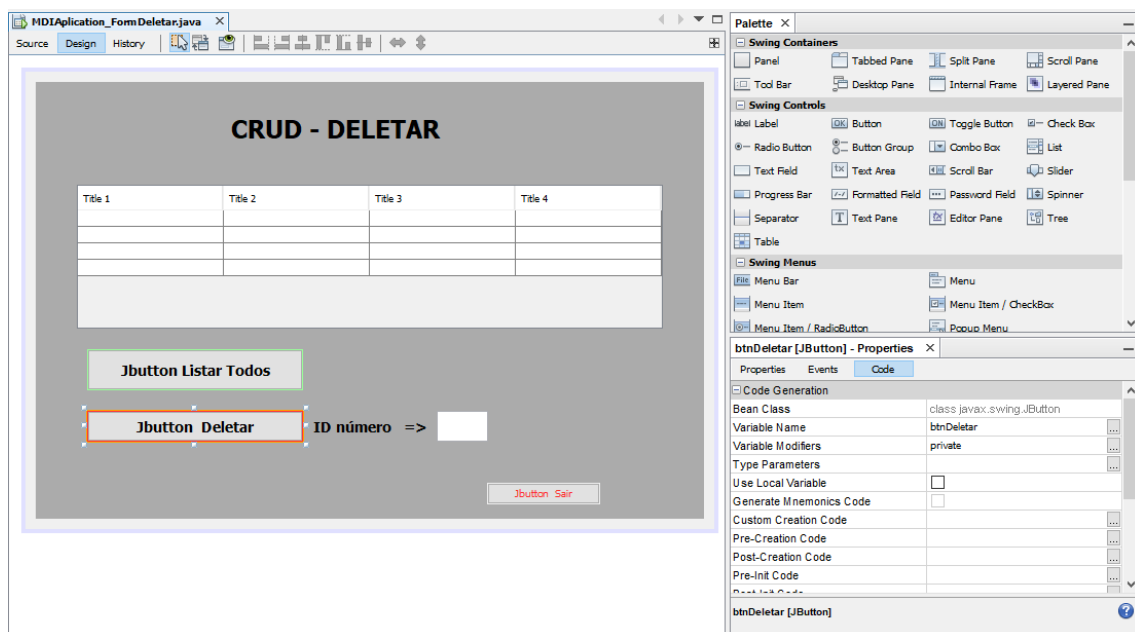
Para maior familiaridade, vou pôr a tela criada com os recursos do Palette utilizado para fins didáticos.



Com estes passos criados, precisamos nomear nossas variáveis, neste momento, basta você clicar no Palette inserido, ir em Propriedades, CODE e nomear nossa variável.

Para este artigo, estou nomeando as variáveis para serem didáticas na exemplificação.

Faremos este processo para todos os campos.



O netbeans, se formos na seção de códigos (source) , veremos que o Netbeans nos fornece todas estas informações que manipulamos.

```
// Variables declaration - do not modify
private javax.swing.JButton btnDeletar;
private javax.swing.JButton btnListarTodos;
private javax.swing.JButton btnSair;
private javax.swing.JDesktopPane desktopPane;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTable jTableResultado;
private javax.swing.JTextField txt_id_Paciente;
// End of variables declaration
```

Agora com nosso Layout da tela pronto, nossas variáveis já nomeadas , vamos aos códigos. Na Opção Source, iremos manipular os códigos, estarei aqui colocando um Passo a Passo para que acompanhem a linha de raciocínio.

Vamos até o final do nosso código, após as declarações das variáveis, vamos identificar a última Chave” e antes dela, vamos incluir os seguintes passos:

```

220 //Passo 1
221 private void configurarFormulario(){
222     this.setTitle("Java Intermediário - Escola Evolua Sumaré");
223     this.setResizable(false);
224     this.setLocationRelativeTo(null);
225     estadoControle(true); // add após criar estado controle
226     preencherTabela(new PacienteDAODeletar().listar()); //para iniciar mostrando os dados da tabela
227     configurarTabela(); //para iniciar sem mostrar dados da tabela
228 }
229 //passo 2
230 private void estadoControle(boolean e){
231     btnListarTodos.setEnabled( e);
232     btnDeletar.setEnabled( e);
233 }
234 //Passo 3
235 private void configurarTabela(){
236     DefaultTableModel m = new DefaultTableModel(){
237
238         @Override
239         public boolean isCellEditable(int row, int column){
240             return false;
241         }
242     };
243
244     m.addColumn("Id do Paciente");
245     m.addColumn("Nome do Paciente");
246     m.addColumn("Peso do Paciente");
247     m.addColumn("Altura do Paciente");
248
249     jTableResultado.setModel(m);
250 }
251 //passo 4
252 private void preencherTabela(List<PacienteDelete> lista){
253     if(lista.size()>0){
254         configurarTabela();
255         DefaultTableModel m = (DefaultTableModel)jTableResultado.getModel();
256
257         //for(PacienteSelect p : lista){
258         lista.forEach((p) -> {
259             m.addRow(new Object[]{
260                 p.getId(),
261                 p.getNome(),
262                 p.getPeso(),
263                 p.getAltura()
264             });
265         });
266     };
267     jTableResultado.setModel(m);
268 }
269 }
270 }
271

```

Agora que criamos nossos métodos, vamos aproveitar, e instanciar o método **configurarformulario** e **EstadoControle** para iniciar junto à aplicação.

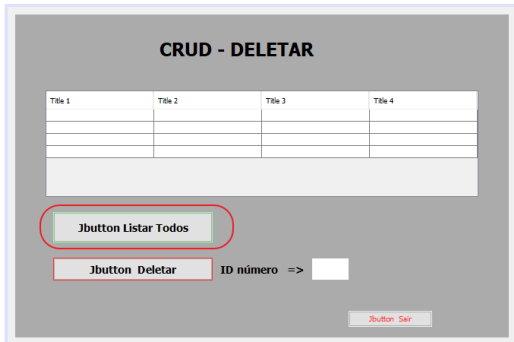
```

10 public MDIAplicacao_FormDeletar() {
11     initComponents();
12     //passo 5
13     configurarFormulario();
14     estadoControle(true);
15 }

```

Feito isso, vamos manipular ou melhor, colocar nossos códigos em nossos botões.

Volte no Design do Projeto e dê um duplo click BOTAO Jbutton Listar Todos

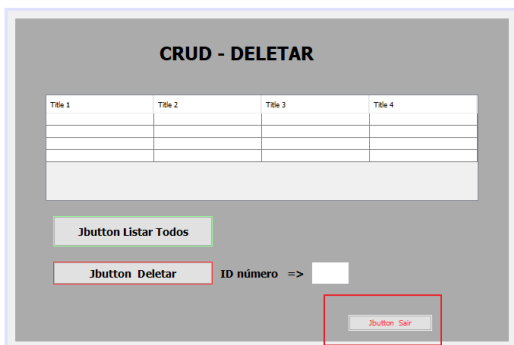


```

129 //passo 5 MONTAR VISUALIZAÇÃO - AÇÃO BOTAO LISTAR
130 private void btnListarTodosActionPerformed(java.awt.event.ActionEvent evt) {
131     preencherTabela(new PacienteDAODeletar().listar());
132 }
133
        
```

Volte no Design do Projeto e dê um duplo click BOTAO Jbutton Sair

Aqui, vamos dar um “dispose” que além de fechar nossa tela, tem como proposito finalizar o processo Java.



```

134 //passo 5 MONTAR VISUALIZAÇÃO - AÇÃO BOTAO SAIR
135 private void btnSairActionPerformed(java.awt.event.ActionEvent evt) {
136     dispose();
137 }
138
        
```

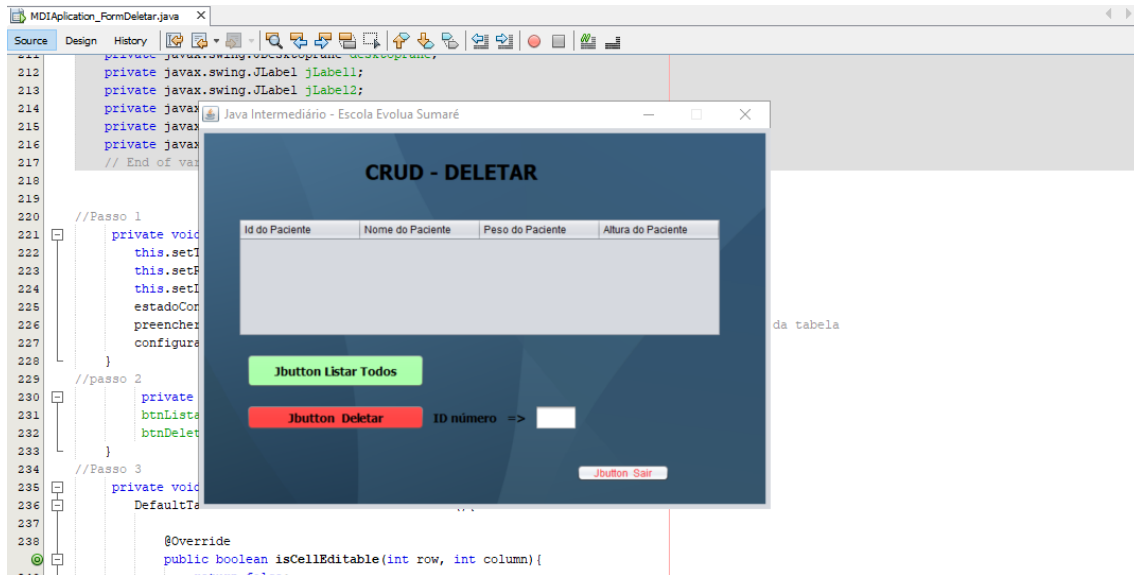
Volte no Design do Projeto e dê um duplo click BOTAO Jbutton Deletar



```

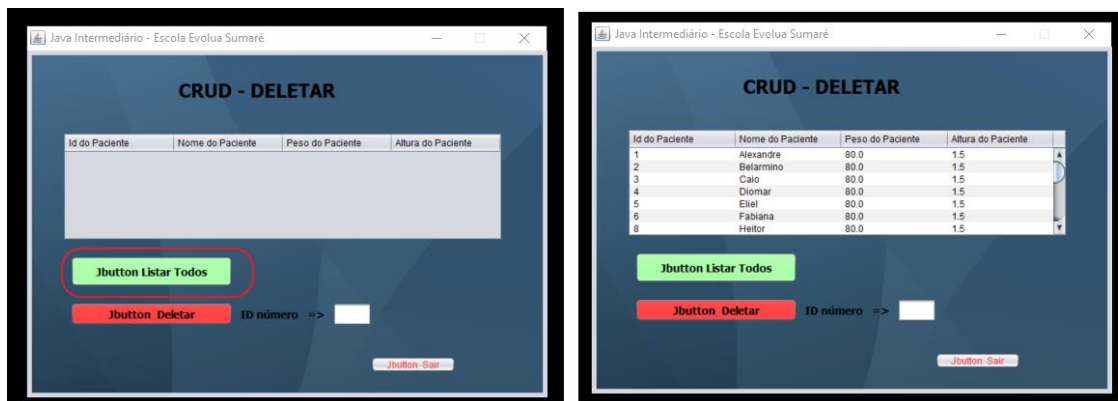
141 //passo 5 MONTAR VISUALIZAÇÃO - AÇÃO BOTAO DELETAR
142 private void btnDeletarActionPerformed(java.awt.event.ActionEvent evt) {
143     PacienteDelete p = new PacienteDelete();
144     p.setid(Integer.parseInt(txt_id_Paciente.getText()));
145     PacienteDAODeletar d = new PacienteDAODeletar();
146     d.deletar(p);
147     preencherTabela(new PacienteDAODeletar().listar());
148     estadoControla(true);
149 }
150
151 private void jTableResultadoMouseClicked(java.awt.event.MouseEvent evt) {
152     int linha = jTableResultado.getSelectedRow();
153     if (linha >= 0) {
154         txt_id_Paciente.setText(jTableResultado.getValueAt(linha, 0).toString());
155     }
156 }
157 //OBSERVANTE
158 //Para montar todos os dados, basta criar os list e nomear as variáveis - entre aspas que coloquei são sugestões de nomes de variáveis
159 txtHOMEPAENTE.setText(jTableResultado.getValueAt(linha, 1).toString());
160 txtPESSOAPACIENTE.setText(jTableResultado.getValueAt(linha, 2).toString());
161 txtALURAPACIENTE.setText(jTableResultado.getValueAt(linha, 3).toString());
162
163 }
164
165
        
```

Agora, que finalizamos nosso projeto, chega o momento mais interessante, vamos rodar. Vamos executar nosso Form – clique com botão direito sobre o MDI do pacote VIEW e vamos selecionar “run”.



Notamos que os dados retornaram normalmente, agora apenas para manipular, vamos:

⇒ Clicar no Listar Todos



⇒ Clicar no Botão Deletar – antes, vamos escolher um ID a ser apagado e informar





⇒ Clicar no Botão Sair

Nota-se que os códigos funcionaram corretamente, no entanto, saliento que este artigo 25/2020 terá uma continuidade no artigo 26/2020 onde mostraremos uma outra maneira de realizar a mesma atividade e estes artigos em questão, são baseados no curso de Java que ministro na Escola Evolua – Ensino Profissionalizante e como proposito de ajuda à comunidade, estamos trazendo parte da didática em forma de artigo comunitário e assim podemos contribuir com a comunidade Tecnológica como um todo.

Agradeço imensamente a Diretoria da Escola Evolua de Sumaré.

Os códigos estarão disponíveis no Git.

Até mais !

Espero ter ajudado !

