

EducaCiência FastCode

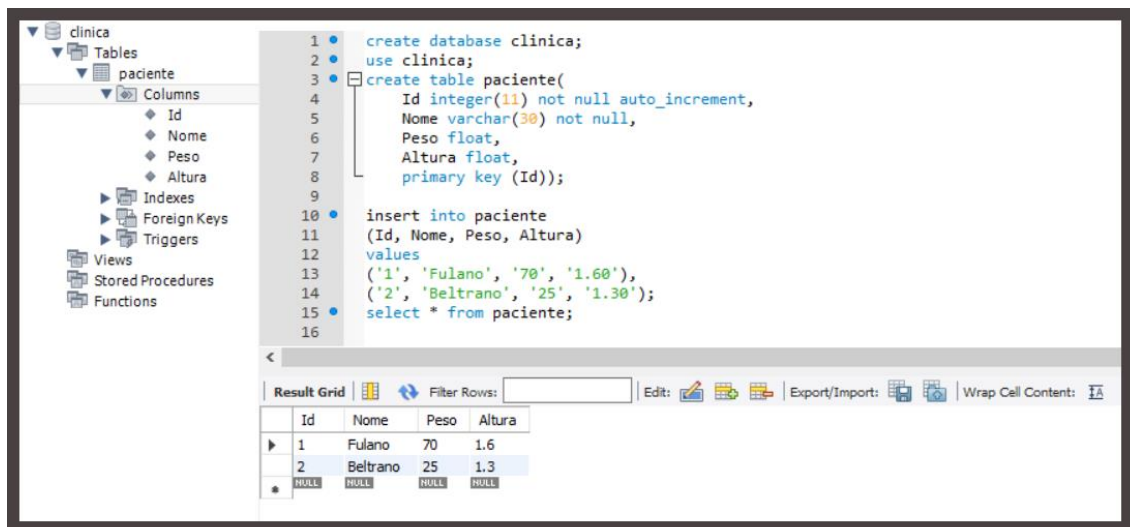
Fala Galera,

- Artigo: 23/2020 Data: Junho/2020
- Público Alvo: Desenvolvedores – Iniciantes
- Tecnologia: Java
- Tema: CRUD – criando Método Listar Todos
- Link: <https://github.com/perucello/DevFP>

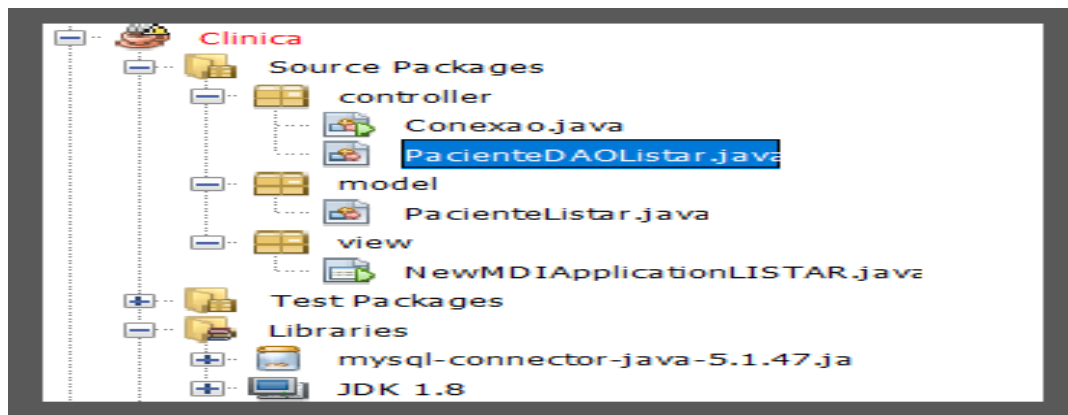
Criaremos uma série de artigos para explanarmos o CRUD em um projeto Desktop. Neste artigo 22/2020 daremos continuidade no nosso proposito, criando o MÉTODO LISTAR, lembrando que o artigo 20/2020 trouxemos o método Inserir.

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

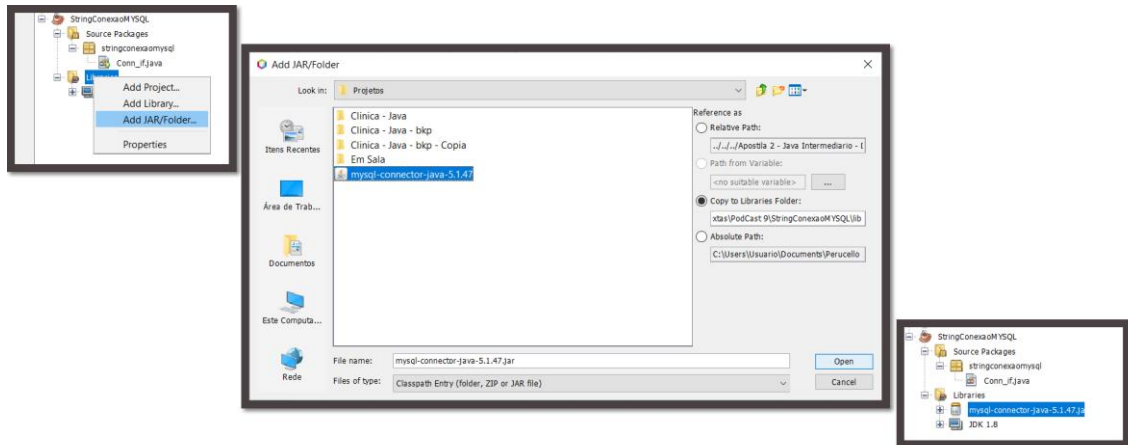
⇒ Banco de Dados MySql



⇒ Padrão MVC de arquitetura



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



CRUD - Listar

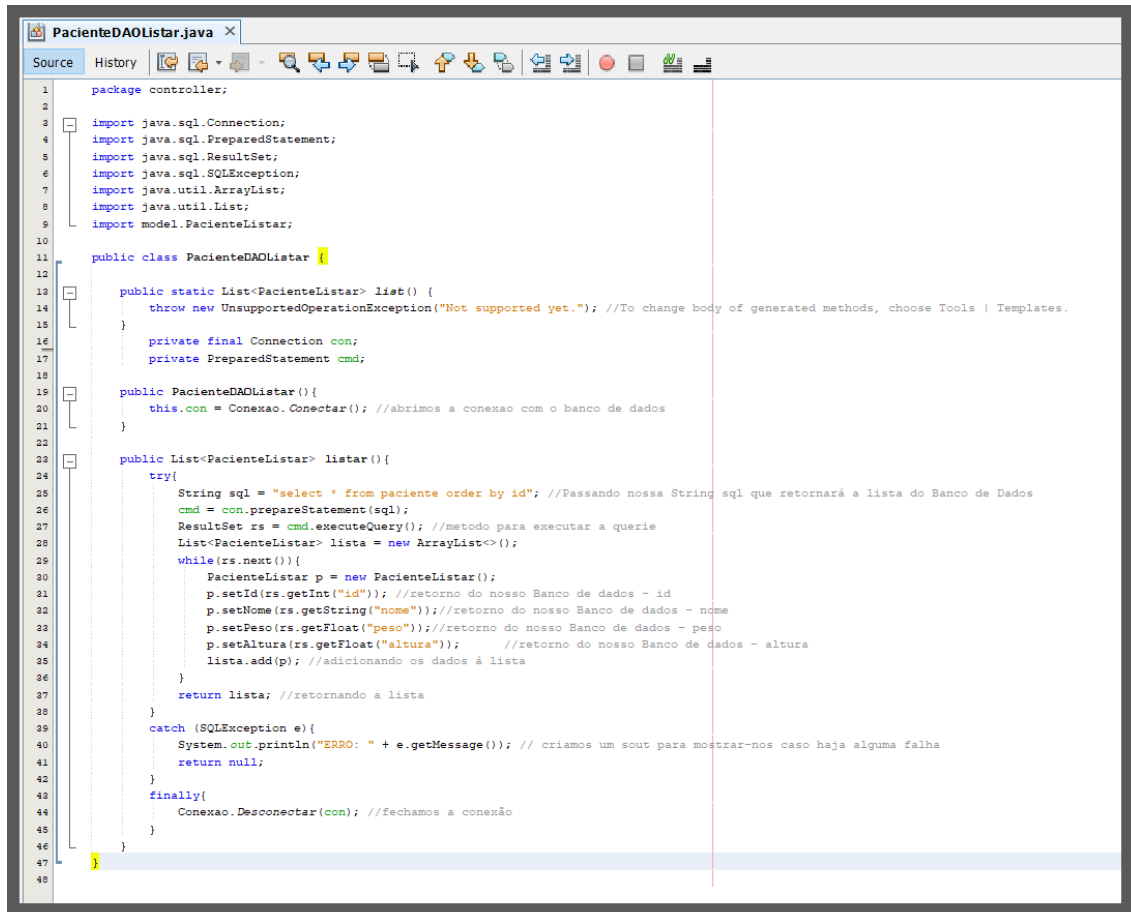
Vou tentar detalhar o máximo possível os passos para que não fique nenhuma dúvida , ou que possa ser replicado sem maiores problemas o nosso Projeto.

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 Conexão {
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    //Metodo 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    //Metodo 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    //Metodo 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 PacienteDAOListar – 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.



```

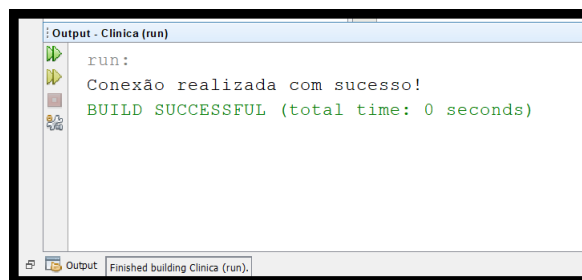
1 package controller;
2
3 import java.sql.Connection;
4 import java.sql.PreparedStatement;
5 import java.sql.ResultSet;
6 import java.sql.SQLException;
7 import java.util.ArrayList;
8 import java.util.List;
9 import model.PacienteListar;
10
11 public class PacienteDAOListar {
12
13     public static List<PacienteListar> list() {
14         throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
15     }
16     private final Connection con;
17     private PreparedStatement cmd;
18
19     public PacienteDAOListar() {
20         this.con = Conexao.Conectar(); //abrimos a conexao com o banco de dados
21     }
22
23     public List<PacienteListar> listar() {
24         try {
25             String sql = "select * from paciente order by id"; //Passando nossa String sql que retornará a lista do Banco de Dados
26             cmd = con.prepareStatement(sql);
27             ResultSet rs = cmd.executeQuery(); //metodo para executar a querie
28             List<PacienteListar> lista = new ArrayList<>();
29             while(rs.next()) {
30                 PacienteListar p = new PacienteListar();
31                 p.setId(rs.getInt("id")); //retorno do nosso Banco de dados - id
32                 p.setNome(rs.getString("nome")); //retorno do nosso Banco de dados - nome
33                 p.setPeso(rs.getFloat("peso")); //retorno do nosso Banco de dados - peso
34                 p.setAltura(rs.getFloat("altura")); //retorno do nosso Banco de dados - altura
35                 lista.add(p); //adicionando os dados a lista
36             }
37             return lista; //retornando a lista
38         } catch (SQLException e) {
39             System.out.println("ERRO: " + e.getMessage()); // criamos um sout para mostrar-nos caso haja alguma falha
40             return null;
41         } finally {
42             Conexao.Desconectar(con); //fechamos a conexão
43         }
44     }
45 }

```

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)

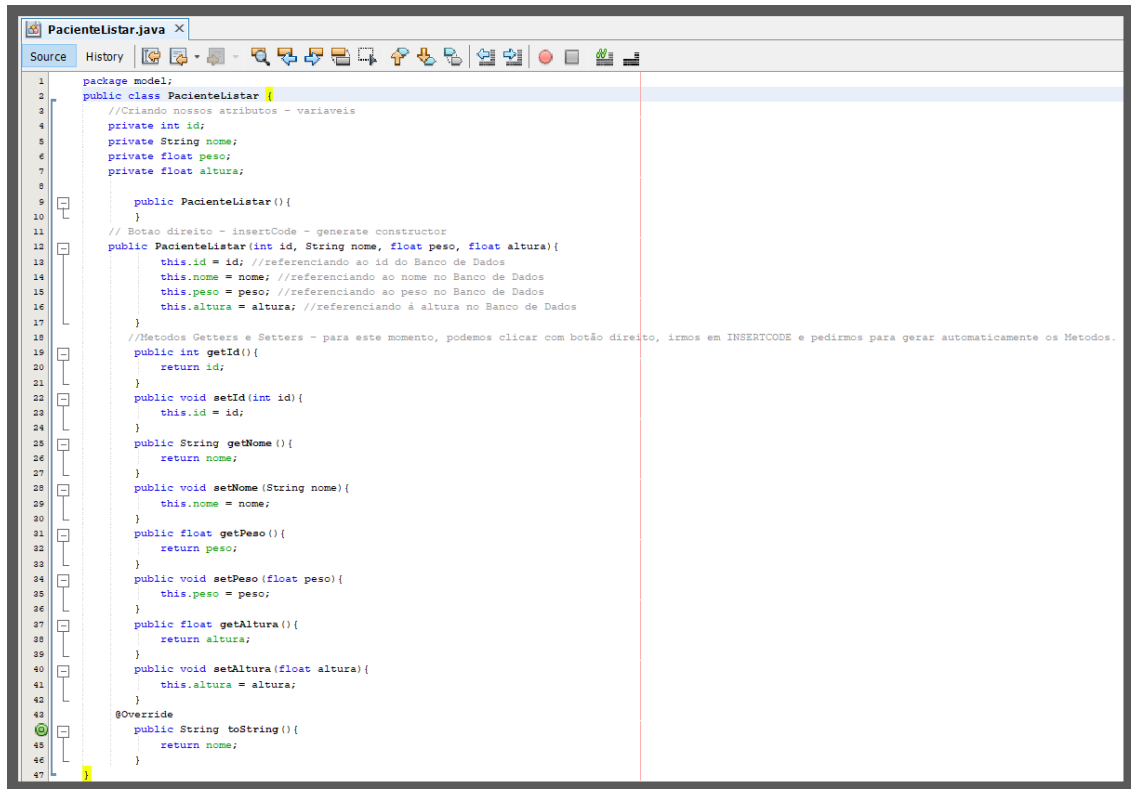


```

Output - Clínica (run)
run:
Conexão realizada com sucesso!
BUILD SUCCESSFUL (total time: 0 seconds)

```

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 automaticos, onde estamos informando o momento em nosso Script.

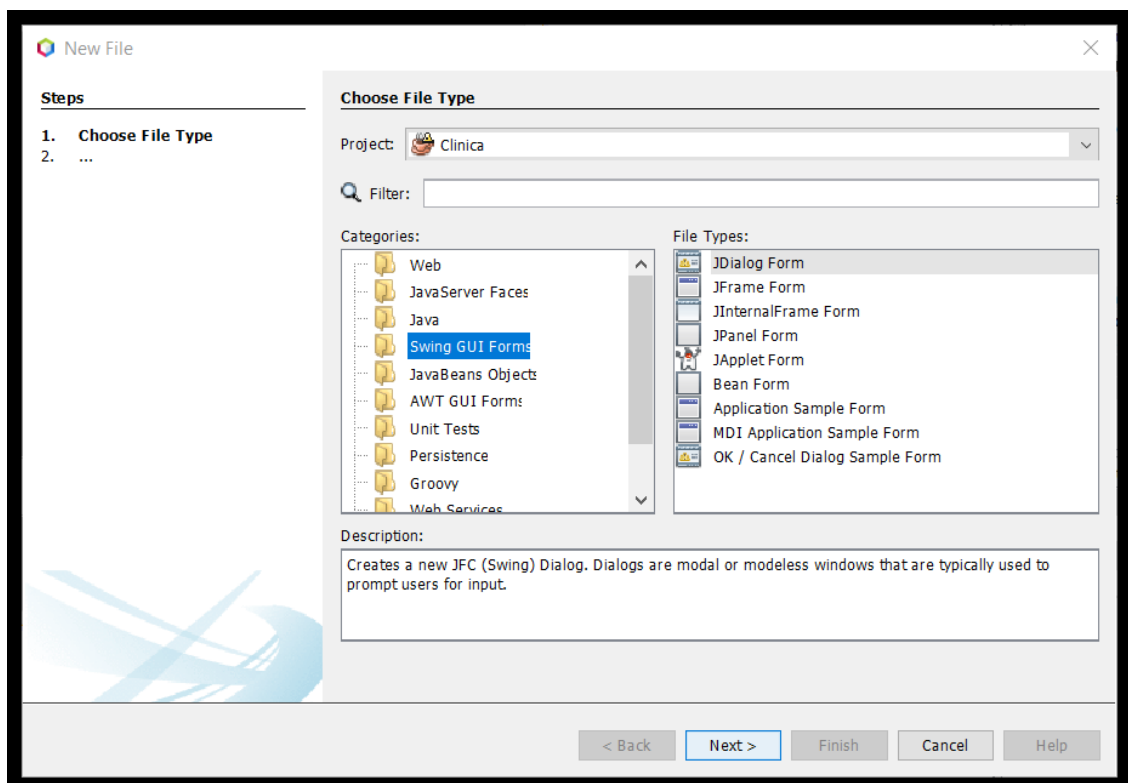


```
1 package model;
2 public class PacienteListar {
3     //Criando nossos atributos - variaveis
4     private int id;
5     private String nome;
6     private float peso;
7     private float altura;
8
9     public PacienteListar() {
10    }
11
12    // Botao direito - insertCode - generate constructor
13    public PacienteListar(int id, String nome, float peso, float altura){
14        this.id = id; //referenciando ao id do Banco de Dados
15        this.nome = nome; //referenciando ao nome no Banco de Dados
16        this.peso = peso; //referenciando ao peso no Banco de Dados
17        this.altura = altura; //referenciando à altura no Banco de Dados
18    }
19
20    //Metodos Getters e Setters - para este momento, podemos clicar com botão direito, irmos em INSERTCODE e pedirmos para gerar automaticamente os Metodos..
21    public int getId(){
22        return id;
23    }
24    public void setId(int id){
25        this.id = id;
26    }
27    public String getNome(){
28        return nome;
29    }
30    public void setNome(String nome){
31        this.nome = nome;
32    }
33    public float getPeso(){
34        return peso;
35    }
36    public void setPeso(float peso){
37        this.peso = peso;
38    }
39    public float getAltura(){
40        return altura;
41    }
42    public void setAltura(float altura){
43        this.altura = altura;
44    }
45
46    @Override
47    public String toString(){
48        return nome;
49    }
50 }
```

Pacote View – Nesse momento, após criarmos as classes nos Pacotes Controller e Model vamos criar a interação do Projeto com o Usuário, 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 Listar.

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.

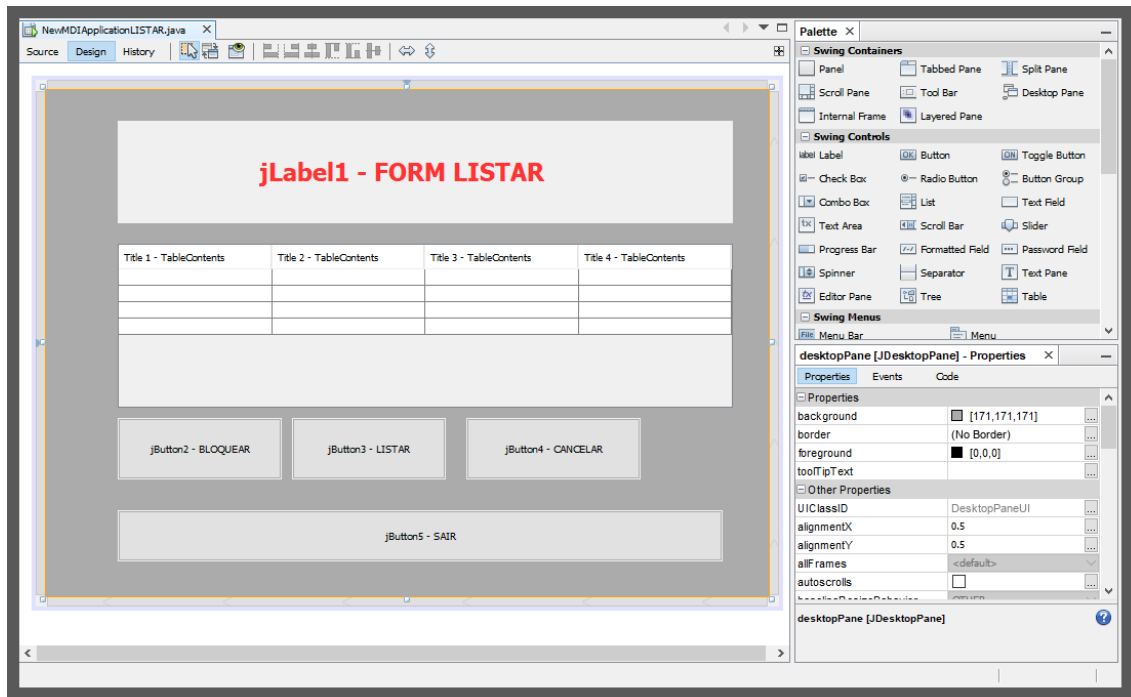
Vamos clicar com Botão direito no pacote View irmos em **New/Other** e selecionar **Swing Gui Forms**, nesta categoria vamos selecionar **MDI Application Sampe Form**.



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.

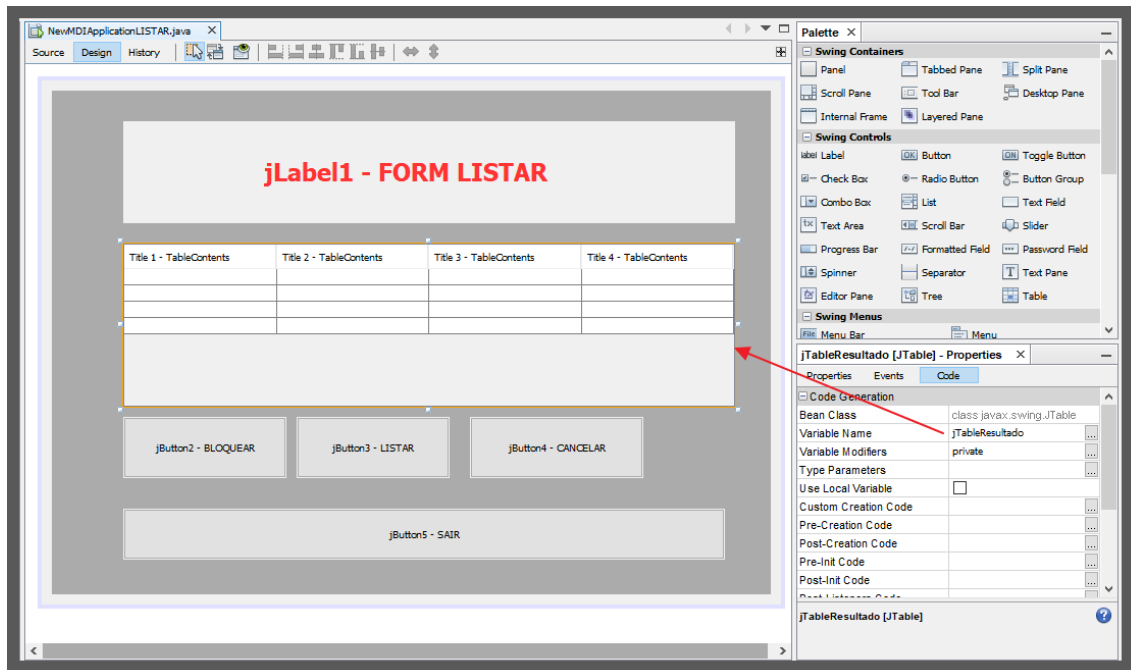
Para maior familiaridade, vou por a tela criada com os recursos do Palette utilizado para fins didáticos.



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.JDesktopPane desktopPane;
private javax.swing.JButton jButtonBlock;
private javax.swing.JButton jButtonCancelar;
private javax.swing.JButton jButtonListar;
private javax.swing.JButton jButtonSair;
private javax.swing.JLabel jLabel1;
private javax.swing.JPanel jPanel1;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTable jTableResultado;
// 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:

```
197 //Passo 1 - nesse momento, iremos configurar nosso formulario
198 private void configurarFormulario(){
199     this.setTitle("Java Intermediário - Escola Evolua Sumaré"); //definiremos o Titulo do nosso Form
200     this.setResizable(false); //Desabilitaremos o botao Maximizar
201     this.setLocationRelativeTo(null); //configuraremos nosso form para iniciar ja no centro da tela
202     estadoControle(true); // add após criar estado controle ( passo 2)
203     preencherTabela(new PacienteDAOListar().listar()); //add apos criar preencherTabela (passo 4)
204 }
205 //passo 2
206 private void estadoControle(boolean e){
207     jButtonBlock.setEnabled(e); //Deixando nosso botão BLOCK habilitado neste momento
208     jButtonCancelar.setEnabled(!e); //Deixando nosso botão CANCELAR desabilitado neste momento
209     jButtonListar.setEnabled(e); //Deixando nosso botão LISTAR desabilitado neste momento
210 }
211 //Passo 3
212 private void configurarTabela(){
213     DefaultTableModel m = new DefaultTableModel(){
214         @Override
215         public boolean isCellEditable(int row, int column){
216             return false;
217         }
218     };
219     m.addColumn("Identificação"); //Nomeando nossa coluna 1
220     m.addColumn("Nome do Paciente"); //Nomeando nossa coluna 2
221     m.addColumn("Peso do Paciente"); //Nomeando nossa coluna 3
222     m.addColumn("Altura do Paciente"); //Nomeando nossa coluna 4
223     jTableResultado.setModel(m);
224 }
225 //passo 4
226 private void preencherTabela(List<PacienteListar> lista){
227     if(lista.size()>0){
228         configurarTabela();
229         DefaultTableModel m = (DefaultTableModel)jTableResultado.getModel();
230
231         for(PacienteListar p : lista){
232             m.addRow(new Object[]{
233                 p.getId(), p.getNome(), p.getPeso(), p.getAltura()
234             });
235         }
236         jTableResultado.setModel(m);
237     }
238 }
239 }
240 }
241 }
```


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

```

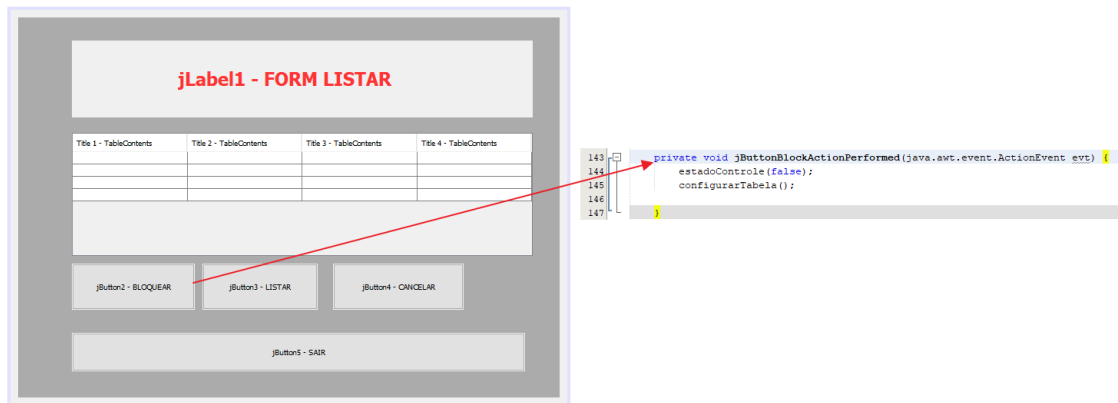
8 public class NewMDIApplicationLISTAR extends javax.swing.JFrame {
9
10 public NewMDIApplicationLISTAR() {
11     initComponents();
12     configurarFormulario();
13 }

```

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

Volte no Design do Projeto e dê um duplo click BOTAO jButton2 - BLOQUEAR

Aqui definimos que o nosso **estadocontrole** ao ter o Botão jButton2 – BLOQUEAR pressionado, habilitará os campos que até então temos ele bloqueado e chamamos o Método Limpar para limpar todos os textos que estiverem disponíveis.



```

143 private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
144     estadoControle(false);
145     configurarTabela();
146 }

```

Volte no Design do Projeto e dê um duplo click BOTAO jButton4 - CANCELAR

Neste botão, vamos fazer o inverso do estadocontrole do botão novo, pois ao clicar em CANCELAR, queremos que os campos textos e os botões SALVAR e CANCELAR sejam bloqueados, no entanto, basta neste momento definirmos o **estadocontrole** como (true).



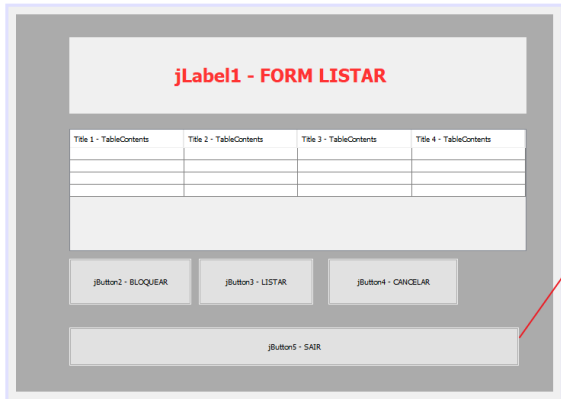
```

135 private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
136     estadoControle(true);
137 }

```

Volte no Design do Projeto e dê um duplo click BOTAO jButton5 - SAIR

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



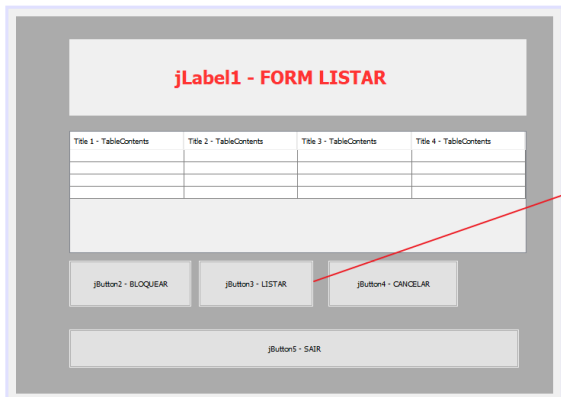
```

149 private void jButtonSairActionPerformed(java.awt.event.ActionEvent evt) {
150     dispose();
151 }

```

Agora está faltando apenas nosso botão LISTAR, então vamos manipular ele, pois é um código um pouco maior.

Volte no Design do Projeto e dê um duplo click BOTAO jButton3 - LISTAR

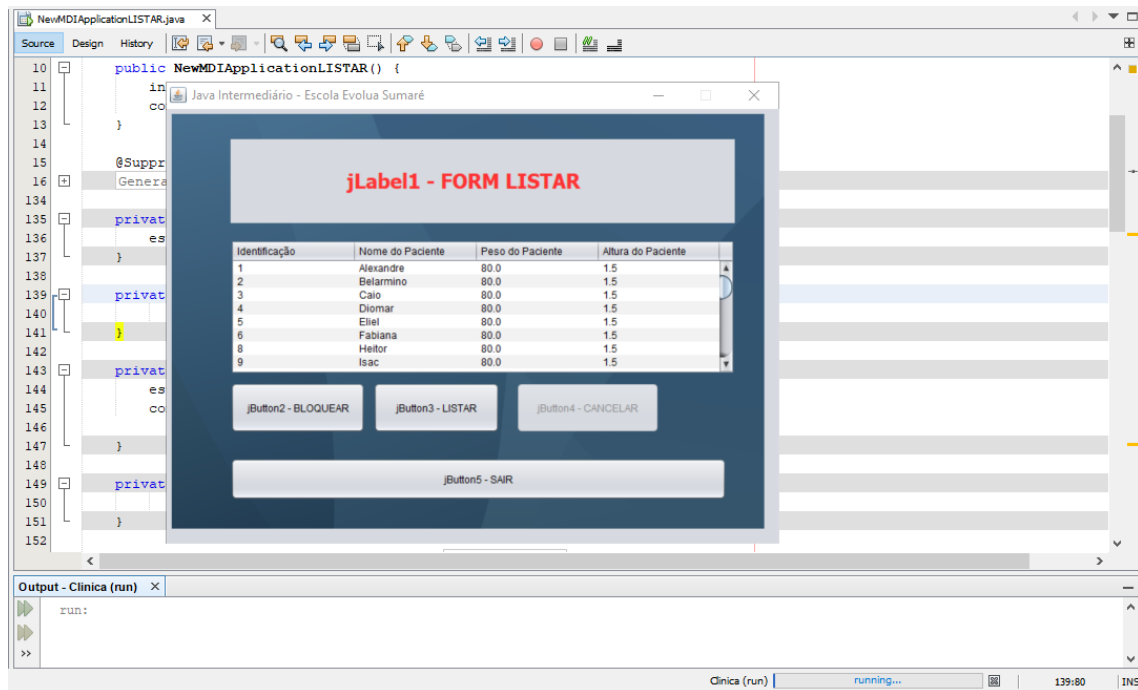


```

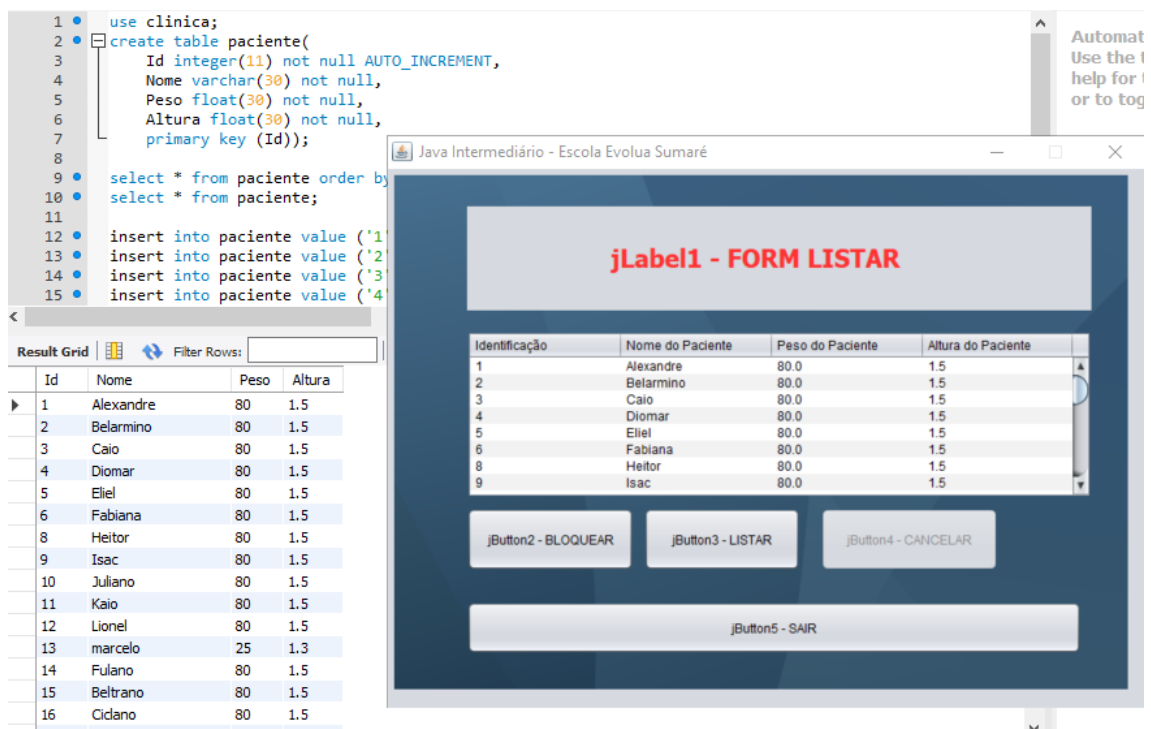
139 private void jButtonListarActionPerformed(java.awt.event.ActionEvent evt) {
140     preencherTabela(new PacienteDAOListar().listar());
141 }

```

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”.

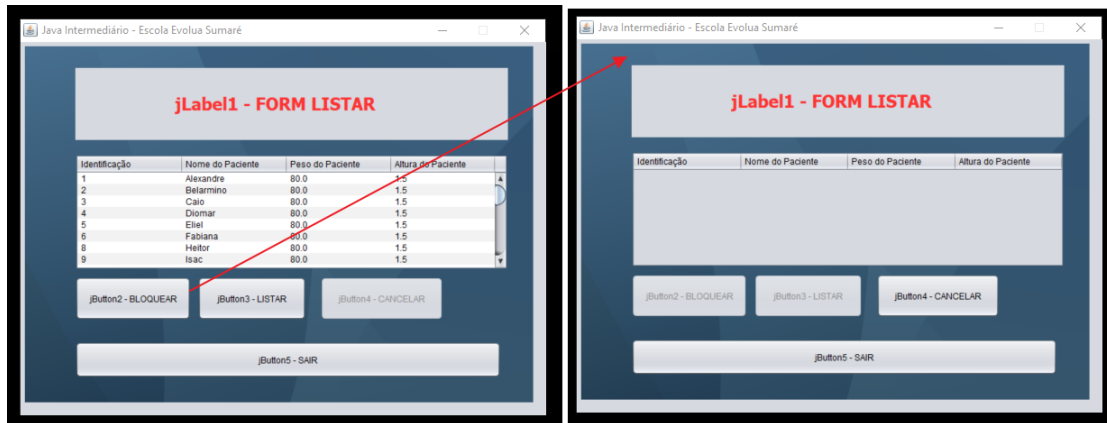


Vimos que iniciou, e trouxe os dados contidos no nosso Banco de Dados, para maior clareza vamos rodar um “Select” no Banco de Dados para comprovar.



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

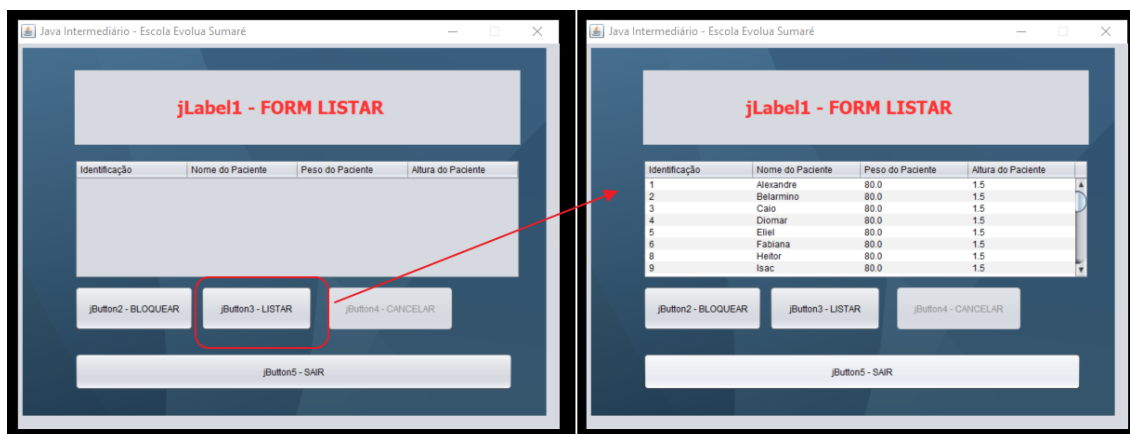
⇒ Clicar no Botão Bloquear



⇒ Clicar no Botão Cancelar



⇒ Clicar no Botão Listar



⇒ Clicar no Botão Sair



Nota-se que os códigos funcionaram corretamente, no entanto, saliento que este artigo 23/2020 assim como o artigo 20/2020 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 !

