Dica: Leia com cautela siga o Passo a Passo, caso você não tenha paciência seu programa não funcionará.

Considerações sobre a plataforma Java

O que vai ser feito.

Para criarmos nosso exemplo, precisaremos:

1. Banco de dados MySQL;
2. IDE NetBeans.

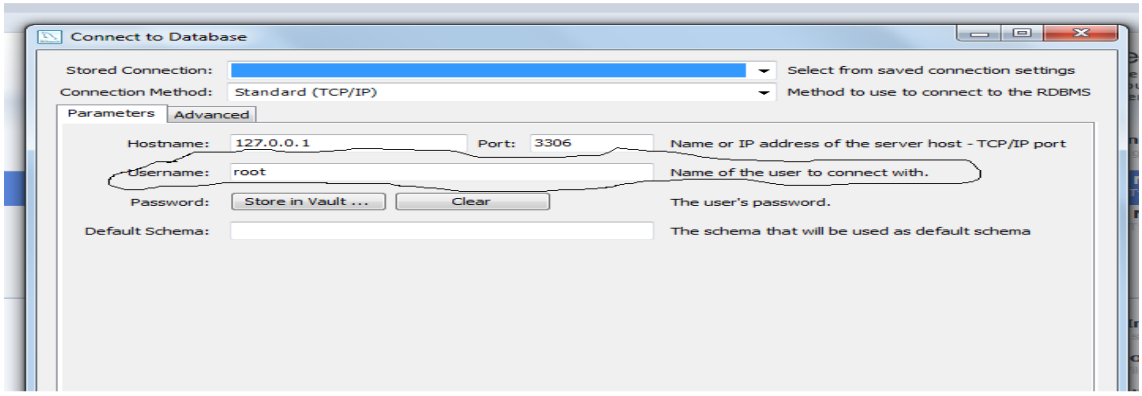
Além das ferramentas, iremos desenvolver alguns passos:

1. Banco de Dados;
2. O Projeto;
3. Factory;
4. Modelo;
5. DAO;
6. GUI;
7. Evento SAIR;
8. Evento LIMPAR;
9. Evento CADASTRAR;
10. Consulta através do console do MySQL Server.

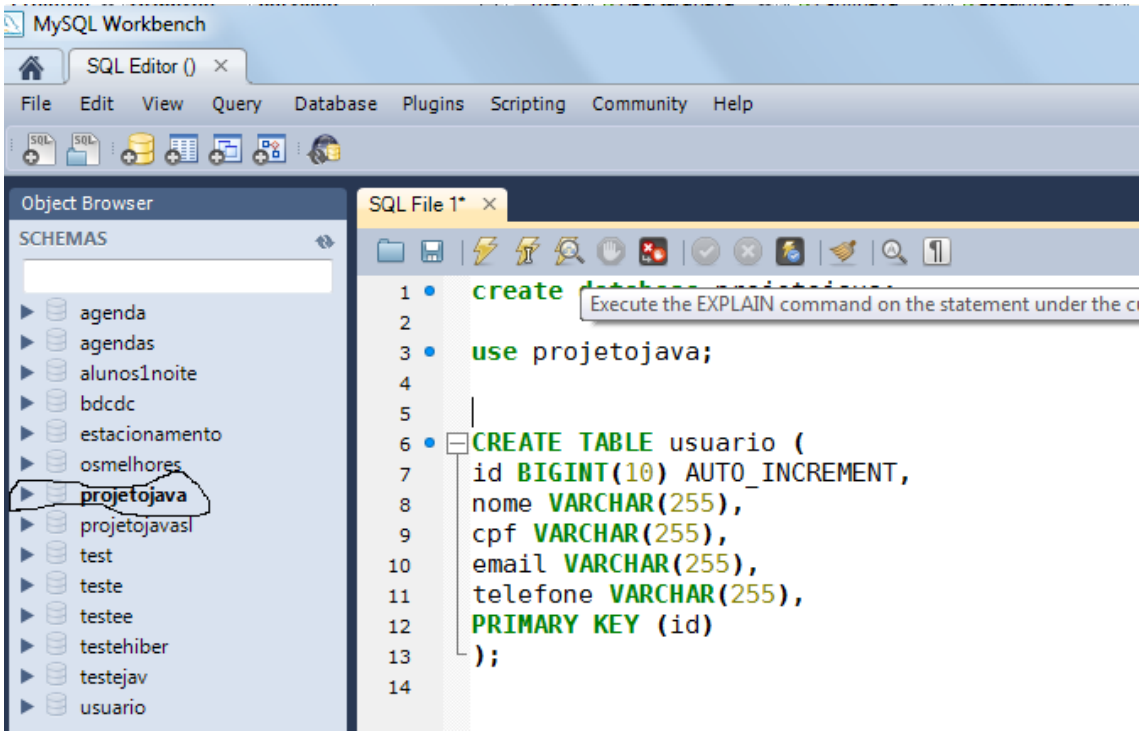
Iniciar a prática.

Passo 1: Banco de Dados

Abra o Workbench digite os seguintes comandos:




```
create database projetojava;
use projetojava;
CREATE TABLE usuario (
id BIGINT(10) AUTO_INCREMENT,
nome VARCHAR(255),
cpf VARCHAR(255),
email VARCHAR(255),
telefone VARCHAR(255),
PRIMARY KEY (id)
);
```



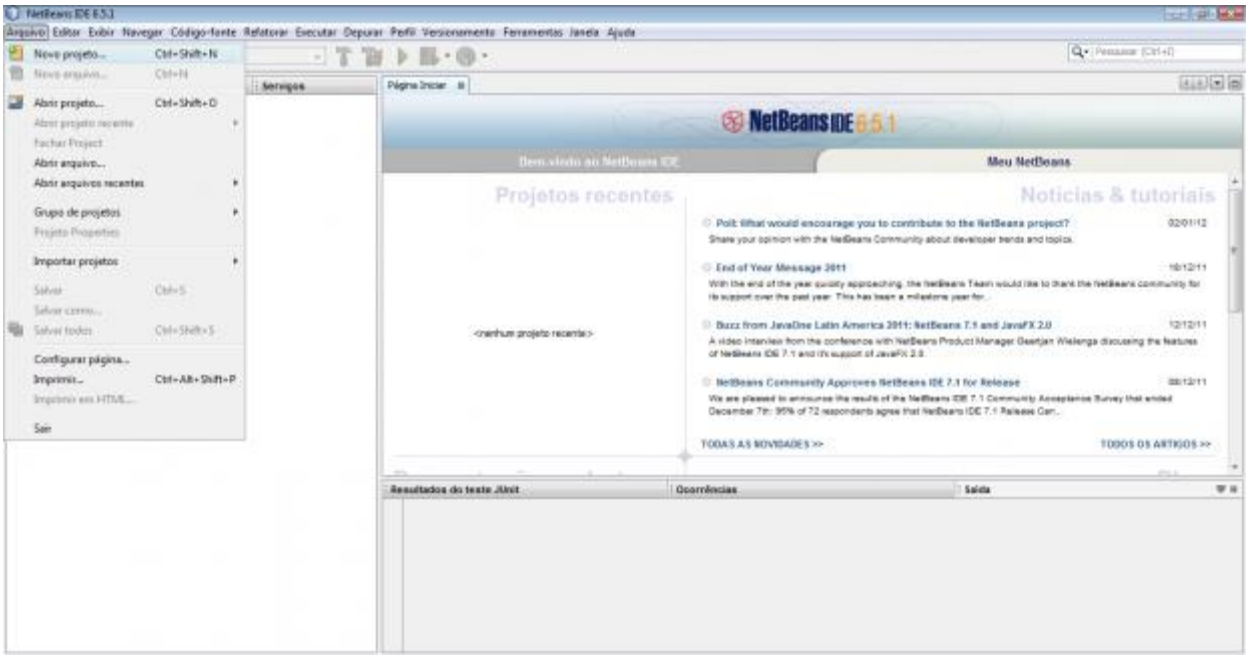
Ou seja, teremos a tabela usuario com 5 atributos (id, nome, CPF, email, telefone).

Agora, vamos abrir o NetBeans em **Iniciar > Todos os programas >NetBeans>NetBeans IDE>NetBeans IDE** e iniciar o desenvolvimento da aplicação.

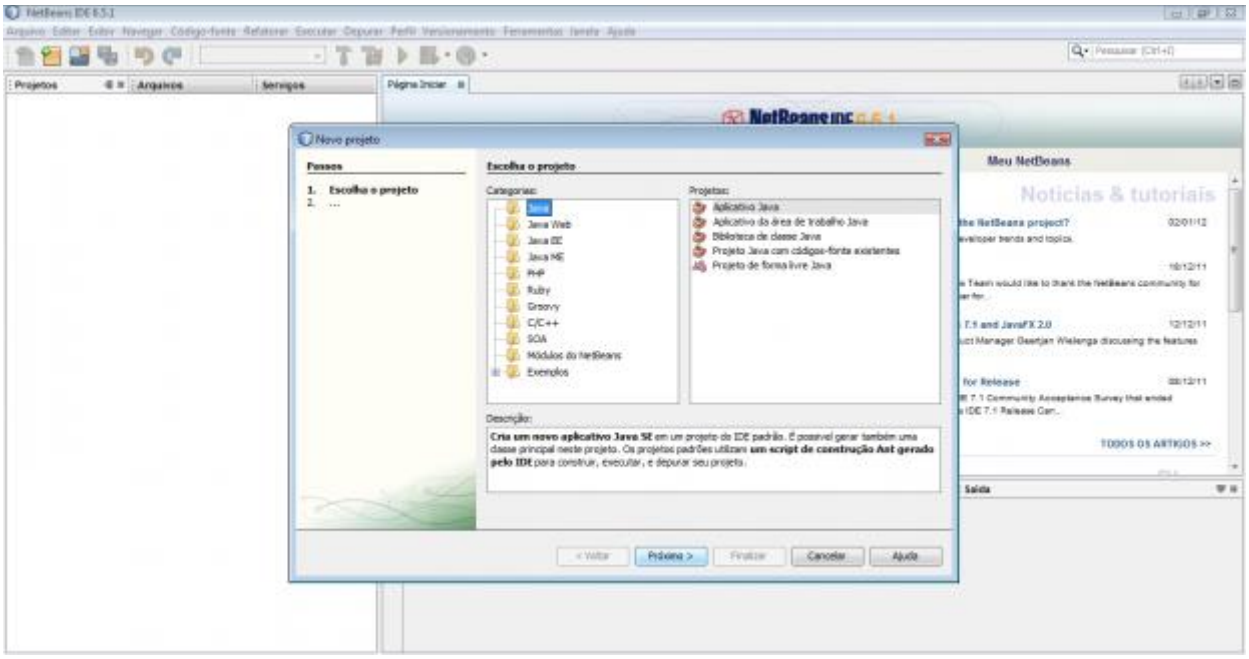
OBS.: durante todo o desenvolvimento da aplicação seguirão imagens para facilitar o aprendizado.

Passo 2: O Projeto

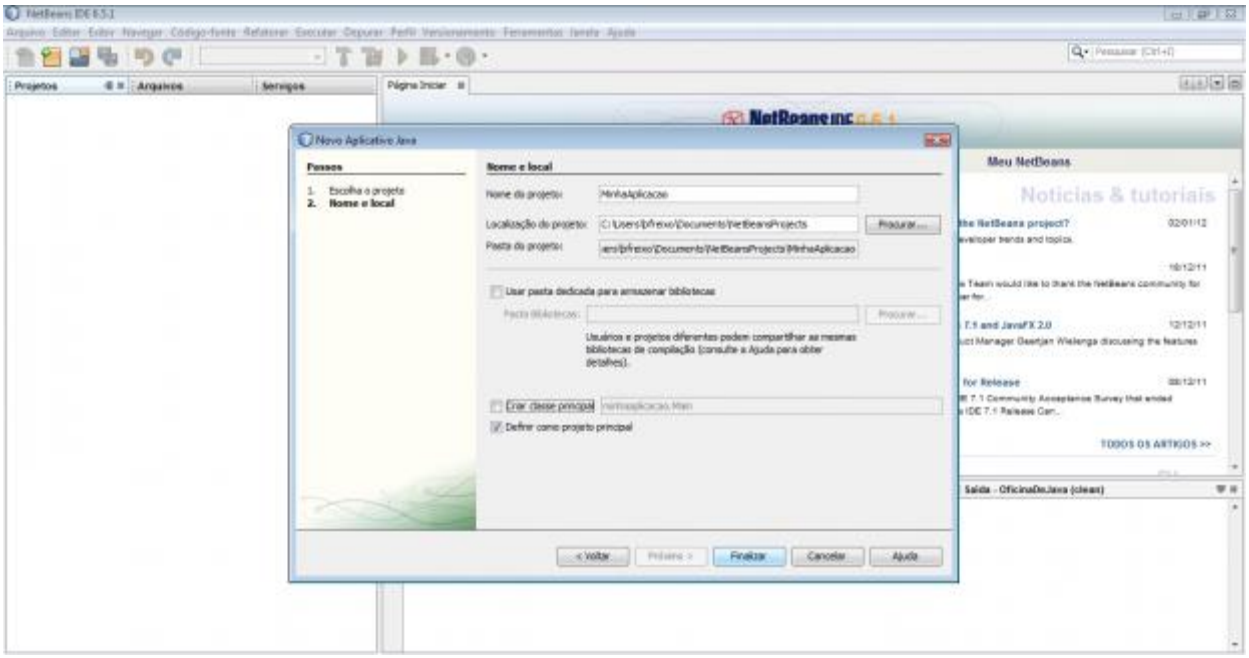
Arquivo > Novo projeto;



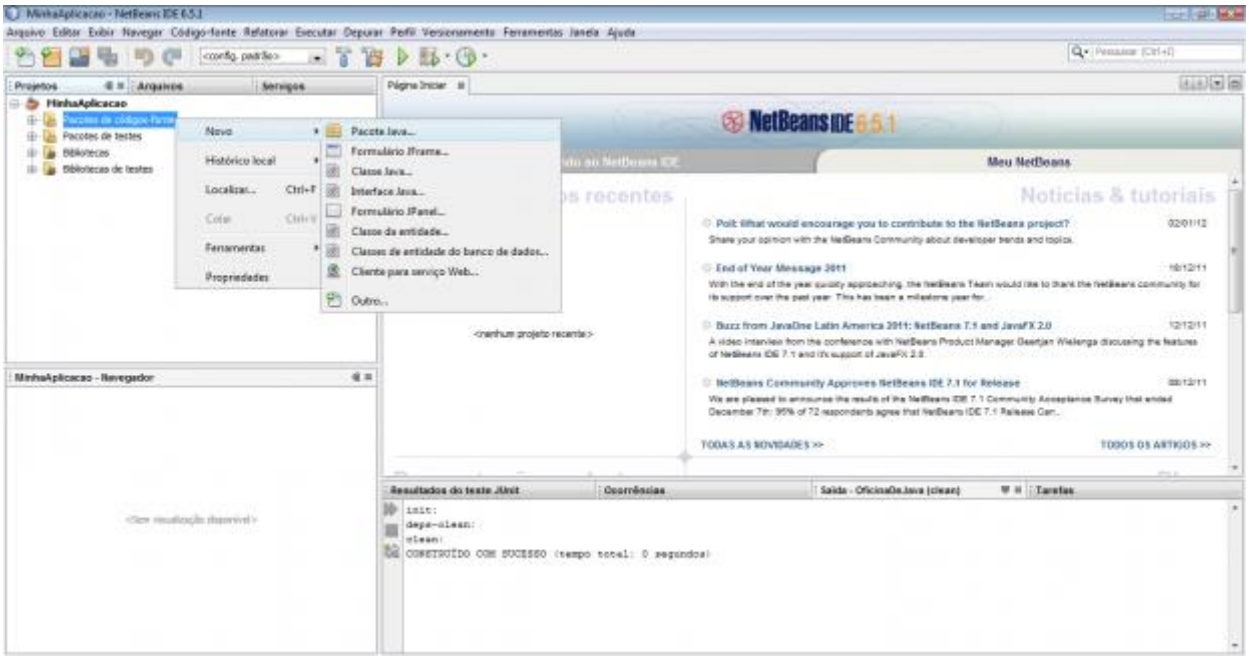
Na aba Novo projeto, clique em Java e em Aplicativo Java. Clique em próximo.



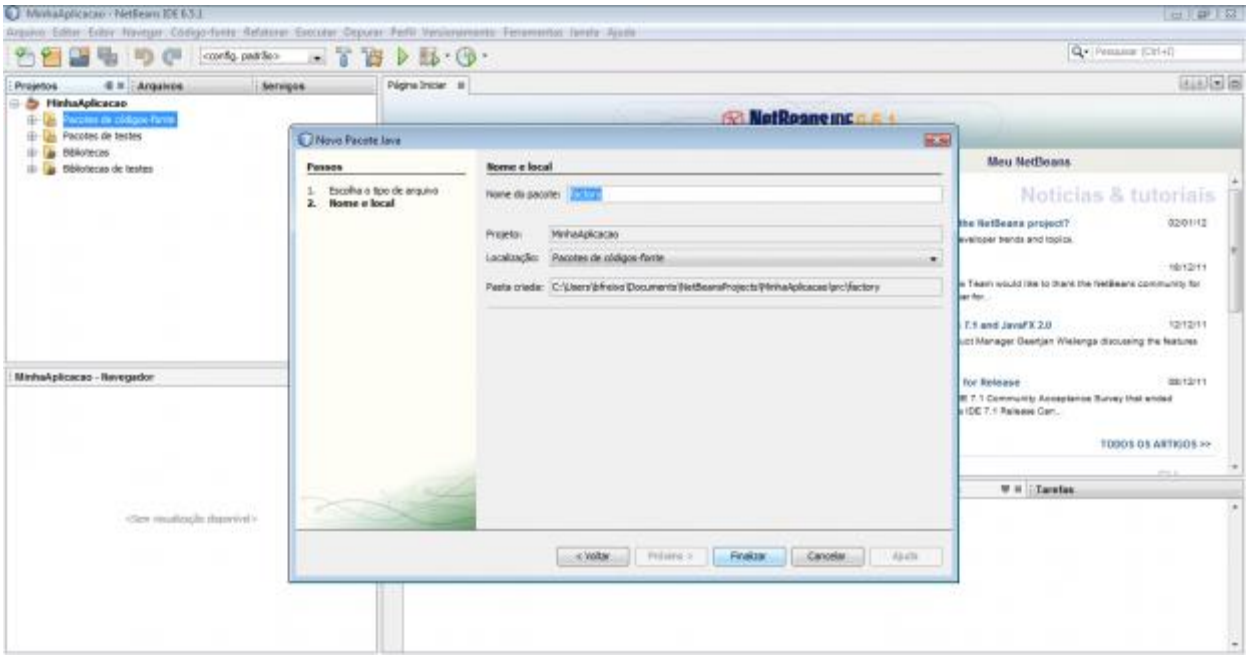
Na aba Novo aplicativo Java, em “Nome do projeto” digite MinhaAplicacao.
Desmarque a opção “Criar classe principal”. Clique em finalizar.



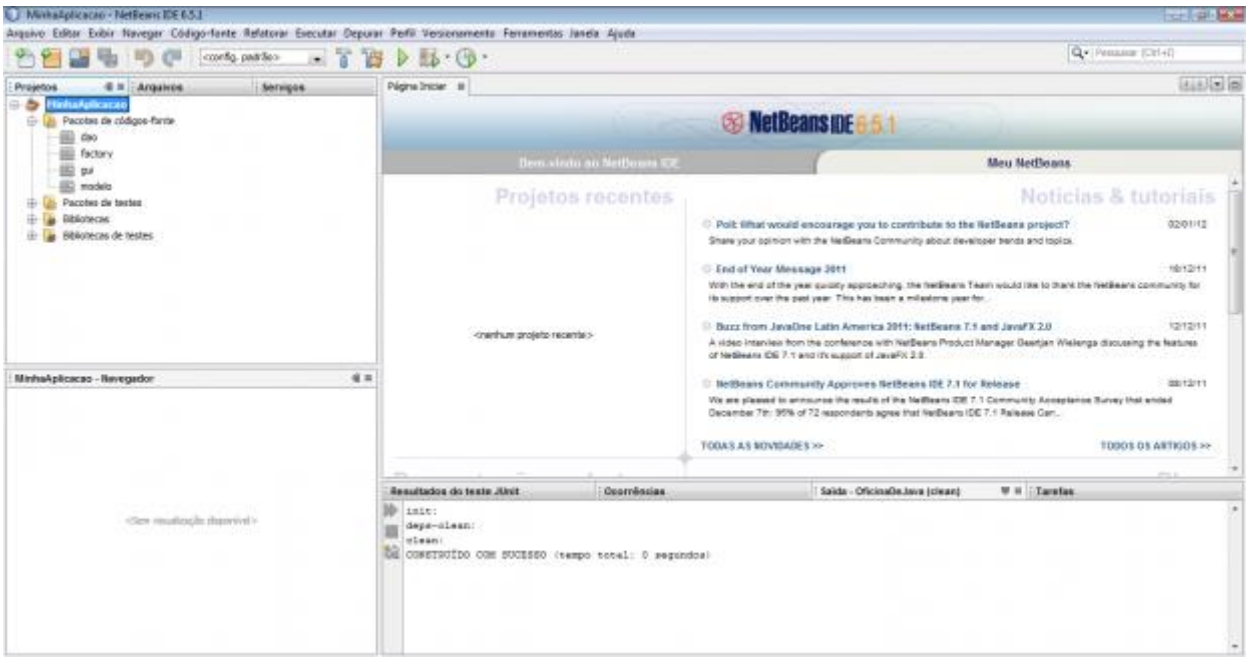
Agora vamos criar os pacotes ou packages. Clique com o botão direito em “Pacotes de códigos-fonte” e com o botão esquerdo do mouse escolha **Novo > Pacote Java...**



Na aba Novo Pacote Java digite "factory” para Nome do pacote.
Clique em Finalizar.



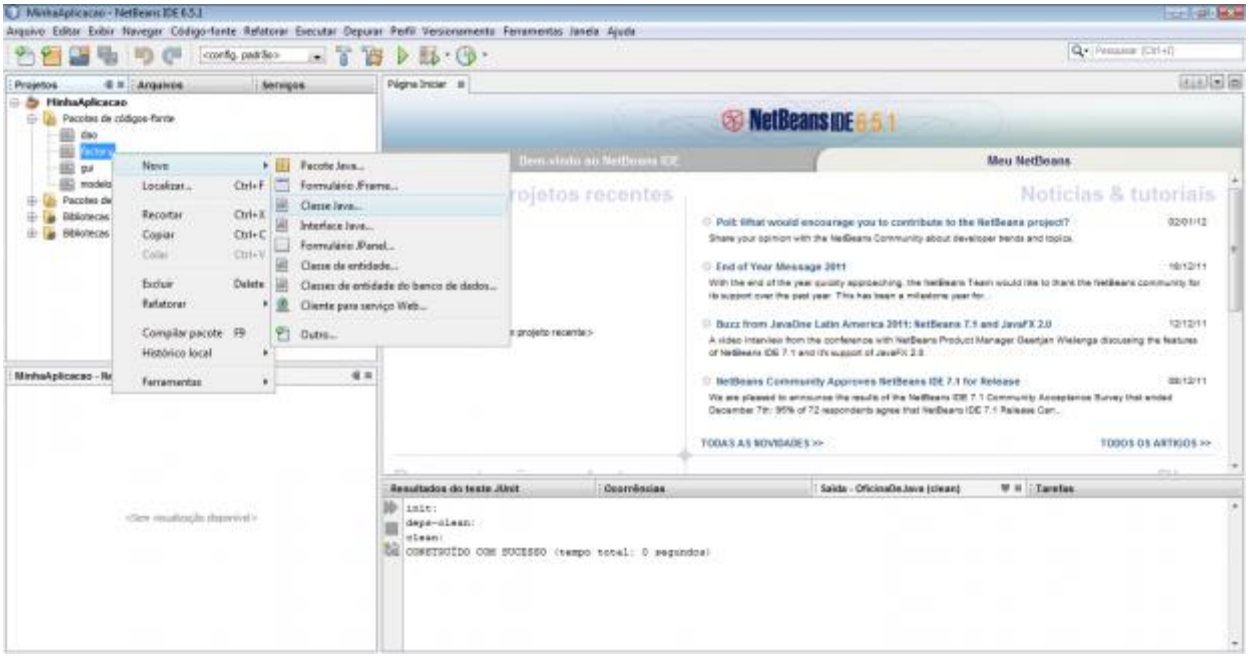
Repita o processo de criação de pacote, criando os seguintes pacotes, além do pacote **factory**:**modelo**, **dao**, **gui**. Assim ficará a visão geral do projeto (no canto esquerdo do NetBeans):



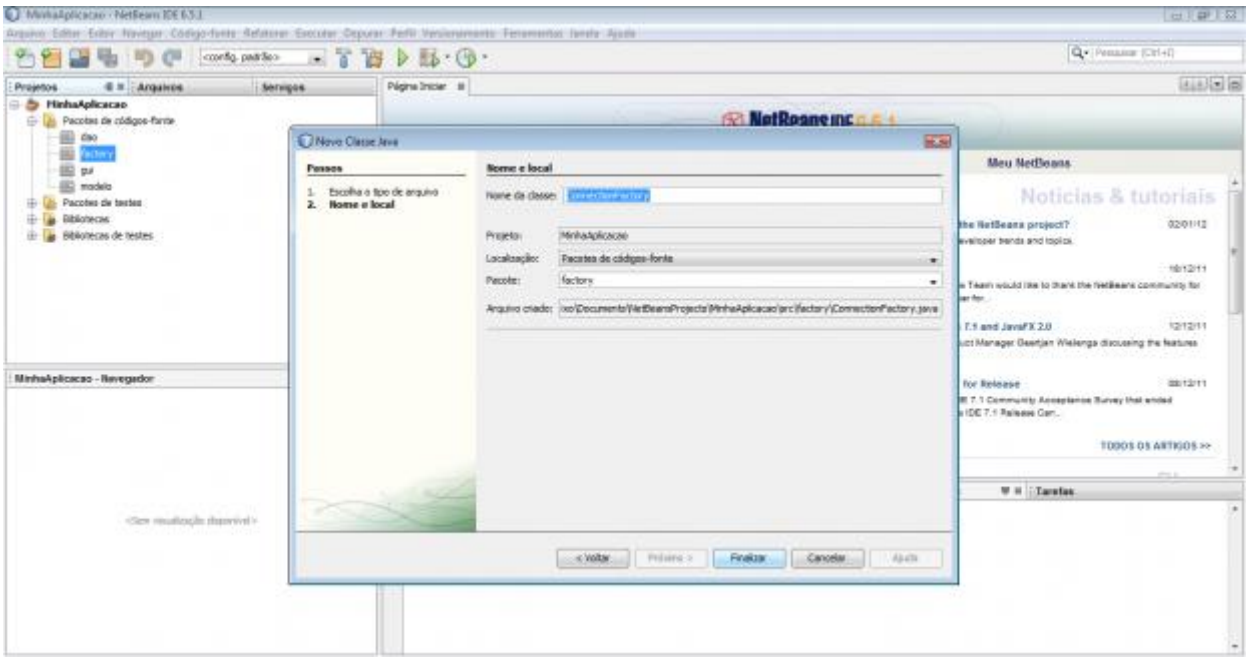
Passo 3: Factory:

Factory significa “fábrica” e ConnectionFactory significa fábrica de conexões. Factory será o nome do pacote e ConnectionFactory o nome da classe que fará a interface com o driver JDBC de conexão a qualquer banco que desejar. Por isso o nome “fábrica”, pois o JDBC permite a conexão a qualquer banco: MySQL, Postgree,

Oracle, SQL Server, etc., somente alterando a linha do método “getConnection”. Vamos começar criando a classe ConnectionFactory no pacote factory. Vá com o botão direito até factory e clique com o botão esquerdo em **Novo > Classe Java**.



Na aba Novo Classe Java em Nome da Classe escolha o nome **ConnectionFactory**. Clique em Finalizar.



O script abaixo representa a classe de conexão ConnectionFactory. Copie e cole na classe ConnectionFactory

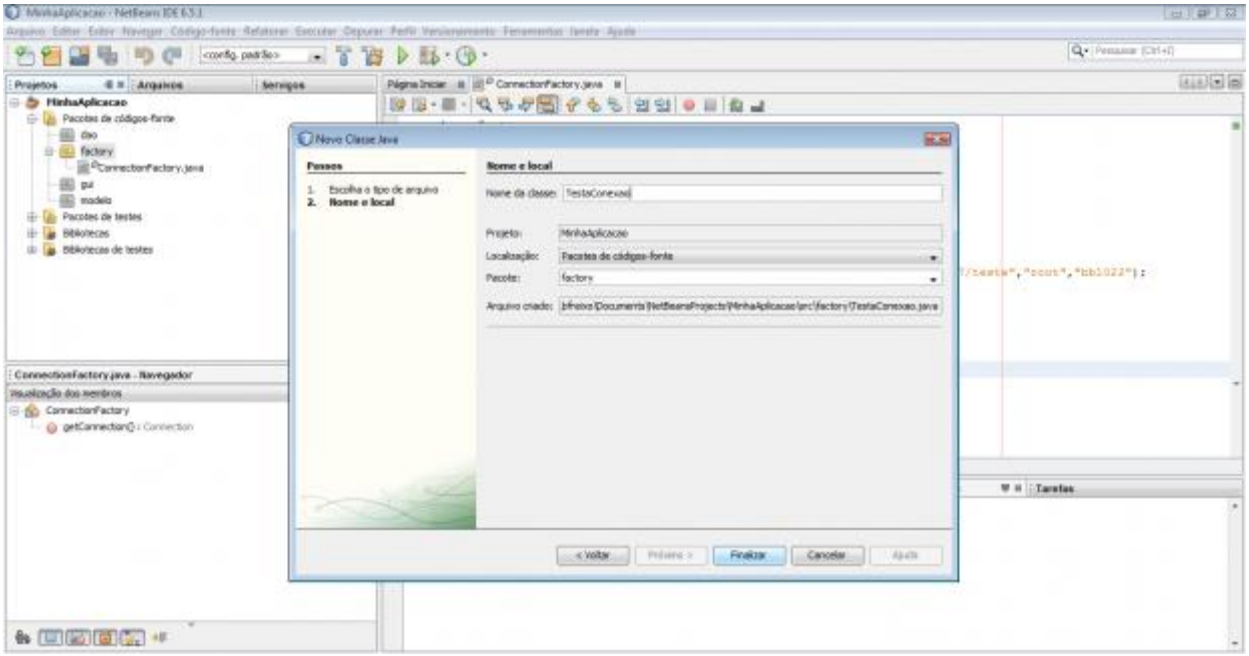
```
// situa em qual package ou “pacote” está a classe

Package factory;

// faz as importações de classes necessárias para o funcionamento do programa

import java.sql.Connection; // conexão SQL para Java

import java.sql.DriverManager; // driver de conexão SQL para Java

import java.sql.SQLException; // classe para tratamento de exceções



public class ConnectionFactory {

    public Connection getConnection() {

        try {

            return
DriverManager.getConnection("jdbc:mysql://localhost/projetojava","seu-nome-de-
usuario","sua-senha");

        }

        catch(SQLException excecao) {

            throw new RuntimeException(excecao);

        }

    }

}
```

Salve a alteração (CTRL+S).

OBS.: não esqueça de salvar todas as alterações nos códigos ao decorrer do tutorial.

OBS 2: altere "seu-nome-de-usuario" e "sua-senha" se tive no entanto vocês irão utilizar o root como usuário nas configurações do seu Banco de Dados.

Vamos criar uma classe para testar a conectividade ao MySQL. Pode ser dentro do pacote factory mesmo...
Coloquemos o nome **TestaConexao**:



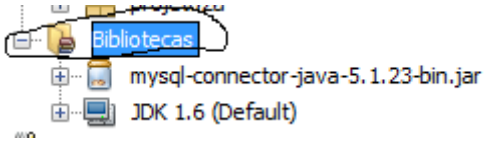
Clique em Finalizar.

Script da classe TestaConexao:

```
package factory;
import java.sql.Connection;
import java.sql.SQLException;
public class TestaConexao {
    public static void main(String[] args) throws SQLException {
        Connection connection = new ConnectionFactory().getConnection();
        System.out.println("Conexão aberta!");
        connection.close();
    }
}
```

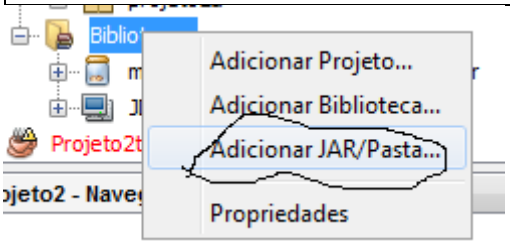
Vocês já possuem o JCONNECTOR.
No entanto caso vocês queiram pode-se baixar no seguinte Link.

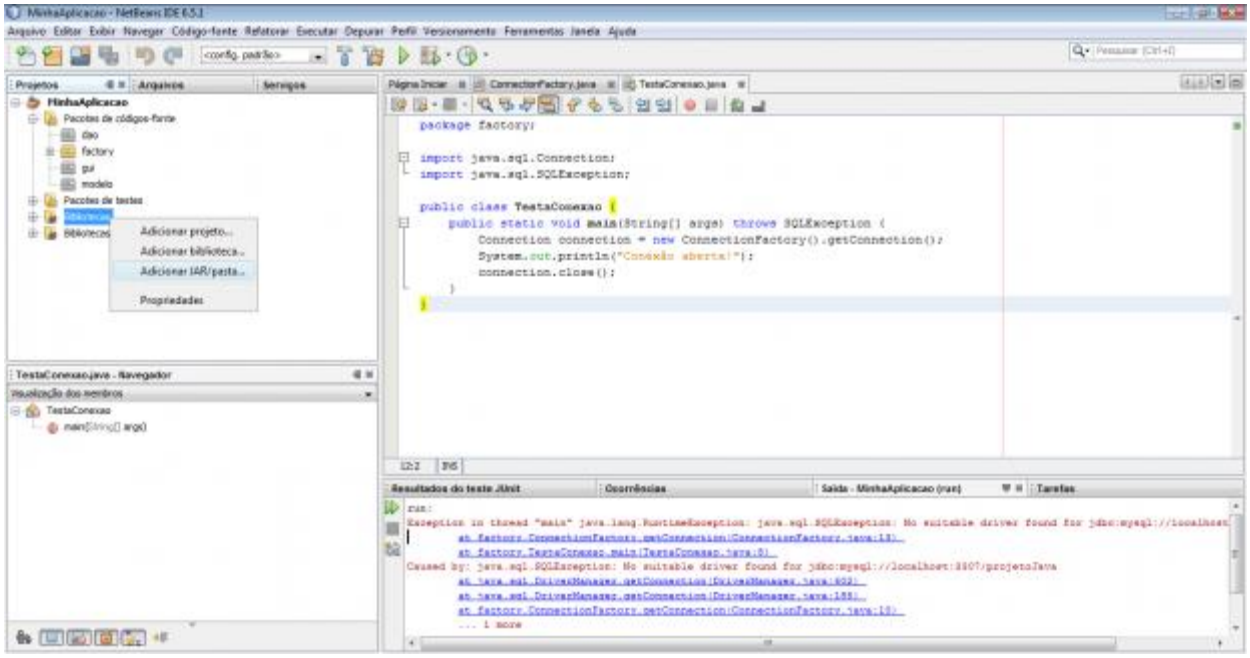
Para executar qualquer aplicativo no NetBeans teclamos **SHIFT+F6**. Faça-o. Perceba que uma mensagem de erro é exibida no console. Esta mensagem de erro significa ausência do driver JDBC. Precisamos baixá-lo para assim fazermos a conexão. Endereço para download: http://dev.mysql.com/downloads/mirror.php?id=404191#mirrors
Se o arquivo vier compactado, descompacte-o e escolha o diretório de sua preferência.
Depois de baixar o driver JDBC, vá em: **Bibliotecas > Adicionar JAR/pasta...**



Esta em bibliotecas.

clique com o botão direito sobre biblioteca / Vá em adicionar Jar.





Escolha o diretório onde instalou o driver JDBC e clique em Open. Execute o projeto. Agora sim funcionou!

Se a mensagem que apareceu no console foi parecida com:

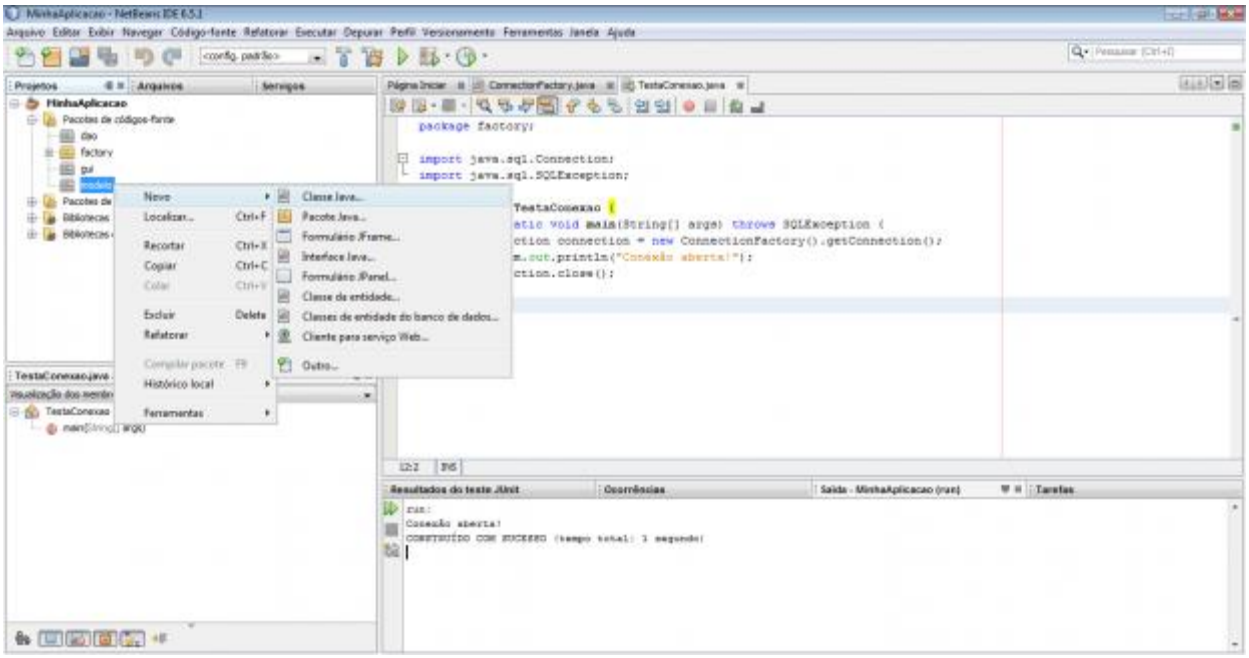
```
run:

Conexão aberta!

CONSTRUÍDO COM SUCESSO (tempo total: 1 segundo)
```

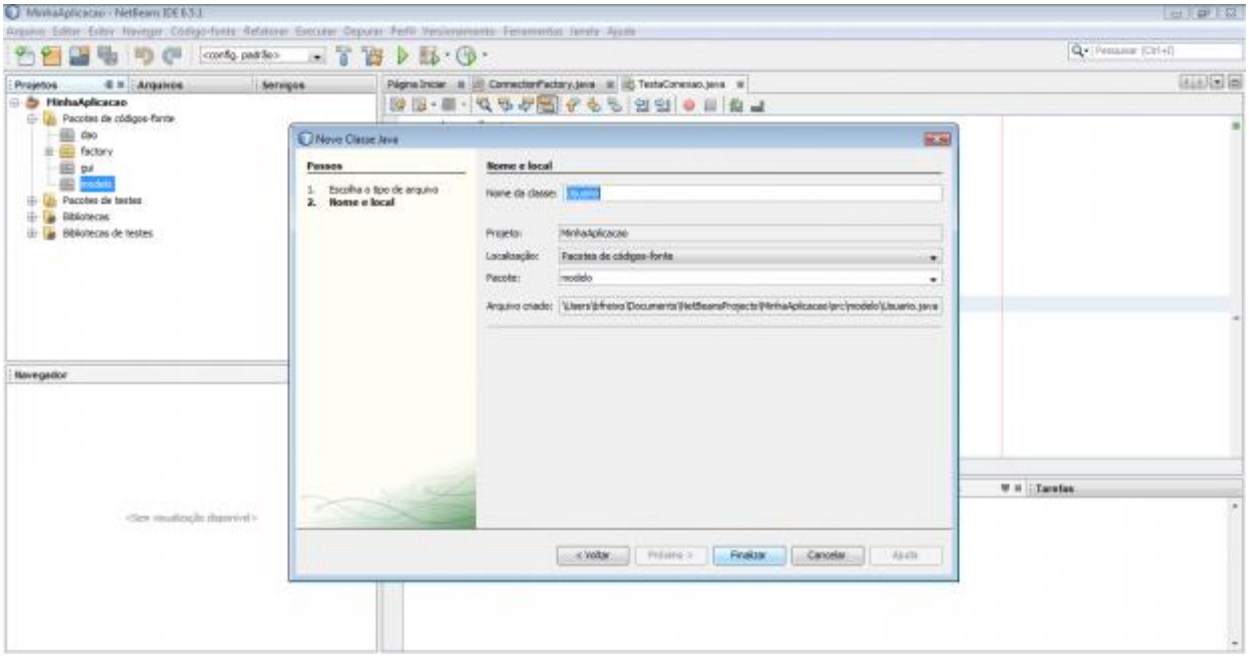
então sua conexão foi estabelecida!

OBS 3: se mesmo assim não funcionar, especifique a porta do servidor ao lado do localhost.
Exemplo: localhost:3307

Passo 4: Modelo:

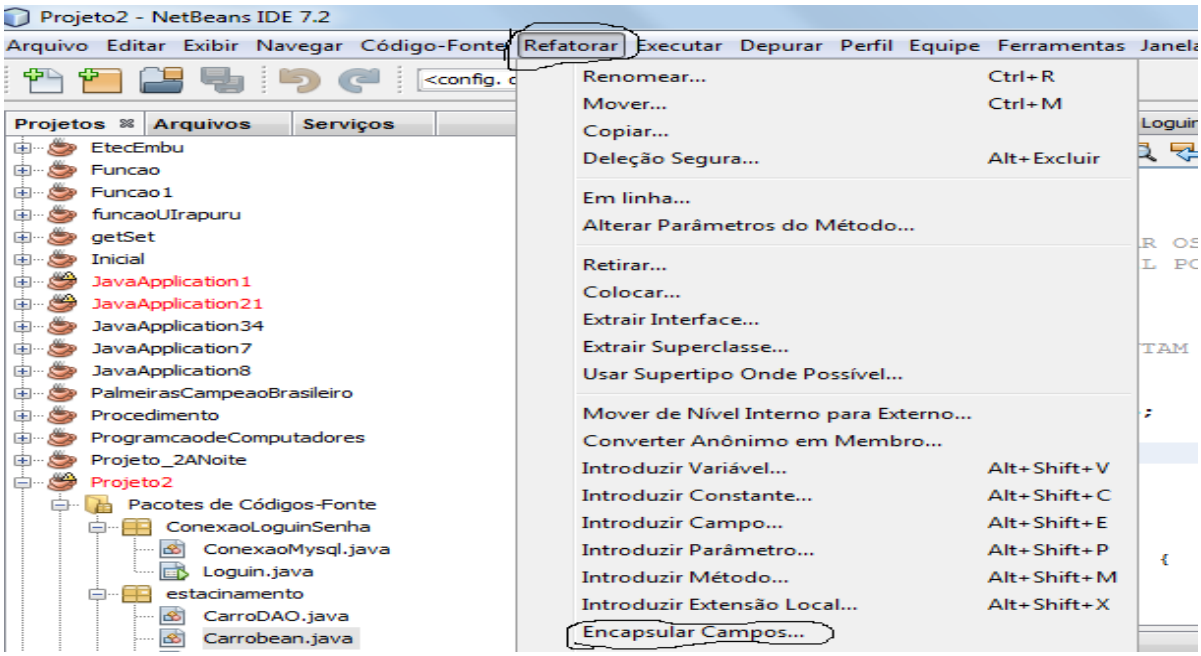
Agora, criemos a classe Usuario, dentro do pacote modelo: **modelo > Novo > Classe Java > Usuario > Finalizar**.

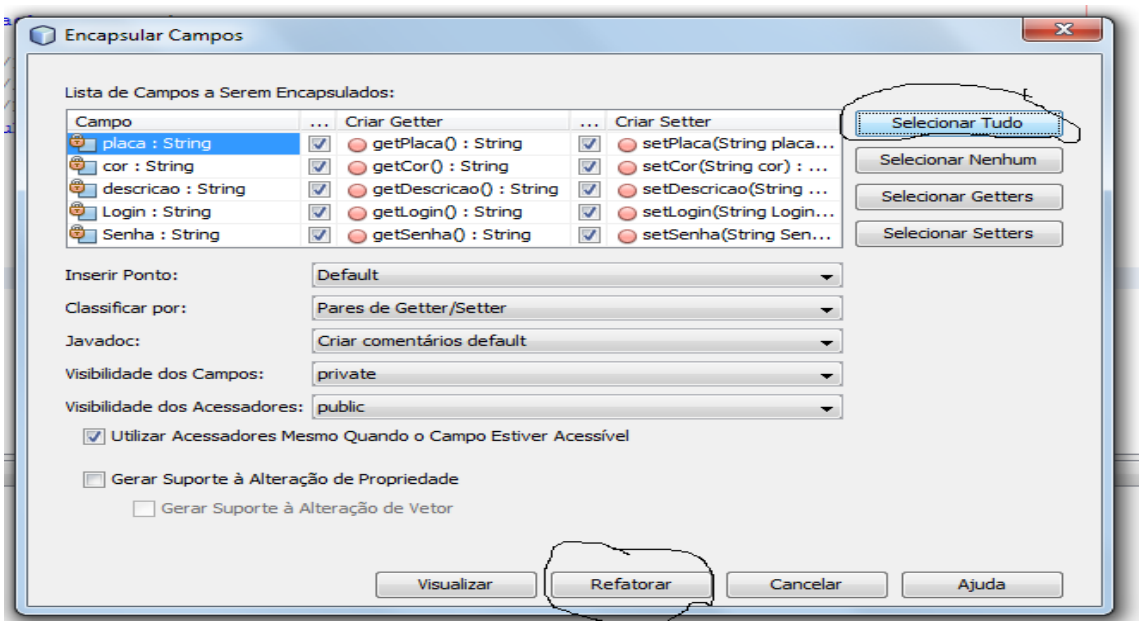




Crie as variáveis id (Long), nome, CPF, email, telefone (todas string) e os métodos getters e setters. Assim ficará o script da classe: Atenção vai no menu refatorar

Aula DS 30/10/2024 Professor Thiago/Sueli






```
package modelo;
public class Usuario {
    Long id;
    String nome;
    String cpf;
    String email;
    String telefone;

    public String getCpf() {
        return cpf;
    }

    public void setCpf(String cpf) {
```

```
        this.cpf = cpf;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getTelefone() {
        return telefone;
    }

    public void setTelefone(String telefone) {
        this.telefone = telefone;
    }
}
```

Passo 5: DAO Uma classe muito importante tem as instruções SQL:

Crie no pacote DAO a classe UsuarioDAO: **dao > Novo > Classe Java > UsuarioDAO > Finalizar**.
Neste pacote ficam as classes que são responsáveis pelo **CRUD** (Create, Retrieve, Update, Delete – ou – Criar, Consultar, Alterar, Deletar), isto é, dados de persistência. Mas no nosso caso não criamos mais que uma tabela na Base de Dados, conseqüentemente, nenhum relacionamento. Além disso, neste exemplo, criaremos o Cadastro de Usuário, isto é, só vamos usar o Create do CRUD. Numa próxima oportunidade podemos aprender os outros métodos (alterar, consultar e

deletar). Em Create, criaremos o método adiciona. Passaremos o próprio objeto "usuario" como parâmetro da função:

adiciona (Usuario usuario). Interessante aqui é que um representa o Bean o outro representa a tabela do banco de dados

Usuario com letra maiúscula representa a classe e com letra minúscula representa o Objeto. Como só vamos representar o método adiciona, não há necessidade de inserir a variável id, pois a mesma é auto-incremento, ou seja, no momento da inserção, este campo será preenchido automaticamente na tabela usuário do Banco de Dados. Se usássemos o método altera ou o método remove, aí sim precisaríamos declarar a variável id. Na classe Usuario do pacote modelo criamos o id pois o modelo do negócio precisa abranger o todo, até mesmo para futuras consultas.

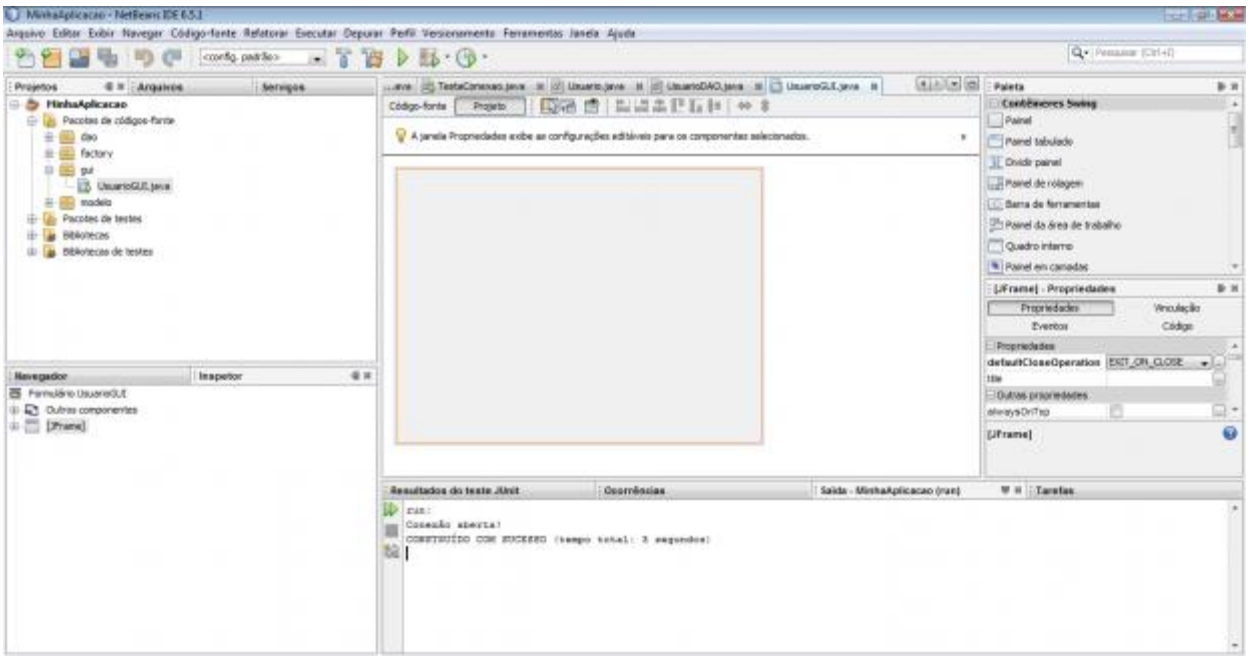
Eis o script abaixo da classe **UsuarioDAO**:

```
package dao;

import factory.ConnectionFactory;
import modelo.Usuario;
import java.sql.*;
import java.sql.PreparedStatement;

public class UsuarioDAO {
    private Connection connection;
    Long id;
    String nome;
    String cpf;
    String email;
    String telefone;


    public UsuarioDAO(){
       this.connection = new ConnectionFactory().getConnection();
    }









    public void adiciona(Usuario usuario){
```

```
String sql = "INSERT INTO usuario(nome,cpf,email,telefone) VALUES(?,?,?,?)";
        try {

            PreparedStatement stmt = connection.prepareStatement(sql);
            stmt.setString(1, usuario.getNome());
            stmt.setString(2, usuario.getCpf());
            stmt.setString(3, usuario.getEmail());
            stmt.setString(4, usuario.getTelefone());
            stmt.execute();
            stmt.close();

        } catch (SQLException u) {
            throw new RuntimeException(u);
        }
    }
}
```

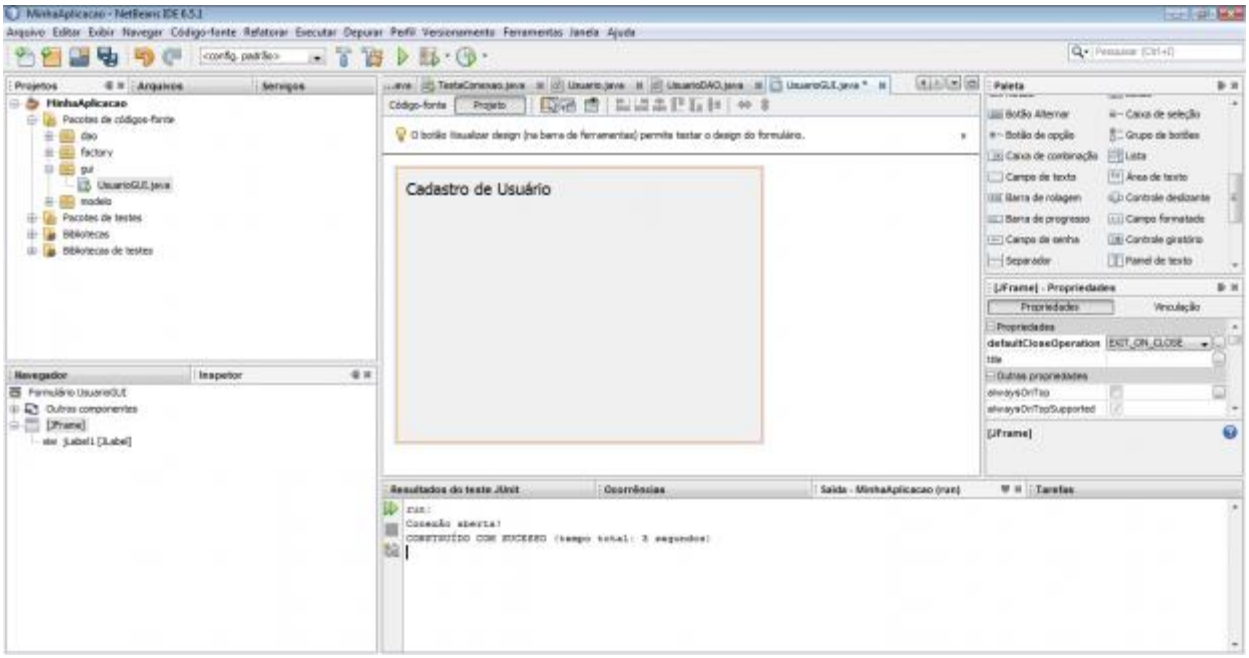
Passo 6: GUI (Graphical User Interface ou Interface Gráfica de Usuário)

Nossa aplicação back-end está toda finalizada. Precisamos aprontar o front-end, isto é, a interface de usuário, a classe que será responsável pela interação com o usuário, ou seja, o formulário de entrada.Vamos criar o formulário que será preenchido pelo usuário: **gui > Novo > Formulario JFrame > UsuarioGUI > Finalizar**. A seguinte tela aparecerá:

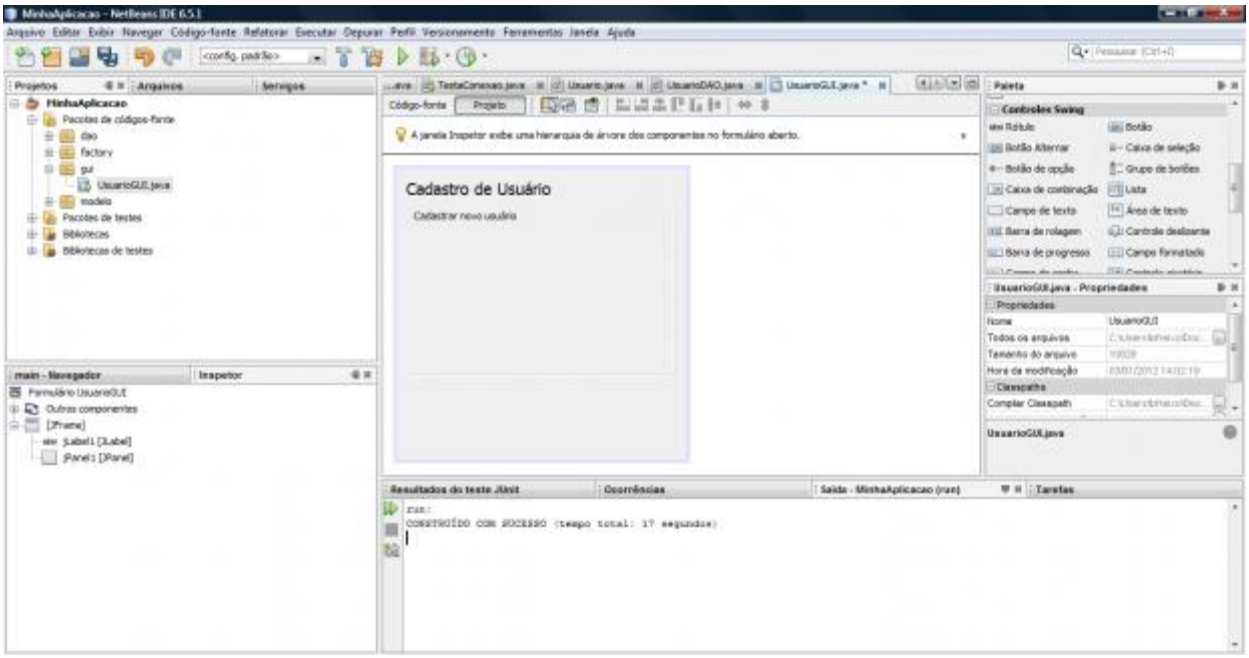


Para criarmos os elementos do formulário é necessário o **arrastar** e **soltar** do mouse. A esse processo, vou criar a sigla **ASM** para facilitar nosso entendimento. Arrasta-se os componentes SWING para o formulário. Portanto, quando eu chamar o nome do componente e colocar ao lado a sigla ASM, subentende-se que é para arrastar componentes da paleta e soltá-los no Formulário.

À direita, na paleta de componentes, em Controles Swing, clique em **Rótulo** ASM. Escreva **Cadastro de Usuário**. Com o botão direito do mouse em cima do rótulo clique em**Propriedades** e em "font" escolha tamanho **18** e clique em **OK**. Veja:

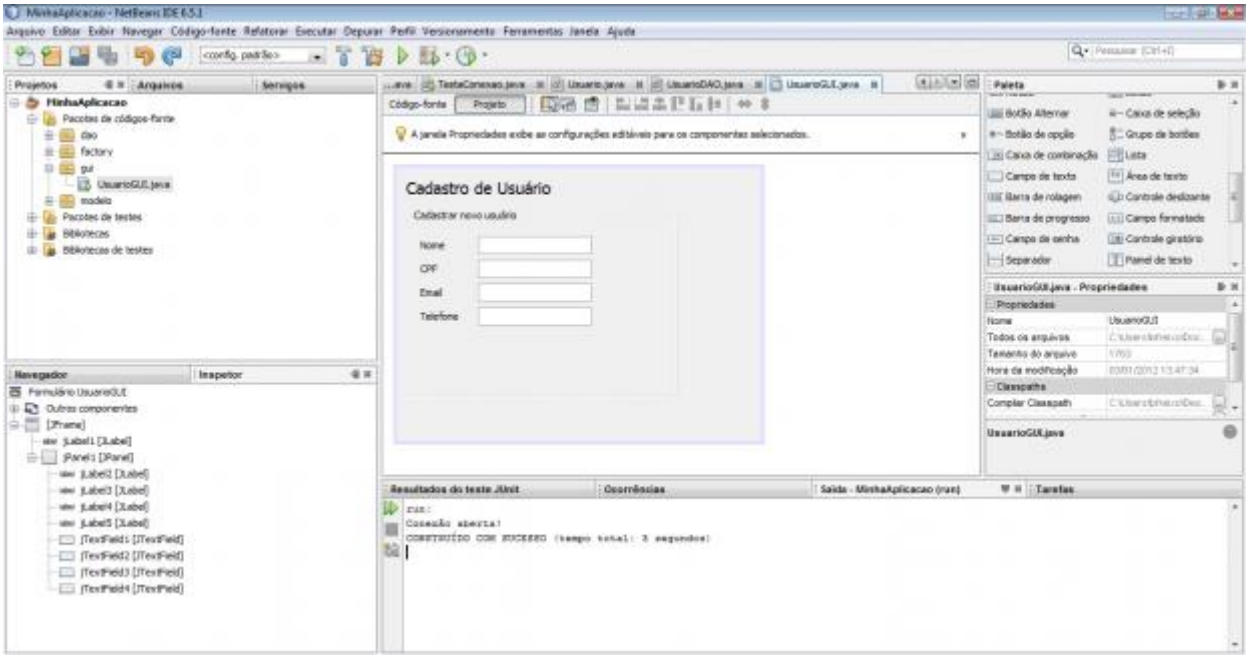


Agora na paleta de componentes, em Contêiners SWING, escolha **Painel** ASM. Clique com o botão direito do mouse e escolha **Propriedades**. Clique em **border > Borda de título**. Intitule "**Cadastrar novo usuário**". Clique em **OK** e depois **fechar**. Veja:

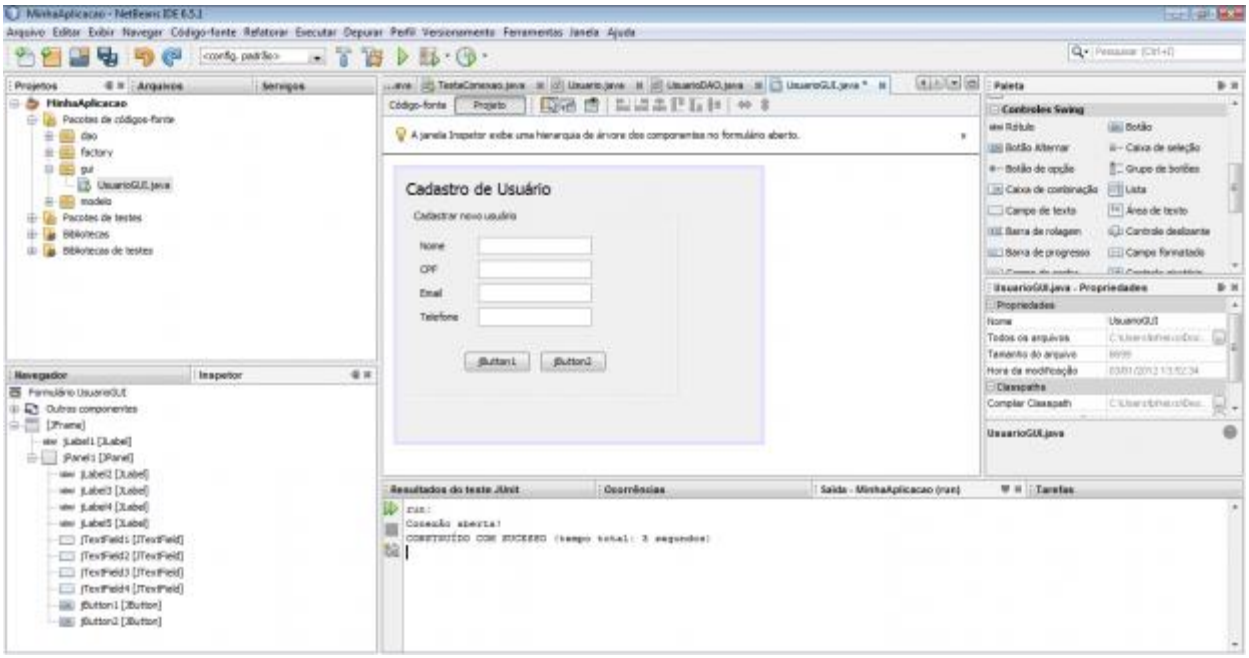


Escreva mais 4 rótulos dentro do painel: **Nome**, **CPF**, **Email**, **Telefone**.

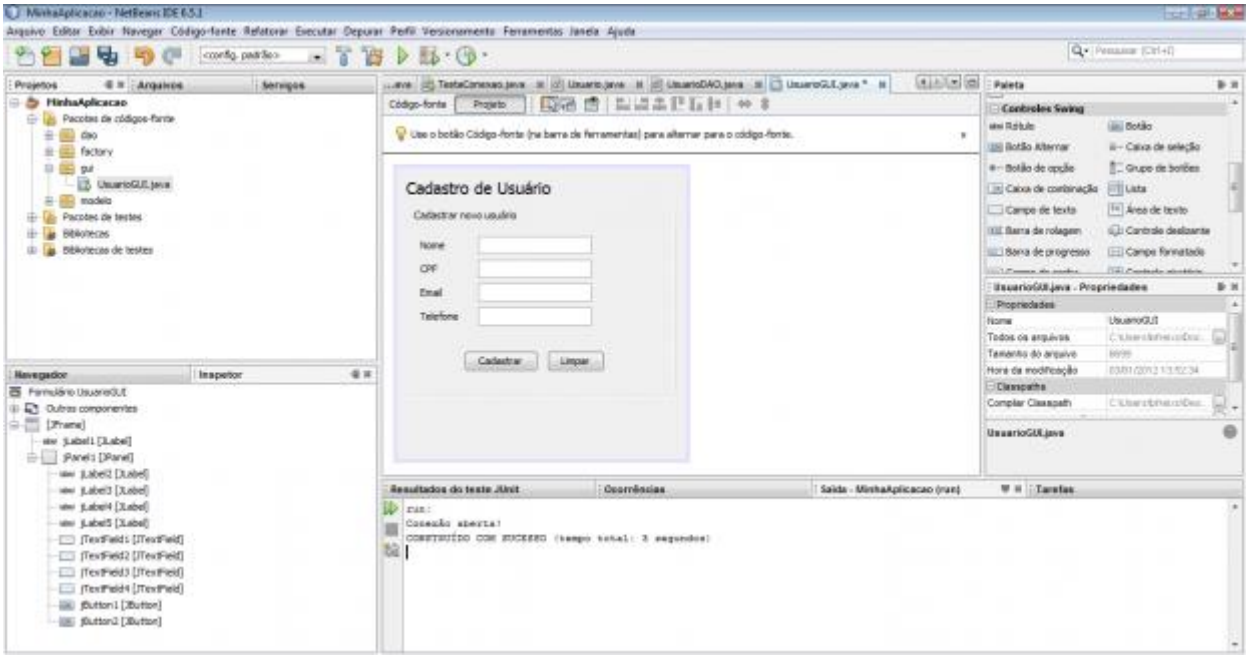
Agora escolha na paleta de componentes 4 **campos de textos** representando de forma respectiva cada um dos rótulos mencionados. Veja:



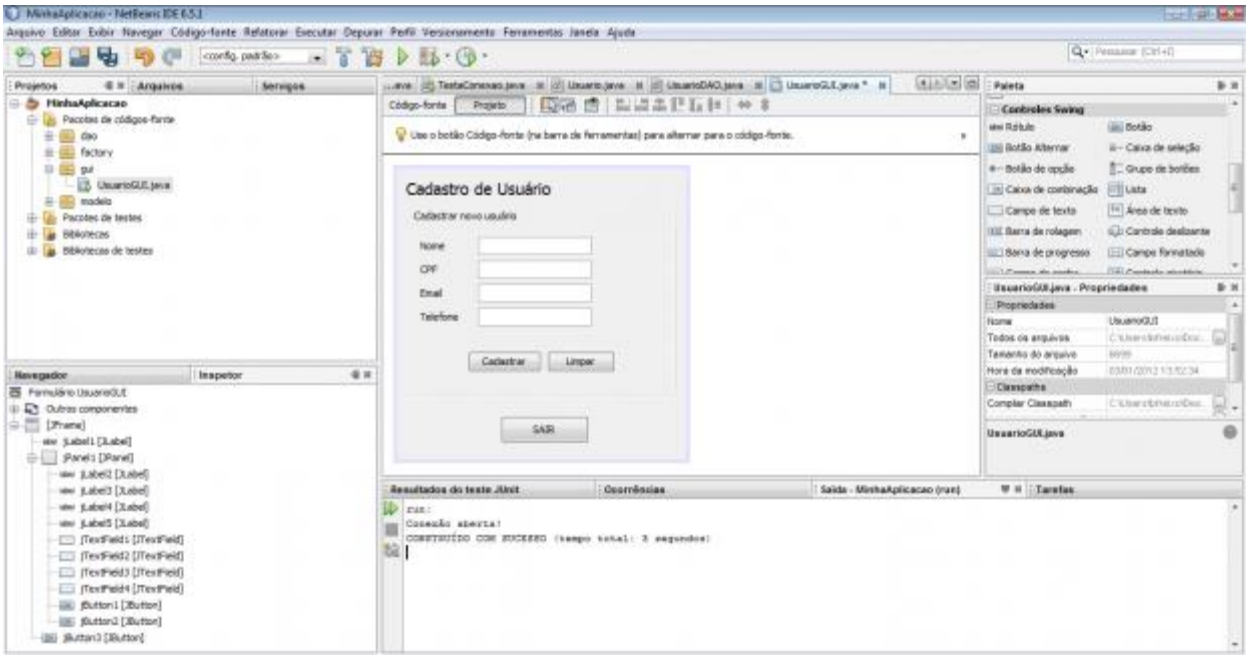
Finalmente vamos criar os botões. Na aba Paleta > Controles SWING, vá até **Botão** ASM. Crie dois botões, conforme mostra a imagem abaixo:



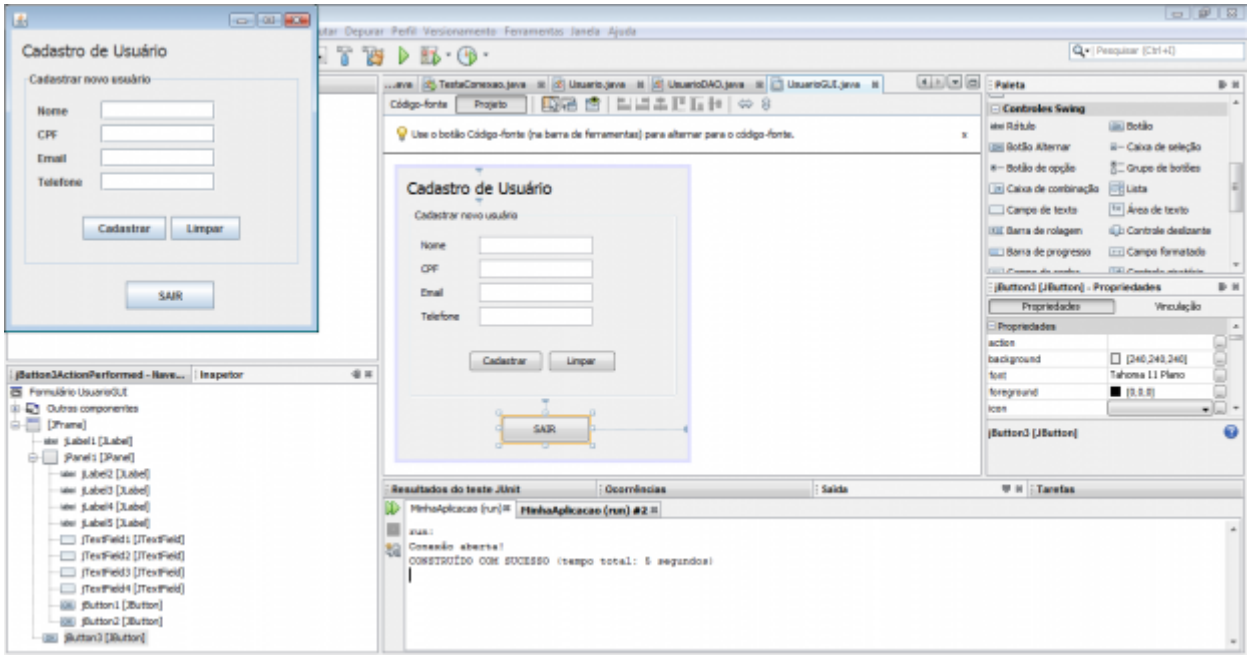
Escreva jButton1 como “**Cadastrar**” e jButton2 como “**Limpar**”. Veja:



Agora, fora do painel, crie o botão **SAIR**. Faça o mesmo processo: sobrescreva jButton3 para SAIR.



Clicando em ALT+F6 temos uma visão geral do projeto em execução:



Passo 7: Evento SAIR

Clique duas vezes no botão "**SAIR**" para criarmos o evento. Na aba Código-Fonte, no método referente a jButton3, isto é, ao botão SAIR, digite:

```
System.exit(0);
```

Este comando fecha a janela em execução. Dê um ALT+F6 e agora clique no botão SAIR. A janela será fechada.

Passo 8: Evento LIMPAR Já é um evento a mais que você cria no sistema.

Agora, na aba Projeto, dê dois cliques em **Limpa**r.
No método jButton2ActionPerformed, na aba Código-fonte, escreva os seguintes scripts:

```
// apaga os dados preenchidos nos campos de texto

jTextField1.setText("");

jTextField2.setText("");

jTextField3.setText("");

jTextField4.setText("");
```

Estes scripts são responsáveis por limpar ou apagar qualquer string escrita pelo usuário em cada um dos 4 campos de texto do formulário.

Passo 9: Evento CADASTRAR

Precisamos criar o principal evento que é literalmente cadastrar o usuário. Para isso, vamos clicar duas vezes no botão "**Cadastrar**" e, na aba Código-fonte, no evento jButton1ActionPerformed ficará assim o código:

```
// instanciando a classe Usuario do pacote modelo e criando seu objeto usuarios

Usuario usuarios = new Usuario();

usuarios.setNome(jTextField1.getText());

usuarios.setCpf(jTextField2.getText());

usuarios.setEmail(jTextField3.getText());

usuarios.setTelefone(jTextField4.getText());



// fazendo a validação dos dados

if ((jTextField1.getText().isEmpty()) || (jTextField2.getText().isEmpty()) || (jTextField3.getText().isEmpty()) || (jTextField4.getText().isEmpty())) {
```

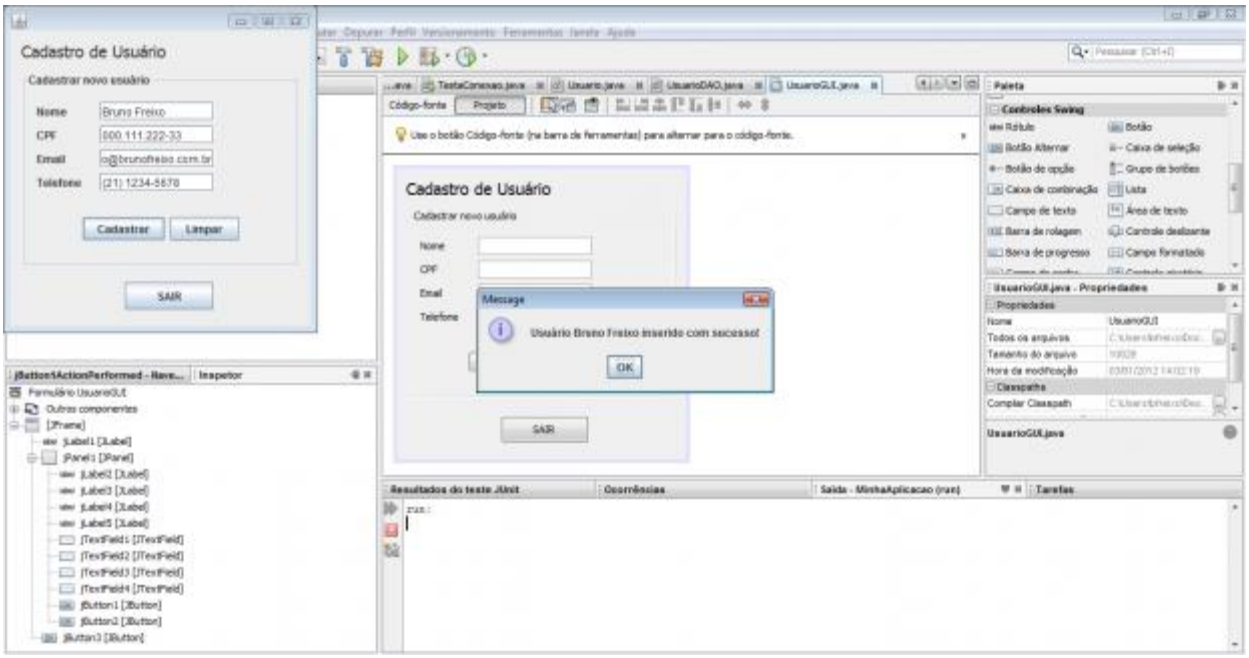
```
  JOptionPane.showMessageDialog(null, "Os campos não podem retornar vazios");
}
else {

   // instanciando a classe UsuarioDAO do pacote dao e criando seu objeto dao
   UsuarioDAO dao = new UsuarioDAO();
   dao.adiciona(usuarios);
   JOptionPane.showMessageDialog(null, "Usuário "+jTextField1.getText()+" inserido com sucesso! ");
}
```

Certamente algumas mensagens de erro aparecerão. Isto porque temos que **importar** no início do código as **classes Usuario** (pacote modelo) e **UsuarioDAO** (pacote dao). Além destas, precisamos importar a **classe JOptionPane**, responsável pelas janelas de validação, aquelas que aparecem dizendo se o usuário foi ou não cadastrado, se os campos estão vazios, etc.

Coloque estas linhas no início do código, abaixo de “package gui”, na aba Código-fonte:

```
import modelo.Usuario;
import dao.UsuarioDAO;
import javax.swing.JOptionPane;
```

Agora sim não aparecerá erro nenhum e o cadastro poderá ser feito.
Faça um teste! Veja:



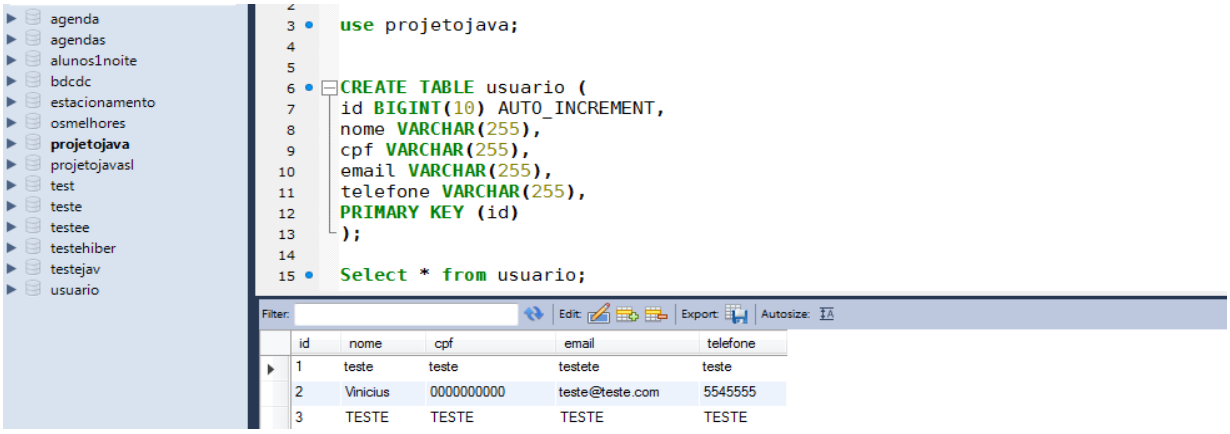
Passo 10: Consulta através do Console do MySQL

Vá até o console do MySQL Server.
Digite:

```
use projetojava;
```

```
select * from usuario;
```

A seguinte tela aparecerá:



Pronto. Se todos os passos foram seguidos corretamente, sua aplicação foi executada com sucesso. O Java não é uma tecnologia fácil de se aprender logo de início, mas depois que você entende os principais conceitos de orientação a objetos (classes, objetos, atributos, métodos, encapsulamento, herança e polimorfismo,

etc.), prática do código limpo (esse que só vem com o tempo utilizando polimorfismo para substituir estruturas de decisão, encapsulamento nos atributos, interfaces para reduzir o acoplamento, e assim por diante...), aí sim o entendimento fica bem mais elucidado.