LOGIN COM JAVA SWING UTILIZANDO NETBEANS E MYSQL

Michel Adriano Medeiros

OBJETIVO

Vamos criar um sistema de login onde a pessoa também possa cadastrar-se caso não tenha um login. O sistema será construído com o Java Swing, JDK 13 e Maven.

Neste ebook uu suponho que você já tenha o ambiente Java configurado e instalado o Netbeans. Acredito também que você saiba criar um projeto maven e saiba trabalhar com o Netbeans.

No blog, tem um artigo de como configurar o ambiente Java https://cursojavanow.com.br/instalacao-do-jdk/

Caso você seja muito iniciante, recomendo você fazer o curso: Java para Iniciantes.

https://www.udemy.com/course/curso-java-para-iniciantes/?referralCode=BCDB27D7DF68341BFEE4

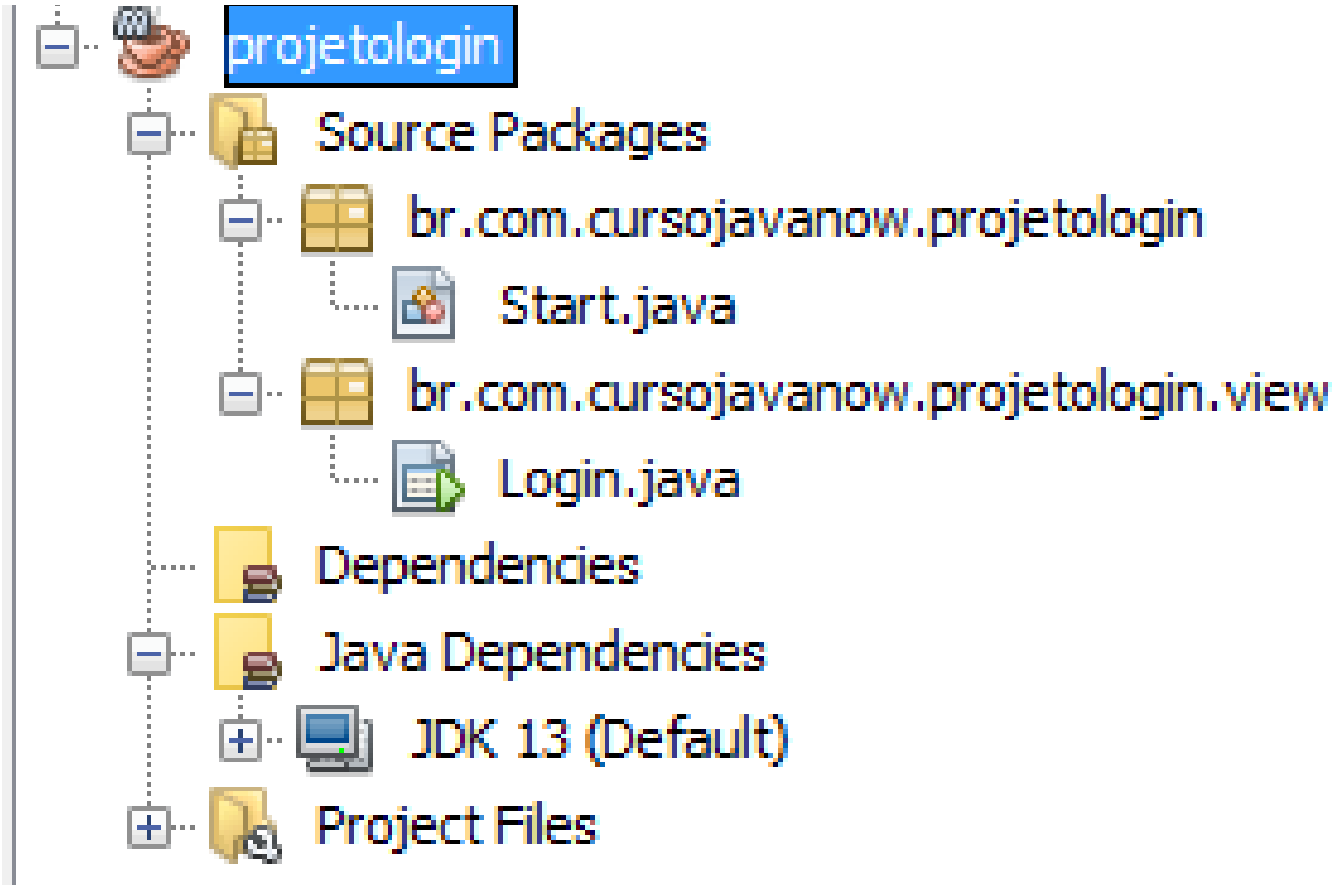
CONFIGURAÇAO DO POM.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <groupId>br.com.micheladrianomedeiros</groupId>
   <artifactId>projetologin</artifactId>
   <version>1.0-SNAPSHOT</version>
   <packaging>jar</packaging>
   <properties>
      <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
      <maven.compiler.source>13</maven.compiler.source>
      <maven.compiler.target>13</maven.compiler.target>
   </properties>
   <build>
      <plugins>
         <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-shade-plugin</artifactId>
            <version>3.2.1</version>

            <executions>
               <!-- Run shade goal on package phase -->
               <execution>
                  <phase>package</phase>
                  <goals>
                     <goal>shade</goal>
                  </goals>
                  <configuration>
                     <transformers>
                        <!-- add Main-Class to manifest file -->
                        <transformer
implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer"
>
                           <mainClass>br.com.micheladrianomedeiros.Start</mainClass>
                        </transformer>
                     </transformers>
                  </configuration>
               </execution>
            </executions>
         </plugin>

      </plugins>
   </build>

   <dependencies>
      <dependency>
         <groupId>mysql</groupId>
         <artifactId>mysql-connector-java</artifactId>
         <version>8.0.18</version>
      </dependency>
   </dependencies>

</project>
```

TELA DE LOGIN

Vou criar as duas telas que teremos no sistema. A tela de login e a de cadastro de usuário.

Escolha a JFrame para criar a tela de login. Eu criei um pacote view para adicionar as telas.



Nas propriedades do JFrame login faça o seguinte:

- marque em propriedade o undecorated
- em propriedade na aba code escolha Generate Center

Próximo passo:

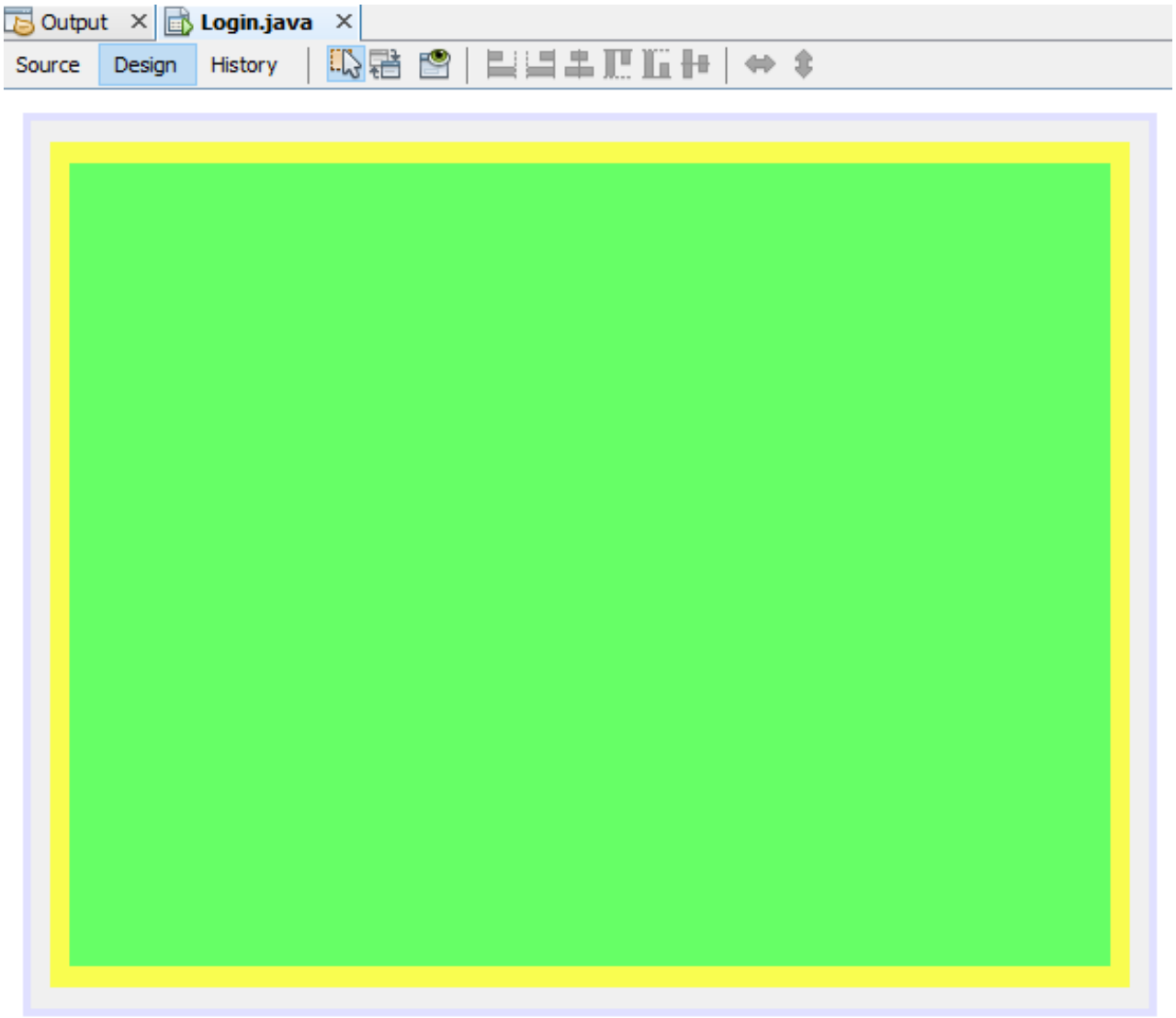
- adicione um JPanel ao JFrame.
- clique com o botão direito do mouse sobre o Jpanel e escolha a opção change variable name. Dê no nome de JPanelBase.
- em propriedades, clique em background e escolha uma cor que você deseja. Eu fui até a aba RGB e adicionei o código em Color Code F9FD50.
- vamos criar um método chamado bordas. Adicione a chamada deste método no início do JFrame login.

O código fica assim:

TELA DE LOGIN

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

    public Login() {
        initComponents();
        bordas();
    }

    public void bordas() {
        Border bordas = BorderFactory
                .createMatteBorder(2, 2, 2, 2, new Color(0x85ef47));
        JPanelBase.setBorder(bordas);
    }
```

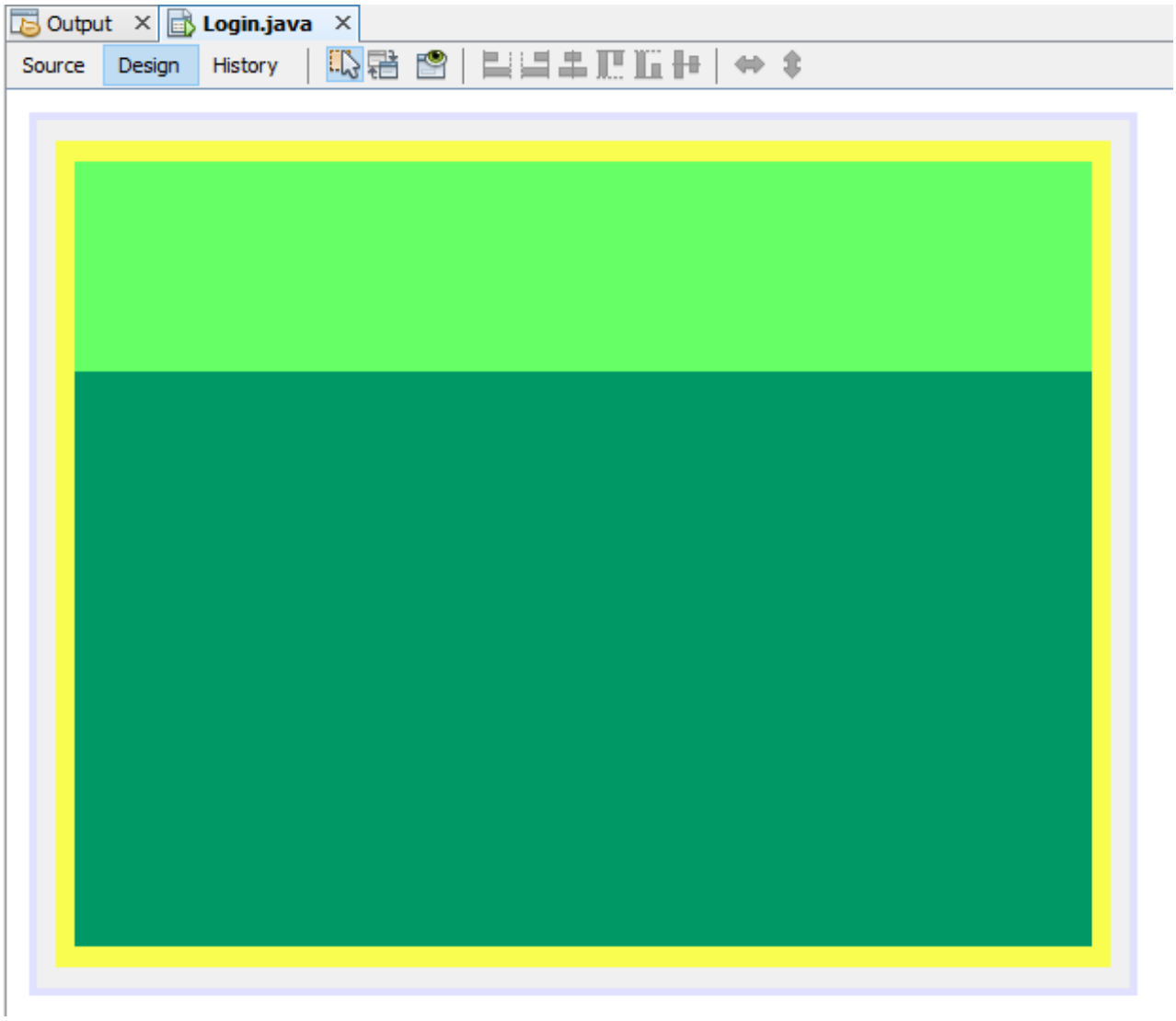
- adicione outro JPanel com o nome de JPanelBase1 sobre o JPanelBase. Não cubra totalmente o JPanelBase, deixe as bordas aparecendo.
- mude a cor de fundo do JPanelBase1. Vai ficar desta maneira:



TELA DE LOGIN

- adicione um JPanel sobre o JPanelBase1. Mude o nome do JPanel para JPanelAcesso.

- não cubra totalmente até o topo o JPanelBase1. Mude a cor de fundo do JPanelAcesso. Veja como deve ficar:



Quando eu escolhi no JFrame a opção undecorated, a janela padrão não aparece. Na janela padrão é possível você colocar o botão minimizar, maximizar e fechar.

Já que escolhi este caminho, vou ter que colocar um meio de fechar e minimizar esta janela.

- adicione ao JPanelBase1 dois JLabel.

- um deles chame de JLabelFechar e o outro JLabelMinimizar.

- no JLabelMinimizar escolhi a font 48 e negrito(bold). Em text coloque o símbolo - (menos).

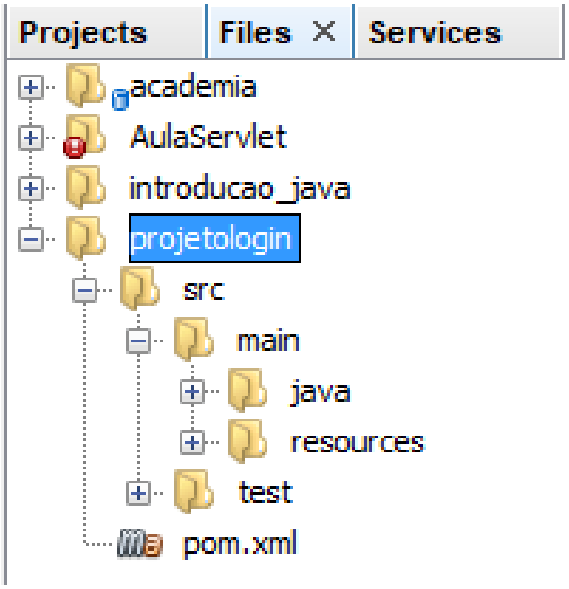
- no JLabelFechar escolhi a font 24 e negrito(bold). Em text coloque a letra x.

TELA DE LOGIN

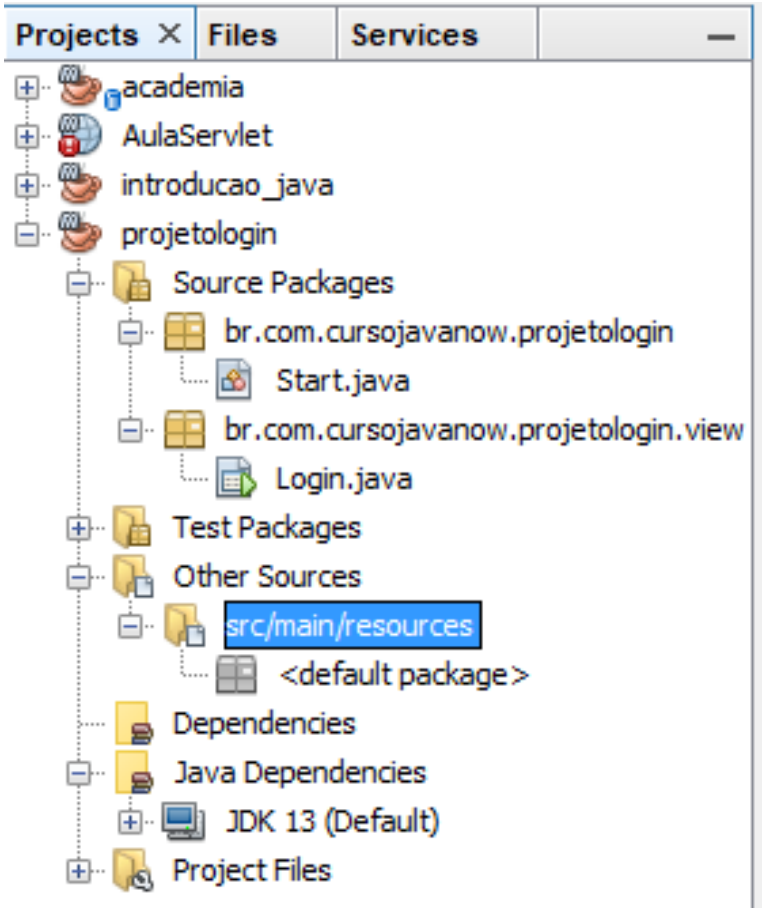
- coloque o JLabelMinimizar e jLabelFechar no canto superior direito do JPanelBase1.

- adicione mais um JLabel ao JPanelBase1. Neste JLabel vou usar para colocar uma imagem. Dê o nome de JLabelImagemTopo.

- no meu projeto não tem a pasta resources, que é onde quero guardar imagens. Para aparecer esta pasta fui até a aba files e adicionei a pasta resources dentro do meu projeto.



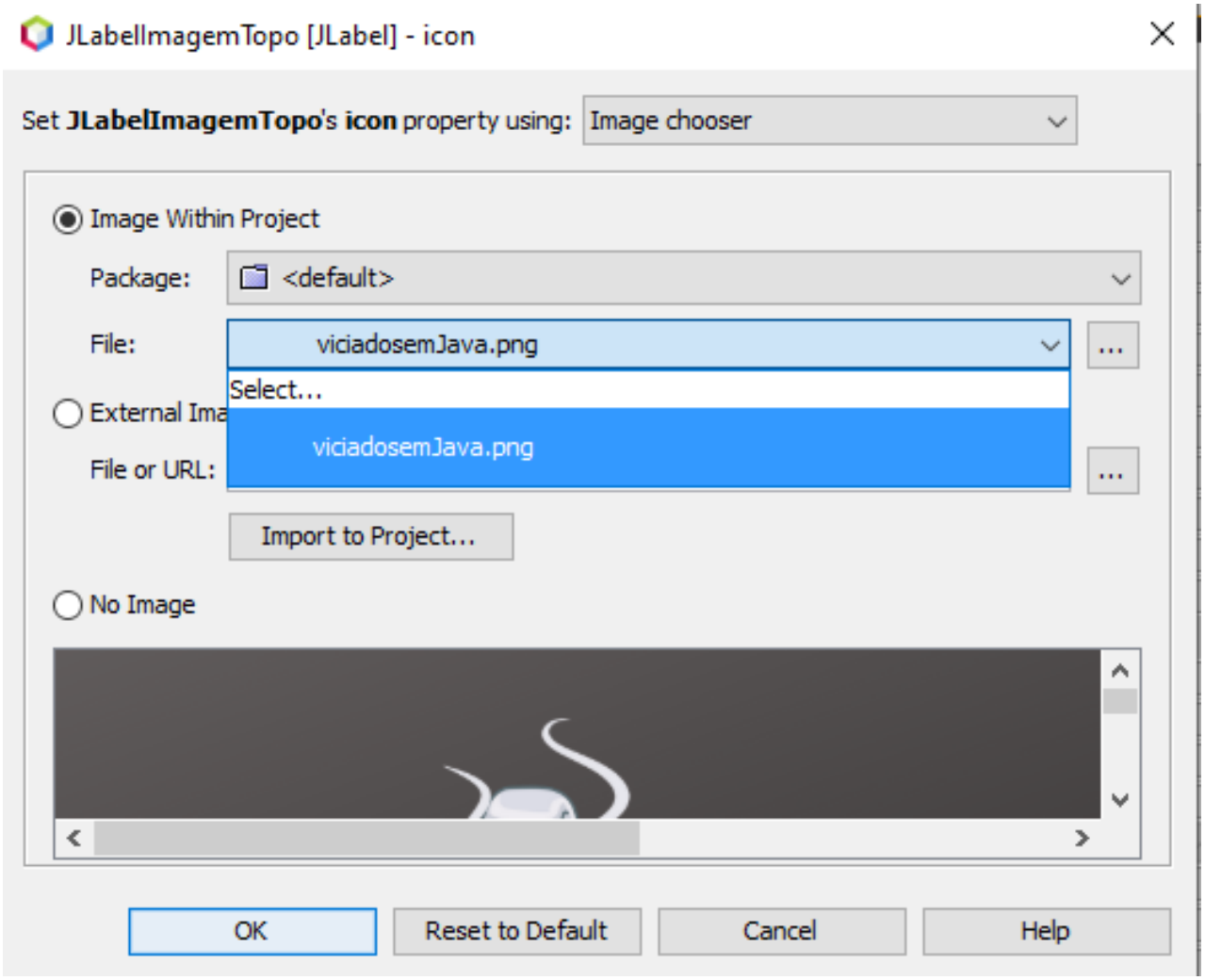
Agora sim a pasta resources aparece no projeto.



TELA DE LOGIN

Agora é só escolher imagens de sua preferência e adicionar neste local. Você pode copiar e colar uma imagem no pacote que ela será transferida para a pasta ou arrastar uma imagem até este pacote.

- entre em propriedades do JLabelImagemTopo.
- em text não esqueça de apagar qualquer coisa que esteja escrito neste campo.
- em icon clique no botão ...
- quando clicar em select, você verá as imagens que estão no pacote resources. Escolha a que você desejar.



TELA DE LOGIN

Faça algo parecido com isto. Veja os botões minimizar e fechar para você ver como ficou.

Vamos colocar bordas no botão minimizar e fechar. Adicione o código em public login().

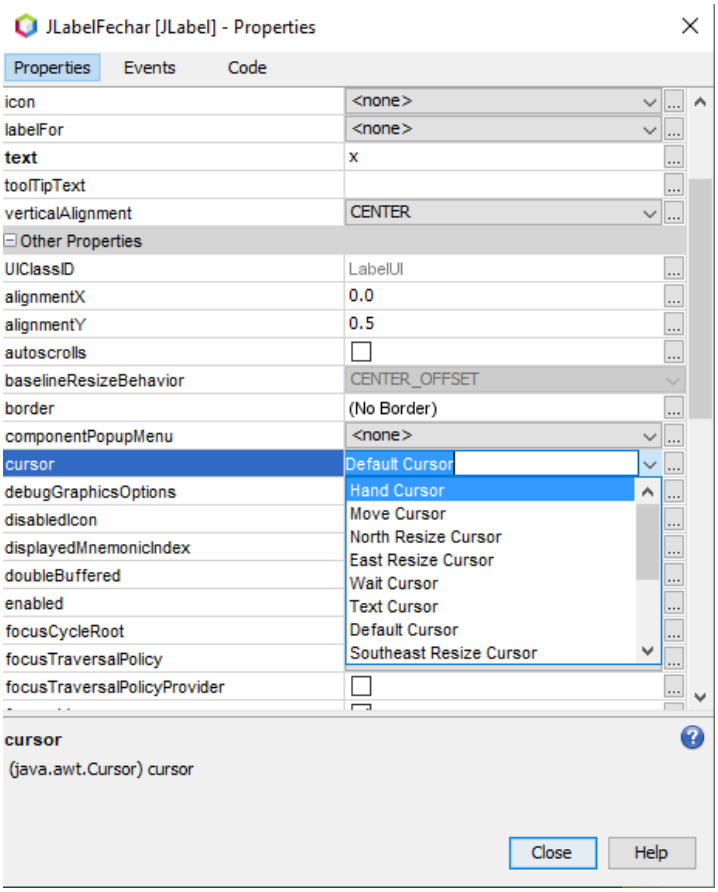
```
public class Login extends javax.swing.JFrame {

    public Login() {
        initComponents();
        bordas();

        Border jlb = BorderFactory
                .createMatteBorder(2, 2, 2, 2, new Color(0x207dff));
        JLabelFechar.setBorder(jlb);
        JLabelMinimizar.setBorder(jlb);
    }
```

TELA DE LOGIN

Adicione dois JLabel ao JPanelAcesso e faça o mesmo procedimento realizado no JLabelImagemTopo.

Os tamanhos utilizados nas imagens são 50x50. Se precisar redimensionar uma imagem utilize o este site: https://resizeimage.net/. Faça algo parecido com isto.

Entre em propriedades no JLabelMinimizar e JLabelFechar e mude em cursor para Hand Cursor.

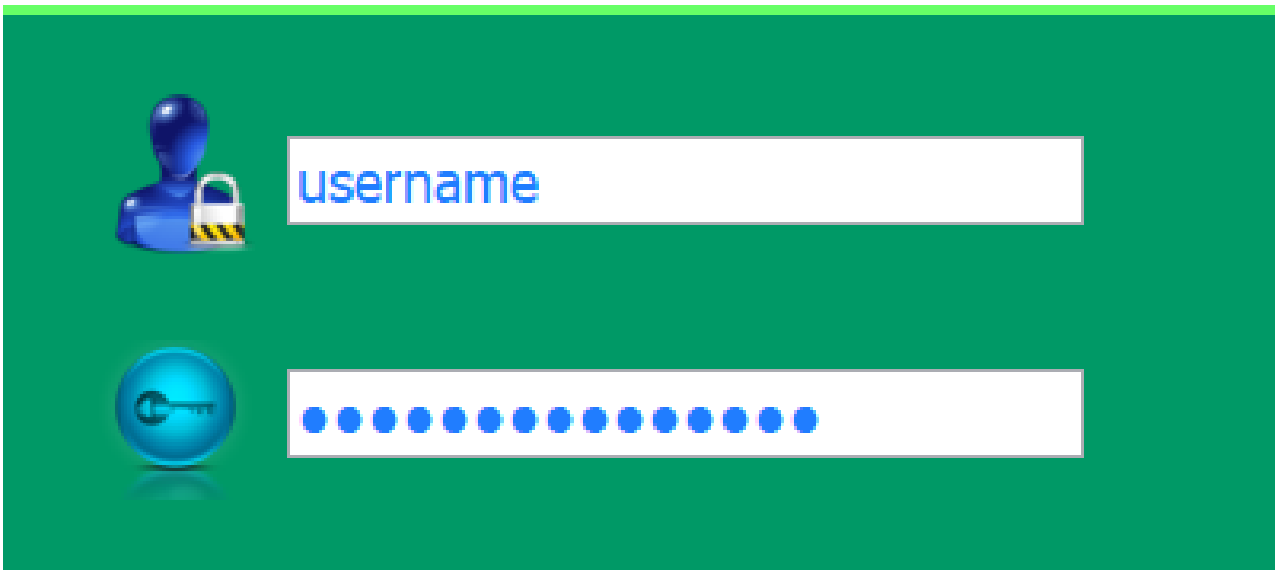


TELA DE LOGIN

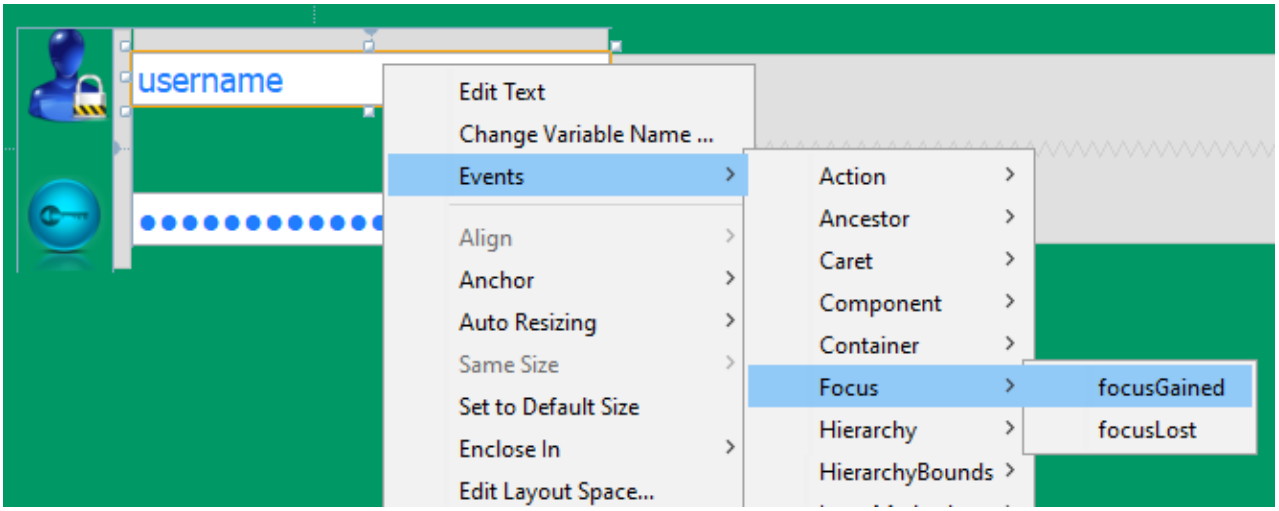
Adicione um JTextField para o usuário digitar o login. Mude o nome do JTextField para JTFUserName. Em text digite username. Mude a Font para 18 e na propriedade foreground mude a cor para azul (o código exato da cor que estou utilizando é 207dff).

Adicione um JPasswordField para o usuário digitar a senha. Mude o nome do JPasswordField para JTFPassword. Mude a Font para 18 e na propriedade foreground mude a cor para azul (o código exato da cor que estou utilizando é 207dff).

Faça algo do tipo:



Clique com o botão direito do mouse no JTFUserName e escolha o focusGainded.



No método criado digite o seguinte código:

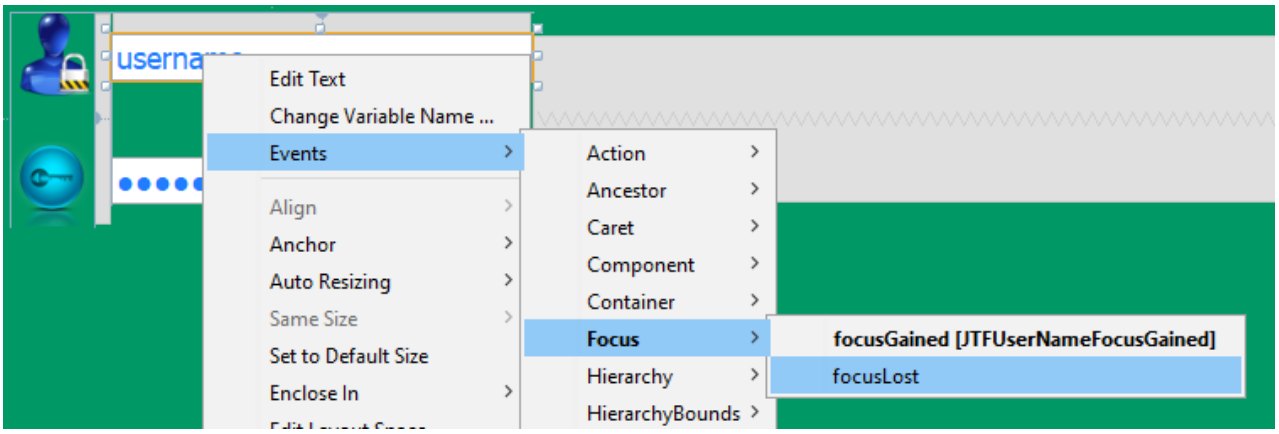
TELA DE LOGIN

```
if (JTFUserName.getText().trim().toLowerCase().equals("username")) {
    JTFUserName.setText("");
    JTFUserName.setForeground(new Color(0x207dff));
}
```

Vamos colocar bordas no JTFUserName e JTFPassword, adicione o código no método bordas.

```
Border bordasJTF = BorderFactory
        .createMatteBorder(1, 5, 1, 1, new Color(0x207dff));
JTFUserName.setBorder(bordasJTF);
JTFPassword.setBorder(bordasJTF);
```

Clique com o botão direito do mouse em JTFUserName e escolha o focusLost.



Digite o seguinte código no método criado.

```
if (JTFUserName.getText().trim().equals("")
        || JTFUserName.getText().trim().toLowerCase()
                .equals("username")) {
    JTFUserName.setText("username");
    JTFUserName.setForeground(new Color(0x207dff));
}
```

Vamos fazer para o JTFPassword o método focusGained. Depois digite o seguite código:

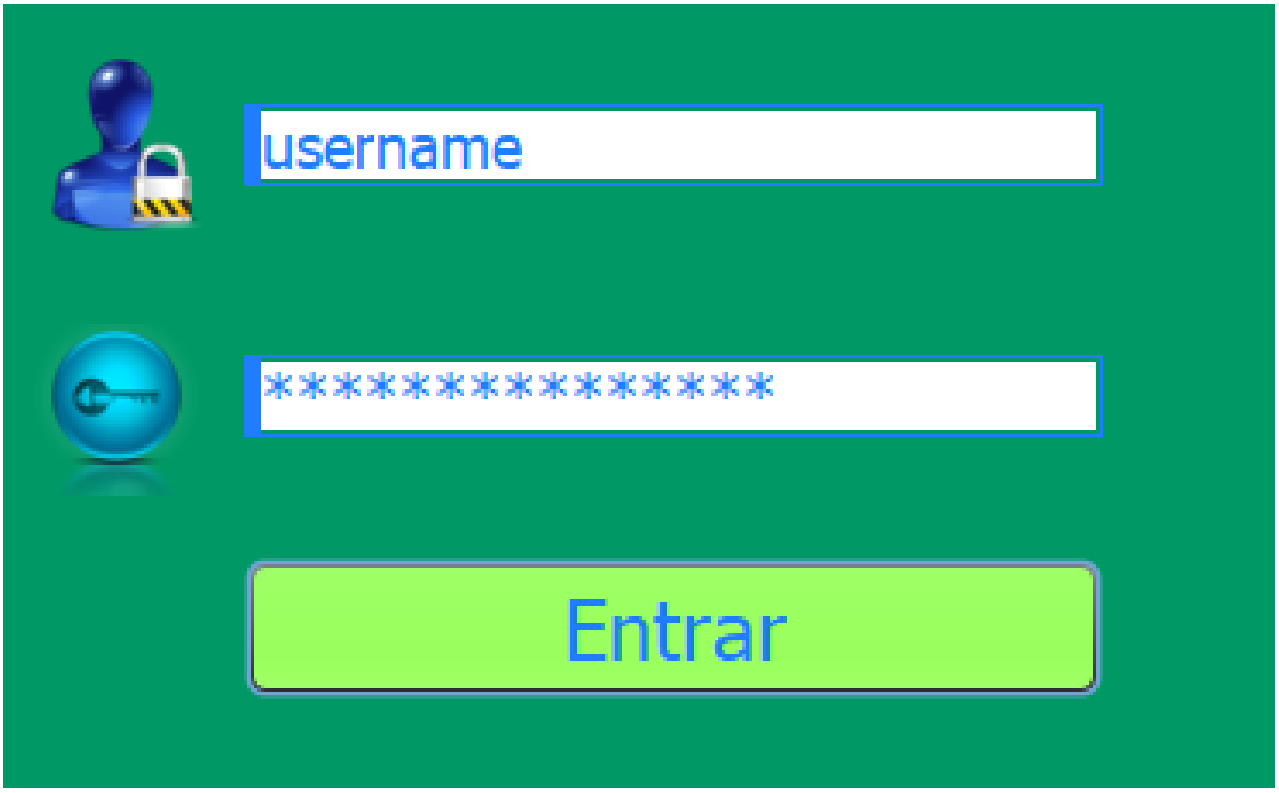
```
JTFPassword.setText("");
JTFPassword.setForeground(new Color(0x207dff));
```

TELA DE LOGIN

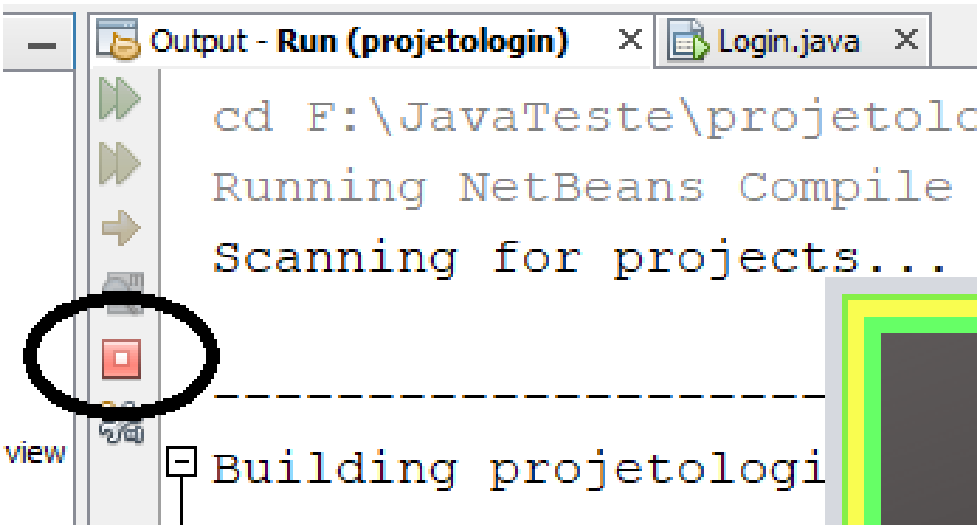
Adicione um JButton para fazer o login no sistema. Mude o nome para JBLogin. Mude a font para 24, background color verde claro (código da cor 85ef47) e foreground azul (código da cor 207dff). Em cursor mude para a opção Hand Cursor. Em text digite Entrar.

Veja como ficou na imagem. Note também estiquei o botão para ficar do mesmo tamanho que os campos de login e password.

Para esticar o botão basta selecionar a borda do botão e esticar. Para ver o botão verde como na imagem temos que executar o programa. Então, execute e veja como está.



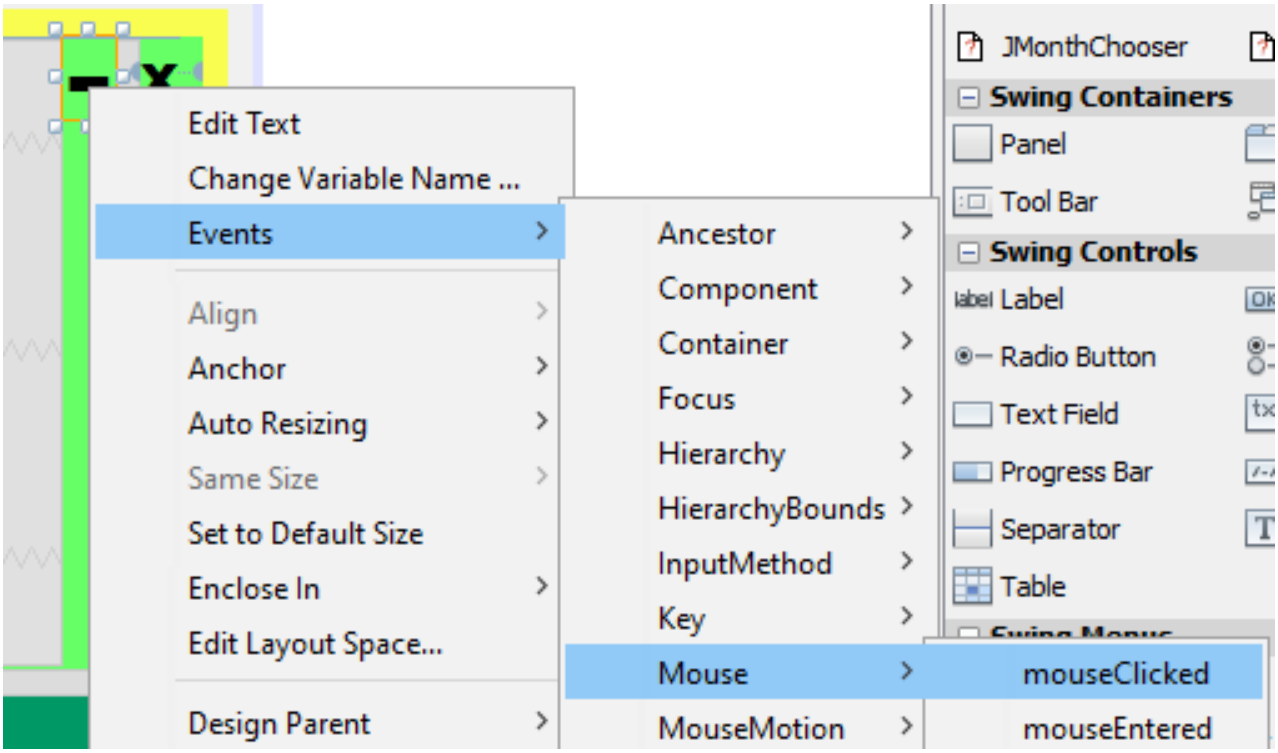
O botão fechar do sistema não está funcionando, então para fechar o programa clique no botão vermelho.



TELA DE LOGIN

Vamos fazer funcionar o botão minimizar e fechar do login funcionar.

Clique como o botão direito do mouse no botão JLabelMinizar e escolha mouseClicked.



No método criado digite o código:

```
this.setState(JFrame.ICONIFIED);
```

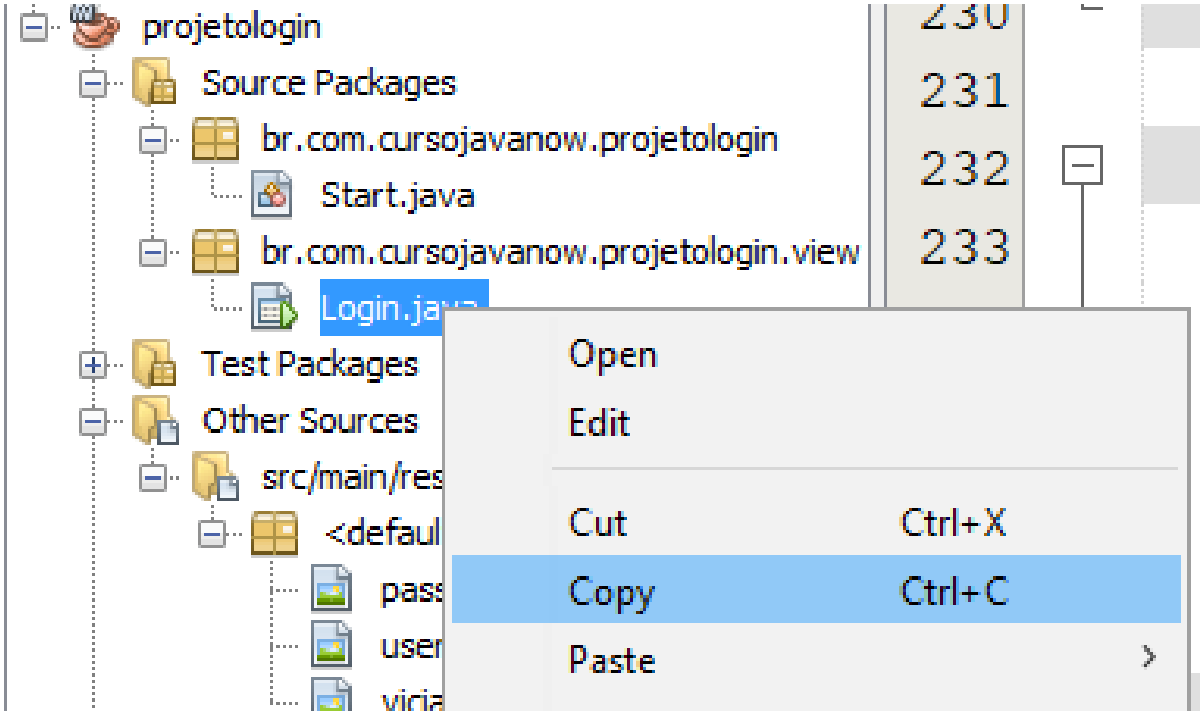
Faça o mesmo procedimento para o JLabelFechar. Digte o código:

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

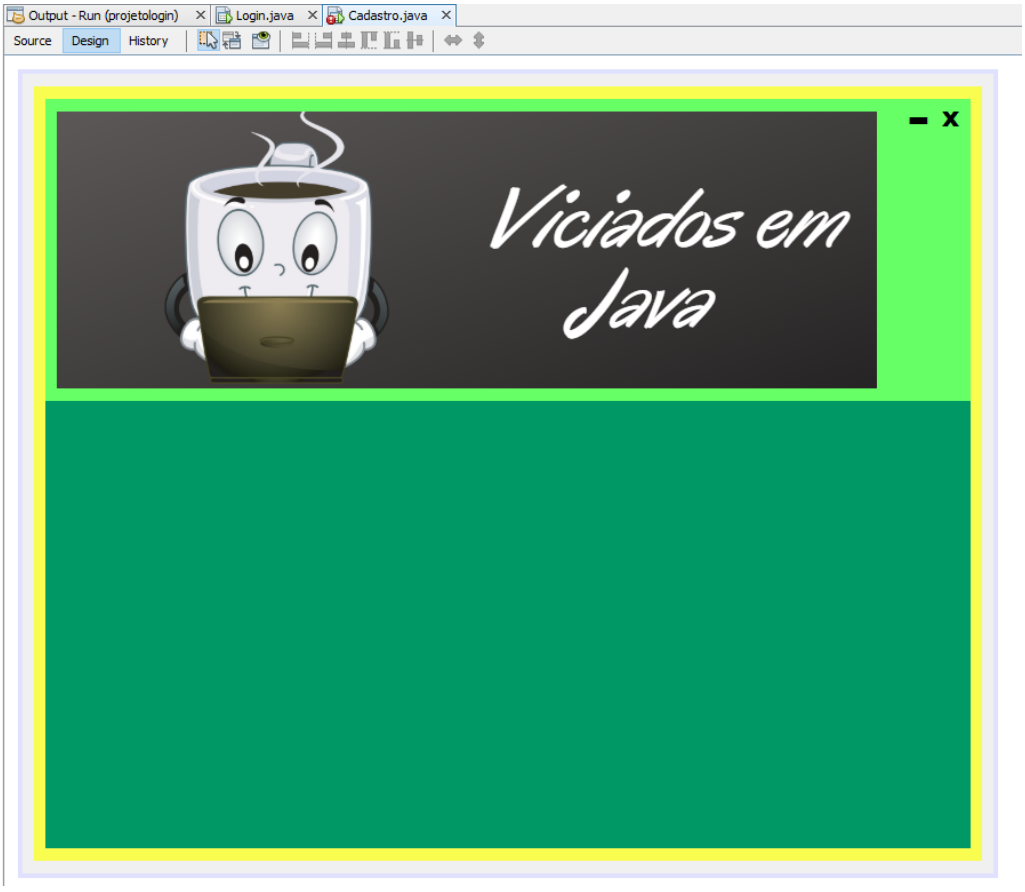
TELA DE CADASTRO DE USUÁRIO

Vamos fazer a outra tela que é a cadastro de usuário, para poder fazer o login no sistema.

Para pegar toda a estrutura da tela do login vou clicar com o botão direito do mouse na tela login e escolher copiar.



E depois colar no pacote view e dar o nome para esta nova tela de Cadastro. Na tela cadastro remova os itens que não vamos utilizar. Deixe como está na figura a seguir.



TELA DE CADASTRO DE USUÁRIO

Depois que você remover os elementos que não for utilizar alguns erros no código irão aparecer. Basta apagar as linhas que estiverem dando erro.

Os dados que serão preenchidos para um novo cadastro serão: nome completo, username, password, telefone, gênero e imagem. Com referência que fizemos na tela de login, você com certeza consegue construir esta tela.

Sua tela de cadastro de usuário deve ficar parecida com esta.

O tamanho da imagem que estou utilizando para o sem imagem é de 225x225.

Agora que a tela de cadastro de usuário está pronta, temos que colocar na tela de login uma maneira de acessar a tela de cadastro de usuário.

Coloquei uma nova JLabel escrito novo cadastro na tela de login para podermos mudar de tela e fazer o cadastro.

TELA DE CADASTRO DE USUÁRIO

O código para mudar para a nova tela fica no mouseClicked, já fizemos este procedimento no botão fechar e minimizar a janela. Código para mudar de tela.

```
private void JLBNovoCadastroMouseClicked(java.awt.event.MouseEvent evt) {
    Cadastro c = new Cadastro();
    c.setVisible(true);
    dispose();
}
```

Agora tenho que fazer a mesma coisa na tela de cadastro de usuário, vou colocar uma JLabel e no mouseClicked o código para voltar para a tela de login.

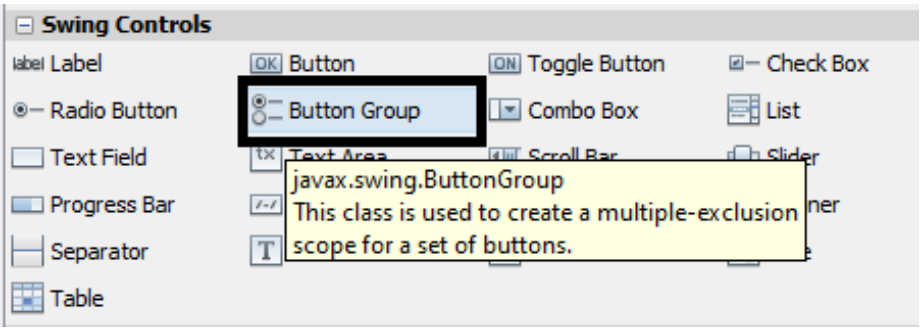
Outra coisa que foi esquecida na tela de cadastro foi adicionar um botão para finalizar o cadastro do novo usuário. A tela ficou assim:

O código do JLabel Voltar fica:

```
private void JLBVoltarLoginMouseClicked(java.awt.event.MouseEvent evt) {
    Login c = new Login();
    c.setVisible(true);
    dispose();
}
```

Na tela de cadastro de usuário o gênero dá para escolher feminino e masculino, temos que dizer ao sistema que só deve ser escolhido um gênero.

Para isto escolha o elemento ButtonGroup e arraste para dentro da tela cadastro de usuário.



TELA DE CADASTRO DE USUÁRIO

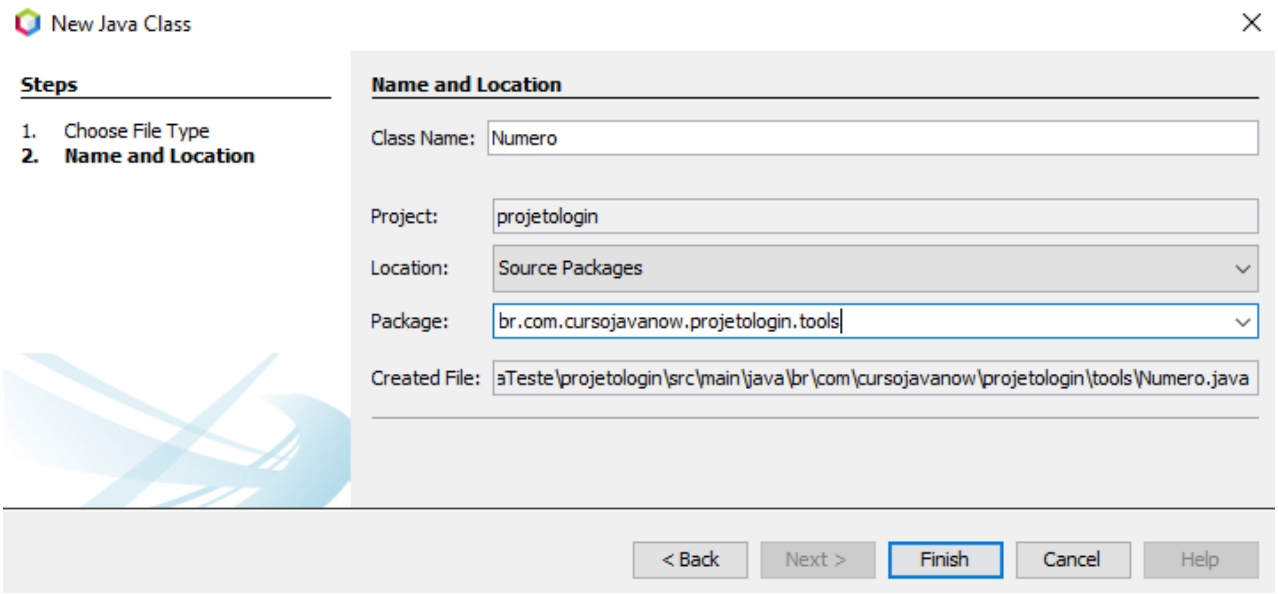
Quando você arrasta o ButtonGroup para dentro da tela, ele não fica exposto como os outros elementos.

No construtor da tela cadastro, vamos adicionar ao ButtonGroup os dois checkbox, feminino e masculino. O código fica assim:

Sendo que o JCBFeminino e JCBMasculino são nomes dos checkboxes adicionados para escolher o gênero.

```
public Cadastro() {
    initComponents();
    bordas();

    buttonGroup1.add(JCBFeminino);
    buttonGroup1.add(JCBMasculino);

    Border jlb = BorderFactory
            .createMatteBorder(2, 2, 2, 2, new Color(0x207dff));
    JLabelFechar.setBorder(jlb);
    JLabelMinimizar.setBorder(jlb);
}
```

O próximo passo é permitir somente números no campo telefone. Vou criar um novo pacote chamado tools e uma nova classe chamada Numero.

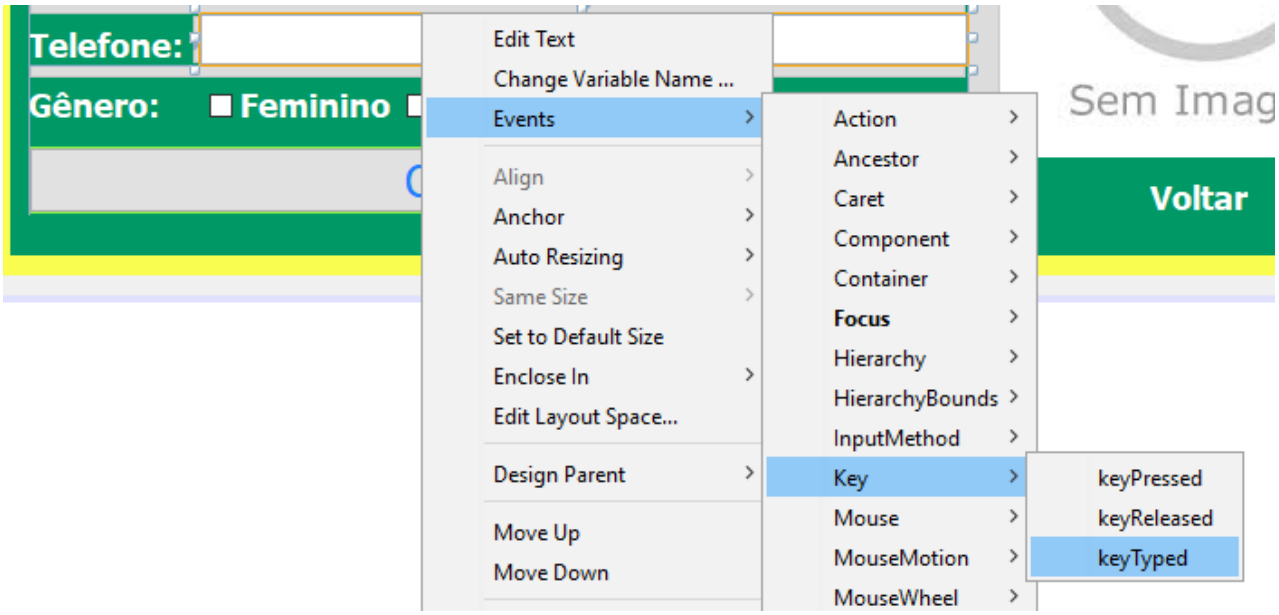


Na classe Numero vou colocar um método chamada justNumbers. Veja o código.

TELA DE CADASTRO DE USUÁRIO

```
package br.com.cursojavanow.projetologin.tools;

import java.awt.event.KeyEvent;

/**
 *
 * @author michel adriano medeiros
 */
public class Numero {

    public static void justNumbers(KeyEvent evt) {
        char vchar = evt.getKeyChar();
        if (!(Character.isDigit(vchar)
                || vchar == KeyEvent.VK_BACK_SPACE
                || vchar == KeyEvent.VK_DELETE)) {
            evt.consume();
        }
    }
}
```

No campo telefone escolha o evento keyTyped.



O código dentro deste evento é:

```
    private void JTFTelefoneKeyTyped(java.awt.event.KeyEvent evt) {
        Numero.justNumbers(evt);
    }
```

Com isto, o campo telefone só vai reconhecer as teclas de números, backspace e delete.

Outra coisa que temos que fazer relativo a esta tela é sobre a imagem. Vamos fazer a seguinte solução: quando o usuário clicar sobre a imagem ele poderá escolher uma outra.

TELA DE CADASTRO DE USUÁRIO

Vamos criar a classe TrabalhandoComImagem que no pacote tools. Com o seguinte código:

```
package br.com.cursojavanow.projetologin.tools;

import java.awt.AlphaComposite;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.StandardCopyOption;
import java.util.UUID;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.imageio.ImageIO;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JFileChooser;
import javax.swing.filechooser.FileNameExtensionFilter;

/**
 *
 * @author michel adriano medeiros
 */
public class TrabalhandoComImagem {

    public static final String FOTOS_DOS_USUARIOS = "F:\\JavaTeste\\alunos/";

    public static JFileChooser arquivoDeImagem() {
        JFileChooser dialogo = new JFileChooser();
        String files[] = {"jpg", "png"};
        FileNameExtensionFilter filtro = new FileNameExtensionFilter(
                "Arquivos de Foto", files
        );
        dialogo.setFileFilter(filtro);
        return dialogo;
    }

    public static String moverArquivo(File arquivo) {
        String uuid = null;
        try {
            UUID idOne = UUID.randomUUID();
            uuid = idOne.toString();
            File dest = new File(FOTOS_DOS_USUARIOS + uuid + ".jpg");
            Files.copy(arquivo.toPath(), dest.toPath(),
                    StandardCopyOption.REPLACE_EXISTING);
        } catch (IOException ex) {
            Logger.getLogger(TrabalhandoComImagem.class.getName())
                    .log(Level.SEVERE, null, ex);
        }
        return uuid;
    }
```

TELA DE CADASTRO DE USUÁRIO

```
    public static Icon retornaIconeRedimensionado(File newName) {
        BufferedImage bi = returnImageResize(
                newName.getAbsolutePath(), 225, 225);
        ImageIcon icon = new ImageIcon(bi);
        return icon;
    }

    public static BufferedImage returnImageResize(String localFile, int IMG_WIDTH,
int IMG_HEIGHT) {
        BufferedImage resizeImagePng = null;
        try {

            BufferedImage originalImage = ImageIO.read(new File(localFile));
            int type = originalImage.getType() == 0 ? BufferedImage.TYPE_INT_ARGB :
originalImage.getType();

            BufferedImage resizeImageJpg = resizeImage(originalImage, type,
IMG_WIDTH, IMG_HEIGHT);
            ImageIO.write(resizeImageJpg, "jpg", new File(localFile));

            resizeImagePng = resizeImage(originalImage, type, IMG_WIDTH,
IMG_HEIGHT);
            ImageIO.write(resizeImagePng, "png", new File(localFile));

            BufferedImage resizeImageHintJpg = resizeImageWithHint(originalImage,
type, IMG_WIDTH, IMG_HEIGHT);
            ImageIO.write(resizeImageHintJpg, "jpg", new File(localFile));

            BufferedImage resizeImageHintPng = resizeImageWithHint(originalImage,
type, IMG_WIDTH, IMG_HEIGHT);
            ImageIO.write(resizeImageHintPng, "png", new File(localFile));

        } catch (IOException e) {
            System.out.println(e.getMessage());
        }
        return resizeImagePng;

    }

    private static BufferedImage resizeImage(BufferedImage originalImage, int
type, int IMG_WIDTH, int IMG_HEIGHT) {
        BufferedImage resizedImage = new BufferedImage(IMG_WIDTH,
IMG_HEIGHT, type);
        Graphics2D g = resizedImage.createGraphics();
        g.drawImage(originalImage, 0, 0, IMG_WIDTH, IMG_HEIGHT, null);
        g.dispose();

        return resizedImage;
    }
```

TELA DE CADASTRO DE USUÁRIO

```
private static BufferedImage resizeImageWithHint(BufferedImage originalImage, int
type, int IMG_WIDTH, int IMG_HEIGHT) {

        BufferedImage resizedImage = new BufferedImage(IMG_WIDTH, IMG_HEIGHT,
type);
        Graphics2D g = resizedImage.createGraphics();
        g.drawImage(originalImage, 0, 0, IMG_WIDTH, IMG_HEIGHT, null);
        g.dispose();
        g.setComposite(AlphaComposite.Src);

        g.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
                RenderingHints.VALUE_INTERPOLATION_BILINEAR);
        g.setRenderingHint(RenderingHints.KEY_RENDERING,
                RenderingHints.VALUE_RENDER_QUALITY);
        g.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
                RenderingHints.VALUE_ANTIALIAS_ON);

        return resizedImage;
    }

}
```

Na imagem do usuário adicione no evento MouseClicked o seguinte código:

```
    private void JLBImagemUsuarioMouseClicked(java.awt.event.MouseEvent evt) {
        try {
            JFileChooser dialogo = TrabalhandoComImagem.arquivoDeImagem();
            int resposta = dialogo.showOpenDialog(this);
            if (resposta == JFileChooser.APPROVE_OPTION) {
                File arquivo = dialogo.getSelectedFile();
                UUID idOne = UUID.randomUUID();
                uuid = idOne.toString();
                uuid = TrabalhandoComImagem.moverArquivo(arquivo);
                JLBImagemUsuario.setIcon(
                        TrabalhandoComImagem
                                .retornaIconeRedimensionado(arquivo));
            }
        } catch (HeadlessException e) {
            System.out.println("ERRO JLB_FotoMouseClicked " + e.getMessage());
        }
    }
```

TELA DE CADASTRO DE USUÁRIO

Adicione uma variável de classe desta maneira (String uuid;) porque no código uuid = idOne.toString(); vamos precisar desta variável.

Não sabe o que é variável de classe? Acesse: https://micheladrianomedeiros.com.br/blog/aprendendo-java-para-certificacao-8/

Outra coisa que você tem que ver, esse código da classe TrabalhandoComImagem:

```
public static final String FOTOS_DOS_USUARIOS = "F:\\JavaTeste\\alunos/";
```

No meu caso as imagens que eu escolher para o usuário serão salvas em F:\\JavaTeste\\alunos/. Escolha um local na sua máquina ou até em uma rede.

Se você executar o programa e escolher uma imagem, você verá que a imagem irá substituir a imagem padrão.

E esta imagem também será enviada para o local que você escolheu, no meu caso F:\\JavaTeste\\alunos/.

Você vai notar também que a imagem vai ser renomeado com um código UUID.

Não sabe o que é UUID? Acesse: https://cursojavanow.com.br/o-que-e-uuid-porque-usa-lo/

TELA DE CADASTRO DE USUÁRIO

Imagem renomeada com o código UUID. Note que o local onde a imagem está, é o mesmo do código F:\\JavaTeste\\alunos/

BANCO DE DADOS DO SISTEMA

Agora vamos partir para a criação das tabelas no banco de dados.

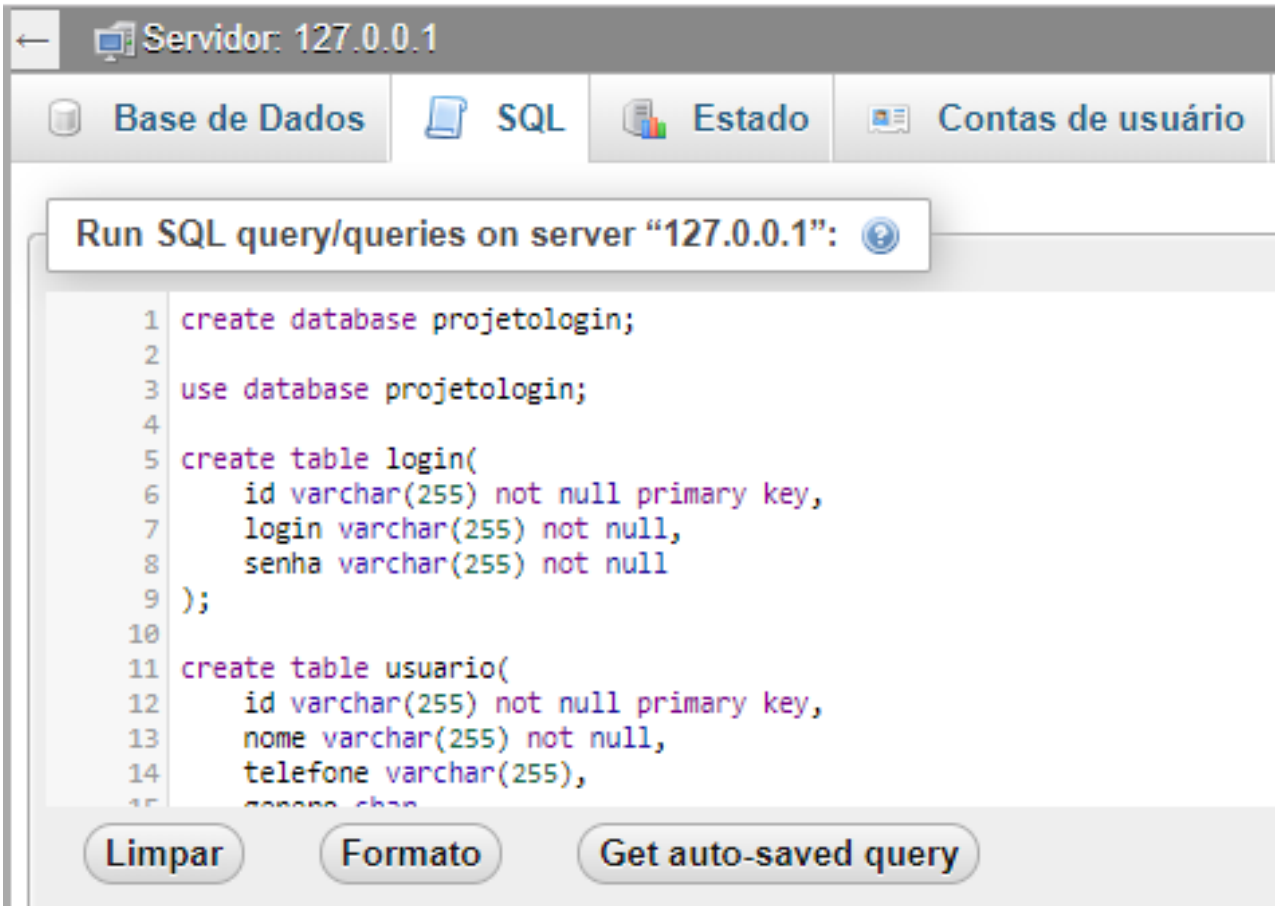
Eu vou utilizar o Xampp, caso não saiba como instalar o Xampp acesse: https://cursojavanow.com.br/3-aulas-bd/

```
create database projetologin;

use database projetologin;

create table login(
    id varchar(255) not null primary key,
    login varchar(255) not null,
    senha varchar(255) not null
);

create table usuario(
    id varchar(255) not null primary key,
    nome varchar(255) not null,
    telefone varchar(255),
    genero char,
    foto varchar(255)
);

create table login_usuario(
    id varchar(255) not null primary key,
    login varchar(255) not null,
    usuario varchar(255) not null,
    foreign key(login) references login(id),
    foreign key(usuario) references usuario(id)
);
```

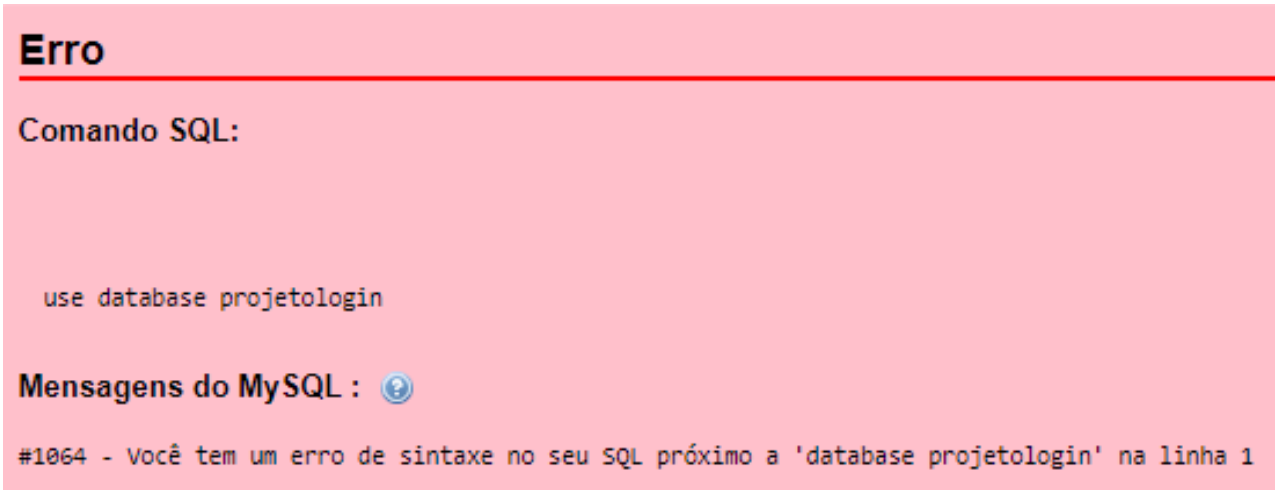
- O meu banco de dados chama-se projetologin.
- Criei três tabelas: login, usuario e login_usuario.
- Tabela login guarda os dados para o usuário entrar no sistema.
- Tabela usuario guarda dados pessoais do usuário.
- Tabela login_usuario liga o id do login ao id do usuario.

Entre na aba SQL do phpmyadmin e cole o script do banco de dados. Clique no botão executar.

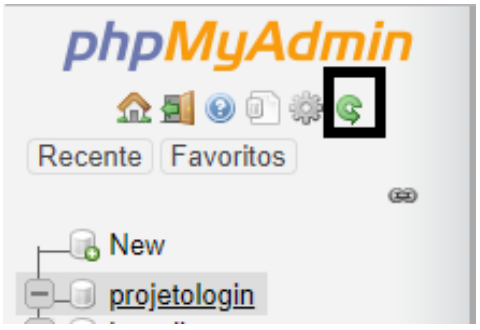
BANCO DE DADOS DO SISTEMA



Provavelmente vai aparecer um erro, mas o banco será criado.

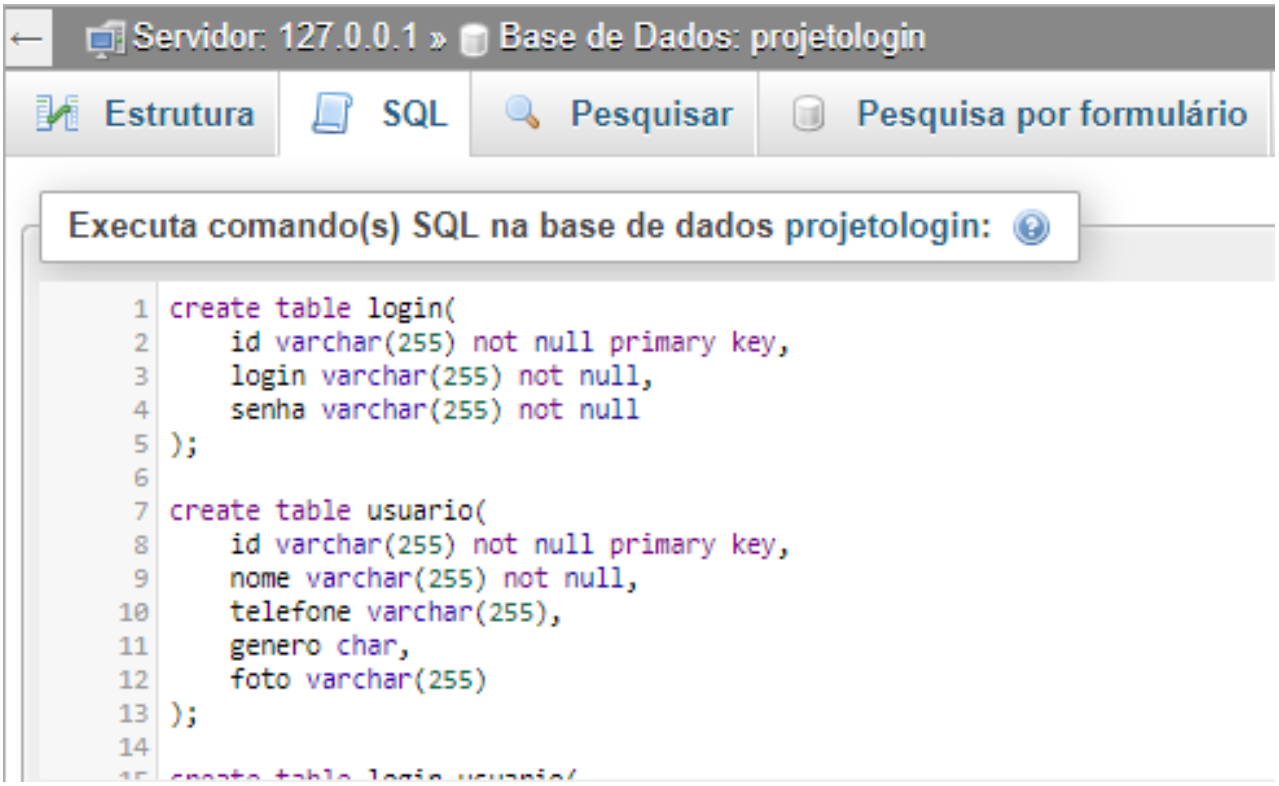


Clique no banco de dados criado. Caso o banco não estiver aparecendo, clique no botão que está marcado na imagem.



BANCO DE DADOS DO SISTEMA

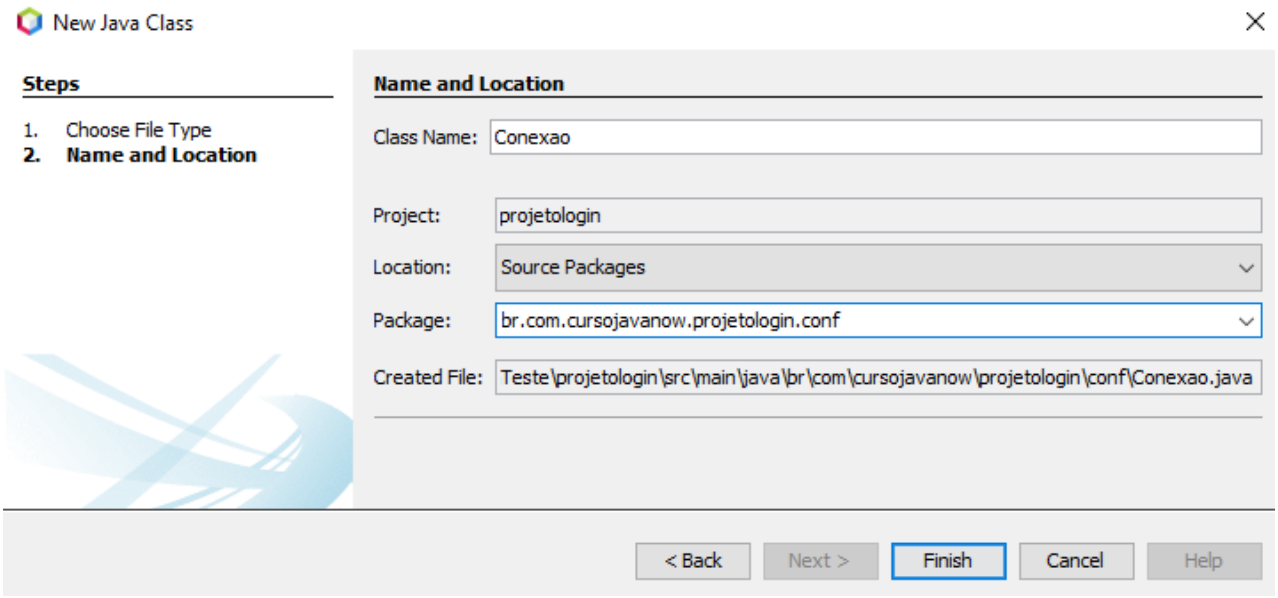
Escolha novamente a aba SQL só que agora adicione o script apenas das tabelas. Clique no botão executar.



CRIANDO A CONEXÃO COM O BANCO DE DADOS

Agora que temos as tabelas, vamos criar a conexão da aplicação com o banco de dados.

Crie uma classe chamada Conexao dentro do pacote conf.



Programação da classe Conexao.

```
package br.com.cursojavanow.projetologin.conf;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import javax.swing.JOptionPane;
/**
 *
 * @author michel adriano medeiros
 */
public class Conexao {

    static Connection con = null;

    public static Connection conectar() {
        try {
            con = DriverManager.getConnection("jdbc:mysql://localhost/projetologin?
serverTimezone=America/Sao_Paulo", "root", "");
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, e);
        }
        return con;
    }
}
```

PROGRAMAÇÃO DO CADASTRO DE USUÁRIO

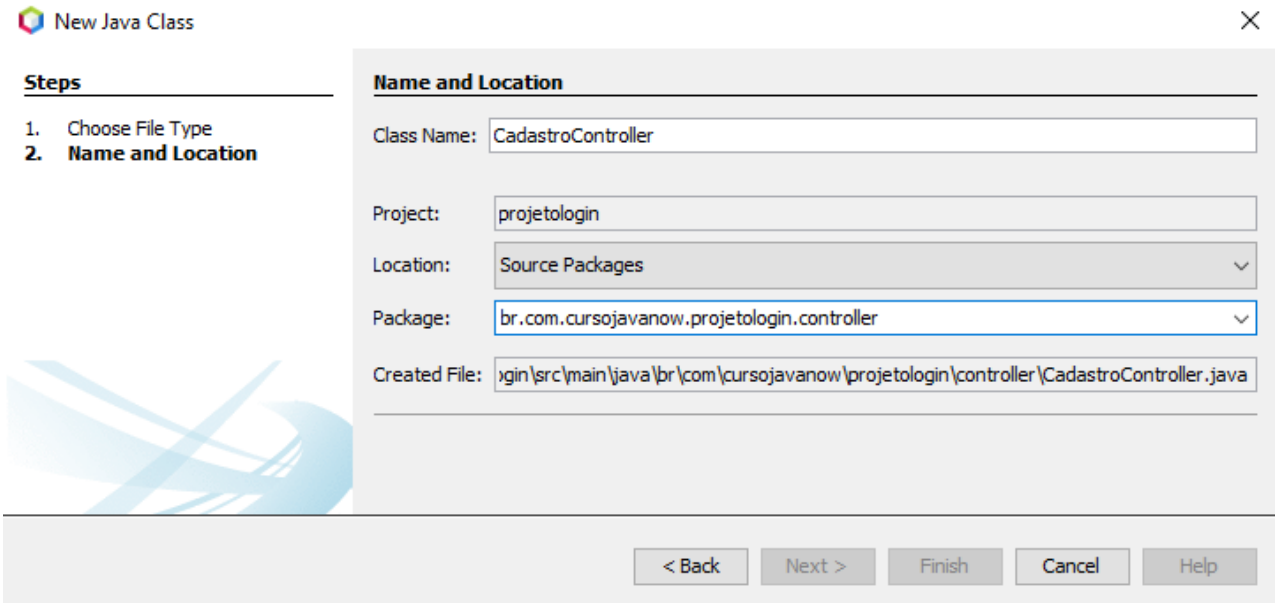
Vamos colocar para funcionar o cadastro de usuário.

Uma regra para este formulário é: quando o usuário digitar o username o sistema deve procurar se há alguém utilizando este username. Se houver, uma mensagem deve aparecer para o usuário escolher outro username.

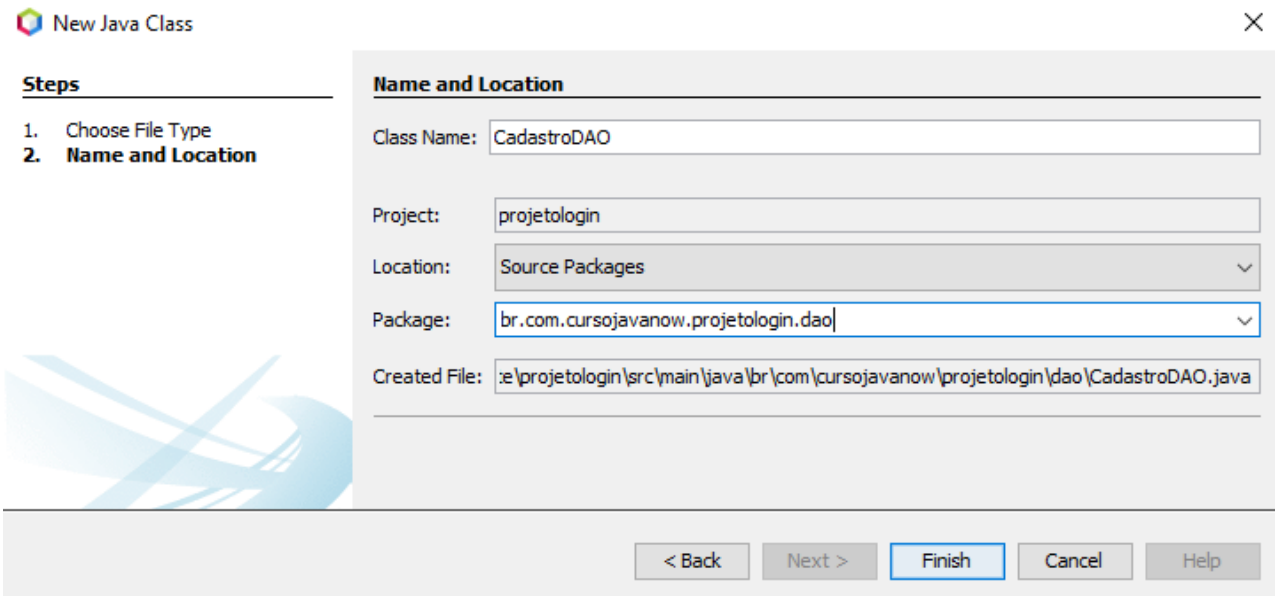
Para esta verificação, podemos ver todas as verificações de uma vez quando o usuário clicar no botão cadastrar.

Mas eu vou verificar quando o usuário passar para outro campo, ou seja, quando o campo username peder o foco, a verificação será feita.

Vamos criar uma classe chamada CadastroController dentro do pacote controller.



Vamos criar também uma classe chamada CadastroDAO dentro do pacote dao.



PROGRAMAÇÃO DO CADASTRO DE USUÁRIO

Agora faça o mesmos passos feito para criar o CadastroController e CadastroDAO, mas com os nomes LoginController e LoginDAO.

Crie no pacote tools uma classe chamada TratamentoConexao, vamos utilizar esta classe para fazer o fechamento das conexões quando não estiverem sendo mais utilizadas.

```
package br.com.cursojavanow.projetologin.tools;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author michel adriano medeiros
 */
public class TratamentoConexao {

    public static void fecharConexao(Connection con) {
        if (con != null) {
            try {
                con.close();
            } catch (SQLException ex) {
                Logger.getLogger(TratamentoConexao.class.getName())
                      .log(Level.SEVERE, null, ex);
            }
        }
    }

    public static void fecharConexaoEResultSet(Connection con, ResultSet rs) {
        if (rs != null) {
            try {
                rs.close();
            } catch (SQLException ex) {
                Logger.getLogger(TratamentoConexao.class.getName())
                      .log(Level.SEVERE, null, ex);
            }
        }
        fecharConexao(con);
    }

}
```

PROGRAMAÇÃO DO CADASTRO DE USUÁRIO

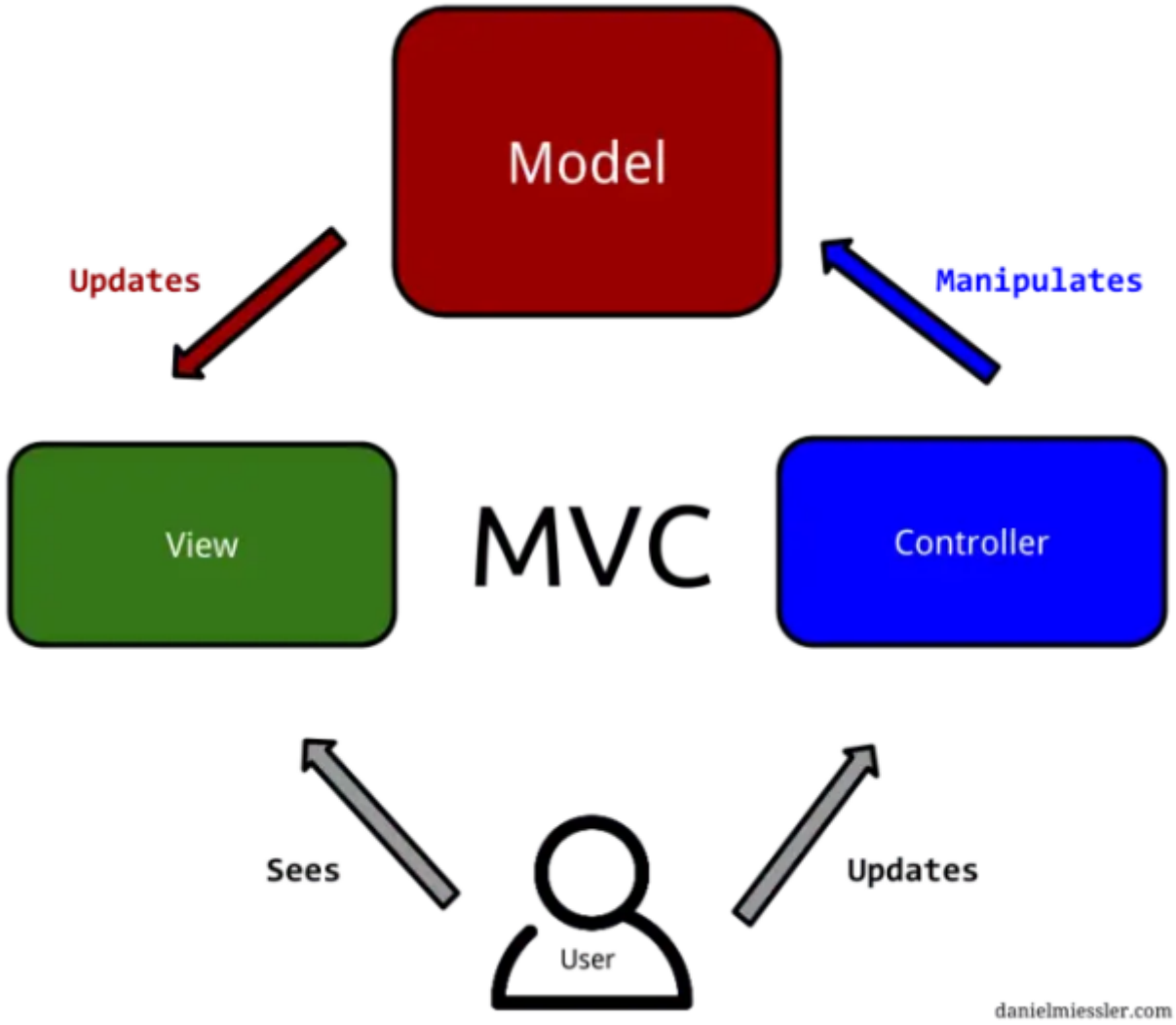
A classe LoginDAO tem o seguinte código:

```
package br.com.cursojavanow.projetologin.dao;

import br.com.cursojavanow.projetologin.conf.Conexao;
import br.com.cursojavanow.projetologin.tools.TratamentoConexao;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

/**
 *
 * @author michel adriano medeiros
 */
public class LoginDAO {

    Connection con = null;

    public LoginDAO() {
        con = Conexao.conectar();
    }

    public boolean verificarUsername(String procurar) {
        boolean erro = false;
        ResultSet rs = null;
        try {
            String sql = "SELECT * FROM `login` WHERE login=?";
            PreparedStatement ps = con.prepareStatement(sql);
            ps.setString(1, procurar);
            rs = ps.executeQuery();
            if (rs.next()) {
                erro = true;
                return erro;
            }
        } catch (SQLException e) {
            System.out.println("ERRO verificarUsername " + e.getMessage());
        } finally {
            TratamentoConexao.fecharConexaoEResultSet(con, rs);
        }
        return erro;
    }
}
```

PROGRAMAÇÃO DO CADASTRO DE USUÁRIO

Se você não notou ou não sabe, estamos fazendo a estrutura MVC. Por exemplo:

A tela cadastro é a view (V), a classe CadastroController é o controller (C) e o CadastroDAO é o model (m).



Criei mais duas classes no pacote tools, uma para mensagens com o nome MessageTool e outra para guardar as imagens do aviso, chamada ImageIconTool.

PROGRAMAÇÃO DO CADASTRO DE USUÁRIO

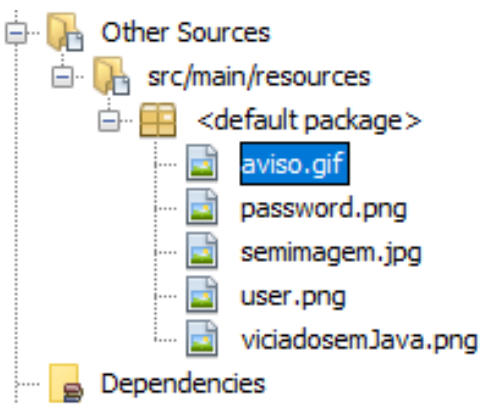
Se você não notou ou não sabe, estamos fazendo a estrutura MVC. Por exemplo:

A tela cadastro é a view (V), a classe CadastroController é o controller (C) e o CadastroDAO é o model (m).

```
package br.com.cursojavanow.projetologin.tools;

import java.net.URL;
import javax.swing.Icon;

/**
 *
 * @author michel adriano medeiros
 */
public class ImageIconTool {

    public static final Icon WARNING = icon("/aviso.gif");

    private static Icon icon(String path) {
        URL resource = ImageIconTool.class.getResource(path);
        if (resource == null) {
            System.out.println("Resource " + path + " does not exist");
            return new javax.swing.ImageIcon();
        }
        return new javax.swing.ImageIcon(resource);
    }
}
```

Lembre-se que as imagens estão armazenadas na pasta src/main/resources. As imagens utilizadas na janela de aviso tem o tamanho de 50x50.

Para redimensionar o gif eu utilizei o site iloveimg.
https://www.iloveimg.com/pt/redimensionar-imagem



PROGRAMAÇÃO DO CADASTRO DE USUÁRIO

```
package br.com.cursojavanow.projetologin.tools;

import javax.swing.JOptionPane;

/**
 *
 * @author michel adriano medeiros
 */
public class MessageTool {

    public static void usuarioJaExiste(String procurar) {
        JOptionPane.showMessageDialog(null,
                "Já existe alguém utilizando o username " + procurar + ".\n"
                + "Por favor escolha outro.",
                "Atenção",
                JOptionPane.INFORMATION_MESSAGE, ImageIconTool.WARNING);
    }

}
```

Código da classe LoginController.

```
package br.com.cursojavanow.projetologin.controller;

import br.com.cursojavanow.projetologin.dao.LoginDAO;
import br.com.cursojavanow.projetologin.tools.MessageTool;

/**
 *
 * @author michel adriano medeiros
 */
public class LoginController {

    public boolean verificarUsername(String procurar) {
        LoginDAO ldao = new LoginDAO();
        boolean erro = ldao.verificarUsername(procurar);

        if (erro) {
            MessageTool.usuarioJaExiste(procurar);
        }
        return erro;
    }
```

PROGRAMAÇÃO DO CADASTRO DE USUÁRIO

Declare a variável erro que é um boolean, no início da classe Cadastro.

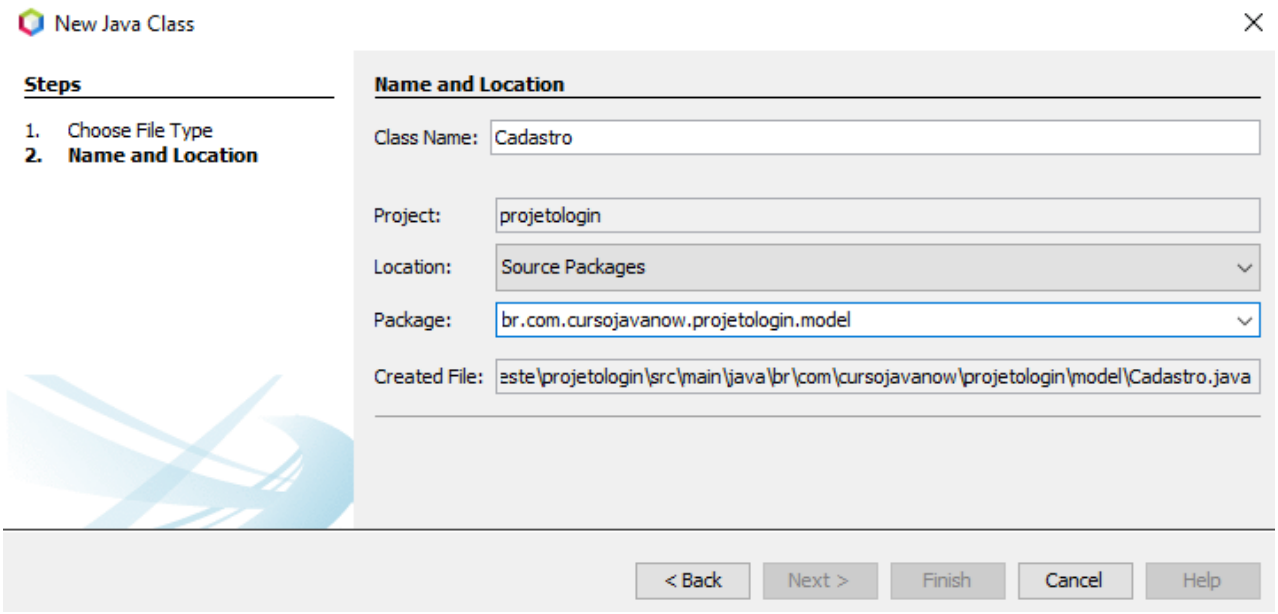
```
public class Cadastro extends javax.swing.JFrame {

    String uuid;
    boolean erro = false;
```

Na tela cadastro do usuário quando o foco for perdido do campo username tem o seguinte código:

```
private void JTFUserNameFocusLost(java.awt.event.FocusEvent evt) {
    LoginController lc = new LoginController();
    erro = lc.verificarUsername(JTFUserName.getText());
}
```

Agora que está tudo preparado, vamos fazer a programação para inserir um novo usuário.

Criei uma classe chamada Cadastro no pacote model.



PROGRAMAÇÃO DO CADASTRO DE USUÁRIO

```
package br.com.cursojavanow.projetologin.model;

public class Cadastro {

    private String id;
    private String nome;
    private String telefone;
    private char genero;
    private String foto;

    public String getId() {
        return id;
    }

    public void setId(String 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;
    }

    public char getGenero() {
        return genero;
    }

    public void setGenero(char genero) {
        this.genero = genero;
    }

    public String getFoto() {
        return foto;
    }

    public void setFoto(String foto) {
        this.foto = foto;
    }

}
```

PROGRAMAÇÃO DO CADASTRO DE USUÁRIO

Criei no mesmo pacote a classe Login.

```
package br.com.cursojavanow.projetologin.model;

/**
 *
 * @author Michel Adriano Medeiros
 */
public class Login extends Cadastro {

    private String id;
    private String login;
    private String senha;

    @Override
    public String getId() {
        return id;
    }

    @Override
    public void setId(String id) {
        this.id = id;
    }

    public String getLogin() {
        return login;
    }

    public void setLogin(String login) {
        this.login = login;
    }

    public String getSenha() {
        return senha;
    }

    public void setSenha(String senha) {
        this.senha = senha;
    }

}
```

PROGRAMAÇÃO DO CADASTRO DE USUÁRIO

E também criei no pacote model a classe LoginUsuario.

```
package br.com.cursojavanow.projetologin.model;

/**
 *
 * @author michel adriano medeiros
 */
public class LoginUsuario {

    private String id;
    public static String login;
    public static String usuario;

    public String getId() {
        return id;
    }

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

    public static String getLogin() {
        return login;
    }

    public static void setLogin(String login) {
        LoginUsuario.login = login;
    }

    public static String getUsuario() {
        return usuario;
    }

    public static void setUsuario(String usuario) {
        LoginUsuario.usuario = usuario;
    }

}
```

PROGRAMAÇÃO DO CADASTRO DE USUÁRIO

Voltemos para classe MessageTool para adicionar o método campoNomeVazio e campoSenhaDiferente.

```
public static void campoNomeVazio() {
    JOptionPane.showMessageDialog(null,
            "Digite o nome.\n",
           "Atenção",
           JOptionPane.INFORMATION_MESSAGE, ImageIconTool.WARNING);
}

public static void campoSenhaDiferente() {
    JOptionPane.showMessageDialog(null,
           "As senhas são diferentes.\n "
           + "Elas precisam ser iguais.",
            "Atenção",
           JOptionPane.INFORMATION_MESSAGE, ImageIconTool.WARNING);
}
```

Na tela cadastro criei um método para fazer as verificações dos campos obrigatórios.

```
public boolean verificarCampos() {
    boolean tudoCerto = false;

    String nome = JTFNome.getText();
    String password = new String(JTFPasswordRepetir.getPassword());
    String repeatPassword = new String(JTFPasswordRepetir.getPassword());

    if (!nome.isBlank()) {
        tudoCerto = true;
    } else {
        tudoCerto = false;
         MessageTool.campoNomeVazio();
    }

    if (password.equalsIgnoreCase(repeatPassword)) {
        tudoCerto = true;
    } else {
        tudoCerto = false;
        MessageTool.campoSenhaDiferente();
    }

    return tudoCerto;
}
```

PROGRAMAÇÃO DO CADASTRO DE USUÁRIO

Programação do botão cadastrar.

```
        String nome = JTFNome.getText();
        String username = JTFUserName.getText();
        String password = new String(JTFPassword.getPassword());
        String telefone = JTFTelefone.getText();
        char genero = 'X';
        String foto = "sem";

        if (JCBFeminino.isSelected()) {
            genero = 'F';
        } else if (JCBMasculino.isSelected()) {
            genero = 'M';
        }

        if (!uuid.isBlank()) {
            foto = uuid;
        }

        if (erro) {
            MessageTool.usuarioJaExiste(username);
        } else {
            if (verificarCampos()) {
                br.com.cursojavanow.projetologin.model.Login c
                        = new br.com.cursojavanow.projetologin.model.Login();
                c.setNome(nome);
                c.setLogin(username);
                c.setSenha(password);
                c.setTelefone(telefone);
                c.setGenero(genero);
                c.setFoto(foto);
            }
        }
```

Crie uma classe chamada GerarUUID no pacote tools.

```
package br.com.cursojavanow.projetologin.tools;
import java.util.UUID;
public class GerarUUID {

    public static String gerarUUID() {
        UUID uuid = UUID.randomUUID();
        String id = uuid.toString();
        return id;
    }

}
```

PROGRAMAÇÃO DO CADASTRO DE USUÁRIO

Adicione na classe LoginDAO o método salvar.

```
public String salvar(Login c) {
    String status = "Falhou";
    String uuid = GerarUUID.gerarUUID();
    LoginUsuario.setLogin(uuid);
     try {
        String sql = "insert into login values(?,?,sha2(?, 256))";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setString(1, uuid);
        ps.setString(2, c.getLogin());
        ps.setString(3, c.getSenha());
        ps.execute();
         status = "OK";
    } catch (SQLException e) {
        System.out.println("Erro salvar " + e.getMessage());
     } finally {
        TratamentoConexao.fecharConexao(con);
    }
    return status;
}
```

Vamos agora criar o método para salvar o cadastro do usuário na classe CadastroDAO.

PROGRAMAÇÃO DO CADASTRO DE USUÁRIO

```
package br.com.cursojavanow.projetologin.dao;
import br.com.cursojavanow.projetologin.conf.Conexao;
import br.com.cursojavanow.projetologin.model.Login;
import br.com.cursojavanow.projetologin.model.LoginUsuario;
import br.com.cursojavanow.projetologin.tools.GerarUUID;
import br.com.cursojavanow.projetologin.tools.TratamentoConexao;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class CadastroDAO {

    Connection con = null;

    public CadastroDAO() {
        con = Conexao.conectar();
    }

    public String salvar(Login c) {
        String status = "Falhou";
        String uuid = GerarUUID.gerarUUID();
        LoginUsuario.setUsuario(uuid);
        try {
            String sql = "insert into usuario values(?,?,?,?,?)";
            PreparedStatement ps = con.prepareStatement(sql);
            ps.setString(1, uuid);
            ps.setString(2, c.getNome());
            ps.setString(3, c.getTelefone());
            ps.setString(4, String.valueOf(c.getGenero()));
            ps.setString(5, c.getFoto());
            ps.execute();
            status = "OK";
        } catch (SQLException e) {
            System.out.println("Erro salvar " + e.getMessage());
        } finally {
            TratamentoConexao.fecharConexao(con);
        }
        return status;
    }

}
```

PROGRAMAÇÃO DO CADASTRO DE USUÁRIO

Vamos agora criar o método para salvar as informações na tabela login_usuario. Para isto vamos criar a classe LoginUsuarioDAO.

```
package br.com.cursojavanow.projetologin.dao;

import br.com.cursojavanow.projetologin.conf.Conexao;
import br.com.cursojavanow.projetologin.model.LoginUsuario;
import br.com.cursojavanow.projetologin.tools.GerarUUID;
import br.com.cursojavanow.projetologin.tools.TratamentoConexao;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class LoginUsuarioDAO {

    Connection con = null;

    public LoginUsuarioDAO() {
        con = Conexao.conectar();
    }

    public String salvar() {
        String status = "Falhou";
        try {
            String sql = "insert into login_usuario values(?,?,?)";
            PreparedStatement ps = con.prepareStatement(sql);
            ps.setString(1, GerarUUID.gerarUUID());
            ps.setString(2, LoginUsuario.getLogin());
            ps.setString(3, LoginUsuario.getUsuario());
            ps.execute();
            status = "OK";
        } catch (SQLException e) {
            System.out.println("Erro salvar " + e.getMessage());
        } finally {
            TratamentoConexao.fecharConexao(con);
        }
        return status;
    }

}
```

PROGRAMAÇÃO DO CADASTRO DE USUÁRIO

Criamos todos os métodos necessários para salvar as informações do login, cadastro e a ligação destas duas informações.

Vamos para a classe MessageTool para adicionar o método erroAoSalvarDadosUsuario.

```
    public static void erroAoSalvarDadosUsuario(String msg) {
        JOptionPane.showMessageDialog(null,
                "Erro ao tentar salvar os dados de " + msg + " do usuário.",
                "Atenção",
                JOptionPane.INFORMATION_MESSAGE, ImageIconTool.WARNING);
    }
```

Na classe CadastroController vamos adicionar o método save.

```
package br.com.cursojavanow.projetologin.controller;

import br.com.cursojavanow.projetologin.dao.CadastroDAO;
import br.com.cursojavanow.projetologin.model.Login;
import br.com.cursojavanow.projetologin.tools.MessageTool;

/**
 *
 * @author michel adriano medeiros
 */
public class CadastroController {

    public static void save(Login c) {
        CadastroDAO u = new CadastroDAO();
        String message = u.salvar(c);
        if (message.equalsIgnoreCase("ok")) {
            LoginController.save(c);
        } else {
            MessageTool.erroAoSalvarDadosUsuario("cadastro");
        }
    }

}
```

PROGRAMAÇÃO DO CADASTRO DE USUÁRIO

Vamos adicionar o método save na classe LoginController.

```
    public static void save(Login c) {
        LoginDAO u = new LoginDAO();
        String message = u.salvar(c);
        if (message.equalsIgnoreCase("ok")) {
            LoginUsuarioController.save();
        } else {
            MessageTool.erroAoSalvarDadosUsuario("login");
        }
    }
```

Crie no pacote controller a classe LoginUsuarioController. E digite o seguinte código:

```
package br.com.cursojavanow.projetologin.controller;

import br.com.cursojavanow.projetologin.dao.LoginUsuarioDAO;
import br.com.cursojavanow.projetologin.tools.MessageTool;

/**
 *
 * @author michel adriano medeiros
 */
public class LoginUsuarioController {

    public static void save() {
        LoginUsuarioDAO u = new LoginUsuarioDAO();
        String message = u.salvar();
        if (message.equalsIgnoreCase("ok")) {

        } else {
            MessageTool.erroAoSalvarDadosUsuario("login e usuário");
        }
    }

}
```

PROGRAMAÇÃO DO CADASTRO DE USUÁRIO

Vamos voltar para a tela de cadastro e no botão cadastrar, vamos chamar o método save da classe CadastroController.

Já temos a programação deste botão, agora só temos que adicionar uma linha código. CadastroController.save(c);

```
...
        if (verificarCampos()) {
            br.com.cursojavanow.projetologin.model.Login c
                = new br.com.cursojavanow.projetologin.model.Login();
            c.setNome(nome);
            c.setLogin(username);
            c.setSenha(password);
            c.setTelefone(telefone);
            c.setGenero(genero);
            c.setFoto(foto);
            CadastroController.save(c);
        }
....
```

Na classe ImageIconTool adicione a variável de classe: public static final Icon USER_SAVED = icon("userSaved.gif");

Outro site para redimensionar gif é o https://ezgif.com/resize

```
...
    public static final Icon WARNING = icon("/aviso.gif");
    public static final Icon USER_SAVED = icon("/userSaved.gif");
....
```

Na classe MessageTool adicione o método sucessoUsuarioCadastrado.

```
    public static void sucessoUsuarioCadastrado() {
        JOptionPane.showMessageDialog(null,
            "Usuário cadastrado.",
            "Atenção",
            JOptionPane.INFORMATION_MESSAGE, ImageIconTool.USER_SAVED);
    }
```

PROGRAMAÇÃO DO CADASTRO DE USUÁRIO

Na classe LoginUsuarioController adicione a linha MessageTool.sucessoUsuarioCadastrado();

```
public class LoginUsuarioController {

    public static void save() {
        LoginUsuarioDAO u = new LoginUsuarioDAO();
        String message = u.salvar();
        if (message.equalsIgnoreCase("ok")) {
            MessageTool.sucessoUsuarioCadastrado();
        } else {
            MessageTool.erroAoSalvarDadosUsuario("login e usuário");
        }
    }

}
```

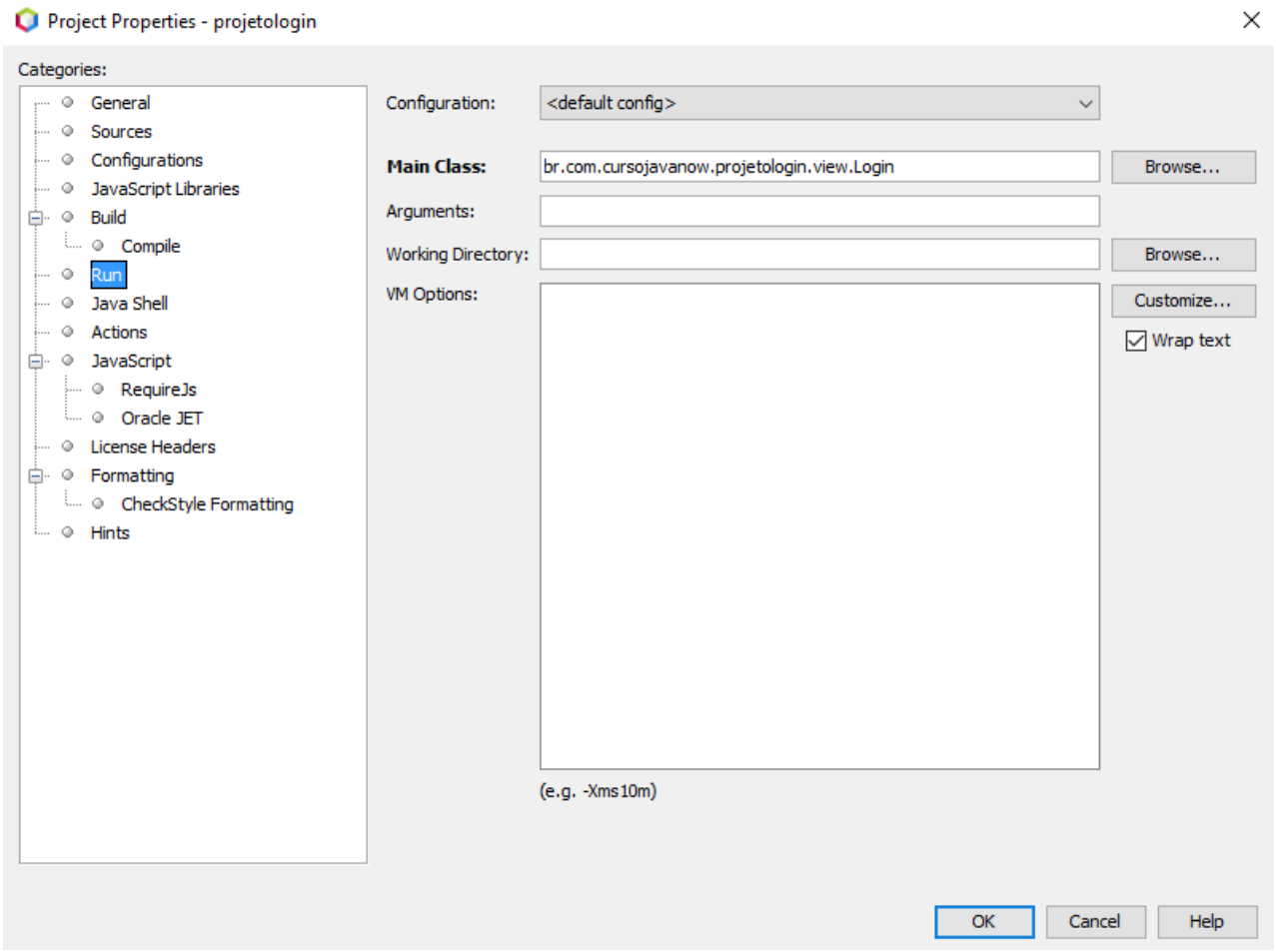
Volte no arquivo pom.xml e modifique o arquivo onde está a anotação <configuration>

Antes estava apontando o inicio do programa o arquivo Start.java, resolvi apontar direto para o arquivo Login.java do pacote view.

```
...
                    <configuration>
                        <transformers>
                            <!-- add Main-Class to manifest file -->
                            <transformer
implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransform
er">

 <mainClass>br.com.cursojavanow.projetologin.view.Login</mainClass>
                            </transformer>
                        </transformers>
                    </configuration>
....
```

Verifique na configuração do projeto para ver se a sua classe principal está apontado para a view Login.java.

PROGRAMAÇÃO DO CADASTRO DE USUÁRIO



Agora teste, tente cadastrar um usuário. O meu deu certo. ;)

PROGRAMAÇÃO DA TELA DE LOGIN

O cadastro está funcionando, agora vamos fazer a programação do botão entrar.

Só que antes, vamos criar na classe LoginDAO o método entrar.

```
public boolean entrar(Login c) {
    boolean erro = false;
    ResultSet rs = null;
    try {
        String sql = "SELECT * FROM `login` WHERE login=? "
                + "and senha=sha2(?, 256)";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setString(1, c.getLogin());
        ps.setString(2, c.getSenha());
        rs = ps.executeQuery();
        if (rs.next()) {
            erro = true;
            return erro;
        }
    } catch (SQLException e) {
        System.out.println("ERRO entrar " + e.getMessage());
    } finally {
        TratamentoConexao.fecharConexaoEResultSet(con, rs);
    }
    return erro;
}
```

Vamos até a classe MessageTool e adicionar as mensagens: você entrou no sistema e username ou senha estão errados.

```
public static void sucessoUsuarioEntrou() {
    JOptionPane.showMessageDialog(null,
            "Você entrou no sistema.",
            "Atenção",
            JOptionPane.INFORMATION_MESSAGE, ImageIconTool.USER_SAVED);
}

public static void falhouUsuarioEntrou() {
    JOptionPane.showMessageDialog(null,
            "Username ou senha estão errados.",
            "Atenção",
            JOptionPane.INFORMATION_MESSAGE, ImageIconTool.WARNING);
}
```

PROGRAMAÇÃO DA TELA DE LOGIN

Na classe LoginController adicione o método entrar.

```
public static void entrar(Login c) {
    LoginDAO u = new LoginDAO();
    boolean b = u.entrar(c);
    if (b) {
        MessageTool.sucessoUsuarioEntrou();
    } else {
        MessageTool.falhouUsuarioEntrou();
    }
}
```

Agora sim, vamos até a tela de login e dê dois cliques no botão entrar e digite o código:

```
private void JBLoginActionPerformed(java.awt.event.ActionEvent evt) {
    String username = JTFUserName.getText();
    String password = new String(JTFPassword.getPassword());
    br.com.cursojavanow.projetologin.model.Login c
            = new br.com.cursojavanow.projetologin.model.Login();
    c.setLogin(username);
    c.setSenha(password);
    LoginController.entrar(c);
}
```

Agora teste e digite uma senha ou username errado para ver a mensagem de erro, e depois entre com os dados corretos.

FUNCIONOU!!!!

Pronto, terminamos por aqui. Agora você tem um modelo de login para um sistema Java desktop.

Tutorial do site https://cursojavanow.com.br/

Link do tutorial: https://cursojavanow.com.br/login-com-java-swing-utilizando-netbeans-e-mysql/