Desenvolvimento de Sistemas

Interface *desktop*: construção de interface de usuário, manipulação de eventos, uso de controles, manipulação de janelas, construção de formulários e listagens (Parte 2)

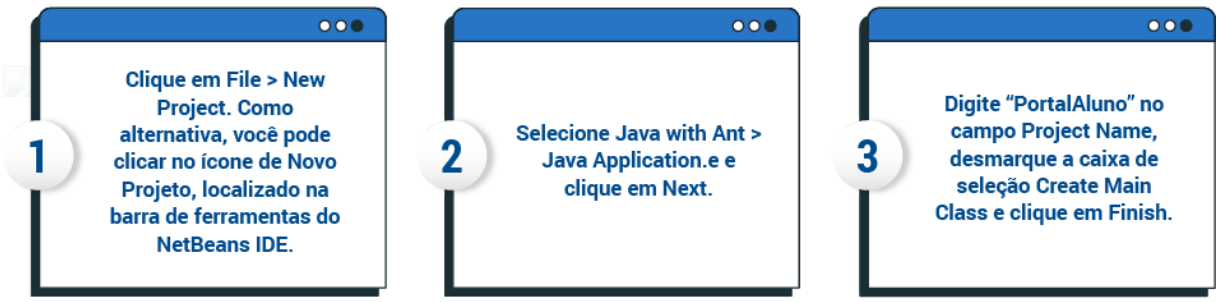
Construção de telas, formulários e listagens

Agora que você já conhece os principais componentes do Swing e sabe como trabalhar com eventos, será preciso colocar em prática tudo o que aprendeu. Para isso, você criará um projeto que englobará todos os conhecimentos desenvolvidos até este momento.

Nota: caso não tenha estudado o conteúdo **Interface *desktop*... (Parte 1)**, recomenda-se que faça isso antes de prosseguir seu estudo neste conteúdo.

Criando um projeto Java com interface gráfica para *desktop*

Para começar, você criará um novo projeto Java no Apache NetBeans IDE. Esse projeto será um sistema para cadastrar e consultar alunos matriculados em uma instituição de ensino. Então, esse projeto será chamado de “PortalAluno”.



Para esse projeto, você deve criar três telas diferentes:

a. Tela inicial

b. Tela de cadastro

c. Tela de listagem

Passo 1

Depois de criar o projeto, crie um novo pacote Java com o nome **br.com.senacead.portalaluno.telas**, pressionando o botão direito do *mouse* sobre o pacote **portalaluno** e selecionando **New > Java Package**. Após isso, o projeto terá a seguinte estrutura:

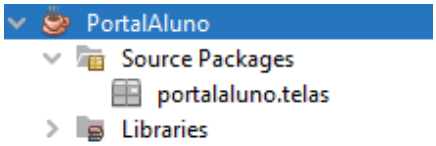


Figura 1 – Estrutura de pastas do projeto PortalAluno
Fonte: Apache NetBeans IDE (2022)

Passo 2

Agora, crie uma tela. Para isso, clique com o botão direito do *mouse* sobre o pacote recém-criado e selecione as opções **New** > **JFrame Form**.

Passo 3

Uma nova janela será aberta, na qual você deverá especificar o nome da classe (*class name*). Para manter uma boa organização no projeto, o ideal é criar as classes com um nome referente ao conteúdo da tela. Crie uma tela inicial para seu projeto. Então, você poderá chamar a classe de **TelaInicial**. Após informar o nome da classe, clique em **Finish**.

Passo 4

Repita o processo e crie mais duas telas: **Cadastro** e **Listagem**.

No final, você terá a seguinte estrutura de arquivos no Apache NetBeans IDE:

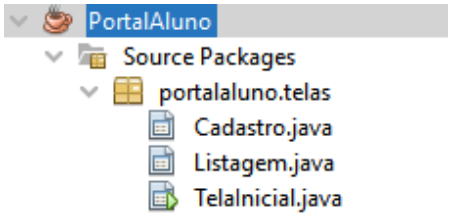


Figura 2 – Estrutura de pastas do projeto “PortalAluno”
Fonte: Apache NetBeans IDE (2022)

Construindo telas do projeto

Embora o GUI Builder do IDE simplifique o processo de criação de GUIs Java, geralmente é útil esboçar a aparência da interface antes de começar a organizá-la por meio de *wireframes*, por exemplo. Muitos *designers* de interface consideram essa uma técnica de "prática recomendada", no entanto, para os propósitos deste conteúdo, será deixada uma prévia das telas para você ter uma ideia do que deve ser construído.

Navegação entre telas

Tela inicial

O objetivo neste momento é simples:

- Quando o usuário clicar no botão **Listagem de alunos**, a tela **Listagem.java** deverá se abrir.
- Quando o usuário clicar no botão **Cadastro de alunos**, a tela “Cadastro.java” deverá se abrir.
- Quando o usuário clicar no botão **Sair**, a tela atual deverá se fechar.

Por padrão, o GUI Builder cria cada componente com um nome de variável que seja facilmente identificado no código-fonte. Para isso, utiliza-se o nome do próprio componente. No caso dos botões nos quais você está trabalhando, existem os seguintes nomes:

- Botão de listagem: **jButton1**
- Botão de cadastro: **jButton2**
- Botão de sair: **jButton3**

Dependendo da ordem na qual você criou os seus botões, é possível que eles tenham outro nome. Para confirmar, clique no item que aparece na listagem do canto inferior esquerdo do Apache NetBeans IDE e repare qual componente o GUI Builder destacará.

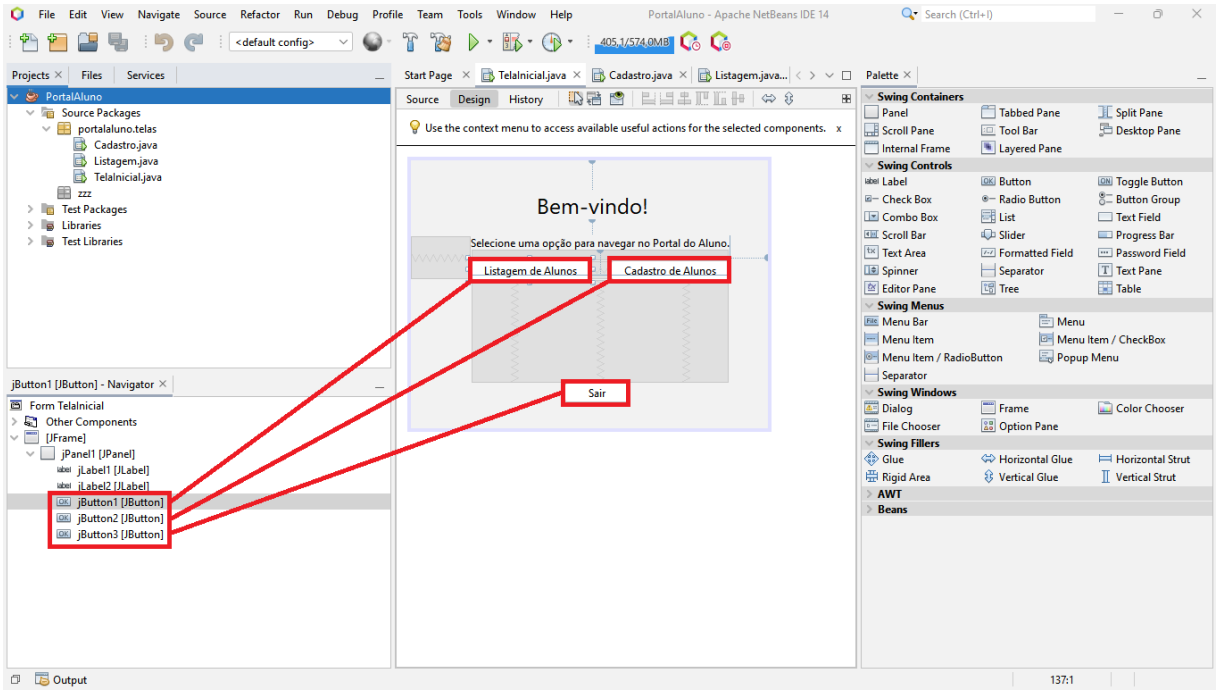


Figura 8 – GUI Builder do Apache NetBeans IDE

Fonte: Apache NetBeans IDE (2022)

Para que você não perca o controle do seu código, é interessante definir os nomes das suas variáveis ao invés de seguir o que o GUI Builder definiu. Afinal, futuramente, ficaria muito difícil diferenciar o **jButton3** do **jButton66**. Então, altere o nome da variável de cada botão. Para isso, clique com o botão direito do *mouse* sobre o componente, selecione a opção **Change Variable Name...** e defina os seguintes nomes para cada botão:

- Listagem de alunos: **botaoListagem**
- Cadastro de alunos: **botaoCadastro**
- Sair: **botaoSair**

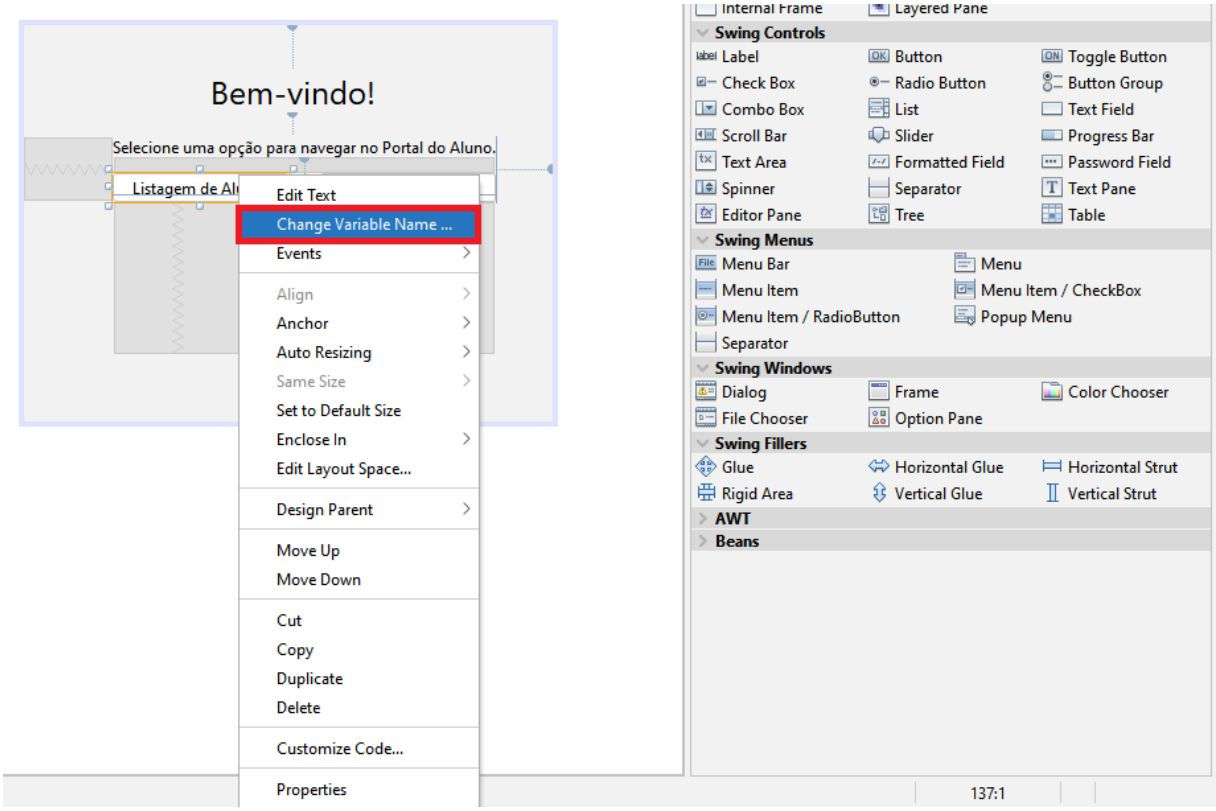


Figura 9 – Acessando a opção **Change Variable Name...** no GUI Builder
Fonte: Apache NetBeans IDE (2022)

Agora, você já tem uma definição clara para cada botão e sabe diferenciar um do outro.

Observação

Outra forma de alterar o nome da variável de um componente é acessando **Properties** > **Code** por meio do GUI Builder e alterando o campo **Variable Name**.

Com as variáveis definidas, agora você pode se concentrar nos eventos de cada botão. Para adicionar o evento de *action*, clique com o botão direito do *mouse* sobre o botão no qual você quer adicionar o evento e selecione as opções **Events** > **Action** > **ActionPerformed**. Comece com o botão **Listagem de alunos**.

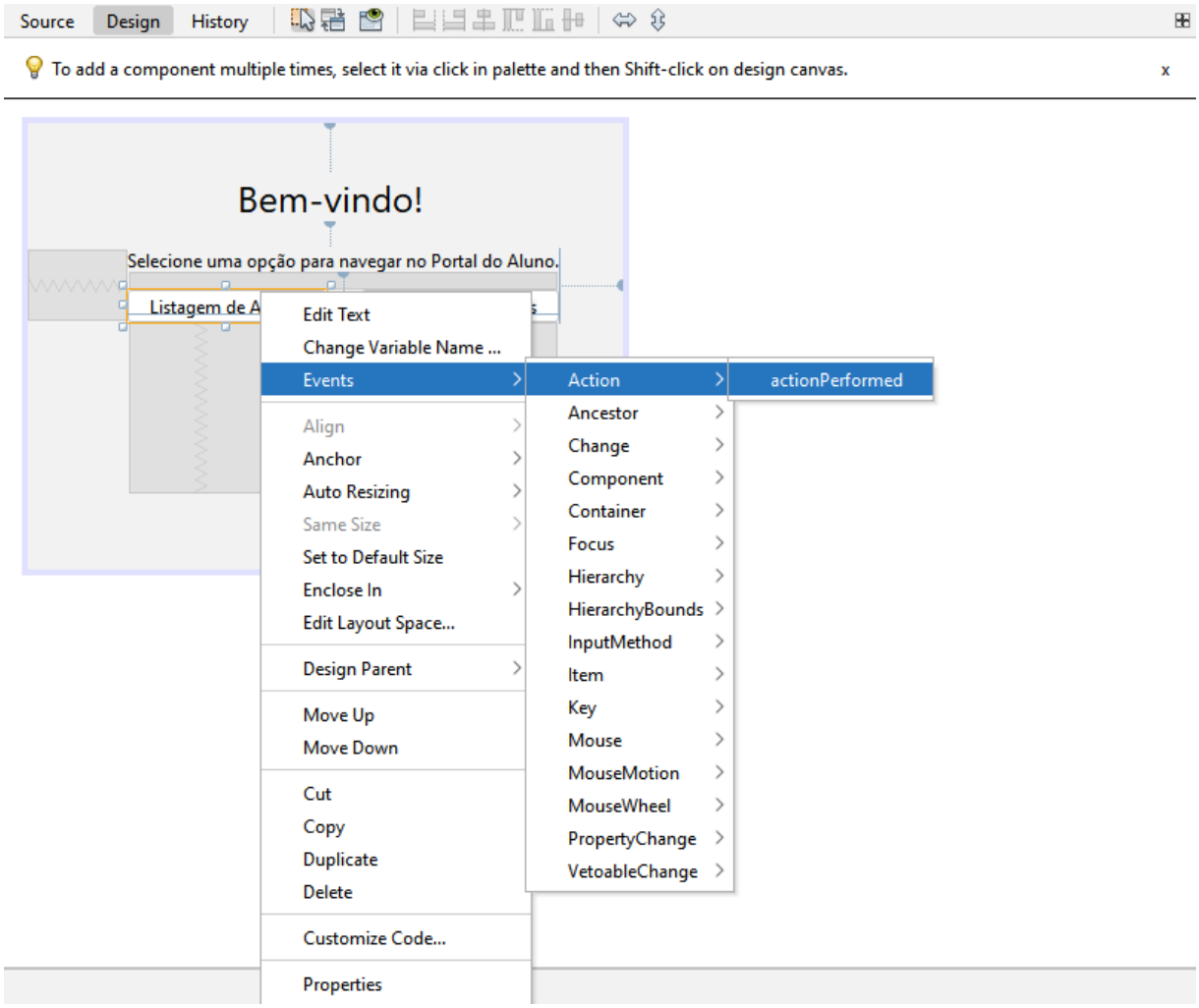


Figura 10 – Acessando eventos do componente no GUI Builder

Fonte: Apache NetBeans IDE (2022)

Isso levará você até a aba de código-fonte na qual terá o seguinte trecho de código focado:

```
private void botaoListagemActionPerformed(java.awt.event.ActionEvent evt) {
  // TODO add your handling code here:
}
```

Perceba que o próprio GUI Builder já se encarregou de criar e configurar o *listener* e o método para se acionar um evento utilizando a variável do componente para nomear o método. Logo, tudo que você precisa fazer é adicionar o trecho de código que abrirá a tela **Listagem.java**. O trecho em questão terá a seguinte estrutura:

```
private void botaoListagemActionPerformed(java.awt.event.ActionEvent evt) {
  Listagem telaListagem = new Listagem();
  telaListagem.setVisible(true);
}
```

O que você fez aqui foi iniciar a tela na memória, instanciando-a como o objeto **telaListagem**. Por meio desse objeto, você aciona o método **setVisible(true)** para que a tela seja aberta para o usuário.

Para o botão de cadastro, repita o procedimento. Porém, neste caso, não se quer abrir a tela de listagem, e sim a de cadastro. Logo, seu código sofrerá alterações e ficará da seguinte maneira:

```
private void botaoCadastroActionPerformed(java.awt.event.ActionEvent evt) {
  Cadastro telaCadastro = new Cadastro();
  telaCadastro.setVisible(true);
}
```

Por fim, adicione o trecho de código que “fechará” a tela inicial. Para isso, você deve estar imaginando usar o método **setVisible(false)**, correto? Mas, se fizermos isso, a tela ainda existirá na memória do computador sem o usuário ver. Para fechar a tela e encerrar o programa, você deve utilizar o método **System.exit(0)**, da seguinte maneira:

```
private void botaoSairActionPerformed(java.awt.event.ActionEvent evt) {
  System.exit(0);
}
```

Agora que sua tela inicial está 100% funcional, implemente a navegação nas outras telas.

Listagem de alunos e Cadastro de alunos

Para ambas as telas, você realizará o mesmo procedimento, já que que ambas contêm apenas um botão voltado para a navegação:

1. Comece alterando o nome da variável do botão **Voltar** para **botaoVoltar**.
2. Após essa alteração, no código-fonte, adicione a ação ao **ActionEvent**.
3. Adicione o seguinte trecho de código:

```
private void botaoVoltarActionPerformed(java.awt.event.ActionEvent evt) {
  dispose();
}
```

Diferentemente do método **setVisible()**, que apenas oculta a tela do usuário enquanto continua na memória, o método **dispose()** fecha a tela e a “limpa” da memória. Perceba também que, nesse caso, não é preciso instanciar a classe da tela como um objeto para acessar o método, porque você já está na tela que quer manipular. Portanto, basta acionar o método direto.

Após implementar esse trecho em ambas as telas, você terá todas as telas com a navegação funcional.

Analise o GIF a seguir:

Implementando as classes de manipulação de dados

Para cadastrar o aluno, são necessárias duas classes Java. A primeira será a classe **Aluno**, responsável por guardar os dados “nome”, “e-mail” e “curso”. Já a segunda, será a classe **ListaAluno**, na qual se pode guardar vários objetos da classe **Aluno** por intermédio de uma lista.

1. Primeiramente, crie um novo pacote dentro de **portalaluno** chamado *model*. Dentro desse pacote, você criará as classes de manipulação de dados do seu sistema.

2. Após a criação do pacote, crie duas novas classes Java: **Aluno** e **ListaAluno**.

A sua estrutura de arquivos no Apache NetBeans IDE ficará assim:

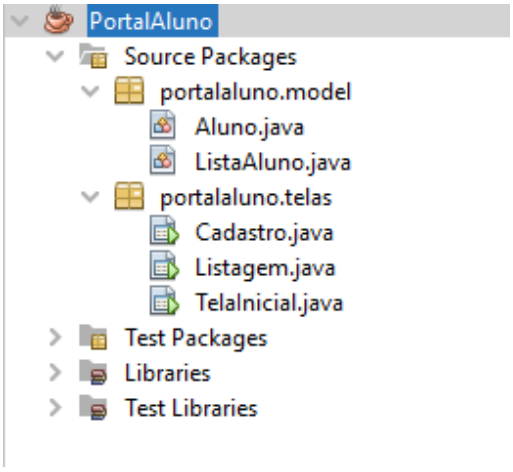


Figura 11 – Estrutura de arquivos do projeto “PortalAluno”

Fonte: Apache NetBeans IDE (2022)

A classe **Aluno** terá o seguinte código:

```
package portalaluno.model;

public class Aluno {

    // Declaração de variáveis
    private String nome;
    private String email;
    private String curso;

    // Métodos para acessar e atualizar dados das variáveis
    public String getNome() {
        return nome;
    }

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

    public String getEmail() {
        return email;
    }

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

    public String getCurso() {
        return curso;
    }

    public void setCurso(String curso) {
        this.curso = curso;
    }

}
```

Já a classe **ListaAluno** terá o seguinte código:

```
package portalaluno.model;

// Importação dos pacotes necessários para usarmos o List e ArrayList
import java.util.ArrayList;
import java.util.List;

public class ListaAluno {
    // Declaração de variáveis
    private static final List<Aluno> lista = new ArrayList<>();

    // Métodos para acessarmos a lista e adicionarmos novos itens
    public static List<Aluno> Listar() {
        return lista;
    }

    public static void Adicionar(Aluno aluno) {
        lista.add(aluno);
    }

}
```

Acessando as classes de manipulação de dados

Estando as classes de manipulação de dados prontas, agora você poderá integrá-las às suas telas para tornar seu sistema funcional para o usuário.

Cadastro de aluno

Comece pela tela **Cadastro.java**. A primeira coisa que precisa fazer nessa tela é atualizar os nomes dos seus componentes para conseguir acessá-los corretamente na aba **Source Code**. Novamente, clique com o botão direito do *mouse* sobre o componente, selecione a opção **Change Variable Name...** e defina os seguintes nomes:

- Altere o nome da **JTextField** abaixo de Nome do **aluno** para **nome**.
- Altere o nome da **JTextField** abaixo de **Email** para **email**.
- Altere o nome da **JComboBox** abaixo de **Curso** para **curso**.

No final, você terá a seguinte configuração:

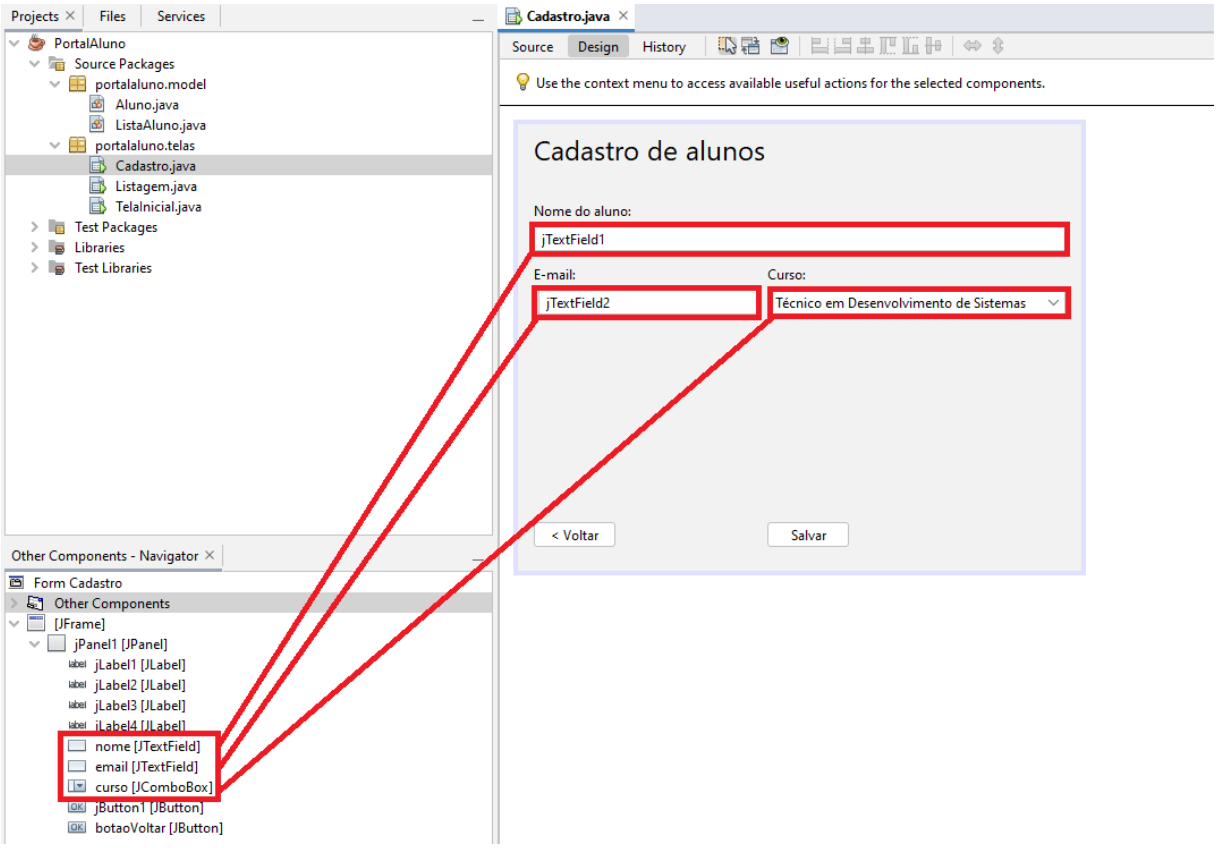


Figura 12 – GUI Builder do Apache NetBeans IDE

Fonte: Apache NetBeans IDE (2022)

Como o que se quer é que o usuário digite os dados em “Nome do aluno” e “E-mail”, convém deixar esses campos limpos para que recebam a entrada de dados do usuário. Então, aproveite e limpe os textos usados como padrões dos componentes **JTextField** clicando com o botão direito sobre cada componente e selecionando **Edit Text...**

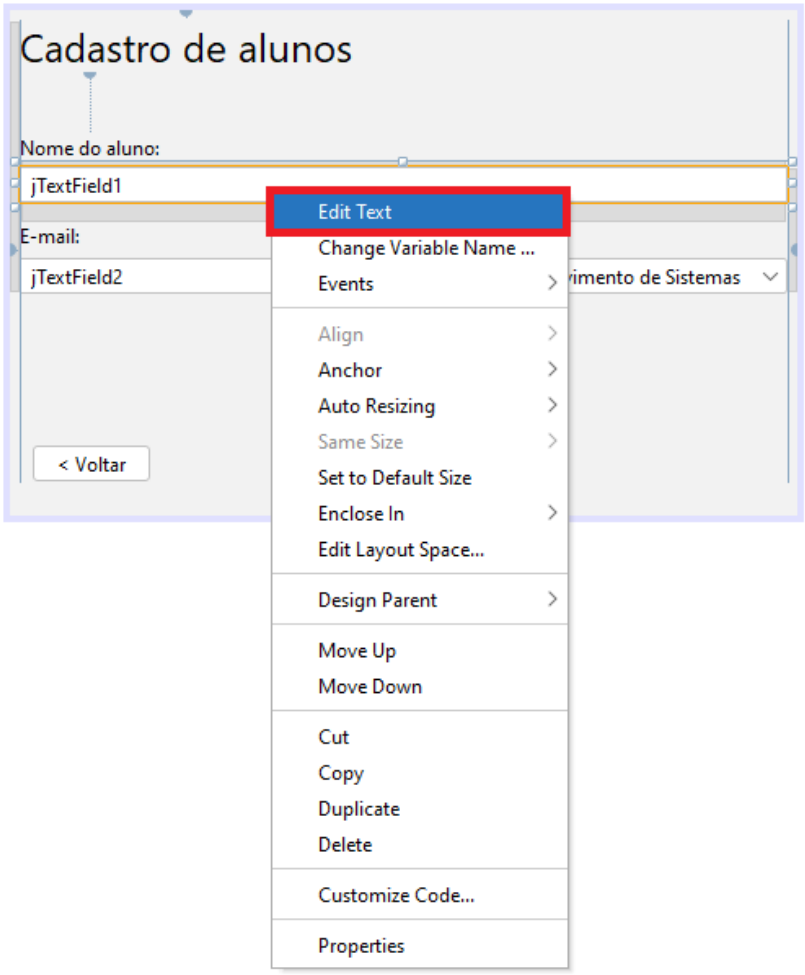


Figura 13 – Acessando o **Edit Text** do componente **JTextField**
Fonte: Apache NetBeans IDE (2022)

Agora, você pode atualizar o nome da variável do botão **Salvar**. Clique com o botão direito do *mouse* sobre o componente, selecione a opção **Change Variable Name...** e altere o nome para **Salvar**. Depois disso, clique com o botão direito do *mouse* sobre o componente e escolha as opções **Events** > **Action** > **ActionPerformed** para escrever o código que será acionado quando o usuário clicar no botão. Você será levado à aba **Source Code** na exata linha para escrever o código do método **SalvarActionPerformed()**.

O código que se quer implementar deve fazer o seguinte:

1. Criar um objeto “aluno” vazio
2. Atualizar os atributos do objeto com os dados do formulário

3. Adicionar o objeto com os atributos definidos a lista de alunos
4. Mostrar uma mensagem para o usuário informando que os dados foram cadastrados com sucesso

Logo, o método **SalvarActionPerformed()** terá o seguinte código:

```
  private void SalvarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    // Criamos o objeto "aluno" com seus respectivos dados
    Aluno aluno = new Aluno();
    aluno.setNome(nome.getText());
    aluno.setEmail(email.getText());
    aluno.setCurso(curso.getSelectedItem().toString());

    // Adicionamos o aluno a lista
    ListaAluno.Adicionar(aluno);

    // Mostramos os dados para o usuário através de um JOptionPane
    JOptionPane.showMessageDialog(null, "Os seguintes dados foram cadastrados com s
ucesso: \n"
            + "\nNome: " + nome.getText()
            + "\nE-mail: " + email.getText()
            + "\nCurso: " + curso.getSelectedItem().toString()
    );
  }
```

Atenção!

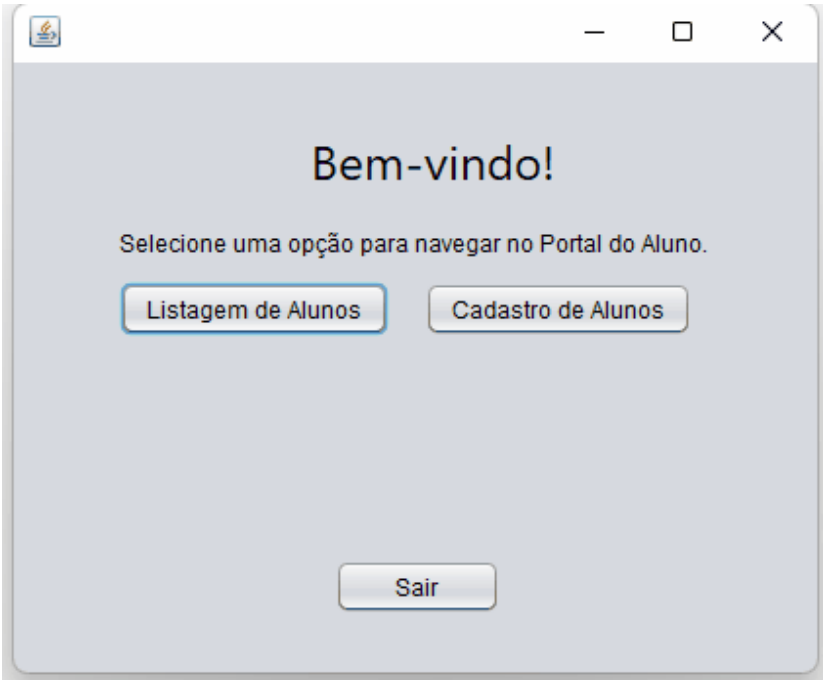
Lembre-se de fazer as importações necessárias no início do código para que as classes Java sejam encontradas e funcionem corretamente.


```
1   package portalaluno.telas;
2
3   import javax.swing.JOptionPane;
4   import portalaluno.model.Aluno;
5   import portalaluno.model.ListaAluno;
6
7   public class Cadastro extends javax.swing.JFrame {
8
9       /** Creates new form Cadastro ...3 lines */
12      public Cadastro() {...3 lines }
15
16      /** This method is called from within the constructor to initialize the form ...5 lines */
21      @SuppressWarnings("unchecked")
22      Generated Code
136
137     private void SalvarActionPerformed(java.awt.event.ActionEvent evt) {
138         // TODO add your handling code here:
139
140         // Criamos o objeto "aluno" com seus respectivos dados
141         Aluno aluno = new Aluno();
142         aluno.setNome(nome.getText());
143         aluno.setEmail(email.getText());
144         aluno.setCurso(curso.getSelectedItem().toString());
145
146         // Adicionamos o aluno a lista
147         ListaAluno.Adicionar(aluno);
148
149         // Mostramos os dados para o usuário através de um JOptionPane
150         JOptionPane.showMessageDialog(null, "Os seguintes dados foram cadastrados com sucesso: \n"
151                 + "\nNome: " + nome.getText()
152                 + "\nE-mail: " + email.getText()
153                 + "\nCurso: " + curso.getSelectedItem().toString()
154         );
155     }
156
157     private void botaoVoltarActionPerformed(java.awt.event.ActionEvent evt) {
158         dispose();
```

Figura 14 – **Source Code** do GUI Builder

Fonte: Apache NetBeans IDE (2022)

Se você testar a aplicação agora, verá o resultado mostrado no seguinte GIF:



Esta aplicação está concluída no conteúdo **Estrutura de dados: sintaxe, bibliotecas de linguagem e aplicabilidade**, desta unidade curricular, utilizando **JTable** para a listagem de itens cadastrados. Neste momento, caso queira verificar se a inserção foi realizada corretamente, você pode usar um laço *for*, percorrer a lista e imprimir cada um dos itens na lista em tela usando um **JOptionDialog** ou mesmo em console **(System.out.println())**.

Encerramento

Assim, conclui-se a segunda parte do conteúdo sobre interfaces *desktops*, no qual foram apresentados e explorados os principais componentes visuais da biblioteca Swing de Java. Com o exemplo, foi possível colocar em prática tudo que você aprendeu na primeira parte e construir um sistema capaz de cadastrar dados da memória.

É importante que você siga testando os outros componentes disponíveis e experimente as funcionalidades e configurações deles. Lembre-se de que somente com a prática você ganhará experiência.