

# LINGUAGEM DE PROGRAMAÇÃO ORIENTADA A OBJETOS

ETEC DE HORTOLÂNDIA

CURSO TÉCNICO EM INFORMÁTICA INTEGRADO AO ENSINO MÉDIO

PROF. RALFE DELLA CROCE FILHO

# CONTEÚDO

- Pacotes
- MVC (model - view - controller)
- JavaFX
  - Estrutura
  - Controles
  - Eventos
  - Navegação
  - SceneBuilder

# MVC (MODEL,VIEW,CONTROLLER)

- É um padrão de projeto estrutural que implementa padrões comportamentais criado na década de 70 e que serviu como base para muitos padrões de projetos criados posteriormente. O MVC possibilita a separação de um projeto em “camadas” onde, basicamente, a lógica de negócios (classes de modelagem com suas definições e funcionalidades) fica na model, a apresentação (interações com o usuário) na view e a interação e controle de fluxo de requisições e respostas entre elas (e demais recursos internos da aplicação) na controller.

## DESIGNER PATTERNS E MVC

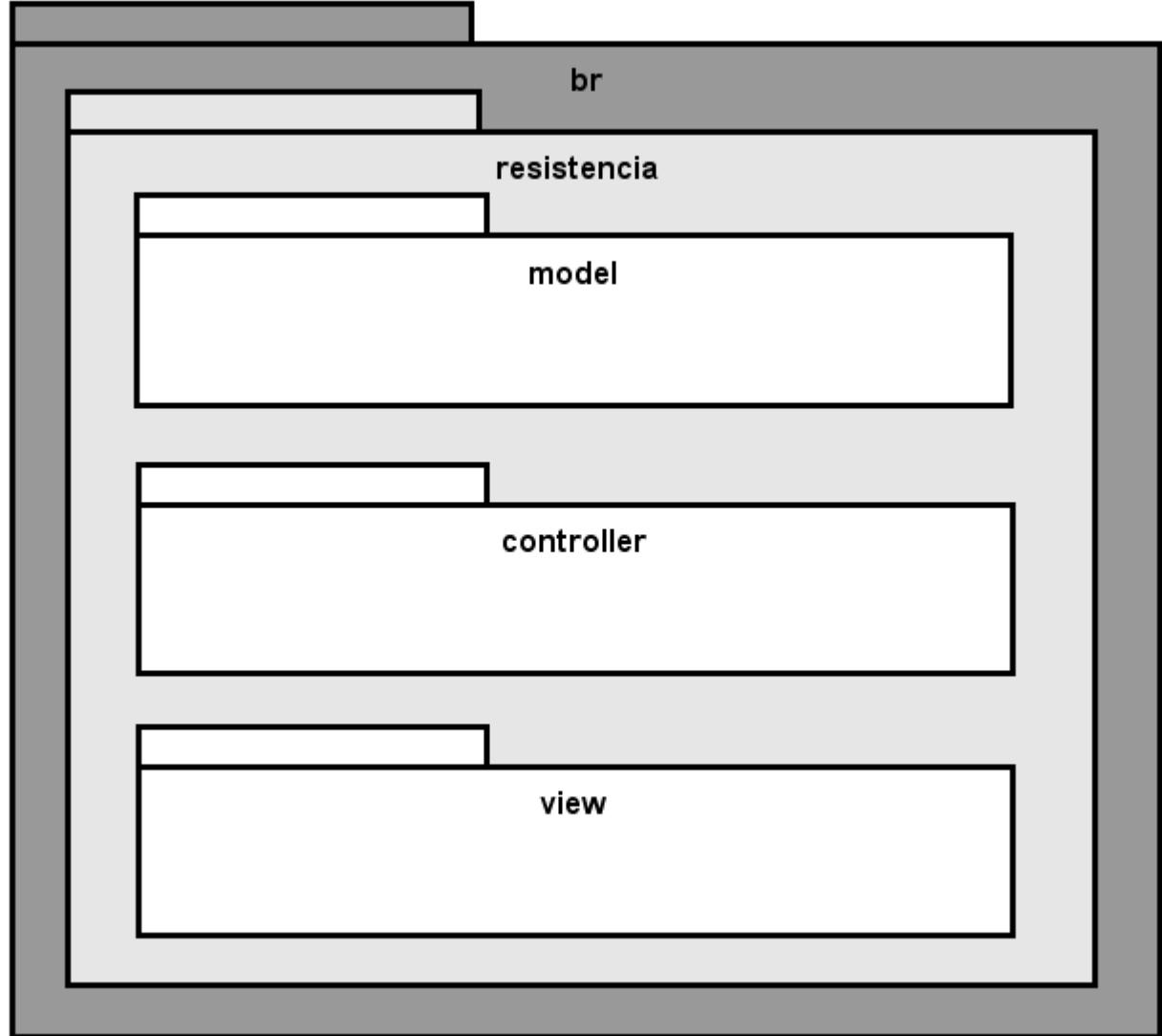
- Os conceitos e definições de Designer Patterns são muito abrangentes quanto a soluções em níveis arquiteturais e muito específicos no tratamento de questões pontuais de modelagem de projeto, ou seja, é um assunto que vai muito além das simples definições descritas acima.
- O MVC não estabelece regras rígidas, ele indica um modelo de separação de responsabilidades em uma aplicação que pode ser adaptado e ampliado de acordo com as necessidades de cada software.

## PACKAGES (PACOTES)

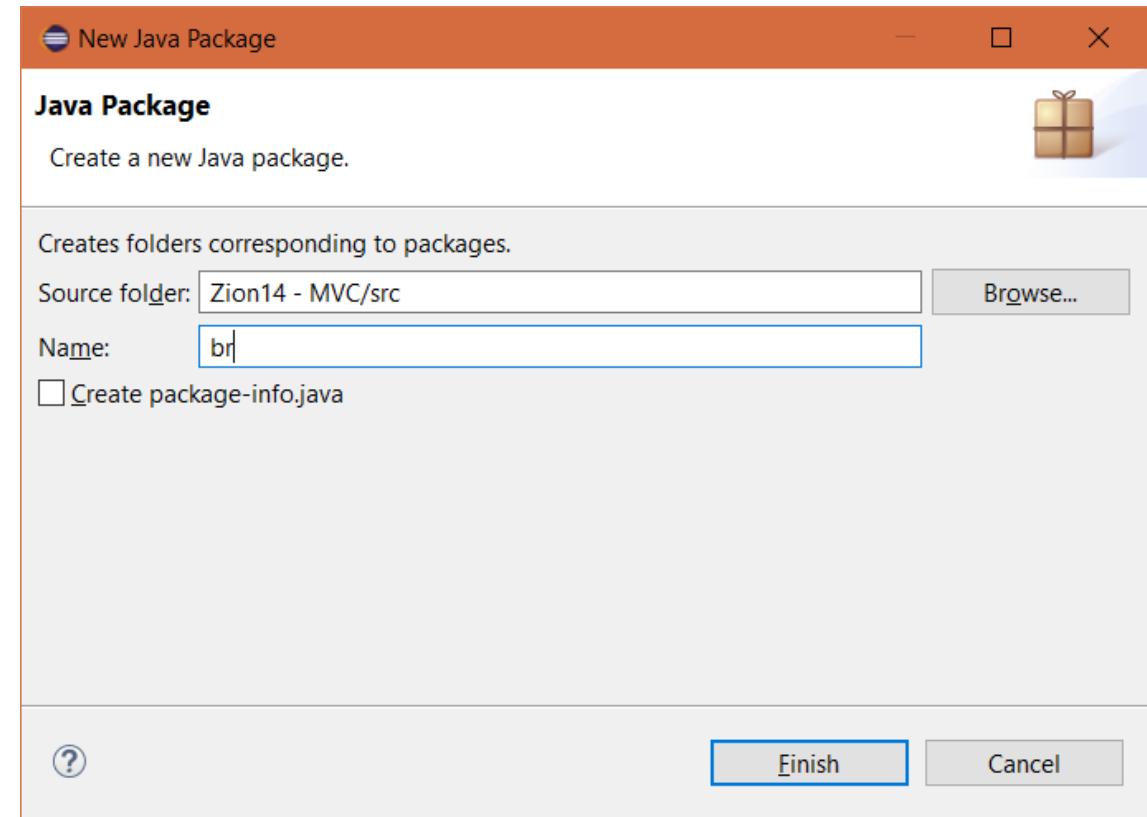
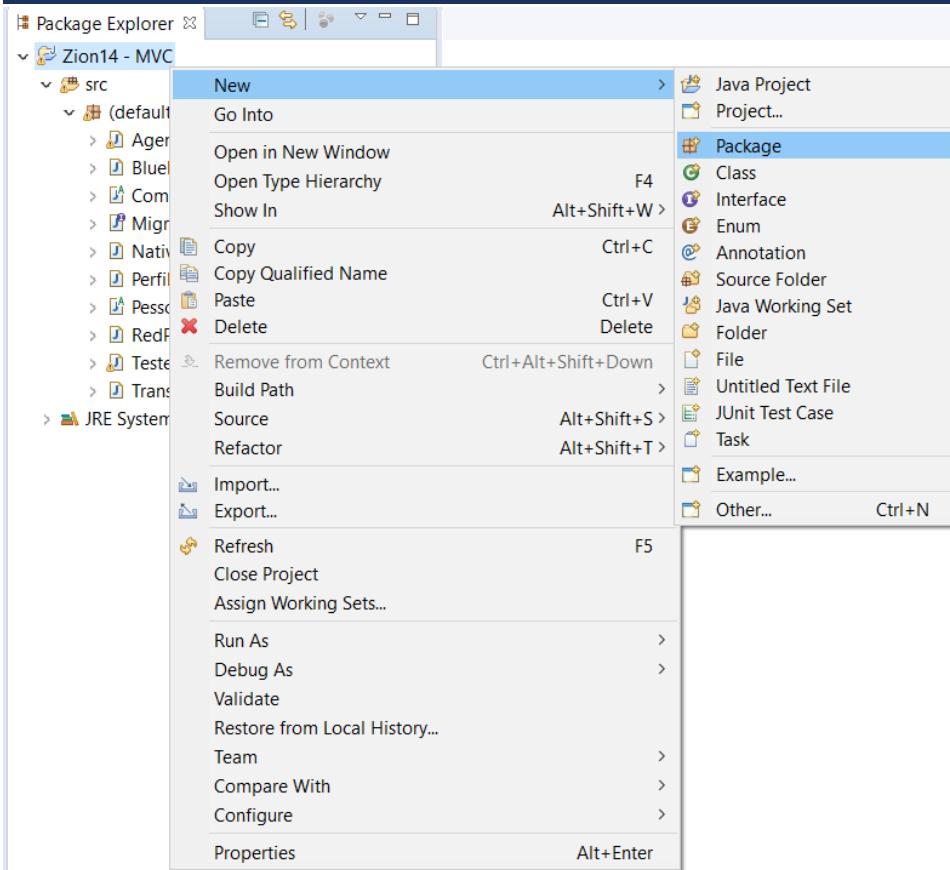
- Os pacotes são utilizados para organização de classes que são agrupadas por finalidade/funcionalidade (possibilitando também sua importação e reutilização) e é o recurso empregado para definir as “camadas” no modelo MVC.
- De acordo com as convenções de nomenclatura de pacotes para garantir nomes únicos utiliza-se o domínio da internet da instituição ou empresa na ordem inversa, por exemplo, metacortex.com.br ficaria br.com.metacortex sendo que a cada ponto (.) é definido uma camada ou sub-pacote (inclusive na organização de pastas e subpastas do projeto). A partir do domínio novas camadas podem ser criadas de acordo com a necessidade e estrutura do projeto, por exemplo, os sub-pacotes model, view e controller.

# PROJETO ZION

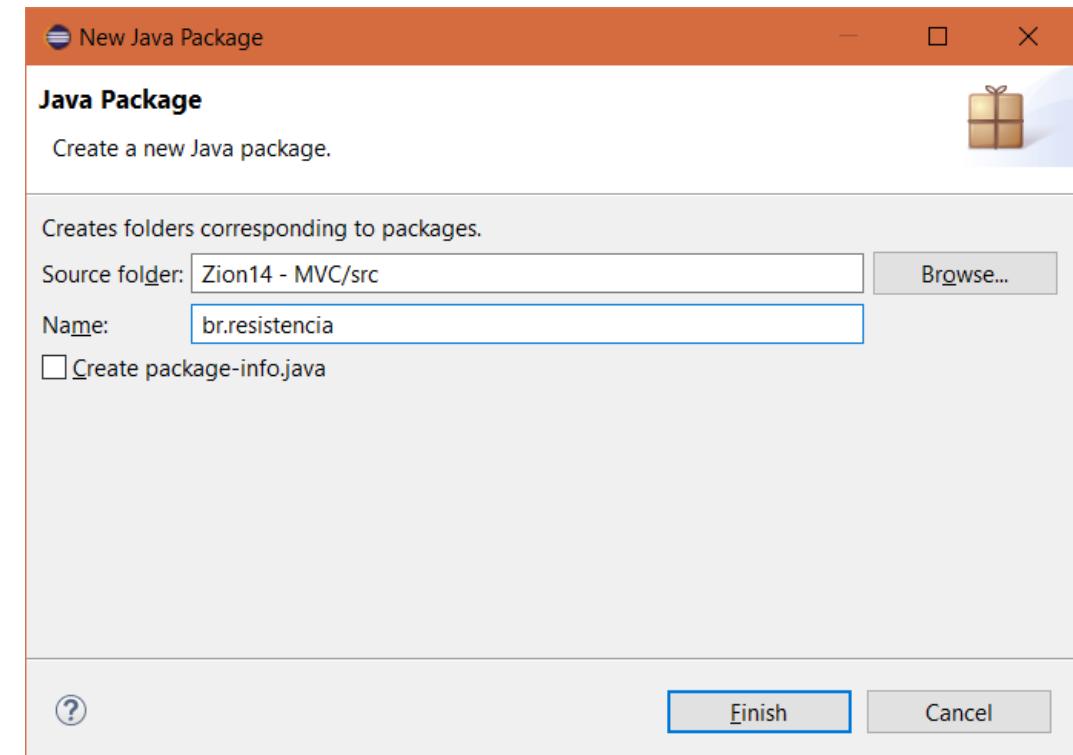
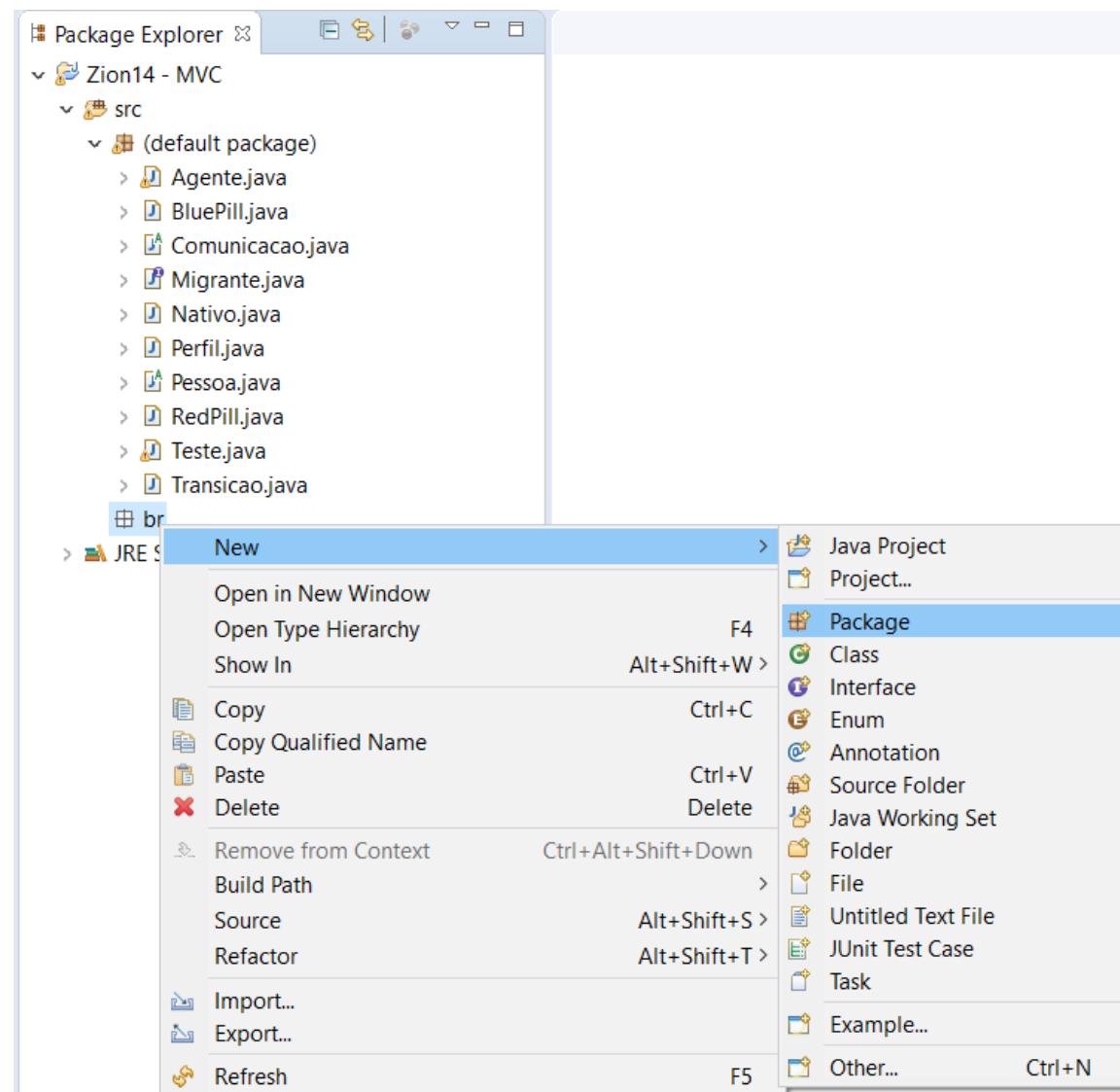
- Seguindo o contexto do universo Matrix utilizaremos o domínio resistencia.br e os sub-pacotes de acordo com o padrão MVC.
- Em UML teremos os seguintes pacotes.



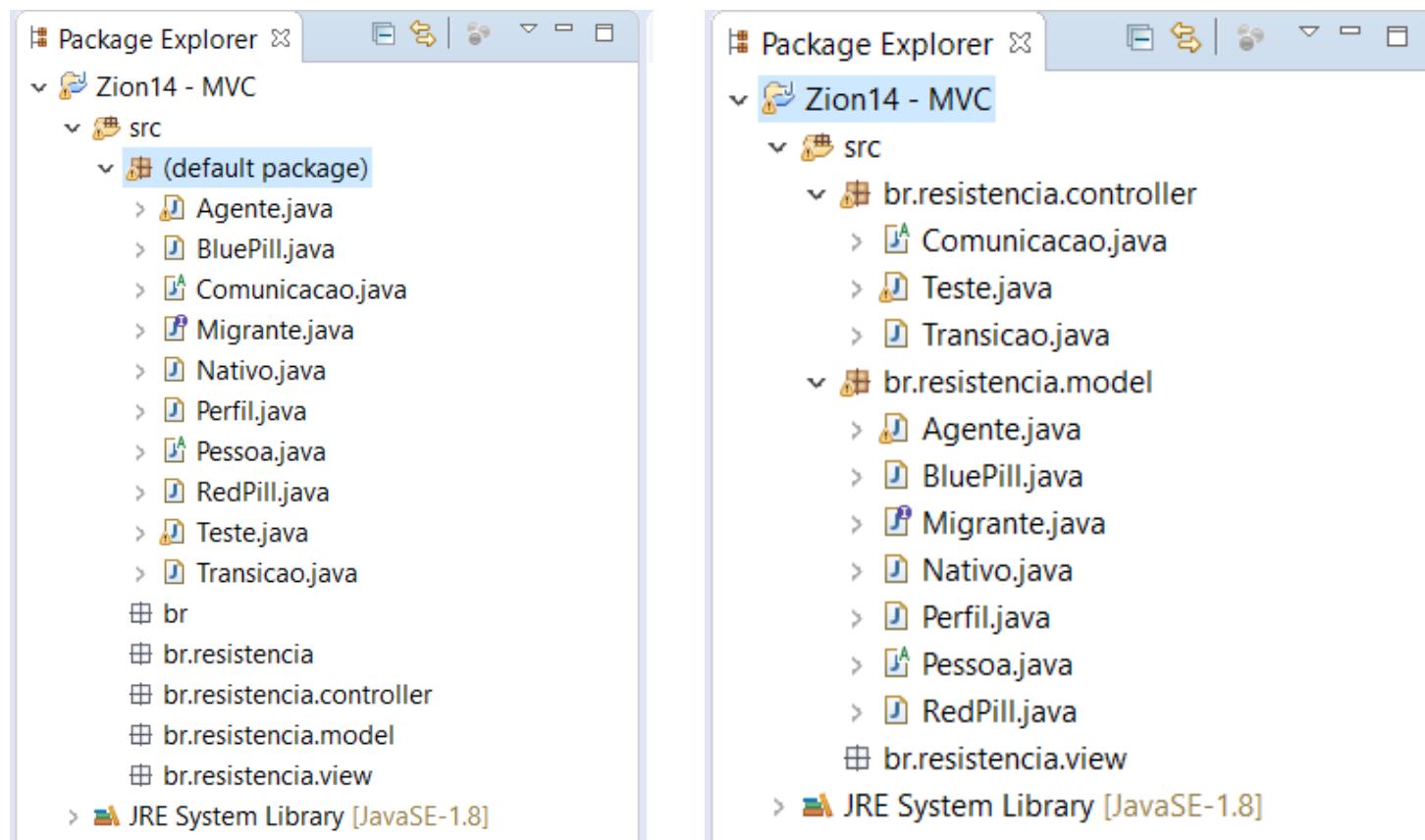
# PACOTE (PACKAGE)



# PACOTE (PACKAGE)



# PACOTE (PACKAGE)



- Para mover as classes para seus pacotes selecione, arraste e solte sobre o pacote.

# PACOTE (PACKAGE)

The screenshot shows the Eclipse IDE interface. On the left, the 'Package Explorer' view displays a project structure for 'Zion14 - MVC'. It includes a 'src' folder containing packages 'br.resistencia.controller' (with files 'Comunicacao.java', 'Teste.java', and 'Transicao.java') and 'br.resistencia.model' (with files 'Agente.java', 'BluePill.java', 'Migrante.java', 'Nativo.java', 'Perfil.java', 'Pessoa.java', and 'RedPill.java'). There is also a 'br.resistencia.view' folder and a 'JRE System Library [JavaSE-1.8]'. On the right, the 'Teste.java' code editor is open, showing the following Java code:

```
6 package br.resistencia.controller;
7
8 import javax.swing.JOptionPane;
9
10 import br.resistencia.model.Agente;
11 import br.resistencia.model.BluePill;
12 import br.resistencia.model.RedPill;
13
14 public class Teste {
15
16     public static void main(String[] args) {
17
18         // Criação dos objetos
19         RedPill redPill = new RedPill("Anderson");
20         BluePill bluePill = new BluePill("Cypher");
21         Agente agente = new Agente("Smith");
22
23         // Passagem de dois objetos que implementam a Interface Migrante
24         JOptionPane.showMessageDialog(null, Transicao.acessarPlataforma(redPill, "Matrix"));
25         JOptionPane.showMessageDialog(null, Transicao.acessarPlataforma(agente, "Zion"));
26     }
27 }
```

- A referência ao pacote que a classe pertence é inserida automaticamente (linha 6).
- Caso seja utilizada uma classe pertencente a outro pacote os imports devem ser atualizados (menu Source> Organize imports ou **ctrl+shift-O**), como por exemplo, nas linhas 10, 11 e 12.

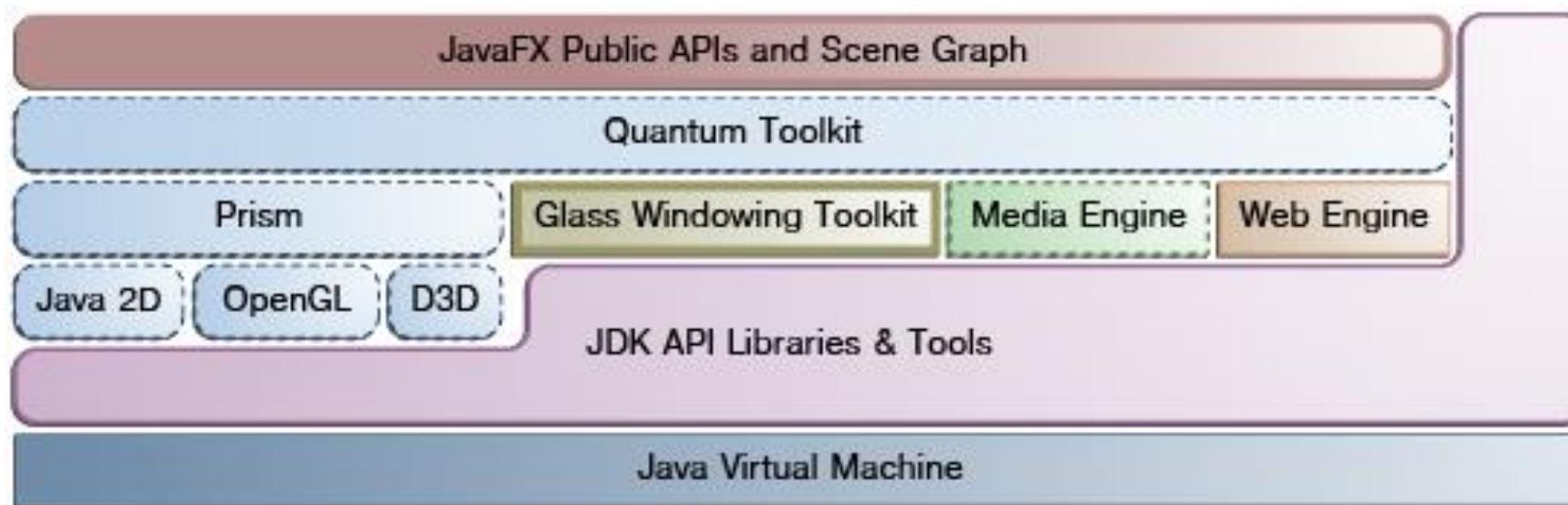
# JAVAFX

- A plataforma JavaFX permite o desenvolvimento de aplicações RIA (Rich Internet Applications) que se comportam de forma consistente em várias plataformas.
- O JavaFX permite a utilização das bibliotecas Java (back-end) no desenvolvimento de interfaces com experiências visuais envolventes (front-end).

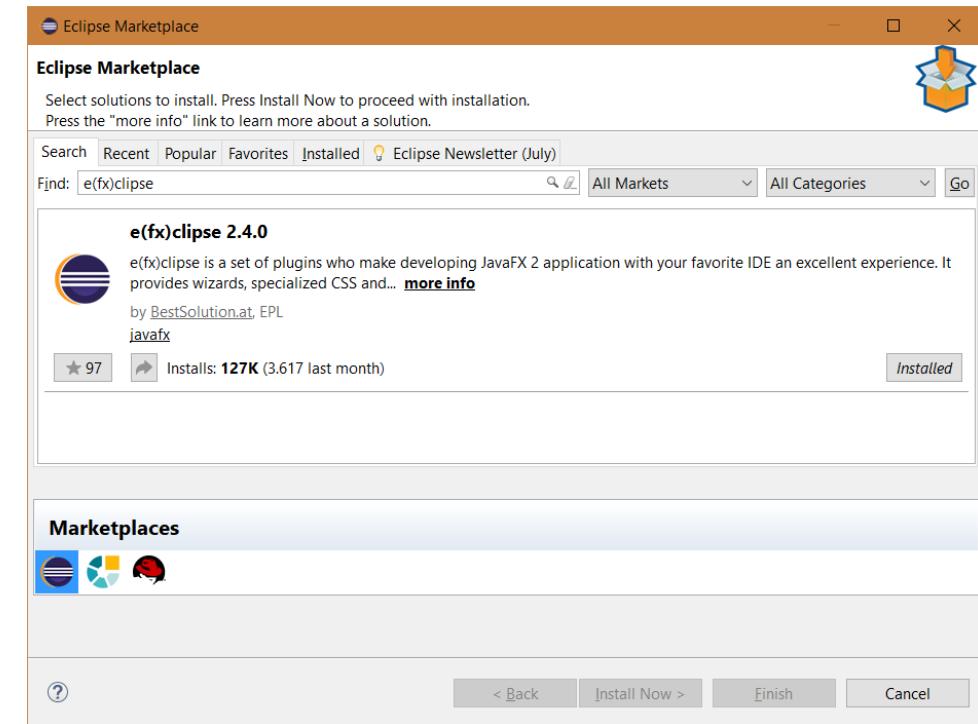
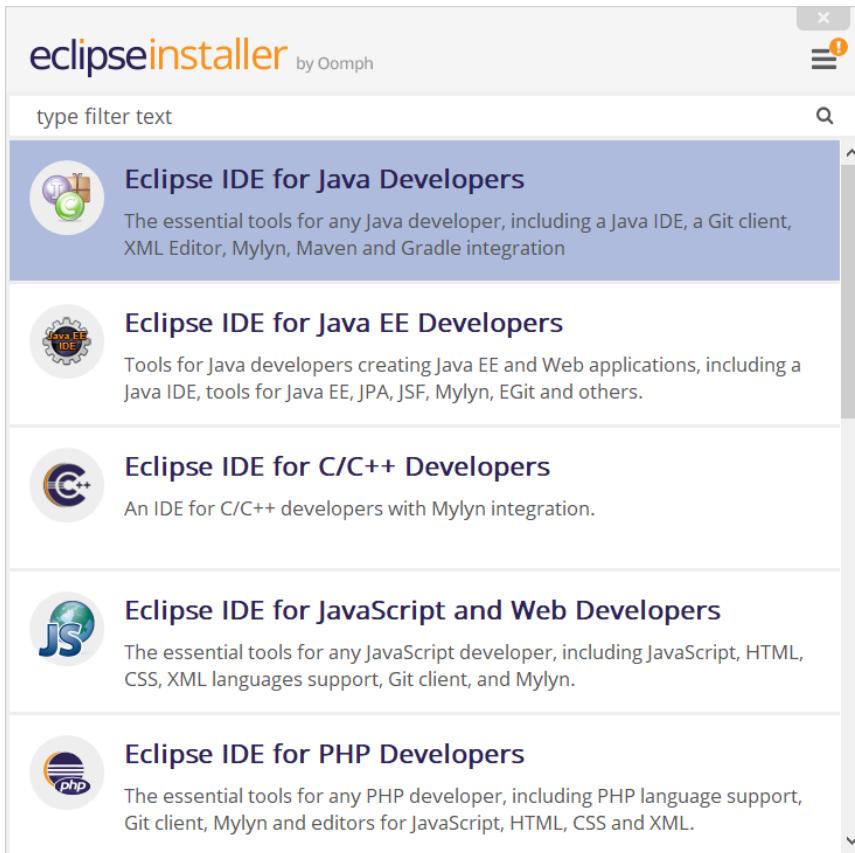


# JAVAFX

## ■ Arquitetura



# INSTALAÇÃO



# SCENE BUILDER (EDITOR DE .FXML)

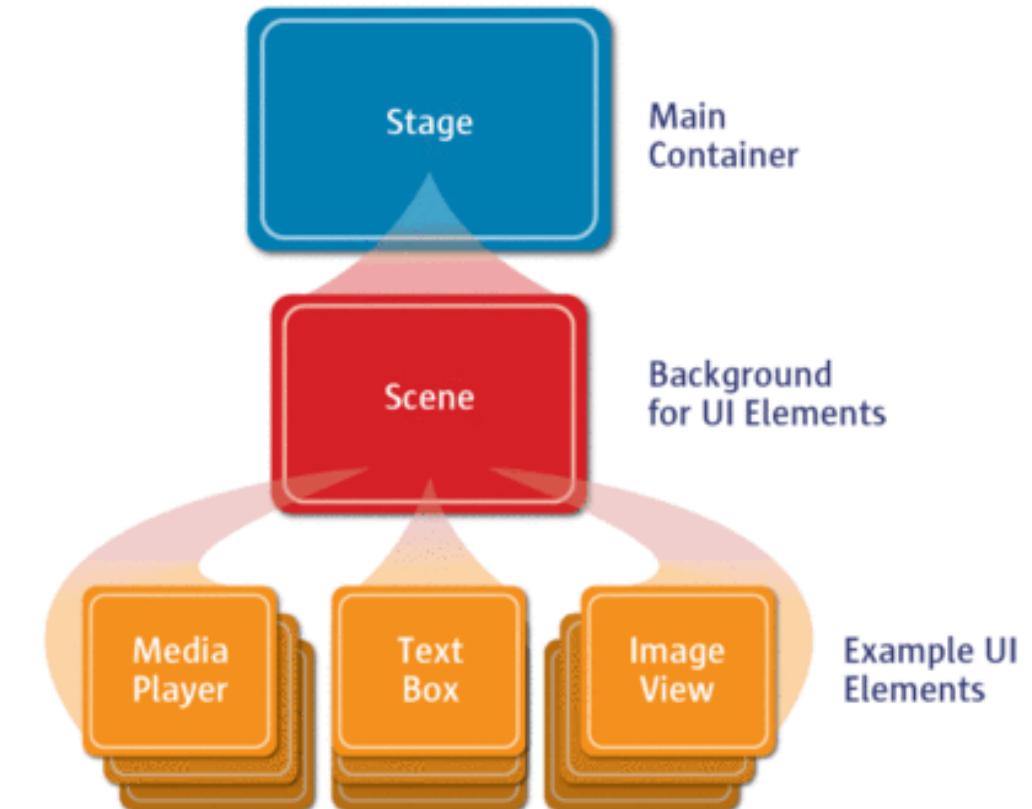
The screenshot shows a web browser displaying the Gluon website at [gluonhq.com/products/scene-builder/](http://gluonhq.com/products/scene-builder/). The page title is "Scene Builder - Gluon". The main content is titled "Download Scene Builder" and notes that the latest version is 8.3.0, released on Dec 16, 2016. It encourages users to subscribe to the [Gluon Newsletter](#). Below this, a table lists download links for various platforms:

Product	Platform	Download
Scene Builder	Executable Jar	<a href="#">Download</a>
Scene Builder	Windows Installer (x86) <a href="#">info</a>	<a href="#">Download</a>
Scene Builder	Windows Installer (x64) <a href="#">info</a>	<a href="#">Download</a>
Scene Builder	Mac OS X dmg	<a href="#">Download</a>
Scene Builder	Linux RPM (64-bit)	<a href="#">Download</a>
Scene Builder	Linux Deb (64-bit)	<a href="#">Download</a>
Scene Builder	Linux RPM (32-bit)	<a href="#">Download</a>
Scene Builder	Linux Deb (32-bit)	<a href="#">Download</a>
Scene Builder Kit <a href="#">info</a>	Jar File	<a href="#">Download</a>

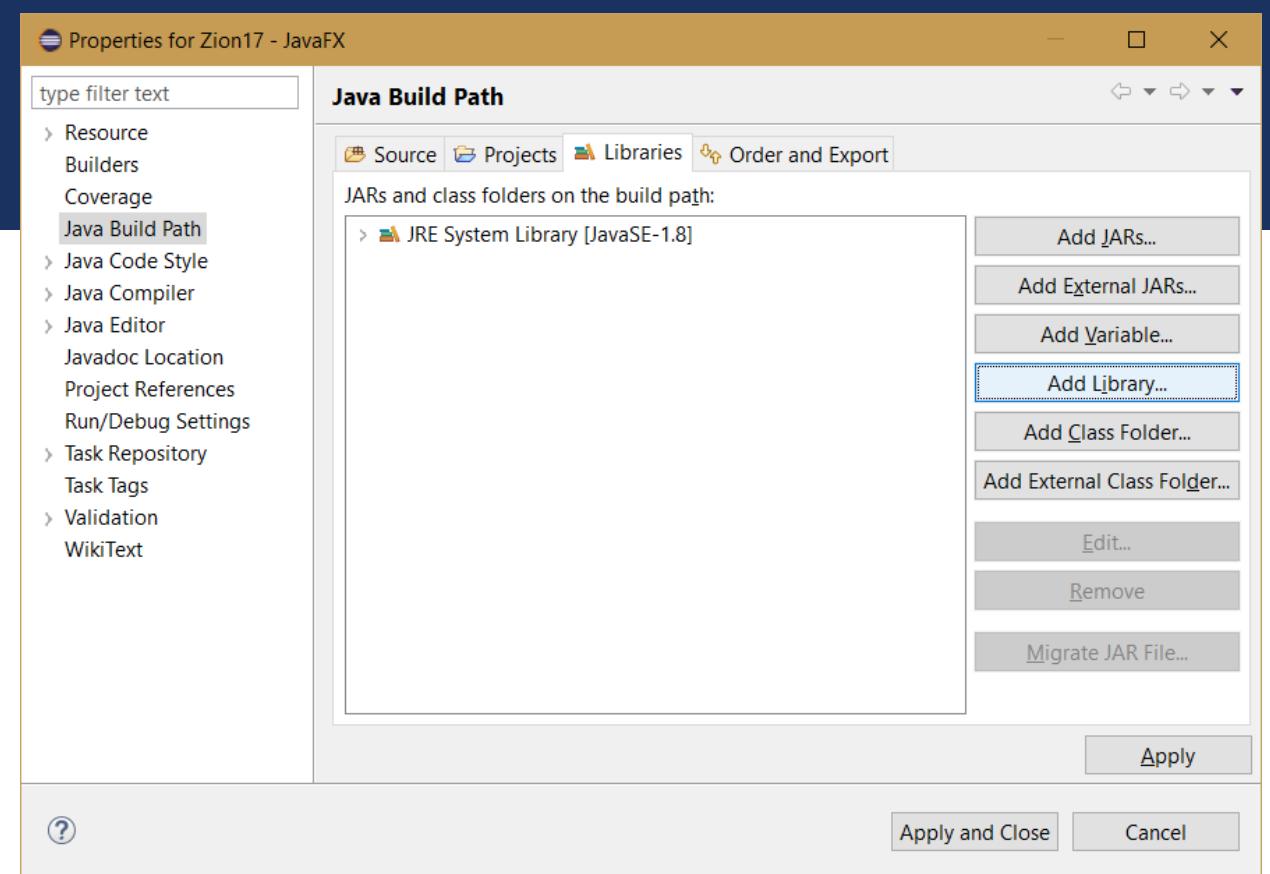
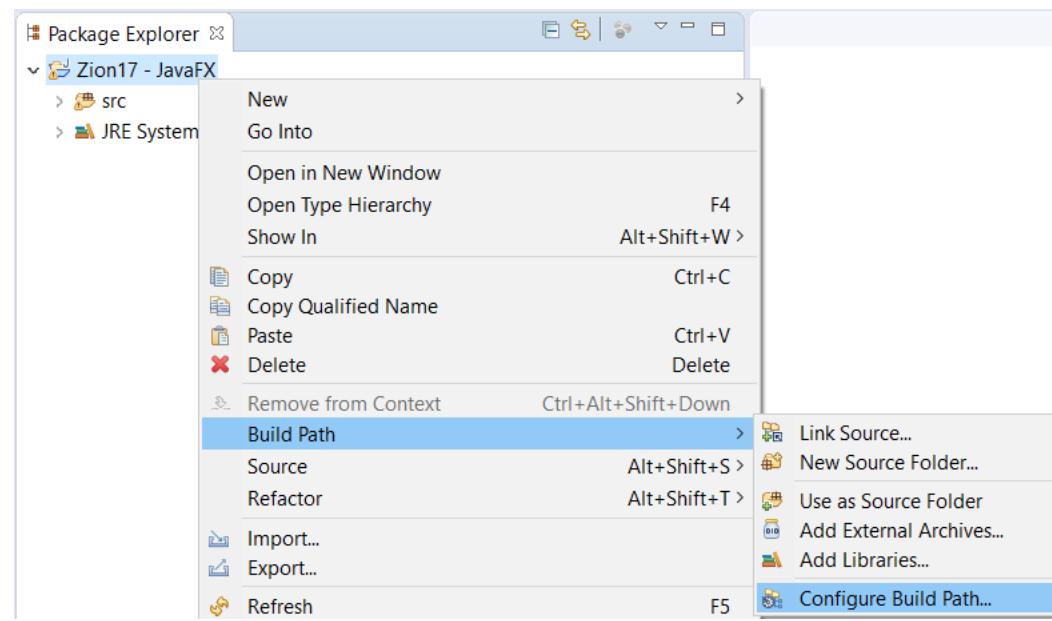
License: Scene Builder is licensed under the BSD license.

# JAVAFX

- Stage (palco): área de exibição (formulário).
- Scene (cena): controles inseridos e organizados.
- Nodes (nó): cada elemento da interface (controles). Organizados a partir de um Node Root (nó raiz) que será um Layout Pane (Gerenciador de layout).



# SDK JAVAFX



- Para o projeto ter acesso as classes do JavaFX a referência ao SDK JavaFX deve ser inserida por meio do Builder Path.

# SDK JAVAFX

The image displays two side-by-side screenshots of Java build path configuration interfaces.

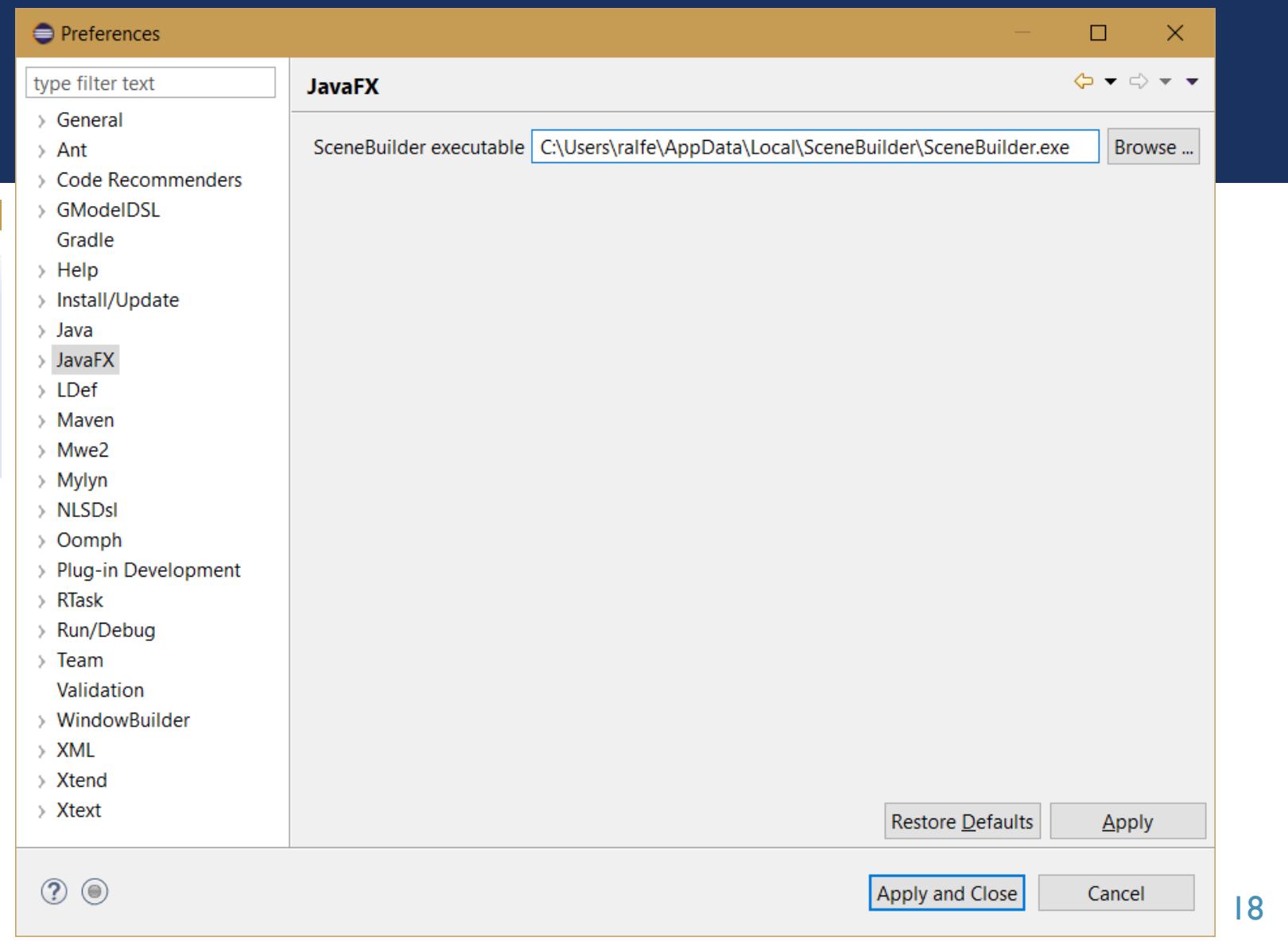
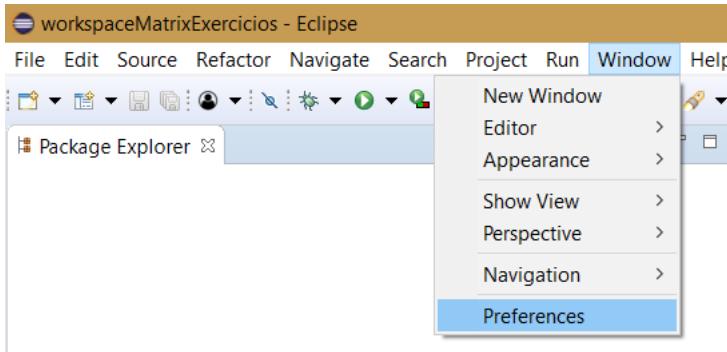
**Left Screenshot: Add Library Dialog**

This dialog is titled "Add Library". It contains a list of library types under the heading "Select the library type to add." The "JavaFX SDK" option is highlighted with a blue selection bar at the top of the list. Other options include "JRE System Library", "JUnit", "Maven Managed Dependencies", "Mobile SDK", "Plug-in Dependencies", "User Library", and "Xtend Library". At the bottom of the dialog are buttons for "?", "< Back", "Next >" (which is highlighted with a blue border), "Finish", and "Cancel".

**Right Screenshot: Properties for Zion17 - JavaFX Dialog**

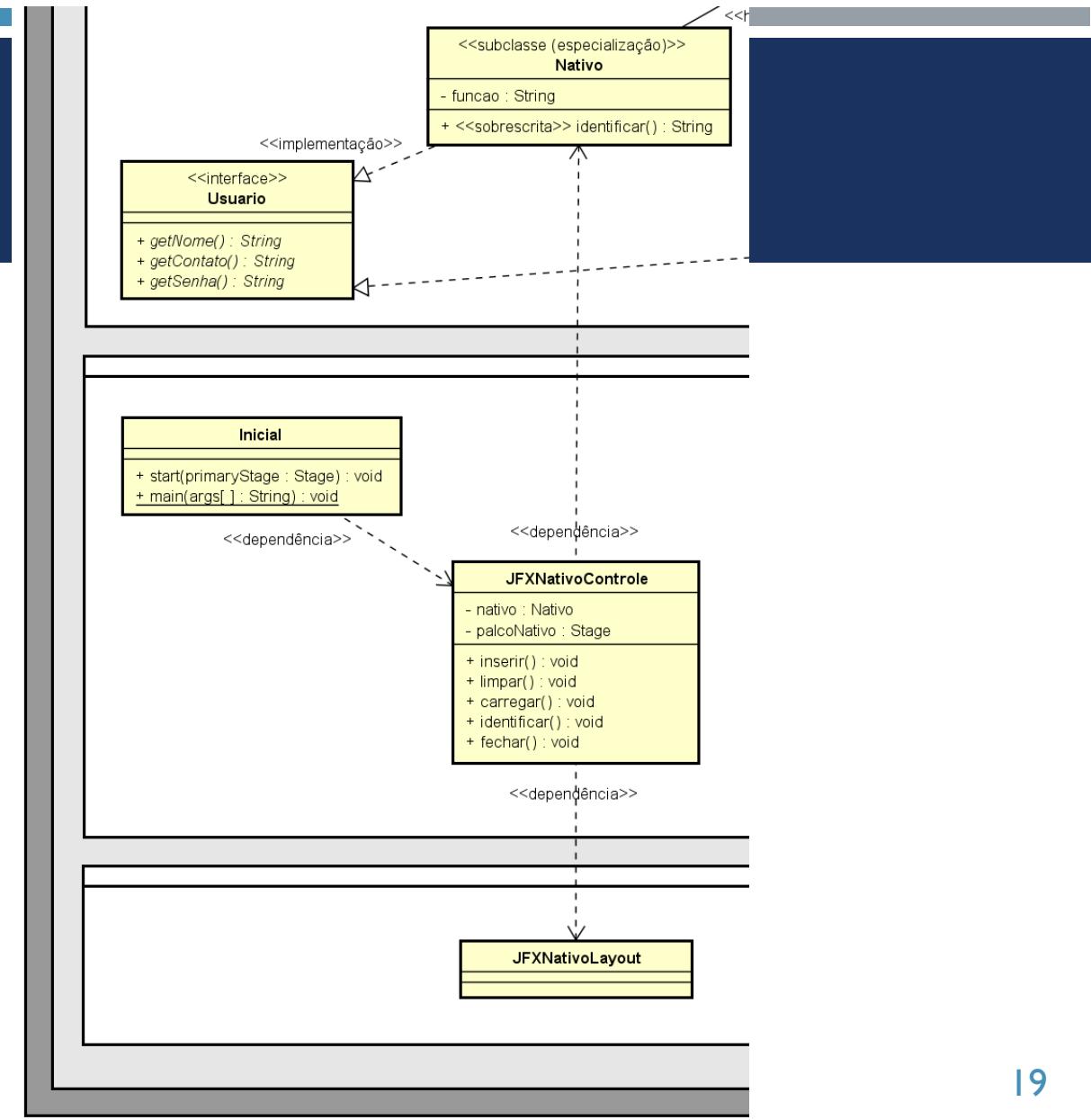
This dialog is titled "Properties for Zion17 - JavaFX". The left pane shows a tree view of project properties with "Java Build Path" selected. The right pane is titled "Java Build Path" and shows the "Libraries" tab selected. Under "JARs and class folders on the build path:", there are entries for "JavaFX SDK" and "JRE System Library [JavaSE-1.8]". To the right of the list are several buttons: "Add JARs...", "Add External JARs...", "Add Variable...", "Add Library..." (which is highlighted with a blue selection bar), "Add Class Folder...", "Add External Class Folder...", "Edit...", "Remove", and "Migrate JAR File...". At the bottom are "Apply" and "Apply and Close" buttons, with "Apply and Close" also having a blue border.

# SCENE BUILDER PELO ECLIPSE



- Para abrir o Scene Builder pelo Eclipse.

# ZION

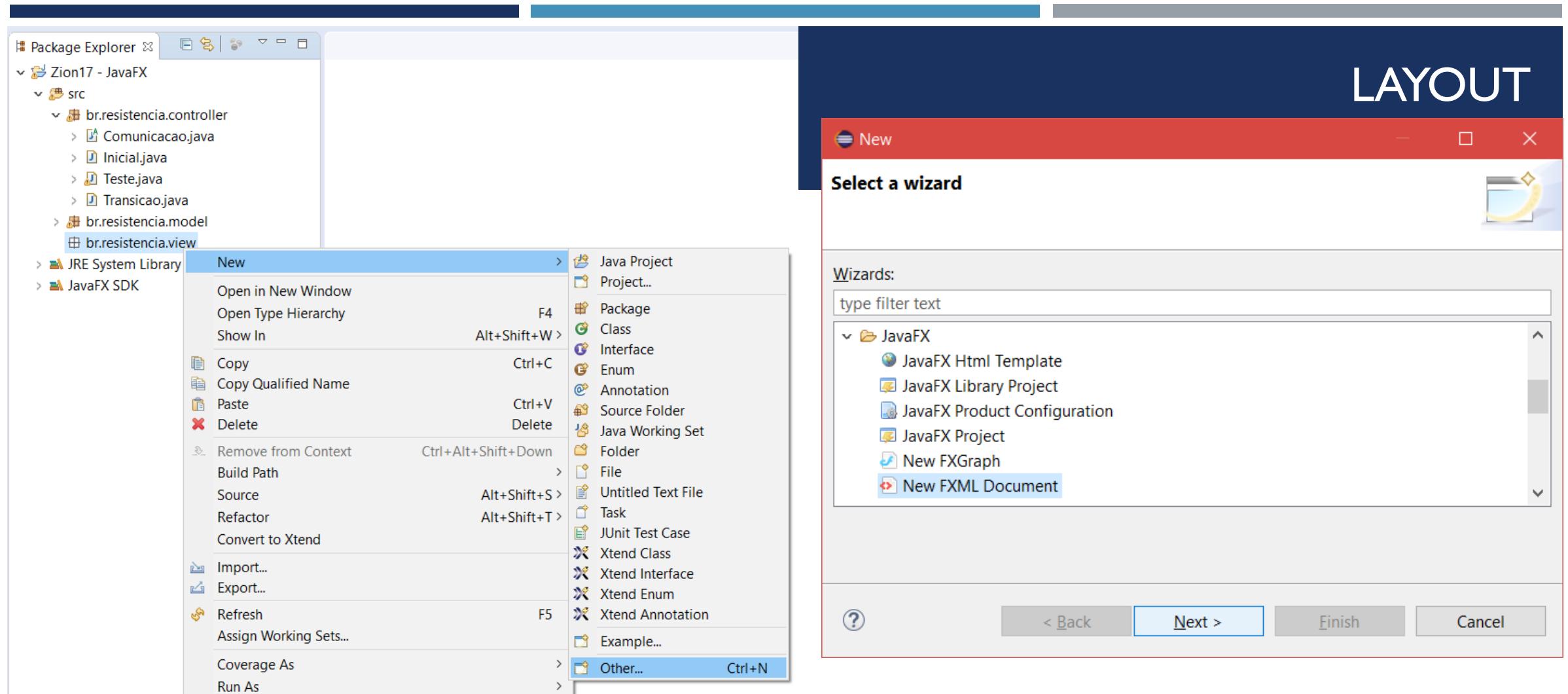


# INICIAL

The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer view displays a project structure for 'Zion17 - JavaFX'. The 'src' folder contains packages 'br.resistencia.controller' and 'br.resistencia.model', along with 'br.resistencia.view' and JavaFX libraries. Inside 'br.resistencia.controller', there are four files: 'Comunicacao.java', 'Inicial.java', 'Teste.java', and 'Transicao.java'. The 'Inicial.java' file is open in the main editor window. The code in 'Inicial.java' is as follows:

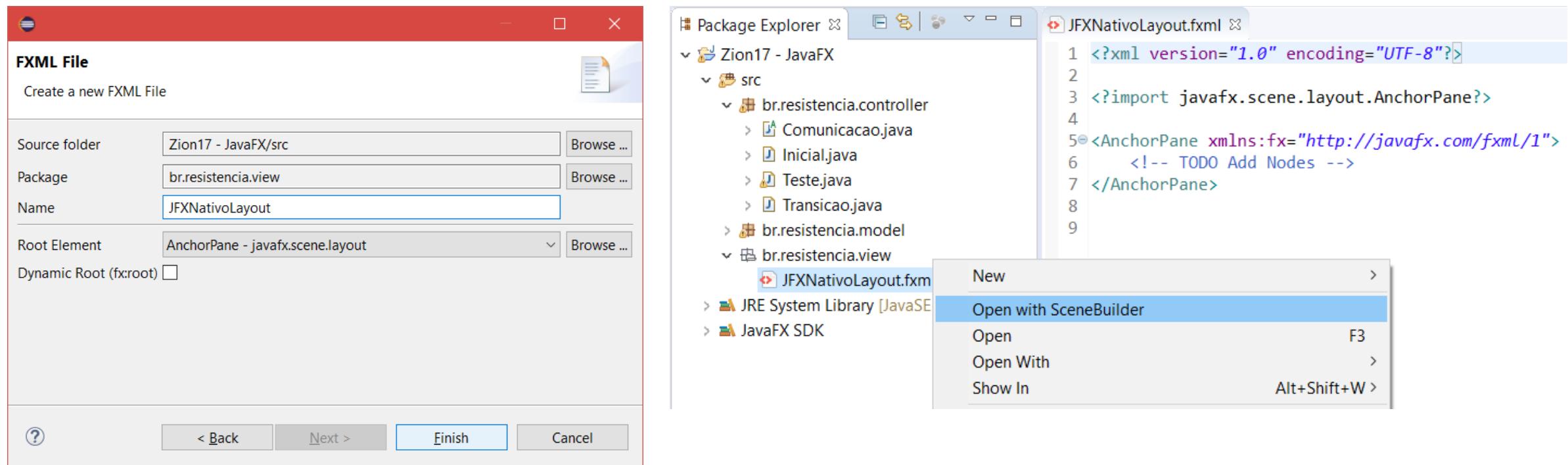
```
2 package br.resistencia.controller;
3
4 import javafx.application.Application;
5 import javafx.stage.Stage;
6
8 * @author Prof. Ralfe
11 public class Inicial extends Application{
12
13     public static void main(String[] args) {
14
15     }
16
17     @Override
18     public void start(Stage primaryStage) throws Exception {
19
20     }
21 }
```

- A classe que inicia uma aplicação com JavaFX deve extender a classe Application que possui o método abstrato (ou seja, obrigatório) start.



## ■ Criação do arquivo .fxml

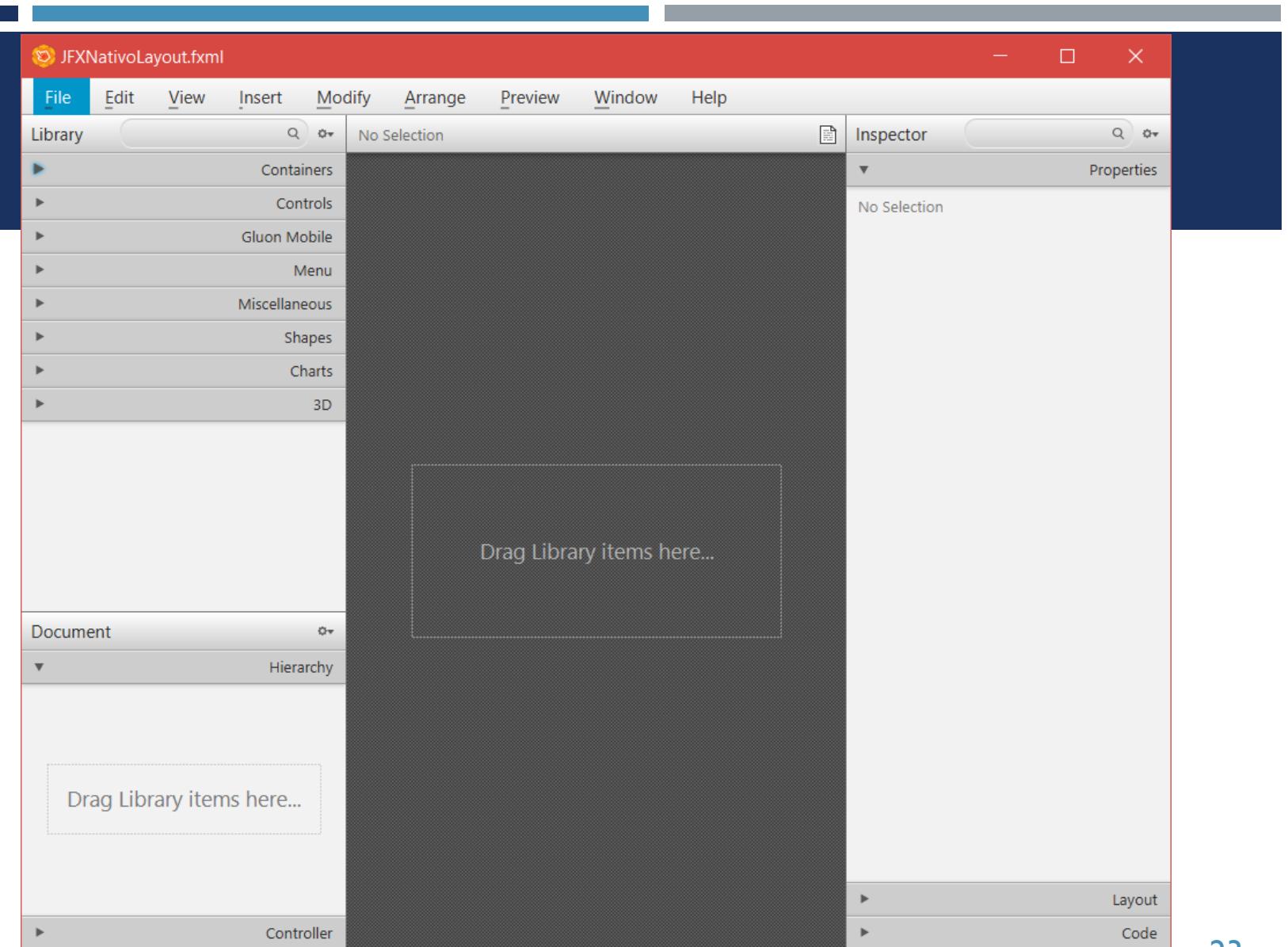
# LAYOUT



- Editar o arquivo pelo Scene Builder.

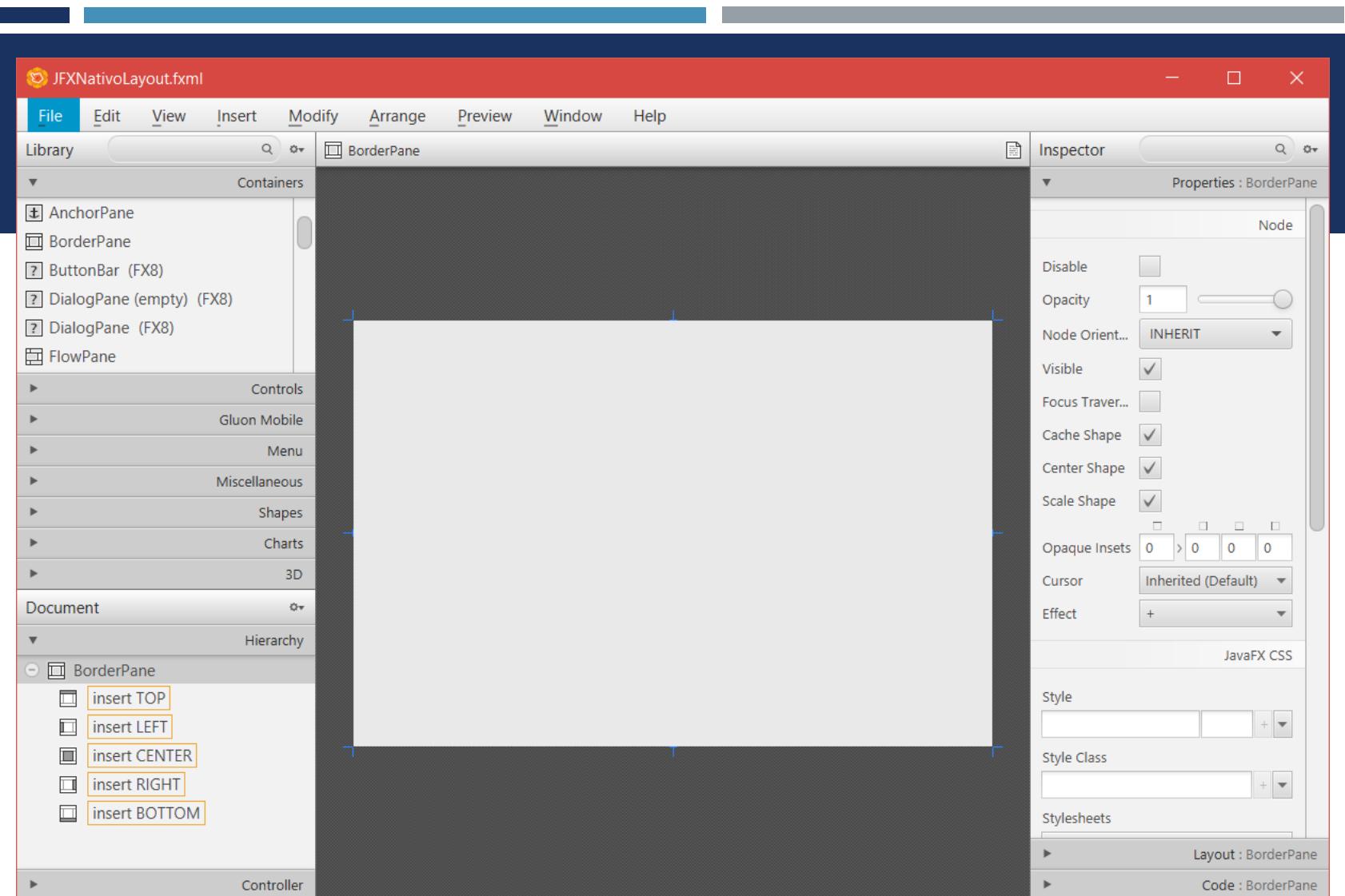
# SCENE BUILDER

- Editor de arquivos .fxml



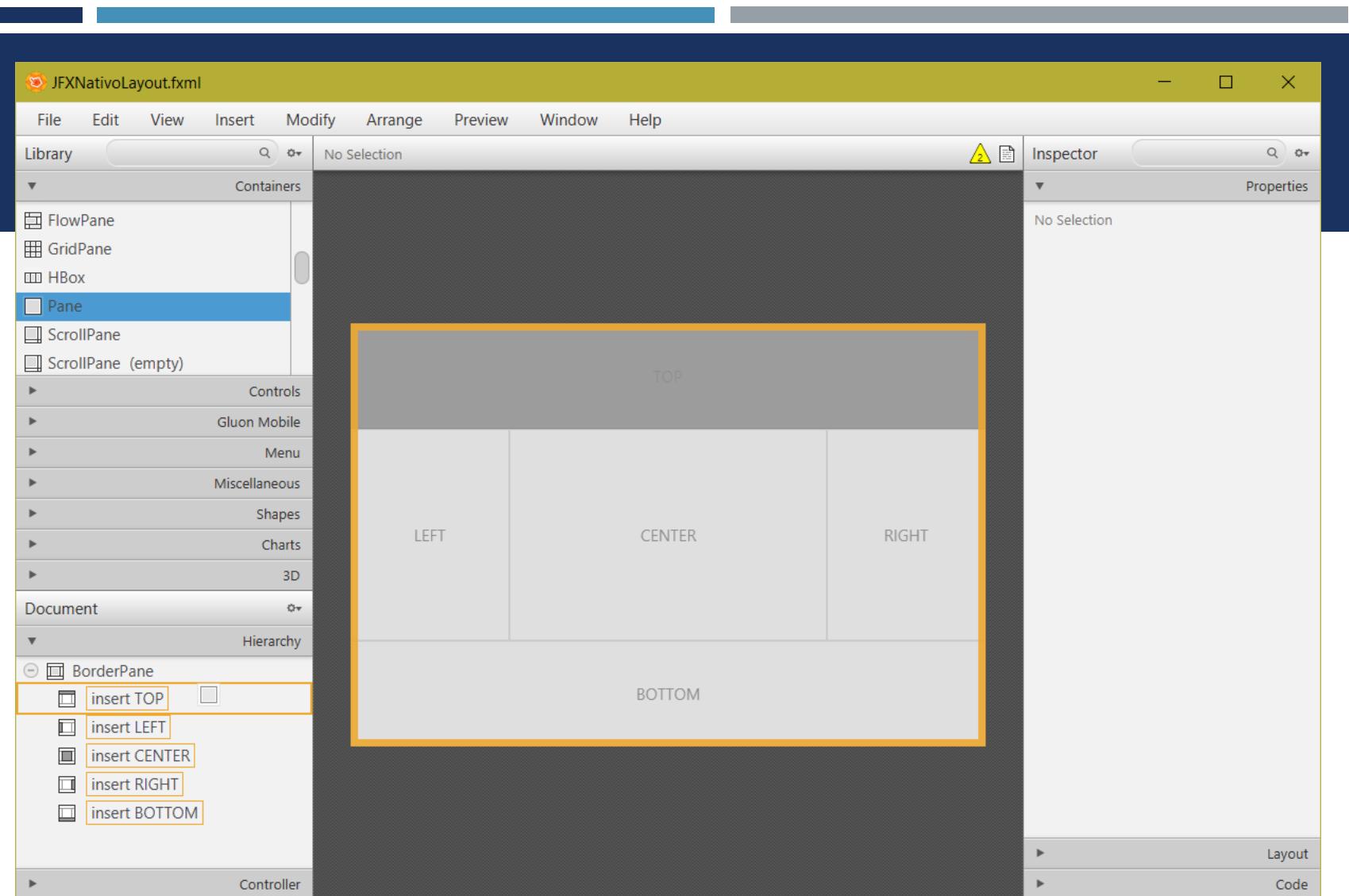
# SCENE BUILDER

- Layout Pane:  
BorderPane  
(paleta Containers)



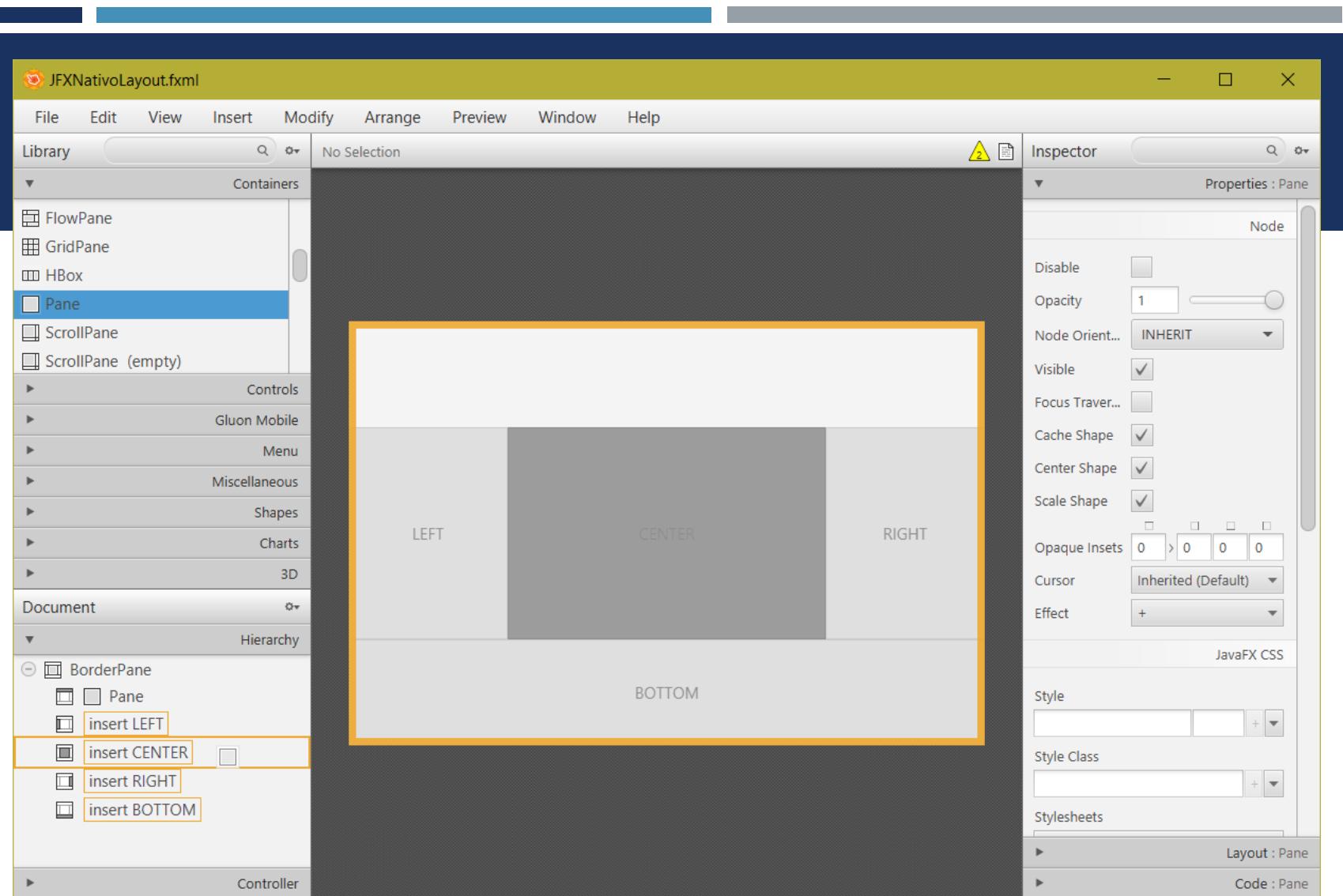
# SCENE BUILDER

- Utilização de Panes nos setores do BorderPane (paleta Containers)



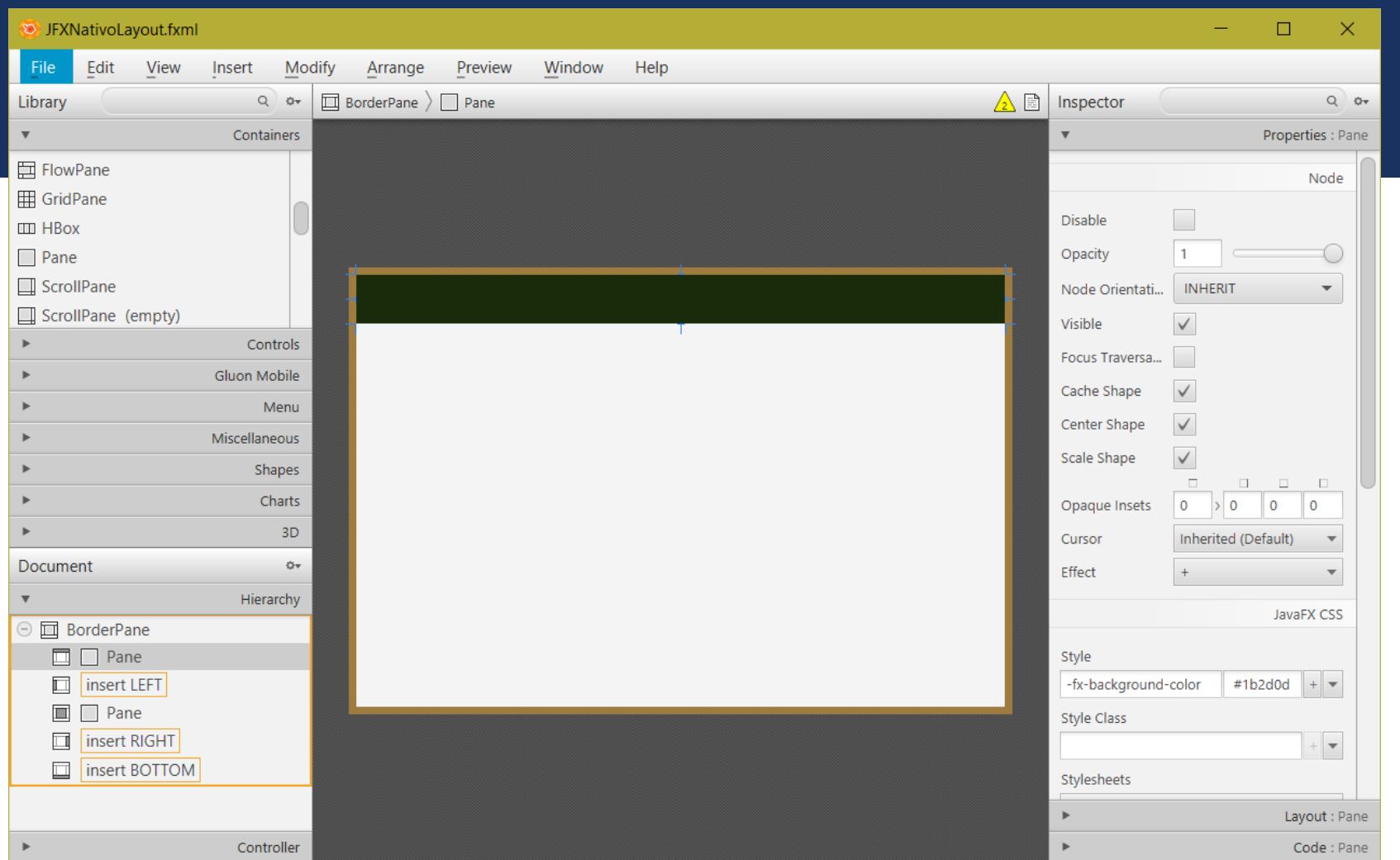
# SCENE BUILDER

- Utilização de Panes nos setores do BorderPane (paleta Containers)



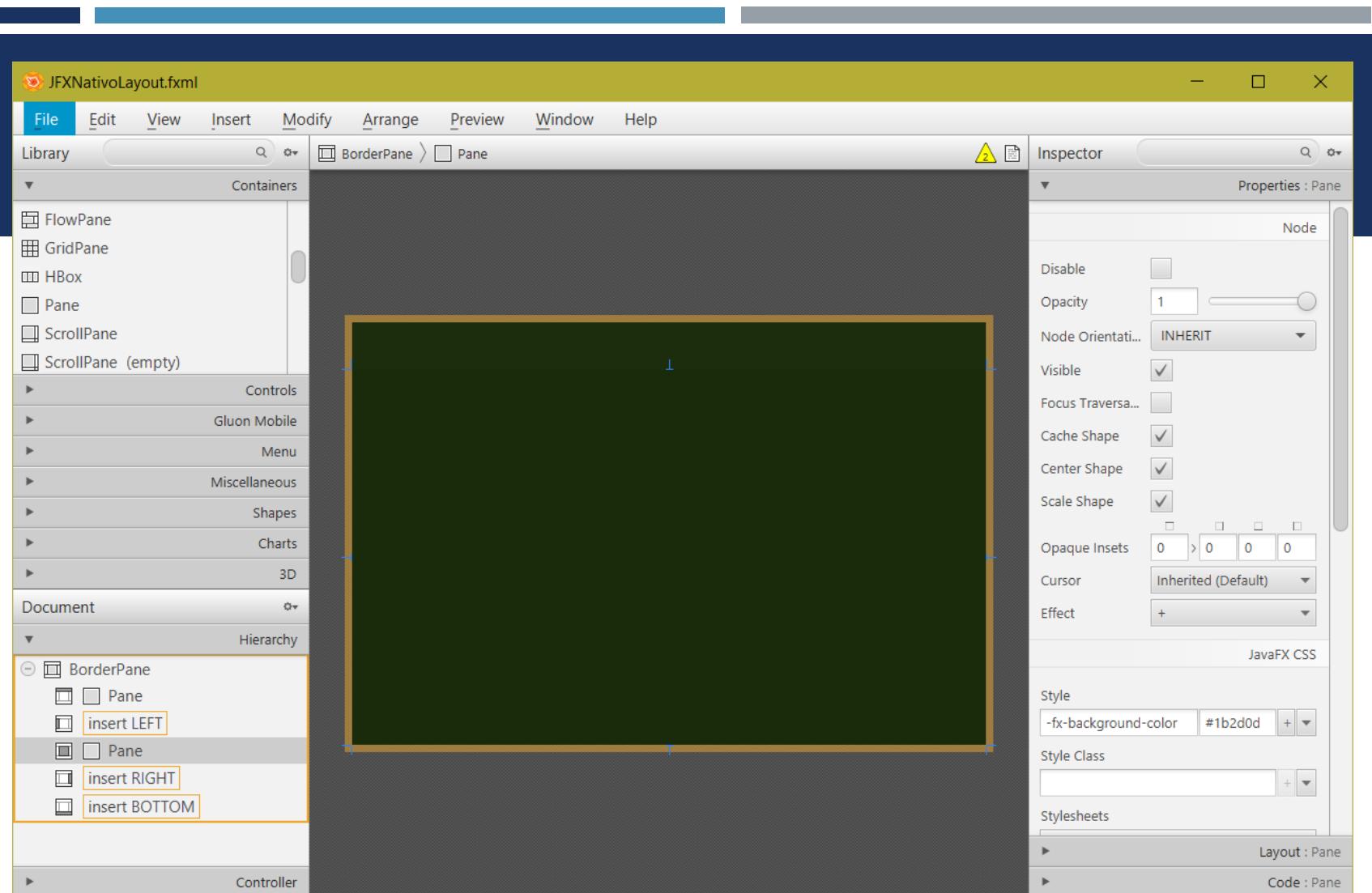
# SCENE BUILDER

- Definição de cores para o Pane (por meio de referências em hexadecimal). (paleta Properties)



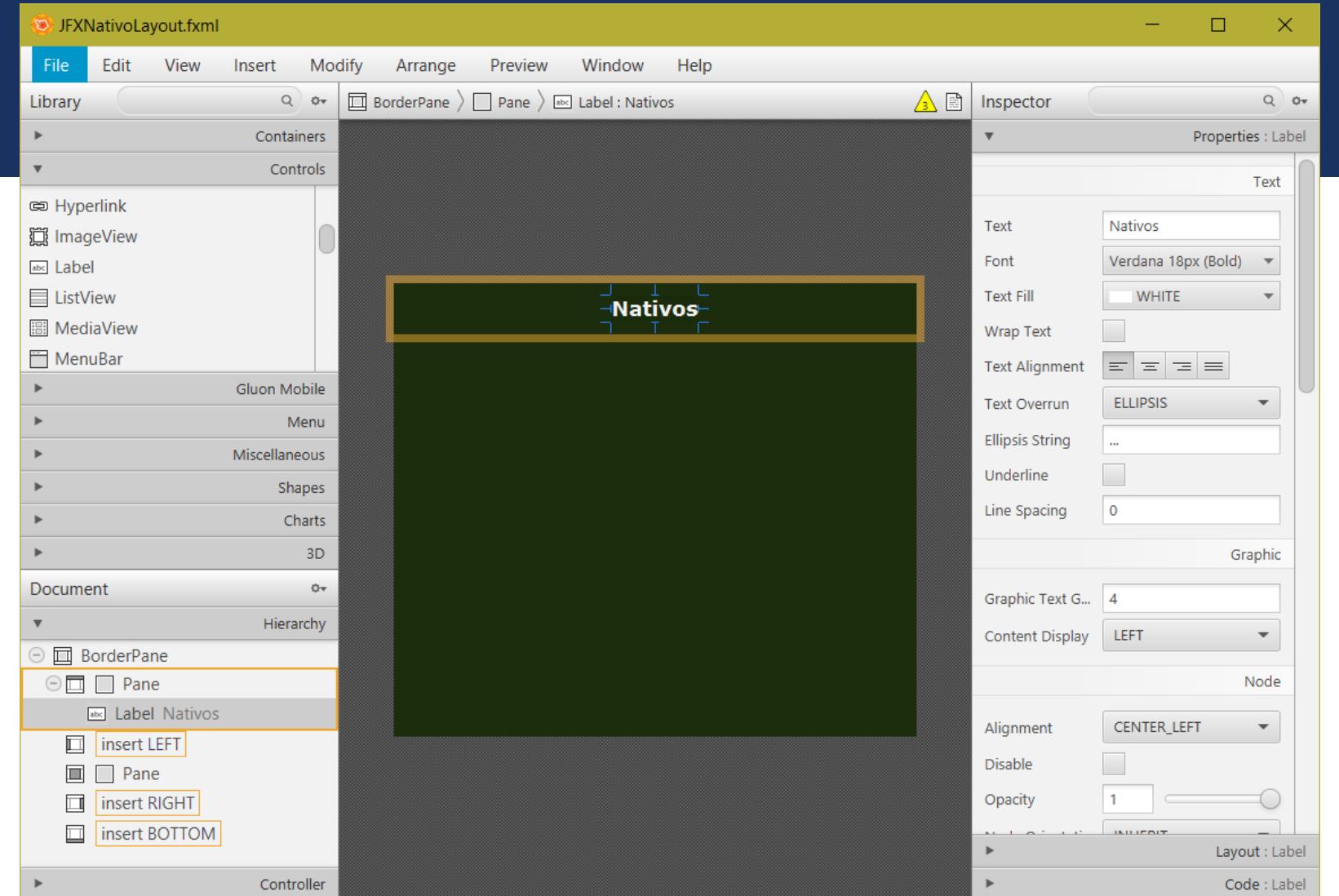
# SCENE BUILDER

- Definição de cores para o Pane (por meio de referências em hexadecimal). (paleta Properties)



# SCENE BUILDER

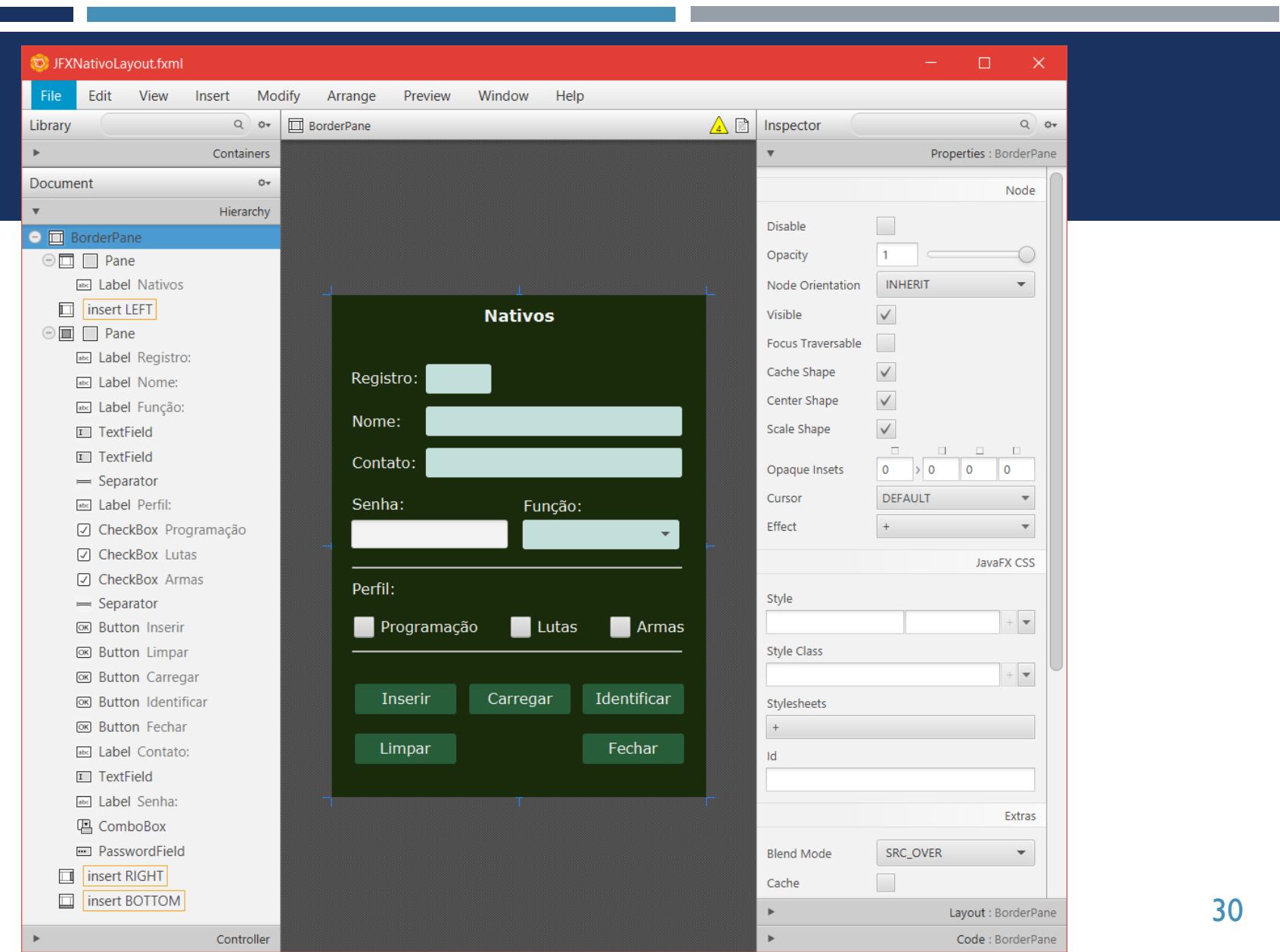
## ■ Paleta Controls



# SCENE BUILDER

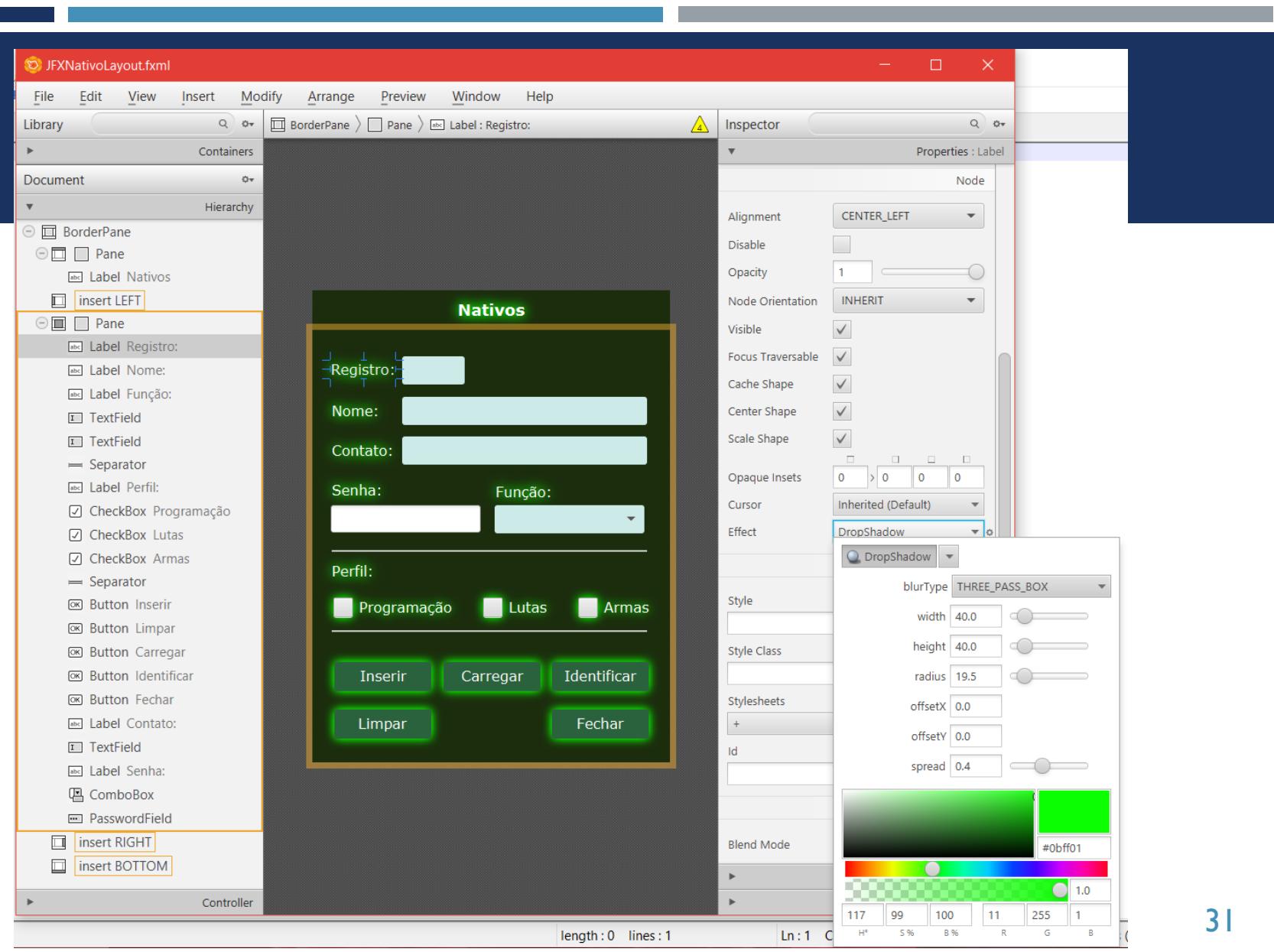
## ■ Paleta Controls

PROF. RALFE DELLA CROCE FILHO



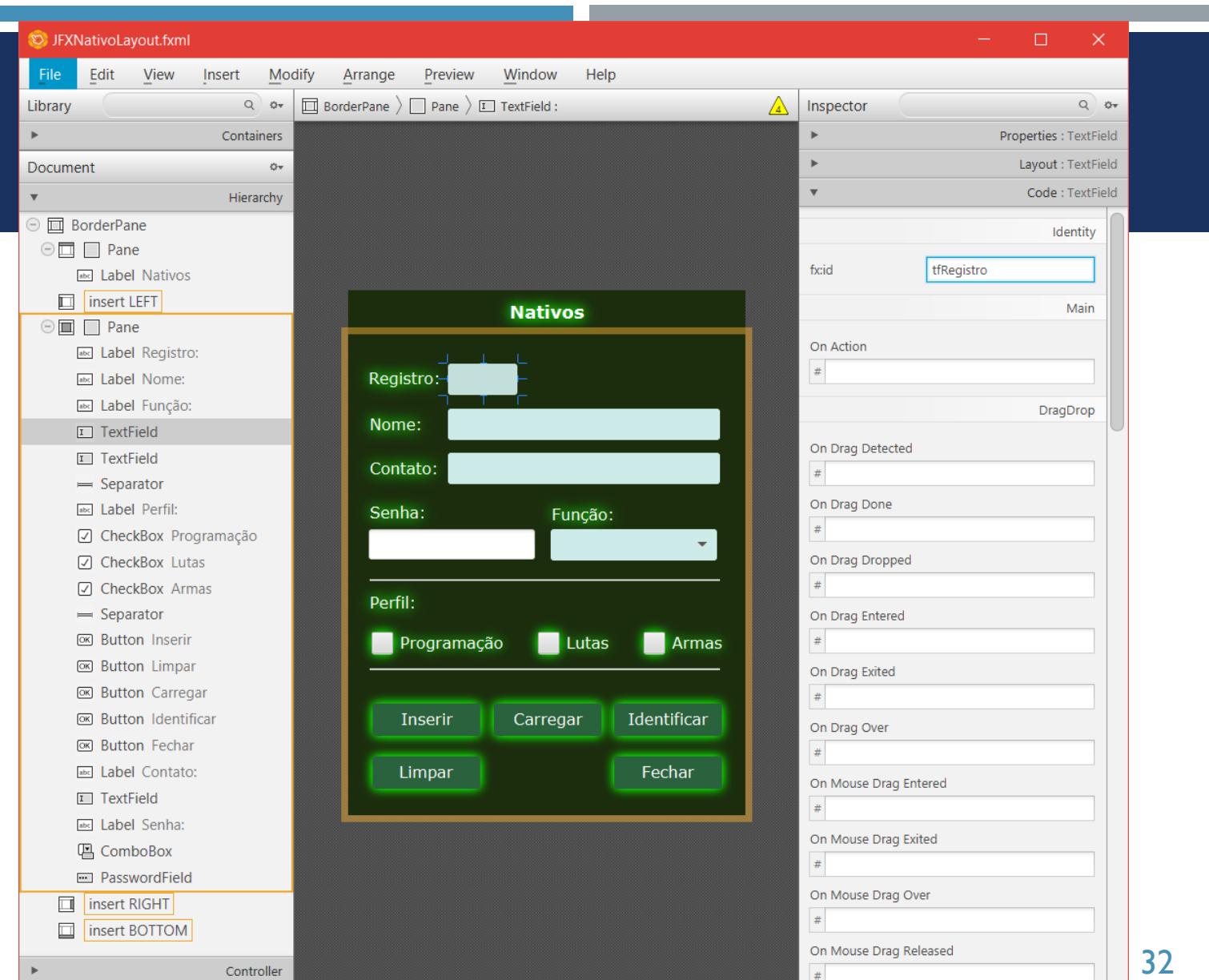
# SCENE BUILDER

## ■ Efeitos (paleta Properties)



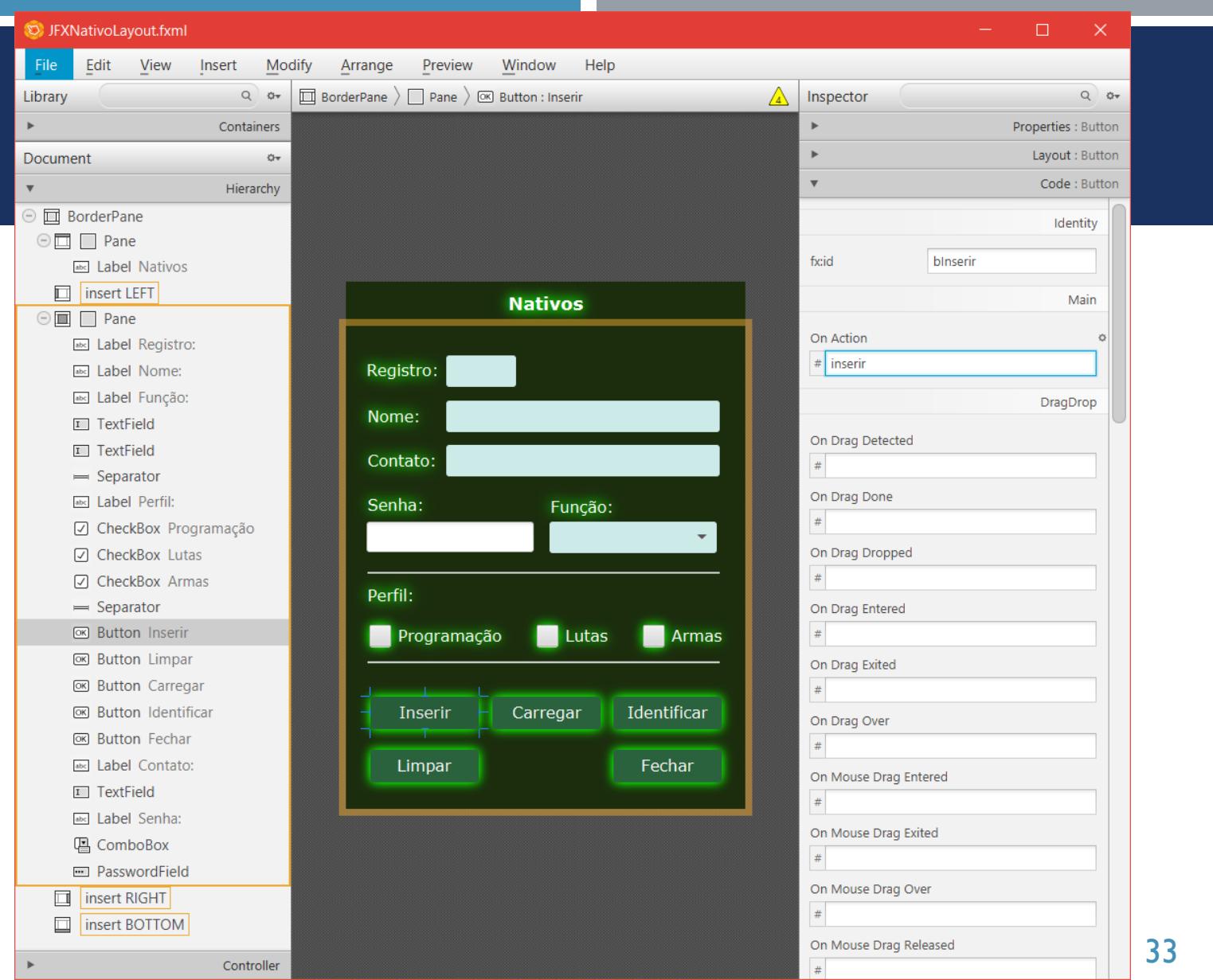
# SCENE BUILDER

- Definição dos identificadores dos controles (paleta Code, campo fx:id) que serão utilizados na programação (classe Controle).

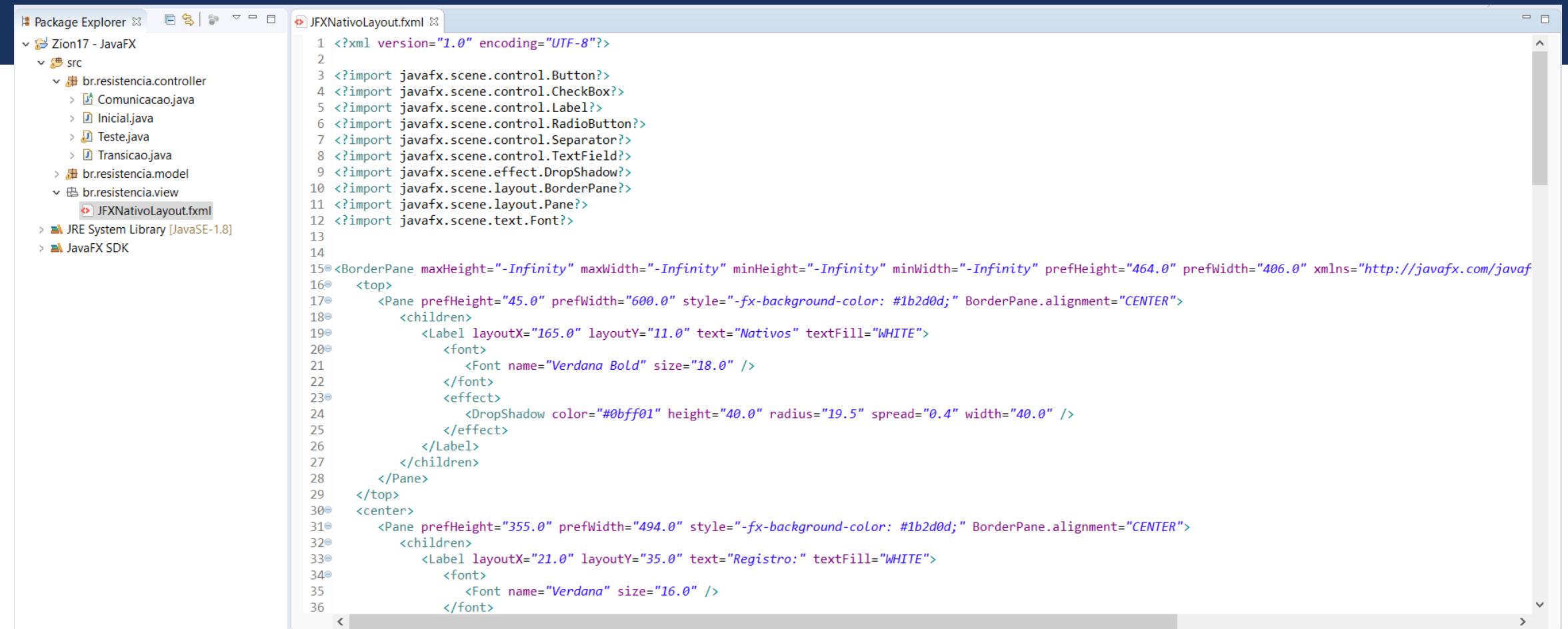


# SCENE BUILDER

- Definição dos eventos dos controles (paleta Code, campo On Action) que serão implementados na programação (classe Controle).



# FXML



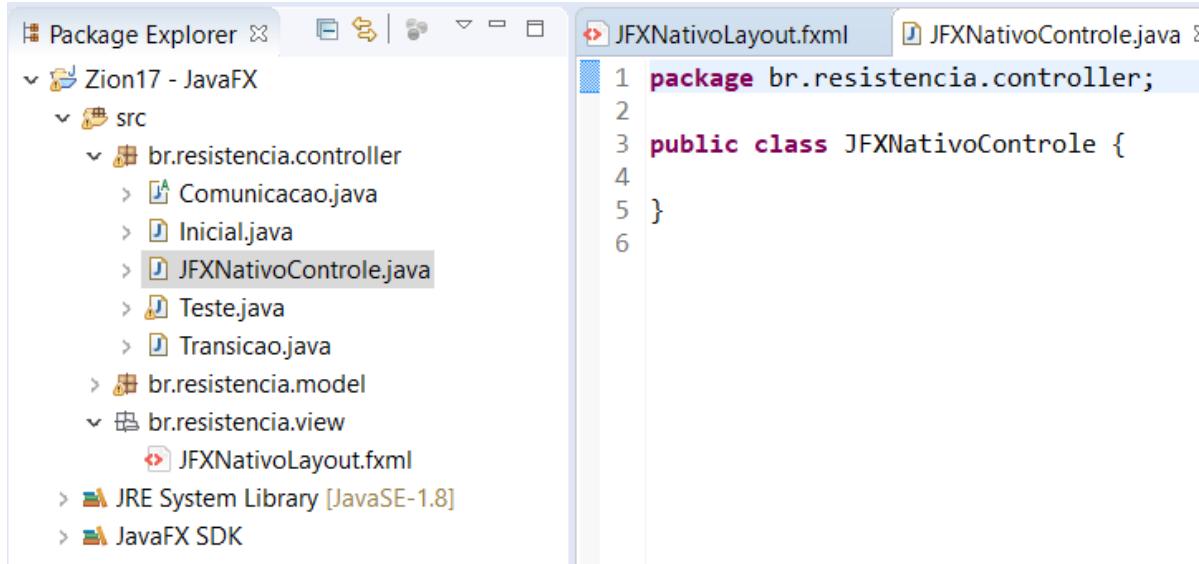
The screenshot shows the Eclipse IDE interface with the JavaFX plugin. The left sidebar displays the 'Package Explorer' with the project structure:

- Zion17 - JavaFX
- src
  - br.resistencia.controller
    - Comunicacao.java
    - Inicial.java
    - Testejava.java
    - Transicao.java
  - br.resistencia.model
  - br.resistencia.view
    - JFXNativoLayout.fxml
- JRE System Library [JavaSE-1.8]
- JavaFX SDK

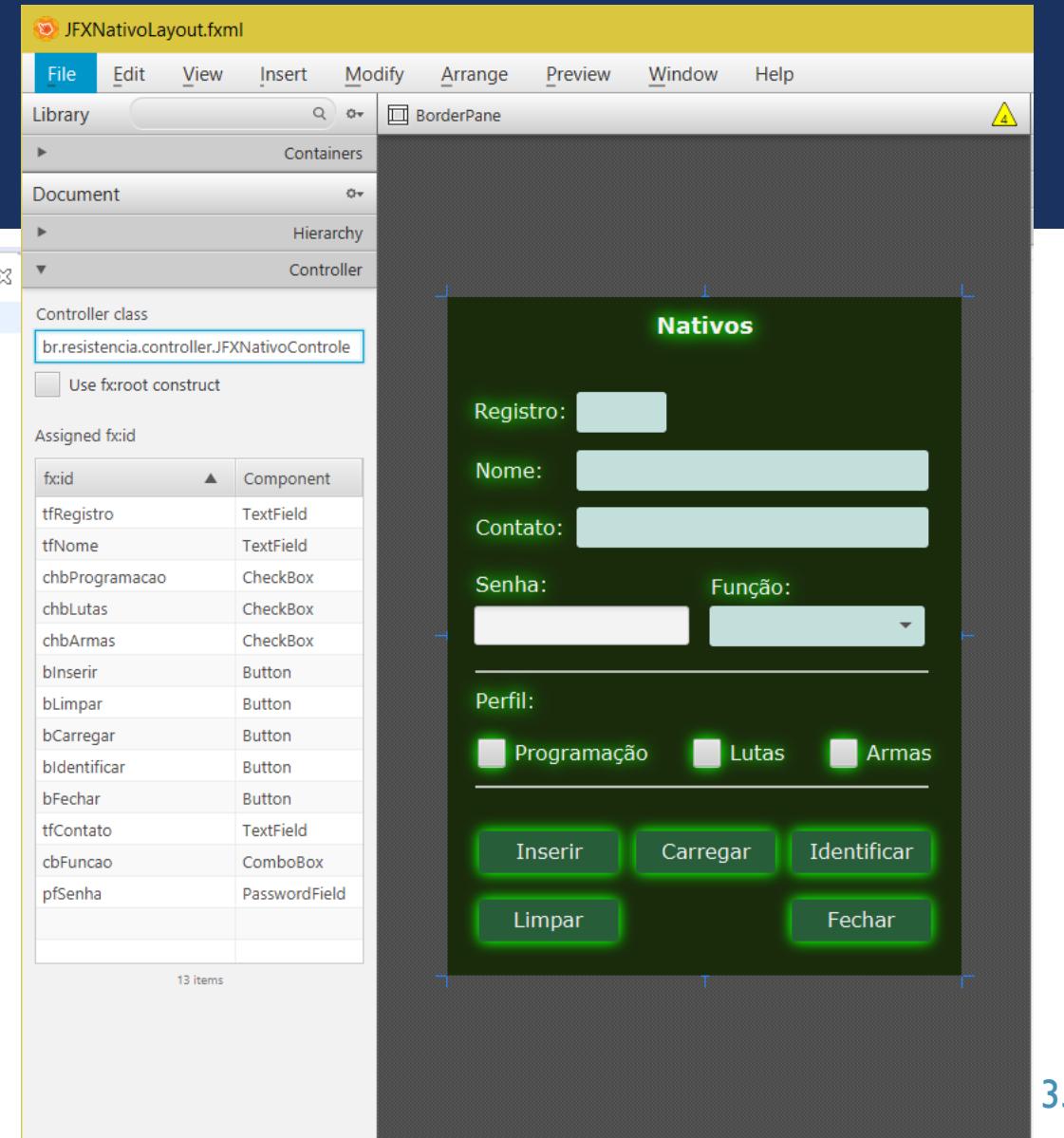
The main editor window shows the FXML code for 'JFXNativoLayout.fxml'. The code defines a BorderPane layout with two children: a top pane and a center pane. The top pane contains a Label with text 'Nativos' and a specific font and effect. The center pane contains a Label with text 'Registro:'.

```
<?xml version="1.0" encoding="UTF-8"?>
<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="464.0" prefWidth="406.0" xmlns="http://javafx.com/javaf
<top>
    <Pane prefHeight="45.0" prefWidth="600.0" style="-fx-background-color: #1b2d0d;" BorderPane.alignment="CENTER">
        <children>
            <Label layoutX="165.0" layoutY="11.0" text="Nativos" textFill="WHITE">
                <font>
                    <Font name="Verdana Bold" size="18.0" />
                </font>
                <effect>
                    <DropShadow color="#0bff01" height="40.0" radius="19.5" spread="0.4" width="40.0" />
                </effect>
            </Label>
        </children>
    </Pane>
</top>
<center>
    <Pane prefHeight="355.0" prefWidth="494.0" style="-fx-background-color: #1b2d0d;" BorderPane.alignment="CENTER">
        <children>
            <Label layoutX="21.0" layoutY="35.0" text="Registro:" textFill="WHITE">
                <font>
                    <Font name="Verdana" size="16.0" />
                </font>
            </Label>
        </children>
    </Pane>
</center>
```

# CONTROLE

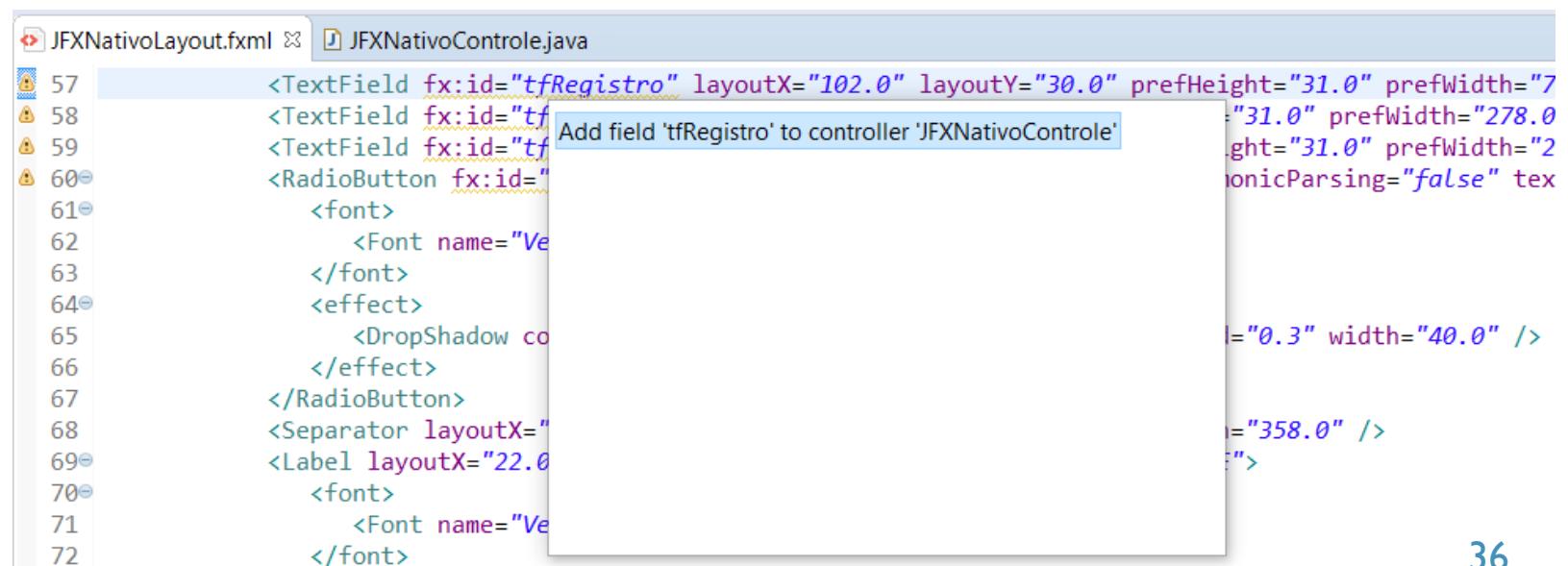


- Vínculo com a classe Controle pelo campo Controller Class da paleta Controller



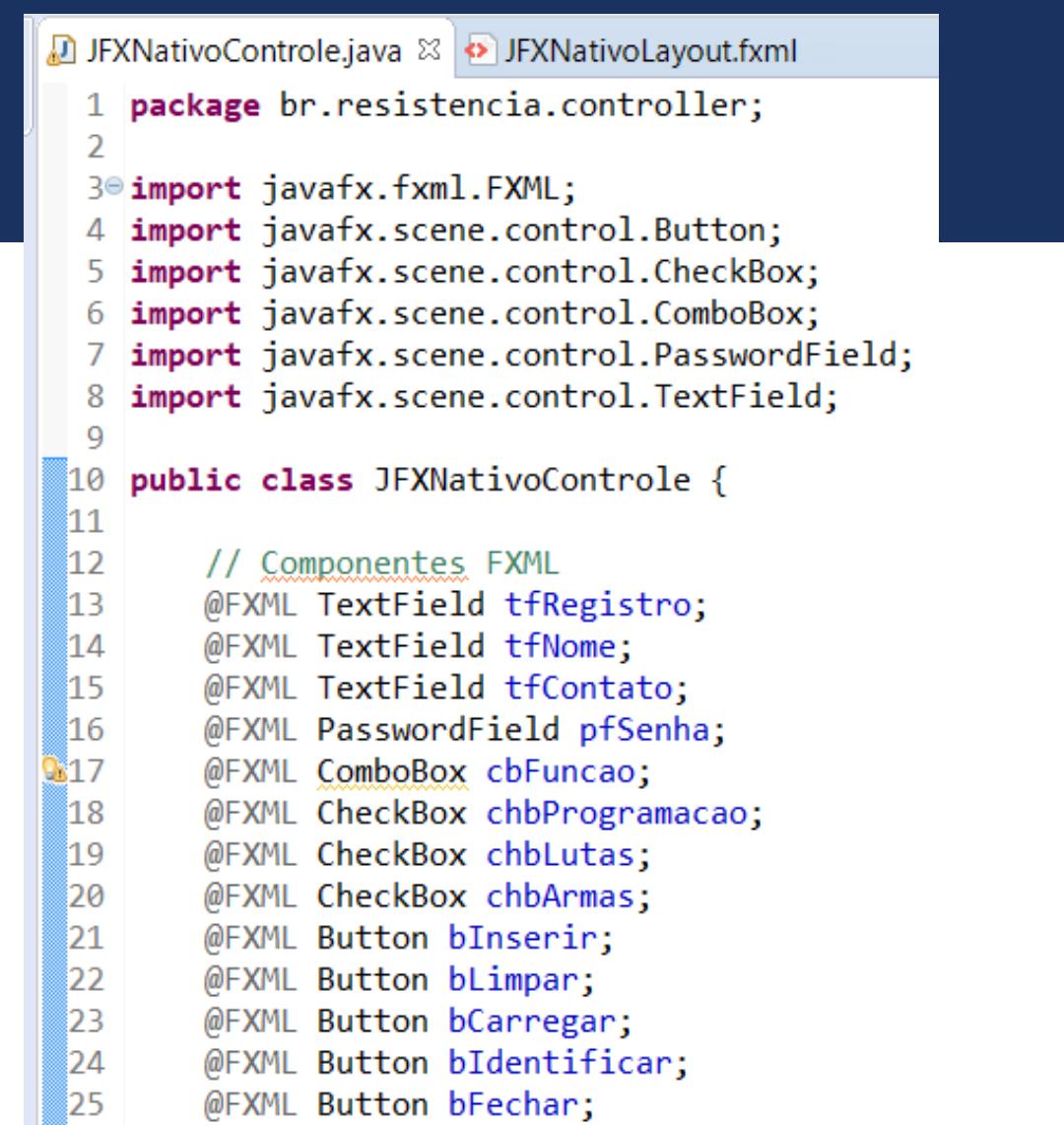
# CONTROLE

- No arquivo .fxml as propriedades fx:id e onAction ficam com observações e erros por não constar a respectiva referência na classe de controle.
- Para resolver utilizando recursos do Eclipse, coloque o cursor (clique) sobre a propriedade e pressione CTRL+1 e clique duas vezes sobre a primeira sugestão que será apresentada



# CONTROLE

- Na classe de controle são inseridas as referências aos controles (parecidas com declarações de atributos) antecedidos da notação `@FXML` que indica tratar-se de controles do arquivo `.fxml`
- Obs.: Essas notações são obrigatórias (não apagar).



The screenshot shows an IDE interface with two files open:

- JFXNativeController.java**: The code defines a class `JFXNativeController` with imports for various JavaFX controls and annotations. It includes annotations for `@FXML` to map Java objects to controls defined in the FXML file.
- JFXNativeLayout.fxml**: This is the associated FXML file, which typically contains the visual structure and logic for the application's user interface.

```
1 package br.resistencia.controller;
2
3 import javafx.fxml.FXML;
4 import javafx.scene.control.Button;
5 import javafx.scene.control.CheckBox;
6 import javafx.scene.control.ComboBox;
7 import javafx.scene.control.PasswordField;
8 import javafx.scene.control.TextField;
9
10 public class JFXNativeController {
11
12     // Componentes FXML
13     @FXML TextField tfRegistro;
14     @FXML TextField tfNome;
15     @FXML TextField tfContato;
16     @FXML PasswordField pfSenha;
17     @FXML ComboBox cbFuncao;
18     @FXML CheckBox chbProgramacao;
19     @FXML CheckBox chbLutas;
20     @FXML CheckBox chbArmas;
21     @FXML Button bInserir;
22     @FXML Button bLimpar;
23     @FXML Button bCarregar;
24     @FXML Button bIdentificar;
25     @FXML Button bFechar;
```

The screenshot shows a JavaFX code editor with two tabs: JFXNativeLayout.fxml and JFXNativeControle.java. The JFXNativeLayout.fxml tab contains XML code for a user interface. In the XML, there is a line of code: <Button fx:id="bInserir" layoutX="25.0" layoutY="303.0" mnemonicParsing="false" onAction="#inserir" prefHeight="32.0" prefWidth="110.0" style="-fx-backgr...". A tooltip is displayed over this line, listing two options: 'Add 'inserir()' to controller 'JFXNativeControle'' and 'Add 'inserir(ActionEvent)' to controller 'JFXNativeControle''. This illustrates the use of the Ctrl+1 keyboard shortcut to quickly add event handlers to FXML controls.

```
86             <font name="Verdana" size="16.0" />
87         </font>
88     <effect>
89         <DropShadow color="#0bff01" height="40.0" radius="19.5" spread="0.3" width="40.0" />
90     </effect>
91 </CheckBox>
92 <CheckBox fx:id="chbArmas" layoutX="302.0" layoutY="230.0" mnemonicParsing="false" text="Armas" textFill="WHITE">
93     <font>
94         <font name="Verdana" size="16.0" />
95     </font>
96     <effect>
97         <DropShadow color="#0bff01" height="40.0" radius="19.5" spread="0.3" width="40.0" />
98     </effect>
99 </CheckBox>
100 <Separator layoutX="22.0" layoutY="267.0" prefHeight="3.0" prefWidth="358.0" />
101 <Button fx:id="bInserir" layoutX="25.0" layoutY="303.0" mnemonicParsing="false" onAction="#inserir" prefHeight="32.0" prefWidth="110.0" style="-fx-backgr
102     <font>
103         <font name="Verdana" size="16.0" />
104     </font>
105     <effect>
106         <DropShadow color="#0bff01" height="30.0" radius="14.5" spread="0.3" width="30.0" />
107     </effect>
108 </Button>
109 <Button fx:id="bLimpar" layoutX="25.0" layoutY="357.0" mnemonicParsing="false" onAction="#limpar" prefHeight="32.0" prefWidth="110.0" style="-fx-backgr
110     <font>
111         <font name="Verdana" size="16.0" />
112     </font>
113     <effect>
114         <DropShadow color="#0bff01" height="30.0" radius="14.5" spread="0.3" width="30.0" />
115     </effect>
116 </Button>
117 <Button fx:id="bCarregar" layoutX="149.0" layoutY="303.0" mnemonicParsing="false" onAction="#carregar" prefHeight="32.0" prefWidth="110.0" style="-fx-backgr
118     <font>
119         <font name="Verdana" size="16.0" />
120     </font>
121     <effect>
```

- O mesmo procedimento será utilizado para inserir as referências do onAction (CTRL+1)

# CONTROLE

- E as referências também serão inseridas na classe controle

```
JFXNativoControle.java ✘ JFXNativoLayout.fxml
1 package br.resistencia.controller;
2
3 import javafx.fxml.FXML;
4 import javafx.scene.control.Button;
5 import javafx.scene.control.CheckBox;
6 import javafx.scene.control.ComboBox;
7 import javafx.scene.control.PasswordField;
8 import javafx.scene.control.TextField;
9
10 public class JFXNativoControle {
11
12     // Componentes FXML
13     @FXML TextField tfRegistro;
14     @FXML TextField tfNome;
15     @FXML TextField tfContato;
16     @FXML PasswordField pfSenha;
17     @FXML ComboBox cbFuncao;
18     @FXML CheckBox chbProgramacao;
19     @FXML CheckBox chbLutas;
20     @FXML CheckBox chbArmas;
21     @FXML Button bInserir;
22     @FXML Button bLimpar;
23     @FXML Button bCarregar;
24     @FXML Button bIdentificar;
25     @FXML Button bFechar;
26
27     // Métodos para eventos FXML
28     @FXML public void inserir() {}
29
30     @FXML public void limpar() {}
31
32     @FXML public void carregar() {}
33
34     @FXML public void identificar() {}
35
36     @FXML public void fechar() {}
37 }
```

# INICIAL



```
Inicial.java x
1 package br.resistencia.controller;
2
3 import javafx.application.Application;
4 import javafx.fxml.FXMLLoader;
5 import javafx.scene.Scene;
6 import javafx.scene.layout.BorderPane;
7 import javafx.stage.Stage;
8
9 * @author Prof. Ralfe
10 public class Inicial extends Application{
11
12     @Override
13     public void start(Stage primaryStage) throws Exception {
14
15         // Criação de um objeto FXMLLoader para carregar o arquivo fxml (layout)
16         FXMLLoader loader = new FXMLLoader();
17         // Carregamento o arquivo fxml
18         loader.setLocation(getClass().getResource("/br/resistencia/view/JFXNativoLayout.fxml"));
19         // Criação do Layout Pane (gerenciador de layout) que será o node root (nó ou componente raiz)
20         // recebendo o layout (arquivo fxml)
21         BorderPane nodeRoot = loader.load();
22         // Atribuição do componente raiz (e do layout) a cena
23         Scene scene = new Scene(nodeRoot);
24         // Atribuição da cena ao palco primário
25         primaryStage.setScene(scene);
26         // Apresenta o formulário
27         primaryStage.show();
28     }
29
30     public static void main(String[] args) {
31         launch(args);
32     }
33 }
34
35 }
```

## ■ Implementação do método start (instruções básicas)

# INICIAL



```
J Inicial.java x
1 package br.resistencia.controller;
2
3 import javafx.application.Application;...
4
5 * @author Prof. Ralfe...
6 public class Inicial extends Application{
7
8     @Override
9     public void start(Stage primaryStage) throws Exception {
10
11         // Criação de um objeto FXMLLoader para carregar o arquivo fxml (layout)
12         FXMLLoader loader = new FXMLLoader();
13         // Carregamento o arquivo fxml
14         loader.setLocation(getClass().getResource("/br/resistencia/view/JFXNativoLayout.fxml"));
15         // Criação do Layout Pane (gerenciador de layout) que será o node root (nó ou componente raiz)
16         // recebendo o layout (arquivo fxml)
17         BorderPane nodeRoot = loader.load();
18         // Atribuição do componente raiz (e do layout) a cena
19         Scene scene = new Scene(nodeRoot);
20         // Atribuição da cena ao palco primário
21         primaryStage.setScene(scene);
22
23         // Retira a barra superior da janela (icone, título, minimizar, maximizar e fechar)
24         primaryStage.initStyle(StageStyle.UNDECORATED);
25         // Não permite o redimensionamento
26         primaryStage.setResizable(false);
27         // Centraliza a apresentação
28         primaryStage.centerOnScreen();
29
30         // Apresenta o formulário
31         primaryStage.show();
32     }
33
34     public static void main(String[] args) {
35         launch(args);
36     }
37
38 }
```

- Implementação do método start (formatações do formulário)

# CONTROLE

```
12 public class JFXNativoControle implements Initializable {  
13     // Componentes FXML  
14     @FXML TextField tfRegistro;  
15     @FXML TextField tfNome;  
16     @FXML TextField tfContato;  
17     @FXML PasswordField pfSenha;  
18     @FXML ComboBox cbFuncao;  
19     @FXML CheckBox chbProgramacao;  
20     @FXML CheckBox chbLutas;  
21     @FXML CheckBox chbArmas;  
22     @FXML Button bInserir;  
23     @FXML Button bLimpar;  
24     @FXML Button bCarregar;  
25     @FXML Button bIdentificar;  
26     @FXML Button bFechar;  
27 }  
28 }
```

1 method to implement:  
- javafx.fxml.Initializable.initialize()

```
JFXNativoControle.java  
14 public class JFXNativoControle implements Initializable {  
15     // Componentes FXML  
16     @FXML TextField tfRegistro;  
17     @FXML TextField tfNome;  
18     @FXML TextField tfContato;  
19     @FXML PasswordField pfSenha;  
20     @FXML ComboBox cbFuncao;  
21     @FXML CheckBox chbProgramacao;  
22     @FXML CheckBox chbLutas;  
23     @FXML CheckBox chbArmas;  
24     @FXML Button bInserir;  
25     @FXML Button bLimpar;  
26     @FXML Button bCarregar;  
27     @FXML Button bIdentificar;  
28     @FXML Button bFechar;  
29 }  
30  
31 @Override  
32 public void initialize(URL arg0, ResourceBundle arg1) {  
33     // Inicializa os items do ComboBox cbPermissao  
34     cbFuncao.getItems().addAll("Navegador", "Piloto", "Armeiro", "Programador");  
35 }  
36  
37 @Override  
38 public void initialize(URL arg0, ResourceBundle arg1) {  
39     // TODO Auto-generated method stub  
40 }  
41 }
```

- Inserção dos itens do ComboBox por meio do método initialize.

# CONTROLE

JFXNativeControle.java

```
45 // Métodos para eventos FXML
46 @FXML public void inserir() {
47
48     nativo.setNome(tfNome.getText());
49     nativo.setContato(tfContato.getText());
50     nativo.setSenha(pfSenha.getText().toString());
51     nativo.setFuncao(cbFuncao.getSelectionModel().getSelectedItem().toString());
52
53     if(chbProgramacao.isSelected()) {
54         nativo.getPerfil().setProgramacao(true);
55     }else {
56         nativo.getPerfil().setProgramacao(false);
57     }
58
59     if(chbArmas.isSelected()) {
60         nativo.getPerfil().setArmas(true);
61     }else {
62         nativo.getPerfil().setArmas(false);
63     }
64
65     if(chbLutas.isSelected()) {
66         nativo.getPerfil().setLutas(true);
67     }else {
68         nativo.getPerfil().setLutas(false);
69     }
70 }
```

JFXNativeControle.java

```
83 @FXML public void carregar() {
84     tfNome.setText(nativo.getNome());
85     tfContato.setText(nativo.getContato());
86     pfSenha.setText(nativo.getSenha());
87     cbFuncao.setValue(nativo.getFuncao());
88
89     if(nativo.getPerfil().isProgramacao()) {
90         chbProgramacao.setSelected(true);
91     }else {
92         chbProgramacao.setSelected(false);
93     }
94
95     if(nativo.getPerfil().isArmas()) {
96         chbArmas.setSelected(true);
97     }else {
98         chbArmas.setSelected(false);
99     }
100
101    if(nativo.getPerfil().isLutas()) {
102        chbLutas.setSelected(true);
103    }else {
104        chbLutas.setSelected(false);
105    }
106
107 }
```

■ Implementação dos métodos inserir e carregar.

# CONTROLE

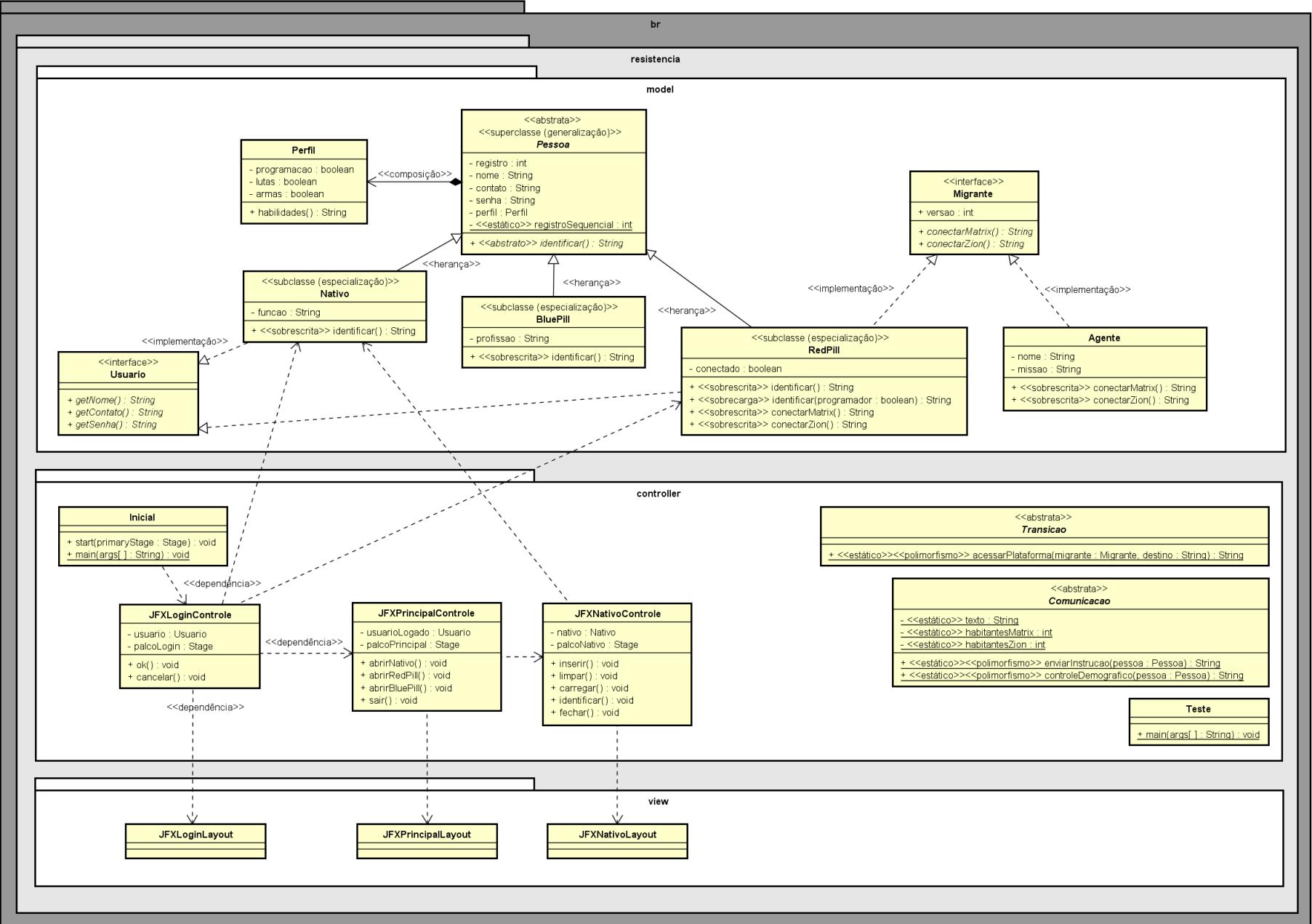
```
JFXNativeControle.java
1 JFXNativeControle.java
2
3 @FXML public void limpar() {
4     tfNome.setText("");
5     tfContato.setText("");
6     pfSenha.setText("");
7     cbFuncao.setValue("");
8     chbProgramacao.setSelected(false);
9     chbArmas.setSelected(false);
10    chbLutas.setSelected(false);
11}
12}
```

```
JFXNativeControle.java
109 @FXML public void identificar() {
110
111     Alert alert = new Alert(AlertType.INFORMATION);
112     alert.setTitle("Nativo");
113     alert.setHeaderText("Identificação");
114     alert.setContentText(nativo.identificar());
115     alert.showAndWait();
116 }
117
118 @FXML public void fechar() {
119     System.exit(0);
120 }
121
122 }
```

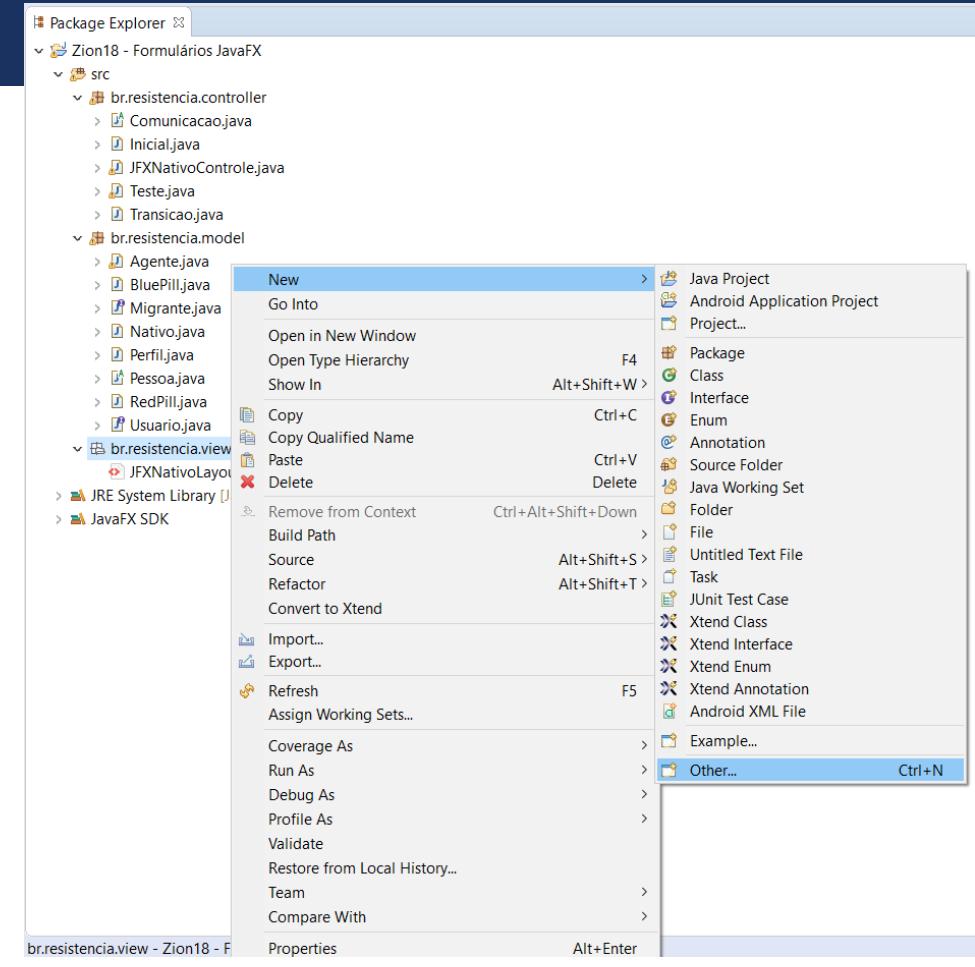
- Implementação dos métodos limpar e identificar.
- Atribuição do número do registro no formulário.

```
JFXNativeControle.java
37 @Override
38 public void initialize(URL arg0, ResourceBundle arg1) {
39     // Inicializa os items do ComboBox cbPermissao
40     cbFuncao.getItems().addAll("Navegador", "Piloto", "Armeiro", "Programador");
41
42     tfRegistro.setText(String.valueOf( nativo.getRegistro()));
43 }
```

# ZION



# LOGIN



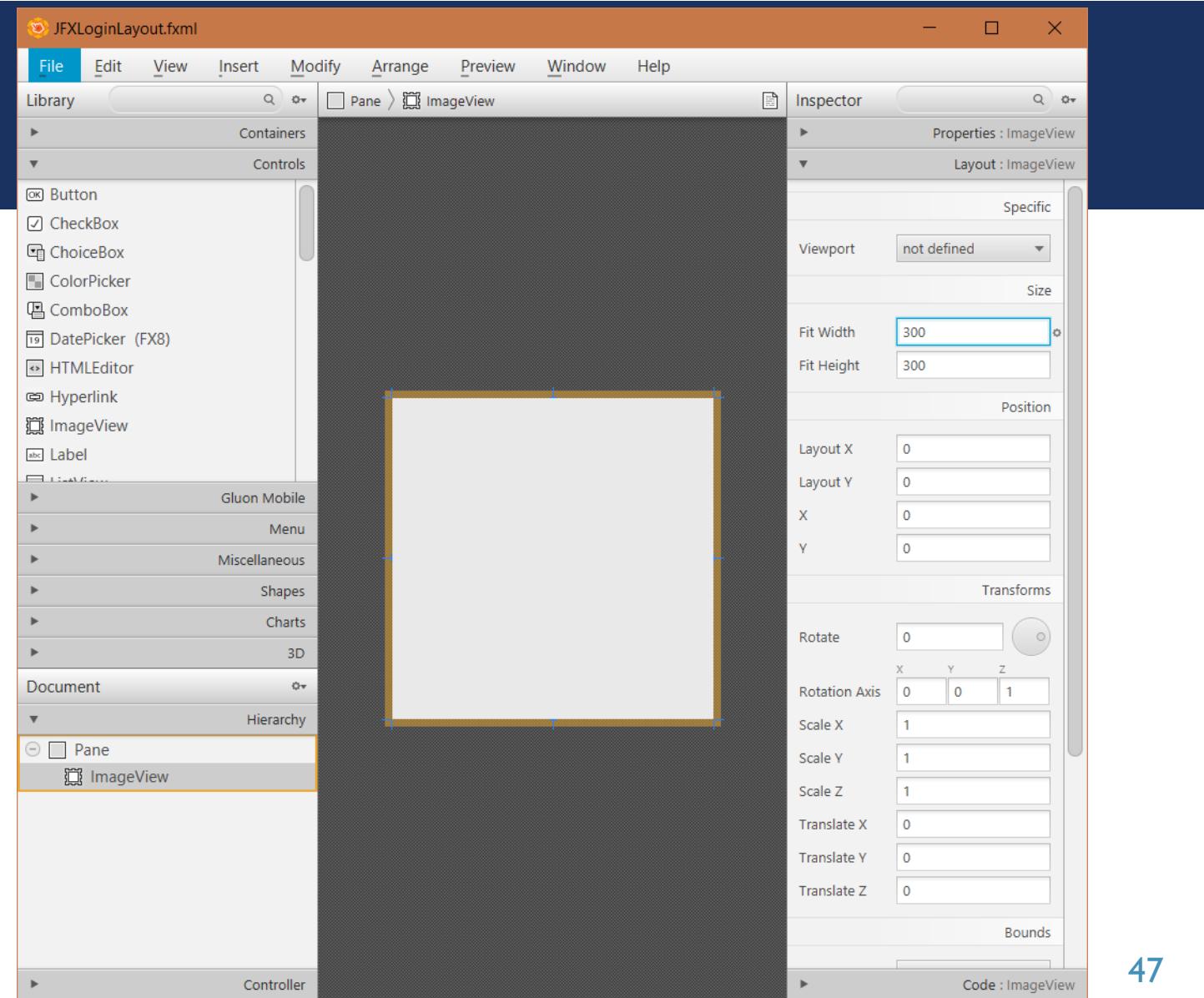
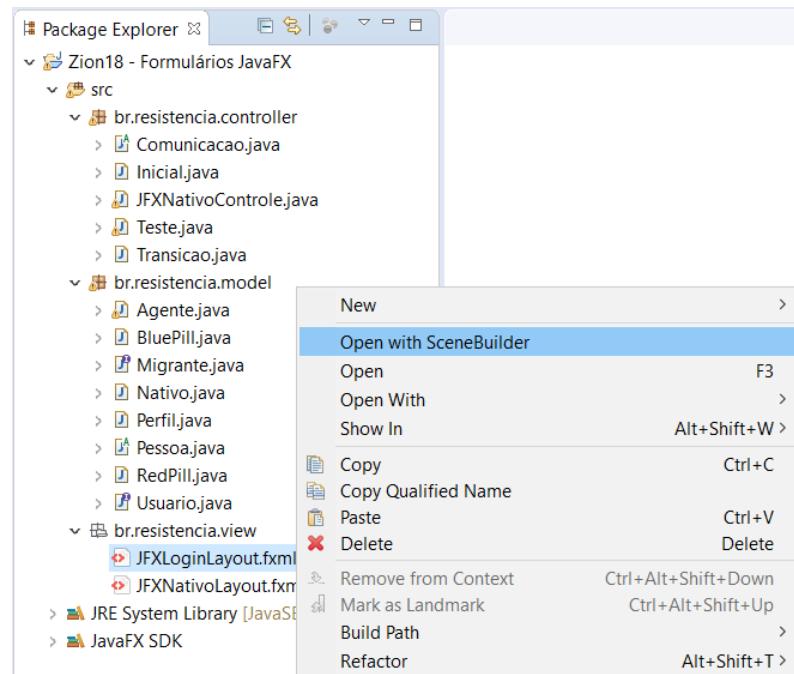
The screenshot shows two windows from the Eclipse IDE:

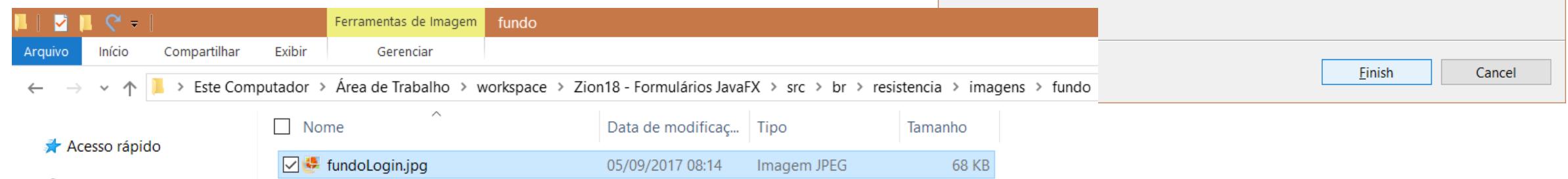
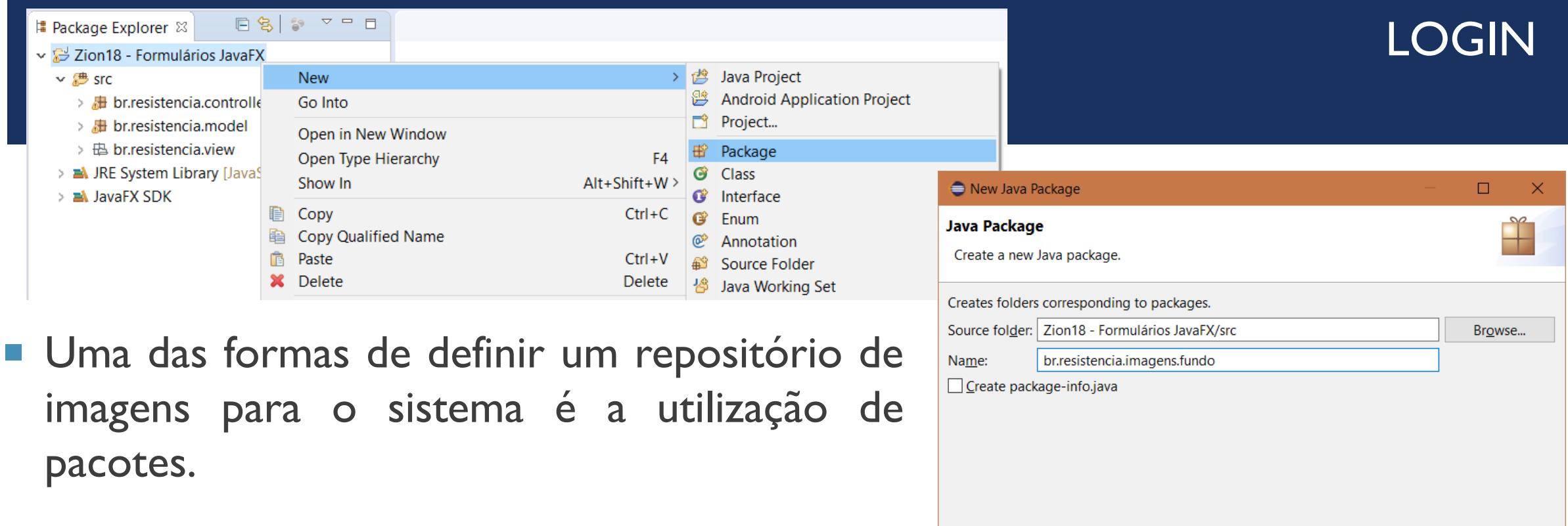
- Select a wizard**: This window is titled 'New' and contains a heading 'Wizards:' with a 'type filter text' input field. It lists several JavaFX-related wizards:
  - Java Project
  - Android Application Project
  - Project...
  - Package
  - Class
  - Interface
  - Enum
  - Annotation
  - Source Folder
  - Java Working Set
  - Folder
  - File
  - Untitled Text File
  - Task
  - JUnit Test Case
  - Xtend Class
  - Xtend Interface
  - Xtend Enum
  - Xtend Annotation
  - Android XML File
  - Example...
  - Other... (highlighted)
  - Ctrl+N
- FXML File**: This window is titled 'Create a new FXML File' and contains fields for defining the new file:

Source folder	Zion18 - Formulários JavaFX/src	Browse ...
Package	br.resistencia.view	Browse ...
Name	JFXLoginLayout	
Root Element	AnchorPane - javafx.scene.layout	Browse ...
Dynamic Root (fx:root)	<input type="checkbox"/>	

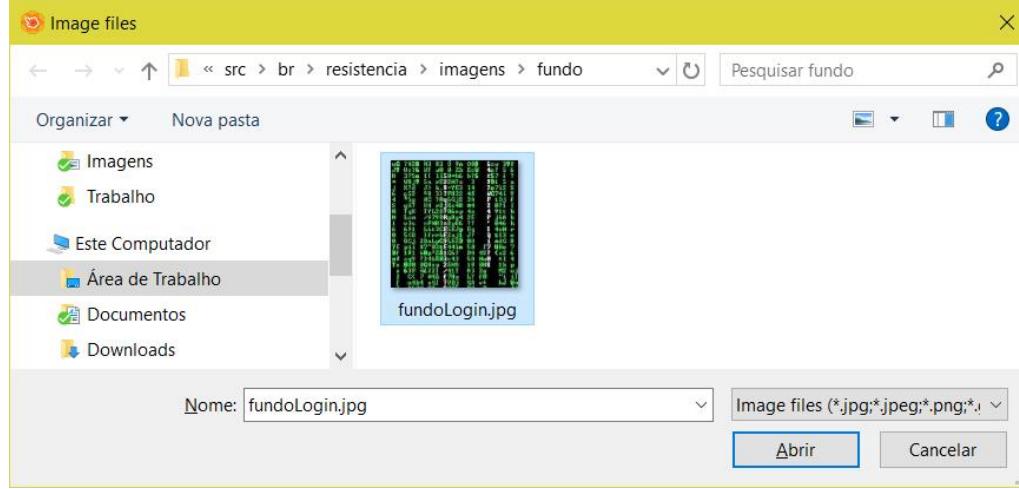
At the bottom of this window are buttons for '?', '< Back' (disabled), 'Next >', 'Finish' (highlighted), and 'Cancel'.

# LOGIN





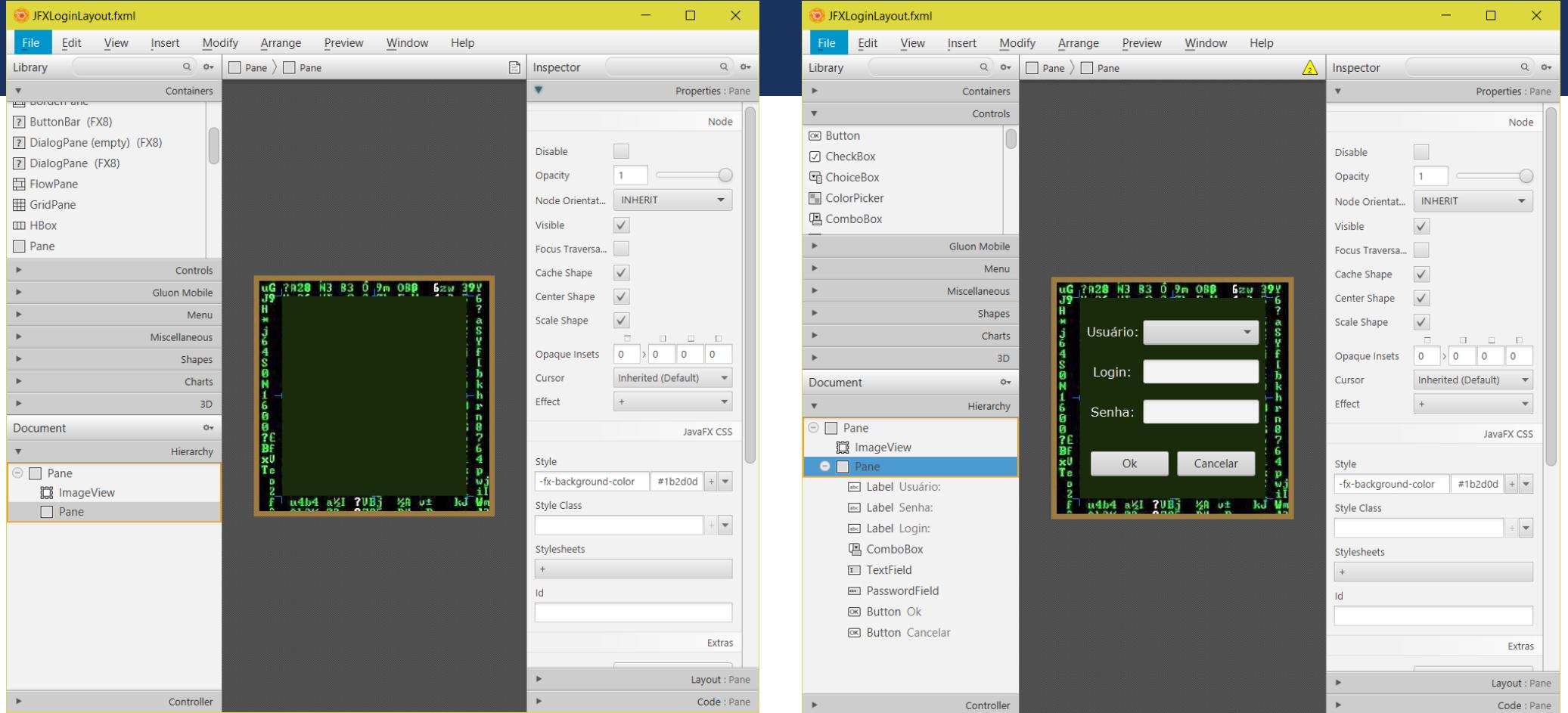
# LOGIN



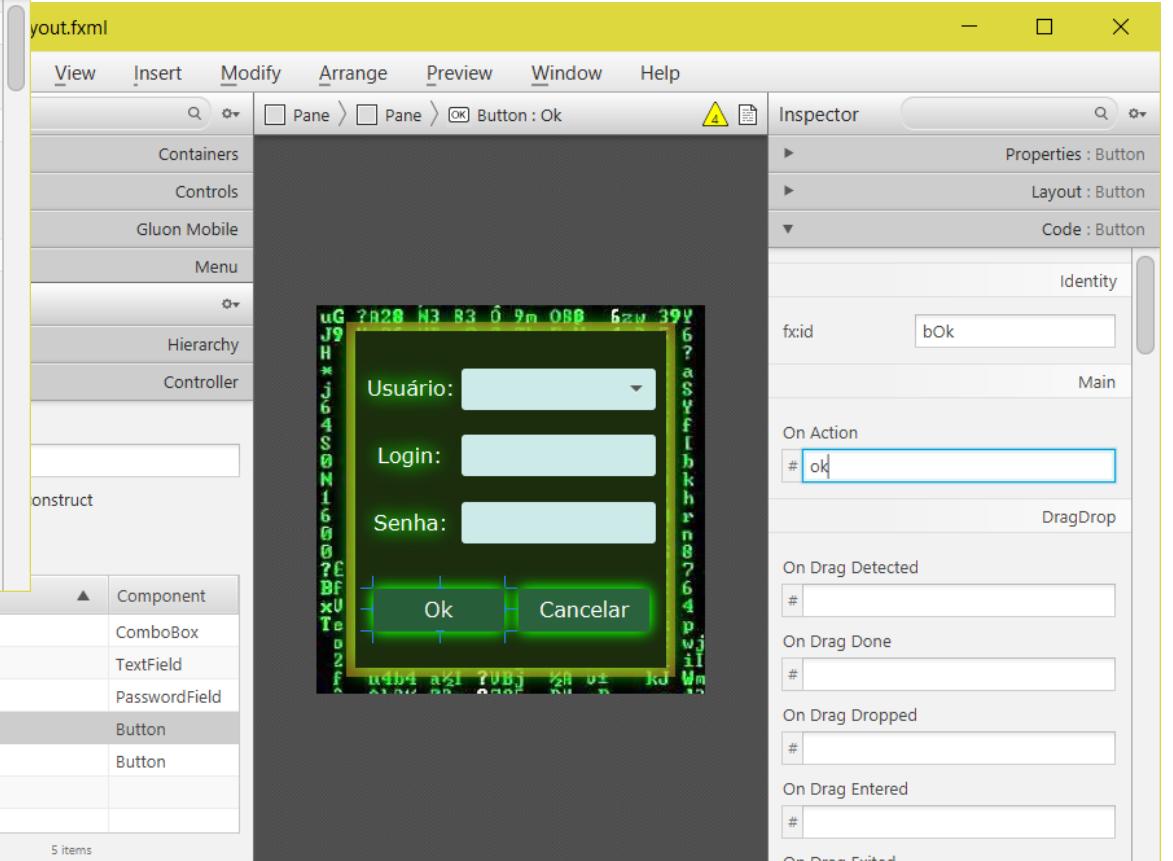
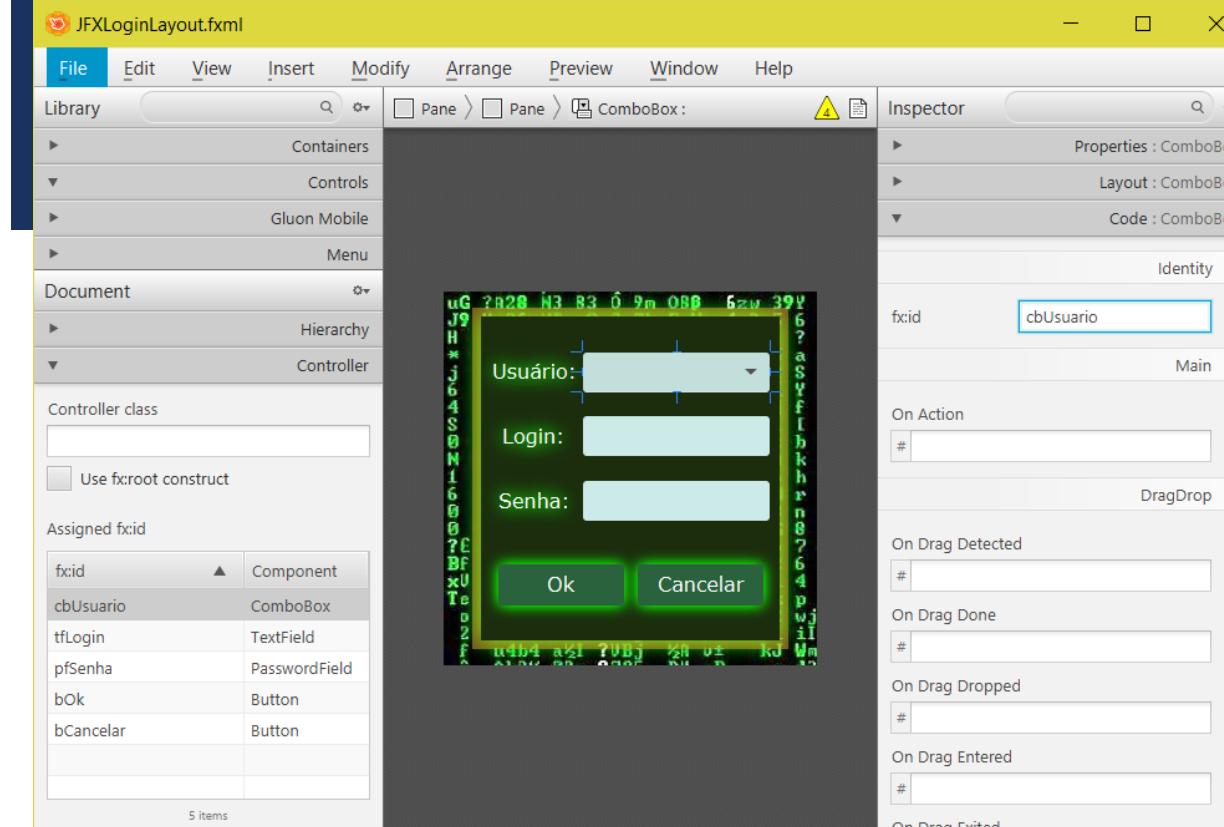
A screenshot of the JavaFX Scene Builder properties panel for an 'ImageView' control. The 'Properties' tab is selected. In the 'Image' section, the URL is set to '@..\\imagens\\fundo\\fundologin.jpg'. Other settings include 'Preserve Ratio' (checked), 'Smooth' (checked), 'Disable' (unchecked), 'Opacity' (set to 1), 'Node Orientation' (set to 'LEFT\_TO\_RIGHT'), 'Visible' (checked), 'Focus Traversable' (unchecked), 'Cursor' (set to 'Inherited (Default)'), 'Effect' (set to '+'), 'JavaFX CSS' (empty), 'Style' (empty), 'Style Class' (empty), 'Id' (empty), 'Extras' (empty), 'Blend Mode' (set to 'SRC\_OVER'), and 'Cache' (unchecked). The 'Layout' section shows 'Layout: ImageView' and the 'Code' section shows 'Code : ImageView'.

## ■ Controle ImageView

# LOGIN



# LOGIN



- Definição dos identificadores dos controles (fx:id) e dos eventos (onAction)

# LOGIN

The screenshot shows the JavaFX Scene Builder interface with a login window and its associated Java code.

**Java Code (JFXLoginControle.java):**

```
1 package br.resistencia.controller;
2
3 public class JFXLoginControle {
4
5 }
6
```

**FXML File (JFXLoginLayout.fxml):**

The FXML file defines a window with three text fields (Usuário, Login, Senha) and two buttons (Ok, Cancelar). The 'Controller class' is set to `br.resistencia.controller.JFXLoginControle`.

```
File Edit View Insert Modify Arrange Preview Window Help
Library ▾
Containers
Controls
Gluon Mobile
Menu
Document ▾
Hierarchy
Controller
Controller class
br.resistencia.controller.JFXLoginControle
Use fx:root construct
Assigned fx:id
fx:id Component
cbUsuario ComboBox
tfLogin TextField
pfSenha PasswordField
bOk Button
bCancelar Button
5 items
```

**Scene Builder Inspector:**

- Properties:** Properties for the `Button : Ok` button, including `fx:id: bOk`.
- Layout:** Layout properties for the `Button : Ok` button.
- Code:** Code for the `Button : Ok` button, showing the `# ok` event handler.
- Identity:** Identity properties for the `Button : Ok` button.

**On Action:** The `# ok` event handler is defined for the `Ok` button.

**DragDrop:** DragDrop properties for the `Button : Ok` button.

**On Drag Detected:** On Drag Detected properties for the `Button : Ok` button.

**On Drag Done:** On Drag Done properties for the `Button : Ok` button.

**On Drag Dropped:** On Drag Dropped properties for the `Button : Ok` button.

**On Drag Entered:** On Drag Entered properties for the `Button : Ok` button.

**On Drag Exited:** On Drag Exited properties for the `Button : Ok` button.

## Vínculo com a classe de controle

PROF. RALFE DELLA CROCE FILHO

# LOGIN



The screenshot shows the Eclipse IDE interface with two tabs open: `JFXLoginControle.java` and `JFXLoginLayout.fxml`. The `JFXLoginControle.java` tab contains Java code for a controller class, and the `JFXLoginLayout.fxml` tab contains FXML code for a login interface. The FXML code defines several UI components: a ComboBox (`cbUsuario`), a TextField (`tfLogin`), a PasswordField (`pfSenha`), a Button (`bOk`), and another Button (`bCancelar`). Each component has its layoutX and layoutY properties set to specific values, and their widths and heights are defined by prefWidth and prefHeight. The font for all components is set to "Verdana" with a size of 16.0. The `JFXLoginControle.java` code includes logic for handling button actions.

```
48     <ComboBox fx:id="cbUsuario" layoutX="82.0" layoutY="29.0" prefWidth="150.0" style="-fx-background-color: #cceeff;" ^
49      <TextField fx:id="tfLogin" layoutX="82.0" layoutY="80.0" prefHeight="31.0" prefWidth="150.0" style="-fx-backgroun^
50          <font>
51              <Font name="Verdana" size="16.0" />
52          </font>
53      </TextField>
54      <PasswordField fx:id="pfSenha" layoutX="82.0" layoutY="132.0" prefHeight="31.0" prefWidth="150.0" style="-fx-bac^
55          <font>
56              <Font name="Verdana" size="16.0" />
57          </font>
58      </PasswordField>
59      <Button fx:id="bOk" layoutX="14.0" layoutY="199.0" mnemonicParsing="false" onAction="#ok" prefHeight="31.0" p^
60          <font>
61              <Font name="Verdana" size="16.0" />
62          </font>
63          <effect>
64              <DropShadow color="#0bff01" height="30.0" radius="14.5" spread="0.3" width="30.0" />
65          </effect>
66      </Button>
67      <Button fx:id="bCancelar" layoutX="126.0" layoutY="199.0" mnemonicParsing="false" onAction="#cancelar" prefHeig^
68          <font>
69              <Font name="Verdana" size="16.0" />
70          </font>
```

- Inserção das referências na classe controle pelo .fxml (CTRL+1)

# LOGIN

## ■ Referências dos controles do arquivo .fxml

The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer view displays the project structure under 'Zion18 - Formulários JavaFX'. It includes packages like 'br.resistencia.controller', 'br.resistencia.model', and 'br.resistencia.view', along with Java files such as 'Comunicacao.java', 'Inicial.java', 'JFXLoginControle.java', 'JFXNativoControle.java', 'Teste.java', 'Transicao.java', and FXML files 'JFXLoginLayout.fxml' and 'JFXNativoLayout.fxml'. It also lists 'JRE System Library [JavaSE-1.8]' and 'JavaFX SDK'. On the right, the Java code editor is open for 'JFXLoginControle.java'. The code imports various JavaFX components and defines a class 'JFXLoginControle' with methods for handling FXML components and events.

```
1 package br.resistencia.controller;
2
3 import javafx.fxml.FXML;
4 import javafx.scene.control.ComboBox;
5 import javafx.scene.control.TextField;
6 import javafx.scene.control.PasswordField;
7 import javafx.scene.control.Button;
8
9 public class JFXLoginControle {
10
11     // Componentes FXML
12     @FXML ComboBox cbUsuario;
13     @FXML TextField tfLogin;
14     @FXML PasswordField pfSenha;
15     @FXML Button bOk;
16     @FXML Button bCancelar;
17
18     // Métodos para eventos FXML
19     @FXML public void ok() {}
20     @FXML public void cancelar() {}
21 }
22 }
```

# LOGIN

- Criação do objetos (nativo e redPill) e do atributo usuario para validação do login.

```
JFXLoginControle.java x
1 package br.resistencia.controller;
2
3 import java.net.URL;
4 import java.util.ResourceBundle;
5 import br.resistencia.model.Nativo;
6 import br.resistencia.model.RedPill;
7 import br.resistencia.model.Usuario;
8 import javafx.fxml.FXML;
9 import javafx.fxml.Initializable;
10 import javafx.scene.control.Alert;
11 import javafx.scene.control.Alert.AlertType;
12 import javafx.scene.control.Button;
13 import javafx.scene.control.ComboBox;
14 import javafx.scene.control.PasswordField;
15 import javafx.scene.control.TextField;
16
17 * @author Prof. Ralfe
18*
19 public class JFXLoginControle implements Initializable {
20
21     // (Por enquanto) Criação dos objetos que serão usuários
22     Nativo nativo = new Nativo("Tank", "root", "123", true, false, true, "admin");
23     RedPill redPill = new RedPill("Trinity", "trinity@zion", "trinity", false, true, true, true);
24     // Declaração de um atributo que receberá o usuário logado
25     // Atente para o polimorfismo por interface.
26     private Usuario usuario;
27
28     // Componentes FXML
29     @FXML ComboBox cbUsuario;
30     @FXML TextField tfLogin;
31     @FXML PasswordField pfSenha;
32     @FXML Button bOk;
33     @FXML Button bCancelar;
34
35     @Override
36     public void initialize(URL arg0, ResourceBundle arg1) {
37         // Inicializa os items do ComboBox cbUsuario
38         cbUsuario.getItems().addAll("Nativo", "Red Pill");
39     }
40
41 }
```

# LOGIN

```
43 // Métodos para eventos FXML
44 @FXML public void ok() {
45     // Declaração de variáveis
46     String contatoUsuario = "", senhaUsuario = "";
47     String contatoInformado = "", senhaInformada = "";
48
49     // Verifica o usuário selecionado e armazena o respectivo objeto em usuário
50     if(cbUsuario.getSelectionModel().getSelectedItem().toString().equals("Nativo")) {
51         usuario = nativo;
52     }
53
54     if(cbUsuario.getSelectionModel().getSelectedItem().toString().equals("RedPill")) {
55         usuario = redPill;
56     }
57
58     // Obtém o contato e senha armazenados no usuário
59     contatoUsuario = usuario.getContato();
60     senhaUsuario = usuario.getSenha();
61
62     // Obtém o contato e senha informados no formulário
63     contatoInformado = tfLogin.getText();
64     senhaInformada = pfSenha.getText();
65
66     // Valida o contato e senha
67     if(contatoInformado.equals(contatoUsuario) && senhaInformada.equals(senhaUsuario)) {
68         // Se for válido, invoca o formulário principal
69         Alert alert = new Alert(AlertType.CONFIRMATION);
70         alert.setTitle("Login");
71         alert.setHeaderText("Validação de usuário.");
72         alert.setContentText("Login válido! Exibe o formulário Principal.");
73         alert.showAndWait();
74     }else {
75         // Se o contato e/ou senha estiverem incorretos
76         Alert alert = new Alert(AlertType.ERROR);
77         alert.setTitle("Login");
78         alert.setHeaderText("Validação de usuário.");
79         alert.setContentText("Contato e/ou senha não encontrados!");
80         alert.showAndWait();
81     }
82 }
```

## ■ Validação do login.

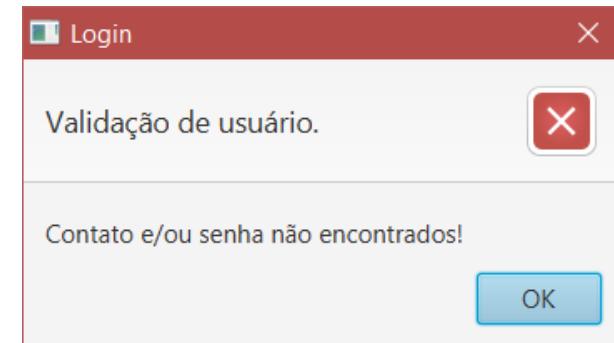
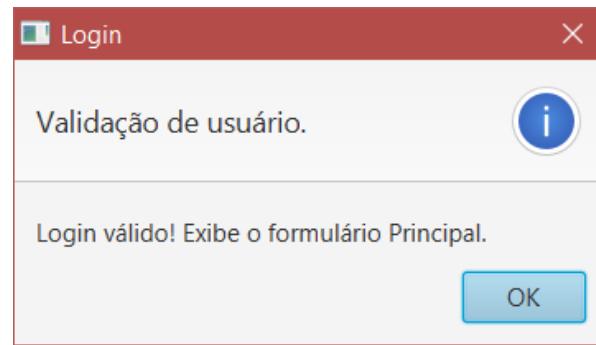
```
JFXLoginControle.java ✘
85 @FXML public void cancelar() {
86     System.exit(0);
87 }
88 }
```

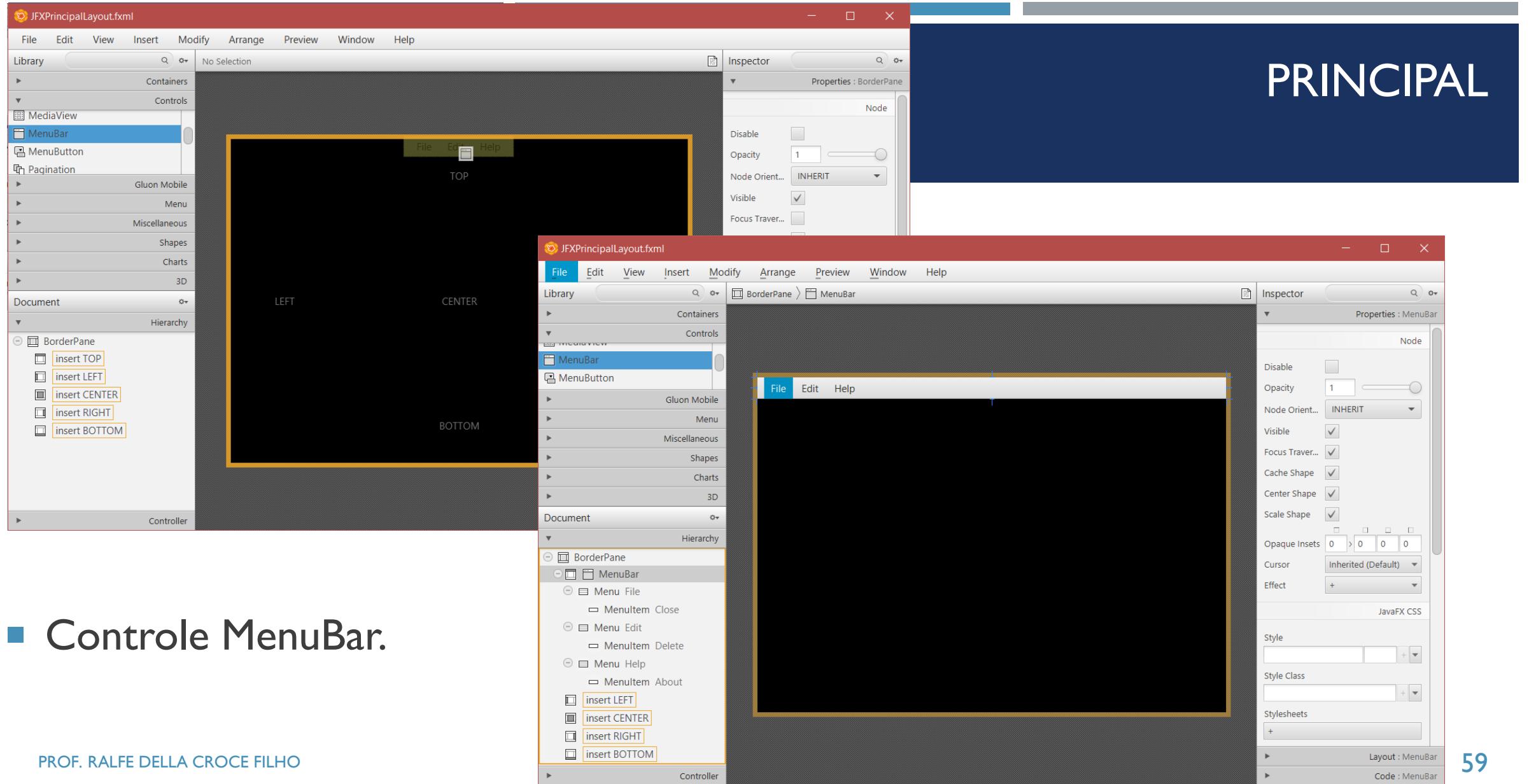
# INICIAL

```
14 public class Inicial extends Application{  
15  
16    @Override  
17    public void start(Stage primaryStage) throws Exception {  
18  
19        // Criação de um objeto FXMLLoader para carregar o arquivo fxml (layout)  
20        FXMLLoader loader = new FXMLLoader();  
21        // Carregamento o arquivo fxml  
22        loader.setLocation(getClass().getResource("/br/resistencia/view/JFXNativoLayout.fxml"));  
23        // Criação do Layout Pane (gerenciador de layout) que será o node root (nó ou componente raiz)  
24        // recebendo o layout (arquivo fxml)  
25        BorderPane nodeRoot = loader.load();  
26        // Atribuição do componente raiz (e do layout) a cena  
27        Scene scene = new Scene(nodeRoot);  
28        // Atribuição da cena ao palco primário  
29        primaryStage.setScene(scene);  
30  
31        // Retira a barra superior da janela (icone, título, minimizar, maximizar e fechar)  
32        primaryStage.initStyle(StageStyle.UNDECORATED);  
33        // Não permite o redimensionamento  
34        primaryStage.setResizable(false);  
35        // Centraliza a apresentação  
36        primaryStage.centerOnScreen();  
37  
38        // Apresenta o formulário  
39        primaryStage.show();  
40    }  
41  
42    // Carregamento o arquivo fxml  
43    loader.setLocation(getClass().getResource("/br/resistencia/view/JFXLoginLayout.fxml"));  
44    // Criação do Layout Pane (gerenciador de layout) que será o node root (nó ou componente raiz)  
45    // recebendo o layout (arquivo fxml)  
46    Pane nodeRoot = loader.load();  
47
```

- Alteração da classe Inicial para a chamada do formulário de login.

# LOGIN

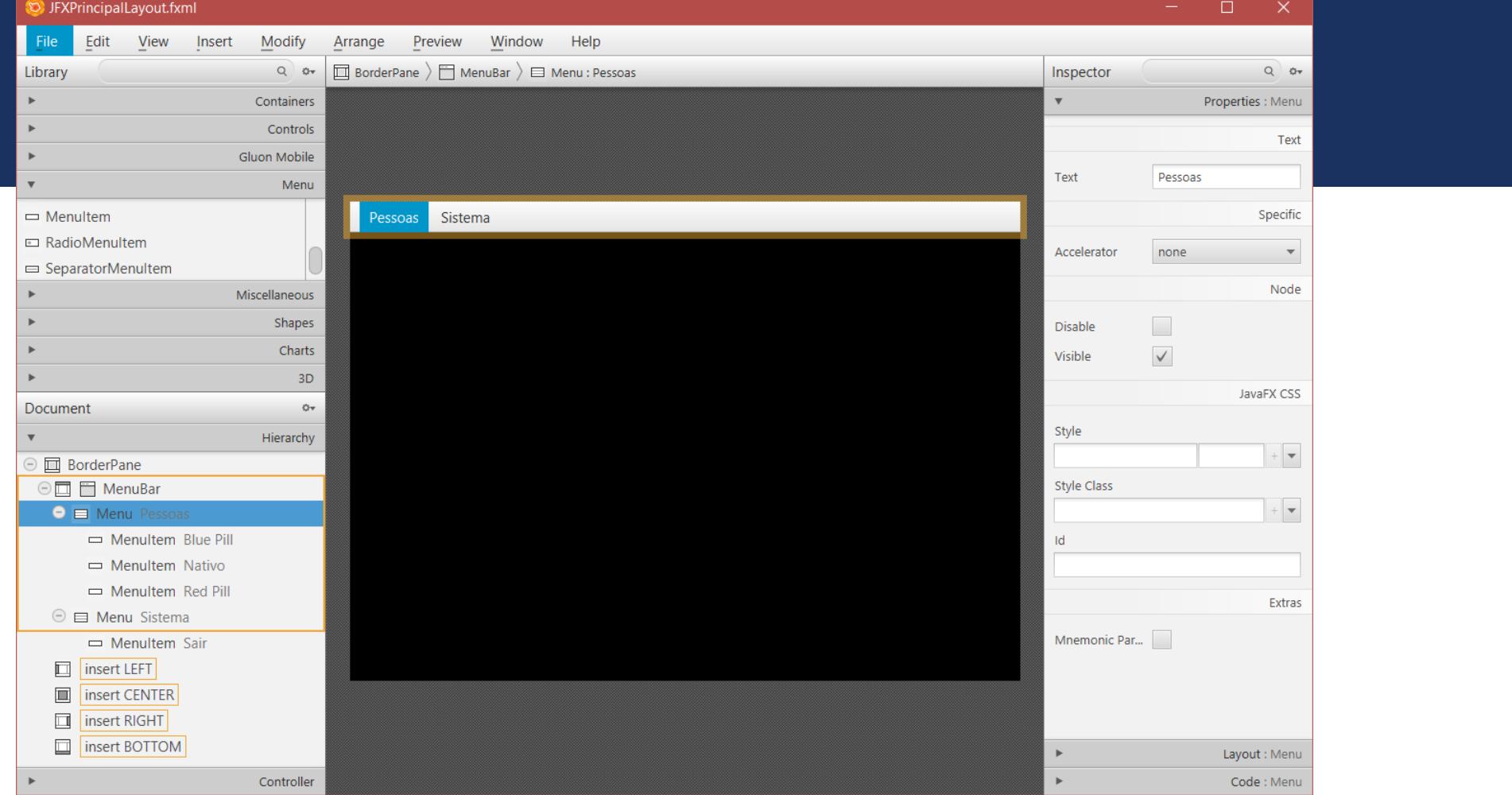




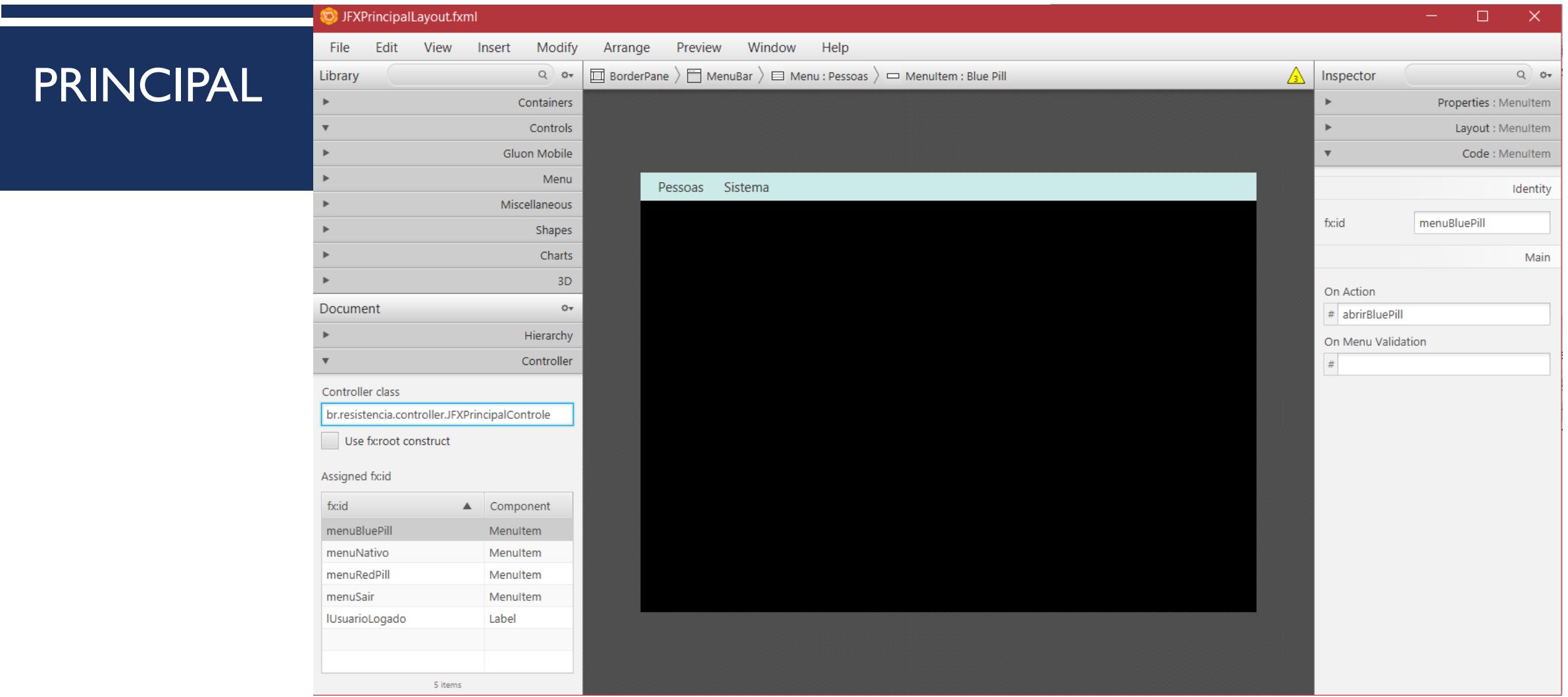
# ControleMenuBar.

PROF. RALFE DELLA CROCE FILHO

# PRINCIPAL



## ■ Alteração/Personalização doMenuBar.



- Definição dos identificadores dos controles (`fx:id`), dos eventos (`onAction`) e vínculo com a classe controle.

# NAVEGAÇÃO

```
JFXPrincipalControle.java ✘ JFXPrincipalLayout.fxml  
1 package br.resistencia.controller;  
2  
3 import javafx.fxml.FXML;  
4 import javafx.scene.control.MenuItem;  
5 import javafx.scene.control.Label;  
6  
7 public class JFXPrincipalControle {  
8  
9     // Componentes FXML  
10    @FXML MenuItem menuBluePill;  
11    @FXML MenuItem menuNativo;  
12    @FXML MenuItem menuRedPill;  
13    @FXML MenuItem menuSair;  
14    @FXML Label lUsuarioLogado;  
15  
16    // Métodos para eventos FXML  
17    @FXML public void abrirBluePill() {}  
18    @FXML public void abrirNativo() {}  
19    @FXML public void abrirRedPill() {}  
20    @FXML public void sair() {}  
21 }
```

PROF. RALFE DELLA CROCE FILHO

```
JFXPrincipalControle.java ✘  
15     // Atributos  
16     private Usuario usuarioLogado;  
17     private Stage palcoPrincipal;  
18  
19     // Métodos de acesso aos atributos  
20     public Usuario getUsuarioLogado() {  
21         return usuarioLogado;  
22     }  
23     public void setUsuarioLogado(Usuario usuarioLogado) {  
24         this.usuarioLogado = usuarioLogado;  
25     }  
26     public Stage getPalcoPrincipal() {  
27         return palcoPrincipal;  
28     }  
29     public void setPalcoPrincipal(Stage palcoPrincipal) {  
30         this.palcoPrincipal = palcoPrincipal;  
31     }  
32  
33     // Componentes FXML  
34     @FXML MenuItem menuBluePill;  
35     @FXML MenuItem menuNativo;  
36     @FXML MenuItem menuRedPill;  
37     @FXML MenuItem menuSair;  
38     @FXML Label lUsuarioLogado;  
39  
40     // Métodos de acesso ao conteúdo armazenado no lUsuarioLogado  
41     // Os métodos recebem e retornam String  
42     public String getlUsuarioLogado() {  
43         return lUsuarioLogado.getText();  
44     }  
45     public void setlUsuarioLogado(String lUsuarioLogado) {  
46         this.lUsuarioLogado.setText(lUsuarioLogado);  
47     }
```

- Declaração de atributos para controle do usuário logado e do formulário

# NAVEGAÇÃO

```
JFXLoginControle.java
```

```
37 // Atributo
38 private Stage palcoLogin;
39
40 // Métodos de acesso ao atributo
41 public Stage getPalcoLogin() {
42     return palcoLogin;
43 }
44 public void setPalcoLogin(Stage palcoLogin) {
45     this.palcoLogin = palcoLogin;
46 }
```

- Chamada do formulário principal definindo (nele) a referência do formulário criado e o usuário logado.

```
JFXLoginControle.java
```

```
84     // Valida o contato e senha
85     if(contatoInformado.equals(contatoUsuario) && senhaInformada.equals(senhaUsuario)) {
86         // Se for válido, invoca o formulário principal
87
88         // Cria um novo palco
89         Stage stage = new Stage();
90         // Criação de um objeto FXMLLoader para carregar o arquivo fxml (layout)
91         FXMLLoader loader = new FXMLLoader();
92         // Carregamento o arquivo fxml
93         loader.setLocation(getClass().getResource("/br/resistencia/view/JFXPrincipalLayout.fxml"));
94         // Criação do Layout Pane (gerenciador de layout) que será o node root (nó ou componente raiz)
95         // recebendo o layout (arquivo fxml)
96         BorderPane node = loader.load();
97         // Atribuição do componente raiz (e do layout) a cena
98         Scene scene = new Scene(node);
99         // Atribuição da cena ao palco
100        stage.setScene(scene);
101
102        // O objeto loader possui a referência da classe JFXPrincipalControle
103        JFXPrincipalControle principalControle = loader.getController();
104        // E acesso a seus métodos.
105        // A referência do palco criado é passada para posterior acesso
106        principalControle.setPalcoPrincipal(stage);
107        // O objeto do usuário logado é passado para o JFXPrincipalControle
108        principalControle.setUsuarioLogado(usuario);
109        // O usuário logado é identificado pelo nome
110        principalControle.setIUsuarioLogado("Usuário: " + usuario.getNome());
111
112        // Retira a barra superior da janela (icone, título, minimizar, maximizar e fechar)
113        stage.initStyle(StageStyle.UNDECORATED);
114        // Não permite o redimensionamento
115        stage.setResizable(false);
116        // Centraliza a apresentação
117        stage.centerOnScreen();
118        // Apresenta o formulário
119        stage.show();
120
121        // Fecha esse formulário
122        this.getPalcoLogin().close();
```

# NAVEGAÇÃO

The screenshot shows two code editors in an IDE. The top editor displays a Java file named JFXLoginControle.java with the following code:

```
95 BorderPane node = loader.load();  
96 // Atribuição do com J! Add throws declaration  
97 Scene scene = new Sc J! Surround with try/catch  
98 // Atribuição da cen  
99 stage.setScene(scene)  
100  
101 // O objeto loader p JFXPrincipalControle  
102 // E acesso a seus m  
103 // A referência do p principalControle.set  
104 // O objeto do usuári principalControle.set  
105 principalControle.set  
106  
107 // Retira a barra su stage.initStyle(StageStyle.UNDECORATED);  
108 // Não permite o redimensionamento stage.setResizable(false);  
109 // Centraliza a apresentação stage.centerOnScreen();  
110 // Apresenta o formulário stage.show();  
111  
112 // Não permite o redimensionamento stage.setResizable(false);  
113 // Centraliza a apresentação stage.centerOnScreen();  
114 // Apresenta o formulário stage.show();  
115  
116 // Não permite o redimensionamento stage.setResizable(false);  
117 // Centraliza a apresentação stage.centerOnScreen();  
118 // Apresenta o formulário stage.show();
```

A tooltip is visible over the line `loader.load()`, suggesting to "Add throws declaration" and "Surround with try/catch". The bottom editor also shows the same Java file with the following code:

```
...  
import java.io.IOException;  
import java.net.URL;  
import java.util.ResourceBundle;  
...  
// Métodos para eventos FXML  
@FXML public void ok() throws IOException {  
    // Declaração de variáveis  
    ...  
}
```

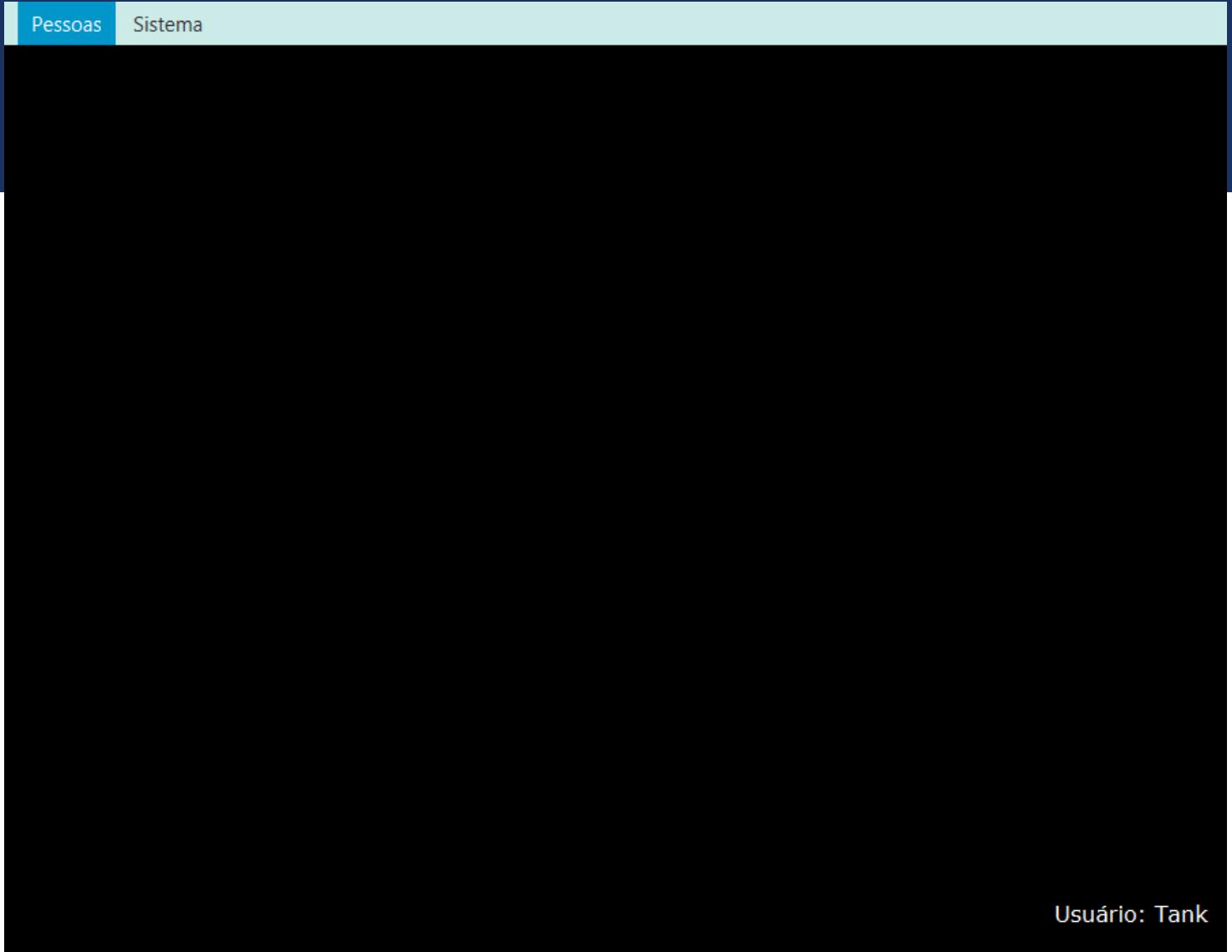
- Uma exception referente ao método load precisará ser tratada. Escolha a primeira opção (uso de throws).
- Tratamento de exceções serão abordados com detalhes posteriormente.

# NAVEGAÇÃO

## ■ Versão final do método start na classe Inicial

```
17 public void start(Stage primaryStage) throws Exception {  
18  
19     // Criação de um objeto FXMLLoader para carregar o arquivo fxml (layout)  
20     FXMLLoader loader = new FXMLLoader();  
21     // Carregamento o arquivo fxml  
22     loader.setLocation(getClass().getResource("/br/resistencia/view/JFXLoginLayout.fxml"));  
23     // Criação do Layout Pane (gerenciador de layout) que será o node root (nó ou componente raiz)  
24     // recebendo o layout (arquivo fxml)  
25     Pane nodeRoot = loader.load();  
26     // Atribuição do componente raiz (e do layout) a cena  
27     Scene scene = new Scene(nodeRoot);  
28     // Atribuição da cena ao palco primário  
29     primaryStage.setScene(scene);  
30  
31     // O objeto loader possui a referência da classe JFXLoginControle  
32     JFXLoginControle loginControle = loader.getController();  
33     // E acesso a seus métodos.  
34     // A referência do palco primário é passada para posterior acesso (fechamento)  
35     loginControle.setPalcoLogin(primaryStage);  
36  
37     // Retira a barra superior da janela (icone, titulo, minimizar, maximizar e fechar)  
38     primaryStage.initStyle(StageStyle.UNDECORATED);  
39     // Não permite o redimensionamento  
40     primaryStage.setResizable(false);  
41     // Centraliza a apresentação  
42     primaryStage.centerOnScreen();  
43  
44     // Apresenta o formulário  
45     primaryStage.show();  
46 }
```

# NAVEGAÇÃO

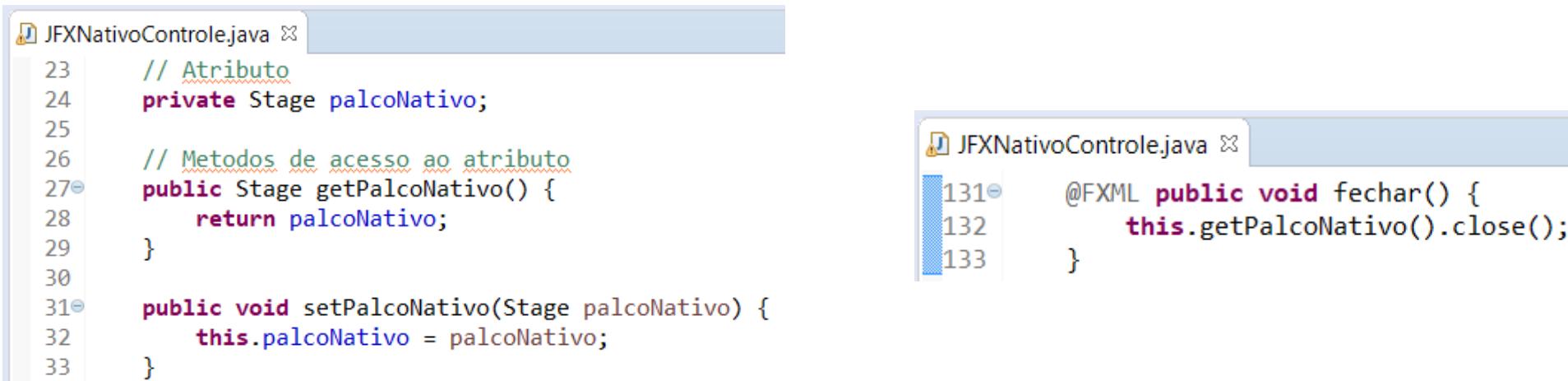


# NAVEGAÇÃO

```
59 @FXML public void abrirNative() throws IOException {  
60  
61     // Cria um novo palco  
62     Stage stage = new Stage();  
63     // Objeto FXMLLoader que carrega o arquivo fxml  
64     FXMLLoader loader = new FXMLLoader();  
65     // Carregamento o arquivo fxml  
66     loader.setLocation(getClass().getResource("/br/resistencia/view/JFXNativeLayout.fxml"));  
67     // Criação do Layout Pane (gerenciador de layout), que será o node/nó/componente raiz, e vinculo com o arquivo fxml  
68     BorderPane node = loader.load();  
69     // Atribuição do componente a cena  
70     Scene scene = new Scene(node);  
71     // Atribuição da cena ao novo palco  
72     stage.setScene(scene);  
73  
74     // O objeto loader possui a referência da classe JFXNativeControle  
75     JFXNativeControle nativeControle = loader.getController();  
76     // E acesso a seus métodos.  
77     // A referência do palco criado é passada para posterior acesso (fechamento)  
78     nativeControle.setPalcoNative(stage);  
79  
80     // Retira a barra superior da janela (icone, titulo, minimizar, maximizar e fechar)  
81     stage.initStyle(StageStyle.UNDECORATED);  
82     // Não permite o redimensionamento  
83     stage.setResizable(false);  
84     // Centraliza a apresentação  
85     stage.centerOnScreen();  
86     // Define o comportamento Modal (bloqueia os demais formulários enquanto ele estiver aberto)  
87     stage.initModality(Modality.WINDOW_MODAL);  
88     // Indica que esse formulário (principal) ficará bloqueado enquanto o formulário de nativos estiver ativo  
89     stage.initOwner(this.getPalcoPrincipal());  
90     // Apresenta o formulário  
91     stage.show();  
92 }
```

- Versão final do método `abrirNative` na classe `JFXPrincipalControle` definindo a referência do formulário criado.

# NAVEGAÇÃO



The image shows two snippets of Java code from a file named JFXNativeControle.java. The left snippet contains methods for setting and getting a Stage object named palcoNativo. The right snippet contains a method named fechar() which closes the Stage.

```
23 // Atributo
24 private Stage palcoNativo;
25
26 // Metodos de acesso ao atributo
27 public Stage getPalcoNativo() {
28     return palcoNativo;
29 }
30
31 public void setPalcoNativo(Stage palcoNativo) {
32     this.palcoNativo = palcoNativo;
33 }
```

```
131 @FXML public void fechar() {
132     this.getPalcoNativo().close();
133 }
```

- A partir da referência do formulário armazenado no atributo palcoNativo ele será fechado.

# NAVEGAÇÃO

Usuário: RedPill  
Login: trinity@zion  
Senha: .....  
Ok Cancelar

Pessoas Sistema

## Nativos

Registro:

Nome:

Contato:

Senha:

Função:

---

Perfil:

Programação  Lutas  Armas

---

Inserir Carregar Identificar

Limpar Fechar

---

---

---

ADENDO

CONTEXTUALIZAÇÃO

# CONTEXTUALIZAÇÃO

- A Orientação a Objetos possui uma cadeia de conceitos que estruturam, organizam e padronizam softwares. Todos esses conceitos são implementáveis, ou seja, eles foram (e são) criados para aplicação prática no desenvolvimento de softwares de acordo com esse paradigma.
- Contextualizar a aplicação prática desses conceitos é, didaticamente, o melhor caminho para a análise e entendimento de seus efetivos objetivos e importância. Com essa intenção, utilizarei dois caminhos...

# CONTEXTUALIZAÇÃO – MATRIX

- Na apresentação dos conceitos utilizarei exemplos baseados no universo da trilogia Matrix (1999, 2003 e 2003), filme dos irmãos Larry e Andy Wachowski (agora irmãs Lana e Lilly Wachowski). O motivo? Primeiro porque é um dos meus filmes favoritos, segundo porque, de forma extremamente visionária e competente, o filme aborda tanto a tecnologia (especificamente a área de softwares) quanto questões filosóficas fundamentais (pois é, se você achava que era “apenas” um filme de ação e ficção (ganhador de 4 óscares) acho que você não entendeu o filme).
- Obviamente, para o entendimento do conteúdo não é necessário assistir/conhecer o filme, mas, se você não assistiu, creio que quem está perdendo é você...

# CONTEXTUALIZAÇÃO

- A outra forma de contextualização será por meio de vários exercícios utilizando simulações de softwares que gerenciam, por exemplo, uma livraria, um controle bancário, uma agência de turismo, um controle escolar e uma imobiliária.
- **Observação importante:** realizar as atividades práticas é fundamental e indispensável. Será MUITO mais difícil (senão impossível) a assimilação dos conteúdos sem a prática.

## ESTEREÓTIPOS NO MODELO

- Um *stereotype* (anotação entre <>>) é um elemento que identifica a finalidade de outros elementos do modelo. Existe um conjunto padrão de estereótipos que podem ser aplicados e são utilizados para refinar o significado de um elemento do modelo.
- Utilizarei os estereótipos fora dos padrões previstos na UML para identificar onde os conceitos de Orientação a Objetos estão sendo aplicados no modelo com intenções puramente didáticas.