La Sénia
Institut d'Educació Secundària
Paiporta

Unión Europea
Fondo Social Europeo
El FSE invierte en tu futuro

Interface Gráfica de Usuario. Java FX

Curso 2023 - 2024

1. Interfaz Grafica de Usuario

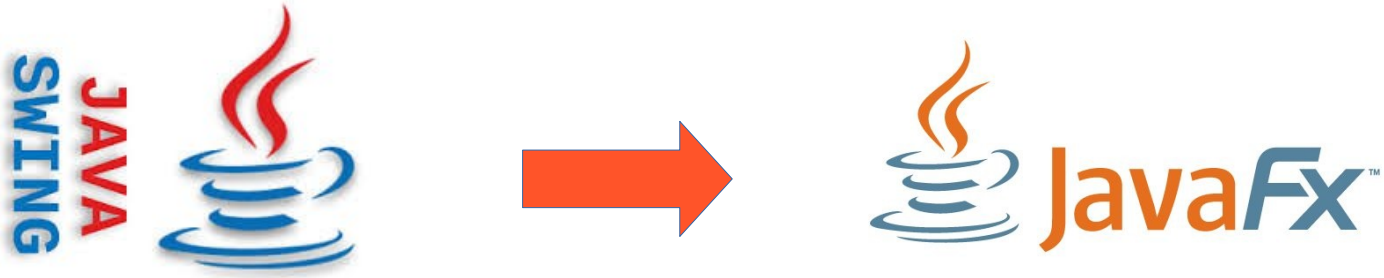
JavaFX:

- Instalación
- Primer proyecto
- Estructura de un proyecto. MVC
  - Modelo
  - Vista
  - Controlador
- Personalización

2. Instalación y puesta en marcha

Java FX

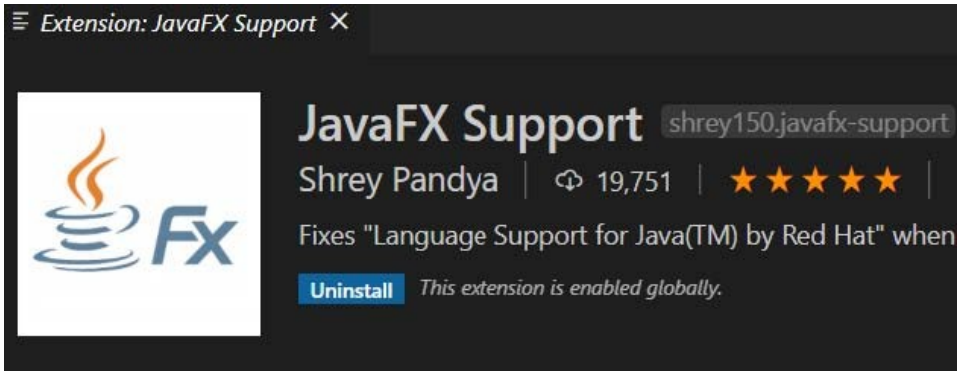
- Paquetes para gráficos y tecnologías para desarrollar, diseñar, crear, probar, depurar e implementar aplicaciones cliente.
- Diseñar aplicaciones de escritorio profesionales.
- Mejora de java swing



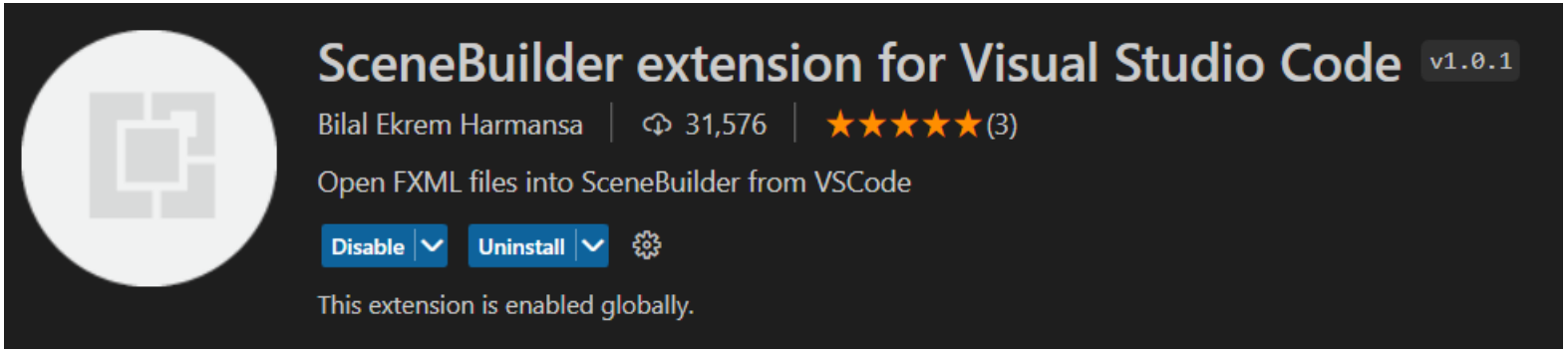
2. Instalación y puesta en marcha

Java FX

- Hay que instalar la extensión para JavaFX



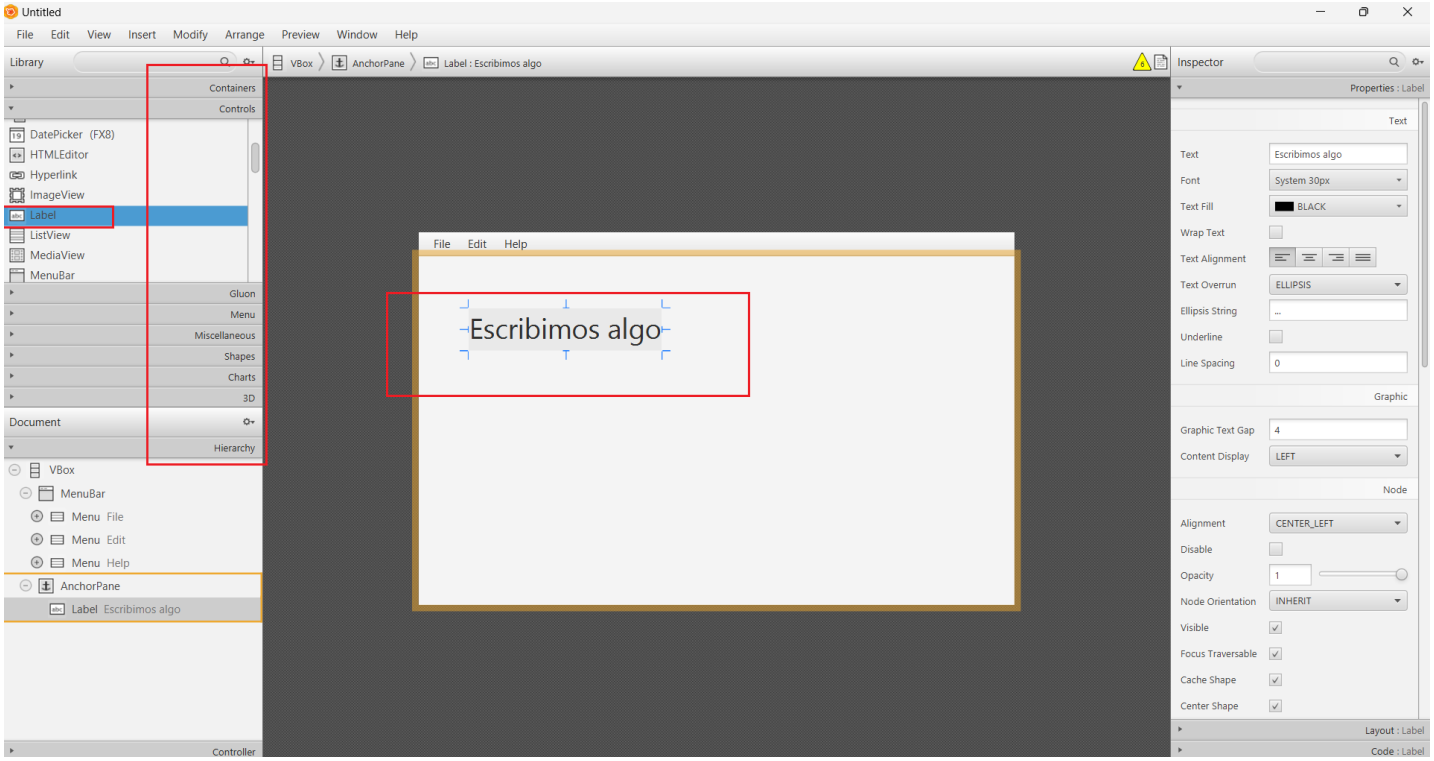
- Instalar la extensión para FXML (Scene Builder)



2. Instalación y puesta en marcha

Scene Builder (facilita la interacción)

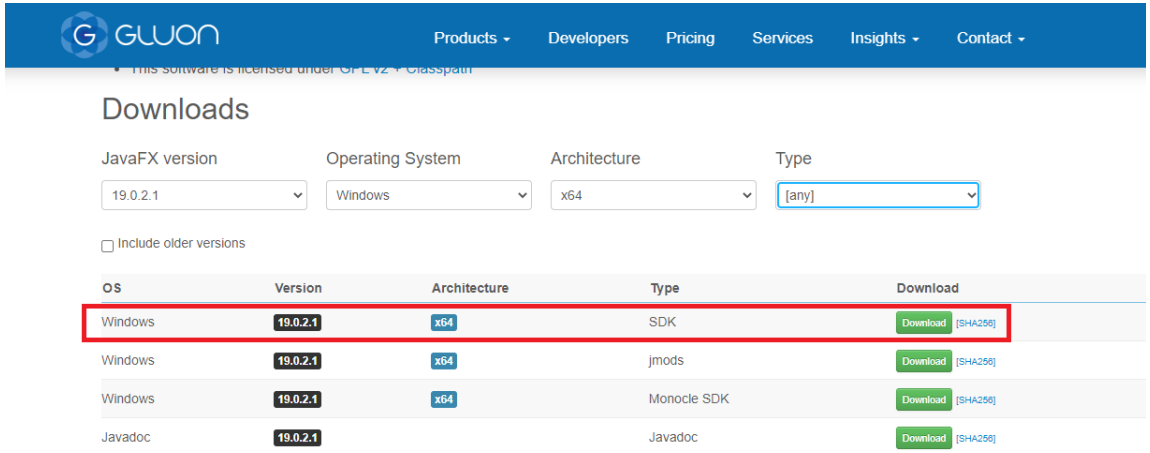
- Instalar la aplicación Scene Builder



2. Instalación y puesta en marcha

Instalación JavaFx

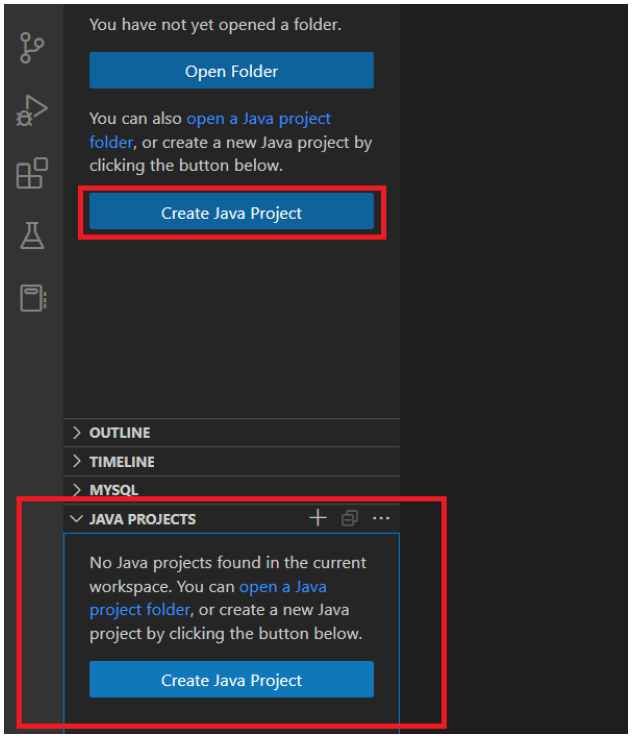
- Descargar JavaFX SDK

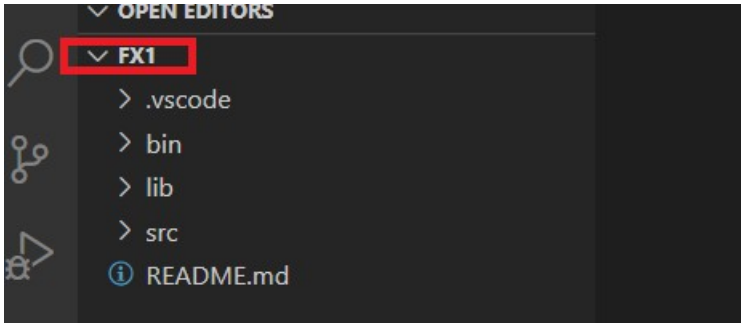


- Descomprimir en una carpeta

3. Primer proyecto JavaFx

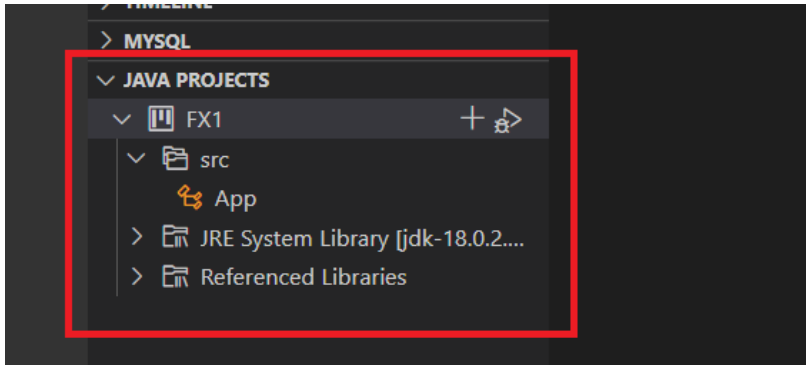
Crear un proyecto

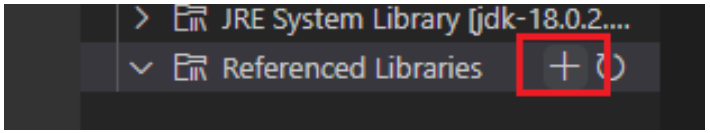


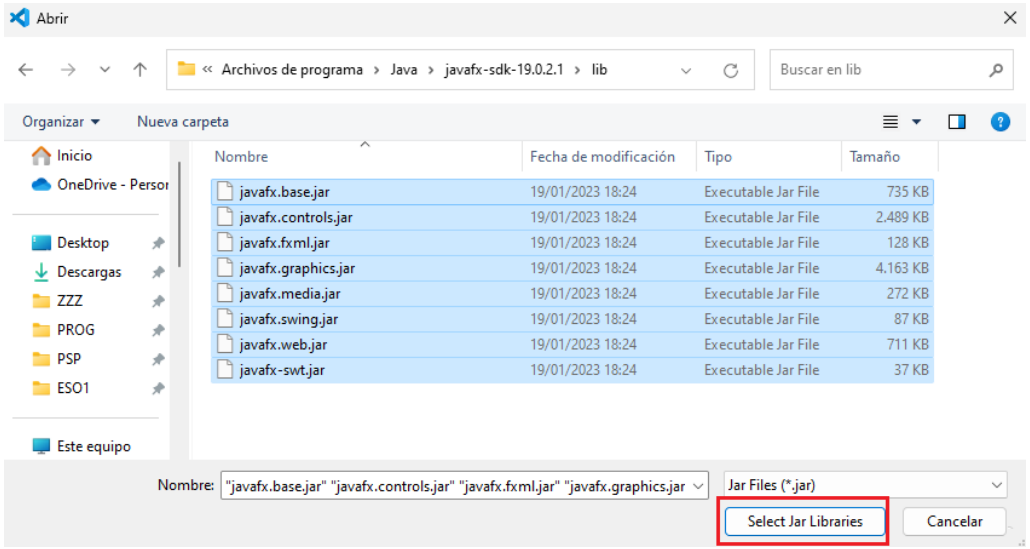


3. Primer proyecto JavaFx

Añadimos las librerías de referencia

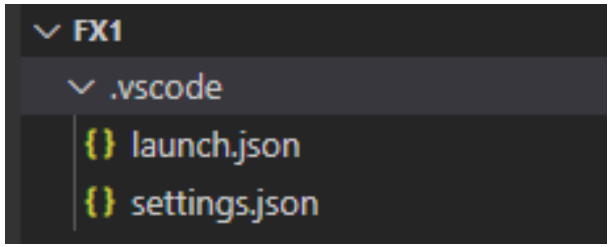






3. Primer proyecto JavaFx

Ficheros de configuración



settings.json

```
{
    "java.project.sourcePaths": ["src"],
    "java.project.outputPath": "bin",
    "java.project.referencedLibraries": [
        "lib/**/*.jar",
        "c:\\Program Files\\Java\\javafx-sdk-19.0.2.1\\lib\\javafx.base.jar",
        "c:\\Program Files\\Java\\javafx-sdk-19.0.2.1\\lib\\javafx.controls.jar",
        "c:\\Program Files\\Java\\javafx-sdk-19.0.2.1\\lib\\javafx.fxml.jar",
        "c:\\Program Files\\Java\\javafx-sdk-19.0.2.1\\lib\\javafx.graphics.jar",
        "c:\\Program Files\\Java\\javafx-sdk-19.0.2.1\\lib\\javafx.media.jar",
        "c:\\Program Files\\Java\\javafx-sdk-19.0.2.1\\lib\\javafx.swing.jar",
        "c:\\Program Files\\Java\\javafx-sdk-19.0.2.1\\lib\\javafx.web.jar",
        "c:\\Program Files\\Java\\javafx-sdk-19.0.2.1\\lib\\javafx-swt.jar"
    ]
}
```

Añadir al fichero launch.json "vmArgs": "**--module-path** <RUTA_SDK> **--add-modules** javafx.controls,javafx.fxml"

launch.json

```
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "type": "java",
            "name": "Current File",
            "request": "launch",
            "mainClass": "${file}"
        },
        {
            "type": "java",
            "name": "App",
            "request": "launch",
            "mainClass": "App",
            "vmArgs": "--module-path C:/'Program Files'/Java/javafx-sdk-19.0.2.1/lib --add-modules javafx.controls,javafx.fxml",
            "projectName": "FX1_8eb50344"
        }
    ]
}
```

3. Primer proyecto JavaFx

Primer ejemplo

App.java 1

src > App.java > App > start(Stage)

```java
import javafx.application.Application;
import javafx.event.*;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.control.Alert.*;
import javafx.scene.layout.*;

public class App extends Application{

    @Override
    public void start(Stage primaryStage){
        Label label = new Label(text: "Hola mundo");
        StackPane root = new StackPane();
        Button btn = new Button(text: "Pulsa aquí");
        btn.setOnAction(e -> {
            Alert alert = new Alert(AlertType.INFORMATION, contentText: "Hola Mundo!!");
            alert.showAndWait();
        });
        root.getChildren().addAll(label, btn);
        Scene scene = new Scene(root, 400, 200);

        primaryStage.setScene(scene);
        primaryStage.setTitle("Mi primer proyecto JavaFX");
        primaryStage.show();
    }


    Run | Debug
    public static void main(String[] args) {
        Application.launch(args);
    }

}
```

Main.java

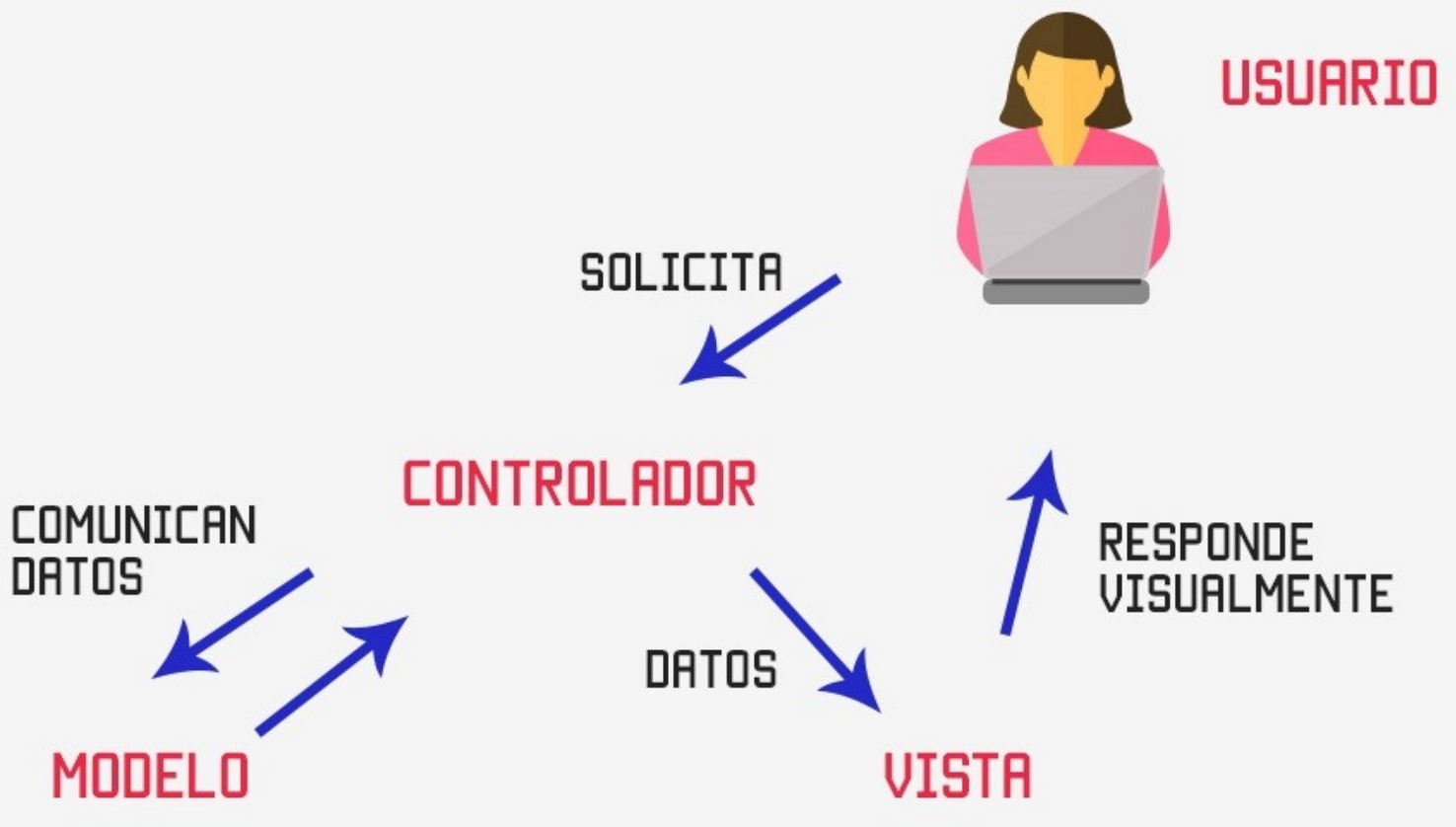
```
@Override
public void start(Stage primaryStage) {

     try {
    String fxml = "vista/HolaMundo.fxml";

    // Cargar la ventana
    Parent root = FXMLLoader.load(getClass().getClassLoader().getResource(fxml));

    // Cargar la Scene
    Scene scene = new Scene(root);

    // Asignar propiedades al Stage
    primaryStage.setTitle("Hola Mundo FX");
    primaryStage.setResizable(false);

    // Asignar la scene y mostrar
    primaryStage.setScene(scene);
    primaryStage.show();

     } catch(Exception e) {
    e.printStackTrace();
     }
}
```

4. Estructura de un proyecto

Modelo-Vista-Controlador

- El patrón MVC permite ordenar nuestro proyecto.
- Separa la capa de presentación de la lógica de la aplicación:
  - **Modelo**: Clases que representa objetos. Clases realizan la lógica del programa y se utilizan para almacenar y gestionar los datos.
  - **Vista**: Interfaz Gráfica de Usuario. Elementos que se muestran e interactúan con el usuario. Mostrar los datos almacenados.
  - **Controlador:** Clases que manejan los eventos. Conecta el modelo y la vista.
- Cuando un usuario interactúa con la **vista**, solicita al **controlador** la ejecución de un evento. El evento comunica con el **modelo** e intercambia datos. Estos datos son devueltos al **controlador** que los mostrará al usuario a través de la **vista**.

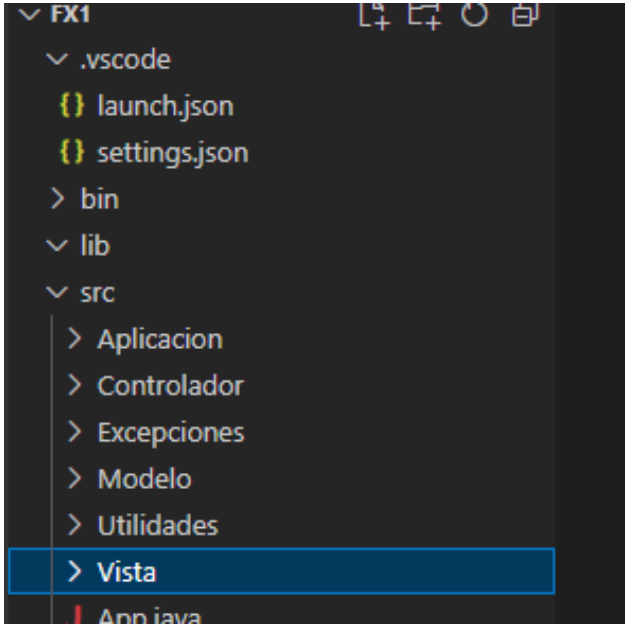
4. Estructura de un proyecto



4. Estructura de un proyecto

MVC

- Separamos en archivos diferentes: archivos para la IGU (vista), archivos para controlar los eventos (controlador) y archivos del dominio de nuestra aplicación (modelo).
  - Aplicacion
  - **Controlador**
  - Excepciones
  - **Modelo**
  - Utilidades
  - **Vista**



4. Estructura de un proyecto

MVC

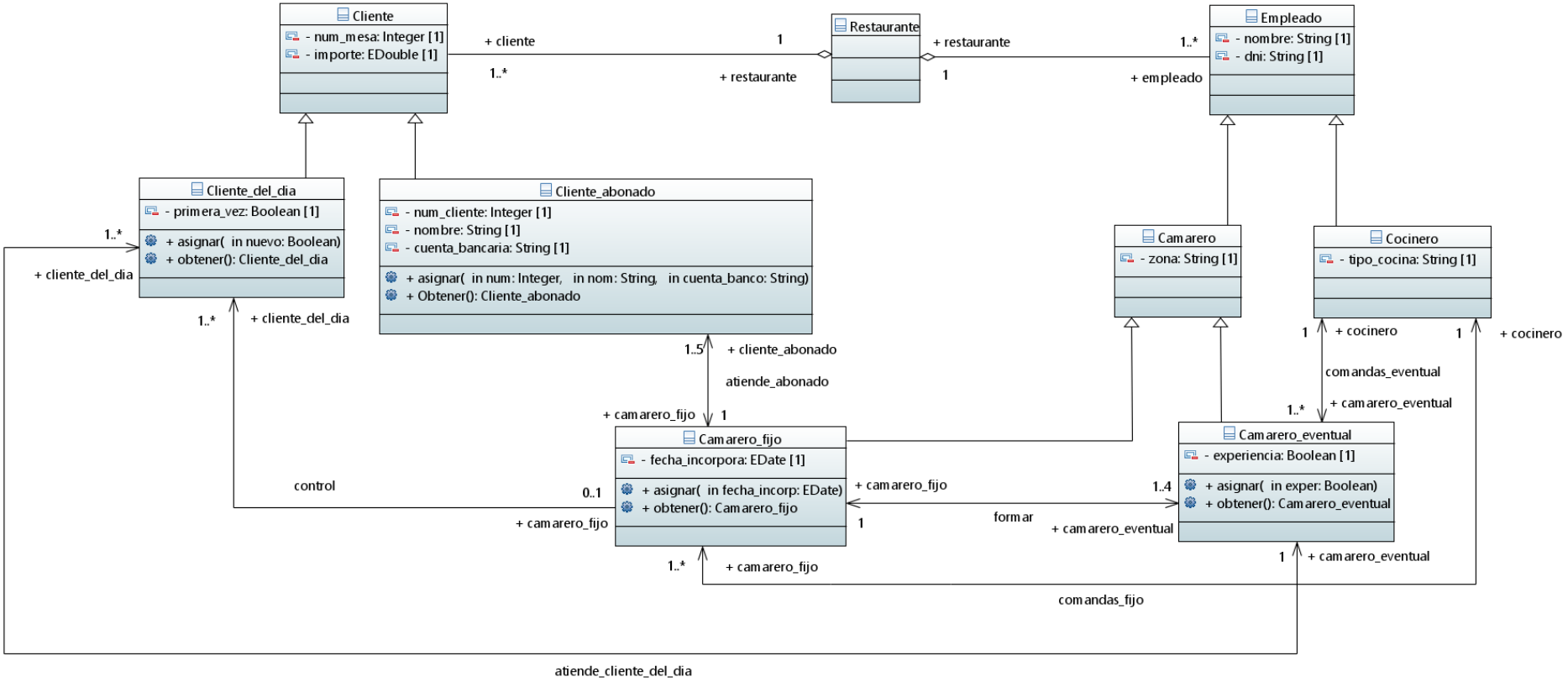
- El patrón Modelo-Vista-Controlador permite organizar el trabajo.
- La separación del trabajo permite el trabajo en paralelo con diferentes equipos de desarrollo.
- Mejora el **mantenimiento**: cada equipo de desarrollo puede corregir mantener y corregir sin tener en cuenta las otras partes del código.
- Hacer modelos, controladores y vistas independientes hace que la organización del código sea más simple, fácil de comprender y más fácil de mantener.

4.1 Modelo

Modelo

- Contiene las clases
- La separación del trabajo permite el trabajo en paralelo con diferentes equipos de desarrollo.
- Mejora el **mantenimiento**: cada equipo de desarrollo puede corregir mantener y corregir sin tener en cuenta las otras partes del código.
- Hacer modelos, controladores y vistas independientes hace que la organización del código sea más simple, fácil de comprender y más fácil de mantener.

4.1 Modelo

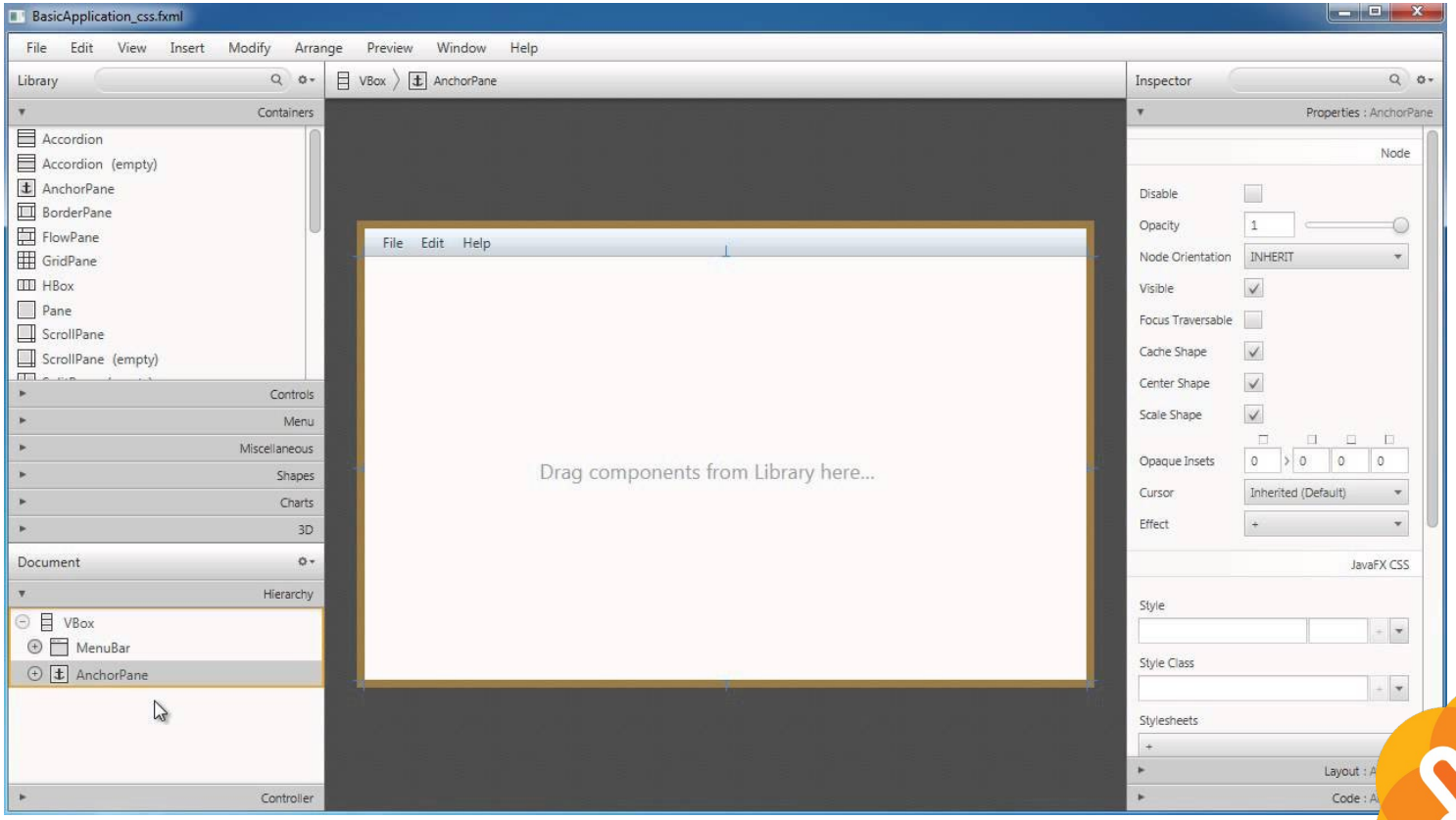

Cliente
- num_mesa: Integer [1]
- importe: EDouble [1]
+ cliente
1..*
1
+ restaurante
Restaurante
+ restaurante
1
1..*
+ empleado
Empleado
- nombre: String [1]
- dni: String [1]
Cliente_del_dia
- primera_vez: Boolean [1]
+ asignar(in nuevo: Boolean)
+ obtener(): Cliente_del_dia
1..*
+ cliente_del_dia
1..*
+ cliente_del_dia
Cliente_abonado
- num_cliente: Integer [1]
- nombre: String [1]
- cuenta_bancaria: String [1]
+ asignar(in num: Integer, in nom: String, in cuenta_banco: String)
+ Obtener(): Cliente_abonado
1..5
+ cliente_abonado
atiende_abonado
+ camarero_fijo
1
Camarero
- zona: String [1]
Cocinero
- tipo_cocina: String [1]
1
+ cocinero
1
+ cocinero
comandas_eventual
+ camarero_eventual
1..*
Camarero_fijo
- fecha_incorpora: EDate [1]
+ asignar(in fecha_incorp: EDate)
+ obtener(): Camarero_fijo
control
0..1
+ camarero_fijo
+ camarero_fijo
1
formar
1..4
+ camarero_eventual
Camarero_eventual
- experiencia: Boolean [1]
+ asignar(in exper: Boolean)
+ obtener(): Camarero_eventual
1..*
+ camarero_fijo
1
+ camarero_eventual
comandas_fijo
atiende_cliente_del_dia

4.2 Vista

Vista (IGU)

- Para crear las vistas se usa Scene Builder
-

4.2 Vista

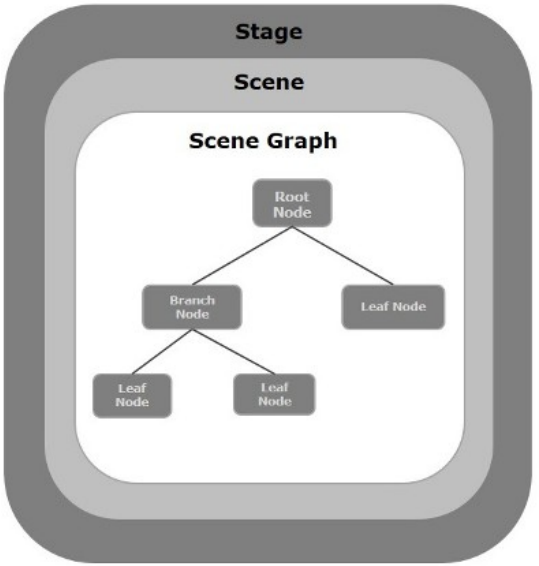

BasicApplication_css.fxml
File Edit View Insert Modify Arrange Preview Window Help
Library
Containers
Accordion
Accordion (empty)
AnchorPane
BorderPane
FlowPane
GridPane
HBox
Pane
ScrollPane
ScrollPane (empty)
Controls
Menu
Miscellaneous
Shapes
Charts
3D
Document
Hierarchy
VBox
MenuBar
AnchorPane
Controller
VBox AnchorPane
File Edit Help
Drag components from Library here...
Inspector
Properties : AnchorPane
Node
Disable
Opacity
Node Orientation
INHERIT
Visible
Focus Traversable
Cache Shape
Center Shape
Scale Shape
Opaque Insets
Cursor
Inherited (Default)
Effect
JavaFX CSS
Style
Style Class
Stylesheets

4.2 Vista

Componentes

- **Stage (ventana):** Es la ventana principal de la aplicación y viene determinada por el SO.
- **Scene (escena):** Describe todo lo que hay dentro de una ventana en una aplicación JavaFX.
- **Nodos:** Componentes gráficos que conforman la escena. Los nodos se almacenan en el fichero FXML.

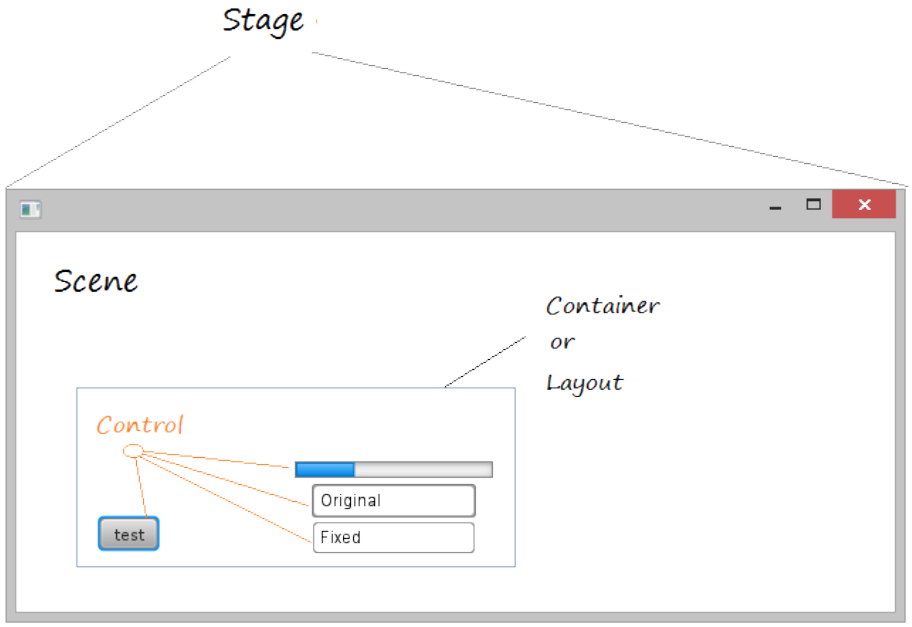
Stage es el contenedor de nivel superior, que, como mínimo, tiene una escena (**Scene**) y guarda un arbol de elementos gráficos (**Nodos**).



4.2 Vista

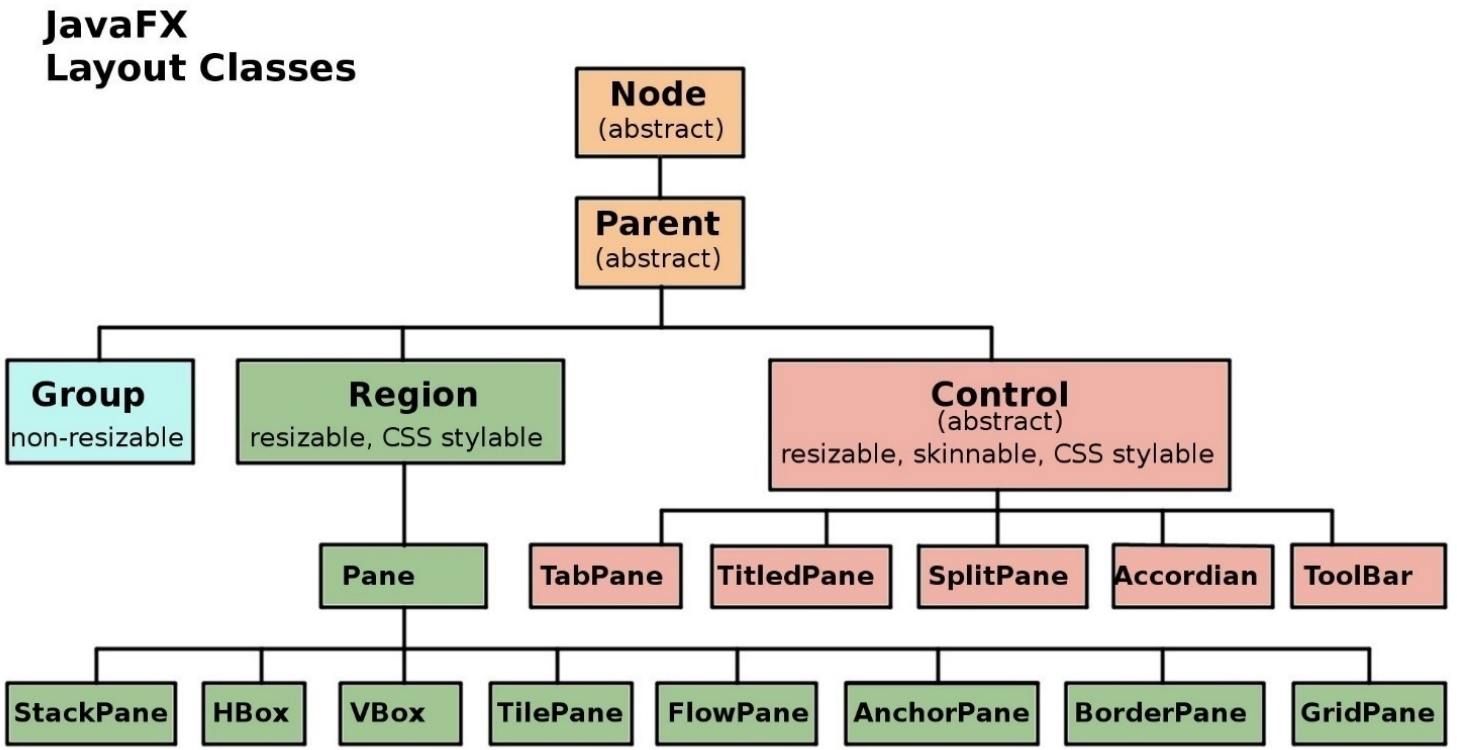
Componentes

- **Stage**: Se encarga del estilo y comportamiento de la IGU
- **Scene**: Almacena todos los nodos en un nodo raiz y maneja los eventos
- **Nodos**: Componentes que pertecen a un layout



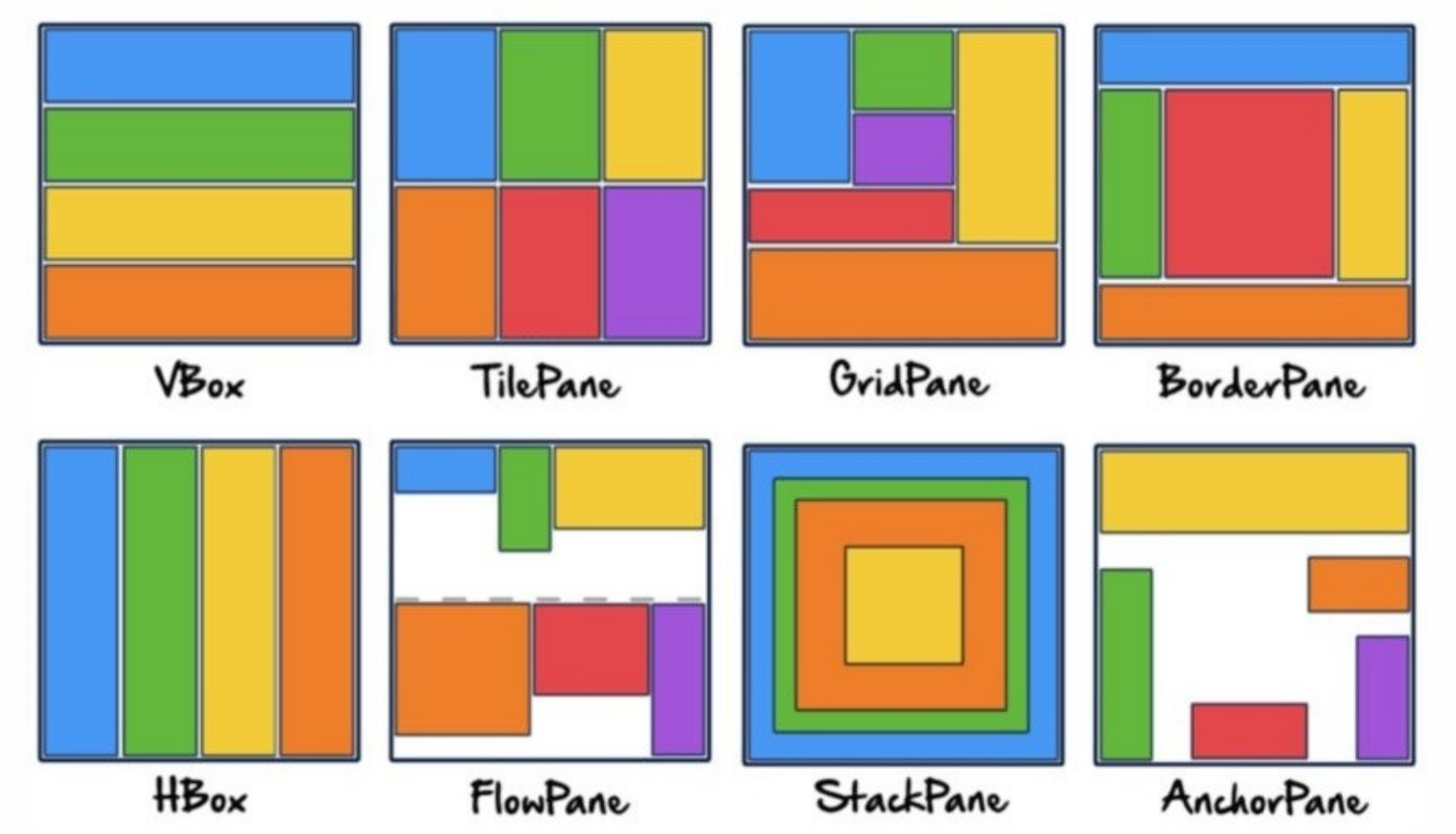
Layout classes

- **Stage**: Se encarga del estilo y comportamiento de la IGU



Layouts Pane

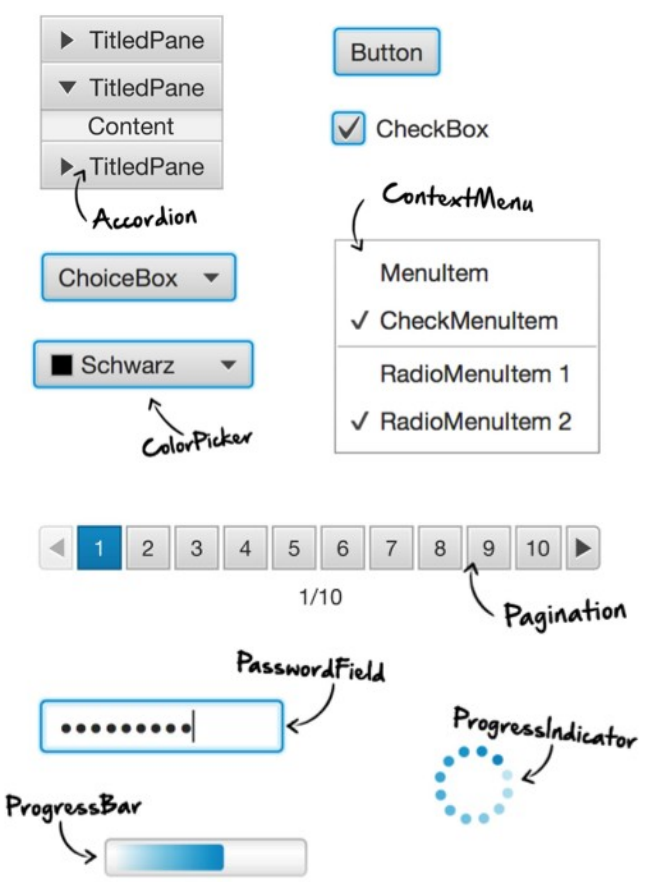
- **Tipos de Paneles**

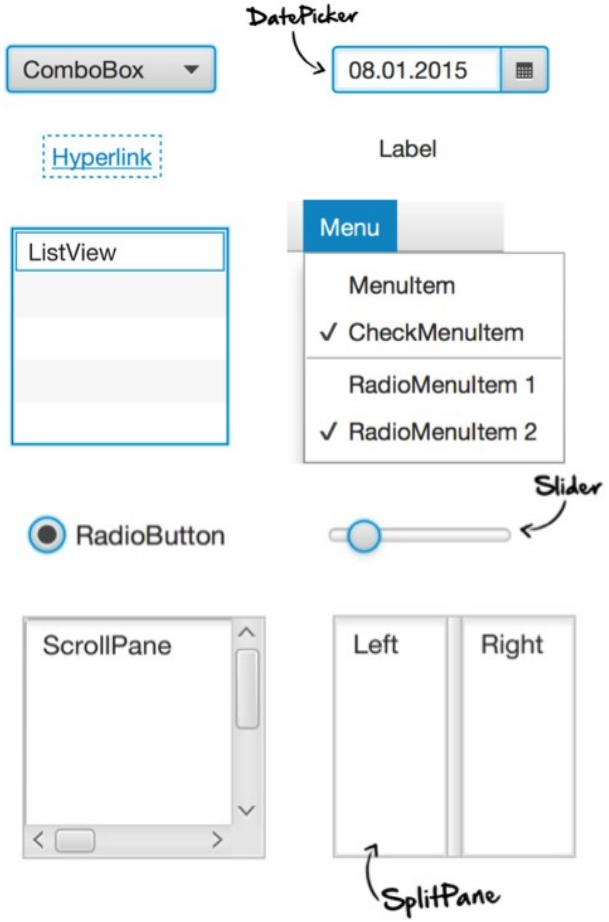


4.2 Vista

Panel	Descripción
VBox	Organiza de una forma sencilla una serie de nodos en una sola columna.
HBox	Organiza de una forma sencilla una serie de nodos en una sola fila.
TilePane	Un panel de mosaico es similar a un panel de flujo. El panel de diseño de TilePane coloca todos los nodos en una cuadrícula en la que cada celda o mosaico tiene el mismo tamaño. Los nodos se pueden colocar horizontalmente (en filas) o verticalmente (en columnas).
GridPane	Permite crear una cuadrícula flexible de filas y columnas en la que distribuir los nodos. Los nodos se pueden colocar en cualquier celda de la cuadrícula y pueden abarcar celdas según sea necesario. Es útil para crear formularios o cualquier diseño que esté organizado en filas y columnas.
BorderPane	Proporciona cinco regiones en las que colocar los nodos: superior, inferior, izquierda, derecha y central. Las regiones pueden ser de cualquier tamaño. Si su aplicación no necesita una de las regiones, no es necesario que la defina y no se le asigna espacio.
FlowPane	Los nodos se distribuyen consecutivamente y se ajustan al límite establecido para el panel. Los nodos pueden fluir verticalmente (en columnas) u horizontalmente (en filas).
StackPane	Organiza todos los nodos dentro de una sola pila con cada nodo nuevo agregado encima del nodo anterior. Este modelo de diseño proporciona una manera fácil de superponer texto en una forma o imagen o de superponer formas comunes para crear una forma compleja.
AnchorPane	Permite anclar nodos en la parte superior, inferior, izquierda, derecha o centro del panel. A medida que se cambia el tamaño de la ventana, los nodos mantienen su posición en relación con su punto de anclaje.

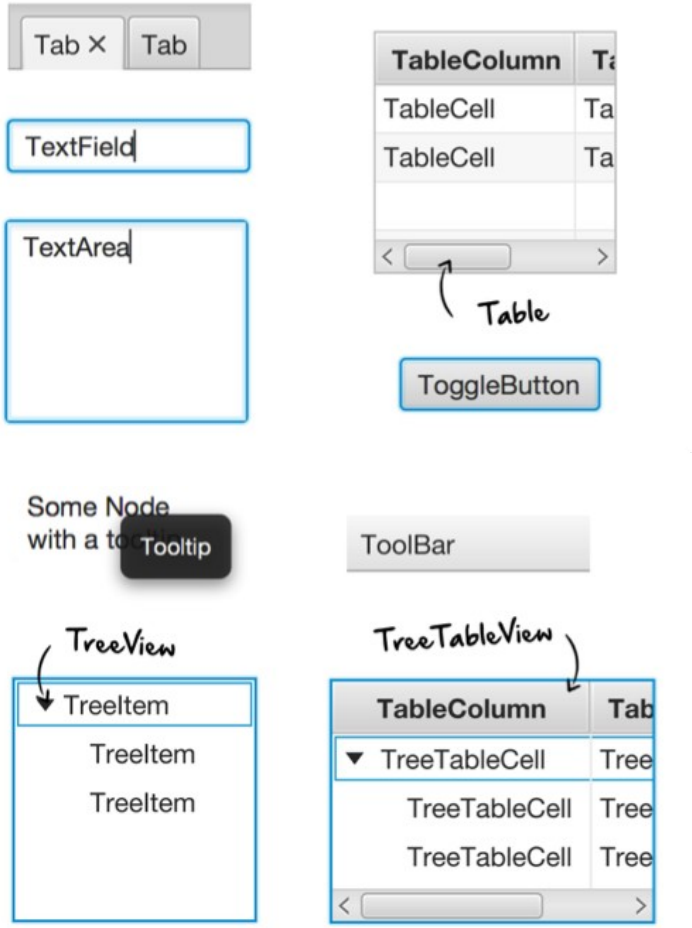
Componentes gráficos





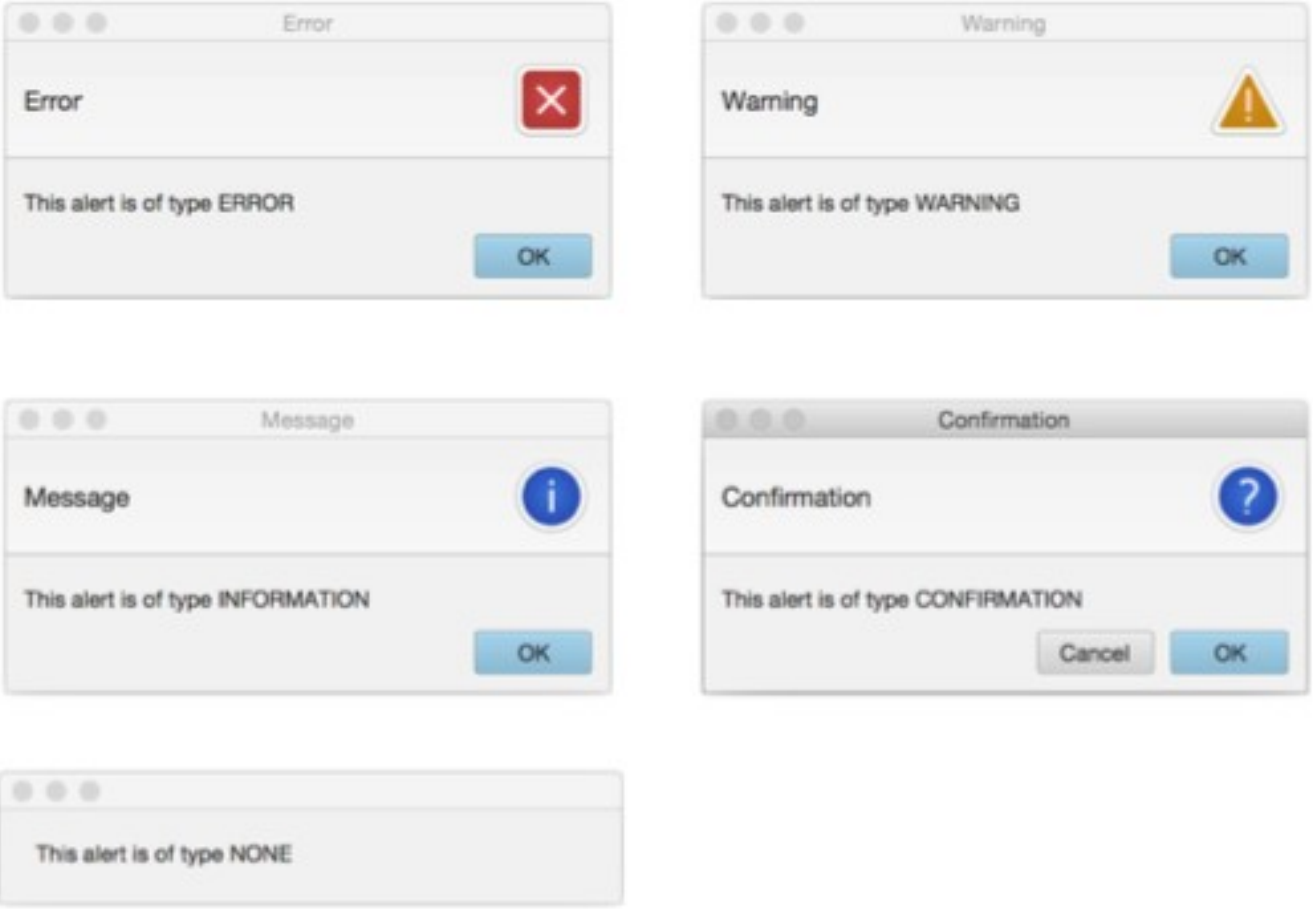
4.2 Vista

Componentes gráficos

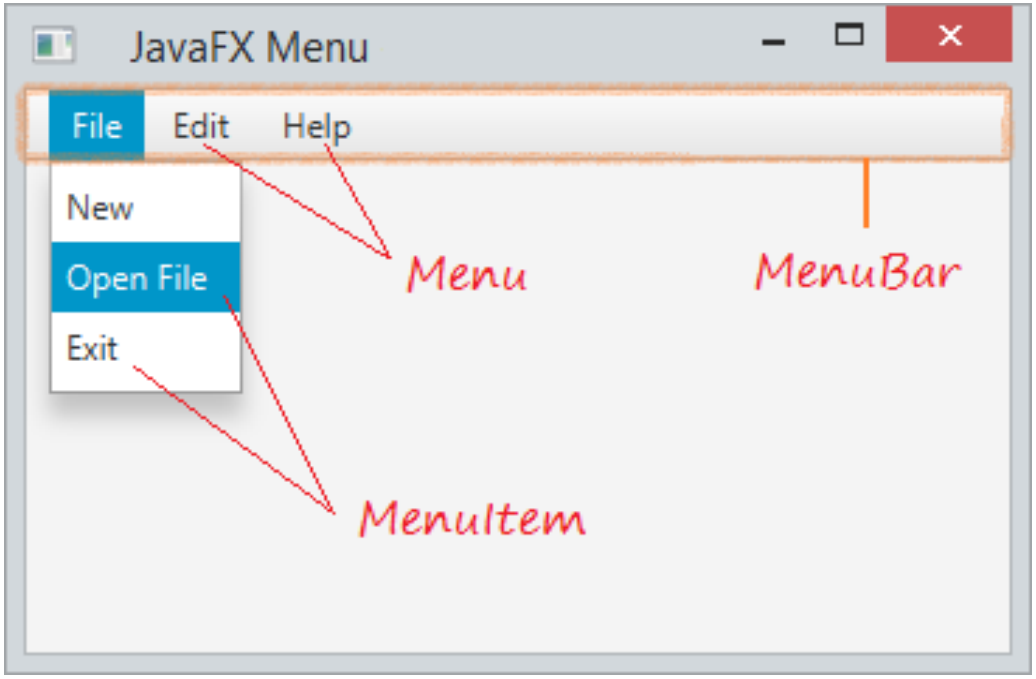


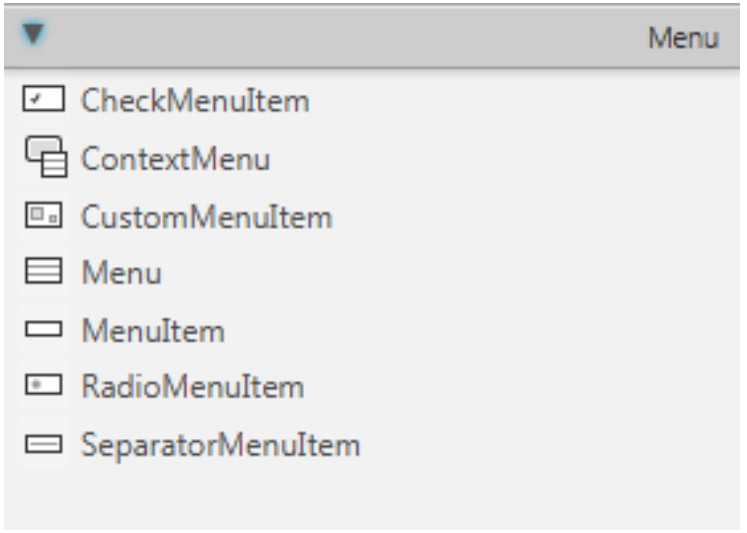
4.2 Vista

Dialogos y Alerts



Menus


JavaFX Menu
File
Edit
Help
New
Open File
Exit
Menu
MenuBar
MenuItem


Menu
CheckMenuItem
ContextMenu
CustomMenuItem
Menu
MenuItem
RadioMenuItem
SeparatorMenuItem

Menus

```
<MenuBar layoutX="0.0" layoutY="0.0" prefHeight="25.0" prefWidth="250.0">
 <menus>
   <Menu mnemonicParsing="false" text="Ver">
     <items>
       <MenuItem fx:id="menu_Historial" mnemonicParsing="false" text="Historial" />
       <MenuItem fx:id="menu_Salir" mnemonicParsing="false" text="Salir" />
     </items>
   </Menu>
   <Menu mnemonicParsing="false" text="Edición">
     <items>
       <MenuItem fx:id="menu_Copiar" mnemonicParsing="false" text="Copiar" />
       <MenuItem fx:id="menu_Pegar" mnemonicParsing="false" text="Pegar" />
     </items>
   </Menu>
   <Menu mnemonicParsing="false" text="Ayuda">
     <items>
       <MenuItem fx:id="menu_Ayuda" mnemonicParsing="false" text="Ver la Ayuda" />
       <MenuItem fx:id="menu_AcercaDe" mnemonicParsing="false" text="Acerca de Calculadora" />
     </items>
   </Menu>
 </menus>
</MenuBar>
```

Ejemplo

```
//************************************************************
// Métodos de la clase CalculadoraController
//************************************************************

@Override
public void initialize(URL url, ResourceBundle resourceBundle) {

// Crear la Calculadora
calculadora = new Calculadora();

// Asignar los Eventos de las opciones de menu
menu_Salir.setOnAction((event) -> salir());

// Asignar los Eventos a los botones
boton_sumar.setOnMouseClicked((event) -> asignarOperacion("+"));
}
```

4.3 Controlador

Eventos

- Los **eventos** se producen cuando el usuario interactúa con los componentes de la aplicación. Representan acciones.
- Generar el evento: uso del ratón, presionar botón del teclado, minimizar o cerrar la ventana.
- Tipos de eventos:
  - **ActionEvent:** Cualquier evento genérico.
  - **InputEvent:** Evento de una entrada de usuario.
  - **WindowEvent:** Evento sobre de mostrar u ocultar ventanas.

4.3 Controlador

Interacción Usuario - Evento

- Opción A. Asignar evento en la vista (.fxml)

Calculadora.fxml

```
<Button fx:id="boton_0" layoutX="160.0" layoutY="270.0" mnemonicParsing="false"
onAction="#pulsar" text="0" />
```

CalculadoraController.java

```
public class CalculadoraController {
    // Atributos graficos FXML
    @FXML private Button boton_0;

    @FXML
    void pulsar(ActionEvent event) {

    }
}
```

4.3 Controlador

Interacción Usuario - Evento

- Opción B. Asignar evento en la vista (.fxml)

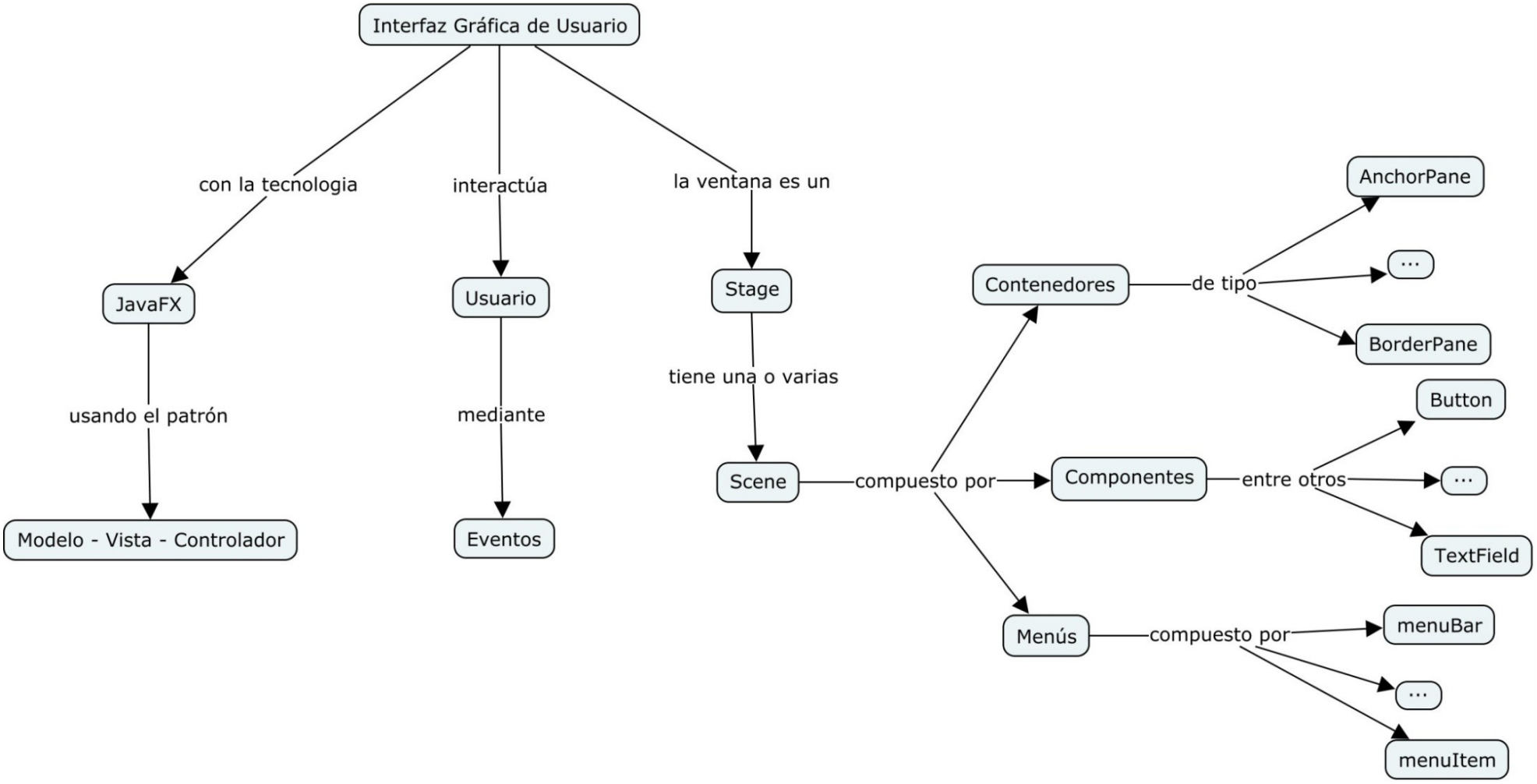
Calculadora.fxml

```
<Button fx:id="botón_0" layoutX="160.0" layoutY="270.0" mnemonicParsing="false"
text="0" />
```

CalculadoraController.java

```
public class CalculadoraController implements Initializable {

    // Atributos graficos FXML
    @FXML private Button boton_0;

    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        // Evento botones, apertura de ventanas
        boton_0.setOnMouseClicked((event) -> pulsarNumero(0));
    }
}
```

4. Resumen


Interfaz Gráfica de Usuario
con la tecnologia
interactúa
la ventana es un
JavaFX
Usuario
Stage
usando el patrón
mediante
tiene una o varias
Modelo - Vista - Controlador
Eventos
Scene
compuesto por
Contenedores
de tipo
AnchorPane
...
BorderPane
Componentes
entre otros
Button
...
TextField
Menús
compuesto por
menuBar
...
menuItem

Icono de Aplicación

Main.java

```
@Override
public void start(Stage primaryStage) {

    ...

    // Asignar icono de la aplicación
    primaryStage.getIcons().add(new
    Image(getClass().getResource("/vista/img/icon.png").toExternalForm()));

    ...

}
```

Mostrar otra ventana

```
//Mostrar otra ventana
private void mostrarVentanaAyuda(String rutaFXML, String titulo) {

    try{
      //Léeme el source del archivo que te digo fxml y te pongo el path
      FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource(rutaFXML));
      Parent root = (Parent) fxmlLoader.load();

      //Creamos un nuevo Stage (una nueva ventana vacía)
      Stage stage = new Stage();

      //Asignar al Stage la escena guardada en root
      stage.setTitle(titulo);
      stage.setResizable(false);
      stage.initModality(Modality.APPLICATION_MODAL);
      stage.setScene(new Scene(root));

      //Mostrar el Stage (ventana)
      stage.show();
    }
    catch (Exception e){
        e.printStackTrace();
    }
}
```

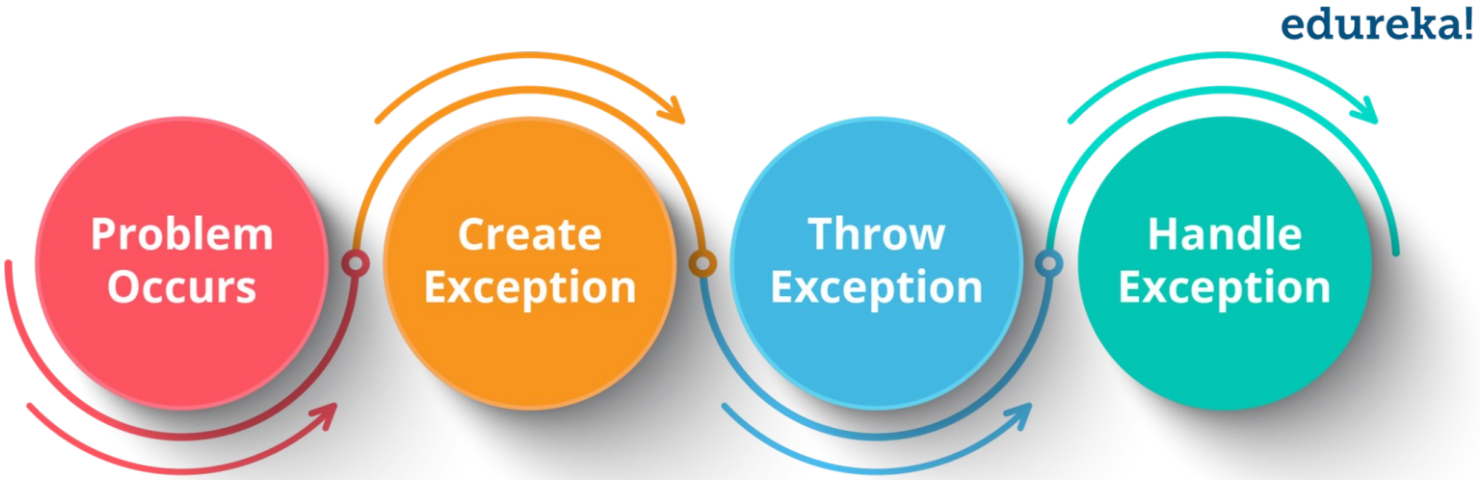
5. Personalización de la Aplicación

Excepciones

- Creamos un paquete de Excepciones.

DivisionPorCeroException.java

RaizNegativaException.java



CSS. Hojas de estilo

- Las hojas de estilo de JavaFX se basan en las reglas CSS.
- El objetivo de JavaFX CSS es permitir que los desarrolladores utilicen CSS para personalizar y desarrollar los controles JavaFX y objetos de gráficos de escena.
- Todos los conceptos CSS estudiados en Lenguaje de Marcas son válidos para el diseño de IGU con JavaFX.
- Los nombres de propiedad de JavaFX han sido prefijados con una extensión "-fx-".

```
-fx-background-color: #FBFCFC;
```

5. Personalización de la Aplicación

Main.java

```
@Override
public void start(Stage primaryStage) {
    ...

    // Asignar hoja de estilos
    scene.getStylesheets().add("/vista/css/estilos.css");
    ...

}
```

Estilos.css

```
.button {
    -fx-background-color: #FBFCFC;
    -fx-text-fill: #979A9A;
}
```

KeyListener

```
//1.- Asignar evento al TextArea
<TextArea    fx:id="display"    editable="false"    layoutX="10.0"    layoutY="37.0" onKeyPressed="#pulsarTecla" prefHeight="100.0" prefWidth="245.0" text="0" />
//2.- Desactivar que puedan quedarse el foco de la aplicación TODOS los botones
<Button fx:id="boton_0" focusTraversable="false" layoutX="10.0" layoutY="400.0" mnemonicParsing="false" prefHeight="45.0" prefWidth="95.0" text="0" />
//3.- Programar evento en el controlador
@FXML
void pulsarTecla(KeyEvent event) {
     switch (event.getCode()) {
         case DIGIT0: case NUMPAD0: insertarNumero("0"); break;
     }
}
//4.- Combinaciones de teclas
KeyCombination ctrlL = new KeyCodeCombination(KeyCode.L, KeyCodeCombination.CONTROL_DOWN);
if (ctrlL.match(event)) memoryClear();
```

5. Personalización de la Aplicación

Internacionalización

1.- Copiar la clase I18N.java al paquete “utilidades”

2.- *Main.java:*

```
// Indicar el idioma
Locale locale = new Locale("es");
ResourceBundle bundle = ResourceBundle.getBundle("strings", locale);
// Asignar propiedades al Stage
//primaryStage.setTitle("Formulario FX");
primaryStage.titleProperty().bind(I18N.createStringBinding("form.titulo"));
```

3.- *Calculadora.fxml*:

```
<Label fx:id="lbl_autor" layoutX="121.0" layoutY="14.0" text="%form.autor"/>
```

4.- *CalculadoraController.fxml*:

```
@FXML
void initialize() {
    //autor de la aplicación
    lbl_autor.textProperty().bind(I18N.createStringBinding("form.autor"));
}
```

5.- Crear un archivo Strings por cada idioma

6. Despliegue de la Aplicación

Exportar proyecto a formato jar

- Exportarlo a **.jar**.
- Opción **runnable JAR file.**
- En library handling escogeremos la segunda opción. Esta opción empaqueta las librerías necesarias dentro del **.jar**.
- Este .jar podrá ejecutarse por consola con el comando:

  java -jar SumaFX.jar

- También se puede hacer doble clic sobre el ejecutable o crear un acceso directo.
- Existen programas para exportar el **.jar** a **.exe** si se considera oportuno.