9

INTERFAZ DE USUARIO: JAVAFX, SCENEBUILDER. INTRODUCCIÓN AL MVC

9.1 INTRODUCCIÓN

T1. JavaFX es una tecnología que nos permite crear aplicaciones de escritorio RIA (Ritch Internet Applications), esto es, aplicaciones web que tienen las características y capacidades de aplicaciones de escritorio, incluyendo aplicaciones multimedia interactivas que pueden ejecutarse en una amplia variedad de dispositivos. JavaFX está destinado a reemplazar a Swing como la biblioteca de GUI estándar para Java SE.

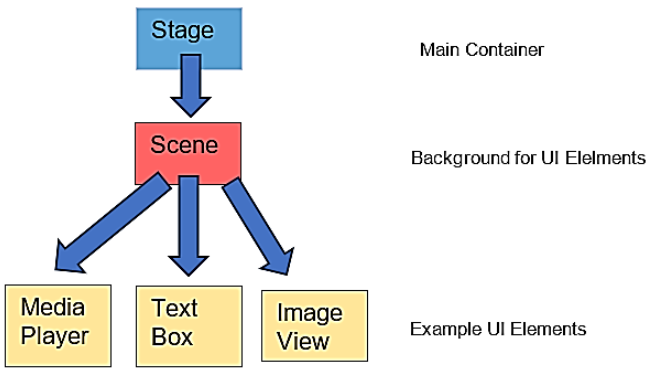
La biblioteca de JavaFX está escrita como una API de Java, las aplicaciones JavaFX pueden hacer referencia a APIs de código de cualquier biblioteca Java. Por ejemplo, las aplicaciones JavaFX pueden utilizar las bibliotecas de API de Java para acceder a las capacidades del sistema nativas y conectarse a aplicaciones de middleware basadas en servidor.

La apariencia de las aplicaciones JavaFX se pueden personalizar. Las Hojas de Estilo en Cascada (CSS) separan la apariencia y estilo de la lógica de la aplicación para que los desarrolladores puedan concentrarse en el código. Los diseñadores gráficos pueden personalizar fácilmente el aspecto y el estilo de la aplicación a través de CSS. Si se tiene un diseño de fondo de la web, o si se desea separar la interfaz de usuario (UI) y la lógica de servidor, entonces, se pueden desarrollar los aspectos de la presentación de la interfaz de usuario en el lenguaje de scripting FXML y utilizar el código de Java para la aplicación lógica. Si se prefiere diseñar interfaces de usuario sin necesidad de escribir código, entonces, utilizaremos JavaFX Scene Builder. Al diseñar la interfaz de usuario con javaFX Scene Builder el crea código de marcado FXML que puede portarse a un entorno de desarrollo integrado (IDE) de forma que los desarrolladores pueden añadir la lógica de negocio.

Lo primero será explicar los tres conceptos que tiene JavaFX:

- El **escenario,** que es representado por la clase *Stage*. El escenario es el que representa al contenedor general de JavaFX.
- La **escena**, es representada por la clase *Scene* y es la que tiene el contenido de lo que queremos representar. La escena, por lógica se monta sobre el escenario.

- Los **nodos** de la escena, son los elementos que componen la escena. La clase superior que representa estos nodos es un Panel.

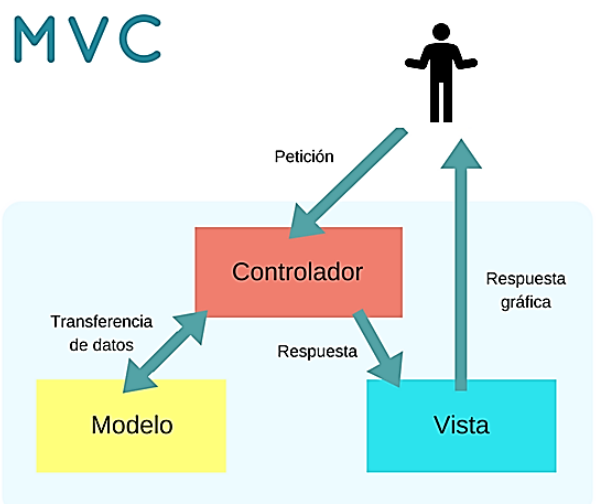


9.1.1 Modelo vista controlador (MVC)

Modelo Vista Controlador (MVC) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

Se trata de un modelo muy maduro y que ha demostrado su validez a lo largo de los años en todo tipo de aplicaciones, y sobre multitud de lenguajes y plataformas de desarrollo.

- El **Modelo** que contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia.
- La **Vista**, o interfaz de usuario, que compone la información que se envía al cliente y los mecanismos interacción con éste.
- El **Controlador**, que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

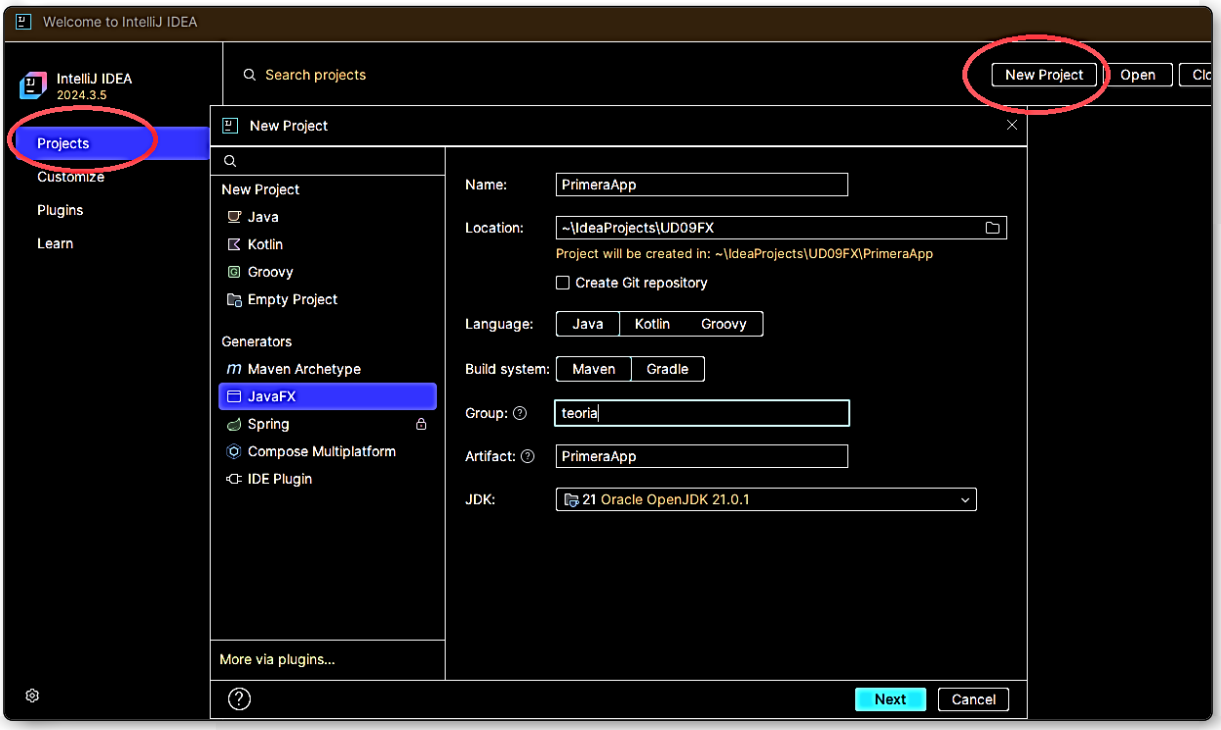


9.2 CREAR UN PROYECTO JAVAFX NUEVO

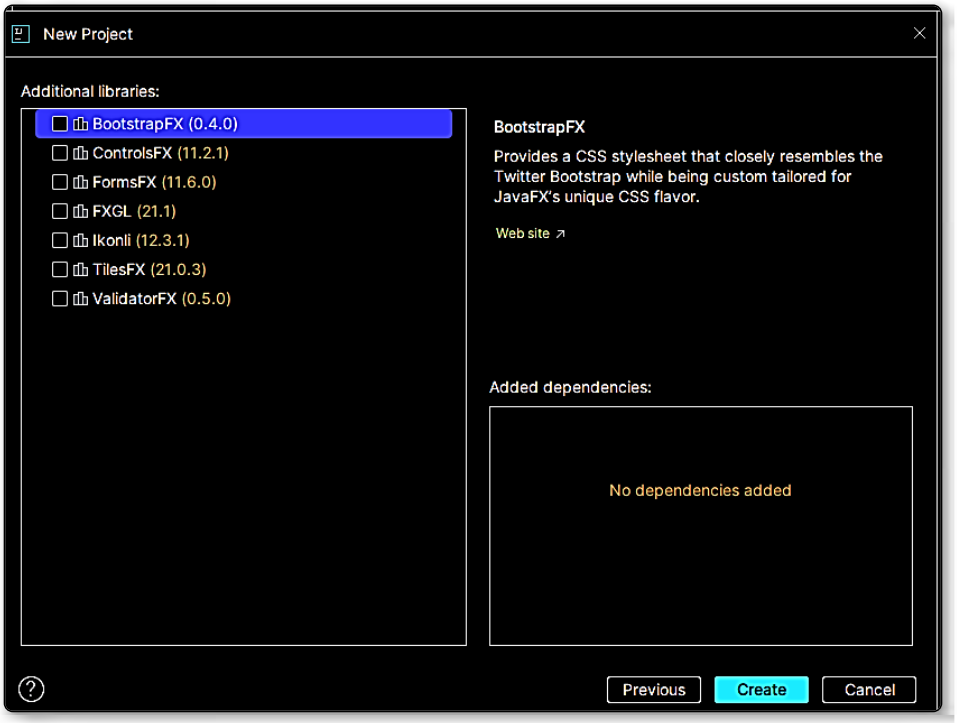
JavaFx es un software para crear y distribuir aplicaciones de escritorio que pueden ejecutarse en una amplia variedad de dispositivos. JavaFx intenta reemplazar a Swing como la librería estándar GUI para Java SE.

ACTIVIDAD TP1:

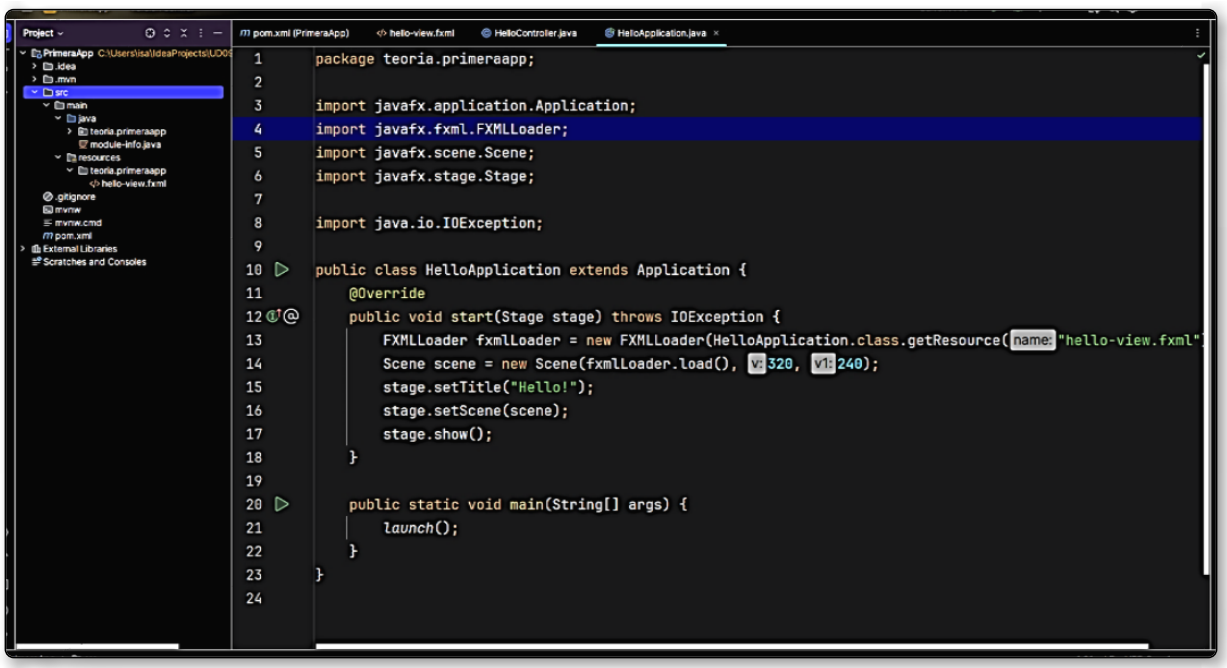
Crear un proyecto nuevo proyecto JavaFx desde Intellij. Para ello iniciamos el programa y seleccionamos ***Project. New Project*** y rellenamos los datos como en la figura:



A continuación, pulsamos la tecla ***Next*** y surgirá la pantalla donde se podrían seleccionar librerías adicionales que podrías incluir y que nosotros no vamos a poner porque queremos aprender puro javaFX en lugar de hacer algo previamente ya creado.



En esta pantalla, sin poner ningún check a las librerías, pulsamos el botón ***Create*** y vemos aparecer nuestro primer proyecto:



Si observamos bien la imagen vemos varios archivos que se han generado automáticamente: *HelloApplication.java, HelloController.java, hello-view.fxml* y pom.xml. Iremos viendo cosillas.

Analicemos el código de programa principal HelloApplication

```
package teoria.primeraapp;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.stage.Stage;

import java.io.IOException;

public class HelloApplication extends Application {
    @Override
    public void start(Stage stage) throws IOException {
        FXMLLoader fxmlLoader = new FXMLLoader(HelloApplication.class.
getResource("hello-view.fxml"));
        Scene scene = new Scene(fxmlLoader.load(), 320, 240);
        stage.setTitle("Hello!");
        stage.setScene(scene);
        stage.show();
    }

    public static void main(String[] args) {
        launch();
    }
}
```

Lo siguiente es importante para el programador aquí:

1. El primer *launch* crea una nueva instancia de la clase de *Application* (*HelloApplication* en este caso). La clase *de Application*, por lo tanto, necesita un constructor sin argumentos.
2. Se llama a init() en la instancia de la *Application* creada. En este caso, la implementación predeterminada de la *Application* no hace nada.
3. Se llama a *start* para la instancia de A*pplication* y la *Stage* primaria (= ventana) se pasa al método. Este método se llama automáticamente en el subproceso de la aplicación JavaFX (subproceso de plataforma).
4. La aplicación se ejecuta hasta que la plataforma determina que es hora de apagarse. Esto se hace cuando se cierra la última ventana en este caso.
5. El método de stop se invoca en la instancia de la *Application*. En este caso la implementación desde la *Application* no hace nada. Este método se llama automáticamente en el subproceso de la aplicación JavaFX (subproceso de plataforma).

En el método de start se construye el gráfico de escena. En este caso la interfaz de usuario solo contiene un botón. Aunque aquí todavía "no sabemos" de donde sale (de hello-view.fxml).

Se crea una *Scene* para mostrar estos Node. Finalmente, la *Scene* se agrega al *Stage* que es la ventana que muestra la IU completa.

9.2.1 La clase Stage

Como ya hemos visto en el ejemplo, cuando creamos una aplicación JavaFX, la clase principal extiende a la clase *Application*, y sobreescribe un método *start*, que crea un objeto *Stage* como parámetro. Este objeto *Stage* es una referencia al contenedor principal de nuestra aplicación. Será una ventana en sistemas operativos como Windows, Mac OS X o Linux; pero también puede ser una ventana completa si nuestra aplicación se ejecuta en un Smartphone o en una Tablet.

La clase *Stage* proporciona algunos métodos útiles para cambiar algunas características de tamaño, comportamiento, etc. Algunos métodos útiles son los siguientes:

- ***setTitle(String)***: para poner título a la aplicación; que podemos ver en la barra de título de arriba en la ventana de la aplicación.
- ***setScene(Scene)***: establece la escena de nuestra aplicación, donde colocaremos todos los controles. Puede haber más de una escena en un escenario.
- ***show***: hace visible la aplicación (stage o escenario), y continúa ejecutando las instrucciones siguientes.
- ***showAndWait***: hace visible la aplicación (stage), y espera hasta que se cierra para continuar.
- ***setMinWidth(double), setMaxWidth(double)***: establece respectivamente el ancho mínimo y máximo de la ventana, así que no será posible redimensionar la ventana más allá de estos límites.
- ***setMinHeight(double), setMaxHeight(double)***: establece respectivamente el alto mínimo y máximo de la ventana, así que no será posible redimensionar la ventana más allá de estos límites.
- ***getMinWidth, getMaxWidth, getMinHeight, getMaxHeight***: obtiene la altura o anchura máxima en la aplicación.
- ***setFullScreen(boolean)***: establece si nuestra aplicación se ejecutará a pantalla completa (por lo que no sería redimensionable y no habrá ninguna barra superior), o no.
- ***setMaximized(boolean)***: establece si nuestra aplicación se maximiza o no.
- ***setIconified(boolean)***: establece si nuestra aplicación se iconifica (minimiza) o no.
- ***setResizable(boolean)***: establece si nuestra aplicación es redimensionable o no.

9.2.2 La clase Scene

Cada programa JavaFX tiene al menos un objeto Scene para contener todos los controles de la aplicación. Cuando se crea, necesitamos especificar su nodo principal (el que devuelve el FXMLLoader cuando analiza sintácticamente (parsea) un FXML).

```
FXMLLoader fxmlLoader = new
    FXMLLoader(HelloApplication.class.getResource("hello-view.fxml"));

Scene scene = new Scene(fxmlLoader.load(), 320, 240);
stage.setTitle("Hello!");
stage.setScene(scene);
stage.show();
```

```
El método load es el que nos crea la estructura de arbol de los nodos.
```

La clase Scene tiene métodos útiles, como por ejemplo:

- ***getWidth, getHeight***: coge la anchura y la altura de la escena actual.
- ***getX, getY***: coge las coordenadas de la escena actual en la pantalla refiriéndose a la esquina superior izquierda.
- ***setRoot (Parent)***: establece un nuevo gestor de layout como el nodo principal de esta escena.

Un Stage puede tener múltiples Scenes, y puede cambiar de una a otra llamando a su método setScene.

Veamos ahora el código **de hello-view.fxml**

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.layout.VBox?>

<?import javafx.scene.control.Button?>
<VBox alignment="CENTER"
  spacing="20.0"
  xmlns:fx="http://javafx.com/fxml"
        fx:controller="teoria.primeraapp.HelloController">
    <padding>
        <Insets bottom="20.0" left="20.0" right="20.0" top="20.0"/>
    </padding>

    <Label fx:id="welcomeText"/>
    <Button text="Hello!" onAction="#onHelloButtonClick"/>
</VBox>
```

¡Si sabemos un poquito de html y css, es fácil de comprender que hay un botón con el texto Hello! que tiene asociada una acción llamada *onHelloButtonClick.* Indicamos que el controlador de esta vista es HelloController.

Para saber lo que hace el botón tenemos que ver el código del controlador, i.e. ***HelloController.java***

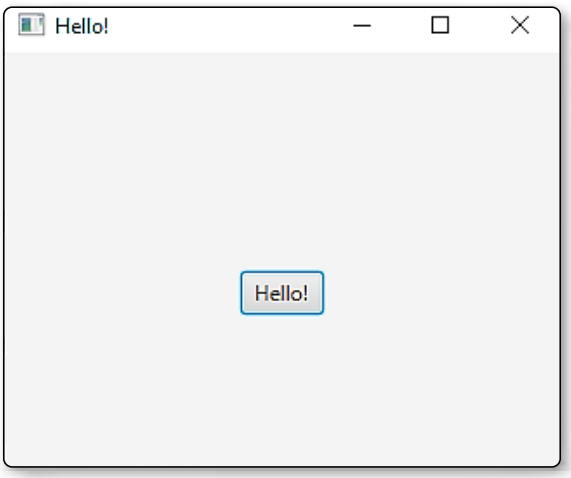
```
package teoria.primeraapp;

import javafx.fxml.FXML;
import javafx.scene.control.Label;

public class HelloController {
    @FXML
    private Label welcomeText;
```

```
    @FXML
    protected void onHelloButtonClick() {
        welcomeText.setText("Welcome to JavaFX Application!");
    }
}
```

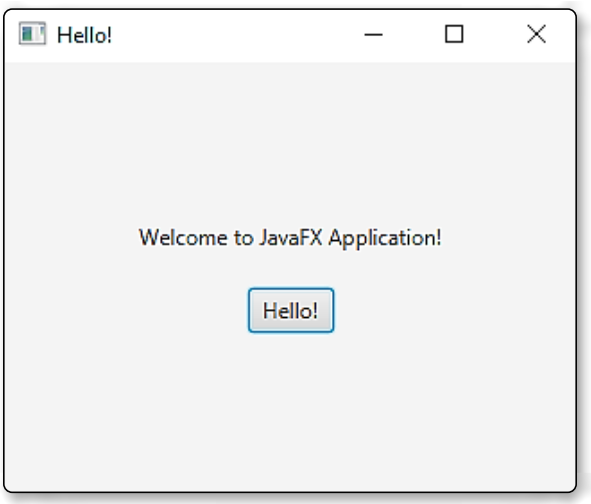
Si ejecutamos la aplicación pulsando el botón verde:

```
0   public class HelloApplication extends Application
```

Nos sale la ejecución del programa:



Y al pulsar el botón Hello!



Sale el mensaje de bienvenida a la apliación JavaFX.

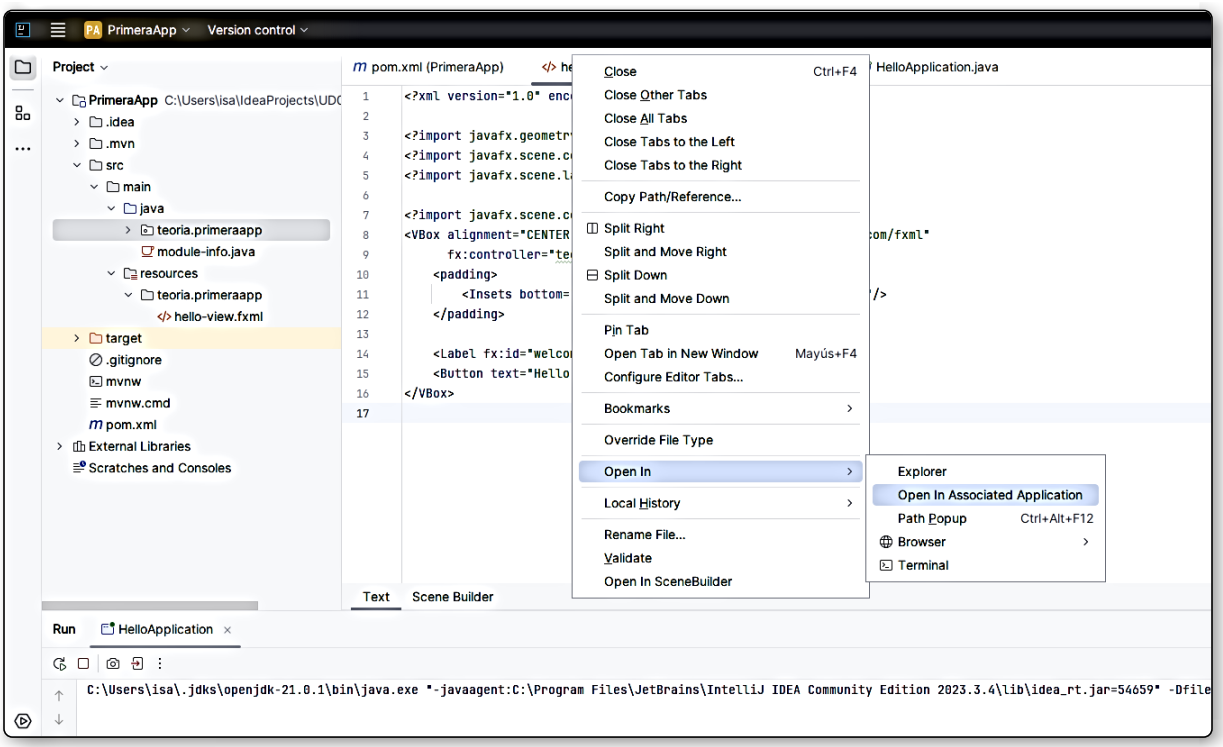
https://docs.oracle.com/javase/8/javafx/api/javafx/scene/Scene.html

9.3 EJECUTAR SCENEBUILDER DESDE INTELLIJ

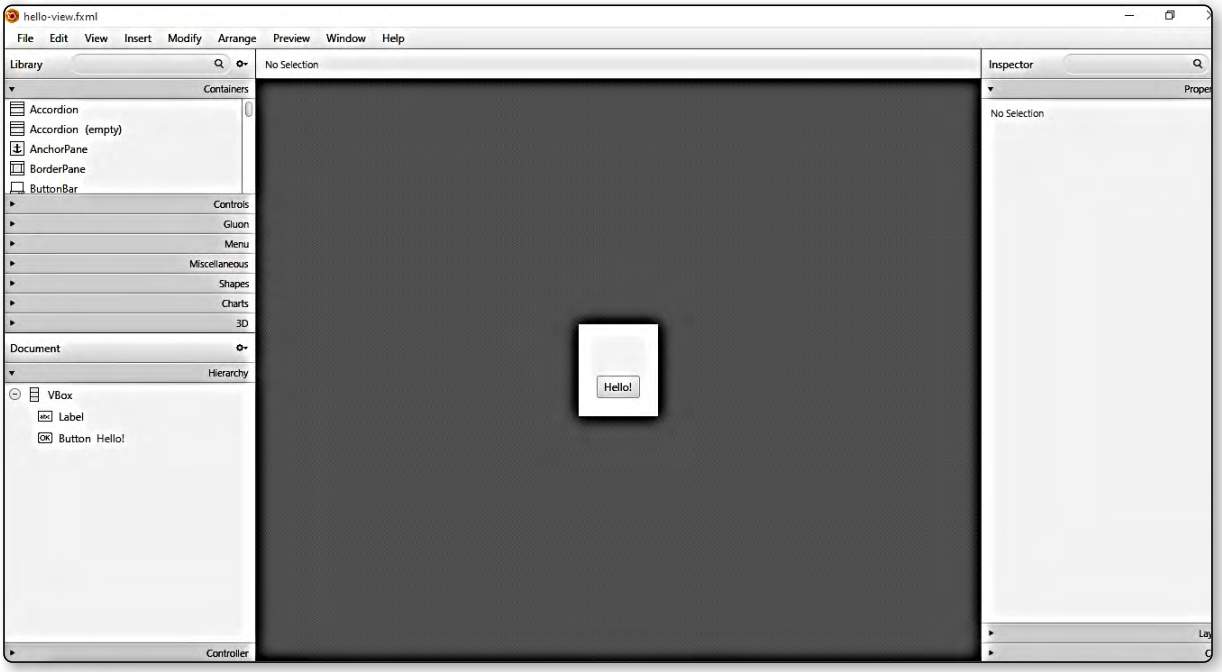
TP3. Como hemos comentado anteriormente en el desarrollo de una aplicación. **Scene Builder** es un editor visual que permite la creación de archivos *FXML* para una UI sin escribir código. Solo arrastrando y soltando elementos podemos dibujar la vista. Podremos realizar un diseño de pantallas WYSIWYG (What You See Is What You Get).

Para abrir el archivo FXML con la herramienta SceneBuilder:

Haz click derecho sobre hello-view.fxml y haz click sobre la pestaña del archivo y selecciona ***Open in. Open in Associated Application*** tal y como se muestra a continuación:

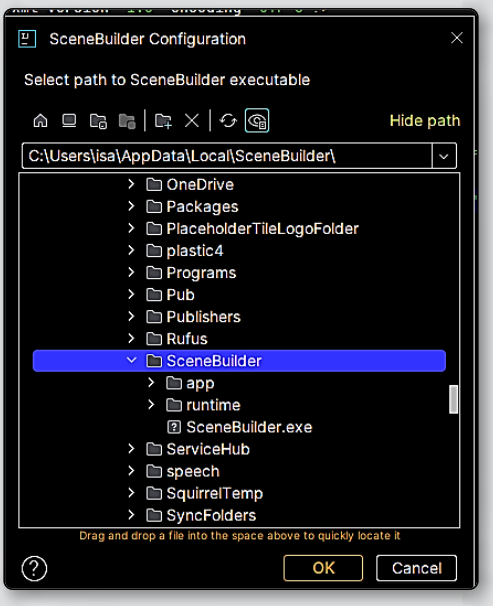


Ahí es cuando se nos abrirá el SceneBuilder y veremos el diseño de la interfaz de usuario correspondiente al archivo ***hello-view.fxml***

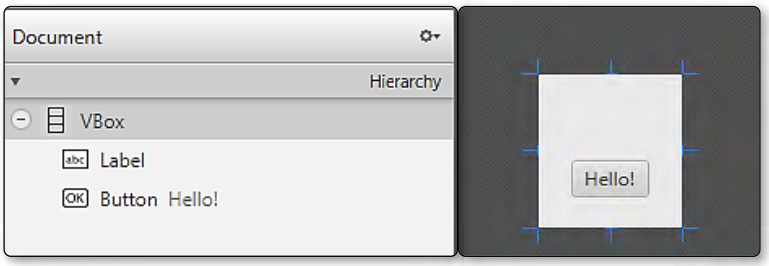


ⓘ NOTA

En caso de que no lo tengas asociado hay que decir dónde está el ejecutable SceneBuilder.exe



El editor de Scene Builder se abrirá con una vista con un contenedor VBox dentro del que ha metido la etiqueta y el botón.

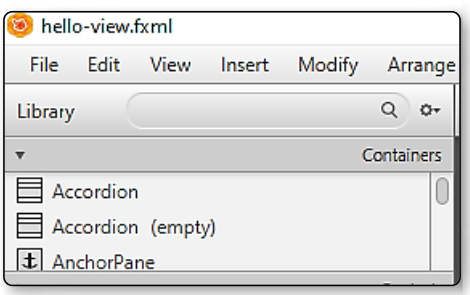


Si miramos el archivo hello-view.fxml podemos editar y cambiar cosas si queremos (aunque es preferible utilizar el Scene Builder).

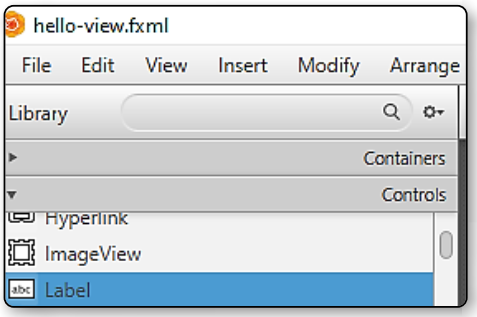
```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.layout.VBox?>

<?import javafx.scene.control.Button?>
<VBox alignment="CENTER"
  spacing="20.0"
  xmlns:fx="http://javafx.com/fxml"
        fx:controller="teoria.primeraapp.HelloController">
    <padding>
        <Insets bottom="20.0" left="20.0" right="20.0" top="20.0"/>
    </padding>

    <Label fx:id="welcomeText"/>
    <Button text="Hello!" onAction="#onHelloButtonClick"/>
</VBox>
```

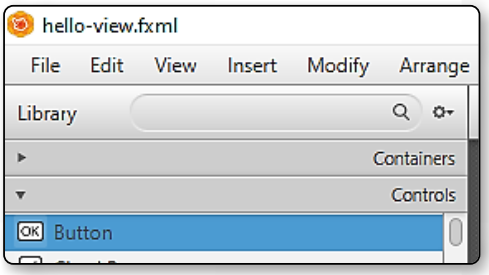
Ahora sigue los pasos siguientes:

1. Borra el VBox por defecto utilizando el Scene Builder.
2. Añade un AnchorPane desde Containers.

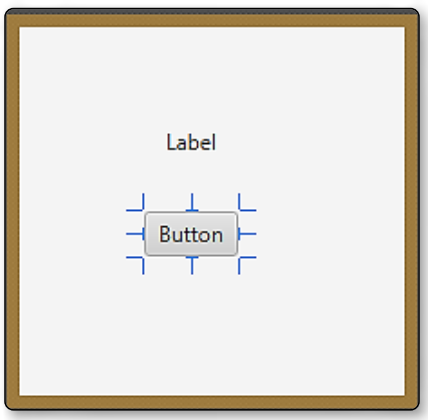


3. Añade un botón y una etiqueta desde Control.

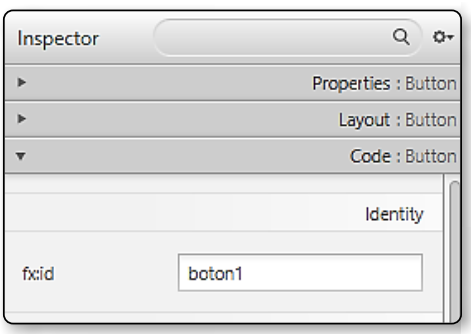


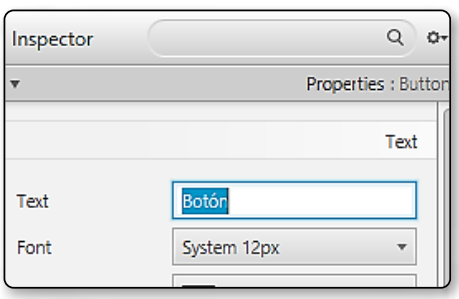


y quedará como en la figura siguiente:

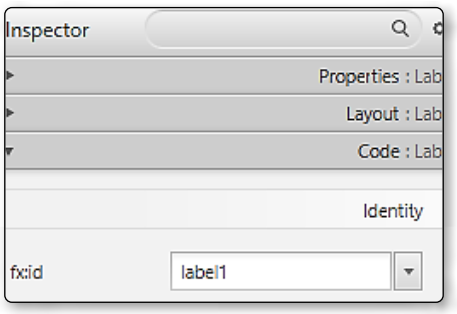


En lugar de teclear código es más sencillo hacerlo con SceneBuilder. Añade un fx:id al botón, en el *Inspector*, desde el desplegable *Code* y dale el nombre que lo identifica *boton1*. Y luego cambia el texto del botón a *Botón.*

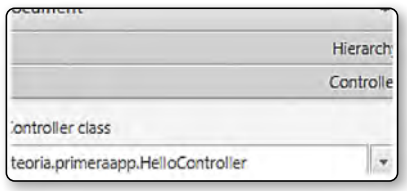




4. A la etiqueta ponle el nombre fx:id *label1*.



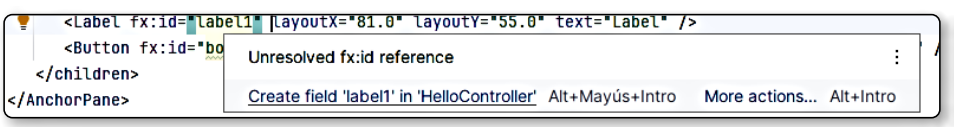
5. Selecciona el AnchorPane y define su clase controlador: *HelloController.*



6. Ahora guardamos y comprobamos el código del archivo *hello-view.fxml:*

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.layout.AnchorPane?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity"
      minHeight="-Infinity" minWidth="-Infinity"
      prefHeight="204.0"
      prefWidth="213.0" xmlns="http://javafx.com/javafx/21"
      xmlns:fx="http://javafx.com/fxml/1"
      fx:controller="teoria.primeraapp.HelloController">
   <children>
      <Label fx:id="label1" layoutX="81.0" layoutY="55.0" text="Label" />
      <Button fx:id="boton1" layoutX="69.0" layoutY="102.0" mnemonicParsing="false"
text="Button" />
   </children>
</AnchorPane>
```

7. Haz click en los id de los nombres y selecciona ***Create a field in HelloController*** para cada fx:id. Estamos asociando los elementos de la vista al controlador.



Realiza la misma operación en el botón.

8. Comprueba que se han creado los campos en *Controller.java.*

```
package teoria.primeraapp;

import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;

public class HelloController {

    public Label label1;
    public Button boton1;
}
```

Cambia el modificador de acceso a private:

```
import javafx.scene.control.Button;
import javafx.scene.control.Label;

public class Controller {
    private Label label1;
    private Button boton1;
}
```

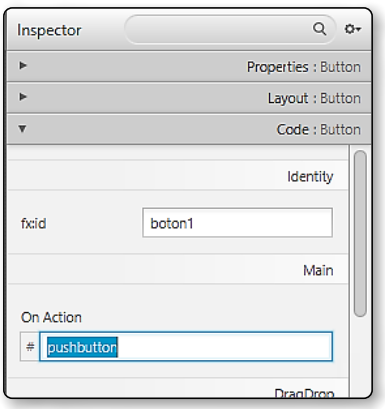
Y ahora pon la anotación tanto para el boton1 como para label1. Si no realizamos este paso nos dará error de ejecución. Este paso se realiza desde el hello-view.xml

Elegimos Annotate field “label1” as @FXML, y lo mismo con boton1.

El archivo controlador queda así:

```
pom.xml (PrimeraApp)   hello-view.fxml   HelloController.java
2
3   import javafx.fxml.FXML;
4   import javafx.scene.control.Button;
5   import javafx.scene.control.Label;
6
7   public class HelloController {
8
9
10      @FXML
11      private Label label1;
12      @FXML
13      private Button boton1;
14  }
```

9. Desde Snece Builder añade un método *pusbutton*, que realice código cuando se pulse el botón.

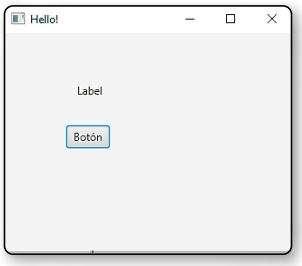


10. Crea el método en el controlador acercando el ratón a #pushbutton. Pulsa el enlace *Create method pushbutton:*

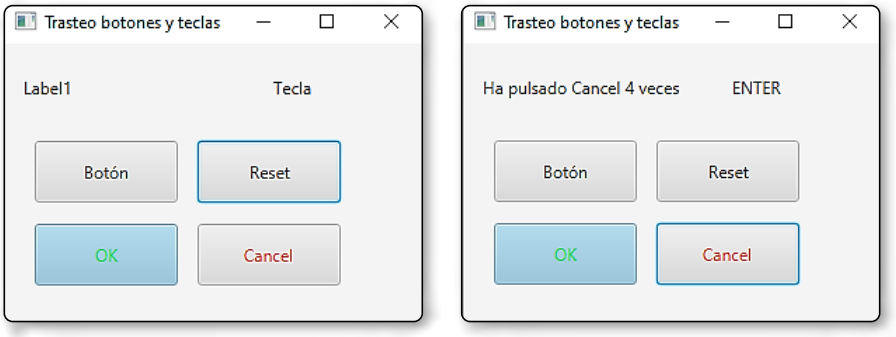
11. Ahora añade el código para el comportamiento del botón.

```
pom.xml (PrimeraApp)   hello-view.fxml   HelloController.java
 6  import javafx.scene.control.Label;
 7
 8  public class HelloController {
 9      @FXML
10      private Label label1;
11      @FXML
12      private Button boton1;
13
14      public void pushbutton(ActionEvent actionEvent) {
15          label1.setText("Ha pulsado el botón");
16      }
17  }
```

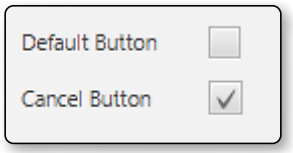
12. Ahora ejecutamos la aplicación y pulsamos el botón.



TP4. ACTIVIDAD 2: modifica el programa anterior para que tenga el aspecto siguiente:



En las propiedades del botón Cancel activa, *Cancel Button*. Y En el botón Ok selecciona Defau*lt Button*.



Si se pulsa el Botón en la etiqueta debe decir “Ha pulsado el botón”. En cambio, si se pulsa cualquiera de los otros dos botones, contaremos las veces que se pulsa cada uno de ellos: al pulsar el botón Ok, la etiqueta dirá “Ha pulsado OK x veces”; y si pulsamos Cancel dirá “Ha pulsado Cancel x veces”.

Además si pulsamo la tecla ENTER, o la tecla ESCAPE la etiqueta label2 lo indicará. Cuando pulse ESCAPE contará como si hubiera pulsado el botón Cancel y por lo tanto sumará 1 al contador. Al pulsar Reset se pone todo como al principio y los contadores a cero.

Pistas:

A continuación pongo un breve esquema de los eventos que voy a controlar según el objeto

Objeto (fx:id)	Evento	Método asignado	
boton1	On Action	pusbutton	
bOK	On Action	pusbutton	
bCancel	On Key Pressed	teclaPulsada	
breset			

```
 //codigos de las teclas
https://docs.oracle.com/javafx/2/api/javafx/scene/input/KeyCode.html
```

KeyCode.***ENTER***
KeyCode.***ESCAPE***

Solución:

Hello-view.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.layout.AnchorPane?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-
Infinity" prefHeight="182.0" prefWidth="325.0" xmlns="http://javafx.com/javafx/21"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="teoria.primeraapp.HelloController">
   <children>
      <Label fx:id="label1" layoutX="26.0" layoutY="21.0" text="Label" />
      <Button fx:id="boton1" layoutX="83.0" layoutY="74.0"
```

```
            mnemonicParsing="false"
            onAction="#pushbutton"
            text="Botón" />
        <Label fx:id="label2" layoutX="222.0" layoutY="21.0" text="Tecla" />
        <Button fx:id="bReset" layoutX="148.0" layoutY="74.0"
            mnemonicParsing="false"
            onAction="#pushbutton"
            onKeyPressed="#teclaPulsada"
            text="Reset" />
        <Button fx:id="bOK" layoutX="83.0" layoutY="114.0"
            mnemonicParsing="false"
            onAction="#pushbutton"
            onKeyPressed="#teclaPulsada"
            prefHeight="25.0" prefWidth="45.0" text="OK" />
        <Button fx:id="bCancel" cancelButton="true" layoutX="148.0" layoutY="114.0"
    mnemonicParsing="false" onAction="#pushbutton" onKeyPressed="#teclaPulsada"
    text="Cancel" />
       </children>
    </AnchorPane>
```

HelloAplication.java

```
    package teoria.primeraapp;

    import javafx.application.Application;
    import javafx.fxml.FXMLLoader;
    import javafx.scene.Scene;
    import javafx.stage.Stage;

    import java.io.IOException;

    public class HelloApplication extends Application {
        @Override
        public void start(Stage stage) throws IOException {
            FXMLLoader fxmlLoader = new FXMLLoader(HelloApplication.class.
    getResource("hello-view.fxml"));
            Scene scene = new Scene(fxmlLoader.load(), 320, 240);
            stage.setTitle("Trasteo botones y teclas");
            stage.setScene(scene);
            stage.show();
        }

        public static void main(String[] args) {
            launch();
        }
    }
```

HelloController.java

```
package teoria.primeraapp;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.input.KeyCode;
import javafx.scene.input.KeyEvent;

public class HelloController {
    @FXML
    private Label label2;
    @FXML
    private Button bReset;
    @FXML
    private Button bOK;
    @FXML
    private Button bCancel;
    @FXML
    private Label label1;
    @FXML
    private Button boton1;
    int contadorOK = 0;
    int contadorCancel = 0;


    public void pushbutton(ActionEvent actionEvent) {
        label1.setText("Ha pulsado el botón");
       //compruebo qué objeto botón ha sido pulsado
       //ok o cancel para contarlo
        Object obj = actionEvent.getSource();
        if (obj == bOK){
            contadorOK ++;
            label1.setText("Ha pulsado OK " + contadorOK + " veces");
        }
        if (obj == bCancel){
            contadorCancel ++;
            label1.setText("Ha pulsado Cancel " + contadorCancel + " veces");
        }
        if (obj == bReset){
            label1.setText("Label1");
            label2.setText("Tecla");
            contadorOK = 0;
            contadorCancel = 0;
        }
    }
    //codigos de las teclas
//https://docs.oracle.com/javafx/2/api/javafx/scene/input/KeyCode.html
```

```
    public void teclaPulsada(KeyEvent ke) {
        KeyCode key = ke.getCode();
        if (key == KeyCode.ENTER)
            label2.setText("ENTER");
        if (key == KeyCode.ESCAPE)
            label2.setText("ESCAPE");
    }
}
```

9.4 CONTROLES BÁSICOS JAVAFX

T5. Algunos ya los hemos utilizado en los ejemplos anteriores. A continuación veamos algunos más:

9.4.1 Label y Button

Una etiqueta o label se usa para mostrar un texto en la escena. Una vez ponemos la etiqueta en el layout, podemos usar los métodos ***getText*** o ***setText*** de la etiqueta.

```
@FXML private Label label
```

Como ya hemos visto en los ejemplos si queremos usar un botón, lo creamos con su texto (hay otros constructores parecidos a los de la clase Label).

```
@FXML private Button boton1;
```

9.4.2 TextField y TextArea

En relación con los campos de texto, los más comunes son los *TextFields,* utilizados para entradas de textos cortos, con una línea simple, y los *TextAreas,* utilizados para textos más largos, con varias filas y columnas.

```
@FXML  private TextField texto;
```

Hay algunos métodos tales como ***getText*** o ***setText*** para obtener y establecer el texto del control respectivamente. También existen métodos como ***setPromptText*** para establecer que el texto se borrará cuando el usuario comience a escribir algo en el campo de texto, o ***setPrefColumnCount*** para establecer el número de caracteres que será visible en el mismo.

Si necesitamos utilizar un ***TextArea***, lo creamos con un constructor vacío o con un texto inicial, y disponemos de un par de métodos para establecer el número de filas y columnas inicial.

Cuando utilizamos un área de texto, definiremos un número de filas y columnas (caracteres por fila) en el archivo FXML. Hay algún otro método que podría resultar útil, como ***setWrapText*** (establece si se deben añadir las líneas que exceden el área de texto) o ***getText/setText***, como en la clase *TextField*.

9.4.3 RadioButton

Normalmente los RadioButton se incluyen dentro de un grupo donde sólo uno de ellos es seleccionado a la vez. Para ello necesitamos definir un ToogleGroup, y añadirle los botones.

Para definir el grupo en la propiedad *ToogleGroup* de cada radio botón, le ponemos el mismo nombre, por ejemplo grupoRB

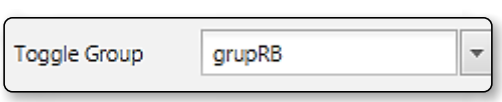
Y para preseleccionar uno de los RadioButton activamos la propiedad *Selected:*

Veamos un ejemplo:

TP6. ACTIVIDAD 3: elige el color. Seleccionar el color a través de un radiobutton y al pulsar el botón decir si es Rojo, Amarillo o Verde a través de un mensaje en una etiqueta.

Crea un proyecto que tenga el siguiente aspecto

- Inicialmente el valor por defecto seleccionado será el Rojo. Para ello desde Scene Builder estableceremos la propiedad S*elected.*
- En las propiedades de cada radioButton, indicaremos que pertenecen a un grupo



- Al pulsar el botón se escribirá en la etiqueta “Has seleccionado: colorDelRadiobutton”.

NOTA

Está resuelto como R1 al final de la unidad y también hay disponible un vídeo-tutorial realizado por la profesora.

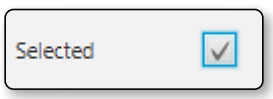
T7.

9.4.4 Checkbox

Una checkbox es un control que podemos seleccionar y deseleccionar alternativamente cada vez que hacemos click en el interior del recuadro.

```
@FXML private CheckBox chkAceptarCondiciones;
```

Para que esté seleccionado por defecto ponemos su propiedad *Selected* a true



Podemos observar las propiedades en el sample.fxml:

```
<CheckBox fx:id="chkAceptarCondiciones" layoutX="320.0" layoutY="174.0"
mnemonicParsing="false" onMouseClicked="#clickAceptaCondiciones" selected="true"
text="Aceptar condiciones" />
```

Podemos utilizar el método *setSelected* para seleccionarlo y el método *isSelected* para saber si el check está seleccionado o no.

```
public void clickAceptaCondiciones(MouseEvent mouseEvent) {
    if (chkAceptarCondiciones.isSelected())
        label2.setText("Acepta Condiciones");
    else
        label2.setText("NO Acepta Condiciones");
}
```

9.4.5 Listas

Hay dos tipos ***principales*** de listas que podemos utilizar en cualquier aplicación ***ListView***, con un tipo fijo de tamaño, donde se muestran algunos elementos y podemos deslizar la lista para buscar los elementos que necesitemos. También tenemos ***ChoiceBox*** o ***ComboBox*** para trabajar con listas desplegables, donde solo se muestra un elemento, y podemos seleccionar otro elemento extendiendo la lista. De todos modos, generalmente utilizado una *ObservableList* de elementos para añadir elementos a este tipo de listas. También disponemos de algunos métodos útiles para coger el elemento seleccionado o su índice. Si tenemos una variable ListView llamada lista, así es como se añadirían elementos a la lista, y luego comprobar el elemento que está seleccionado actualmente:

```
<ListView fx:id="listaListView" layoutX="29.0" layoutY="191.0" prefHeight="100.0"
prefWidth="200.0" />
```

```
@Override
public void initialize(URL url, ResourceBundle resourceBundle) {
    listaListView.setItems(
            FXCollections.observableArrayList( "Windows", "Linux", "Mac OS
X","Ubuntu"));
    //listaListView.getSelectionModel().setSelectionMode(SelectionMode.MULTIPLE);
    listaListView.getSelectionModel().setSelectionMode(SelectionMode.SINGLE);
}
```

Se puede definir el modo de selección de la lista, es decir, si queremos que se seleccionen múltiples elementos, o queremos uno solo. Cambiamos esta característica con los métodos:

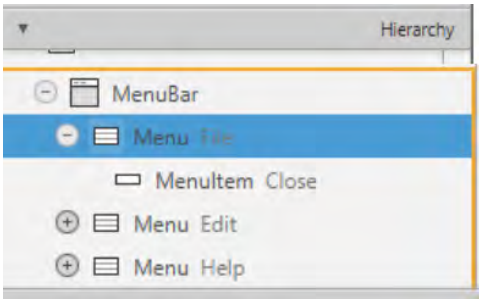
```
listaListView.getSelectionModel().setSelectionMode(SelectionMode.MULTIPLE);
listaListView.getSelectionModel().setSelectionMode(SelectionMode.SINGLE);
```

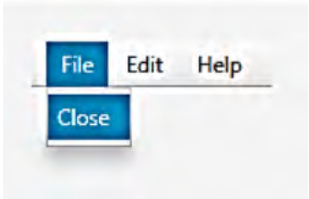
Si queremos coger los ítems seleccionados de la lista, tenemos que coger el modelo de selección de la lista y llamar a ***getSelectedItem*** (para listas de selección simples) o bien ***getSelectedItems*** (para listas de selección múltiple). También disponemos de los métodos ***getSelectedIndex*** (para listas de selección simples) o bien ***getSelectedIndexes*** (para listas de selección múltiple).

Si utilizamos una lista desplegable (como ***ChoiceBox*** o ***ComboBox***), podemos utilizar el método ***getValue*** para coger el valor actual seleccionado en lugar de llamar a *getSelectionModel().getSelectedItem(),* lo que es más apropiado para listas fijas.

9.4.6 Menus

Como en cualquier aplicación de escritorio, podemos añadir un menú a nuestra aplicación JavaFx (no es usual cuando estamos desarrollando aplicaciones para móvil). El patrón que normalmente seguimos es poner una barra de menús (con unos menús por defecto dentro, que podemos editar), definimos las categorías (Menu) y añadimos los elementos del menú a las categorías





Como podemos ver en la jerarquía, necesitamos utilizar tres elementos diferentes:

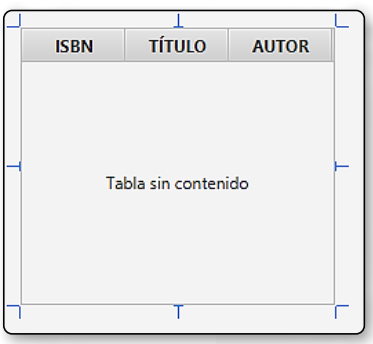
- ***MenuBar*** para definir la barra donde se colocarán todos los menús .
- ***Menu*** para definir las categorías de los ítems de menú. Una categoría es un elemento que se muestra, pero que no tiene acciones asociadas (si hacemos clic en él, no ocurre nada salvo que se muestren las categorías que tiene).
- ***MenuItem*** para definir cada ítem de nuestro menú. Si hacemos clic sobre un ítem, podemos definir código asociado a dicha acción.
  - También hay algunos subtipos MenuItem, tales como ***CheckMenuItem*** (ítems que pueden seleccionarse/deseleccionados, como las checboxes), o ***RadioMenuItem*** (grupos de ítems donde solo podemos seleccionar uno de ellos, como los radio botones). También podemos utilizar un ***SeparatorMenuItem*** para crear una separación entre grupos de elementos de menú.

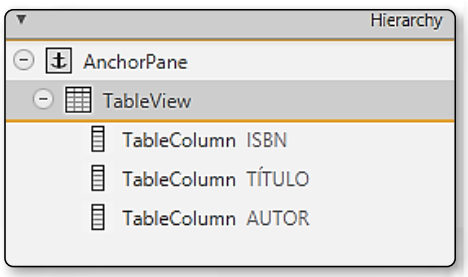
9.4.7 Tables

Hay un control muy útil que podemos utilizar cuando trabajamos con grandes cantidades de información, o con algunos datos estructurados que necesitamos mantener visibles a la vez: tables. Este control nos permite organizar la información, así que podemos mostrar diferentes registros en distintas columnas, e información diferente sobre el mismo registro en distintas columnas.

Para tratar con las tables, utilizamos el control ***TableView.*** Podemos definir las cabeceras de las columnas con la clase ***TableColumn.***

TP8. ACTIVIDAD 5. Por ejemplo, si queremos poner una lista de libros con su ISBN, título, autor y, crearemos una tabla como la siguiente:





Una vez tenemos creada nuestra tabla, necesitamos llenarla con algún dato que hayamos almacenado previamente. Es recomendable crear una public class que almacene cada objeto de la tabla (cada fila), y asocie cada columna a un atributo de esa clase. En nuestro ejemplo podemos crear una clase Libro con tres atributos: ISBN, TITULO y AUTOR con sus correspondientes getters y setters:

Libro.java

```
package sample;

public class Libro {
    String isbn;
    String titulo;
    String autor;

    //constructor
    public Libro(String isbn, String titulo, String autor) {
        this.isbn=isbn;
        this.titulo=titulo;
        this.autor=autor;
    }

    //getters y setters
    public String getIsbn() {
        return isbn;
    }

    public void setIsbn(String isbn) {
        this.isbn=isbn;
    }

    public String getTitulo() {
        return titulo;
    }

    public void setTitulo(String titulo) {
        this.titulo=titulo;
    }

    public String getAutor() {
        return autor;
    }

    public void setAutor(String autor) {
        this.autor=autor;
    }
}
```

Ahora debemos enlazar en el controlador la vista de la tabla y sus columnas. Date cuenta que definimos el tipo de los ítems que tendrá la tabla (Libro), y para cada columna, especificamos también el tipo de dato que vamos a mostrar (String, Integer,…)

```
@FXML private TableView<Libro> tabla;
@FXML private TableColumn <Libro, String> colisbn;
@FXML private TableColumn <Libro, String> colautor;
@FXML private TableColumn <Libro, String> coltitulo;
ObservableList<Libro> datos;
```

Y asociamos cada propiedad de Libro con una columna de la tabla:

```
@Override
public void initialize(URL url, ResourceBundle rb) {
    //texto a mostrar cuando la tabla está vacía
    tabla.setPlaceholder(new Label("No hay items para mostrar..."));

    //También asociamos cada propiedad de libro a un objeto TableColumn
    colisbn.setCellValueFactory(new PropertyValueFactory("isbn"));
    coltitulo.setCellValueFactory(new PropertyValueFactory("titulo"));
    colautor.setCellValueFactory(new PropertyValueFactory("autor"));
}
```

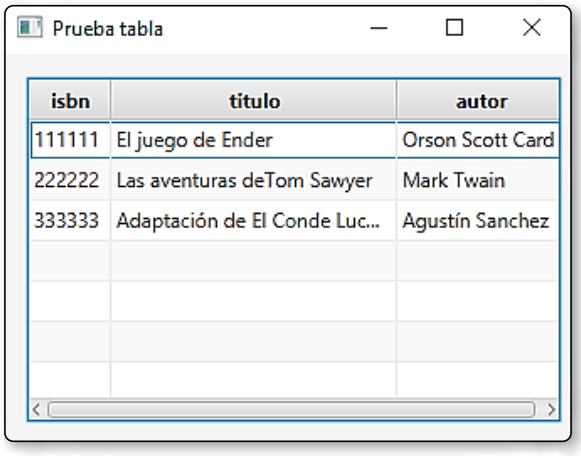
NOTA

Es **importante** que utilicemos el mismo nombre de parámetro en `PropertyValueFactory` correspondiente al getter pero sin el prefijo "get". Por ejemplo si tenemos un getter que se llama getPrecioTotal, debemos utilizar el nombre "PrecioTotal" en la columna.

Una vez hecho lo anterior, podemos crear o leer colecciones de libros con nuestra clase *FXcollections:*

```
datos = FXCollections.observableArrayList(
new Libro("111111", "El juego de Ender", "Orson Scott Card"),
new Libro("222222", "Las aventuras de Tom Sawyer", "Mark Twain"),
new Libro("333333", "Adaptación de El Conde Lucanor", "Agustín Sanchez")
);
tabla.setItems(datos);
```

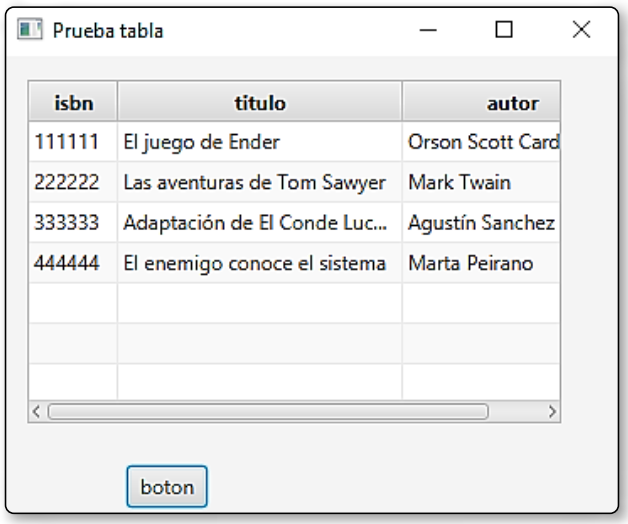
Y veremos algo así:



Añadir filas a una *TableView* es tan sencillo como añadir objetos nuevos a la lista asociada a la misma. En nuestro ejemplo si añadimos o borramos datos de nuestra lista observable (variable datos), la *TableView* se actualizará automáticamente.

Podemos utilizar el método ***add*** de la interfaz *ObservableList* para añadir elementos a la lista. Por ejemplo cuando pulsamos un botón.

```
public void accionBoton1(ActionEvent actionEvent) {
    datos.add(
       new Libro("444444", "El enemigo conoce el sistema", "Marta Peirano")
    );
}
```



También podemos utilizar el método ***remove*** para borrar un elemento (o un conjunto de ellos) desde una posición (o rango) dados. Esta línea borra el último elemento de la lista/tabla:

```
datos.remove(datos.size()-1);
```

Para saber qué fila de la tabla está seleccionada actualmente, podemos utilizar los mismos métodos que habíamos visto en el control *ListView*. Típicamente, cogemos el índice del elemento que está seleccionado y cogemos el objeto del arraylist:

```
int index = tabla.getSelectionModel().getSelectedIndex();
Libro selLibro = datos.get(index);

Por lo que el código completo de Controller.java es:
package sample;


import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import java.net.URL;
import java.util.ResourceBundle;
import javafx.scene.control.cell.PropertyValueFactory;

public class Controller implements Initializable{

    @FXML private Button boton1;
    @FXML private TableView<Libro> tabla;
    @FXML private TableColumn <Libro, String> colisbn;
    @FXML private TableColumn <Libro, String> colautor;
    @FXML private TableColumn <Libro, String> coltitulo;
    ObservableList<Libro> datos;

    @Override
    public void initialize(URL url, ResourceBundle rb) {
        //texto a mostrar cuando la tabla está vacía
        tabla.setPlaceholder(new Label("No hay items para mostrar..."));

        //También asociamos cada propiedad de libro a un objeto TableColumn
        colisbn.setCellValueFactory(new PropertyValueFactory("isbn"));
        coltitulo.setCellValueFactory(new PropertyValueFactory("titulo"));
        colautor.setCellValueFactory(new PropertyValueFactory("autor"));

        /*ObservableList<Libro> */
        datos = FXCollections.observableArrayList(
          new Libro("111111", "El juego de Ender", "Orson Scott Card"),
```

```
            new Libro("222222", "Las aventuras de Tom Sawyer", "Mark Twain"),
            new Libro("333333", "Adaptación de El Conde Lucanor", "Agustín Sanchez"));
        tabla.setItems(datos);

    }

    public void accionBoton1(ActionEvent actionEvent) {
        datos.add(new Libro("444444", "El enemigo conoce el sistema", "Marta Peirano"));
        //datos.remove(datos.size()-1);
        //int index = tabla.getSelectionModel().getSelectedIndex();
        //Libro selLibro = datos.get(index);
    }
}
```

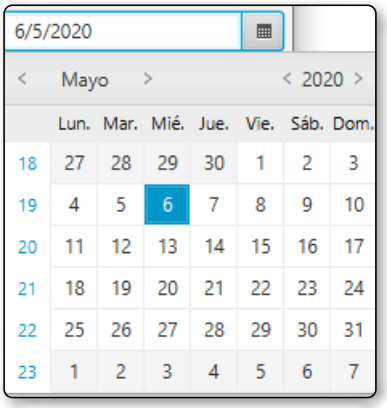
Sample.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.layout.AnchorPane?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-
Infinity" prefHeight="326.0" prefWidth="439.0" xmlns="http://javafx.com/javafx/11.0.1"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.Controller">
   <children>
      <TableView fx:id="tabla" layoutX="14.0" layoutY="14.0"
         prefHeight="204.0" prefWidth="317.0">
        <columns>
          <TableColumn fx:id="colisbn" prefWidth="75.0" text="isbn" />
          <TableColumn fx:id="coltitulo" prefWidth="100.0" text="titulo" />
           <TableColumn fx:id="colautor" prefWidth="100.0" text="autor" />
        </columns>
      </TableView>
      <Button fx:id="boton1" layoutX="73.0" layoutY="243.0"
        mnemonicParsing="false" onAction="#accionBoton1" text="boton" />
   </children>
</AnchorPane>
```

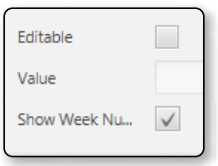
9.4.8 DatePicker

T9.

El control selector de fecha o *DatePicker*, permite al usuario seleccionar una fecha de un calendario emergente que aparece cuando pulsas su botón derecho. Podemos acceder al TextFieled dentro del control DatePicker utilizando el método getEditor(). Una vez hemos accedido a ese campo de texto, podemos coger su valor fácilmente. También podemos coger el objeto *LocalDate* que representa la fecha actual utilizando *getValue()*.




```
@FXML private DatePicker datepicker;


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

    //inicializo el datepicker
    datepicker.setShowWeekNumbers(true); // mostrar los días de la semana
    datepicker.setValue(LocalDate.now()); // Seleccionar hoy por defecto
    datepicker.setEditable(false); // El usuario no puede modificar el campo de texto
}
```



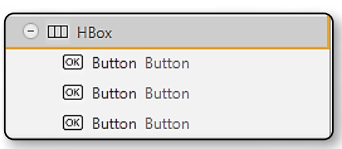
9.5 CONTENEDORES JAVAFX

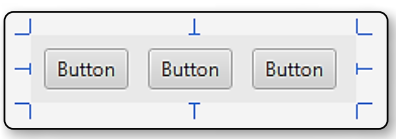
T10. Cada control que podemos colocar en una aplicación, como botones, etiquetas etcétera, tiene que colocarse dentro de un contenedor, también conocido como *layout managers*. Estos componentes nos permiten colocar los controles de una manera determinada, de forma que no necesitamos situar dichos componentes manualmente.

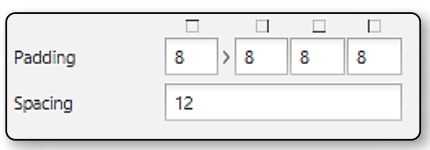
Algunos de los contenedores más comunes en JavaFX son:

- **HBox:** coloca los controles horizontalmente, una al lado del otro.
- **VBox:** coloca los controles verticalmente, uno encima/debajo del otro.

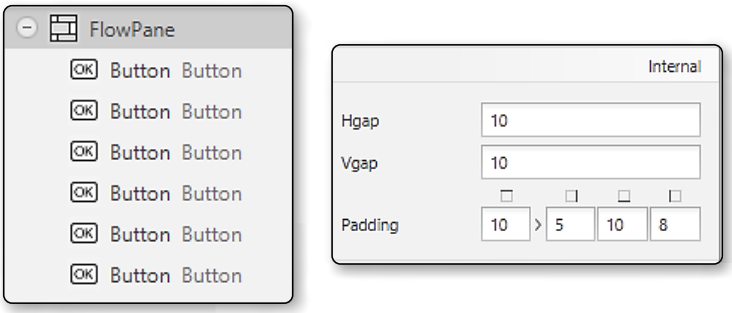
  Estos dos contenedores tienen algunas propiedades interesante en la pestaña *Properties*, tal como *Spacing* para separar automáticamente cada control del resto.



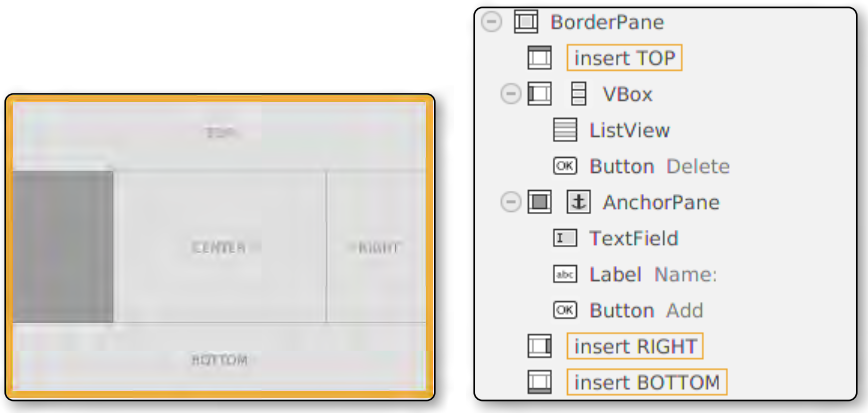


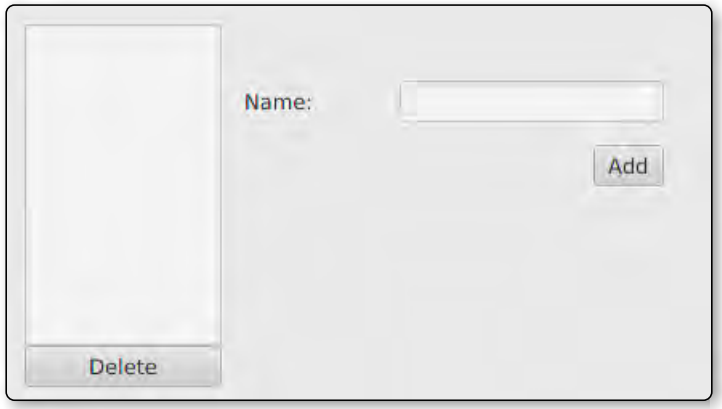


- **FlowPane:** coloca los controles uno al lado del otro hasta que no hay más espacio (vertical o horizontalmente). Entonces, va a la fila (o columna, dependiendo de su configuración), para continuar colocando más controles en un FlowPane, podemos controlar el espacio horizontal entre elementos (Hgap) y vertical (Vgap).

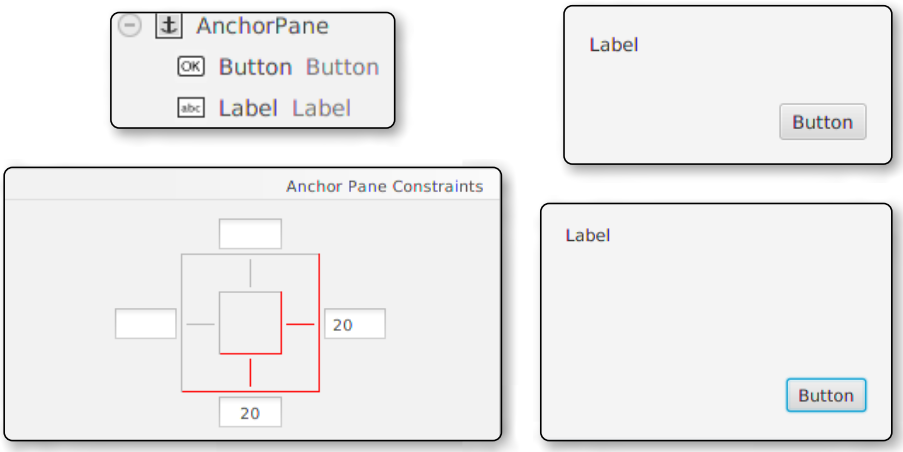


- **BorderPane**: este diseño divide el panel en cinco regiones: top, bottom, left, right y center; y podemos añadir un control o un contenedor con varios controles en cada región.





- **AnchorPane**: este es el más flexible ya que nos permite poner los elementos en cualquier lugar. Podemos indicar una distancia fija de un elemento a otro (o más) de los bordes del AnchorPane (top, right, bottom, left), así que cuando la ventana (y el AnchorPane contenida en ella) crece, los elementos siempre mantienen la misma distancia a los límites del contenedor.



Hay otros paneles contenedores, como *GridPane* (crea una clase de tabla en el panel para distribuir los controles), *TilePane* (similar a FlowPane, pero dejando el mismo espacio para cada control, y alguno más.

9.6 EJERCICIOS RESUELTOS

- **R1. Solución ACTIVIDAD 3 elige el color.** Seleccionar el color a través de un radioButton y al pulsar el botón decir si es Rojo, Amarillo o Verde a través de un mensaje en una etiqueta.

Main.java

```
package sample;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        Parent root = FXMLLoader.load(getClass().getResource("sample.fxml"));
        primaryStage.setTitle("Prueba Controles");
        primaryStage.setScene(new Scene(root, 463, 295));
        primaryStage.show();
```

```
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

module-info.java

```
module ProbarControles {
    requires javafx.fxml;
    requires javafx.controls;
    opens sample;
}
```

Controller.java

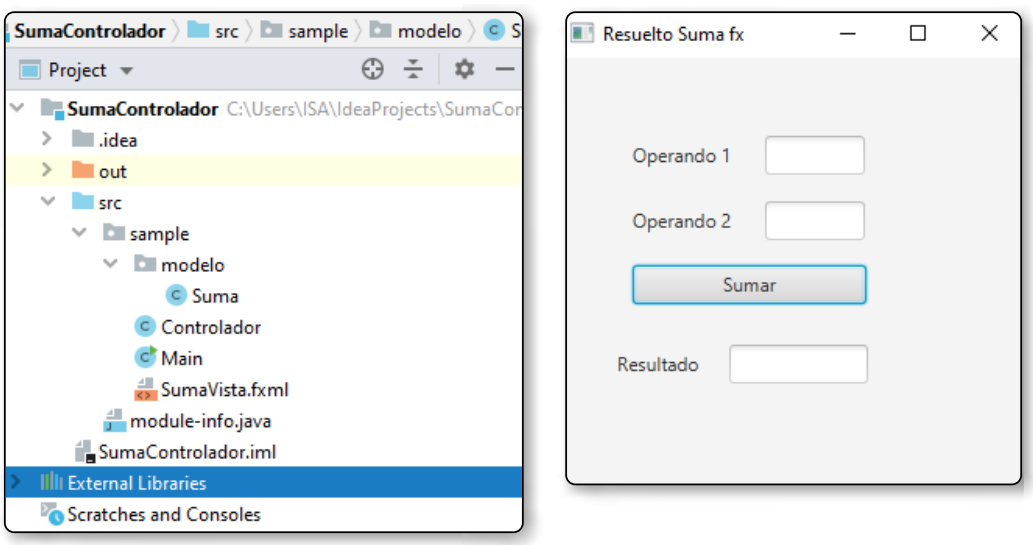
```
public class Controller {

    @FXML private RadioButton rbRojo;
    @FXML private ToggleGroup grupRB;
    @FXML private RadioButton rbAmarillo;
    @FXML private RadioButton rbVerde;
    @FXML private Label label1;
    @FXML private Button btnPulsa;

    public void pulsaBoton(ActionEvent actionEvent) {
        RadioButton selected = (RadioButton)grupRB.getSelectedToggle();
        label1.setText("Has seleccionado: " + selected.getText());
    }
}
```

sample.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.RadioButton?>
<?import javafx.scene.control.ToggleGroup?>
<?import javafx.scene.layout.AnchorPane?>


<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"
            minWidth="-Infinity" prefHeight="295.0" prefWidth="463.0"
            xmlns="http://javafx.com/javafx/11.0.1"
```

```
            xmlns:fx="http://javafx.com/fxml/1"
            fx:controller="sample.Controller">
   <children>
      <Button fx:id="btnPulsa" layoutX="192.0" layoutY="133.0"
mnemonicParsing="false" onAction="#pulsaBoton" text="Pulsa" />
      <Label fx:id="label1" layoutX="175.0" layoutY="185.0" text="Label" />
      <RadioButton fx:id="rbRojo" layoutX="77.0" layoutY="77.0"
mnemonicParsing="false" selected="true" text="Rojo">
         <toggleGroup>
            <ToggleGroup fx:id="grupRB" />
         </toggleGroup>
      </RadioButton>
      <RadioButton fx:id="rbAmarillo" layoutX="175.0" layoutY="77.0"
mnemonicParsing="false" text="Amarillo" toggleGroup="$grupRB" />
      <RadioButton fx:id="rbVerde" layoutX="288.0" layoutY="77.0"
mnemonicParsing="false" text="Verde" toggleGroup="$grupRB" />
   </children>
</AnchorPane>
```

- **R2. ACTIVIDAD 4 SUMADOR** creamos con JavaFX el proyecto con la siguiente estructura y pantalla:



Si nos fijamos he cambiado el nombre que sale por defecto *Controller* por *Controlador*, y *sample.fxml* por *SumaVista.fxml*, recordemos que es mediante la opción *Refactor-> Rename*.

En este ejemplo vamos a ver cómo añadir cambios a la clase controladora. Necesitamos añadir algunos cambios manuales al controlador antes de que se enlace al fichero FXML. Para ser más precisos, necesitamos implementar la interfaz ***Initalizable***, asi que se añadirá el método ***initialize***. Este método se llama automáticamente cuando se muestra la aplicación, al principio, así que todo el código relacionado para controlar la inicialización, carga de ficheros y demás, se iniciará desde aquí.

```
import javafx.fxml.Initializable;
import java.net.URL;
import java.util.ResourceBundle;

@Override
    public void initialize(URL url, ResourceBundle rb) {
        // TODO
    }
```

Lo primero que haremos es dibujar los controles, crear los campos y poner las referencias @FXML; así nos quedará el archivo de la interfaz:

SumaVista.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.AnchorPane?>

<AnchorPane prefHeight="256.0" prefWidth="253.0"
            xmlns="http://javafx.com/javafx/11.0.1"
            xmlns:fx="http://javafx.com/fxml/1"
            fx:controller="sample.Controlador">
    <children>
        <Button fx:id="btnSumar" layoutX="48.0" layoutY="133.0"
                mnemonicParsing="false" onAction="#sumar"
                prefHeight="25.0" prefWidth="150.0" text="Sumar"/>

        <TextField fx:id="txtOp1" layoutX="133.0" layoutY="50.0"
                   prefHeight="25.0" prefWidth="64.0"/>
        <TextField fx:id="txtOp2" layoutX="133.0" layoutY="92.0"
                   prefHeight="25.0" prefWidth="64.0"/>
        <TextField fx:id="txtResultado" editable="false" layoutX="110.0"
                   layoutY="184.0" prefHeight="25.0" prefWidth="89.0"/>

        <Label layoutX="48.0" layoutY="54.0" text="Operando 1"/>
        <Label layoutX="48.0" layoutY="96.0" text="Operando 2"/>
        <Label layoutX="38.0" layoutY="188.0" text="Resultado"/>
    </children>
</AnchorPane>
```

El programa principal se nos genera automáticamente y cambiamos el título por “Resuelto suma fx”.

Main.java

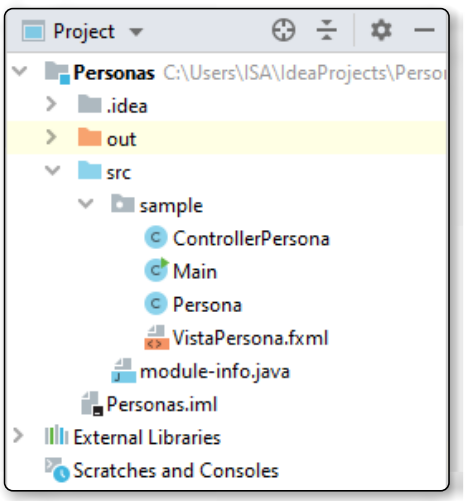
```
package sample;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        Parent root =
                    FXMLLoader.load(getClass().getResource("SumaVista.fxml"));
        primaryStage.setTitle("Resuelto Suma fx");
        primaryStage.setScene(new Scene(root, 300, 275));
        primaryStage.show();
    }


    public static void main(String[] args) {
        launch(args);
    }
}
```

Controlador.java

```
package sample;

import javafx.fxml.Initializable;
import java.net.URL;
import java.util.ResourceBundle;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import sample.modelo.Suma;

public class Controlador implements Initializable {

    @FXML  private Button btnSumar;
    @FXML  private TextField txtOp1;
    @FXML  private TextField txtOp2;
    @FXML  private TextField txtResultado;
    @Override
    public void initialize(URL url, ResourceBundle rb) {
```

```
        // TODO
    }

    public void sumar(ActionEvent actionEvent) {
        try {
            // Obtengo los parametros
            int op1 = Integer.parseInt(this.txtOp1.getText());
            int op2 = Integer.parseInt(this.txtOp2.getText());

            // Creo una instancia del modelo
            Suma s = new Suma(op1, op2);

            // Realizo la suma
            int resultado = s.suma();

            // Muestro el resultado
            this.txtResultado.setText(resultado + "");

        } catch (NumberFormatException e) {
            // Alerta de error
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setHeaderText(null);
            alert.setTitle("Error");
            alert.setContentText("Formato incorrecto");
            alert.showAndWait();
        }
    }
}
```

R3. ACTIVIDAD TableView 1: Personas



- Al seleccionar en un registro, mostrará los datos en los txt.
- Al pulsar el botón Modificar, el registro seleccionado se modificará con los datos que hay en los txt (editando el objeto).
- Al pulsar el botón Eliminar, el registro seleccionado se eliminará de la tabla.

Solución:

Main.java

```
package sample;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
import java.io.IOException;

import static javafx.fxml.FXMLLoader.load;

public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        try {
            Parent root=load(getClass().getResource("VistaPersona.fxml"));
            primaryStage.setTitle("Lista de personas");
            primaryStage.setScene(new Scene(root, 300, 275));
            primaryStage.show();
        }catch(IOException e){
            System.out.println(e.getMessage());
        }
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

ControllerPersona.java

```
package sample;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
```

```
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.MouseEvent;

import java.net.URL;
import java.util.ResourceBundle;

public class ControllerPersona implements Initializable {



    @FXML private TableView tblPersonas;
    @FXML private TableColumn colEdad;
    @FXML private  TableColumn colApellidos;
    @FXML private  TableColumn colNombre;
    @FXML private TextField txtEdad;
    @FXML private TextField txtApellidos;
    @FXML private  TextField txtNombre;
    @FXML private Button btnAgregar;
    @FXML private  Button btnEliminar;
    @FXML private Button btnModificar;
    private ObservableList<Persona> personas;

    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        personas = FXCollections.observableArrayList();

        colNombre.setCellValueFactory(new PropertyValueFactory("nombre"));
        colApellidos.setCellValueFactory(new PropertyValueFactory("apellidos"));
        colEdad.setCellValueFactory(new PropertyValueFactory("edad"));
    }

    public void agregarPersona(ActionEvent actionEvent) {
        try {
            // Obtengo los datos del formulario
            String nombre=txtNombre.getText();
            String apellidos=txtApellidos.getText();
            int edad=Integer.parseInt(txtEdad.getText());

            // Creo una persona
            Persona p=new Persona(nombre, apellidos, edad);
            // Compruebo si la persona esta en el lista
            if (!personas.contains(p)) {
                // Lo añado a la lista
                personas.add(p);
                // Seteo los items
                tblPersonas.setItems(personas);

                Alert alert = new Alert(Alert.AlertType.INFORMATION);
                alert.setHeaderText(null);
```

```
                alert.setTitle("Info");
                alert.setContentText("Persona añadida");
                alert.showAndWait();

                //pongo en blanco los txt
               txtNombre.setText("");
               txtApellidos.setText("");
               txtEdad.setText("");

            } else {

                Alert alert = new Alert(Alert.AlertType.ERROR);
                alert.setHeaderText(null);
                alert.setTitle("Error");
                alert.setContentText("La persona existe");
                alert.showAndWait();
            }

        } catch (NumberFormatException e) {
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setHeaderText(null);
            alert.setTitle("Error");
            alert.setContentText("Formato incorrecto");
            alert.showAndWait();
        }
    }

    public void seleccionar(MouseEvent mouseEvent) {
        // Obtengo la persona seleccionada en el TableView tblPersonas
        Persona p =(Persona) tblPersonas.getSelectionModel().getSelectedItem();

        // Si no es nula actualizo los campos
        if (p != null) {
            txtNombre.setText(p.getNombre());
            txtApellidos.setText(p.getApellidos());
            txtEdad.setText(p.getEdad() + "");
        }
    }

    @FXML
    private void modificar(ActionEvent event) {

        // Obtengo la persona seleccionada en el TableView tblPersonas
        Persona p = (Persona) tblPersonas.getSelectionModel().getSelectedItem();

        // Si la persona es nula, lanzo error
        if (p == null) {
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setHeaderText(null);
            alert.setTitle("Error");
```

```
            alert.setContentText("Debes seleccionar una persona");
            alert.showAndWait();
        } else {

            try {
                // Obtengo los datos del formulario
                String nombre = txtNombre.getText();
                String apellidos = txtApellidos.getText();
                int edad = Integer.parseInt(txtEdad.getText());

                // Creo una persona
                Persona aux = new Persona(nombre, apellidos, edad);

                // Compruebo si la persona esta en el lista
                if (!personas.contains(aux)) {

                    // Modifico el objeto
                    p.setNombre(aux.getNombre());
                    p.setApellidos(aux.getApellidos());
                    p.setEdad(aux.getEdad());

                    // Refresco la tabla
                    tblPersonas.refresh();

                    Alert alert = new Alert(Alert.AlertType.INFORMATION);
                    alert.setHeaderText(null);
                    alert.setTitle("Info");
                    alert.setContentText("Persona modificada");
                    alert.showAndWait();

                } else {

                    Alert alert = new Alert(Alert.AlertType.ERROR);
                    alert.setHeaderText(null);
                    alert.setTitle("Error");
                    alert.setContentText("La persona existe");
                    alert.showAndWait();
                }
            } catch (NumberFormatException e) {

                Alert alert = new Alert(Alert.AlertType.ERROR);
                alert.setHeaderText(null);
                alert.setTitle("Error");
                alert.setContentText("Formato incorrecto");
                alert.showAndWait();
            }

        }

    }
```

```
    @FXML
    private void eliminar(ActionEvent event) {

        // Obtengo la persona seleccionada
        Persona p = (Persona) tblPersonas.getSelectionModel().getSelectedItem();

        // Si la persona es nula, lanzo error
        if (p == null) {
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setHeaderText(null);
            alert.setTitle("Error");
            alert.setContentText("Debes seleccionar una persona");
            alert.showAndWait();
        } else {

            // La elimino de la lista
            personas.remove(p);
            // Refresco la lista
            tblPersonas.refresh();

            Alert alert = new Alert(Alert.AlertType.INFORMATION);
            alert.setHeaderText(null);
            alert.setTitle("Info");
            alert.setContentText("Persona eliminada");
            alert.showAndWait();

        }

    }
}
```

Veamos ahora el modelo:

Persona.java

```
package sample;

import java.util.Objects;

public class Persona {
    private String nombre;
    private String apellidos;
    private int edad;

    public Persona(String nombre, String apellidos, int edad) {
        this.nombre = nombre;
        this.apellidos = apellidos;
        this.edad = edad;
```

```
}

public String getNombre() {
    return nombre;
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}

public String getApellidos() {
    return apellidos;
}

public void setApellidos(String apellidos) {
    this.apellidos = apellidos;
}

public int getEdad() {
    return edad;
}

public void setEdad(int edad) {
    this.edad = edad;
}

@Override
public int hashCode() {
    int hash = 3;
    return hash;
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final Persona other = (Persona) obj;
    if (this.edad != other.edad) {
        return false;
    }
    if (!Objects.equals(this.nombre, other.nombre)) {
        return false;
    }
```

```
        if (!Objects.equals(this.apellidos, other.apellidos)) {
            return false;
        }
        return true;
    }
}
```

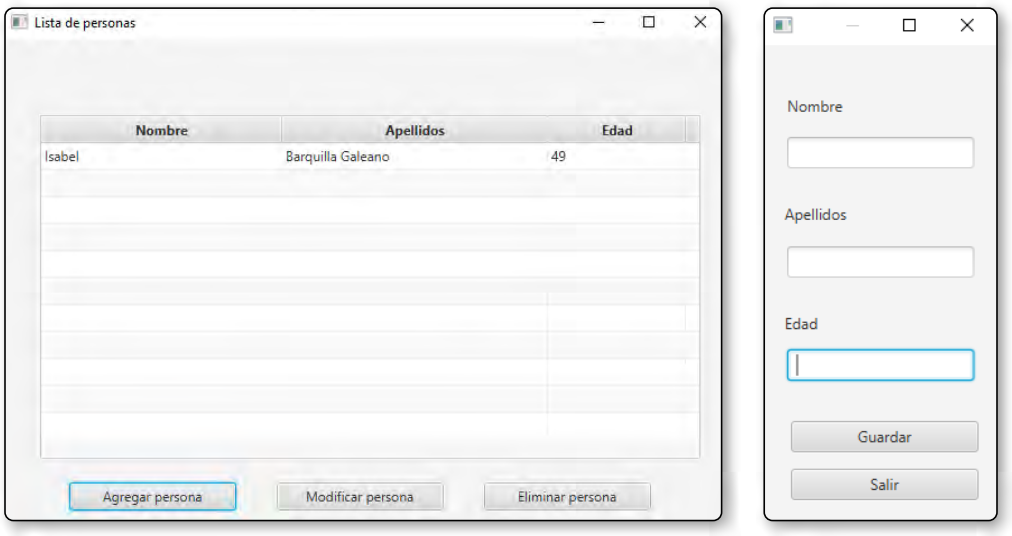
VistaPersona.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>


<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"
            minWidth="-Infinity" prefHeight="445.0" prefWidth="637.0"
            xmlns="http://javafx.com/javafx/11.0.1"
            xmlns:fx="http://javafx.com/fxml/1"
            fx:controller="sample.ControllerPersona">
    <children>
        <Button fx:id="btnAgregar" layoutX="25.0" layoutY="348.0"
                mnemonicParsing="false" onAction="#agregarPersona"
                prefHeight="25.0" prefWidth="149.0" text="Agregar persona"/>
        <TextField fx:id="txtNombre" layoutX="25.0" layoutY="102.0"/>
        <Label layoutX="28.0" layoutY="67.0" text="Nombre"/>
        <Label layoutX="25.0" layoutY="158.0" text="Apellidos"/>
        <TextField fx:id="txtApellidos" layoutX="25.0" layoutY="193.0"/>
        <TextField fx:id="txtEdad" layoutX="25.0" layoutY="287.0"/>
        <Label layoutX="25.0" layoutY="255.0" text="Edad"/>
        <TableView fx:id="tblPersonas" layoutX="220.0"
                   layoutY="67.0" onMouseClicked="#seleccionar"
                   prefHeight="306.0" prefWidth="403.0">
            <columns>
                <TableColumn fx:id="colNombre" prefWidth="147.0"
                             text="Nombre"/>
                <TableColumn fx:id="colApellidos" prefWidth="145.0"
                             text="Apellidos"/>
                <TableColumn fx:id="colEdad" prefWidth="105.0" text="Edad"/>
            </columns>
        </TableView>
        <Button fx:id="btnModificar" layoutX="241.0" layoutY="394.0"
                mnemonicParsing="false" onAction="#modificar"
                prefHeight="25.0" prefWidth="169.0" text="Modificar"/>
        <Button fx:id="btnEliminar" layoutX="444.0" layoutY="394.0"
                mnemonicParsing="false" onAction="#eliminar"
```

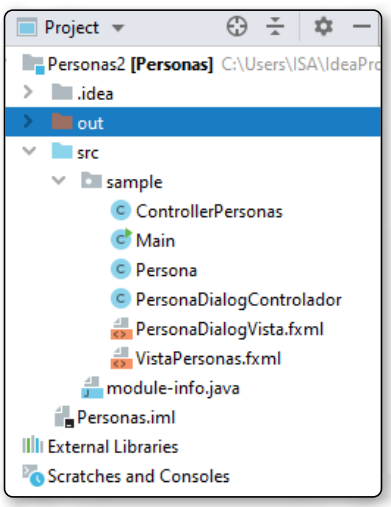
```
                prefHeight="25.0" prefWidth="156.0" text="Eliminar"/>
    </children>
</AnchorPane>
```

R4. ACTIVIDAD TableView 2: Personas2

Copia el proyecto anterior y modifícalo para que quede así:



- Al pulsar el botón *Agregar persona*, saldrá la ventana que permitirá introducir los datos. Al guardar, se cerrará la ventana y se añadirán los datos a la tabla.
- Al pulsar el botón *Modificar*, el registro seleccionado se modificará desde la ventana.
- Al pulsar el botón Eliminar, el registro seleccionado se eliminará de la tabla.



Solución:

Persona.java

```
(igual que el anterior)
```

Main.java

```
package sample;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
import java.io.IOException;

import static javafx.fxml.FXMLLoader.load;

public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        try {
            Parent root=load(getClass().getResource("VistaPersonas.fxml"));
            primaryStage.setTitle("Lista de personas");
            primaryStage.setScene(new Scene(root, 637, 445));
            primaryStage.show();
        }catch(IOException e){
            System.out.println(e.getMessage());
        }
    }


    public static void main(String[] args) {
        launch(args);
    }
}
```

VistaPersonas.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
```

```
<AnchorPane prefHeight="445.0" prefWidth="637.0" xmlns="http://javafx.com/
javafx/11.0.1"
            xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.
ControllerPersonas">
    <children>
        <Button fx:id="btnAgregar" layoutX="60.0" layoutY="394.0"
                mnemonicParsing="false" onAction="#agregarPersona"
                prefHeight="25.0" prefWidth="149.0" text="Agregar persona"/>
        <TableView fx:id="tblPersonas" layoutX="34.0"
                   layoutY="67.0" prefHeight="306.0"
                   prefWidth="589.0">
            <columns>
                <TableColumn fx:id="colNombre" prefWidth="215.0"
                             text="Nombre"/>
                <TableColumn fx:id="colApellidos" prefWidth="237.0"
                             text="Apellidos"/>
                <TableColumn fx:id="colEdad" prefWidth="123.0" text="Edad"/>
            </columns>
        </TableView>
        <Button fx:id="btnModificar"  layoutX="245.0" layoutY="394.0"
                mnemonicParsing="false"onAction="#modificar"prefHeight="25.0"
                prefWidth="149.0" text="Modificar persona"/>
        <Button fx:id="btnEliminar"   layoutX="430.0" layoutY="394.0"
                mnemonicParsing="false onAction="#eliminar" refHeight="25.0"
                prefWidth="149.0" text="Eliminar persona"/>
    </children>
</AnchorPane>
```

ControllerPersonas.java

```
package sample;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.stage.Modality;
import javafx.stage.Stage;

import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;
```

```
public class ControllerPersonas implements Initializable {

    @FXML private Button btnModificar;
    @FXML private Button btnEliminar;
    @FXML private Button btnAgregar;
    @FXML private TableView <Persona> tblPersonas;
    @FXML private TableColumn colEdad;
    @FXML private  TableColumn colApellidos;
    @FXML private  TableColumn colNombre;
    ObservableList <Persona> personas;
    /**
     * Inicializa la clase controladora
     */
    @Override
    public void initialize(URL url, ResourceBundle rb) {
        personas = FXCollections.observableArrayList();
        tblPersonas.setItems(personas);

        colNombre.setCellValueFactory(new PropertyValueFactory("nombre"));
        colApellidos.setCellValueFactory(new PropertyValueFactory("apellidos"));
        colEdad.setCellValueFactory(new PropertyValueFactory("edad"));
    }

    @FXML
    private void agregarPersona(ActionEvent event) {
        try {
            // Cargo la vista
            FXMLLoader loader = new FXMLLoader(getClass().
getResource("PersonaDialogVista.fxml"));

            // Cargo la ventana
            Parent root = loader.load();

            // Cojo el controlador
            PersonaDialogControlador controlador = loader.getController();
            controlador.inicializaAtributos(personas);

            // Creo el Scene
            Scene scene = new Scene(root);
            Stage stage = new Stage();
            stage.initModality(Modality.APPLICATION_MODAL);
            stage.setScene(scene);
            stage.showAndWait();

            // Cojo la persona devuelta
            Persona p = controlador.getPersona();
            if (p != null) {

                // Añado la persona
                personas.add(p);

                // Refresco la tabla
```

```
                    tblPersonas.refresh();
                }

            } catch (IOException ex) {
                Alert alert = new Alert(Alert.AlertType.ERROR);
                alert.setHeaderText(null);
                alert.setTitle("Error");
                alert.setContentText(ex.getMessage());
                alert.showAndWait();
            }

        }

        @FXML
        private void modificar(ActionEvent event) {

            Persona p = tblPersonas.getSelectionModel().getSelectedItem();

            if (p == null) {
                Alert alert = new Alert(Alert.AlertType.ERROR);
                alert.setHeaderText(null);
                alert.setTitle("Error");
                alert.setContentText("Debes seleccionar una persona");
                alert.showAndWait();
            } else {
                try {
                    // Cargo la vista
                    FXMLLoader loader = new FXMLLoader(getClass().
    getResource("PersonaDialogVista.fxml"));

                    // Cargo la ventana
                    Parent root = loader.load();

                    // Cojo el controlador
                    PersonaDialogControlador controlador = loader.getController();
                    controlador.inicializaAtributos(personas, p);

                    // Creo el Scene
                    Scene scene = new Scene(root);
                    Stage stage = new Stage();
                    stage.initModality(Modality.APPLICATION_MODAL);
                    stage.setScene(scene);
                    stage.showAndWait();

                    // Cojo la persona devuelta
                    Persona aux = controlador.getPersona();
                    if (aux != null) {
                        tblPersonas.refresh();
                    }

                } catch (IOException ex) {
                    Alert alert = new Alert(Alert.AlertType.ERROR);
                    alert.setHeaderText(null);
```

```
                alert.setTitle("Error");
                alert.setContentText(ex.getMessage());
                alert.showAndWait();
            }
        }

    }

    @FXML
    private void eliminar(ActionEvent event) {

        Persona p = tblPersonas.getSelectionModel().getSelectedItem();

        if (p == null) {
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setHeaderText(null);
            alert.setTitle("Error");
            alert.setContentText("Debes seleccionar una persona");
            alert.showAndWait();
        } else {

            // Elimino la persona
            personas.remove(p);
            tblPersonas.refresh();

            Alert alert = new Alert(Alert.AlertType.INFORMATION);
            alert.setHeaderText(null);
            alert.setTitle("Info");
            alert.setContentText("Se ha borrado la persona");
            alert.showAndWait();
        }
    }
}
```

PersonaDialogVista.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<AnchorPane id="AnchorPane" focusTraversable="true" maxHeight="383.0"
        maxWidth="185.0" minHeight="345.0"  minWidth="185.0"
        prefHeight="383.0" prefWidth="185.0" snapToPixel="false"
        xmlns="http://javafx.com/javafx"
        xmlns:fx="http://javafx.com/fxml"
        fx:controller="sample.PersonaDialogControlador">
    <children>
```

```
        <Label layoutX="17.0" layoutY="213.0" text="Edad" />
        <TextField fx:id="txtEdad" layoutX="19.0" layoutY="242.0" />
        <TextField fx:id="txtApellidos" layoutX="19.0" layoutY="160.0" />
        <Label layoutX="17.0" layoutY="126.0" text="Apellidos" />
        <Label layoutX="19.0" layoutY="40.0" text="Nombre" />
        <TextField fx:id="txtNombre" layoutX="19.0" layoutY="73.0" />
        <Button fx:id="btnGuardar"
                layoutX="22.0"
                layoutY="299.0"
                mnemonicParsing="false"
                onAction="#guardar"
                prefHeight="25.0" prefWidth="149.0" text="Guardar" />
        <Button fx:id="btnSalir" layoutX="22.0" layoutY="337.0"
                mnemonicParsing="false" onAction="#salir"
                prefHeight="25.0" prefWidth="149.0" text="Salir" />
    </children>

</AnchorPane>
```

PersonaDialogControlador.java

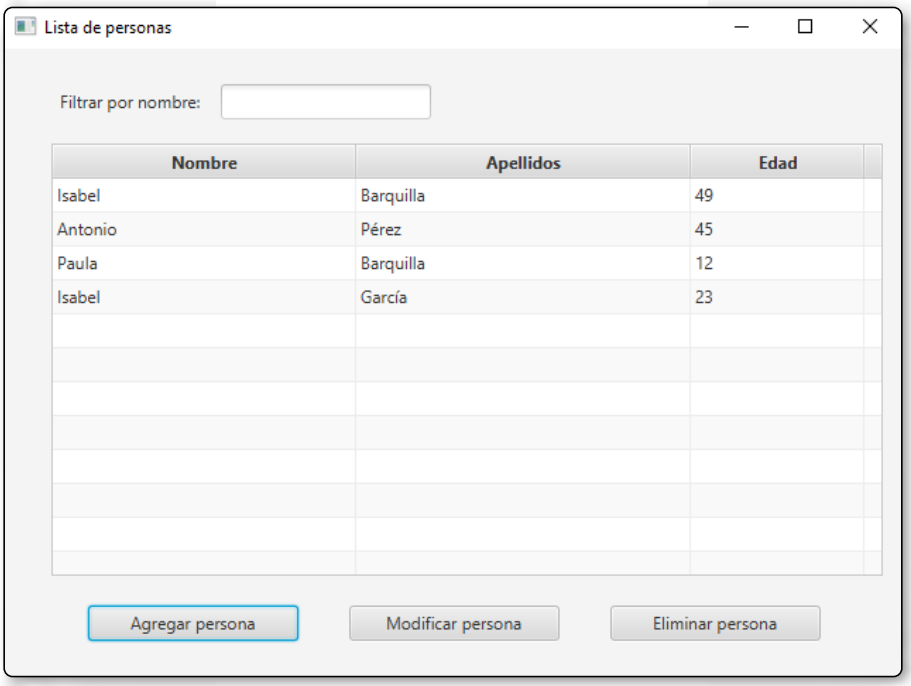
```
package sample;

import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.stage.Stage;

import java.net.URL;
import java.util.ResourceBundle;

public class PersonaDialogControlador implements Initializable {

    @FXML private TextField txtNombre;
    @FXML private Button btnGuardar;
    @FXML private Button btnSalir;
    @FXML private TextField txtEdad;
    @FXML private TextField txtApellidos;

    private Persona persona;
    private ObservableList<Persona> personas;

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

    }
    public Persona getPersona() {
```

```
        return persona;
    }
    public void inicializaAtributos(ObservableList<Persona> personas) {
        this.personas = personas;
    }
    public void inicializaAtributos(ObservableList<Persona> personas, Persona p){
        this.personas = personas;
        this.persona = p;

        // Cargo los datos de la persona
        this.txtNombre.setText(p.getNombre());
        this.txtApellidos.setText(p.getApellidos());
        this.txtEdad.setText(p.getEdad() + "");
    }


    public void guardar(ActionEvent actionEvent) {
        // Cojo los datos
        String nombre = txtNombre.getText();
        String apellidos = txtApellidos.getText();
        int edad = Integer.parseInt(txtEdad.getText());

        // Creo la persona
        Persona p = new Persona(nombre, apellidos, edad);

        // Compruebo si la persona existe
        if (!personas.contains(p)) {

            // Modificar o insertar
            if (this.persona != null) {

                // Modifico el objeto
               persona.setNombre(nombre);
               persona.setApellidos(apellidos);
               persona.setEdad(edad);

                Alert alert = new Alert(Alert.AlertType.INFORMATION);
                alert.setHeaderText(null);
                alert.setTitle("Información");
                alert.setContentText("Se ha modificado correctamente");
                alert.showAndWait();

            } else {
                // Insertando persona nueva
                persona = p;
                Alert alert = new Alert(Alert.AlertType.INFORMATION);
                alert.setHeaderText(null);
                alert.setTitle("Información");
                alert.setContentText("Se ha añadido correctamente");
                alert.showAndWait();

            }
```

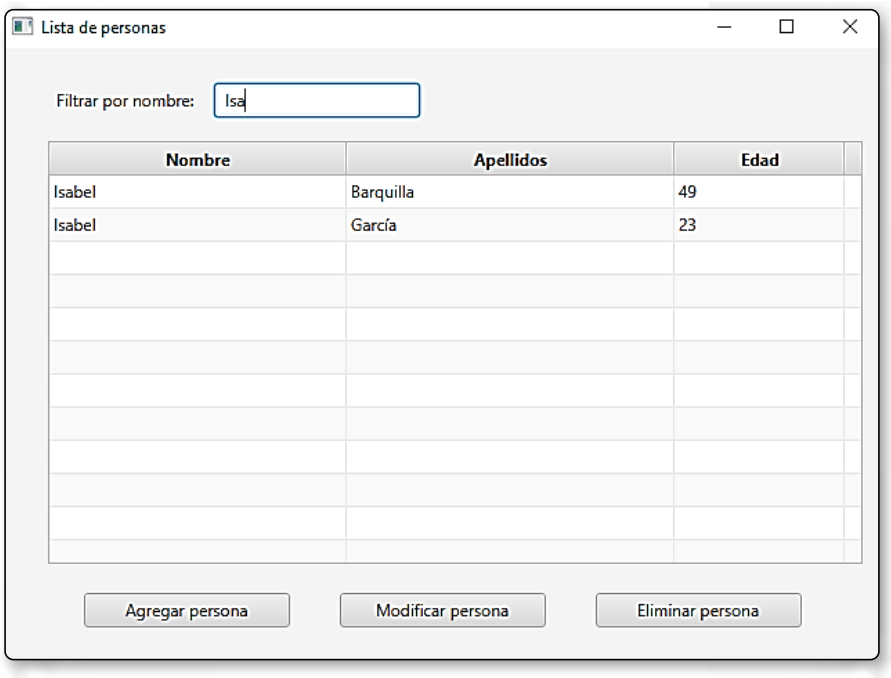
```
            // Cerrar la ventana
            Stage stage =(Stage) btnGuardar.getScene().getWindow();
            stage.close();
        } else {
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setHeaderText(null);
            alert.setTitle("Error");
            alert.setContentText("La persona ya existe");
            alert.showAndWait();
        }
    }

    public void salir(ActionEvent actionEvent) {
        persona = null;
        // Cerrar la ventana
        Stage stage = (Stage) btnGuardar.getScene().getWindow();
        stage.close();
    }
}
```

R5. ACTIVIDAD Filtro en TableView : Personas

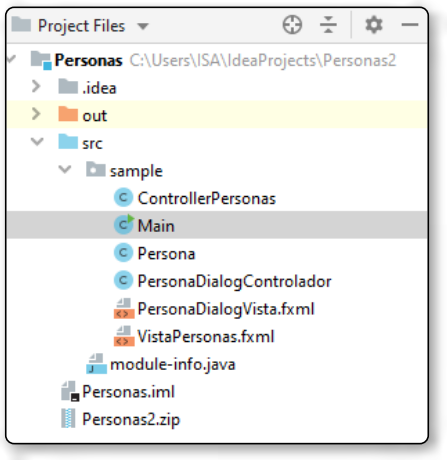
Modifica el proyecto anterior y añade un textField para poder poner un filtro a la tabla de personas cuando escribamos en el txt de arriba, filtrara los nombre de las personas que contengan el texto que se indique.



Así:



Veamos el código:



VistaPersonas.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
```

```
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.AnchorPane?>

<AnchorPane prefHeight="445.0" prefWidth="637.0"
            xmlns="http://javafx.com/javafx/11.0.1"
            xmlns:fx="http://javafx.com/fxml/1"
            fx:controller="sample.ControllerPersonas">
    <children>
        <Button fx:id="btnAgregar" layoutX="60.0" layoutY="394.0"
                mnemonicParsing="false" onAction="#agregarPersona"
                prefHeight="25.0" prefWidth="149.0" text="Agregar persona" />
        <TableView fx:id="tblPersonas" layoutX="34.0" layoutY="67.0"
                prefHeight="306.0" prefWidth="589.0">
            <columns>
                <TableColumn fx:id="colNombre" prefWidth="215.0"
                             text="Nombre" />
                <TableColumn fx:id="colApellidos" prefWidth="237.0"
                           "Apellidos" />
                <TableColumn fx:id="colEdad" prefWidth="123.0" text="Edad" />
            </columns>
        </TableView>
        <Button fx:id="btnModificar" layoutX="245.0" layoutY="394.0"
                mnemonicParsing="false" onAction="#modificar"
                prefHeight="25.0" prefWidth="149.0"
                text="Modificar persona" />
        <Button fx:id="btnEliminar" layoutX="430.0" layoutY="394.0"
                mnemonicParsing="false" onAction="#eliminar"
                prefHeight="25.0" prefWidth="149.0"
                text="Eliminar persona" />
        <Label layoutX="40.0" layoutY="29.0" text="Filtrar por nombre:" />
        <TextField fx:id="txtFiltrarNombre" layoutX="154.0" layoutY="25.0"
                   onKeyReleased="#filtrarNombre" />
    </children>
</AnchorPane>
```

ControllerPersonas.java

```
package sample;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
```

```
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.KeyEvent;
import javafx.stage.Modality;
import javafx.stage.Stage;

import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;

public class ControllerPersonas implements Initializable {

    @FXML private Button btnModificar;
    @FXML private Button btnEliminar;
    @FXML private Button btnAgregar;
    @FXML private TableView<Persona> tblPersonas;
    @FXML private TableColumn colEdad;
    @FXML private TableColumn colApellidos;
    @FXML private TableColumn colNombre;
    @FXML private TextField txtFiltrarNombre;
    private ObservableList<Persona> personas;
    private ObservableList<Persona> filtroPersonas;

    /**
     * Inicializa la clase controladora
     */
    @Override
    public void initialize(URL url, ResourceBundle rb) {
        personas=FXCollections.observableArrayList();
        filtroPersonas = FXCollections.observableArrayList();

        tblPersonas.setItems(personas);

        colNombre.setCellValueFactory(new PropertyValueFactory("nombre"));
        colApellidos.setCellValueFactory(new PropertyValueFactory("apellidos"));
        colEdad.setCellValueFactory(new PropertyValueFactory("edad"));
    }

    @FXML
    private void agregarPersona(ActionEvent event) {
        try {
            // Cargo la vista
            FXMLLoader loader=new FXMLLoader(getClass().
getResource("PersonaDialogVista.fxml"));

            // Cargo la ventana
            Parent root=loader.load();

            // Cojo el controlador
            PersonaDialogControlador controlador=loader.getController();
```

```
            controlador.inicializaAtributos(personas);

            // Creo el Scene
            Scene scene=new Scene(root);
            Stage stage=new Stage();
            stage.initModality(Modality.APPLICATION_MODAL);
            stage.setScene(scene);
            stage.showAndWait();

            // Cojo la persona devuelta
            Persona p = controlador.getPersona();
            if (p != null) {
                personas.add(p);
                if (p.getNombre().toLowerCase().contains(this.txtFiltrarNombre.
getText().toLowerCase())) {
                    this.filtroPersonas.add(p);
                }
                this.tblPersonas.refresh();
            }

        } catch (IOException ex) {
            Alert alert=new Alert(Alert.AlertType.ERROR);
            alert.setHeaderText(null);
            alert.setTitle("Error");
            alert.setContentText(ex.getMessage());
            alert.showAndWait();
        }

    }

    @FXML
    private void modificar(ActionEvent event) {

        Persona p=tblPersonas.getSelectionModel().getSelectedItem();

        if (p == null) {
            Alert alert=new Alert(Alert.AlertType.ERROR);
            alert.setHeaderText(null);
            alert.setTitle("Error");
            alert.setContentText("Debes seleccionar una persona");
            alert.showAndWait();
        } else {
            try {
                // Cargo la vista
                FXMLLoader loader=new FXMLLoader(getClass().
getResource("PersonaDialogVista.fxml"));

                // Cargo la ventana
                Parent root=loader.load();
```

```
                // Cojo el controlador
                PersonaDialogControlador controlador=loader.getController();
                controlador.inicializaAtributos(personas, p);

                // Creo el Scene
                Scene scene=new Scene(root);
                Stage stage=new Stage();
                stage.initModality(Modality.APPLICATION_MODAL);
                stage.setScene(scene);
                stage.showAndWait();

                // Cojo la persona devuelta
                Persona pSeleccionado = controlador.getPersona();
                if (pSeleccionado != null) {
                    if (!pSeleccionado.getNombre().toLowerCase().
                            contains(txtFiltrarNombre.getText().toLowerCase())) {
                        filtroPersonas.remove(pSeleccionado);
                    }
                    tblPersonas.refresh();
                }

            } catch (IOException ex) {
                Alert alert=new Alert(Alert.AlertType.ERROR);
                alert.setHeaderText(null);
                alert.setTitle("Error");
                alert.setContentText(ex.getMessage());
                alert.showAndWait();
            }
        }

    }

    @FXML
    private void eliminar(ActionEvent event) {

        Persona p=tblPersonas.getSelectionModel().getSelectedItem();

        if (p == null) {
            Alert alert=new Alert(Alert.AlertType.ERROR);
            alert.setHeaderText(null);
            alert.setTitle("Error");
            alert.setContentText("Debes seleccionar una persona");
            alert.showAndWait();
        } else {

            // Elimino la persona
            personas.remove(p);
            filtroPersonas.remove(p);
            tblPersonas.refresh();
```

```
            Alert alert=new Alert(Alert.AlertType.INFORMATION);
            alert.setHeaderText(null);
            alert.setTitle("Info");
            alert.setContentText("Se ha borrado la persona");
            alert.showAndWait();
        }
    }

    @FXML
    private void filtrarNombre(KeyEvent event) {

        String filtroNombre=this.txtFiltrarNombre.getText();

        // Si el texto del nombre esta vacio, seteamos la tabla de personas con el
original
        if (filtroNombre.isEmpty()) {
            this.tblPersonas.setItems(personas);
        } else {

            // Limpio la lista
            this.filtroPersonas.clear();

            for (Persona p : this.personas) {
                if (p.getNombre().toLowerCase().contains(filtroNombre.toLowerCase()))
{
                    this.filtroPersonas.add(p);
                }
            }

            this.tblPersonas.setItems(filtroPersonas);

        }
    }
}
```

PersonaDialogcontrolador.java

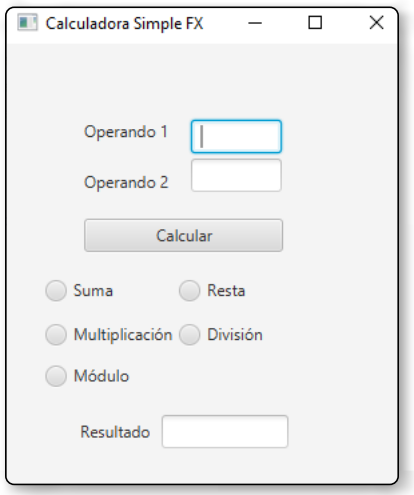
```
package sample;

import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.stage.Stage;

import java.net.URL;
```

```
import java.util.ResourceBundle;

public class PersonaDialogControlador implements Initializable {

    @FXML private TextField txtNombre;
    @FXML private Button btnGuardar;
    @FXML private Button btnSalir;
    @FXML private TextField txtEdad;
    @FXML private TextField txtApellidos;

    private Persona persona;
    private ObservableList<Persona> personas;

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

    }
    public Persona getPersona() {
        return persona;
    }
    public void inicializaAtributos(ObservableList<Persona> personas) {
        this.personas = personas;
    }
    public void inicializaAtributos(ObservableList<Persona> personas, Persona p){
        this.personas = personas;
        this.persona = p;

        // Cargo los datos de la persona
        this.txtNombre.setText(p.getNombre());
        this.txtApellidos.setText(p.getApellidos());
        this.txtEdad.setText(p.getEdad() + "");
    }


    public void guardar(ActionEvent actionEvent) {
        // Cojo los datos
        String nombre = txtNombre.getText();
        String apellidos = txtApellidos.getText();
        int edad = Integer.parseInt(txtEdad.getText());

        // Creo la persona
        Persona p = new Persona(nombre, apellidos, edad);

        // Compruebo si la persona existe
        if (!personas.contains(p)) {

            // Modificar o insertar
            if (this.persona != null) {

                // Modifico el objeto
```

```
                persona.setNombre(nombre);
                persona.setApellidos(apellidos);
                persona.setEdad(edad);

                 Alert alert = new Alert(Alert.AlertType.INFORMATION);
                 alert.setHeaderText(null);
                 alert.setTitle("Información");
                 alert.setContentText("Se ha modificado correctamente");
                 alert.showAndWait();

            } else {
                // Insertando persona nueva
                persona = p;
                Alert alert = new Alert(Alert.AlertType.INFORMATION);
                alert.setHeaderText(null);
                alert.setTitle("Información");
                alert.setContentText("Se ha añadido correctamente");
                alert.showAndWait();

            }

            // Cerrar la ventana
            Stage stage =(Stage) btnGuardar.getScene().getWindow();
            stage.close();
        } else {
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setHeaderText(null);
            alert.setTitle("Error");
            alert.setContentText("La persona ya existe");
            alert.showAndWait();
        }
    }

    public void salir(ActionEvent actionEvent) {
        persona = null;
        // Cerrar la ventana
        Stage stage = (Stage) btnGuardar.getScene().getWindow();
        stage.close();
    }
}
```

9.7 EJERCICIOS PROPUESTOS

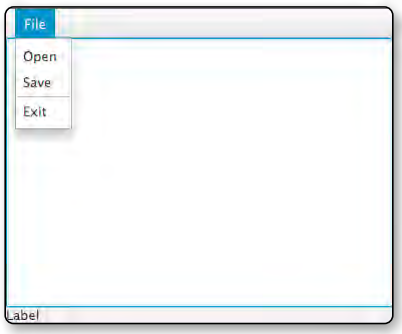
- **EJERCICIO P1 CALCULADORA SIMPLE**: a partir de la actividad 4 del sumador, modifica el programa para que sea una calculadora sencilla con el aspecto siguiente:



El usuario seleccionará la operación a realizar a través de los radiobotones. Recuerda agruparlos en un *ToggleGroup*. Utiliza try{…}catch para controlar que el usuario introduzca bien los operandos y que no se permita la división por cero.

- **EJERCICIO P2 TableView.** Modifica la actividad 5 para que se introduzcan en la lista los elementos que el usuario introduzca el isbn, título y autor. Una vez introducidos los datos en la lista, deja limpios los campos de texto.
  - Al seleccionar en un registro, mostrara los datos en los txt.
  - Al pulsar el botón Modificar, el registro seleccionado se modificará con los datos que hay en los txt (editando el objeto).
  - Al pulsar el botón Eliminar, el registro seleccionado se eliminará de la tabla.

- **EJERCICIO P3. Bloc de notas**. Crea un proyecto denominado BlocNotasJavaFx. Utiliza un *BorderPane* (400 pixels ancho x 300 pixels de alto), y añade un menú en la parte de arriba, una *TextArea* en el centro y una *Label* al pie. El menú debe tener un *File* con cuatro elementos: Open, Save, un separador y Exit. Tendrá el aspecto siguiente:



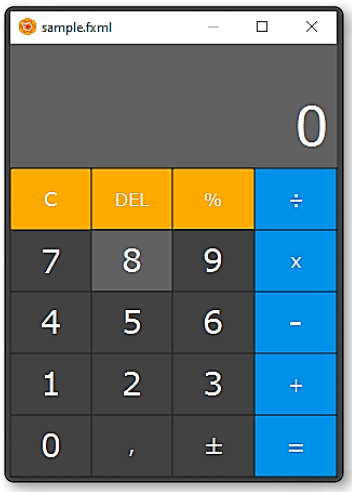
Los menús deben funcionar.

(Pista: https://docs.oracle.com/javafx/2/ui_controls/file-chooser.htm

http://acodigo.blogspot.com/2014/12/file-chooser-javafx-abrir-archivos.html

http://javainutil.blogspot.com/2013/02/javafx-abrir-y-guardar-archivos.html)

- **EJERCICIO P4. Calculadora**. Crea un proyecto javaFx que implemente una calculadora que tenga los siguientes elementos. Averigua cómo aplicar css para que tenga el aspecto siguiente:



9.8 EJERCICIO AMPLIACIÓN

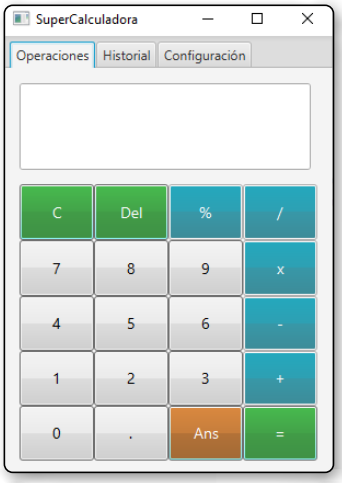
- **EJERCICIO P5 CalculadoraPro**

  Realiza una calculadora tal cual la imagen siguiente. Se dispondrán de tres pestañas: Operaciones, Historial, Configuración.

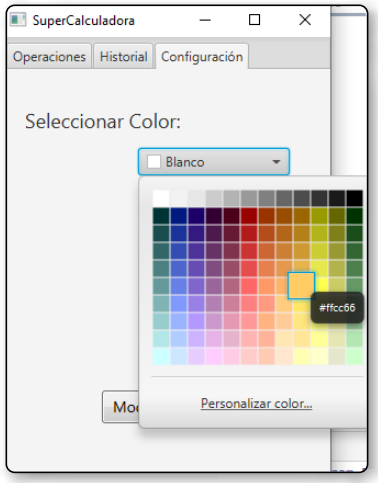
  Nota que en las operaciones se ha puesto la tecla Ans, que proporcionará el resultado de la última operación realizada.

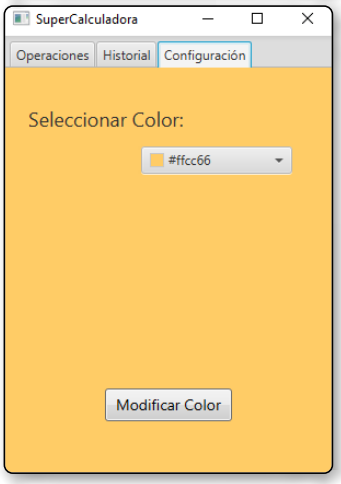
  El historial guarda las operaciones que se han ido realizando.

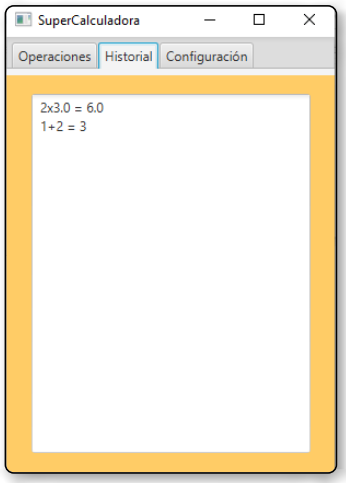
  En configuración se permite seleccionar el color del fondo de la calculadora y cambiarlo.

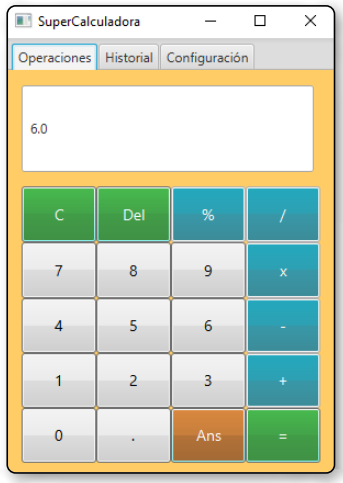

SuperCalculadora
Operaciones
Historial
Configuración
C
Del
%
/
7
8
9
x
4
5
6
-
1
2
3
+
0
.
Ans
=


SuperCalculadora
Operaciones
Historial
Configuración
2x3.0 = 6.0
1+2 = 3


SuperCalculadora
Operaciones
Historial
Configuración
Seleccionar Color:
Blanco
#ffcc66
Personalizar color...


SuperCalculadora
Operaciones
Historial
Configuración
Seleccionar Color:
#ffcc66
Modificar Color


SuperCalculadora
Operaciones
Historial
Configuración
2x3.0 = 6.0
1+2 = 3


SuperCalculadora
Operaciones
Historial
Configuración
6.0
C
Del
%
/
7
8
9
x
4
5
6
-
1
2
3
+
0
.
Ans
=

9.9 ACTIVIDADES DE REFUERZO/ AMPLIACIÓN

TUTORIAL APLICACIÓN SENCILLA JAVA INTELLIJ
https://www.youtube.com/playlist?list=PLuEZQoW9bRnR4CbS0Q6SJbGIVfspUTT5Y

TUTORIAL SPACE RUNNER
https://www.youtube.com/watch?v=DkIuA5ZEZ_U

INTRODUCCIÓN JAVAFX PARA DESARROLLO DE JUEGOS
https://gamedevelopment.tutsplus.com/es/tutorials/introduction-to-javafx-for-game-development--cms-23835

INFORMACIÓN GENERAL API JAVAFX
https://docs.oracle.com/javafx/2/api/index.html

EJEMPLOS LAYOUTS
https://www.javamexico.org/blogs/jose_manuel/iniciando_con_javafx_layouts_ejemplos_de_uso

Tutorial
https://openjfx.io/openjfx-docs/

Rellenando una tabla con bases de datos
https://www.youtube.com/watch?v=wIPXaKwhYJg

Hojas de estilo css
https://javiergarciaescobedo.es/programacion-en-java/96-javafx/481-hojas-de-estilo-css-con-javafx
https://code.makery.ch/es/library/javafx-tutorial/part4/

Guardar arraylist en ficheros
https://jnjsite.com/java-serializando-objetos-para-guardar-y-recuperar-en-ficheros/

Editor en ficheros
https://www.youtube.com/watch?v=17uSUxhRQ80

FileChooser
https://docs.oracle.com/javafx/2/ui_controls/file-chooser.htm