



P1. SCENE BUILDER

Interfaces Persona Computador

Depto. Sistemas Informáticos y Computación

UPV

Índice

- Introducción a JavaFX (visto en teoría)
- FXML
- JavaFX y el patrón MVC
- SceneBuilder
- NetBeans y la clase Controller
- Ejemplo guiado
- Ejercicio

Introducción a JavaFX:(vista en teoría)

1- Stage, Scene y grafo de escena

2- Aplicación JavaFX con fichero FXML

```
public class Hola extends Application {
```

```
    @Override
```

```
    public void start(Stage primaryStage) { // ventana principal
```

```
        // crear el grafo de escena
```

```
        FXMLLoader loader= new FXMLLoader(getClass().getResource("FXMLDocument.fxml"));
```

```
        Parent root = loader.load();
```

```
        //crear la escena
```

```
        Scene scene = new Scene(root, 300, 250); // Escena que muestra raíz
```

```
        //añadir la escena al stage
```

```
        primaryStage.setScene(scene);
```

```
        primaryStage.show();
```

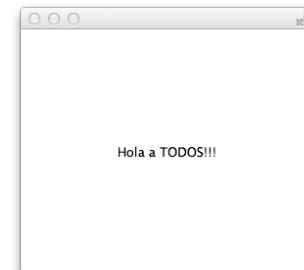
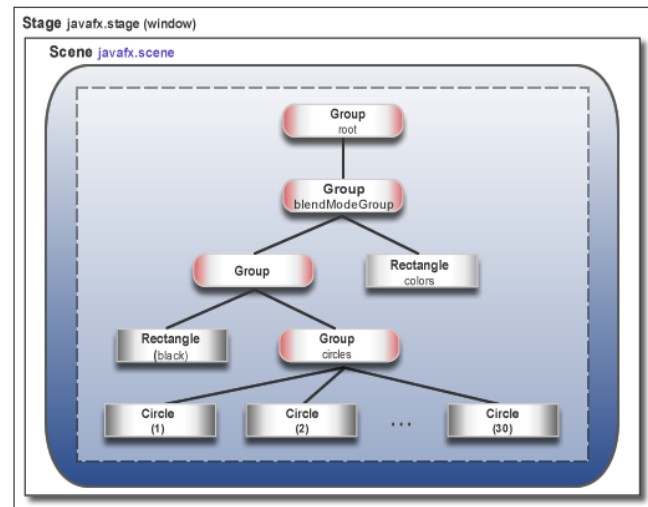
```
    }
```

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

```
        launch(args);
```

```
    }
```

```
}
```



```
<?xml version="1.0" encoding="UTF-8"?>
...
<StackPane id="Raiz" prefHeight="200" prefWidth="320"
xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/8" >
  <children>
    <Text layoutX="110.0" layoutY="97.0" strokeType="OUTSIDE" strokeWidth="0.0"
text="Hola a TODOS!!!" id="texto"/>
  </children>
</StackPane>
```

FXML:

```
<?xml version="1.0" encoding="UTF-8"?>
...
<StackPane id="Raiz" prefHeight="200" prefWidth="320"
xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/8" >
    <children>
        <Text layoutX="110.0" layoutY="97.0" strokeType="OUTSIDE" strokeWidth="0.0"
text="Hola a TODOS!!!" id="texto"/>
    </children>
</StackPane>
```

- Los ficheros FXML contienen la descripción total o parcial del grafo de escena que representa una interfaz de usuario. El fichero tiene formato XML. En tiempo de ejecución se utiliza para crear los objetos de java de la escena (node) y componerlos en el grafo de escena. Además puede incluir los nombres de los manejadores de eventos asociados a estos objetos.
- Los beneficios son:
 - que el diseñador puede trabajar con la interfaz mientras que el programador puede trabajar con el código sin la necesidad de trabajar sobre el mismo fichero
 - Se obliga a mantener la separación entre vista y controlador

¿Dónde esta el código de java asociado a los objetos aquí descritos?

FXML:

- En JavaFX la clase **Controladora** es una clase Java que contiene referencias (variables) a los objetos de la interfaz y los métodos (manejadores) que se encargan de atender los eventos sobre estos objetos
- **Un fichero FXML debe tener asignada una clase java Controladora** (normalmente llamada *Controller*). Un objeto de esta clase es creado de manera automática cuando se carga el fichero fxml (`load()`)
- Para enlazar las variables definidas en el controlador con los objetos creados en ejecución se utiliza inyección. Consiste en asignar la referencia a los objetos definidos en la clase no en el constructor sino en el momento de creación del grafo de escena (**`FXMLLoader.load()`**)
- También se automatiza el registro de los manejadores de eventos.

JavaFX y el patrón MVC :

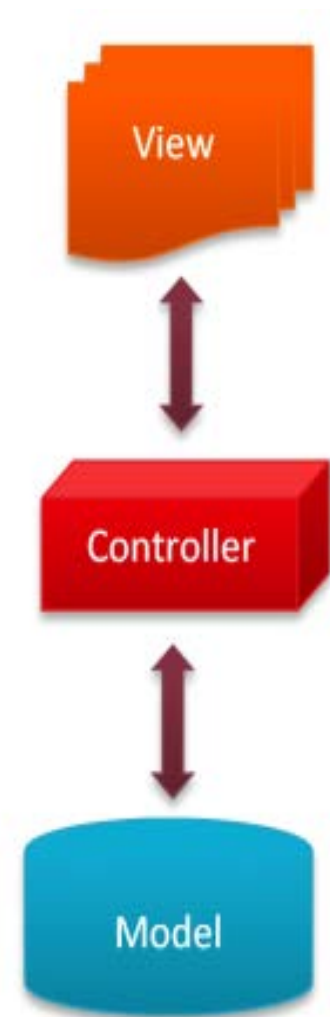
- Un fichero FXML es una descripción de la **vista**

```
<?xml version="1.0" encoding="UTF-8"?>
<?import java.lang.*?>
<?import java.util.*?>
<AnchorPane id="AnchorPane" prefHeight="200" prefWidth="320"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="hellofx.FXMLDocumentController">
  <children>
    <Button layoutX="126" layoutY="90" text="Click Me!" onAction="#handleButtonAction"
fx:id="button" />
    <Label layoutX="126" layoutY="120" minHeight="16" minWidth="69" fx:id="label" />
  </children>
</AnchorPane>
```

Admite una única clase **Controlador** con métodos para manejar los eventos

```
public class FXMLDocumentController implements Initializable {
    @FXML
    private Label label;
    @FXML
    private void handleButtonAction(ActionEvent event) {
        label.setText("Hello World!");
    }
    @Override
    public void initialize(URL url, ResourceBundle rb) {
    }
}
```

- Además podemos tener Clases de Java que definen los objetos de la aplicación ajenos a la interfaz. Se conocen como el **modelo**.



JavaFX y el patrón MVC :

• vista

```
<?xml version="1.0" encoding="UTF-8"?>
<?import java.lang.*?>
<?import java.util.*?>
<AnchorPane id="AnchorPane" prefHeight="200" prefWidth="320"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="hellofx.FXMLDocumentController">
  <children>
    <Button layoutX="126" layoutY="90" text="Click Me!" onAction="#handleButtonAction"
fx:id="button" />
    <Label layoutX="126" layoutY="120" minHeight="16" minWidth="69" fx:id="label" />
```

• Controlador

```
public class FXMLDocumentController implements Initializable {
    @FXML
    private Label label;
    @FXML
    private void handleButtonAction(ActionEvent event) {
        label.setText("Hello World!");
    }
    @Override
    public void initialize(URL url, ResourceBundle rb) {
    }
}
```



Netbeans dispone de una herramienta que permite:

- **CREAR** la clase de Java
- **SINCRONIZAR** los cambios entre FXML y Java

Modelo-Vista-Controlador (MVC)

- Modelo-Vista-Controlador (MVC) es un patrón de diseño que separa la lógica, la interfaz y los datos de la aplicación.
- **Vista:** es la presentación visual del modelo (los datos), no puede cambiar el modelo directamente y puede ser notificada cuando hay un cambio de estado del modelo
- **Controlador:** reacciona a la petición del usuario, ejecuta la acción adecuada y actualiza el modelo pertinente, o notifica cambios en el modelo a la vista.
- **Modelo:** no sabe nada del controlador/vista. Representa los datos (estado) y la lógica de la aplicación



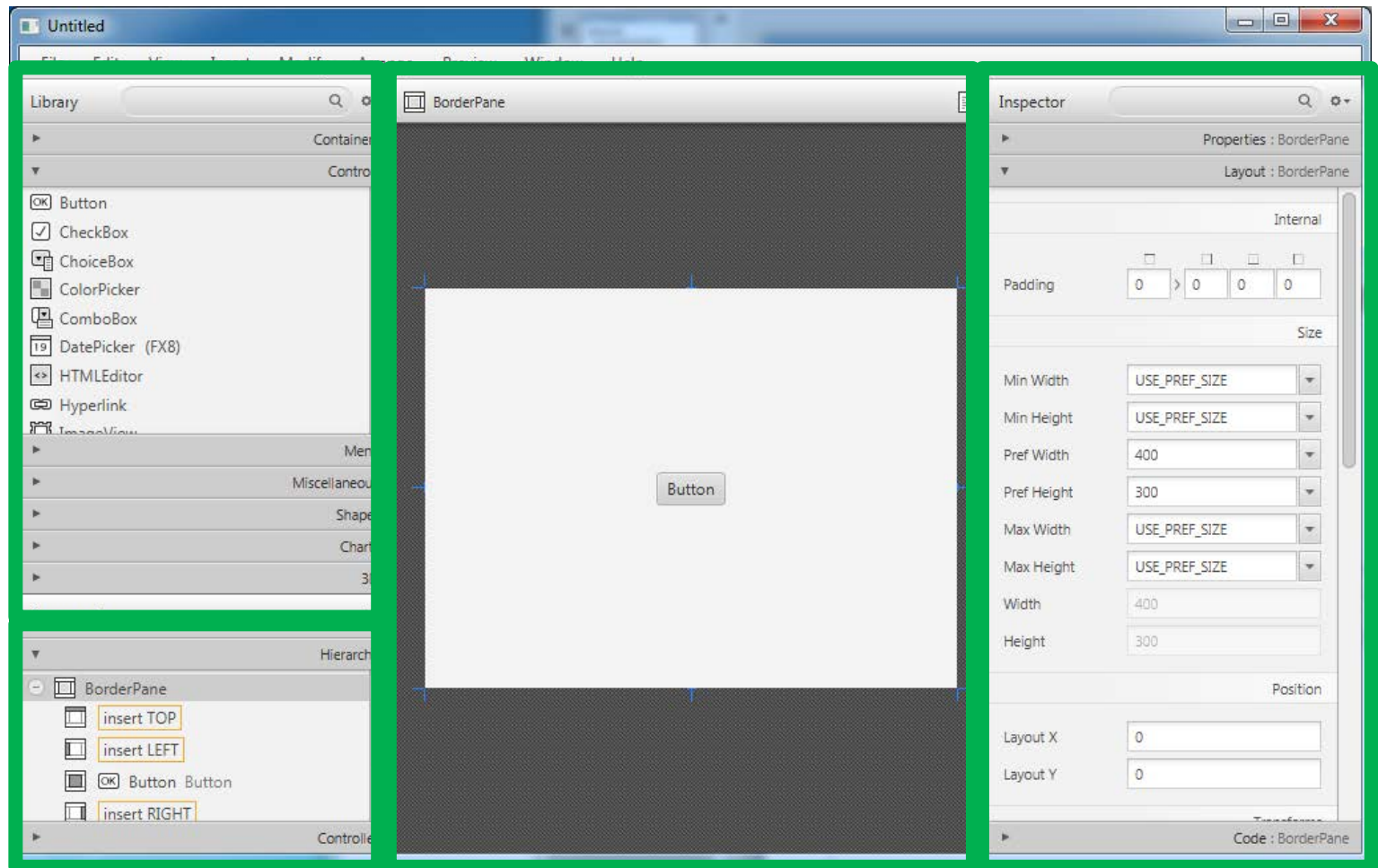
Scene Builder:

- **Scene Builder** es un editor externo de ficheros FXML desarrollado por Oracle y ahora continuado por Gluon <https://gluonhq.com/products/scene-builder/> que nos permite diseñar nuestras interfaces de forma visual
 - Scene Builder contiene todos los controles y contenedores definidos para JavaFX. Se pueden añadir nuevos componentes
 - Las ventanas se configuran arrastrando y soltando sobre el área de trabajo dichos controles
 - Se pueden ajustar las propiedades de los controles en un panel separado
 - El resultado se almacena en un fichero XML (con extensión FXML)
 - Se puede integrar con Netbeans, Eclipse, IntelliJ

Scene Builder

Organización de la pantalla principal

Librería
de
controles



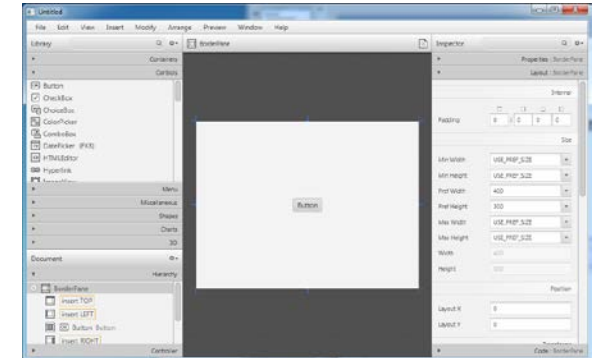
Jerarquía del
documento

Zona de trabajo

Inspector

Scene Builder

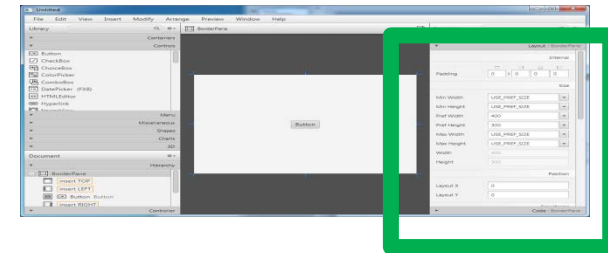
Cómo utilizar Scenebuilder:



- Para añadir un elemento lo arrastraremos desde la librería de controles hasta la zona de trabajo o hasta la jerarquía de controles en el documento.
- Es posible filtrar los controles por nombre
- Con el control seleccionado podremos modificar cualquiera de sus propiedades. Las propiedades son los atributos disponibles de cada control (posición, tamaño, apariencia, etc.)

Scene Builder

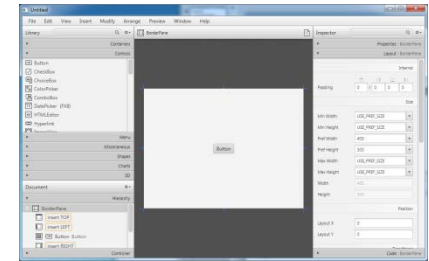
Cómo utilizar Scenebuilder:



- En el panel *Inspector* tenemos las secciones Properties, Layout y Code:
 - La sección **Properties** nos permite definir el estilo del elemento seleccionado en el área de trabajo. En JavaFX se utiliza plantillas CSS para definir el estilo de los elementos (lo veremos mas adelante)
 - La sección **Layouts** nos permite especificar el comportamiento en tiempo de ejecución del contenedor cuando cambiamos el tamaño de la ventana. También permite definir el tamaño del control. La información que aparece en esta sección dependerá del contenedor
 - La sección **Code** especifica los métodos que se ejecutarán ante la interacción del usuario sobre el control. El campo `fx:id` determina el nombre de la variable que tendrá este objeto dentro de la clase controladora.
 - Esta sección es muy importante para relacionar correctamente el diseño con el código Java. Además de definir el nombre del objeto podemos asignar los manejadores de eventos. Para aprovechar la generación automática de código desde NetBeans, es recomendable dar nombre a los manejadores en el Scene Builder
- Para asignar una clase Java Controlador debemos de añadirlo en el panel situado la parte inferior de la jerarquía del documento.

Scene Builder

Cómo utilizar SceneBuilder:

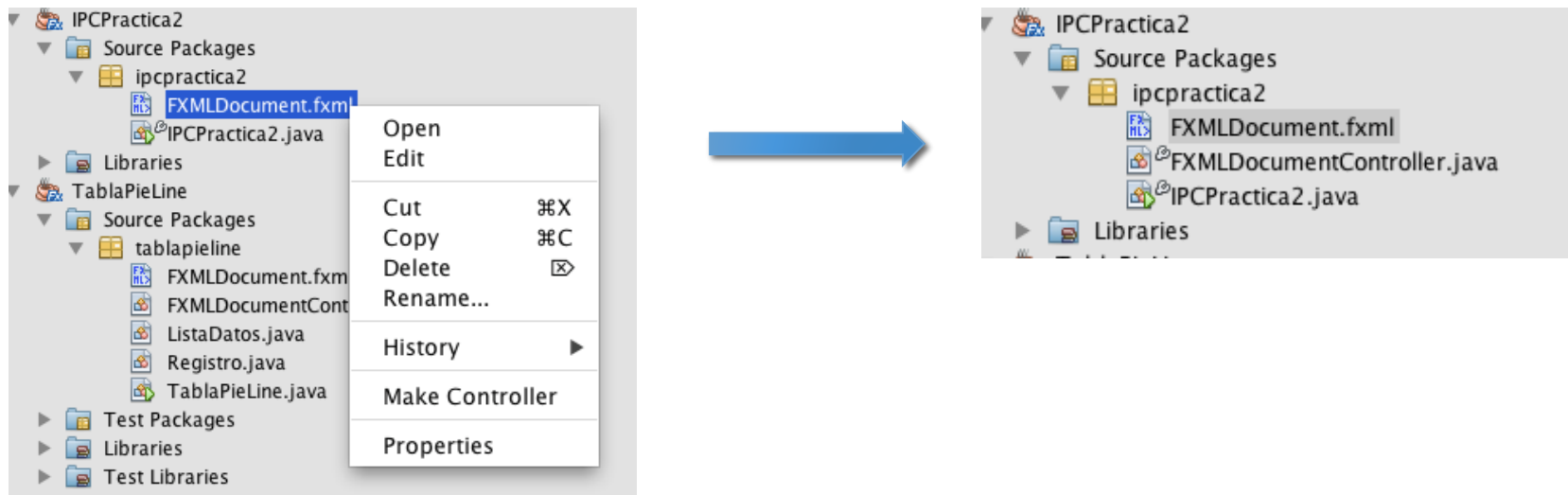


- Con el comando “**Wrap in**” situamos los controles seleccionados dentro de uno de los contenedores disponibles
- El comando “**Unwrap**” elimina el contenedor seleccionado pero deja sus controles inalterados
- Con el comando “**Fit to Parent**” cambiamos el tamaño del control seleccionado hasta que ocupe el área de su contenedor
- El comando “**Use Computed Sizes**” resetea los valores de las propiedades del contenedor a `USE_COMPUTED_SIZE`
- El comando “**Show/Hide Simple Data**” muestra datos ficticios en aquellos controles del tipo lista, tabla o árbol. Los datos no se guardan en el fichero FXML
- El comando “**Show Preview**” muestra en una ventana el resultado final del fichero FXML que se está editando
- El comando “**Show Sample Controller Skeleton**” abre una ventana y muestra una plantilla de código para crear una clase controlador a partir del fichero FXML

NetBeans, clase Controlador

Cómo generar de manera automática la clase controlador:

- Desde el explorador de NetBeans seleccionaremos el fichero FXML y con el botón derecho accederemos al menú **Make Controller**

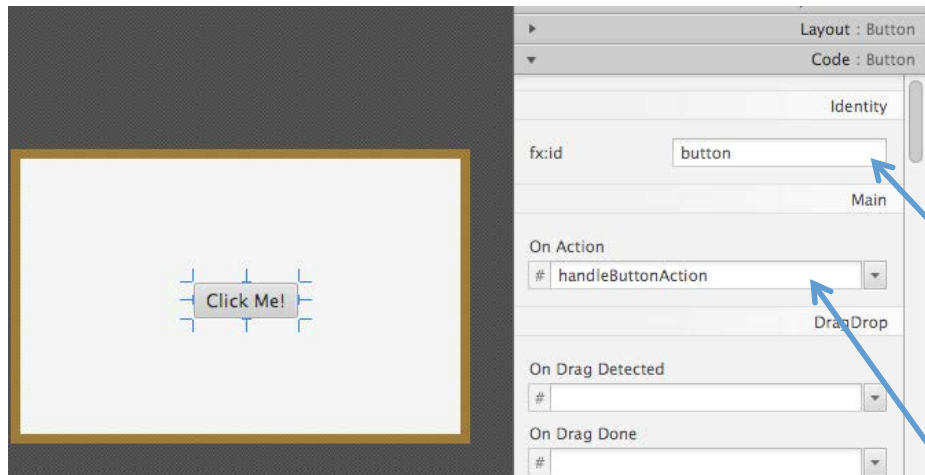


- Si el fichero de la clase Controlador ya existe, éste se actualizara con los datos del fichero XML. En ningún caso borra el código existente.

Netbeans, clase Controlador

Cómo generar de manera automática la clase controlador:

- Además de generar la Clase Controlador con los manejadores y las referencias a los controles, modifica el fichero FXML y le añade la referencia a la clase controlador.



```
package ipcpractica2;

import java.net.URL;
import java.util.ResourceBundle;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.Label;

/**
 * FXML Controller class
 *
 * @author jsoler
 */
public class FXMLDocumentController implements Initializable {

    @FXML
    private Button button;
    @FXML
    private Label label;

    /**
     * Initializes the controller class.
     */
    @Override
    public void initialize(URL url, ResourceBundle rb) {
        // TODO
    }

    @FXML
    private void handleButtonAction(ActionEvent event) {
    }

}
```

Carga de un fichero FXML

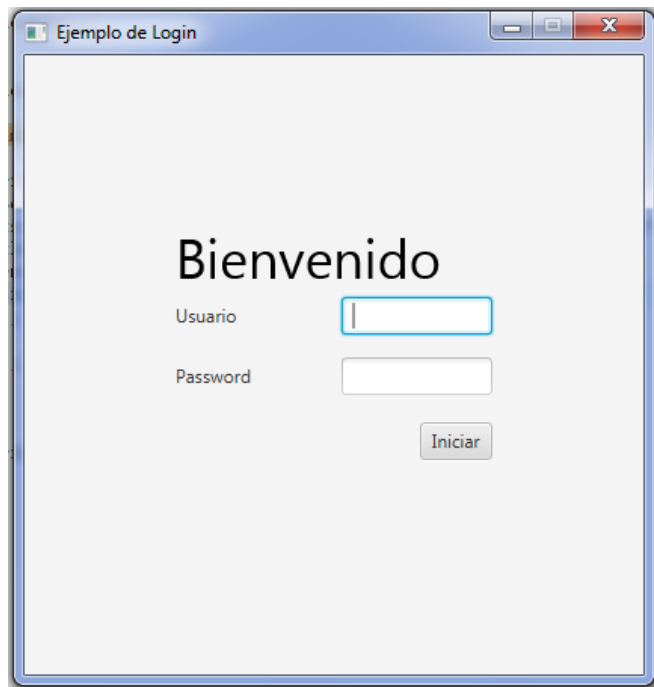
- Un fichero FXML se carga utilizando el método load() de la clase FXMLLoader. :

```
FXMLLoader loader= new FXMLLoader(getClass().getResource("FXMLDocument.fxml"));  
Parent raiz = loader.load();
```

- Durante la carga del fichero se realizan las siguientes tareas:
 1. Se crea un objeto de la clase controladora
 2. Se crean todos los objetos definidos en el fichero FXML, se les asignan manejadores y se construye el grafo de escena
 3. Mediante inyección se enlazan las variables de la clase controladora con los objetos creados en la etapa anterior.
 4. Se ejecuta el método Initialize en el objeto de la clase controladora(si esta definido). Es en este método donde añadiremos la inicialización adicional que necesite nuestra aplicación.

Ejercicio guiado

- Ejemplo de login



The image shows a screenshot of a login window titled "Ejemplo de Login". The window has a light gray background and a blue border. At the top, there is a title bar with the text "Ejemplo de Login" and standard window control buttons (minimize, maximize, close). The main content area displays the word "Bienvenido" in a large, bold, black font. Below this, there are two input fields: one for "Usuario" (User) and one for "Password". The "Usuario" field is highlighted with a blue border. Below the input fields, there is a button labeled "Iniciar" (Start).

Ejercicio propuesto

- Crear un proyecto JavaFX FXML con la siguiente vista



Bibliografía

- Puedes encontrar más información en:
 - <https://openjfx.io/openjfx-docs/>
 - https://openjfx.io/javadoc/17/javafx.fxml/javafx/fxml/doc-files/introduction_to_fxml.html
 - <http://docs.oracle.com/javafx/scenbuilder/1/overview/jsbpub-overview.htm>
 - <https://www.oracle.com/java/technologies/javase/javafxscenbuilder-info.html>
 - <http://code.makery.ch/library/javafx-8-tutorial/es/>
- Documentación online:
 - Java: <https://docs.oracle.com/en/java/javase/11/>
 - JavaFX: <https://openjfx.io/>
- Carl Dea y otros. JavaFX 8.
Introduction by Example. Apress 2014.

