



P1. SCENE BUILDER

Interfaces Persona Computador

Depto. Sistemas Informáticos y Computación

UPV

DOCENCIA VIRTUAL

Finalidad:

Prestación del servicio Público de educación superior (art. 1 LOU)

Responsable:

Universitat Politècnica de València.

Derechos de acceso, rectificación, supresión, portabilidad, limitación u oposición al tratamiento conforme a políticas de privacidad:

<http://www.upv.es/contenidos/DPD/>

Propiedad intelectual:

Uso exclusivo en el entorno de aula virtual.

Queda prohibida la difusión, distribución o divulgación de la grabación de las clases y particularmente su compartición en redes sociales o servicios dedicados a compartir apuntes.

La infracción de esta prohibición puede generar responsabilidad disciplinaria, administrativa o civil



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



Índice

- Conceptos de un framework GUI
- FXML
- SceneBuilder
- NetBeans y la clase Controller
- Ejemplo guiado
- Ejercicio

Conceptos de un Framework GUI

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



Conceptos de un Framework GUI

El hilo de ejecución de un GUI

- El GUI corre sobre un hilo diferente al hilo principal
- Esto se hace así para disponer de una GUI que responda rápidamente a las acciones del usuario
- Hay que separar para ello el código de la Interfaz de usuario del código de la lógica de la aplicación
- El código de usuario puede correr en el hilo de la GUI, pero para grandes bucles o acciones costosas (operaciones de red o bases de datos) es preferible normalmente ejecutar el código en otro hilo

FXML:

- 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.
- Es similar a lo que ocurre con HTML y como se trabaja con Android
- 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

```
<?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:

- 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 contener una clase Controladora (normalmente llamada *Controller*) que actuará como controlador de la interfaz definida en el fichero. Un objeto de esta clase es creado cuando se carga el fichero
- Para enlazar las variables definidas en el controlador con los objetos creados en tiempo de 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 sobre los objetos. De esta manera se reduce considerablemente el código necesario para crear y registrar los nodos de la escena

JavaFX y el patrón MVC :

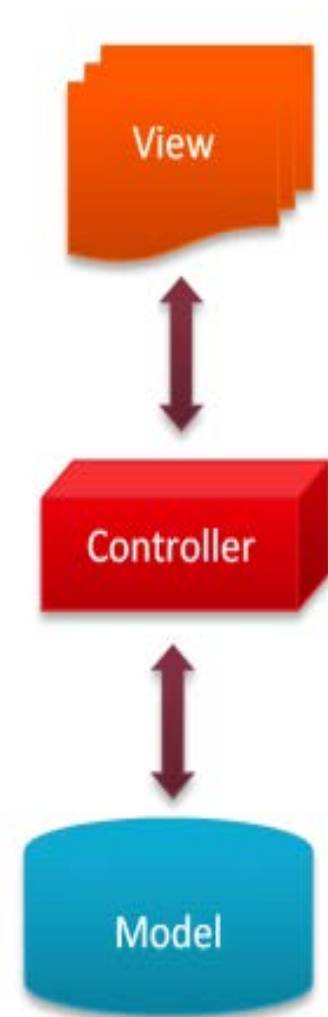
- Un fichero FXML es una descripción de la **vista**

```
<?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) {
    }
}
```

- Clases de Java que definen los objetos de la aplicación ajenos a la interfaz forman el **modelo**.



Arquitectura de JavaFX

- Scene Graph (fichero **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>
```

```
public class HolaFXM extends Application {
```

```
    @Override
```

```
    public void start(Stage stage) throws Exception {
```

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

```
        Scene scene = new Scene(raiz);
```

```
        stage.setScene(scene);
```

```
        stage.show();
```

```
    }
```

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

```
        launch(args);
```

```
    }
```

```
}
```



Arquitectura de JavaFX

- Scene Graph (fichero **FXML**)

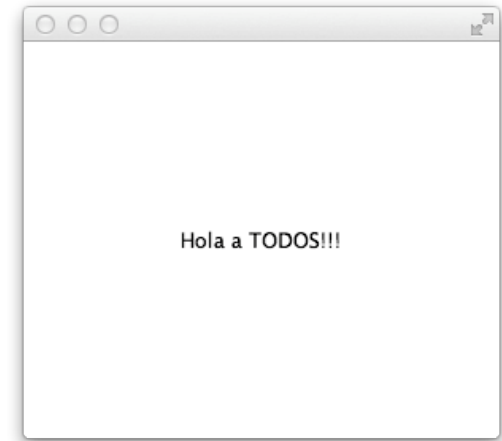
```
public class HolaFXM extends Application {  
  
    @Override  
    public void start(Stage stage) throws Exception {
```

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

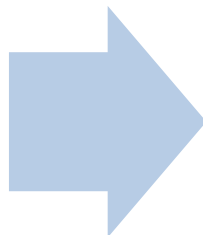
```
        Scene scene = new Scene(raiz);  
        stage.setScene(scene);
```

```
        stage.show();  
    }
```

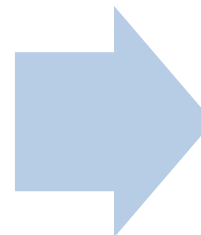
```
    public static void main(String[] args) {  
        launch(args);  
    }
```



Crea los
nodos,
construye el
árbol



Crea la escena
y asigna la
escena a la
ventana



Pinta la
ventana y
cede el
control al
S.O.

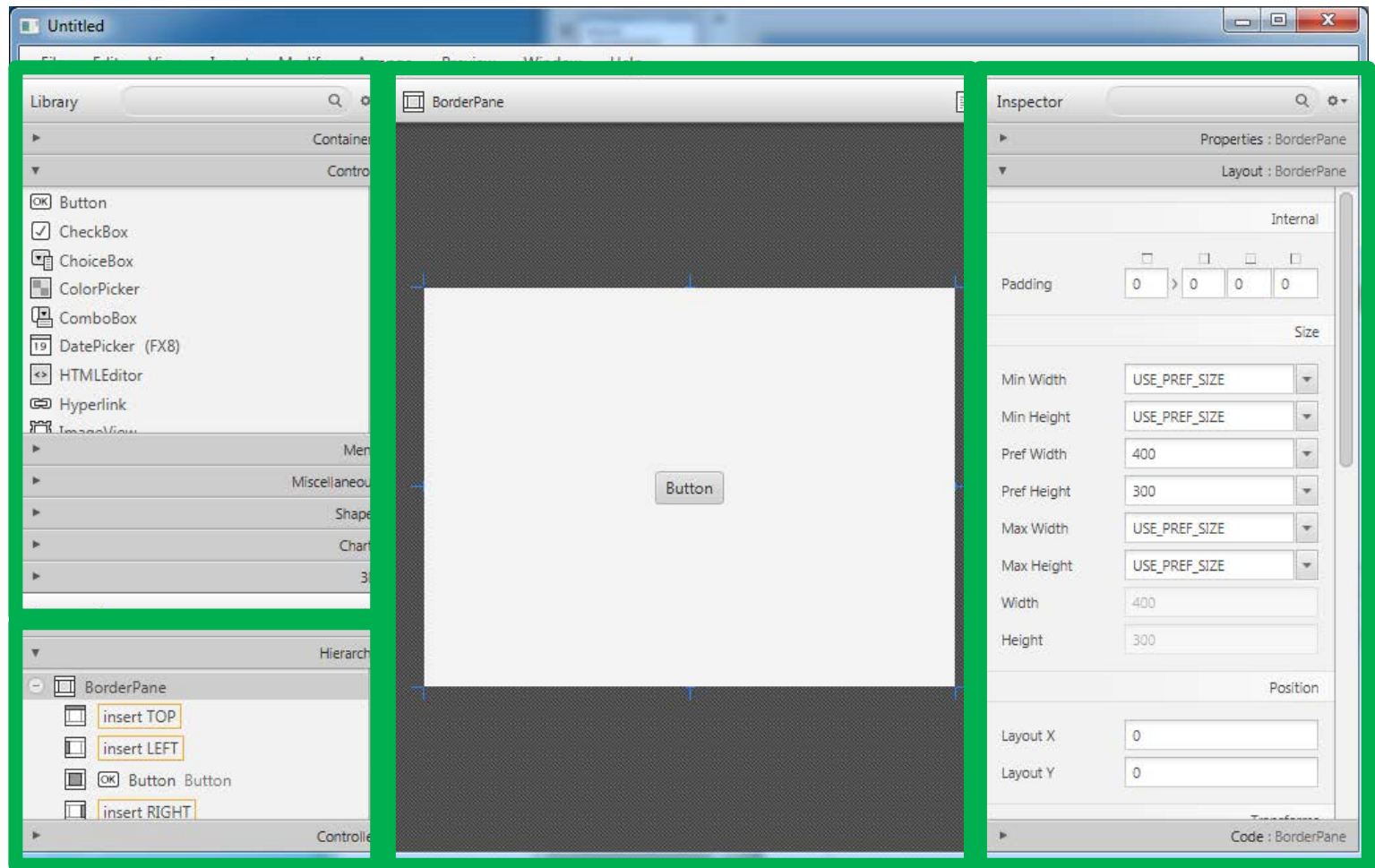
Scene Builder:

- **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



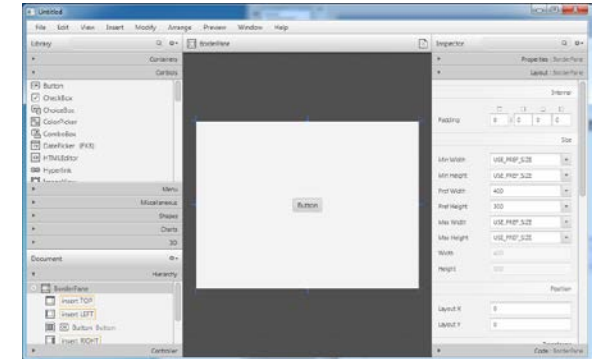
Jerarquía del
documento

Zona de trabajo

Inspector

Scene Builder

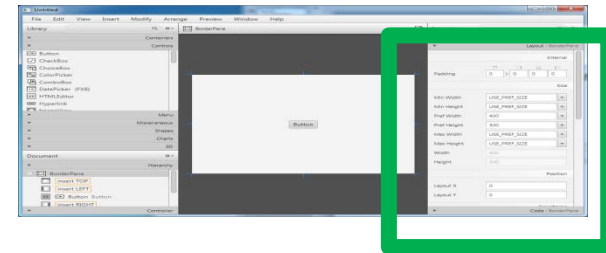
Cómo utilizar Scenebuilder:



- 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

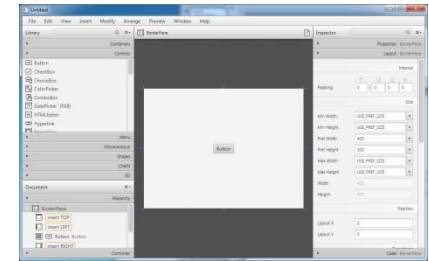
Cómo utilizar Scenebuilder:



- 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:

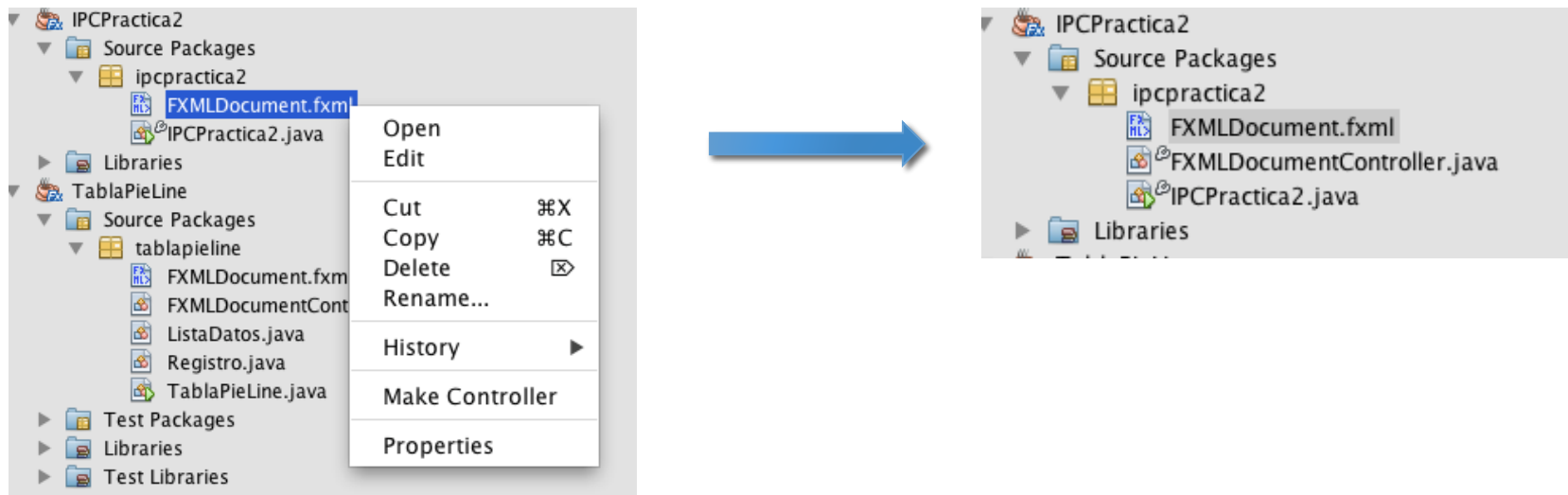


- Con el comando “**Wrap in**” situamos los controles seleccionados dentro de uno de los contenedores disponibles
- El comando “**Unwrap**” elimina el contenedor seleccionado pero deja sus controles inalterados
- Con el comando “**Fit to Parent**” cambiamos el tamaño del control seleccionado hasta que 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**

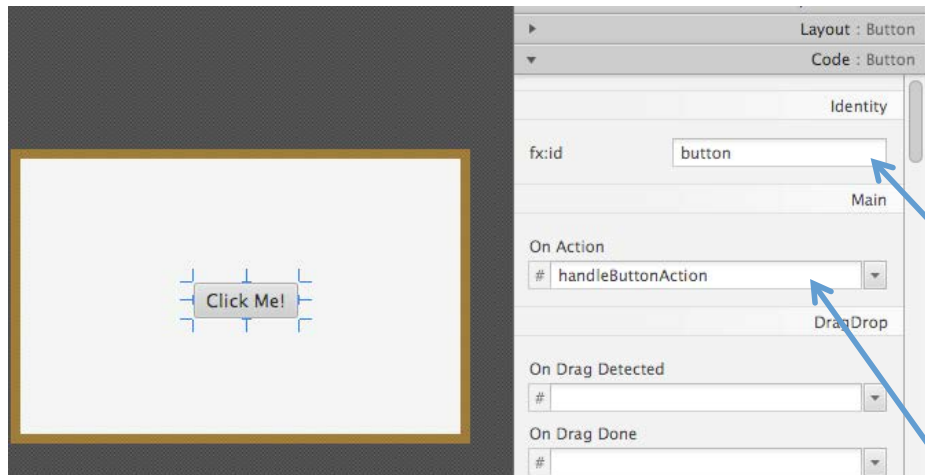


- Si el fichero de la clase Controlador ya existe, éste se actualizara con los datos del fichero XML. En ningún caso borra el código existente.

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`. Hay dos opciones, utilizar el método estático de la clase o crear previamente un instancia de la clase:

```
(opción 1) Parent raiz = FXMLLoader.load(getClass().getResource("FXMLDocument.fxml"));
```

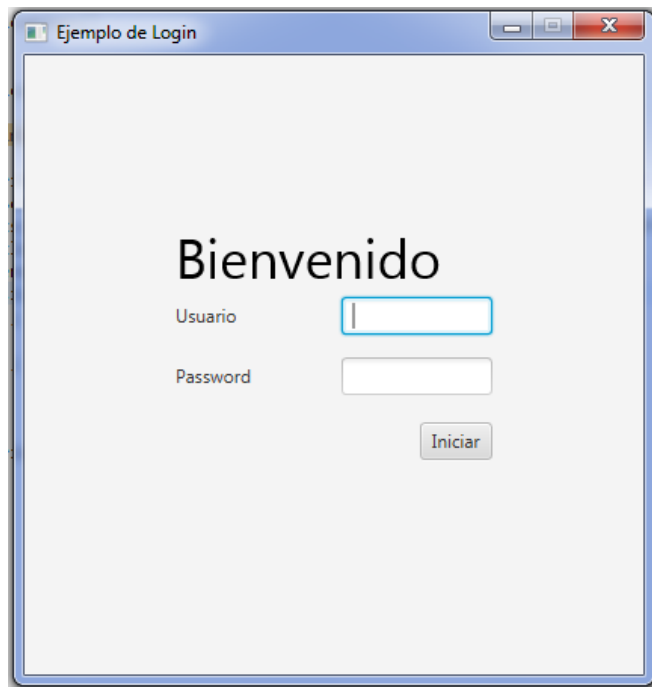
```
(opción 2) FXMLLoader loader= new FXMLLoader(getClass().getResource("FXMLDocument.fxml"));
```

```
(PREFERIBLE) 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 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



Ejemplo de Login

Bienvenido

Usuario

Password

Iniciar

Ejercicio propuesto

- Crear un proyecto JavaFX FXML con la siguiente vista



Bibliografía

- Puedes encontrar más información en:

- <https://docs.oracle.com/javase/8/javase-clienttechnologies.htm>
- https://docs.oracle.com/javase/8/javafx/get-started-tutorial/get_start_apps.htm
- http://docs.oracle.com/javafx/2/api/javafx/fxml/doc-files/introduction_to_fxml.html
- <http://docs.oracle.com/javafx/scenebuilder/1/overview/jsbpub-overview.htm>
- <http://code.makery.ch/library/javafx-8-tutorial/es/>

- Documentación online:

- Java: <http://docs.oracle.com/javase/8/docs/api/>
- JavaFX8: <http://docs.oracle.com/javase/8/javafx/api/>
- JavaFX: <https://openjfx.io/>

- Carl Dea y otros. JavaFX 8.
Introduction by Example. Apress 2014.

