Confección de Interfaces de Usuario a partir de XML

Tema 2. Desarrollo de Interfaces

Objetivos

- Conocer JavaFX
- Diferencias principales con Swing
- Lenguaje FXML
- Aplicación del MVC

JavaFx

JavaFX

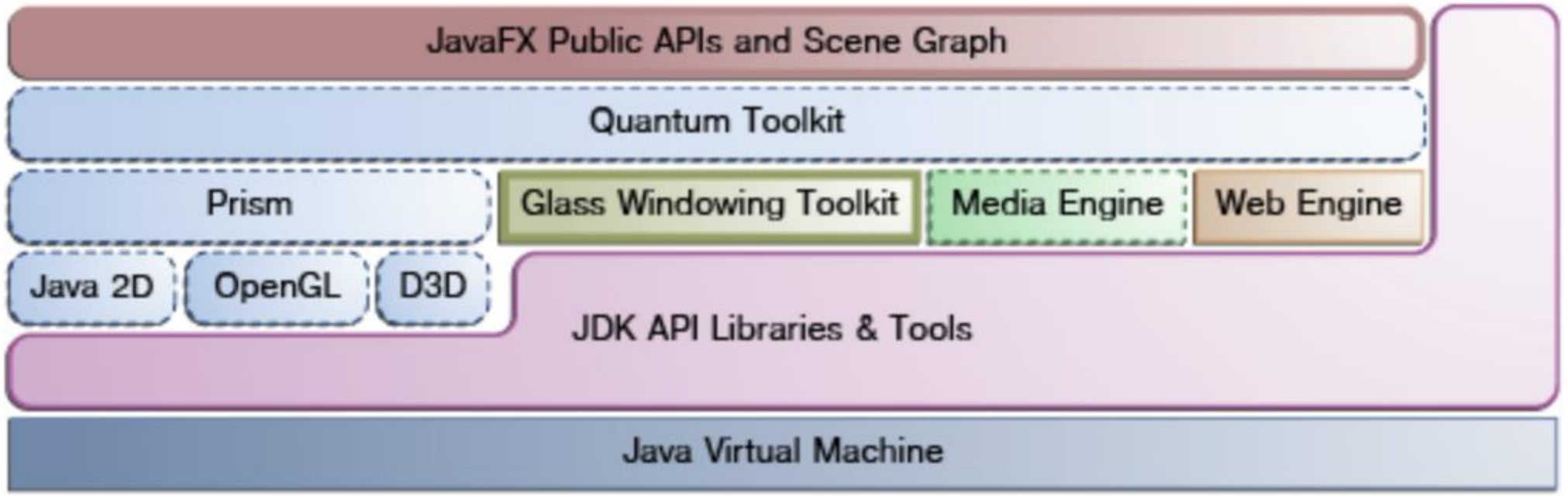
JavaFX es una familia de productos y tecnologías inicialmente de Oracle, para la creación de Rich Internet Applications (RIAs).

Se planteó como alternativa a SilverLight y Flash.

Mejoras respecto a Swing

- Optimizacion de la arquitectura MVC con la separación de las vistas con FXML
- Se permite la incorporación de tecnologías web como el CSS
- Se produce una separación del desarrollo de JavaFX del núcleo de desarrollo del JDK.

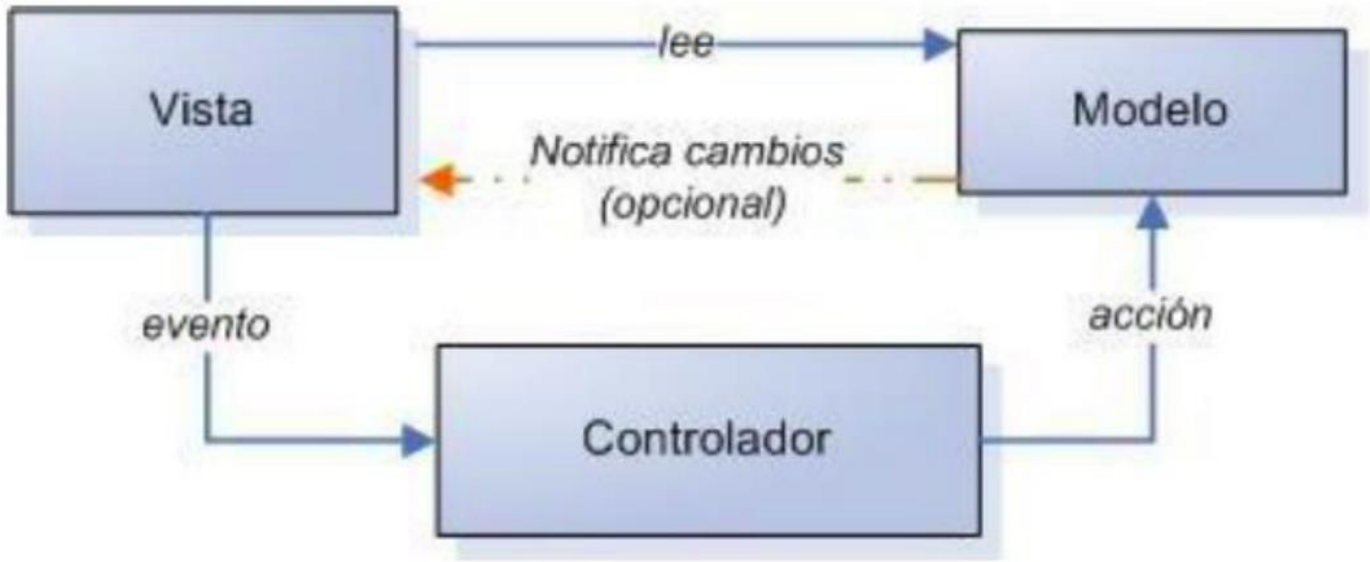
Arquitectura



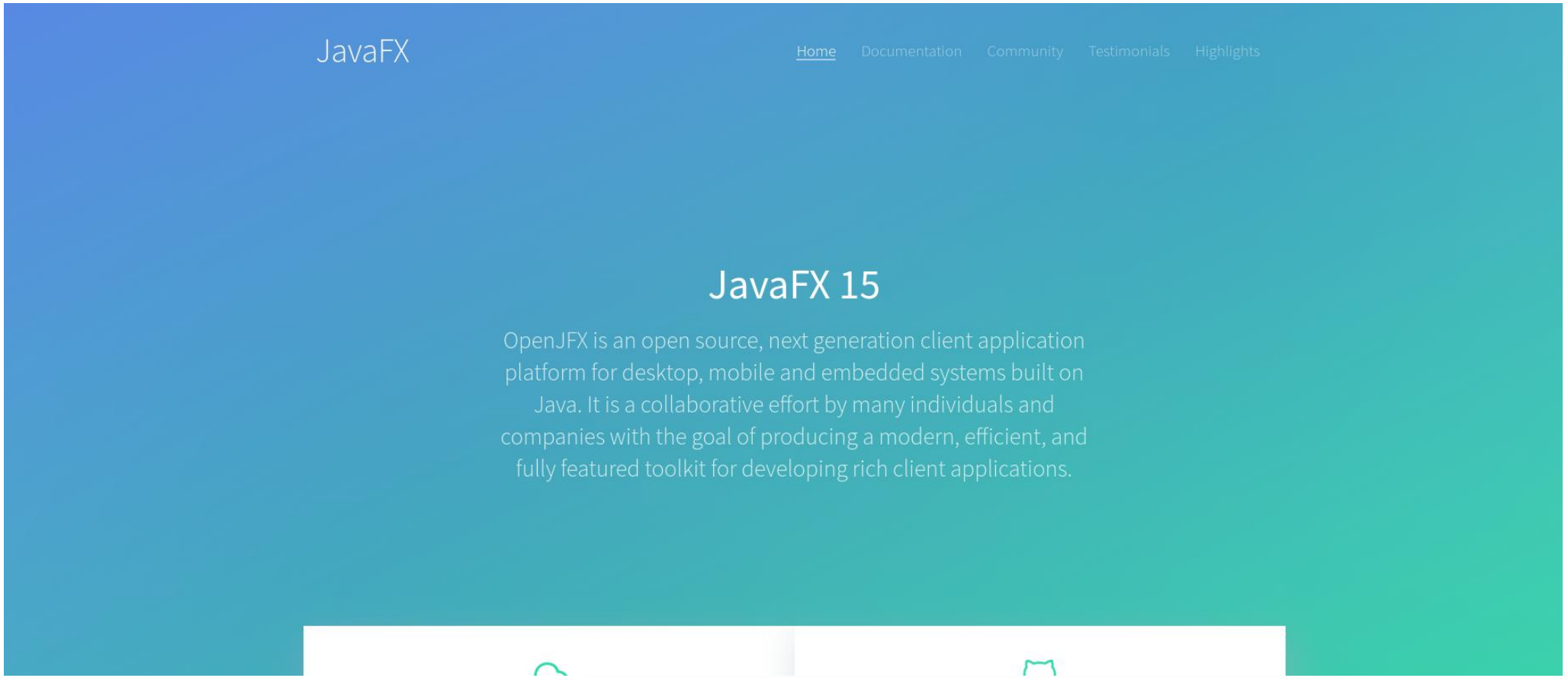
Paquetes JavaFX

- javafx.**base**: define la base del API de JavaFX
- javafx.**controls**: define los controles, gráficos y skins disponibles
- javafx.**fxml**: define el API para el manejo e interpretación de los ficheros FXML.
- javafx.**graphics**: define el entorno de layouts y containers
- javafx.**media**: define el API para la gestión de material audiovisual.
- javafx.**swing**: define la inclusión de Swing dentro de JavaFX.
- javafx.**web**: define el API para contenedores tipo WebView

Mejoras respecto a Swing



OpenJFX


JavaFX
Home
Documentation
Community
Testimonials
Highlights
JavaFX 15
OpenJFX is an open source, next generation client application platform for desktop, mobile and embedded systems built on Java. It is a collaborative effort by many individuals and companies with the goal of producing a modern, efficient, and fully featured toolkit for developing rich client applications.

OpenJFX

JavaFX

Long Term Support

JavaFX 11 is the first long term support release of JavaFX by Gluon. For commercial, long term support of JavaFX 11, please review our JavaFX Long Term Support options.

The JavaFX 11 runtime is available as a platform-specific SDK, as a number of jmods, and as a set of artifacts in maven central.

The OpenJFX page at openjfx.io is a great starting place to learn more about JavaFX 11.

The Release Notes for JavaFX 11 are available in the OpenJFX GitHub repository: Release Notes.

This software is licensed under GPL v2 + Classpath (see http://openjdk.java.net/legal/gplv2+ce.html).

Product	Public version	LTS version	Platform	Download
JavaFX Windows SDK	11.0.2	11.0.8 More info	Windows	Download [SHA256]
JavaFX Windows jmods	11.0.2	11.0.8 More info	Windows	Download [SHA256]
JavaFX Mac OS X SDK	11.0.2	11.0.8 More info	Mac	Download [SHA256]
JavaFX Mac OS X jmods	11.0.2	11.0.8 More info	Mac	Download [SHA256]
JavaFX Linux SDK	11.0.2	11.0.8 More info	Linux	Download [SHA256]
JavaFX Linux jmods	11.0.2	11.0.8 More info	Linux	Download [SHA256]

Usando Maven

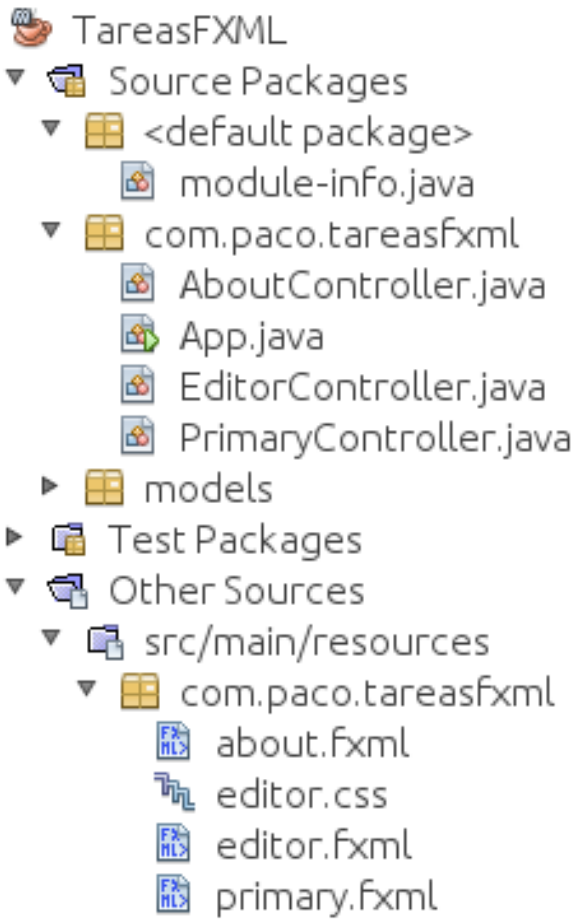
```
<plugins>
    <plugin>
        <groupId>org.openjfx</groupId>
        <artifactId>javafx-maven-plugin</artifactId>
        <version>0.0.4</version>
        <configuration>
            <mainClass>HelloFX</mainClass>
        </configuration>
    </plugin>
</plugins>
```

```
<dependencies>
  <dependency>
    <groupId>org.openjfx</groupId>
    <artifactId>javafx-controls</artifactId>
    <version>14</version>
  </dependency>
</dependencies>
```

Opciones de la maquina JVM

- --module-path="/opt/javafx/lib"
- --add-modules=javafx.controls,javafx.fxml

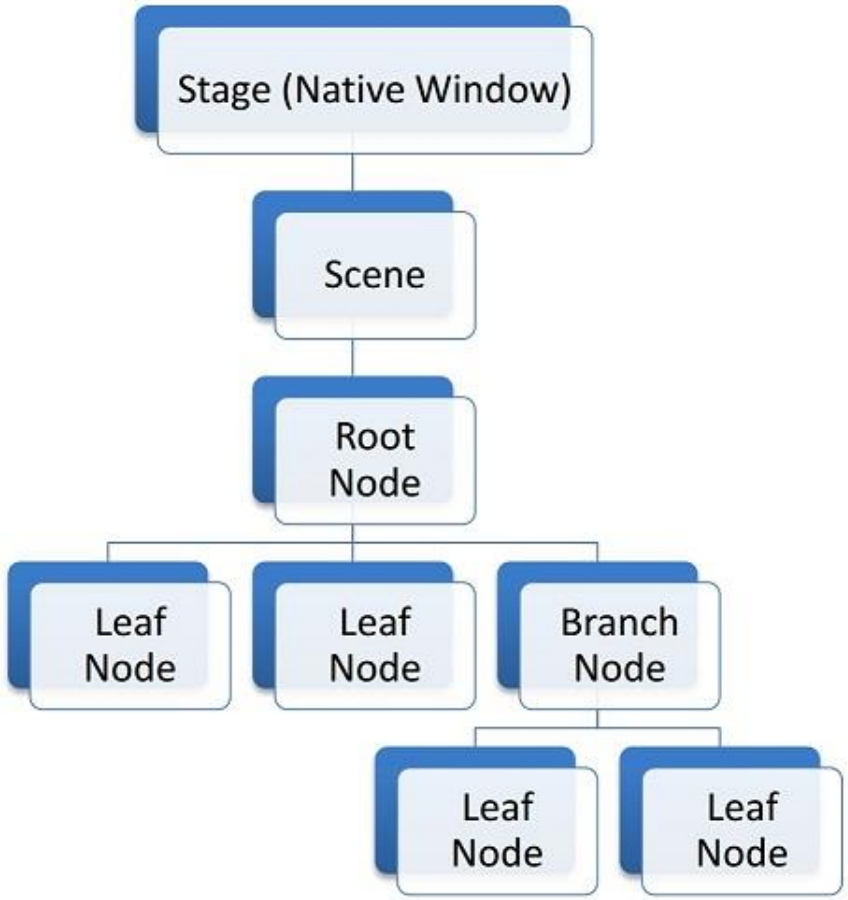
Estructura de proyecto MVC



Clases básicas

- **Clase Stage**, en javafx.stage que es el contenedor padre y sobre el cual "mostraremos" la/s escenas. Equivale a la clase JFrame de Swing.
- **Clase Scene**, en javafx.scene y donde incluiremos el resto de controles.

Clases básicas



Clases básicas

- Primero se crean las escenas y luego se añaden al "escenario"

```
Scene scene = new Scene(root, 800, 600);
primaryStage.setTitle("Mi primera aplicación");
primaryStage.setScene(scene);
primaryStage.show();
```

Clases básicas

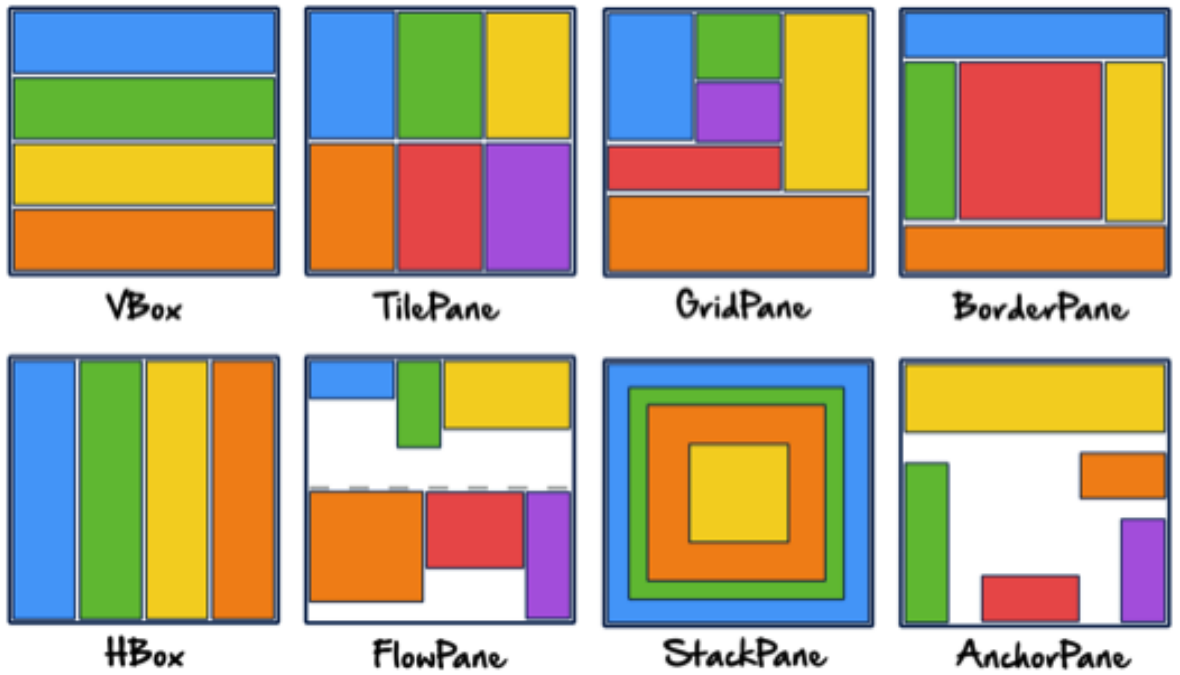
- Primero se crean las escenas y luego se añaden al "escenario"

```
Scene scene = new Scene(root, 800, 600);
primaryStage.setTitle("Mi primera aplicación");
primaryStage.setScene(scene);
primaryStage.show();
```

Layouts

Pane: es la clase base para el resto de clases.

- BorderPane
- FlowPane
- HBox
- VBox
- GridPane
- StackPane
- TilePane
- AnchorPane



Javafx.shape

- Line: representa una línea con coordenadas x, y.
- Rectangle: define un rectángulo a partir de segmentos.
- Circle: crea un nuevo círculo 2D
- Ellipse: crea una nueva elipse 2D
- Polygon: crea un nuevo polígono definido como un array de coordenadas x, y.

Javafx.controls

- Button
- CheckBox
- ColorPicker
- Label
- Menu, MenuBar, MenuItem
- TextArea
- List

Hello World en JavaFX

XML para la creación de interfaces

Ventajas

El uso de XML como lenguaje base para la descripción de nuestros interfaces nos permite:

- Tener un código independiente de la plataforma final.
- Tener un código independiente de la tecnología de desarrollo.
- Tener un código que se puede separar de la lógica y de los datos.

Formatos para GUIs

- XAML (eXtensible Application Markup Language)
- XUL (XML User Interface Language)
- UIML (User Interface Markup Language)
- SVG (Scalable Vector Graphics)
- MXML
- FXML

XAML

- Permite la creación de elementos visibles de la interfaz de usuario.
- Permite asociar un archivo de código distinto para cada archivo XAML que puede responder a eventos.
- Admite el intercambio de herramientas de diseño y un IDE, o entre los desarrolladores.

XAML

```
<Window x:Class = "HelloWorld.MainWindow"
   xmlns = "http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x = "http://schemas.microsoft.com/winfx/2006/xaml"
   Title = "MainWindow" Height = "350" Width = "604">

   <Grid>
      <TextBlock x:Name = "textBlock" HorizontalAlignment = "Left"
         Margin = "235,143,0,0" TextWrapping = "Wrap" Text = "Hello World!"
         VerticalAlignment = "Top" Height = "44" Width = "102" />
   </Grid>

</Window>
```

XUL

Permite crear una interfaz usando un lenguaje de marcado, definir la apariencia de esta interfaz con hojas de estilo CSS y usar JavaScript para manipular su comportamiento.

Tiene un conjunto extenso de componentes gráficos usados para crear menús, barras de herramientas, cajas de texto y otros componentes.

XUL

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>

<window id="vbox example" title="Example 3...."
xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
  <vbox>
    <button id="yes1" label="Yes"/>
    <button id="no1" label="No"/>
    <button id="maybe1" label="Maybe"/>
  </vbox>
</window>
```

SVG

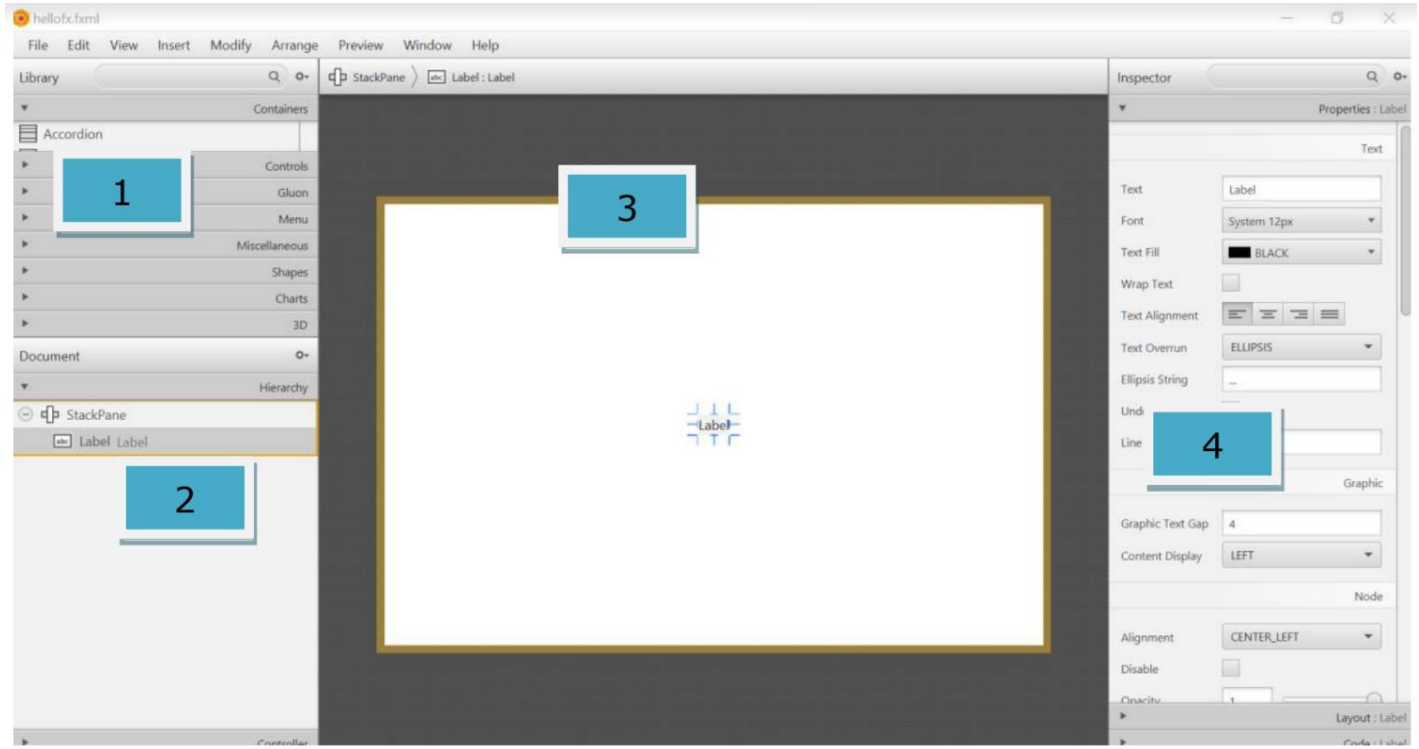
Permite crear dinámicamente gráficos de alta calidad en tiempo real con precisión estructural y control visual.

Permite crear gráficos interactivos y de este modo generar aplicaciones con lenguajes de scripting comunes en la web tales como JavaScript, Java o Microsoft Visual Basic.

Permite personalizar los gráficos, por ejemplo, contiene tipos de fuentes no romanas.

SceneBuilder

1. Libreria de componentes
2. Arbol del documento
3. Escena actual
4. Inspector propiedades y eventos



FXML

Dentro de FXML, un elemento XML representa los siguientes casos:

- Una instancia de una clase
- Una propiedad dentro de una instancia de una clase
- Una propiedad estática
- Un bloque de definición (traducción de “define” block).
- Un bloque de código

Ejemplo FXML

```
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<VBox xmlns:fx="http://javafx.com/fxml">
    <children>
        <Label text="Hello, World!"/>
        <fx:include source="my_button.fxml"/>
    </children>
</VBox>
```

Ejemplo Contenedor FXML

```
<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-
Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0">
        <center>
            <Label text="Etiqueta centrada"/>
        </center>
        <north>
            <Label text="Etiqueta arriba"/>
        </north>
</BorderPane>
```

Ejemplo rectángulo FXML

```
<?import javafx.scene.*?>
<?import javafx.scene.shape.*?>

<Group xmlns:fx="http://javafx.com/fxml">
    <children>
        <Rectangle fx:id="rectangle" x="10" y="10" width="320"
        height="240" fill="#ff0000"/>
        <Button text="Pulsame"></Button>
    </children>
</Group>
```

Cargando una vista FXML

```
public class App extends Application {

    private static Scene scene;

    @Override
    public void start(Stage stage) throws IOException {
        FXMLLoader fxmlLoader = new FXMLLoader(App.class.getResource("primary.fxml"));
        scene = new Scene(fxmlLoader.load());
        stage.setScene(scene);
        stage.setFullScreen(true);
        stage.show();
    }

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

}
```

Saludador con SceneBuilder

Controladores

```
public class SecondaryController {

    @FXML
    private TextField txtNombre;
    @FXML
    private Button btnSaludar;
    @FXML
    private Button secondaryButton;

    @FXML
    private void switchToPrimary() throws IOException {
        App.setRoot("primary");
    }

    @FXML
    private void saludar(ActionEvent event) {
        String nombre = txtNombre.getText();
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setHeaderText(nombre);
        alert.setTitle(nombre);
        alert.setContentText("Hola "+nombre);
        alert.showAndWait();
    }
}
```

Archivo FXML

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

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

<VBox alignment="CENTER" prefHeight="336.0" prefWidth="209.0" spacing="20.0" xmlns="http:/
    <children>
      <Label text="Nombre" />
      <TextField fx:id="txtNombre" text="nombre" />
      <Button fx:id="btnSaludar" mnemonicParsing="false" onAction="#saludar" text="Saludar
        <Label text="Secondary View" />
        <Button fx:id="secondaryButton" onAction="#switchToPrimary" text="Switch to Primar
    </children>
    <padding>
        <Insets bottom="20.0" left="20.0" right="20.0" top="20.0" />
    </padding>
</VBox>
```

Código en FXML

```
<?language javascript?>
...
<VBox>
    <children>
        <Button text="Click Me!"
        onAction="java.lang.System.out.println('You clicked me!');"/>
    </children>
</VBox>
```

Código en FXML

```
<?language javascript?>
...
<VBox>
    <children>
        <Button text="Click Me!"
        onAction="#clickMe"/>
    </children>
</VBox>
```

```
@FXML
private void clickMe(ActionEvent event) {
    String nombre = txtNombre.getText();
    Alert alert = new Alert(Alert.AlertType.INFORMATION);
    alert.setHeaderText(nombre);
    alert.setTitle(nombre);
    alert.setContentText("Hola "+nombre);
    alert.showAndWait();
}
```

ImageView

```
<ImageView fitHeight="102.0" fitWidth="107.0" preserveRatio="true">
  <image>
    <Image url="@usuario.png" />
  </image>
</ImageView>
```

Propiedades de solo lectura

```
<HBox alignment="CENTER" spacing="10.0">
  <children>
    <Button fx:id="primaryButton" defaultButton="true"text="Aceptar" />
    <Button cancelButton="true" mnemonicParsing="false" text="Cancelar" />
  </children>
</HBox>
```

Propiedades estáticas

```
<GridPane alignment="TOP_CENTER" hgap="15.0" vgap="15.0">
  <children>
    <Label text="Usuario" GridPane.halignment="RIGHT"></Label>
    <TextField GridPane.columnIndex="1" />
    <Label text="Contraseña" textAlignment="RIGHT"
     GridPane.halignment="RIGHT" GridPane.rowIndex="1"></Label>
    <TextField GridPane.columnIndex="1" GridPane.rowIndex="1" />
  </children>
</GridPane>
```

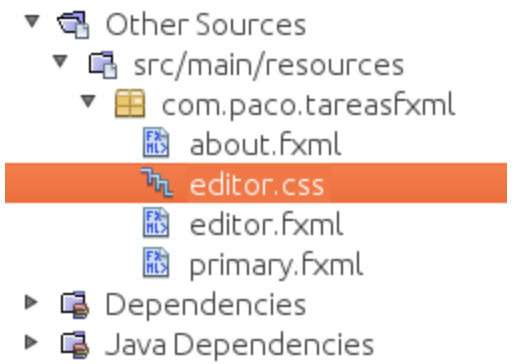
Parseo del FXML

```
<GridPane>
      <children>
            <Label text="My Label">
                  <GridPane.rowIndex>0</GridPane.rowIndex>
                  <GridPane.columnIndex>0</GridPane.columnIndex>
            </Label>
      </children>
</TabPane>
```

```
GridPane gridPane = new GridPane();
Label label = new Label();
label.setText("My Label");
GridPane.setRowIndex(label, 0);
GridPane.setColumnIndex(label, 0);
gridPane.getChildren().add(label);
```

CSS en FXML

```
<AnchorPane id="AnchorPane" prefHeight="206.0" prefWidth="245.0" styleClass="mainF
    <stylesheets>
        <URL value="@editor.css" />
    </stylesheets>
   <children>
      <TextField fx:id="txtNombre" layoutX="33.0" layoutY="26.0" styleClass="textF
      <TextField fx:id="txtResponsable" layoutX="33.0" layoutY="63.0" styleClass=
```




```
.mainFxmlClass {
    -fx-background-color:gray;

}

.textField{
    -fx-font-size:15px;
}

.SplitMenuButton{
    -fx-padding:20px;
    -fx-font-family: "Courier";
}
```

CSS en FXML

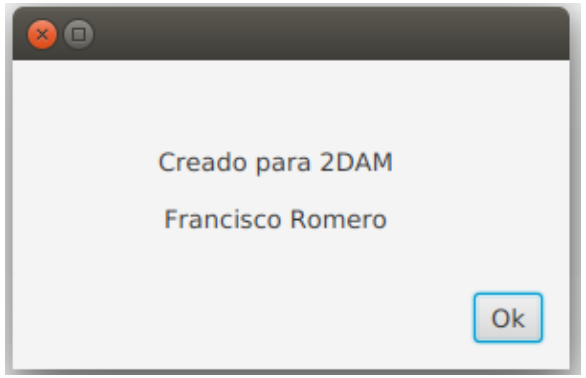
https://docs.oracle.com/javase/8/javafx/api/javafx/scene/doc-files/cssref.html

Ejercicio de diseño

Ejercicio de diseño


```
<StackPane xmlns="http://javafx.com/javafx/11.0.1"
xmlns:fx="http://javafx.com/fxml/1"
fx:controller="com.paco.ejemplofxml.PrimaryController">
  <children>
    <VBox alignment="CENTER" spacing="20.0">
      <children>
        ...
      <Label><ImageView><TextField><Button>
      <Hyperlink ><CheckBox>
        ...
      </children>
    </VBox>
  </children>
</StackPane>
```

Diseño de interfaz



Abrir y cerrar ventanas

```
    // Cargo la ventana
    Parent root = App.loadFXML("about");

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

```
@FXML
private void salir(ActionEvent event) {
    tarea_actual=null;
    Stage stage = (Stage) txtNombre.getScene().getWindow();
    stage.close();
}
```

Alertas

```
Alert alert = new Alert(Alert.AlertType.ERROR);
alert.setHeaderText(null);
alert.setTitle("Error");
alert.setContentText("La tarea ya existe");
alert.showAndWait();
```

```
Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
alert.setHeaderText(null);
alert.setTitle("Borrando...");
alert.setContentText("¿Desea borrar?");

ButtonType okButton = new ButtonType("Si", ButtonBar.ButtonData.YES);
ButtonType noButton = new ButtonType("No", ButtonBar.ButtonData.NO);
alert.getButtonTypes().setAll(okButton, noButton);
var result = alert.showAndWait();
if(result.get()==okButton){
        /* ... */
}
```

Inicializar listas

- Se gestionan a través de OservableList desde el controlador
- Se suele hacer desde el método initialize

```
personas = FXCollections.observableArrayList();
comboPersona.setItems(personas);
personas.addAll("Paco","Juan","Pepe");
```

Obtener el controlador de una vista

- FXMLLoader fxl = ReadFXML("primary");
- PrimaryController controlador = fxl.getController();

De esta manera podemos acceder a métodos del controlador y pasarle parámetros.

Tablas en JavaFX

- Se gestionan a traves de OservableList desde el controlador

```
/**
 * Initializes the controller class.
 */
@Override
public void initialize(URL url, ResourceBundle rb) {
    listado = FXCollections.observableArrayList();
    this.tabla.setItems(listado);

    colPersona.setCellValueFactory( new PropertyValueFactory("Responsable") );
    colNombre.setCellValueFactory( new PropertyValueFactory("Nombre") );
    colPrioridad.setCellValueFactory( new PropertyValueFactory("Prioridad") );
}
```

Tablas en JavaFX

- Se gestionan a traves de OservableList desde el controlador

```
@Override
public void initialize(URL url,ResourceBundle rb) {
    listado = FXCollections.observableArrayList();
    tabla.setItems(listado);

    colCiclo.setCellValueFactory( (var cellData) ->{
        Pedido p = cellData.getValue();
        return new SimpleStringProperty(p.getCiclo());
    });
```

Tablas en JavaFX

- El formato de las celdas se hace desde setCellFactory()

```
colEstado.setCellFactory(column -> {
    return new TableCell<Pedido, String>() {
        @Override
        protected void updateItem(String estado, boolean empty) {
            super.updateItem(estado, empty);



        }
    };
});
```

Toggle Groups

- Se gestionan en el método initialize, donde se asignan como propiedades de cada uno de los elementos del grupo.

```
@Override
public void initialize(URL url, ResourceBundle rb) {
    ToggleGroup tg = new ToggleGroup();
    this.rdbSuma.setToggleGroup(tg);
    this.rdbResta.setToggleGroup(tg);
    this.rdbMult.setToggleGroup(tg);
    this.rdbDivision.setToggleGroup(tg);
}
```

Pasar datos entre ventanas

- Se hace a nivel de controlador. Hay que crear un método de manera que se encapsulen los datos que se van a pasar.

```
FXMLLoader fxmlLoader = App.readFXML("editor");
Parent root = fxmlLoader.load();

EditorController editor = fxmlLoader.getController();
editor.setTarea(actual,listado);


public void setTarea(Tarea actual, ObservableList<Tarea> listatareas) {
    tarea_actual = actual;
    this.tareas = listatareas;

    txtNombre.setText(tarea_actual.getNombre());
    txtResponsable.setText(tarea_actual.getResponsable());
    txtPrioridad.setText(tarea_actual.getPrioridad()+"");
}
```

WebEngine y WebViews

- Para usar estos elementos, hay que incluir el módulo javafx-web en el archivo POM.xml y en el archivo module-info (solo si usamos MAVEN)

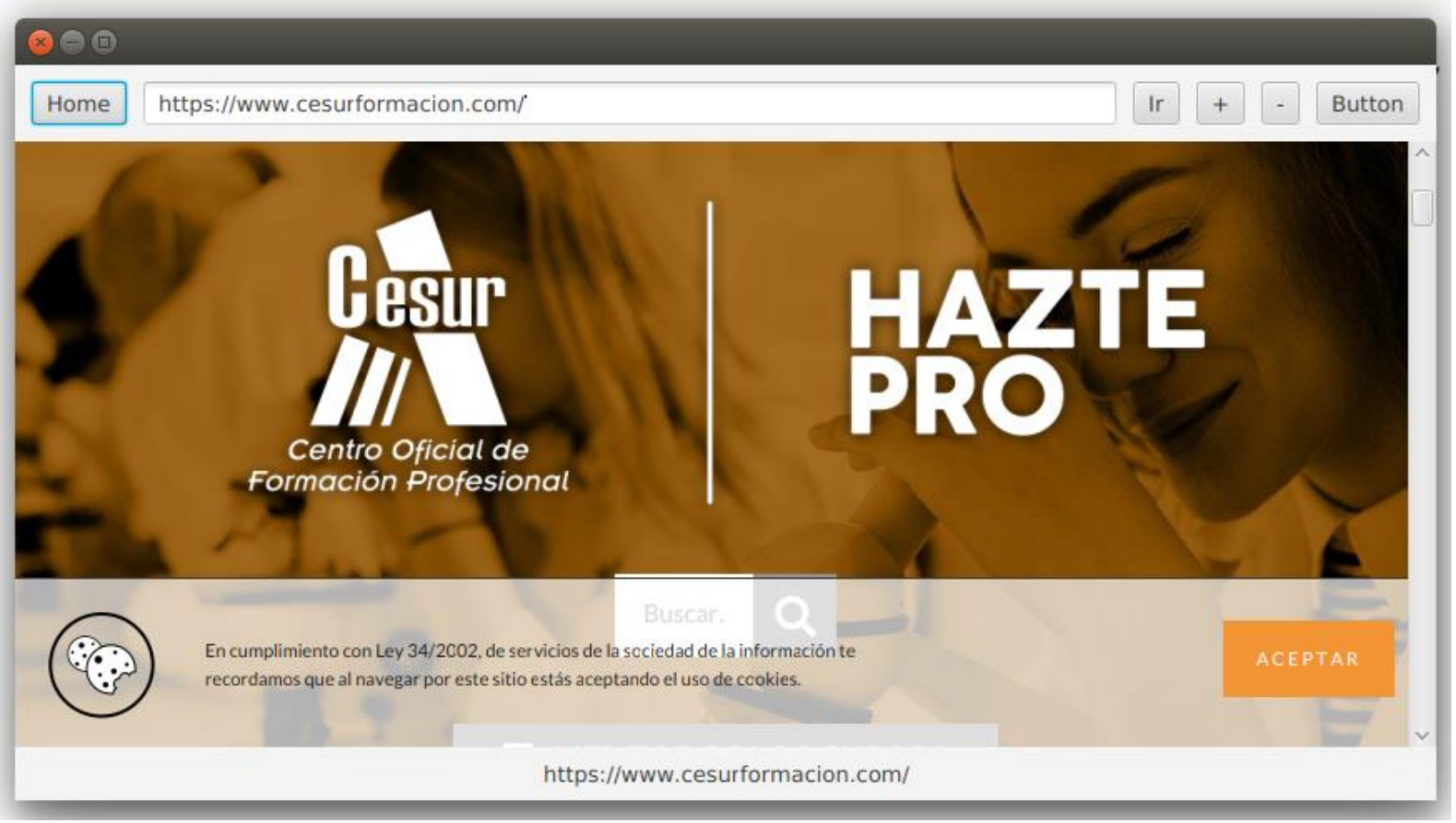
```xml
<dependency>
    <groupId>org.openjfx</groupId>
    <artifactId>javafx-web</artifactId>
    <version>11</version>
    <type>jar</type>
</dependency>
```

```
module com.paco.webbrowserjavafx {
    requires javafx.controls;
    requires javafx.fxml;
    requires javafx.web;

    opens com.paco.webbrowserjavafx to javafx.fxml;
    exports com.paco.webbrowserjavafx;
    requires javafx.webEmpty;
}
```

```
import javafx.scene.web.WebView;
import javafx.scene.web.WebEngine;
```

WebEngine y WebViews



Observar cambios en propiedades

- ChangeListener<T> : lo que hacemos es crear instancias de ChangeListener y añadirlas a las propiedades de los objetos que nos interesen.
- Es una clase con métodos abstractos que dependiendo de la propiedad hay que implementar al crearlos.

```
@Override
public void changed(ObservableValue<? extends String> observable,
                    String oldValue,
                    String newValue){
    /* aqui el valor de la propiedad de tipo String ha cambiado */
}
```

Observar cambios en propiedades

```
ChangeListener<String> listener= new ChangeListener<String>(){
    @Override
    public void changed(ObservableValue<? extends String> observable,
                        String oldValue,
                        String newValue){
        /* aqui el valor de la propiedad de tipo String ha cambiado */
        lblUrl.setText(newValue);
        url.setText(newValue);
    }
};
webEngine.locationProperty().addListener(listener);
```

Programar tareas

- Podemos hacerlo con Timer, pero asegurándonos que se ejecute en el hilo de JavaFx

```
Timer timer = new Timer();
TimerTask task = new TimerTask(){
    @Override
    public void run() {
        Platform.runLater(new Runnable() {
            public void run() {
                /*  TO DO   */
            }
        });

    }

};
timer.scheduleAtFixedRate(task, 0, 5000);
```

Crear un frontend del programa de gestión de desayunos visto en Acceso a Datos

Pedidos para hoy

Ciclo	Alumno	Pedido		Precio	Estado
		Tipo	Producto		
2DAM	Juan	pitufo	Mixto	1.9€	ENTREGADO
2DAM	Juan	pitufo	Mixto	1.9€	ENTREGADO
2DAM	Juan	pitufo	Mixto	1.9€	ENTREGADO