

Interfaces gráficas de usuario (gui) con uso de JavaFX

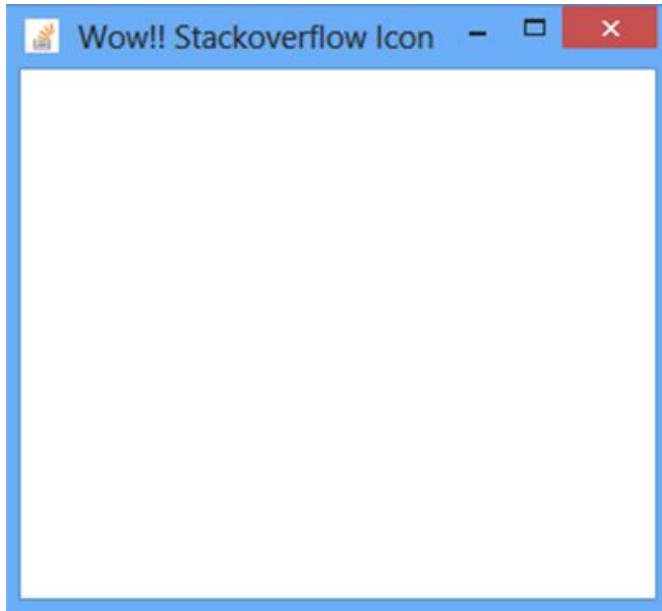
¿Qué es JavaFX?

JavaFX es una plataforma de desarrollo de interfaces de usuario (UI) que se utiliza comúnmente para crear aplicaciones de escritorio con una interfaz gráfica de usuario (GUI) en Java.

En JavaFX, los componentes gráficos se denominan "**nodos**" y se utilizan para construir la interfaz de usuario de la aplicación.



Componentes gráficos: Stage (Escenario)



El escenario es una ventana que consta de todos los objetos de la aplicación JavaFX.

Es un objeto de la clase **Stage** de java

Tiene dos parámetros de creación: anchura y altura, estos determinan su posición.

Se divide en una Barra de título y bordes, es decir, Área de contenido y Decoraciones.

Componentes gráficos: Scene (Escena)

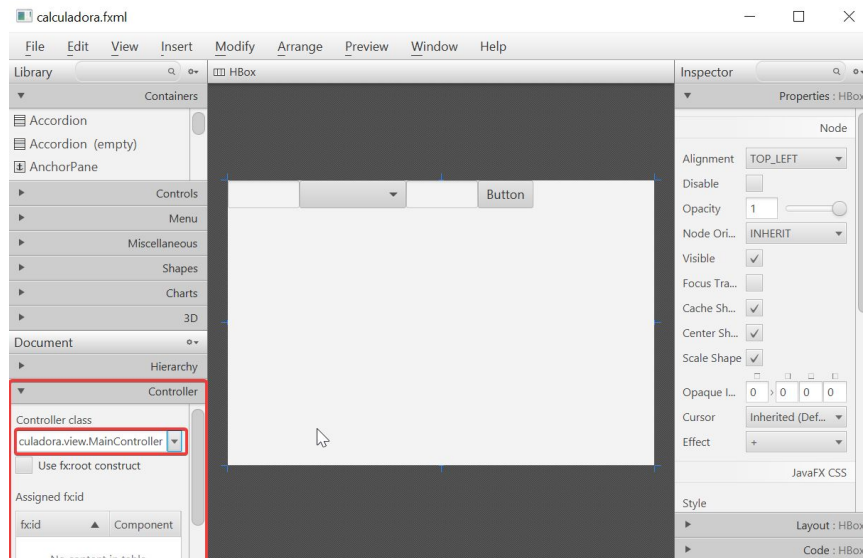
Escena (Scene): La escena es el contenedor principal que se utiliza para organizar todos los nodos en la interfaz de usuario.

Cada ventana de la aplicación tiene al menos una escena.

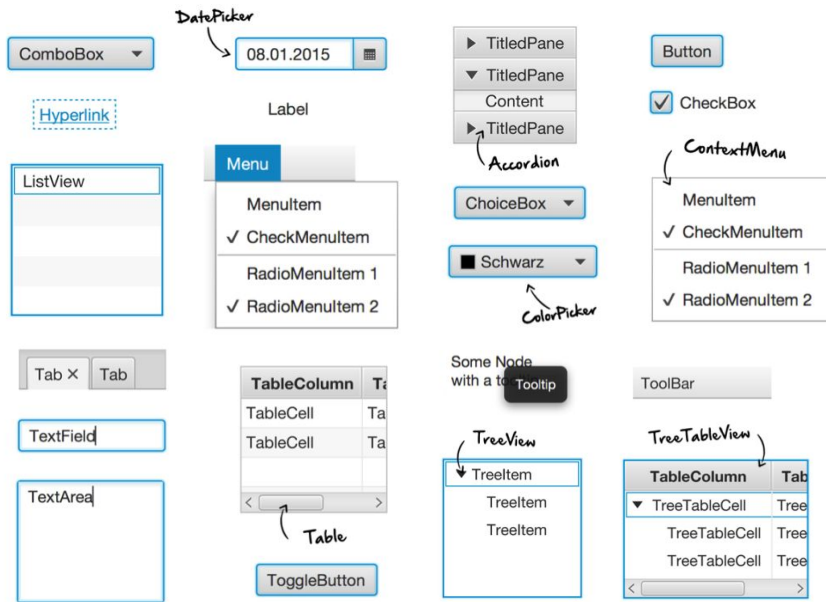
Una escena se crea por medio de un objeto de la clase **Scene**, que toma como argumentos el nodo raíz de la escena y las dimensiones de la escena.

Luego, se asigna esta escena en una ventana (**Stage**) específica utilizando el método **setScene()**.

Las escenas se diseñan con Scene Builder generando archivos .fxml



Componentes gráficos: Node (Nodo)



Los nodos son objetos gráficos que se utilizan para construir la interfaz de usuario de una aplicación.

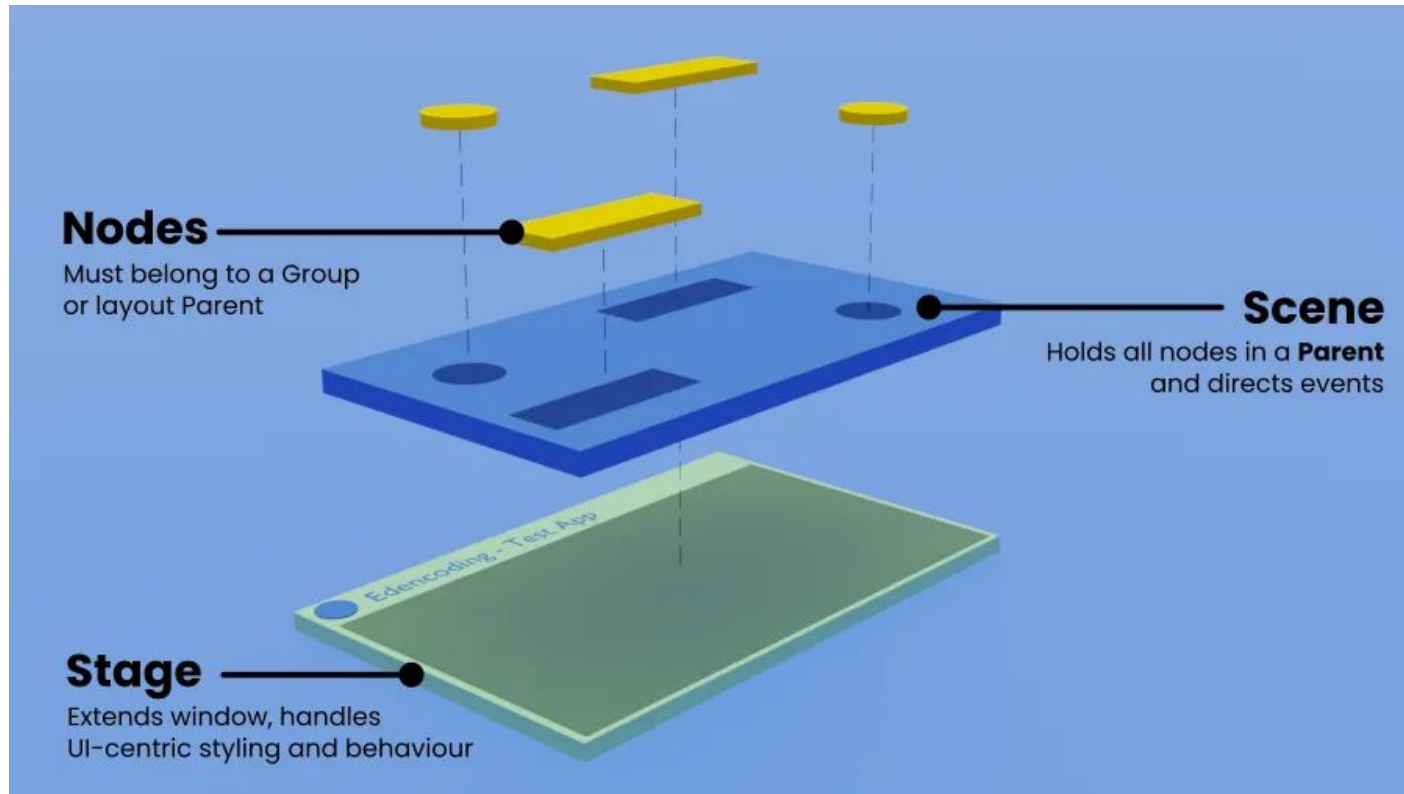
Se organizan en una jerarquía de nodos para formar la estructura visual de la aplicación.

Controles de interfaz de usuario como áreas de texto, casillas de verificación, botones, cuadros de selección, etc.

Objetos geométricos (gráficos) 2D y 3D como polígonos, círculos, rectángulos, etc.

Varios elementos multimedia, como objetos de imagen, vídeo y audio.

Paneles de diseño o contenedores como panel de flujo, panel de bordes, panel de cuadrícula, etc.



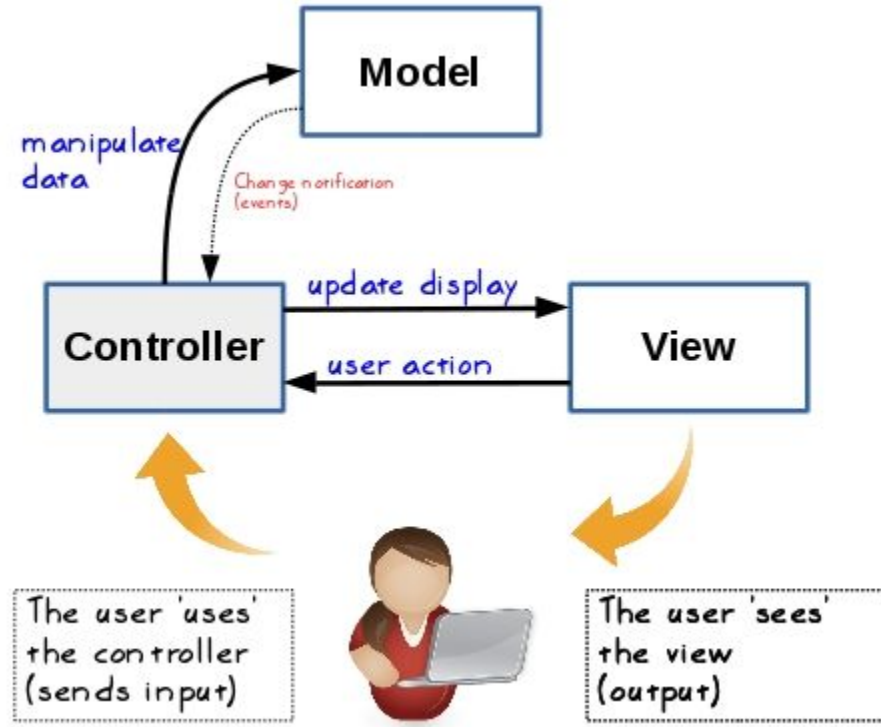
Componentes gráficos: Jerarquía

Patrones de diseño para las interfaces gráficas

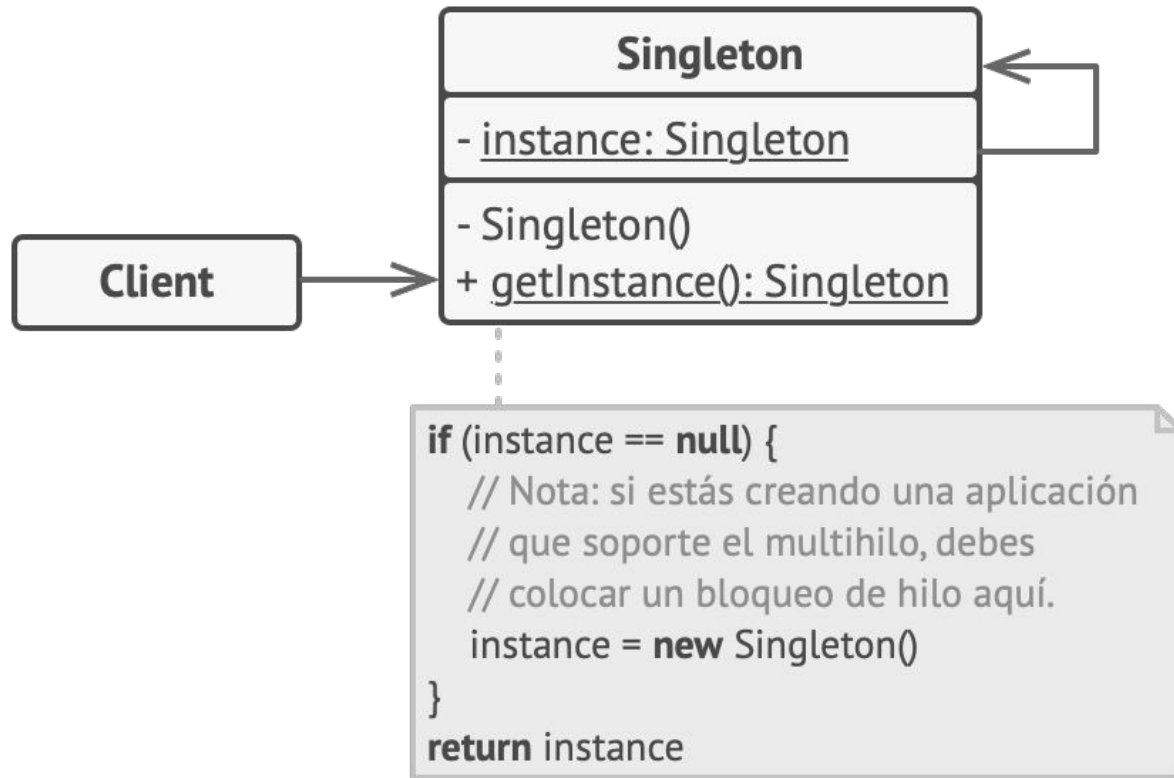
Los patrones son soluciones probadas y generalmente aceptadas para problemas comunes que surgen al diseñar aplicaciones y sistemas informáticos.

Estos patrones son soluciones reutilizables que ayudan a abordar problemas específicos de diseño.

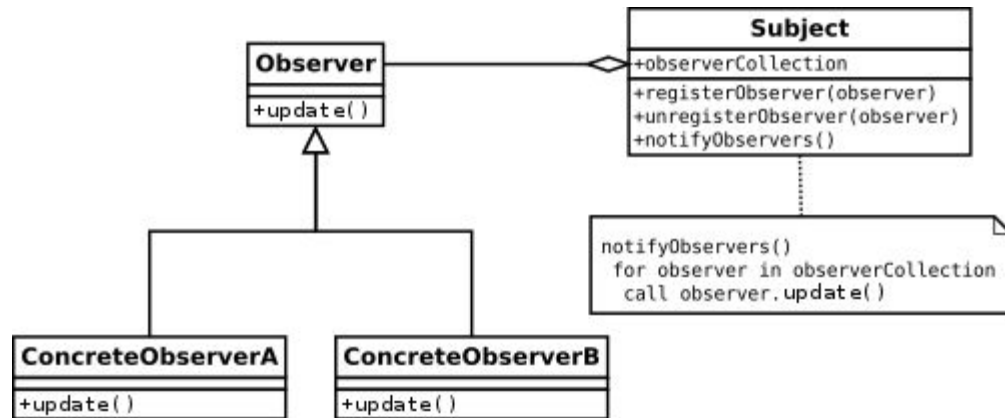
- Patrón MVC (Model-View-Controller): Divide una aplicación en tres componentes principales: el Modelo (que gestiona los datos y la lógica empresarial), la Vista (que muestra la información al usuario) y el Controlador (que maneja las interacciones del usuario).
- Patrón Singleton: Garantiza que una clase tenga una única instancia y proporciona un punto de acceso global a esa instancia.
- Patrón Observer: Define una relación de uno a muchos entre objetos, de modo que cuando un objeto cambia de estado, todos los objetos dependientes sean notificados y actualizados automáticamente.



Patrón MVC (Model-View-Controller)



Patrón de diseño creacional: Singleton



Patrón de comportamiento: Observer (o Publisher)

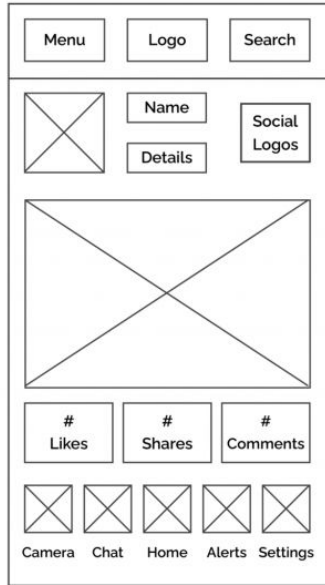
Wireframes, Mockups y Prototipos

Un wireframe es una representación visual básica y esquemática de la estructura de una interfaz de usuario.

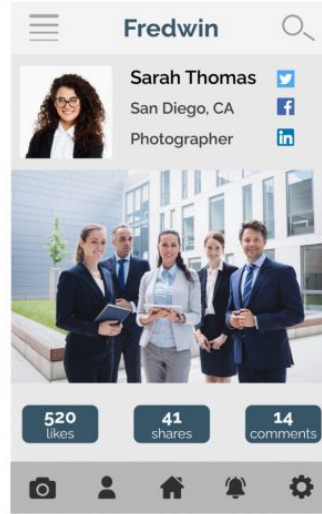
Permiten planificar y comunicar la disposición de los elementos en una pantalla, sin preocuparse por detalles visuales como colores, fuentes o imágenes.

Por el contrario, los mockups son una representación visual más compleja de una interfaz de usuario. Permiten evaluar la estética y la intuitividad.

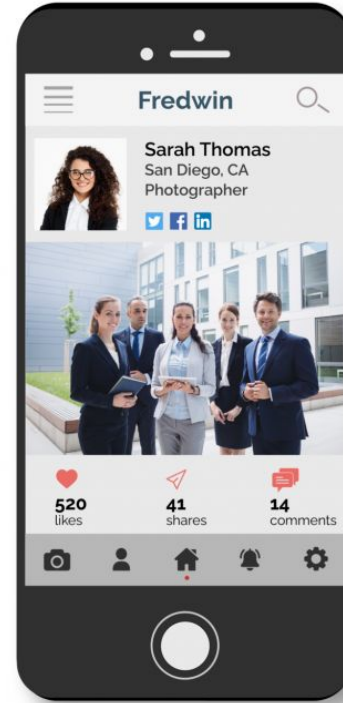
Finalmente, un prototipo es un mockup llevado a la total realidad pues se le agregan simulación de interacciones dentro de la plataforma final para evaluar el funcionamiento visual en tiempo real. Incluye animaciones, transiciones, etc.



Wireframe



Mockup



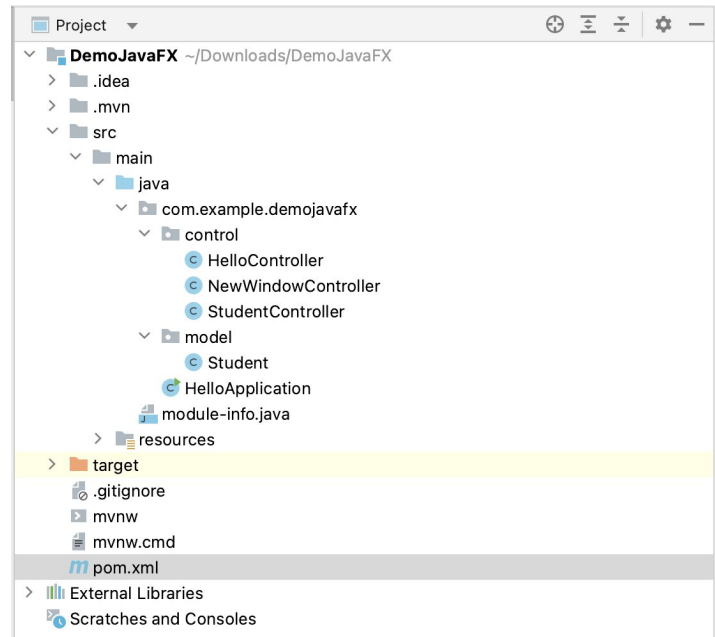
Prototype

Diferencias entre Wireframe, Mockup y Prototipo

Configuración de JavaFX con IntelliJ IDEA: Estructura

El proyecto tiene una nueva configuración:

- Paquete control/controller
- Paquete model
- Main de la aplicación
- Paquete resources (vista)

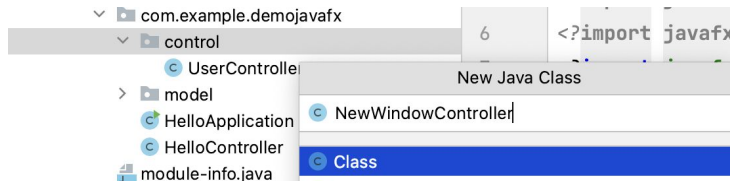


Configuración de JavaFX con IntelliJ IDEA: Crear proyecto

1. Crear la vista en fxml de la nueva ventana, en la carpeta resources



2. Crear el controlador



3. Actualizar el .fxml con el controlador

```

<AnchorPane xmlns="http://javafx.com/javafx"
             xmlns:fx="http://javafx.com/fxml"
             fx:controller="com.example.demojavafx.NewWindowController"
             prefHeight="400.0" prefWidth="600.0"
    
```

4. Ajustar el archivo de configuración del módulo, **module-info.java**

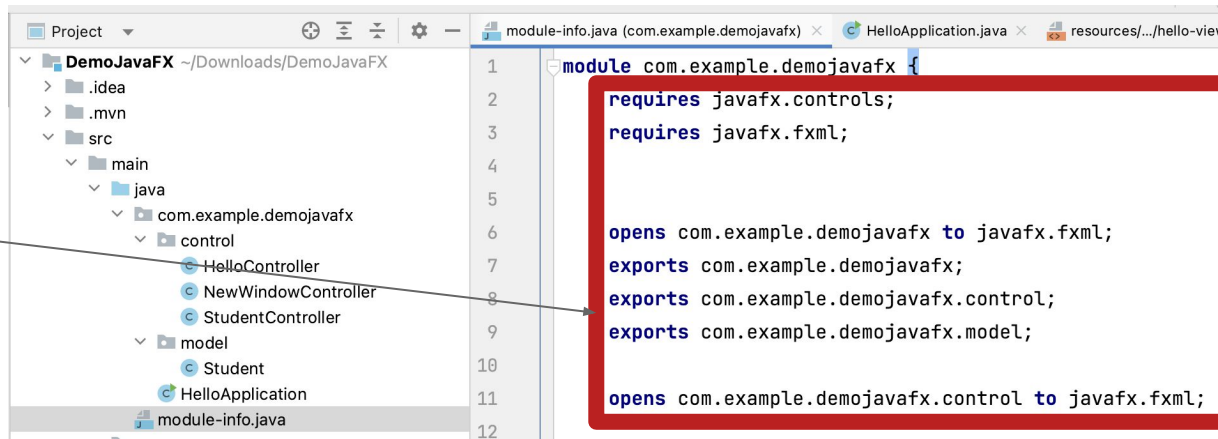
module-info.java

El archivo module-info.java se utiliza para definir un módulo en una aplicación Java.

Los módulos permiten dividir y organizar el código de una aplicación en unidades lógicas, lo que ayuda a mejorar la modularidad, la seguridad y la reutilización del código.

Configuración de JavaFX con IntelliJ IDEA: Archivo module-info.java

Líneas que hay que incluir en el archivo



- Definición
- Requires, para indicar las dependencias de tu módulo con otros módulos
- Exports, para especificar qué paquetes del módulo son accesibles desde otros módulos
- Opens, para permitir el acceso a paquetes específicos del módulo desde otros módulos.

Ejecución del proyecto JavaFX

Dependencia Maven

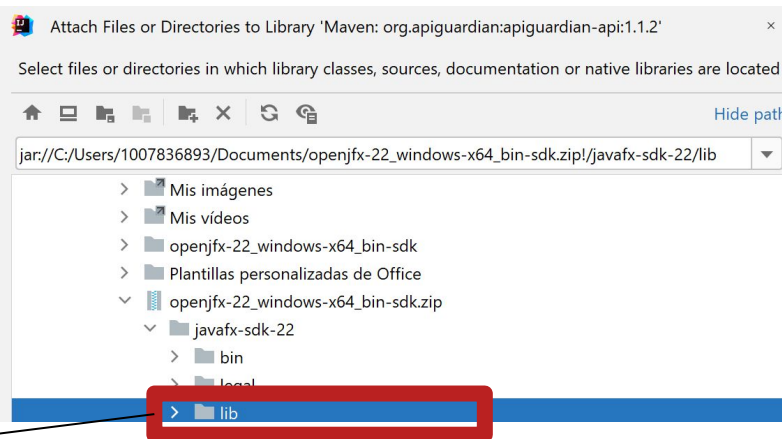
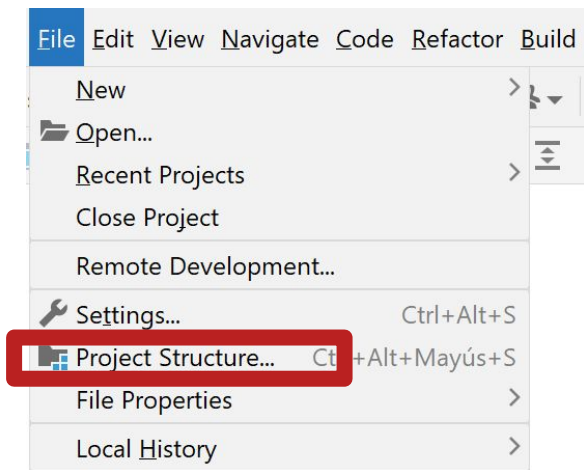
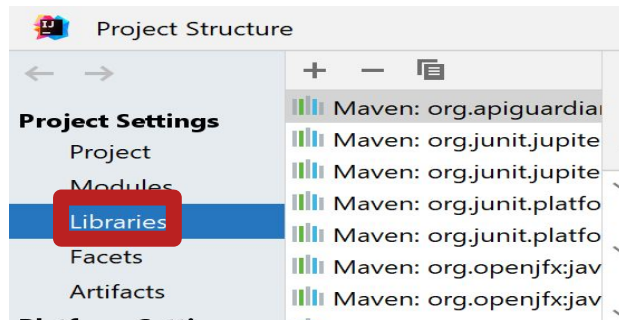
```
<dependency>
  <groupId>org.openjfx</groupId>
  <artifactId>javafx-controls</artifactId>
  <version>17.0.6</version>
</dependency>
<dependency>
  <groupId>org.openjfx</groupId>
  <artifactId>javafx-fxml</artifactId>
  <version>17.0.6</version>
</dependency>
```

<https://mvnrepository.com/artifact/org.openjfx/javafx/11>

Ejecución del proyecto JavaFX

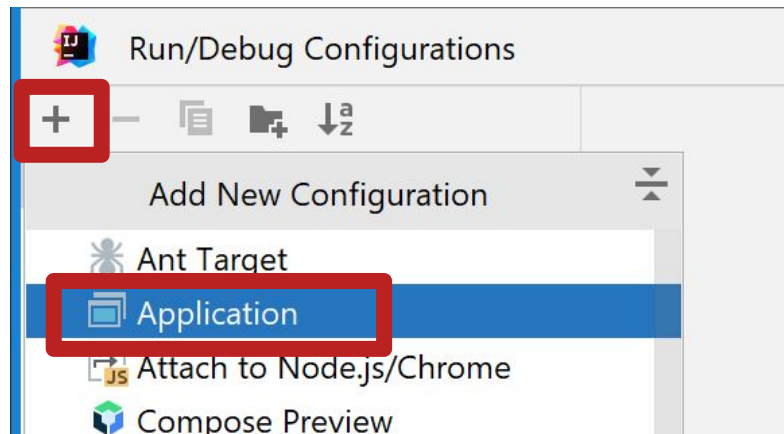
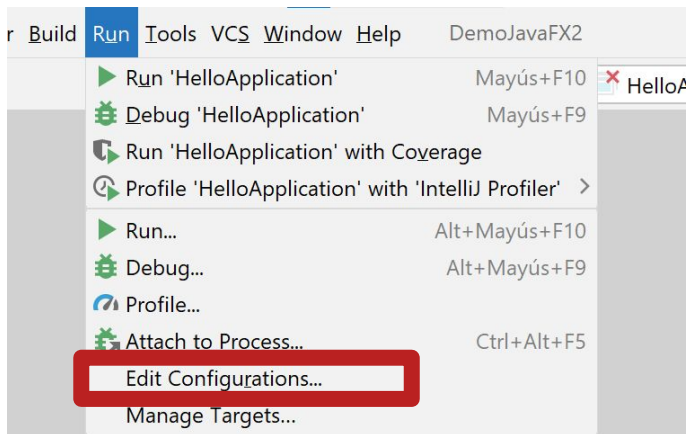
JavaFX SDK

<https://openjfx.io/>

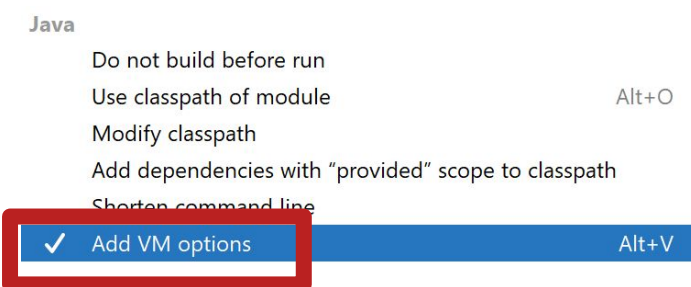
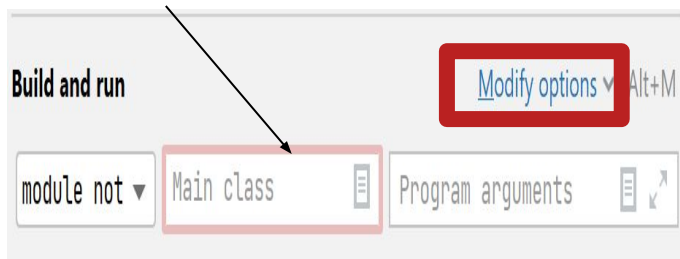


Añadir la lib de javafx

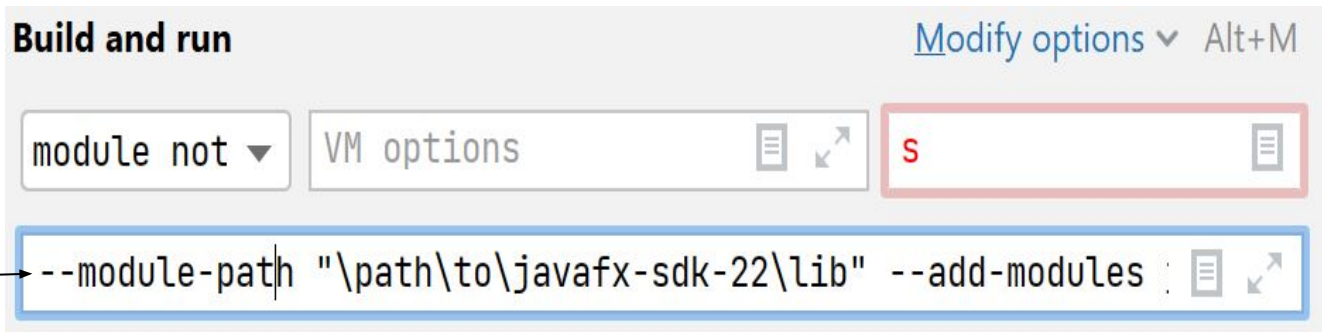
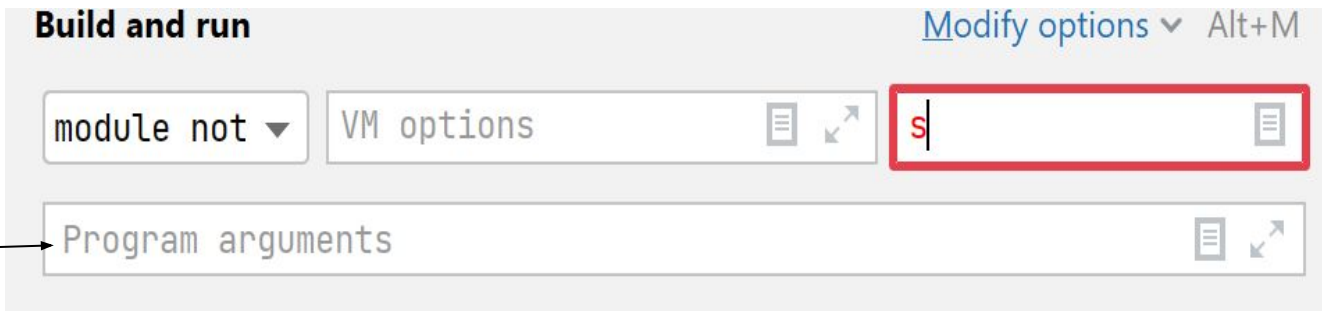
Ejecución del proyecto JavaFX



Especificar clase principal

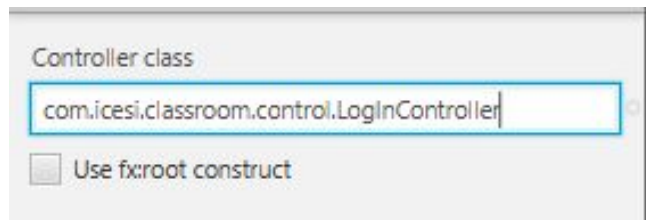


Ejecución del proyecto JavaFX

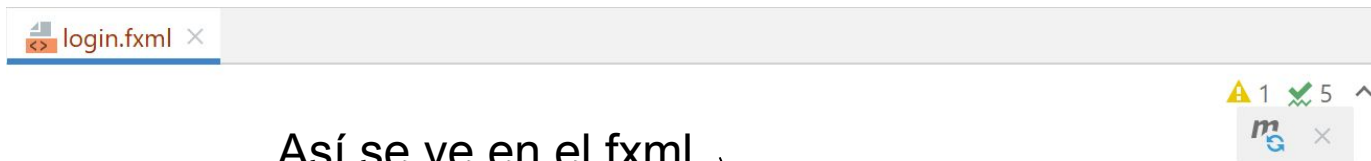


--module-path "\path\to\javafx-sdk-22\lib" --add-modules javafx.controls,javafx.fxml

Configuración de Controller en SceneBuilder



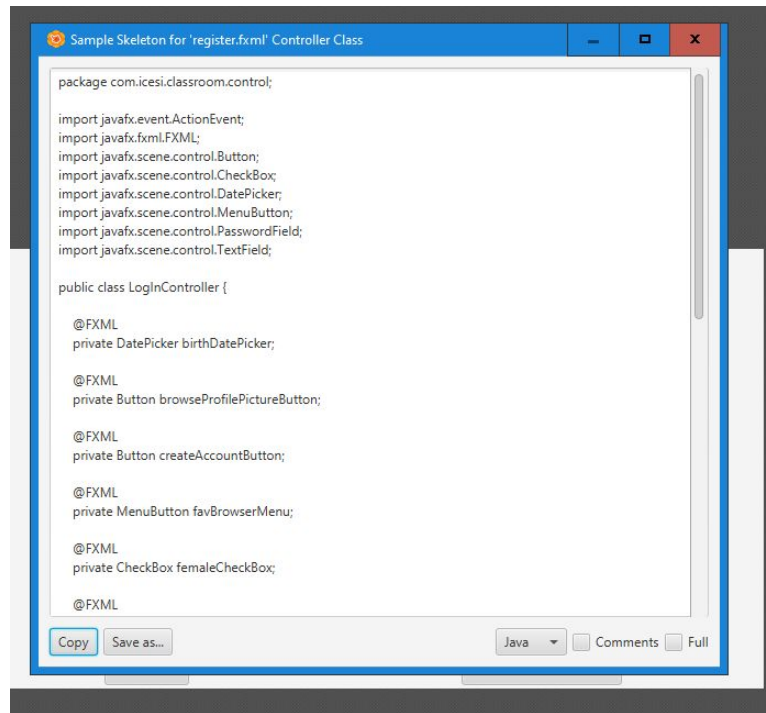
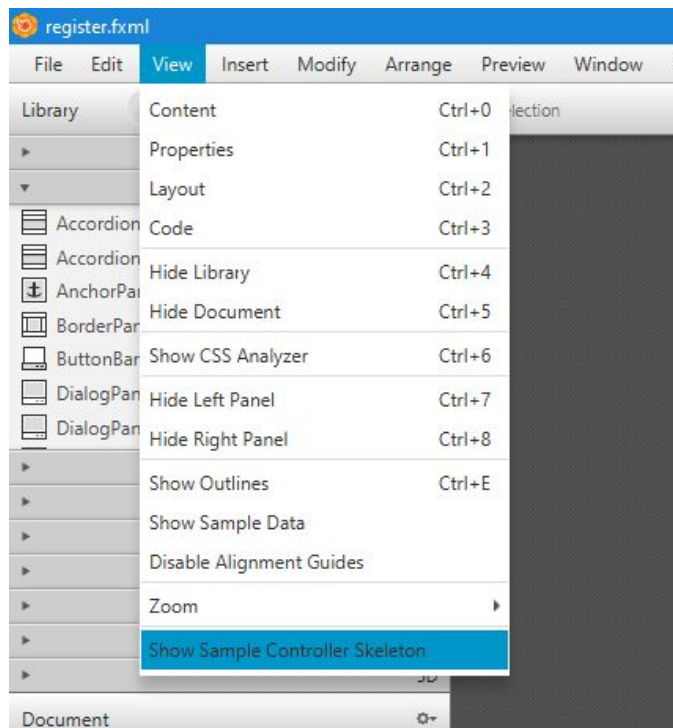
Agregar la controller en su respectivo paquete



Así se ve en el fxml

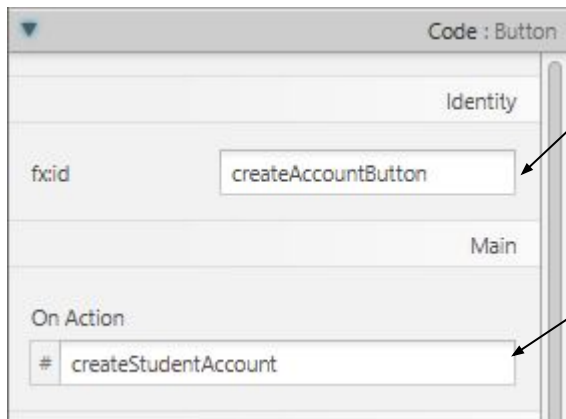
```
/javafx.com/fxml/1" fx:controller="com.icesi.classroom.control.LogInController">
```

Configuración de Controller en SceneBuilder



Generar el esqueleto de la controller en SceneBuilder

Configuración de Nodos en SceneBuilder



Agregar el id, nombre del nodo

Agregar la acción debida del nodo (Aplica para nodos que generan acción, botones) otros que retienen información como un campo de texto no necesitan “On Action”.

@FXML

```
private Button createAccountButton;
```

@FXML

```
void createStudentAccount(ActionEvent event) {  
  
}
```

Así se ven reflejados en el código de la controller

EJERCICIO PRÁCTICO # 1

Registro de estudiantes del curso

Mockup de ejemplo

Classroom

Join to Classroom

Create your account

Username:

Password:

Profile photo: **Browse**

Gender: ☐ Male ☐ Female ☐ Other

Career: ☐ Software Engineering ☐ Telematic Engineering ☐ Industrial Engineering

Birthday:

Favorite Browser:

Sign In **Create account**

Validation Error

You must fill each field in the form

OK

Account created

The new account has been created

OK

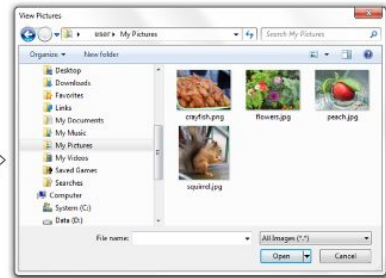
View Pictures

Organize...

- Desktop
- Downloads
- Favorites
- Links
- My Computer
- My Recent Places
- Search
- Computer
- System
- Data


October 2014

Mo	Tu	We	Th	Fr	Sa	Su
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4
5	6	7	8	9	10	11



Classroom

seyerman




Log out

User account list

Username	Gender	Career	Birthday	Browser
row 1, cell 1	row 1, cell 2	row 1, cell 1	row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2	row 2, cell 1	row 2, cell 1	row 2, cell 2
aadsfdfs	lkjhghj	aadsfdfs	aadsfdfs	lkjhghj
row 1, cell 1	row 1, cell 2	row 1, cell 1	row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2	row 2, cell 1	row 2, cell 1	row 2, cell 2
aadsfdfs	lkjhghj	aadsfdfs	aadsfdfs	lkjhghj
row 1, cell 1	row 1, cell 2	row 1, cell 1	row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2	row 2, cell 1	row 2, cell 1	row 2, cell 2

A small dialog box with a title bar that says "Validation Error" and a blue circular icon. The main text inside the box reads "You must fill each field in the form". At the bottom, there is a button labeled "OK".

Account created 

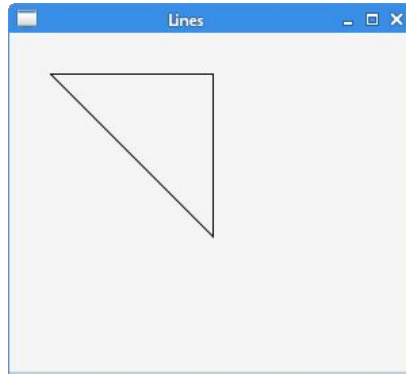
The new account has been created

OK

Gráficos 2D: Uso de Canvas en JavaFX

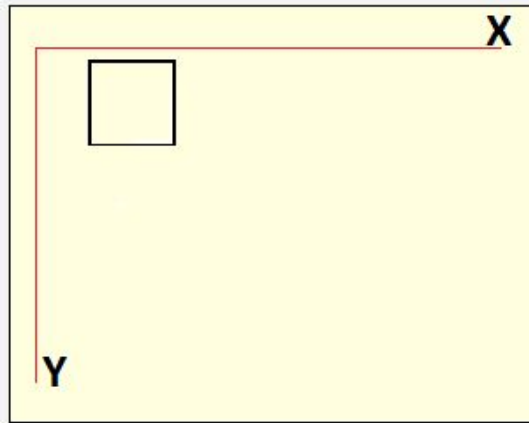
El control `javafx.scene.canvas.Canvas` nos proporciona una superficie de dibujo sobre la cual podemos mostrar texto, formas geométricas e imágenes.

Al crear una instancia del objeto Canvas debemos establecer su tamaño, usando el constructor o los correspondientes métodos `set` y obtener el `javafx.scene.canvas.GraphicsContext` que nos permitirá acceder a los comandos de dibujo requeridos.



Gráficos 2D: Uso de Canvas en JavaFX

Antes de comenzar a dibujar debemos agregar el Canvas de nuestro Scene, la clase Canvas hereda de la clase Node por lo que podemos manipularla y agregarla al Scene como cualquier otro control, luego debemos obtener el GraphicsContext, lo hacemos con el método `getGraphicsContext()` proporcionado por la clase Canvas. Importante tener en cuenta el origen de los ejes para este control.



Gráficos 2D: Dibujar figuras

La clase `GraphicsContext` proporciona comandos de dibujo para figuras básicas como: líneas, rectángulos, elipses y otros, a través de dos métodos, los comandos `fillXxx` y `strokeXxx` donde Xxx corresponde al nombre de la figura a dibujar, `fillXxx` dibujará la figura con su interior relleno con el color especificado y `strokeXxx` solo dibujara el contorno de la figura.

```
private void drawInCanvas(GraphicsContext gc)
{
    gc.fillRect(20, 20, 100, 100);
    gc.strokeRect(140, 20, 100, 100);
}
```



Gráficos 2D: Dibujar Texto

También podemos dibujar texto, ya sea solo el borde o con un color de relleno, contamos con los siguientes métodos

- `void strokeText(String text, double x, double y)`
- `void strokeText(String text, double x, double y, double maxWidth)`
- `void fillText(String text, double x, double y)`
- `void fillText(String text, double x, double y, double maxWidth)`

Ambos métodos “fill y stroke” poseen una versión sobrecargada, la cual permite indicar, además del texto a dibujar y la posición (x, y), el ancho máximo del texto, si se sobrepasa este valor el texto será escalado.

```
gc.setStroke(Color.BLACK);  
gc.strokeText("Drawing Text", 10, 10);  
gc.strokeText("Drawing Text", 100, 10, 40);
```

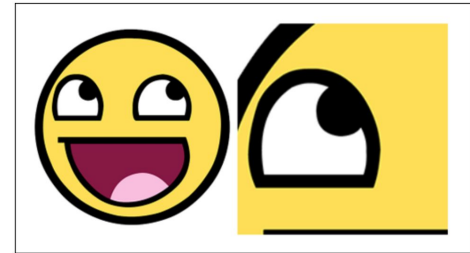
Stroke Text - Canvas JavaFX

Fill Text - Canvas JavaFX

Gráficos 2D: Dibujar Imágenes

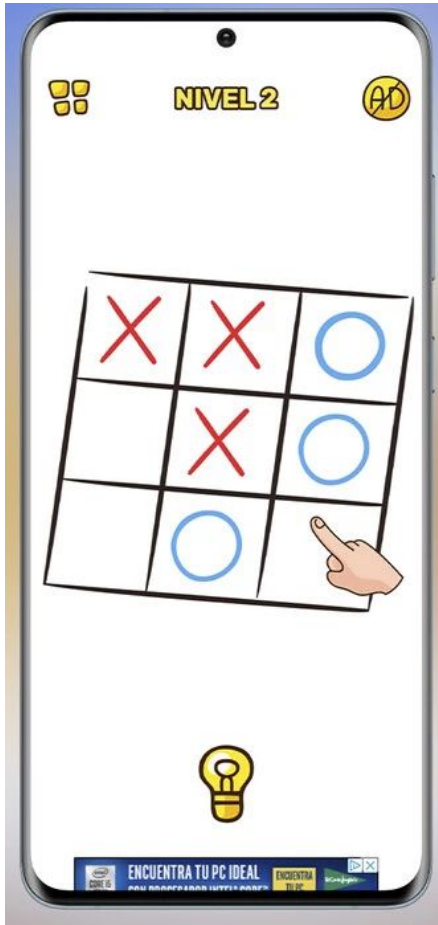
Para dibujar imágenes usaremos el comando `drawImage()` donde debemos indicar el objeto `Image` que deseamos dibujar, la posición y tamaño del mismo, este método cuenta con tres sobrecargas, la primera dibuja la imagen en la posición (x, y), la segunda nos permite definir el tamaño (w, h) y la tercera nos permite dibujar una región de la imagen (sx, sy, sw, sh) en la posición y tamaño indicados por los parámetros (dx, dy, dw, dh).

```
Image image = new Image(getClass().getResourceAsStream("image.png"));  
gc.drawImage(image, 20, 260);  
gc.drawImage(image, 20, 260, 256, 256);  
gc.drawImage(image, 50, 80, 200, 200, 20, 260, 256, 256);
```



EJERCICIO PRÁCTICO # 2

Juego de tricky (TicTacToe)



Condiciones:

- Usar MVC
- El tablero debe estar dibujado solo usando el Canvas
- No se pueden usar otras librerías gráficas de apoyo para desarrollo de juegos
- No se usa animación con hilos

Mockup de ejemplo

Gracias. 😊