

# JavaFX en NetBeans



## 1. Introducción

## 2. Proyectos JavaFX con el IDE NetBeans

2. 1. Cree una nueva biblioteca global en:
2. 2. Crea un proyecto de Java
2. 3. Establecer JDK
2. 4. Configurar el proyecto.
2. 5. Agregar clases `JavaFX`
2. 6. Añadir opciones a la VM
2. 7. Prueba final de la aplicación

## 3. Píldoras informáticas relacionadas

## 4. Fuentes de información

# 1. Introducción

---

Vamos a ver cómo crear una aplicación `JavaFX` en NetBeans. Necesitamos:

- Apache Netbeans 16 o posterior
- OpenJDK 17 (seguramente funcionará con una posterior)

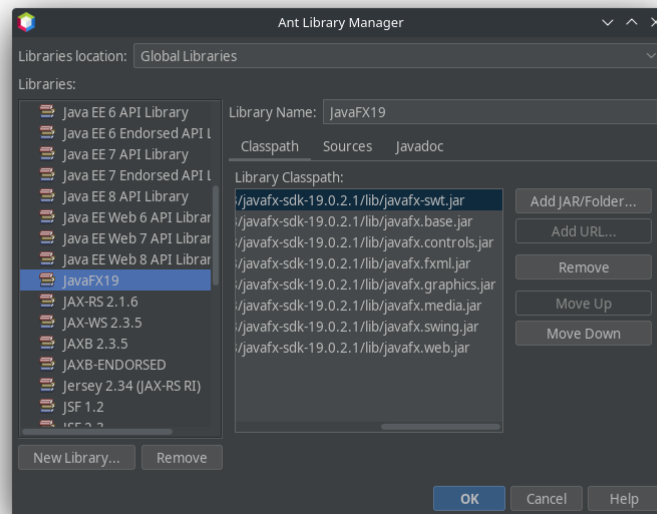
## 2. Proyectos JavaFX con el IDE NetBeans

Sigue estos pasos para crear un proyecto `JavaFX` utilizando las herramientas IDE para compilarlo y ejecutarlo.

- Descargar el SDK (versión 19) de `JavaFX` <https://gluonhq.com/products/javafx/> apropiado para tu sistema operativo.
- Descomprímelo en la ubicación deseada, por ejemplo, `/Users/your-user/Downloads/javafx-sdk-19`. Es **IMPORTANTÍSIMO** que recuerdes la ruta (`path`) hasta esta librería.

### 2.1. Cree una nueva biblioteca global en:

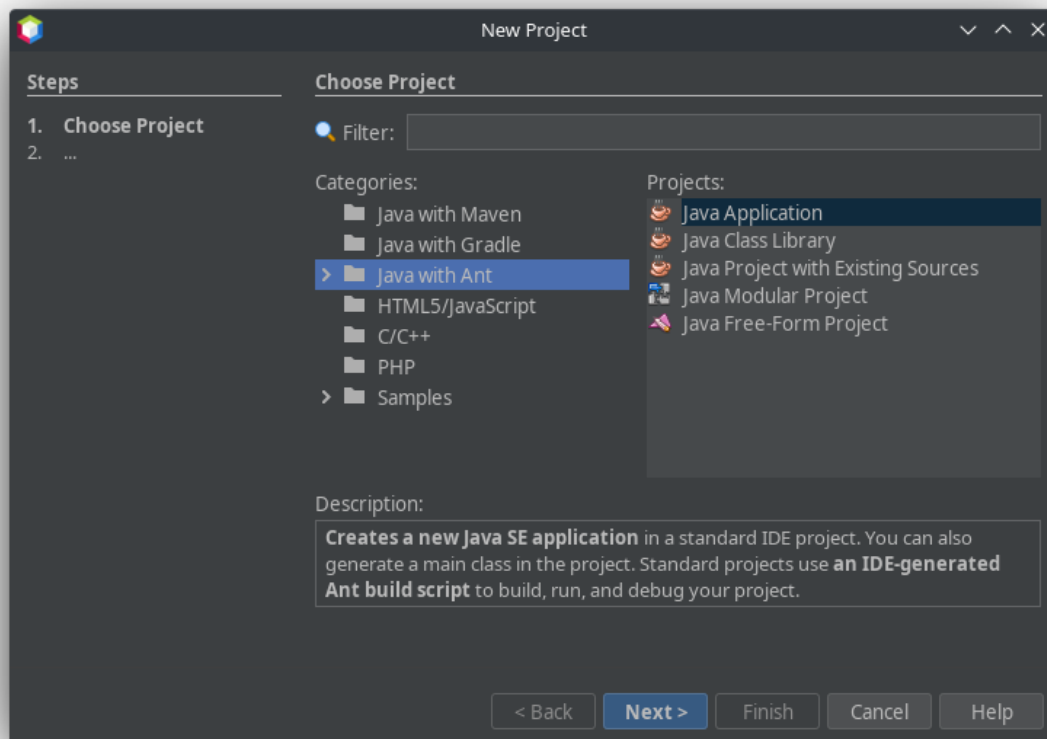
- `Tools` -> `Library` -> `New Library` (abajo a la izquierda)
- Nómbrala `JavaFX19` (por ejemplo) que incluya los archivos jar en la carpeta lib de JavaFX 19.



**Nota importante:** asegúrate de no añadir excepción al ejecutar el proyecto.

### 2.2. Crea un proyecto de Java

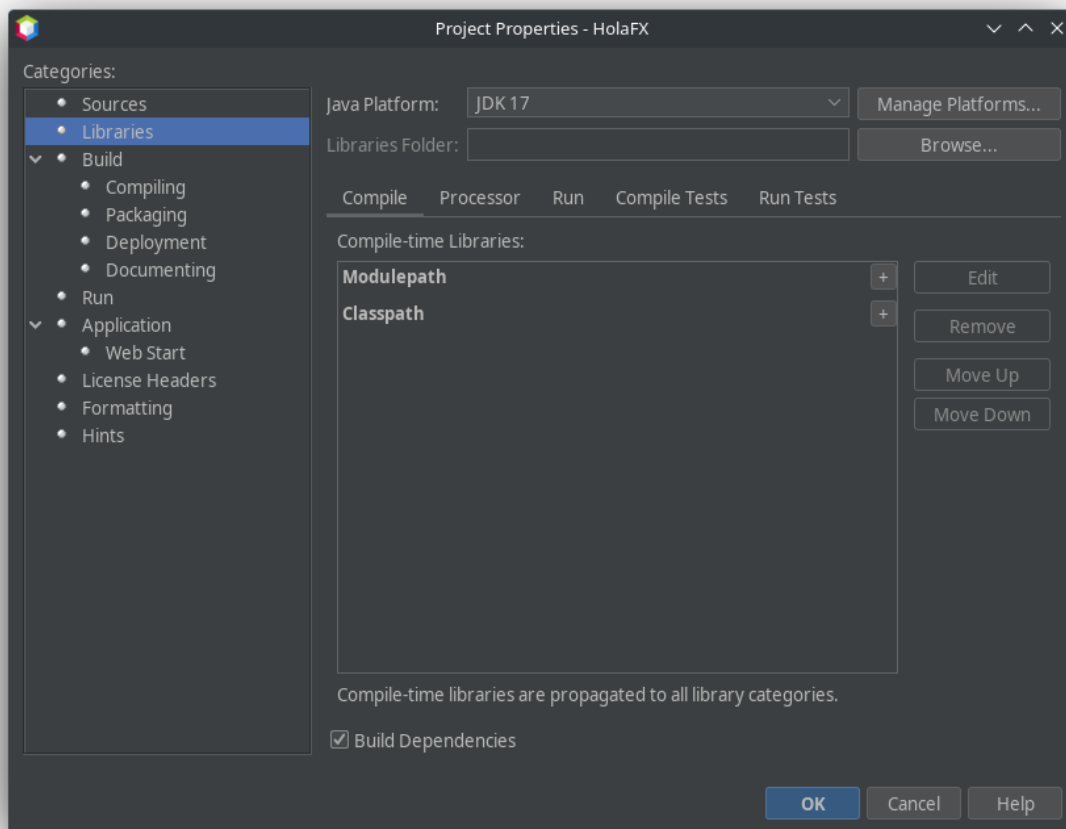
Proporciona un nombre para el proyecto, como `Ho1aFX`, y una ubicación. Se abrirá un proyecto predeterminado.



**Advertencia:** no intente crear un proyecto `JavaFX`. Las tareas `JavaFX` `Ant` de la versión actual de Apache NetBeans aún no están listas para `JavaFX` 19, a menos que tenga un JDK personalizado que incluya `JavaFX`.

## 2.3. Establecer JDK

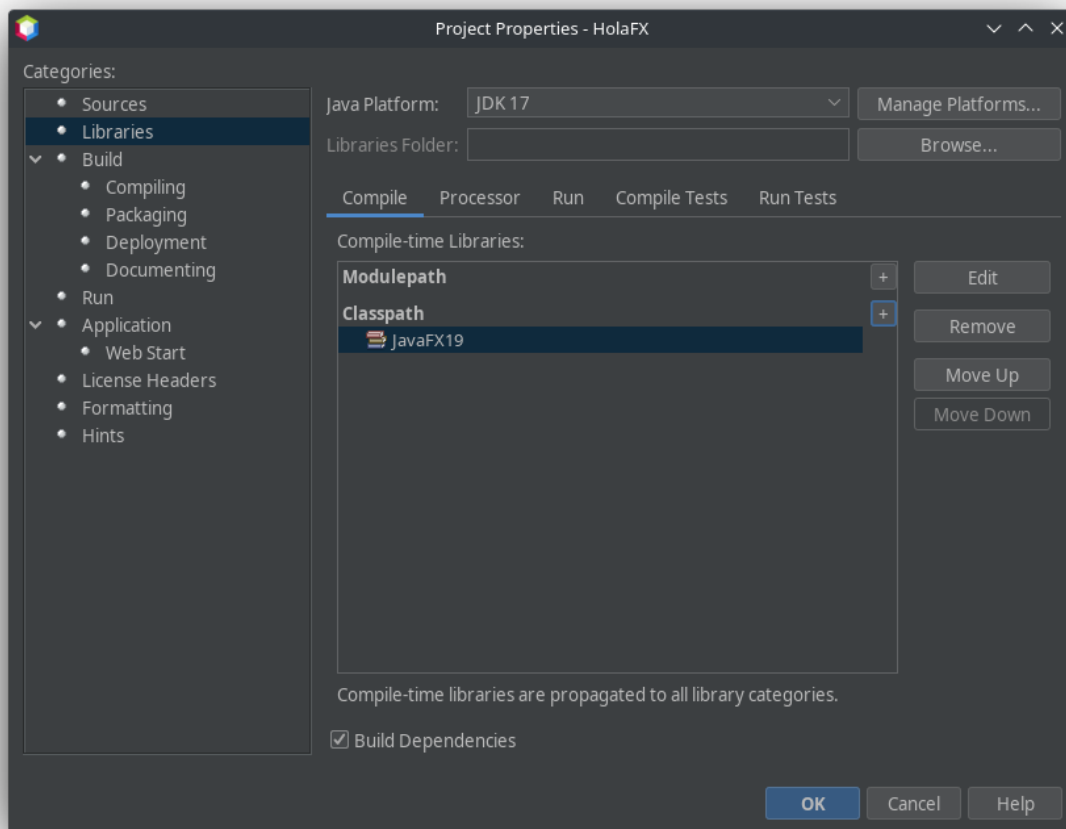
Asegúrate de que su proyecto esté configurado para ejecutarse con JDK 19 o posterior. Abrimos las propiedades del proyecto (botón derecho sobre el nombre del proyecto) y `Properties` -> `Libraries` -> (campo) `Java Platform` y configúralo en tu JDK preferido.



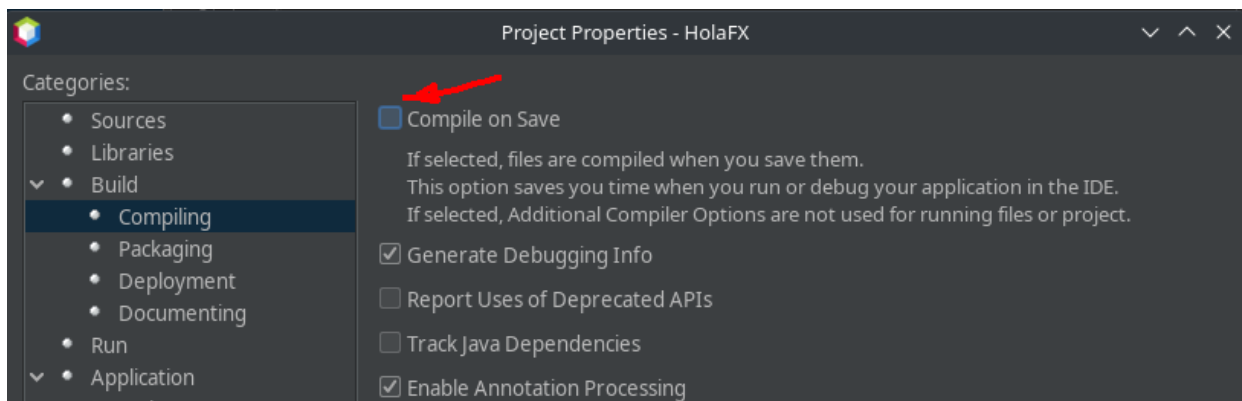
## 2.4. Configurar el proyecto.

Añadir la biblioteca:

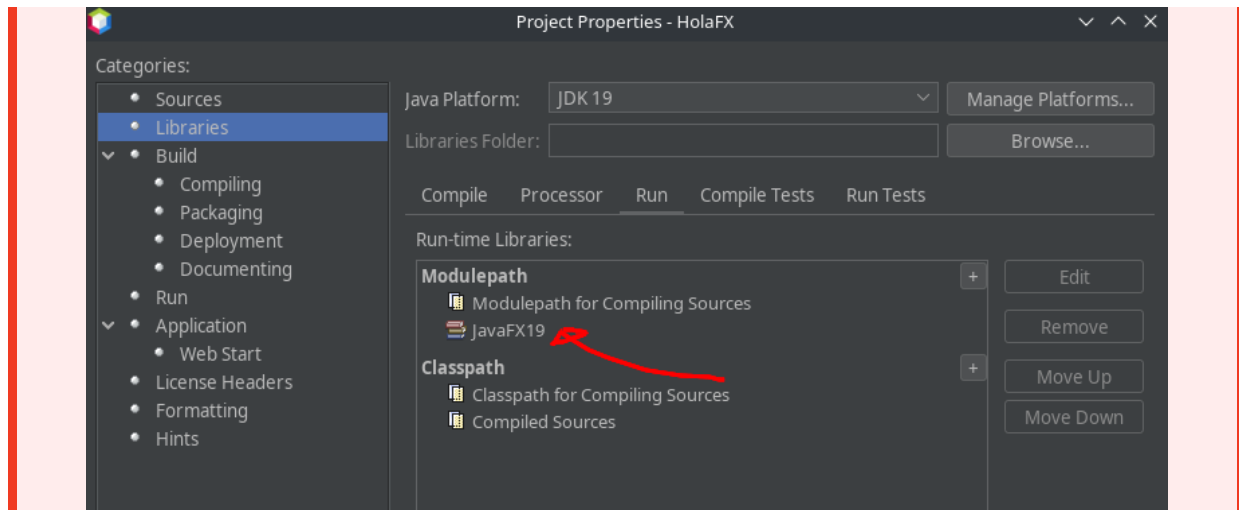
Iremos a **Properties** -> **Libraries** -> Selecciona **Classpath** -> + -> **Add Library** y añadimos la biblioteca **JavaFX**.



Ir a **Propiedades** -> **Generar** -> **Compilar** y asegúrese de anular la selección de la opción **Compilar al guardar**.



**Advertencia:** si NetBeans se compila cada vez que se guarda, también agregará las clases a la ruta del módulo, evitando cualquier cambio adicional en la ruta del módulo. Se recomienda agregar la librería al Modulepath accediendo a **Properties** -> **Library** -> **Run** -> **Modulepath**.



Una vez que se establece el `classpath`, el IDE reconocerá las clases `JavaFX`.

Por ejemplo, vamos comenzar

1. heredando la clase principal de la clase `Application`.
2. tendremos que importar las clases de la librería.
3. implementar el método abstracto, añadir el `launch()` al `main()` y añadir los imports necesarios.

```

1  package holaFX;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Parent;
6  import javafx.scene.Scene;
7  import javafx.stage.Stage;
8
9  public class HolaFX extends Application {
10
11      /**
12       * @param args the command line arguments
13       */
14      public static void main(String[] args) {
15          launch(args);
16      }
17
18      @Override
19      public void start(Stage stage) throws Exception {
20          Parent root = FXMLLoader.load(getClass().getResource("FXML
21  HolaFX.fxml"));
22          stage.setTitle("Hola Mundo");
23          stage.setScene(new Scene(root));
24          stage.show();
25      }
26  }

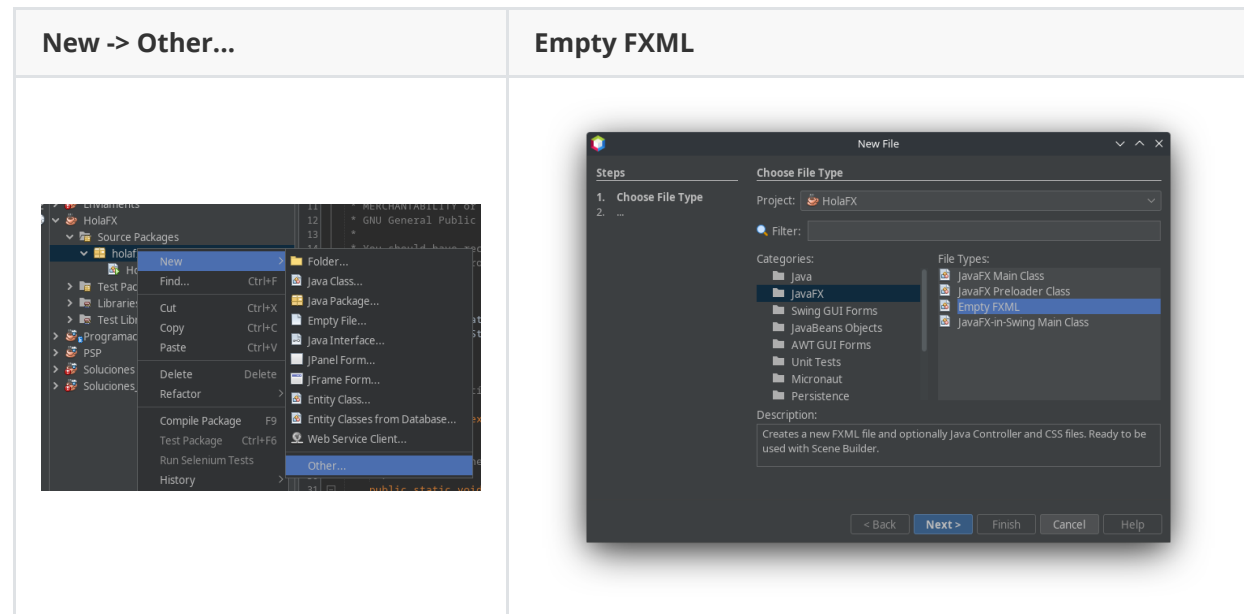
```



## 2.5. Agregar clases JavaFX

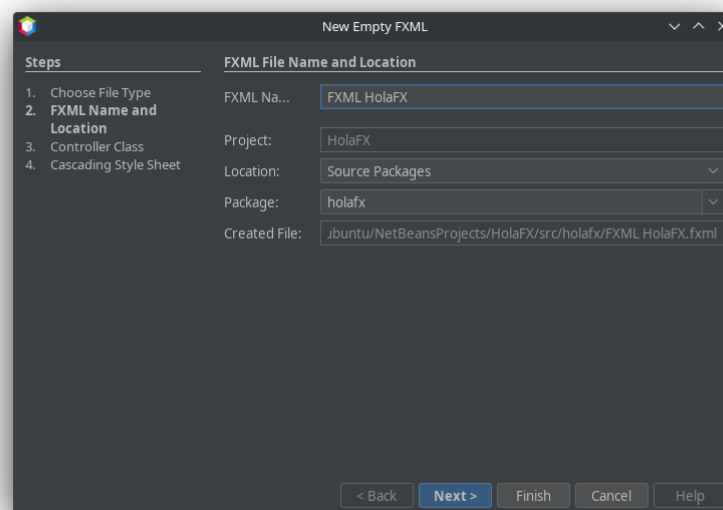
Cuando queramos trabajar con el modelo MVC (en los primeros ejemplos no lo haremos), podremos añadir un archivo FXML con su controlador y una hoja de estilo.

Botón derecho sobre el Package -> **new** -> **Other** -> **JavaFX** -> **Empty FXML**

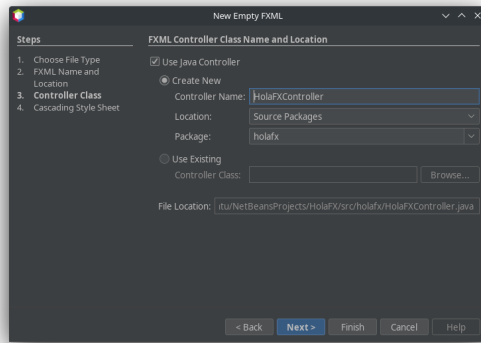
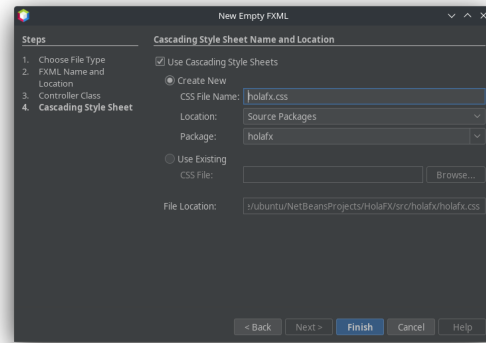


Nota: Si no aparece **new Empty FXML** file, accede a **Tools** -> **Options** -> **Java** -> **JavaFX** o prueba a cambiar la versión de tu JDK.

Indicamos el nombre de la Vista:



Elegimos crear un nuevo Controlador y también CSS:

**Controlador****CSS**

Advertencia: si ejecuta ahora el proyecto, se compilará pero obtendrá este error: **Error:** JavaFX runtime components are missing, and are required to run this application.

Este error se muestra porque Java 19 verifica si la clase principal extiende de `javafx.application.Application`. Si ese es el caso, es necesario tener el módulo `javafx.graphics` en la ruta del módulo.

## 2.6. Añadir opciones a la VM

Para resolver el problema, Ir a **Properties** -> **Run** del proyecto y añadir estas opciones en el campo VM:

- Para Linux y Mac:

```
1 --module-path /path/to/javafx-sdk-19/lib --add-modules
  javafx.controls,javafx.fxml
```

- Para Windows:

```
1 --module-path "\path\to\javafx-sdk-19\lib" --add-modules
  javafx.controls,javafx.fxml
```

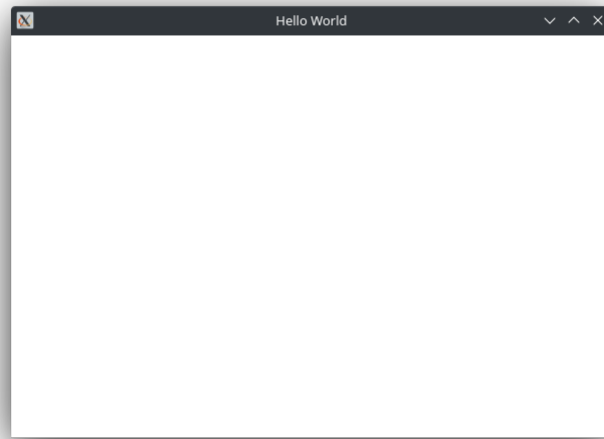
Donde /path/to debes sustituirlo por tu ruta a la librería, por ejemplo:

```
1 --module-path /media/DADES/PRG_2223/javafx-sdk-19.0.2.1/lib/ --add-modules
  javafx.controls,javafx.fxml
```

**IMPORTANTE:** un espacio en blanco detrás de la coma hará que el error continúe.

## 2.7. Prueba final de la aplicación

Si hemos seguido todos los pasos correctamente, podremos ejecutar nuestra aplicación y ver la ventana titulada "Hola Mundo":



### 3. Píldoras informáticas relacionadas

---

- <https://www.youtube.com/playlist?list=PLNjWMbvTjAljLRW2qyuc4DEgFVW5YFRSR>
- <https://www.youtube.com/playlist?list=PLaxZkGILWHGUWZxuadN3J7KKaICRIhz5->

## 4. Fuentes de información

---

- Apuntes de Jose Antonio Diaz-Alejo
- <https://github.com/openjfx/openjfx-docsopen>
- <https://github.com/openjfx/samples>
- [FXDocs](#)
- <https://openjfx.io/openjfx-docs/>
- <https://docs.oracle.com/javase/8/javafx/user-interface-tutorial>
- <https://github.com/JonathanGiles/scenic-view>