

Instalación y Configuración

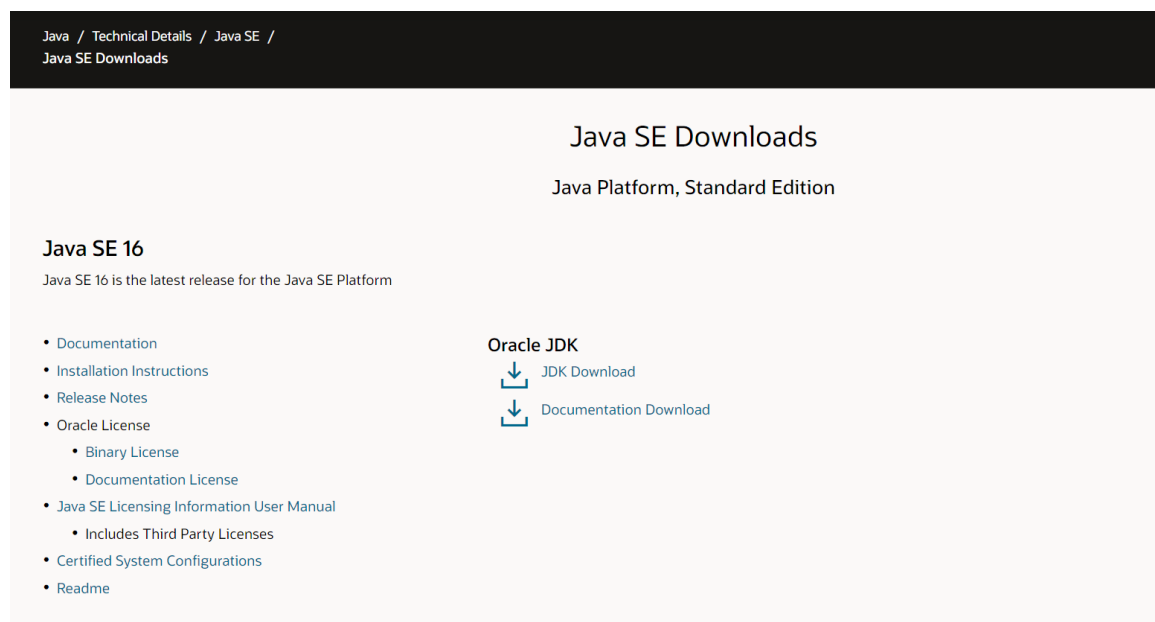
Para comenzar vamos a instalar y configurar el software. Utilizaremos **NetBeans** 12.3, la versión 16 de **JDK**, la versión 16.0.1 de **JavaFX**, y la versión 16.0.0 de **Scene Builder**.

Para otros IDEs (Eclipse, IntelliJ), el procedimiento es similar, e indicaremos donde consultar los pasos a seguir.

JDK

Primero, suponiendo que ya tenemos instalada la última versión de nuestro IDE, en este caso NetBeans 12.3 (<https://NetBeans.apache.org/download/index.html>), procederemos a actualizar la versión del JDK a la más reciente, la versión 16, además de instalar su documentación para consultar desde el IDE.

<https://www.oracle.com/java/technologies/javase-downloads.html>



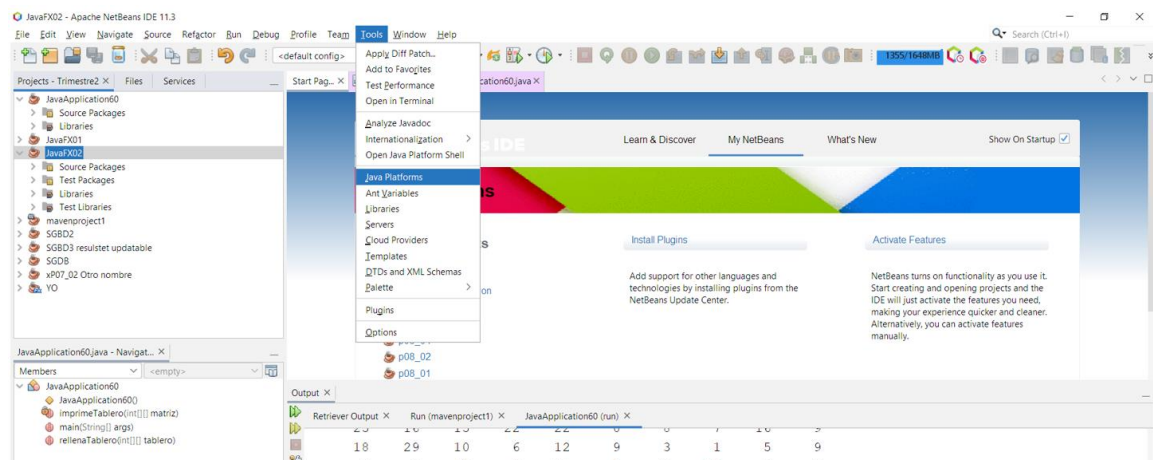
The screenshot shows the Oracle Java SE Downloads page for Java SE 16. The page has a dark header with the navigation path: Java / Technical Details / Java SE / Java SE Downloads. The main content area is titled "Java SE Downloads" and "Java Platform, Standard Edition". Under "Java SE 16", it states "Java SE 16 is the latest release for the Java SE Platform". On the left, there is a list of links: Documentation, Installation Instructions, Release Notes, Oracle License (with sub-links for Binary License and Documentation License), Java SE Licensing Information User Manual (with a sub-link for Includes Third Party Licenses), Certified System Configurations, and Readme. On the right, under "Oracle JDK", there are two download links: "JDK Download" and "Documentation Download", each with a download icon.

Pulsamos sobre **JDK Download** y sobre **Documentation Download**. Este último es un fichero .zip (**jdk-16_doc-all.zip**). Para el primero tendremos que seleccionar la opción deseada en función de nuestra plataforma:

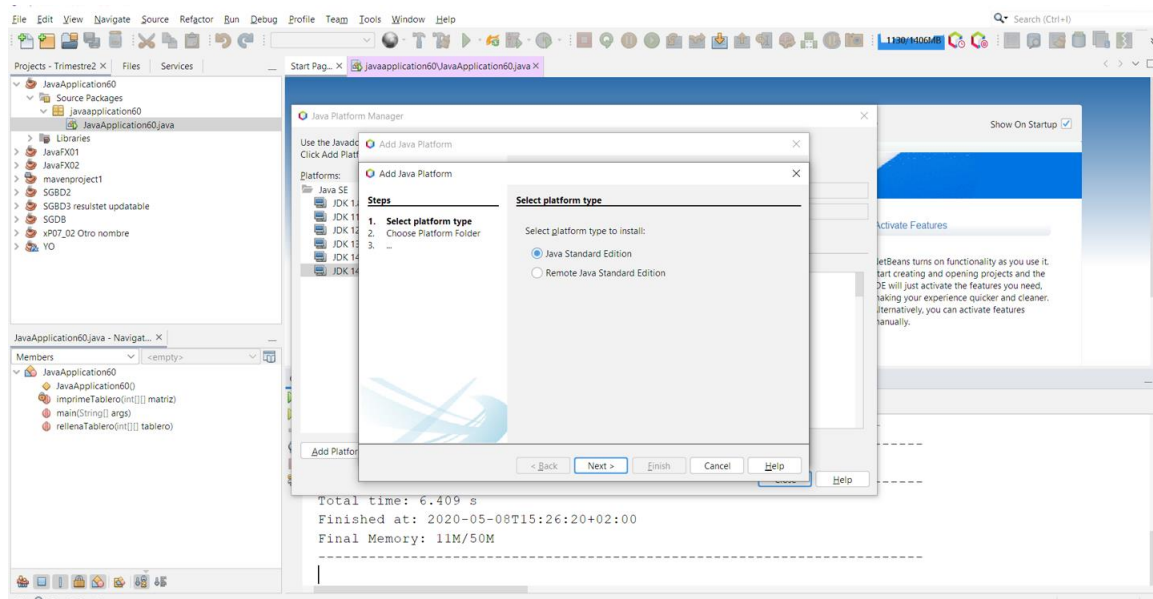
Linux ARM 64 RPM Package	144.84 MB	jdk-16_linux-aarch64_bin.rpm
Linux ARM 64 Compressed Archive	160.69 MB	jdk-16_linux-aarch64_bin.tar.gz
Linux x64 Debian Package	146.14 MB	jdk-16_linux-x64_bin.deb
Linux x64 RPM Package	152.96 MB	jdk-16_linux-x64_bin.rpm
Linux x64 Compressed Archive	170 MB	jdk-16_linux-x64_bin.tar.gz
macOS Installer	166.56 MB	jdk-16_osx-x64_bin.dmg
macOS Compressed Archive	167.16 MB	jdk-16_osx-x64_bin.tar.gz
Windows x64 Installer	150.55 MB	jdk-16_windows-x64_bin.exe

En mi caso, escogeré la Windows x64 Installer. Una vez bajada, la ejecutamos y ya tenemos la última versión del JDK; pero hay que configurar NetBeans para que la reconozca y, en este caso, además la pondremos como plataforma por defecto. Esto no es necesario para el funcionamiento de JavaFX, pero aprovechamos para tener las últimas versiones del software.

En NetBeans vamos a **Tols -> Java Platforms**

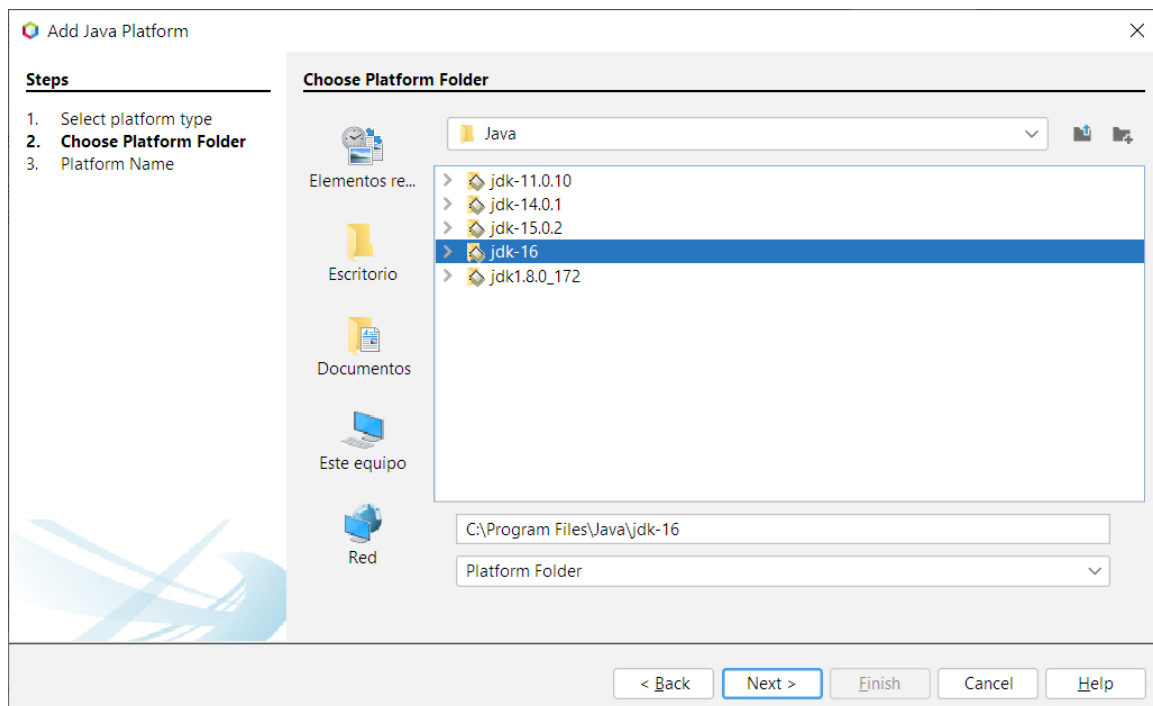


Y escogemos Add Platform

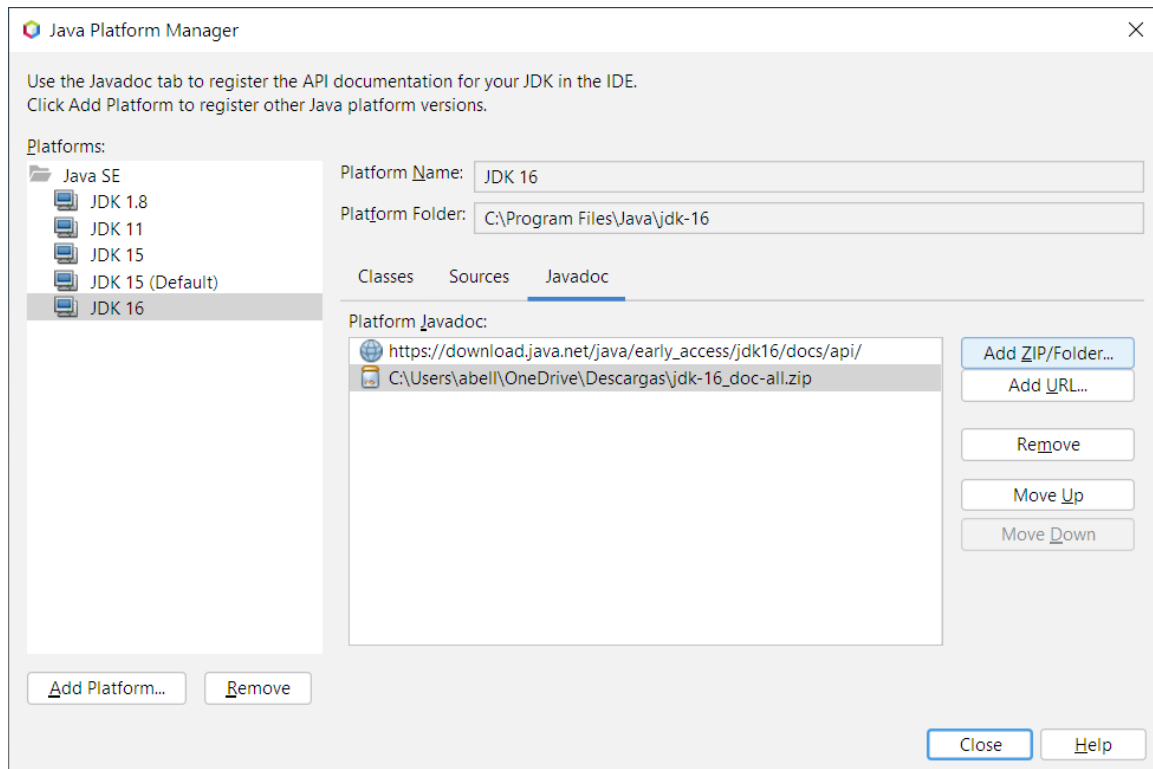


escogemos el nuevo JDK. En Windows se instala por defecto debajo del directorio:

C:\Program Files\Java



Una vez instalado, para añadir la documentación, seleccionamos la pestaña **Javadoc** y pulsamos **Add Zip/Folder**. Introducimos el fichero zip que hemos descargado anteriormente.



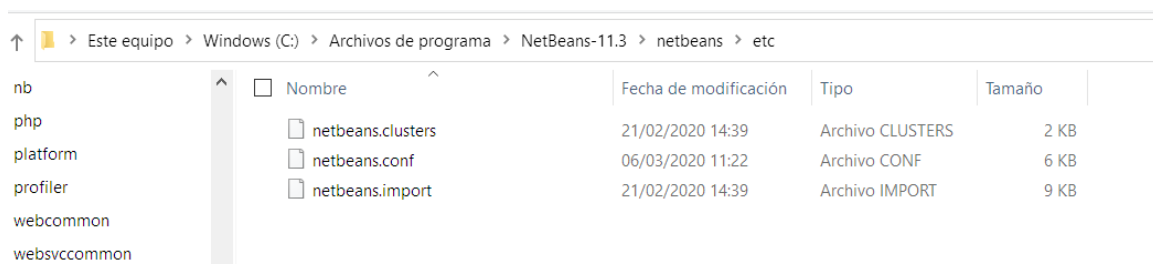
Vemos que ahora ya tenemos la plataforma JDK 16, sin embargo, no es la plataforma por defecto, ni hay manera de fijarla como tal desde el IDE. Tenemos que cerrar NetBeans para editar su fichero de configuración. En Windows este fichero se encuentra en:

C:\Archivos de programa\NetBeans-12.3\netbeans\etc

El fichero se llama:

netbeans.conf

Está protegido contra escritura, con lo que, o bien le cambiáis los permisos antes y después de editarlo, o lo abríis con permisos de administrador.



En este fichero hay que buscar la línea que se muestra a continuación y modificarla de modo que haga referencia al nuevo JDK:

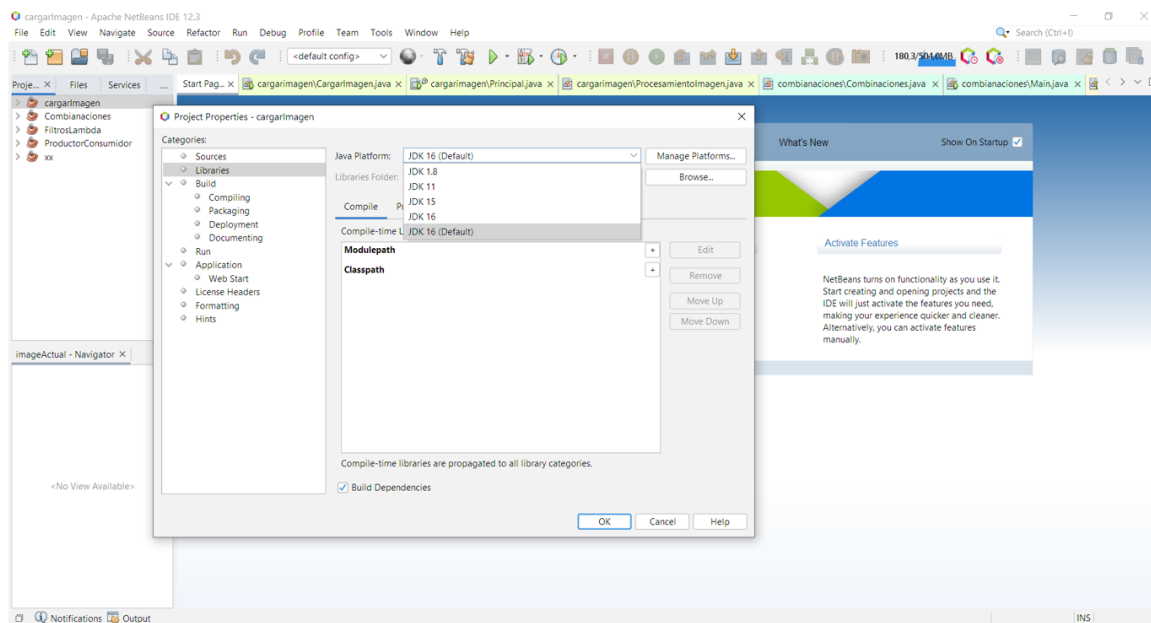
```

61 # as a workaround -J-Djdk.gtk.version=2.2 has been added to the
62 # default command line arguments.
63 # (see: https://issues.apache.org/jira/browse/NETBEANS-1344)
64 #
65 netbeans_default_options="-J-XX:+UseStringDeduplication -J-Djdk.lang.Process
66
67 # Default location of JDK:
68 # (set by installer or commented out if launcher should decide)
69 #
70 # It can be overridden on command line by using --jdkhome <dir>
71 # Be careful when changing jdkhome.
72 # There are two NetBeans launchers for Windows (32-bit and 64-bit) and
73 # installer points to one of those in the NetBeans application shortcut
74 # based on the Java version selected at installation time.
75 #
76 netbeans_jdkhome="C:\Program Files\Java\jdk-13.0.1"
77
78 # Additional module clusters:
79 # using ${path.separator} ( ';' on Windows or ':' on Unix ):
80 #
81 # netbeans_extraclusters="/absolute/path/to/cluster1:/absolute/path/to/clu
82

```

En este caso tendría que cambiar el **13.0.1** por **16**

Al volver a arrancar NetBeans ya tendremos como plataforma por defecto la 16. Pero podemos utilizar la que queramos (de las instaladas) en nuestros proyectos, cambiándolo en las propiedades de estos:



Librerías JavaFX

Empezamos ahora con JavaFX. Las librerías JavaFX no pertenecen ya a Oracle. Ahora se encarga de las mismas la empresa Gluon. Iremos a su web para descargarlas. En su web también tenemos ayuda paso a paso para su instalación y configuración en los principales IDEs (NetBeans, IntelliJ, Eclipse), además de ejemplos, tutoriales, y la aplicación Scene Builder.

JavaFX es un conjunto de librerías que nos permiten desarrollar aplicaciones gráficas en Java (no son las únicas, hay más opciones disponibles). Dentro de las propias JavaFX hay varias maneras de proceder al desarrollo. Nosotros lo haremos utilizando FXML, un lenguaje de marcado basado en XML, para el diseño de la interfaz, y Java para el desarrollo de la lógica del programa. Cuando programamos en interfaces gráficas utilizamos otro paradigma de la programación denominado **Programación Orientada a Eventos**. La idea, a grandes rasgos, es asociar código a eventos que se puedan producir durante la ejecución de la aplicación: pulsar un botón de ratón, introducir un texto, coger o perder el foco de un elemento,

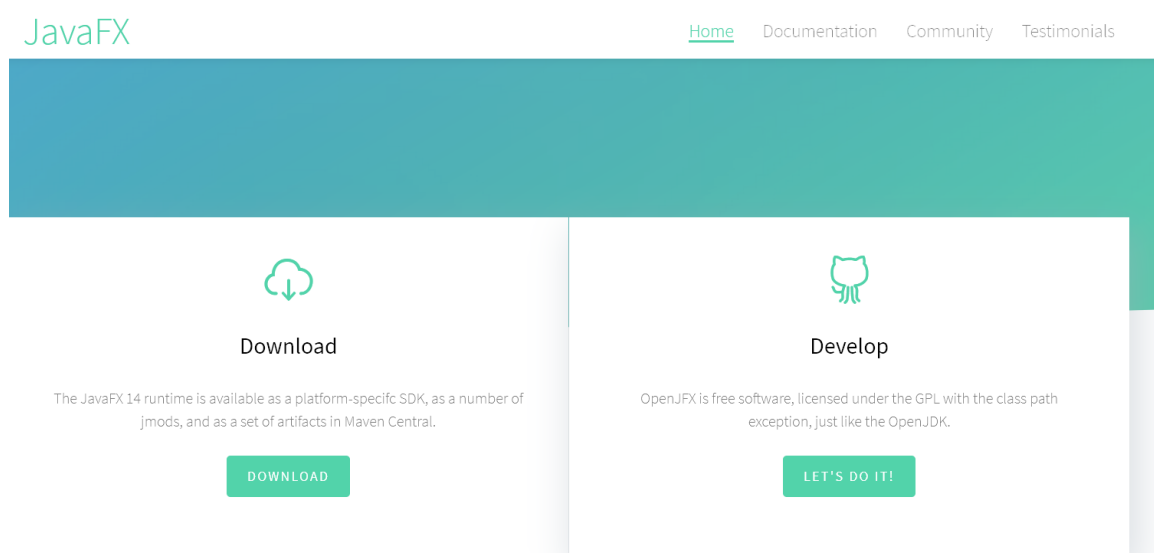
Aunque como hemos dicho, el diseño de la interfaz será mediante FXML, no será necesario escribir el código. Utilizaremos la aplicación **Scene Builder** para realizar el diseño de modo visual arrastrando y colocando elementos (casillas de verificación, botones, cuadros de texto,) y asignándoles propiedades y eventos.

Vamos pues a instalar JavaFX. Accedemos a:

NOTA: estos pasos, de descarga de las librerías JavaFX no son necesarios si vamos a utilizar MAVEN, ya que el propio gestor se encargará de su descarga y configuración

<https://openjfx.io>

Y nos desplazamos hasta la sección de Descargas:



Escogiendo la versión correspondiente a nuestra plataforma. A día de hoy, la versión LTS (long term support) tanto del JDK de Java como de estas librerías, es la 11, pero utilizaremos las últimas disponibles, la 16. Para Windows de 64 bits:

JavaFX Windows X64 SDK

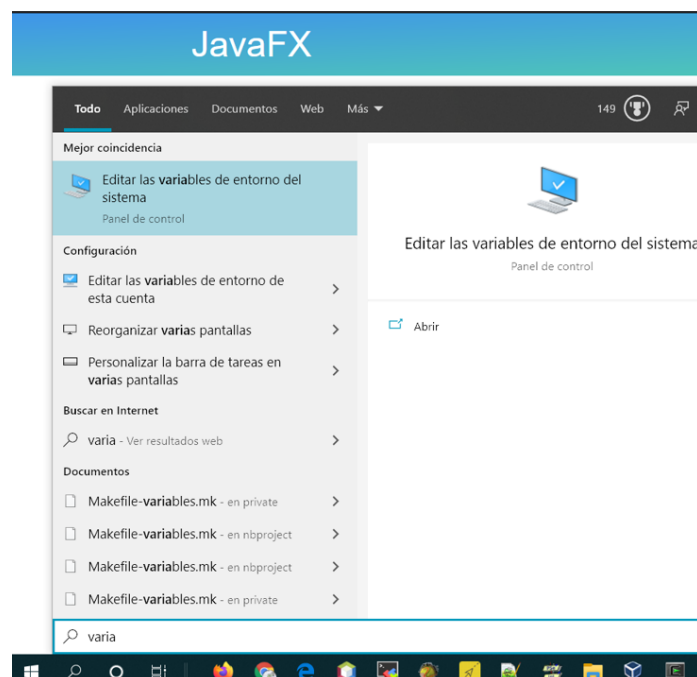
Bajamos también la documentación: **JavaFXDocumentacion**

Gluon				
Products ▾	Developers	Pricing	Services	Insights ▾
Contact ▾				
Product	Version	Platform	Download	
JavaFX Windows x64 SDK	16	Windows x64	Download	[SHA256]
JavaFX Windows x64 jmods	16	Windows x64	Download	[SHA256]
JavaFX Windows x86 SDK	16	Windows x86	Download	[SHA256]
JavaFX Windows x86 jmods	16	Windows x86	Download	[SHA256]
JavaFX Mac OS X SDK	16	Mac	Download	[SHA256]
JavaFX Mac OS X jmods	16	Mac	Download	[SHA256]
JavaFX Linux SDK	16	Linux	Download	[SHA256]
JavaFX Linux jmods	16	Linux	Download	[SHA256]
JavaFX Documentation	16	Javadoc	Download	[SHA256]

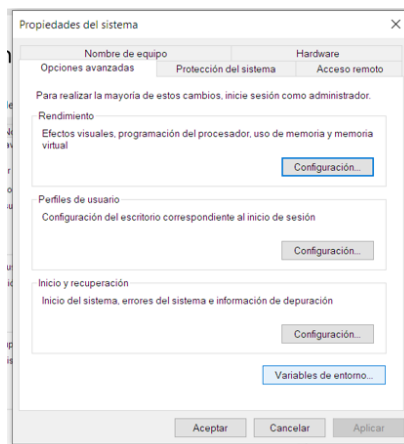
El SDK es un fichero zip. Una vez bajado, lo descomprimos donde nos sea más conveniente. En mi caso:

C:\javafx-sdk-16

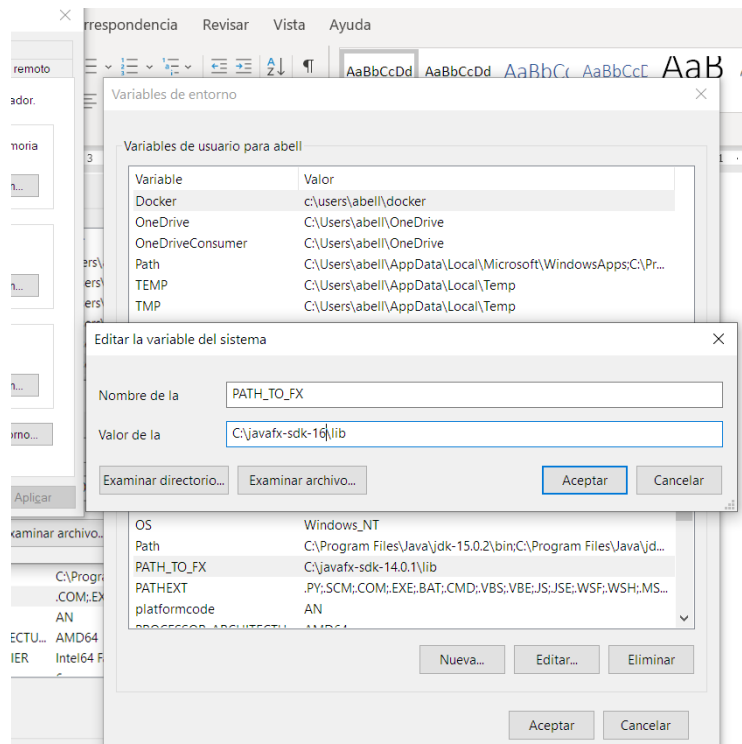
Para su correcto funcionamiento debemos añadir una variable al sistema. Una manera de hacerlo: pulsamos sobre el botón Windows (a la izquierda de la barra de tareas) y escribimos variables. Se muestra:



Pulsamos el botón “Variables de entorno.”



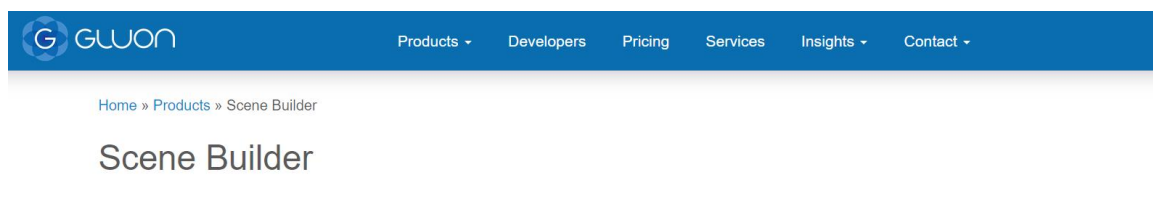
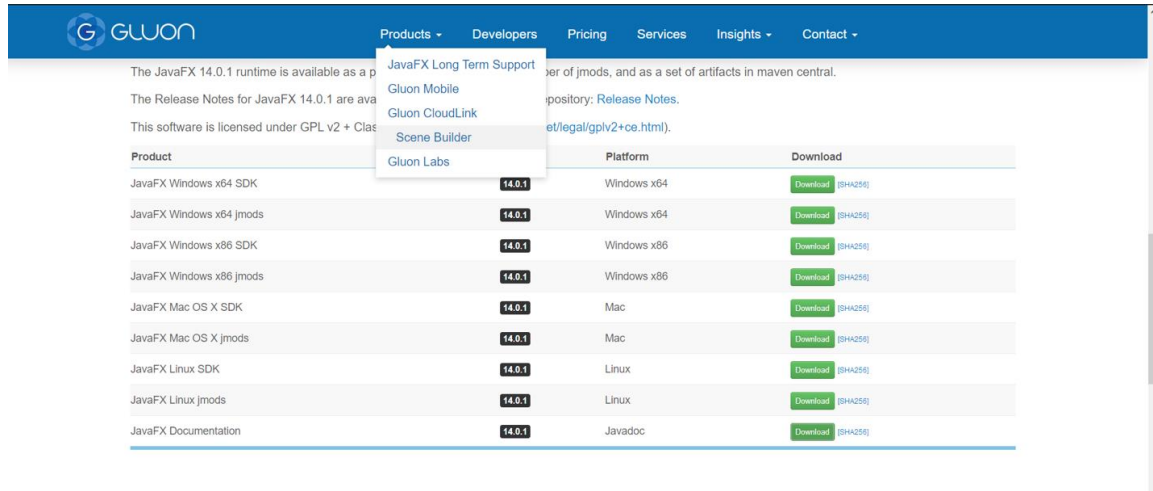
Y añadimos una nueva variable a nivel del Sistema, con el nombre **PATH_TO_FX** y como valor, el directorio donde tenemos el **JavaFX**, y dentro de él, el subdirectorio **lib**



El fichero de documentación los descomprimos bajo la carpeta **C:\javafx-sdk-16**

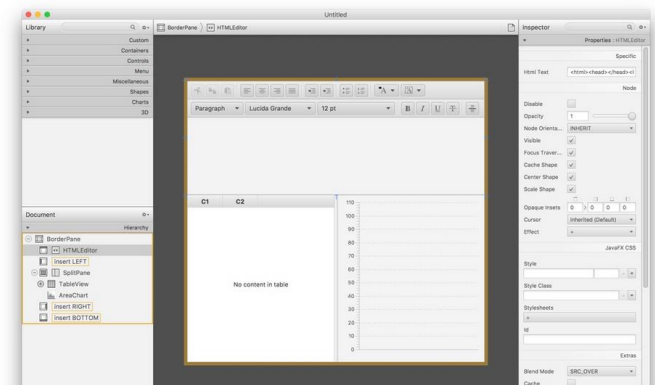
Scene Builder

Dentro de la misma página de Gluon, pulsando sobre la pestaña Products, escogemos la opción de Scene Builder.



Home » Products » Scene Builder

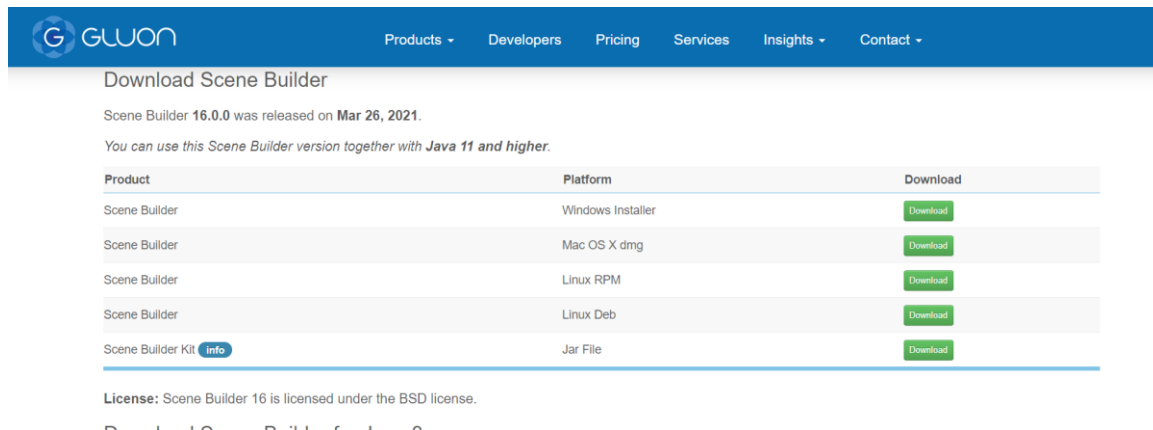
Scene Builder



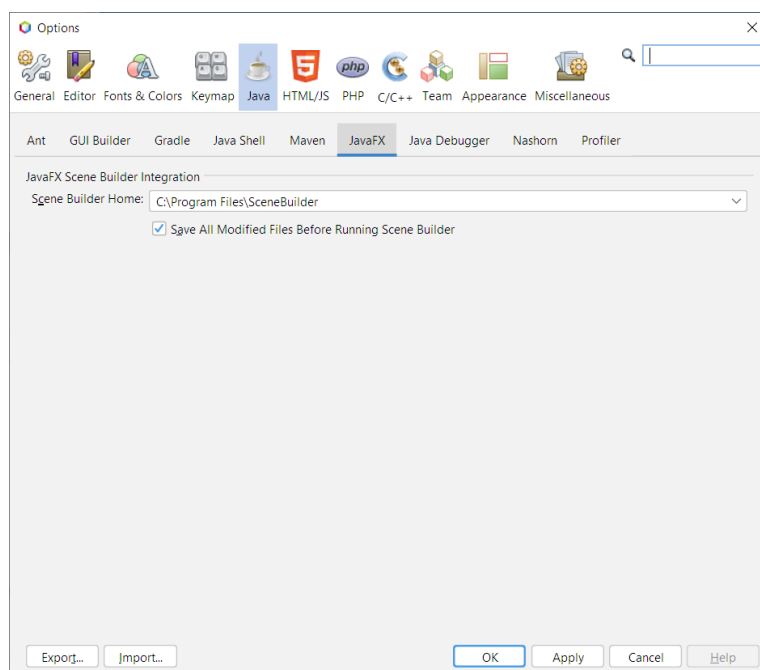
Drag & Drop,
Rapid Application
Development.

[Download Now](#)

Y seleccionamos la correspondiente a nuestra plataforma:



Una vez bajado lo ejecutamos y ya tenemos listo el programa, que se abrirá automáticamente desde NetBeans, aunque también podemos manejarlo de modo independiente. Para realizar la integración con NetBeans vamos dentro de NetBeans a **Tools->Options**, y escogemos Java, y dentro de este, la pestaña **JavaFX**, y seleccionamos el directorio base en el cual hemos instalado Scene Builder



Ahora nos queda configurar nuestro IDE para la utilización de JavaFX. En la siguiente captura se puede ver como desde la propia página se nos indica los pasos a seguir para los principales IDEs: IntelliJ, Eclipse y NetBeans, además de para los principales gestores de proyectos. Aunque hasta ahora hemos utilizado Ant en todos nuestros proyectos, con JavaFX veremos otro gestor de

proyectos, Maven, el cual es junto con Gradle uno de los más utilizados actualmente. Tanto Ant, como Maven con Gradle, vienen integrados en NetBeans.

JavaFX

Getting Started with JavaFX

- Introduction
- Install Java
- Run HelloWorld using JavaFX
- Run HelloWorld via Maven
- Run HelloWorld via Gradle
- Runtime images
- JavaFX and IntelliJ
- JavaFX and NetBeans**
 - Non-modular from IDE
 - Non-modular with Maven
 - Non-modular with Gradle
 - Modular from IDE
 - Modular with Maven
 - Modular with Gradle
- JavaFX and Eclipse
- Next Steps

JavaFX and NetBeans

This section explains how to create a JavaFX application in NetBeans. JavaFX 12.0.2 and Apache NetBeans 11.1 were used for the IDE screenshots.

Download an appropriate JDK for your operating system and set `JAVA_HOME` to the JDK directory. Refer to [Install Java](#) section for more information.

You can create a JavaFX 14 modular or non-modular project and use the IDE tools, Maven or Gradle build tools.

Note: We recommend you to use NetBeans 11.3 or later.

Non-modular projects

IDE

Follow these steps to create a JavaFX non-modular project and use the IDE tools to build it and run it. Alternatively, you can download a similar project from [here](#).

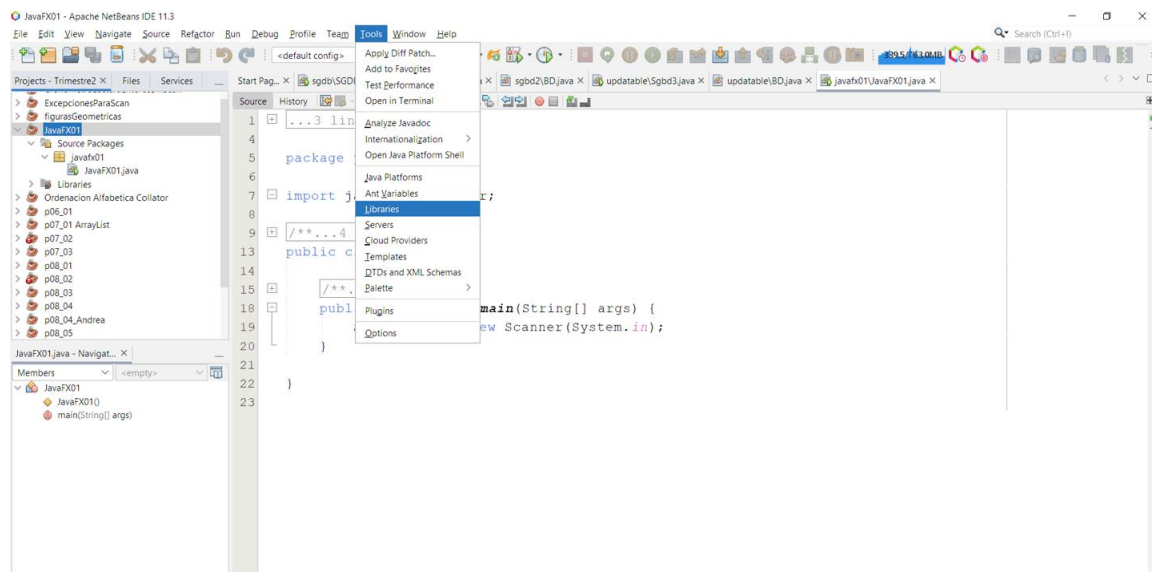
Download the appropriate [JavaFX SDK](#) for your operating system and unzip it to a desired location, for instance

Creando un proyecto en NetBeans con Maven

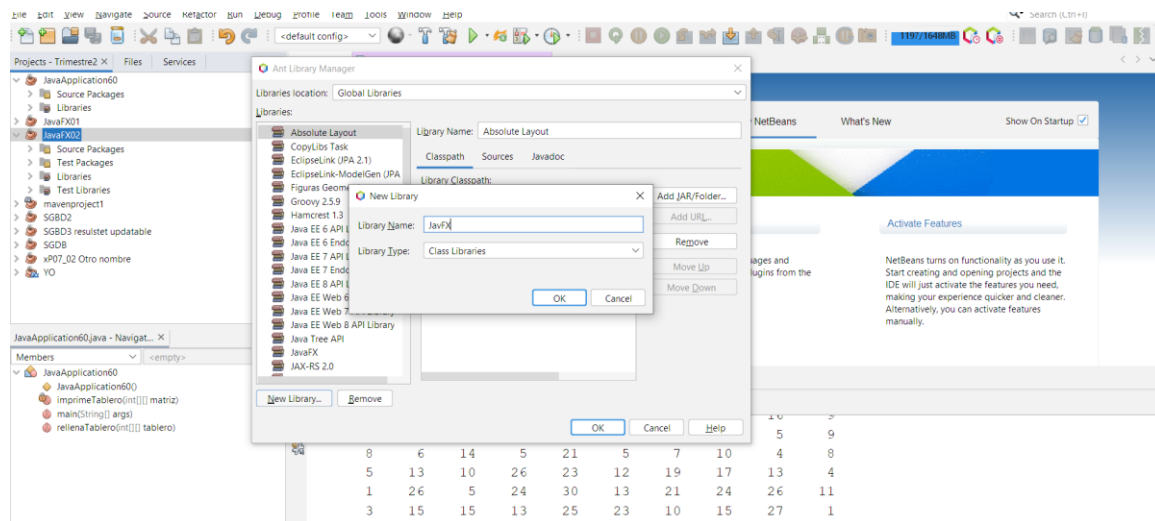
1) Nueva librería

Haremos el proyecto con Maven (en todo caso, por si alguien quiere probar con Ant, recordaros que deberíamos crear una nueva librería). Eso habría que hacerlo una única vez. El resto de los pasos son por proyecto. Vamos a **Tools -> Libraries**

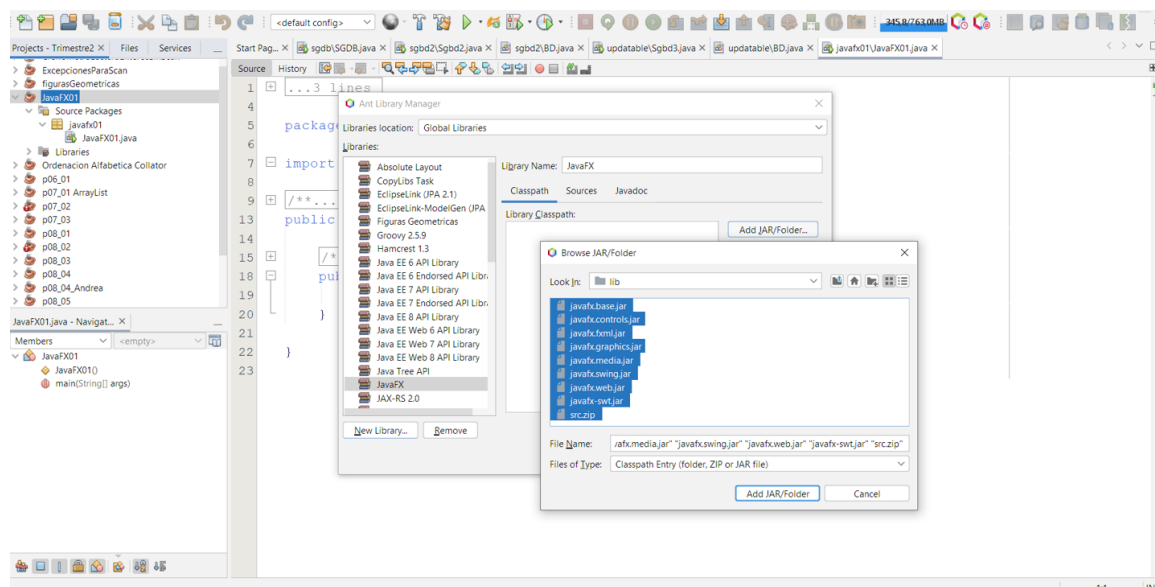
Con Maven no es necesario este primer paso



Pulsamos sobre **New Library**, y rellenamos los campos:



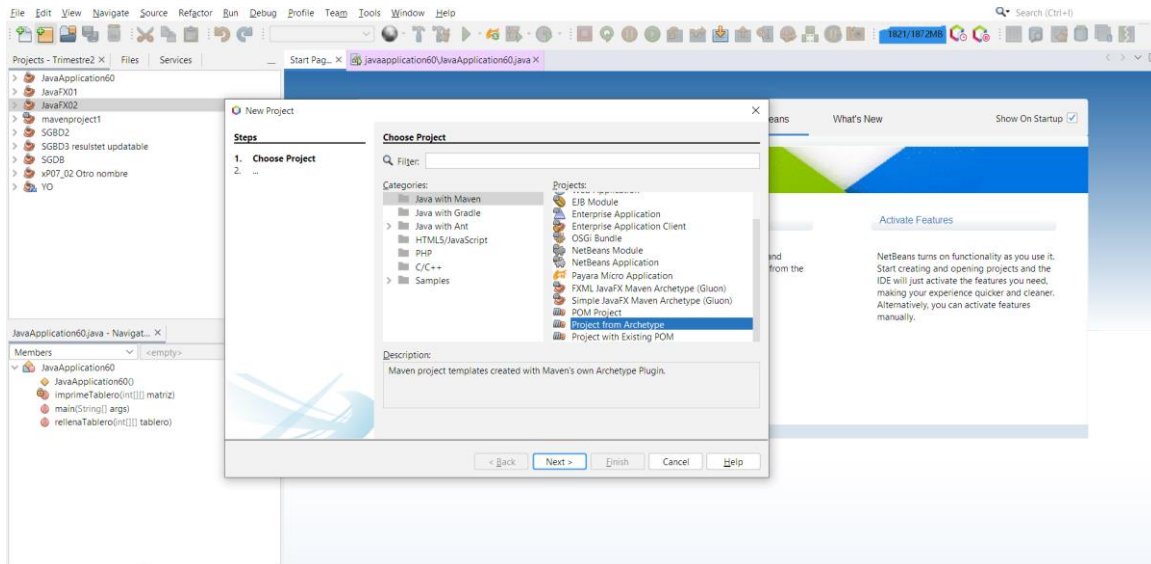
A continuación, pulsando en **Add Jar/Folder**, nos desplazamos al subdirectorio **lib** dentro del directorio donde hemos instalado JavaFX y seleccionamos todos los **.jar**



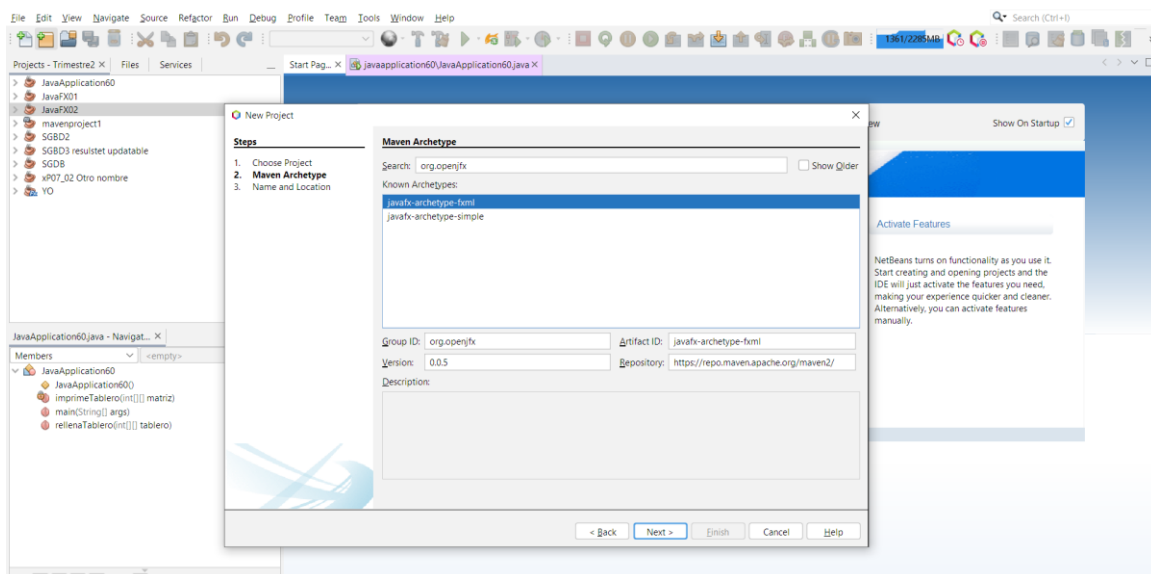
2) Creación del proyecto Maven.

Escogemos New Project, y dentro de el, en caso de existir la opción:
FXML JavaFX Maven Archetype (Gluon)

Si no nos aparece esa opción, escogemos:
Java with Maven -> Project from Archetype



En la siguiente ventana ponemos **org.openjfx** en la casilla de búsqueda. Nos saldrán dos opciones. Escogemos **javafx-archetype-fxml**



En la siguiente pantalla ponemos el nombre del proyecto. En este caso **PruebaFX**

En **Group id**, introducimos un identificador de grupo, por ejemplo, nuestro dominio.

En Versión la versión de la aplicación

Package lo forma por defecto uniendo el **Group id** y el **Project Name** (en minúsculas). Podemos cambiarlo.

En **javafx-versión** ponemos la versión que deseemos utilizar. Ahora mismo la última es la 16.

La versión más reciente de **javafx-maven-plugin-version** es 0.05

Dejamos la Y (yes) en **add-debug-configuration** (añadir información para depuración)

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name:

Project Location:

Project Folder:

Artifact Id:

Group Id:

Version:

Package: (Optional)

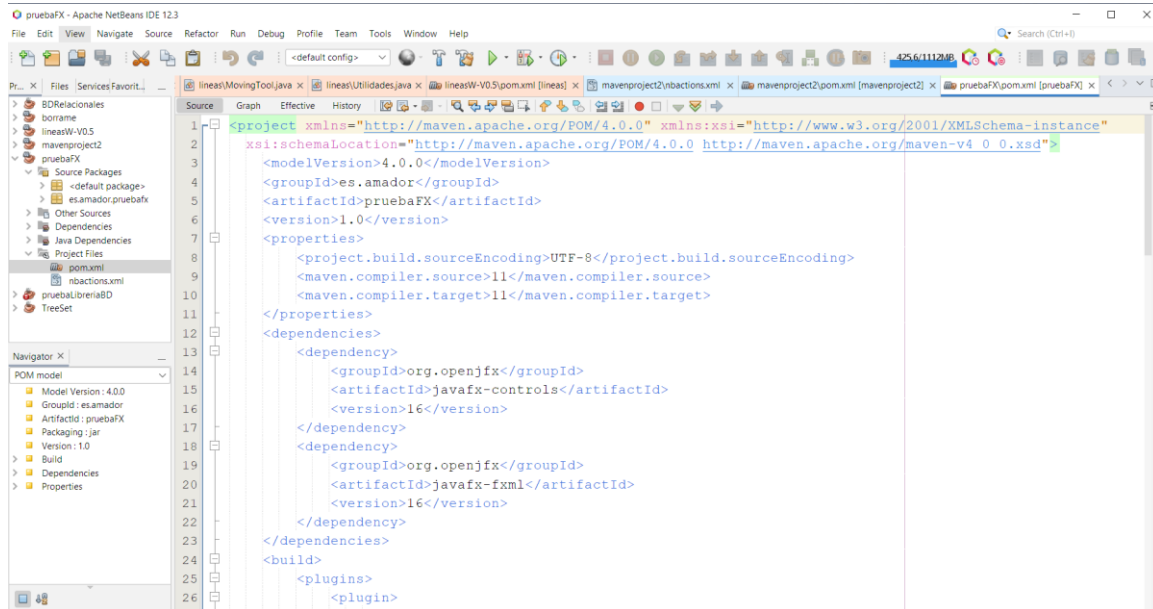
Additional Creation Properties:

Key	Value
javafx-version	16
javafx-maven-plugin-version	0.0.5
add-debug-configuration	Y

< Back Next > **Finish** Cancel Help

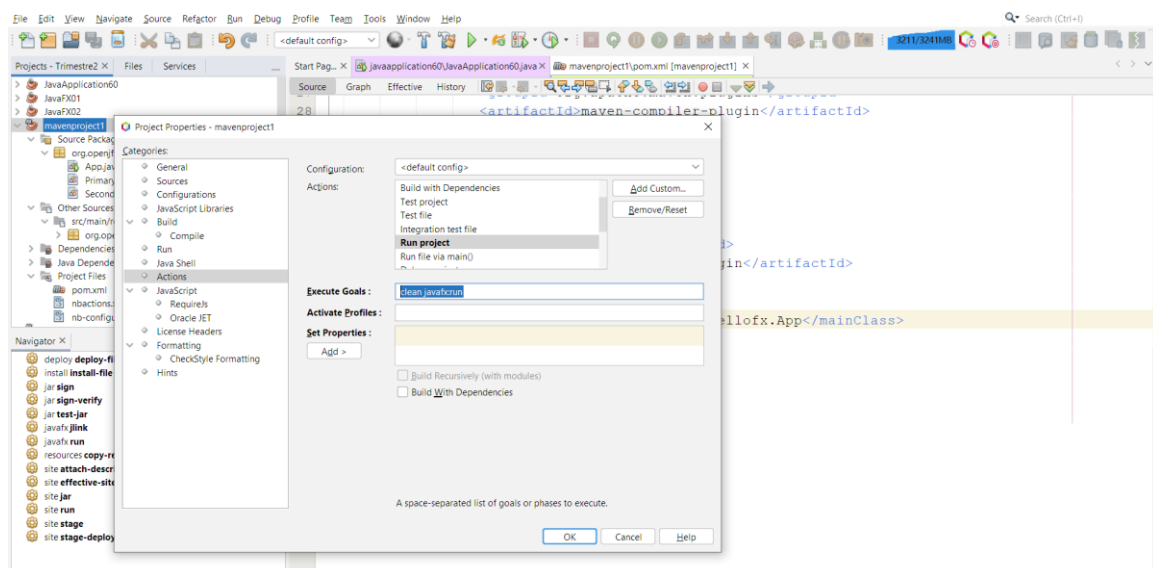
3) Fichero pom.xml

En el fichero **pom.xml** (bajo la carpeta Project Files) se graba la configuración del proyecto.



4) Ejecutar el proyecto

Pulsamos con el botón derecho sobre el proyecto y escogemos **Properties** (la última opción). Escogemos la opción **Actions**. En el recuadro de Actions escogemos **Run Project**, y rellenamos **Execute Goals** con: **clean javafx:run**



Ya podemos ejecutar el proyecto. Este proyecto de prueba que se genera consiste en una ventana con dos vistas y un botón. Cada vez que se pulsa el botón se cambia de vista. Nada que sea muy útil, pero se trata simplemente de comprobar que nuestra instalación está operativa. En el siguiente documento veremos cómo modificar los .fxml y los .java para hacer nuestra propia aplicación.

