Taller UD10_3: Librerias Maven vs Jar

1. Gestión de librerías con `Maven` vs Manualmente (Jar)

1.1. ¿Qué es Maven?

Maven es una herramienta de gestión y comprensión de proyectos, principalmente utilizada en proyectos Java. Su principal función es gestionar las dependencias (librerías y otros componentes que tu proyecto necesita), construir el proyecto y gestionar la configuración del ciclo de vida del desarrollo. Maven utiliza un archivo de configuración denominado `pom.xml` (Project Object Model), donde se especifican las dependencias y otros detalles del proyecto.

1.2. Beneficios de Usar Maven

1. **Gestión Automática de Dependencias:** Maven descarga automáticamente las dependencias del proyecto desde repositorios remotos y las incluye en el proyecto, evitando la necesidad de descargar y agregar manualmente los archivos JAR.
2. **Estandarización del Proyecto:** Proporciona una estructura estándar para los proyectos, lo que facilita la organización y el mantenimiento.
3. **Reproducibilidad:** El uso de `pom.xml` permite que cualquier desarrollador que clone el repositorio tenga exactamente las mismas versiones de las dependencias, asegurando que el proyecto se construya de manera consistente en diferentes entornos.
4. **Integración con IDEs:** Herramientas como IntelliJ IDEA tienen soporte nativo para Maven, lo que facilita la configuración y gestión del proyecto dentro del IDE.
5. **Gestión del Ciclo de Vida del Proyecto:** Maven puede automatizar tareas como compilación, pruebas, empaquetado y despliegue, facilitando la integración continua.

1.3. Diferencias con agregar las librerías directamente al proyecto

1. Gestión de Dependencias: en **Maven** las dependencias se especifican en el archivo `pom.xml`. Maven descarga las dependencias automáticamente desde los repositorios configurados. Mientras que en **Manual** las dependencias se descargan manualmente y se agregan al proyecto, lo que puede llevar a inconsistencias y errores si diferentes desarrolladores utilizan versiones distintas de las librerías.

2. Actualización de Dependencias: en **Maven** actualizar una dependencia es tan simple como cambiar la versión en el `pom.xml`. Maven se encarga de descargar la nueva versión. Por contra, en **Manual** requiere descargar la nueva versión manualmente y reemplazar los archivos JAR antiguos en el proyecto.

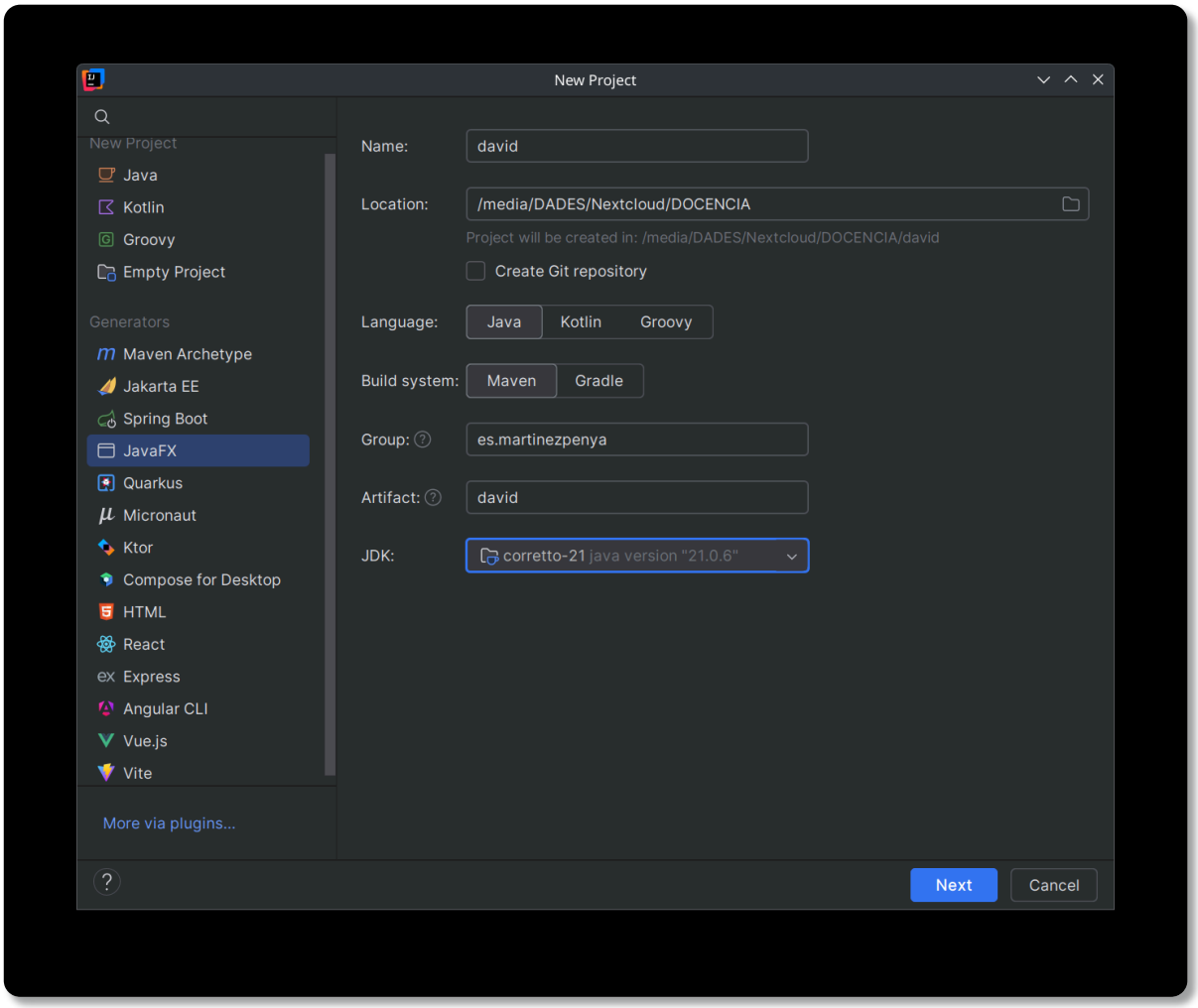
3. Estructura del Proyecto: en **Maven** proporciona una estructura de proyecto estandarizada, lo que facilita la comprensión y la navegación del proyecto. Mientras que en **Manual** la estructura del proyecto puede variar según el desarrollador, lo que puede llevar a confusión y desorganización.

4. Repositorios de Dependencias: en **Maven** utiliza repositorios remotos (como Maven Central) para buscar y descargar dependencias. También permite configurar repositorios internos. Pero en **Manual** las dependencias deben ser gestionadas y almacenadas manualmente, lo que puede ser engorroso y propenso a errores.

5. Integración con Herramientas de Construcción: Maven se integra fácilmente con herramientas de construcción y CI/CD, permitiendo la automatización de tareas como compilación, pruebas y despliegue. Por otro lado en **Manual** se requieren scripts personalizados y configuración adicional para integrar tareas de construcción y despliegue.

2. Crea un proyecto de JavaFx (con Maven)

Cuando creas un proyecto en Intellij, el mismo IDE ofrece un tipo de proyecto JavaFx:

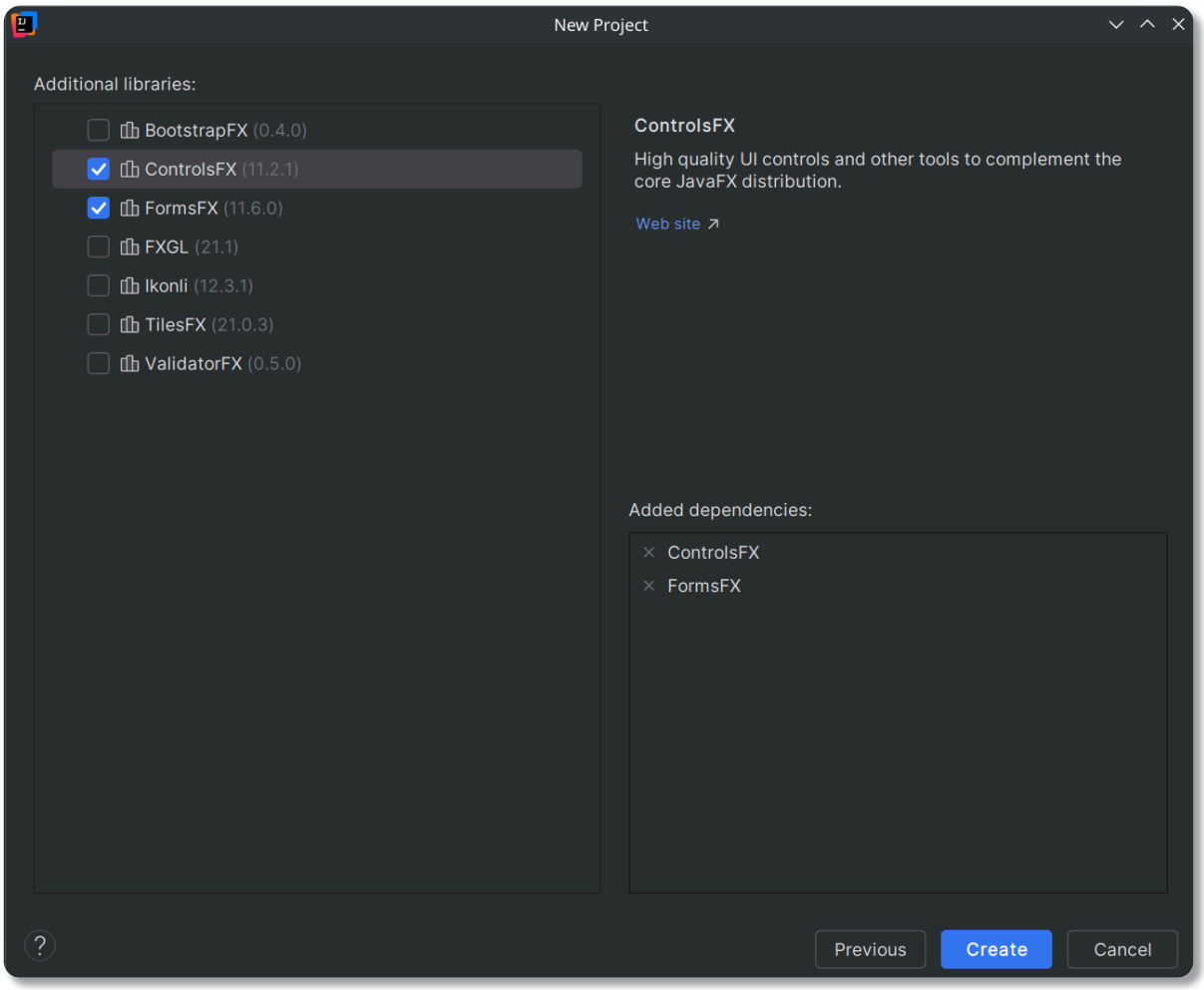


Importante Fíjate que al elegir el tipo de proyecto JavaFX a la parte derecha se selecciona el gestor de librerías Maven.

En el siguiente tema veremos como gestionar manualmente Maven, de momento dejaremos que IntelliJ genere el projecto Maven para JavaFx por nosotros.

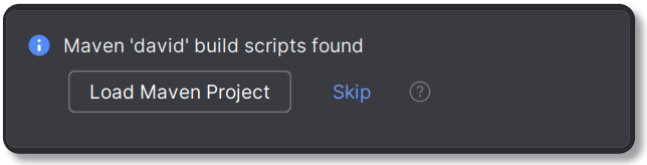
Solo debes fijarte en escribir tu nombre y apellidos como en la imagen y seleccionar la versión 21 del jdk.

En la siguiente ventana debes marcar al menos estas dos opciones:



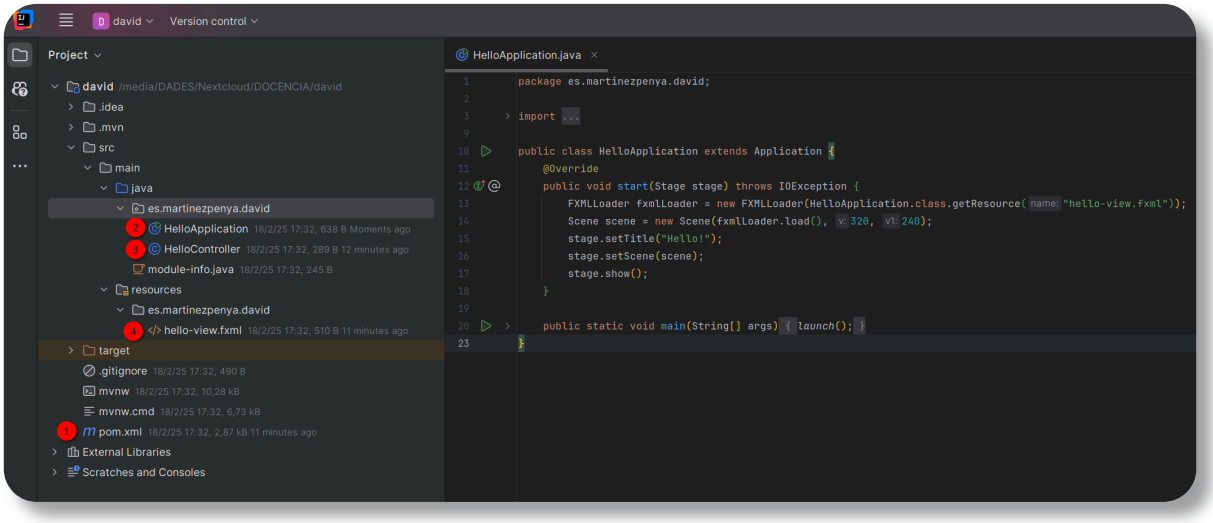
Y por último pulsar el botón [Create]

Si esperas unos segundos aparecerá una notificación en la parte inferior derecha:



Debes elegir la opción Load Maven Project, y esperar a que procese las dependencias solicitadas.

Estructura del proyecto JavaFx recien creado con el asistente de IntelliJ:



En la imágen anterior puedes encontrar:

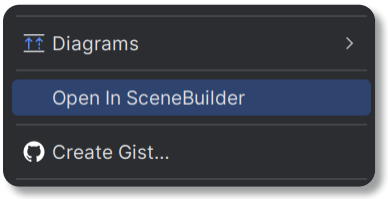
1. fichero pom.xml que gestiona las dependencias del proyecto maven.
2. HelloApplication clase que contiene el main de la aplicación JavaFx
3. HelloController clase con el controlador de la aplicación

4. hello-view.fxml archivo que contiene el formulario (vista) de la apliación

El proyecto sigue el patrón MVC que hemos visto en teoria, solo que es tan simple que no tiene ningún Modelo.

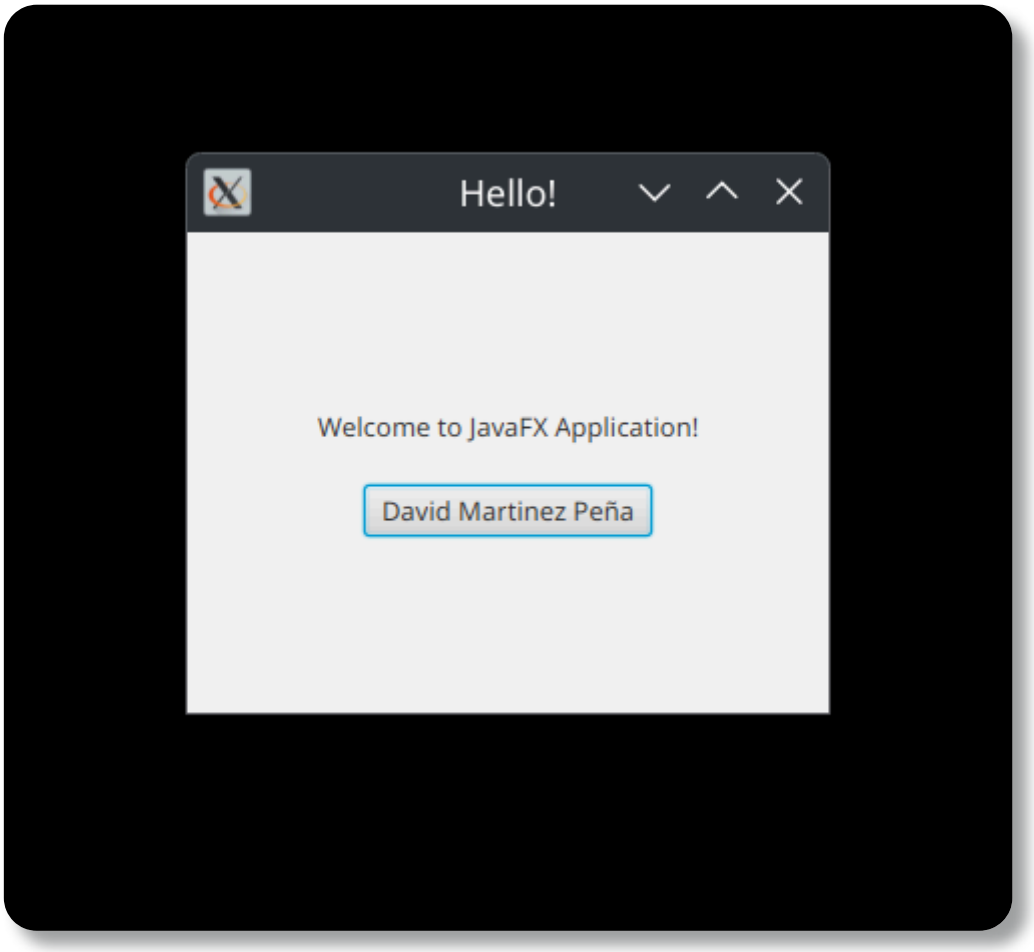
3. Modificar el formulario con SceneBuilder

Si has configurado correctamente SceneBuilder en tu ordenador para usarlo desde IntelliJ, deberias poder pulsar con el botón derecho del ratón sobre el archivo 4 (hello-view.fxml) y elegir la opción Abrir en SceneBuilder:



4. Actividades

Usa el asistente de IntelliJ para crear proyecto JavaFX con maven. Modifica el formulario con SceneBuilder y cambia el texto del botón por tu nombre y apellidos:



Genera un zip con el proyecto de IntelliJ (o la carpeta src del IDE que uses). Envía el archivo zip a la tarea de Aules.

5. Fuentes de información

- Wikipedia
- Programación (Grado Superior) - Juan Carlos Moreno Pérez (Ed. Ra-ma)
- Apuntes IES Henri Matisse (Javi García Jimenez?)
- Apuntes AulaCampus
- Apuntes José Luis Comesaña
- Apuntes IOC Programació bàsica (Joan Arnedo Moreno)
- Apuntes IOC Programació Orientada a Objectes (Joan Arnedo Moreno)
- FXDocs
- https://openjfx.io/openjfx-docs/
- https://arturoblasco.github.io/pr