DIGITALHOUSE

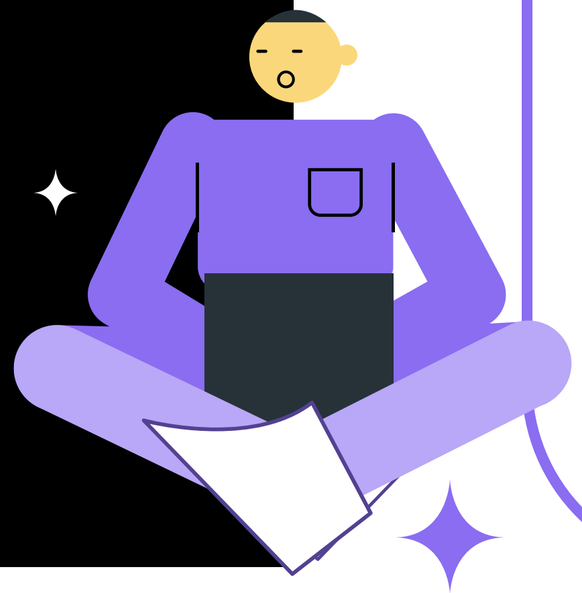
Esqueleto de carpetas proyecto con Maven + Selenium Webdriver

Certified Tech Developer
The Ultimate Degree

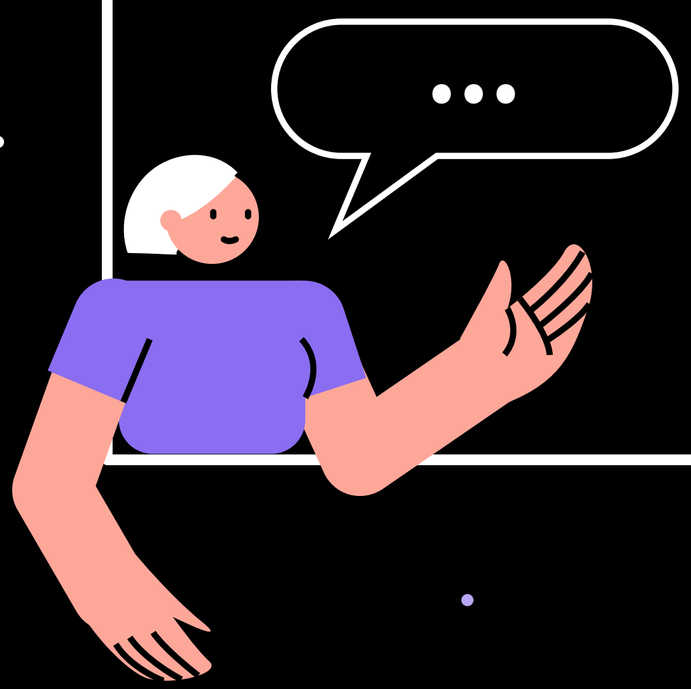
Índice

01 Introducción

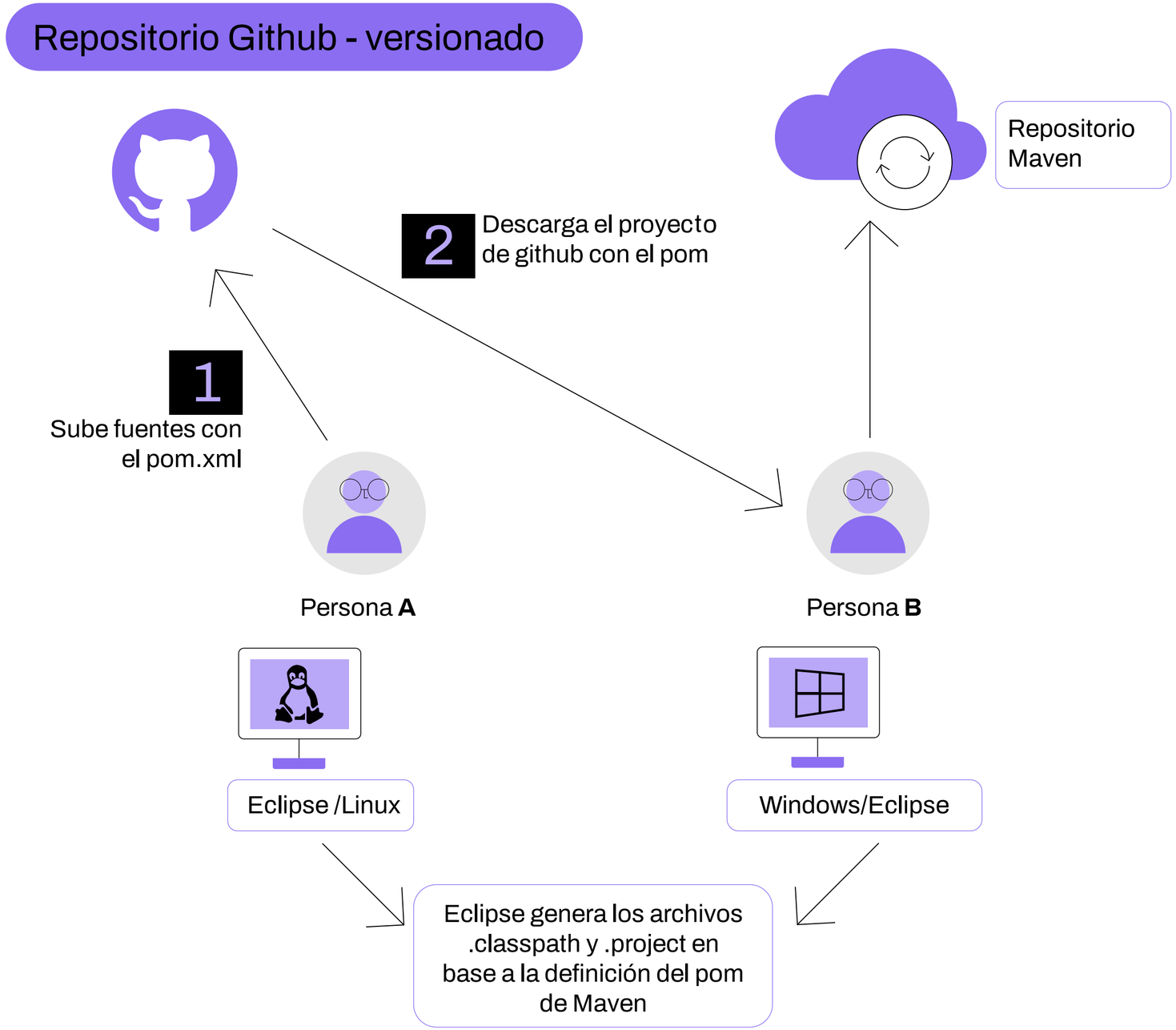
Antes de comenzar a trabajar con nuestro primer proyecto en Selenium debemos conocer cómo es el **esqueleto básico** en común de todos los proyectos.



Para ello, tengamos en cuenta que vamos a utilizar **Maven**, que nos permite gestionar nuestros proyectos con Selenium.



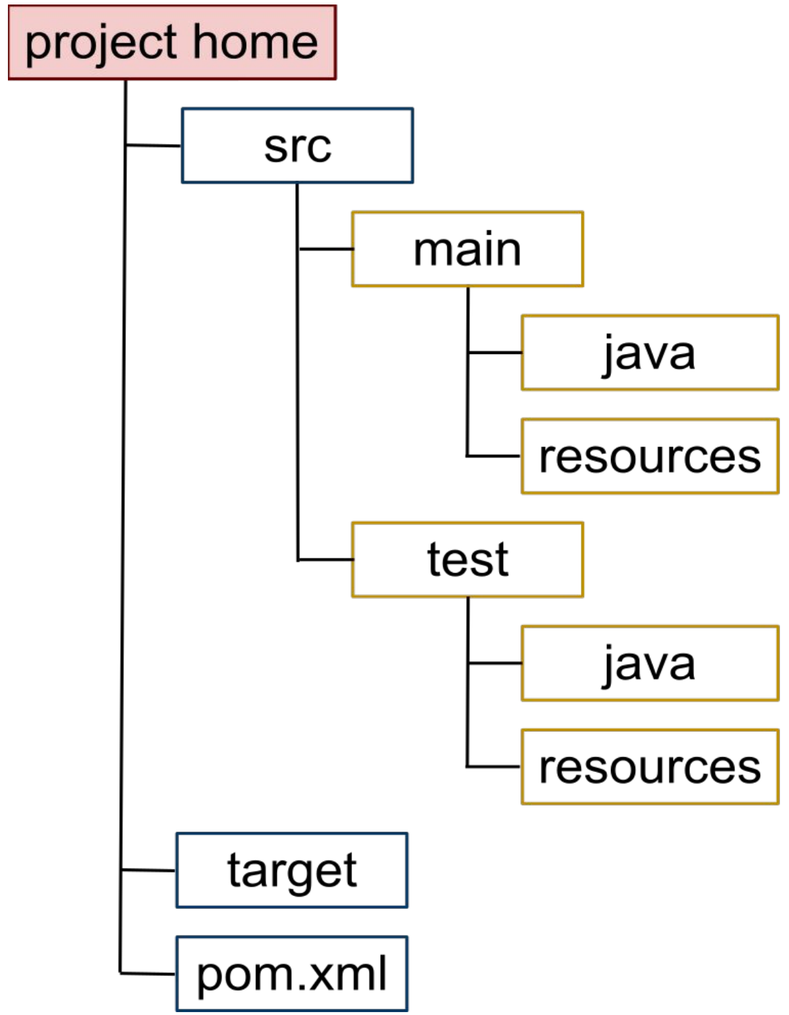
¿Cómo funciona **Maven**?



Adaptado de:
https://wiki.ugbar.org/wiki/articles/maven.html

Estructura de un proyecto **Maven**

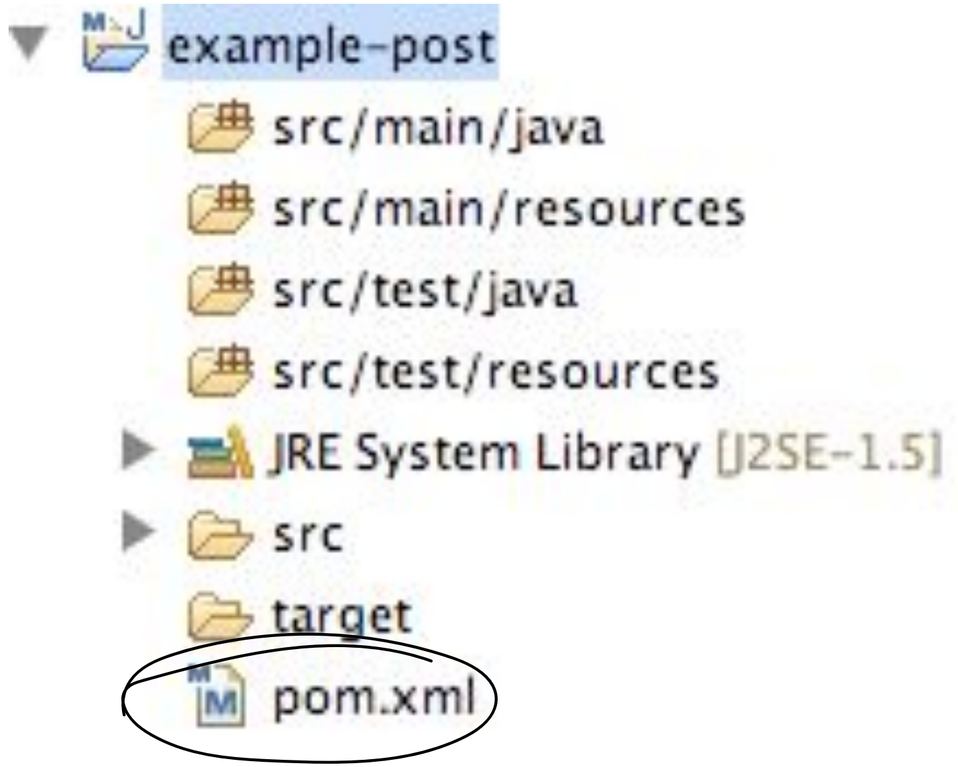
Al crear un proyecto de **Maven**, automáticamente se nos generará una estructura de carpetas muy concreta que ya viene predefinida. Cada uno de los directorios tiene una funcionalidad dentro del proyecto.



02 Archivo pom.xml

pom.xml

Podemos decir que este archivo es **el corazón del proyecto**. Project Object Model (POM) es el encargado de gestionar y construir los proyectos, contiene el listado de dependencias que son necesarias para que el proyecto funcione, plugins, etc. Toda la información del proyecto está basada en este archivo. Tiene extensión .xml.



pom.xml

El archivo **pom** con la estructura más básica posible, contendrá:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project>
 <modelVersion>4.0.0</modelVersion>
 <groupId>com.mycompany.app</groupId>
 <artifactId>my-app</artifactId>
 <version>1</version>
</project>
```

Project: root. Es el esqueleto principal del archivo pom

ModelVersion: se debe establecer en 4.0.0. Es la última versión de Maven

GroupID: es el id del grupo del proyecto.

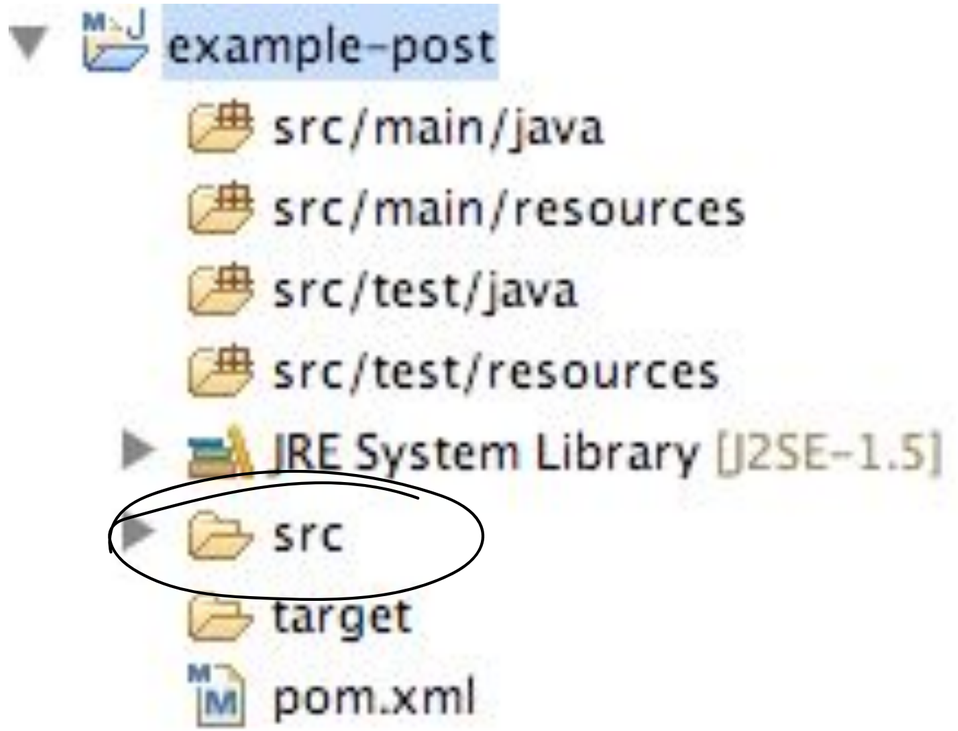
ArtifactId: es el id del artefacto, es decir del proyecto.

Versión: es la versión del proyecto.

03 Directorio src

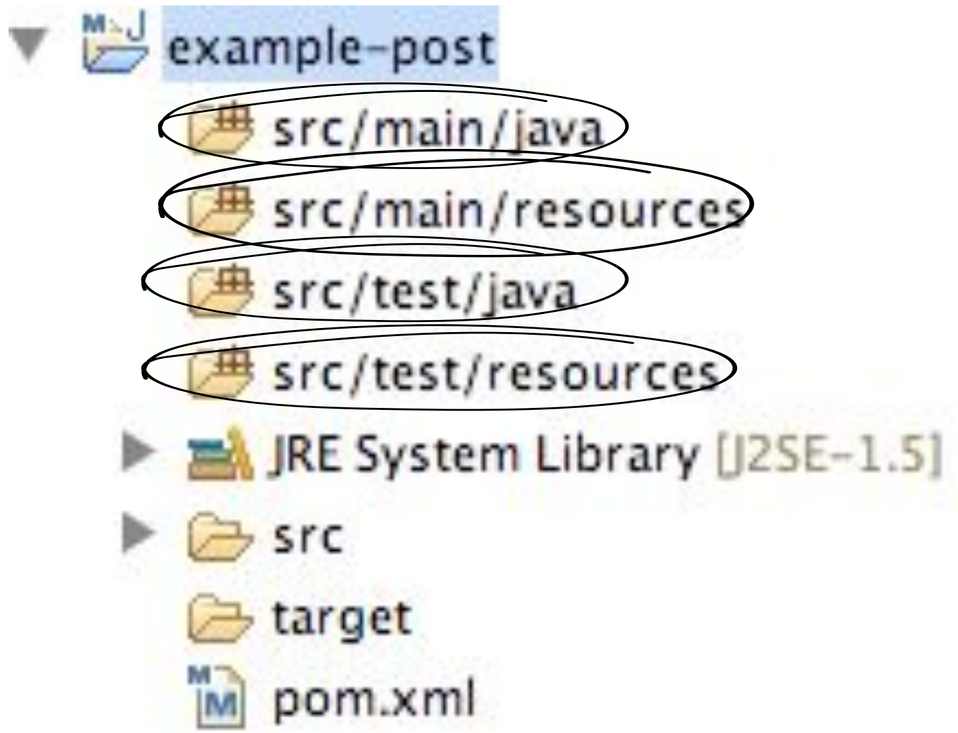
Directorio src

El directorio **src** contiene todo el material de origen para construir el proyecto, su sitio, sus pruebas, etc.
Tiene dos subdirectorios principales: **main** y **test**.



Directorio src

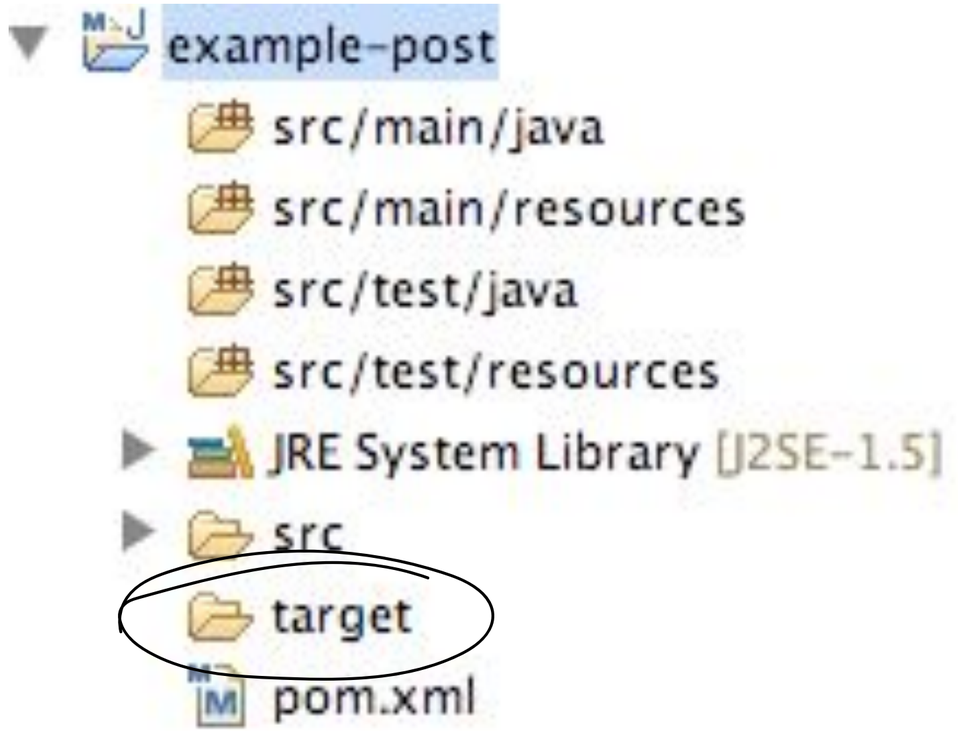
- **src/main/java:** contiene el código fuente. Aquí escribiremos —o estarán, si el proyecto es cargado— las clases con extensión .java. El contenido de este directorio se conoce bajo el nombre de módulo.
- **src/main/resources:** contiene los recursos estáticos —XML, propiedades, imágenes...— que necesita nuestro módulo para funcionar correctamente.
- **src/test/java:** contiene los ficheros de pruebas —testing— para verificar el correcto funcionamiento del módulo.
- **src/test/resources:** almacena los archivos que genera Maven al usar los comandos —compile, package...—.



04 Directorio target

Directorio target

El directorio **target** se utiliza para albergar todos los resultados de la compilación. Incluso contiene los resultados de la compilación y los informes de las pruebas.



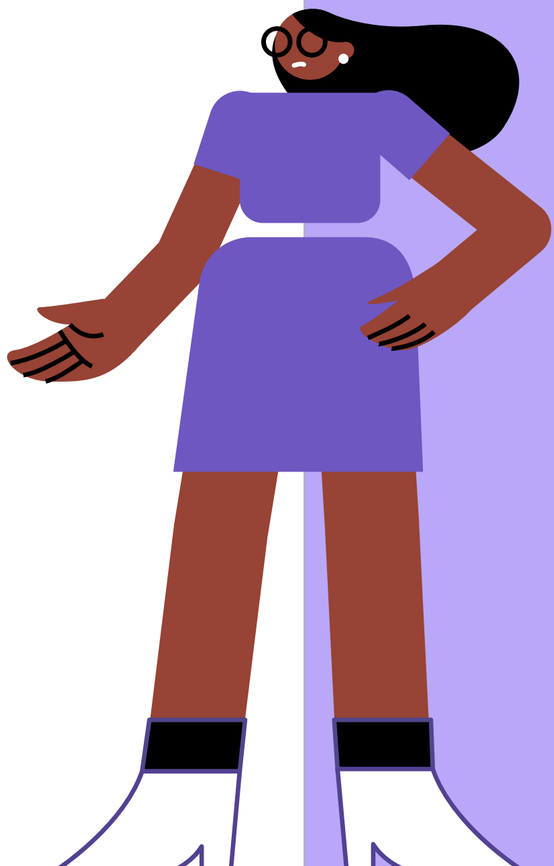
05 Dependencias

Dependencias

Las **dependencias** son componentes que nuestro software necesitará para su correcta ejecución durante algún ciclo de vida.

Si descargamos un proyecto y lo implementamos en nuestro sistema, Maven obtendrá las dependencias del proyecto que son necesarias o bien desde el repositorio local, o desde un repositorio remoto.

Esto permite a los usuarios de Maven **reutilizar las dependencias entre proyectos** para garantizar que los problemas de compatibilidad con versiones anteriores se resuelvan adecuadamente.



Anatomía de JUnit como dependencia

Al sumar el **JUnit** como dependencia se verá:

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13.2</version>
    <scope>test</scope>
</dependency>
```

Repositorio de dependencias

Un repositorio bastante popular de dependencias a nivel mundial cuando estamos usando Maven es:

DIGITALHOUSE

¡Muchas gracias!