Juan Sebastián Velandia

Juan Felipe Ochoa

Lab02

1. Maven contribuye a:

* Administrar dependencias de forma automatizada.
* Estandarizar la organización de los proyectos.
* Automatizar los procedimientos de compilación, evaluaciones y embalaje.
* Integrar constante con herramientas como Jenkins y otras similares.
* **validate** → Verifica si el proyecto es correcto y si tiene todas las dependencias necesarias.
* **compile** → Compila el código fuente del proyecto.
* **test** → Ejecuta pruebas unitarias.
* **package** → Genera el artefacto empaquetado (por ejemplo, un .jar o .war).
* **verify** → Verifica que el paquete cumple con los criterios de calidad.
* **install** → Instala el artefacto en el repositorio local.
* **deploy** → Sube el artefacto a un repositorio remoto para compartirlo.

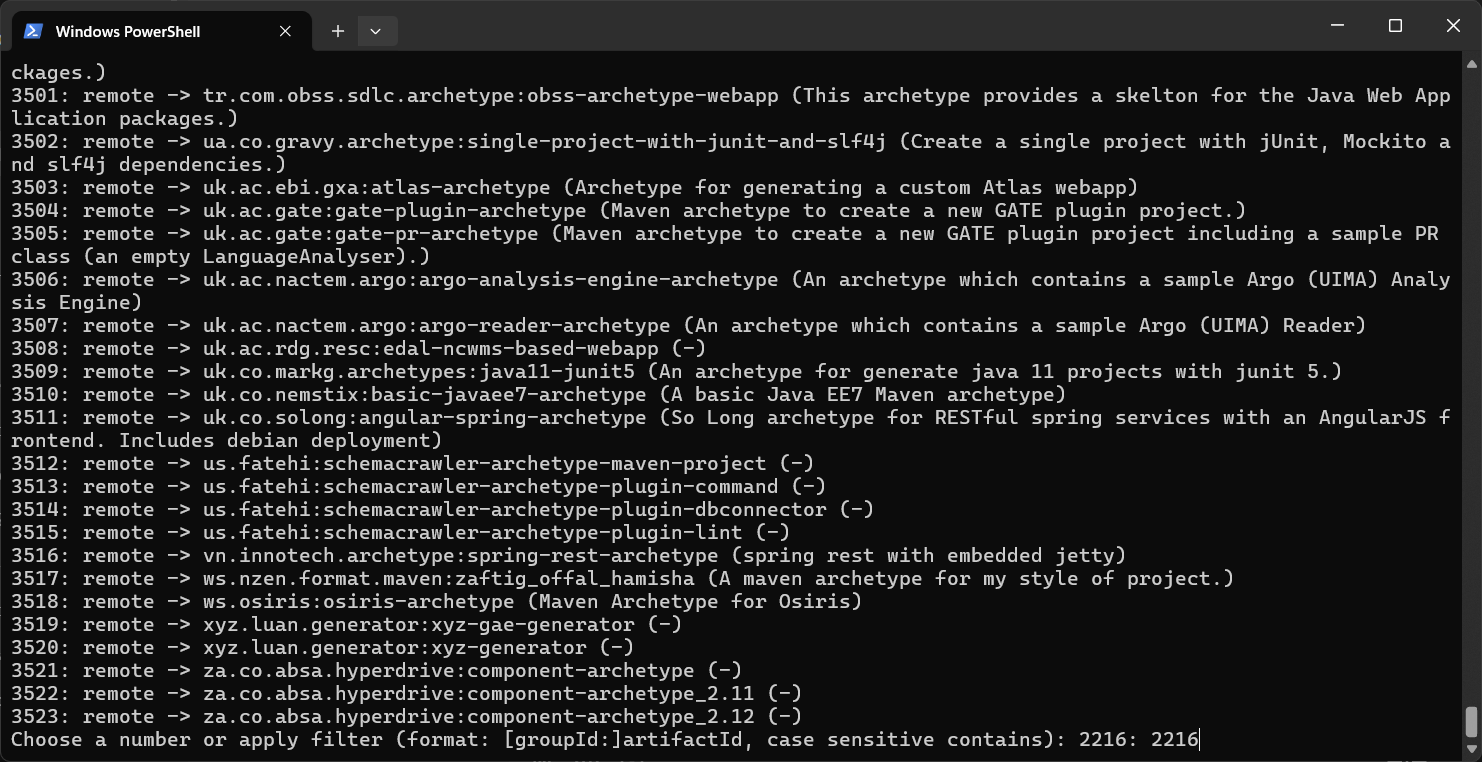
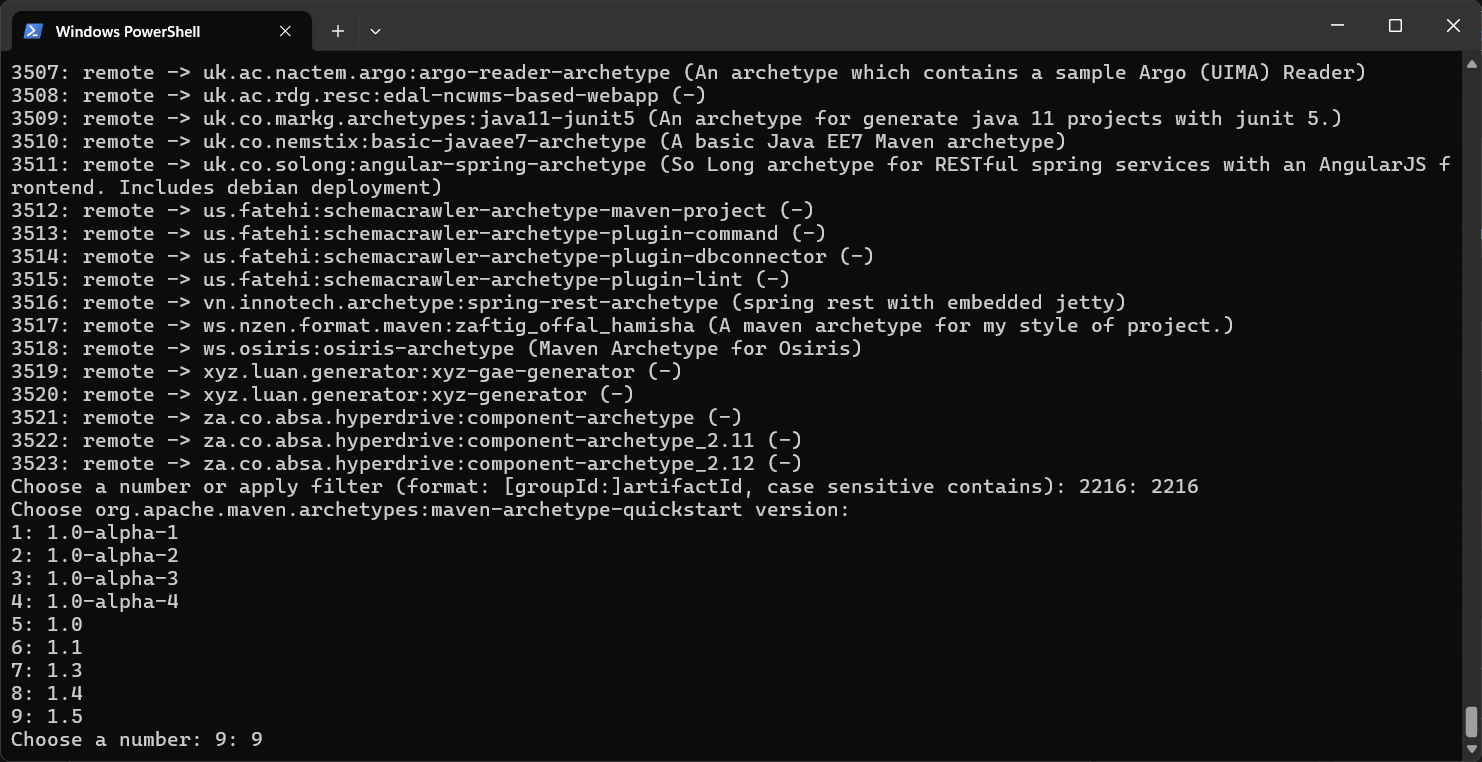
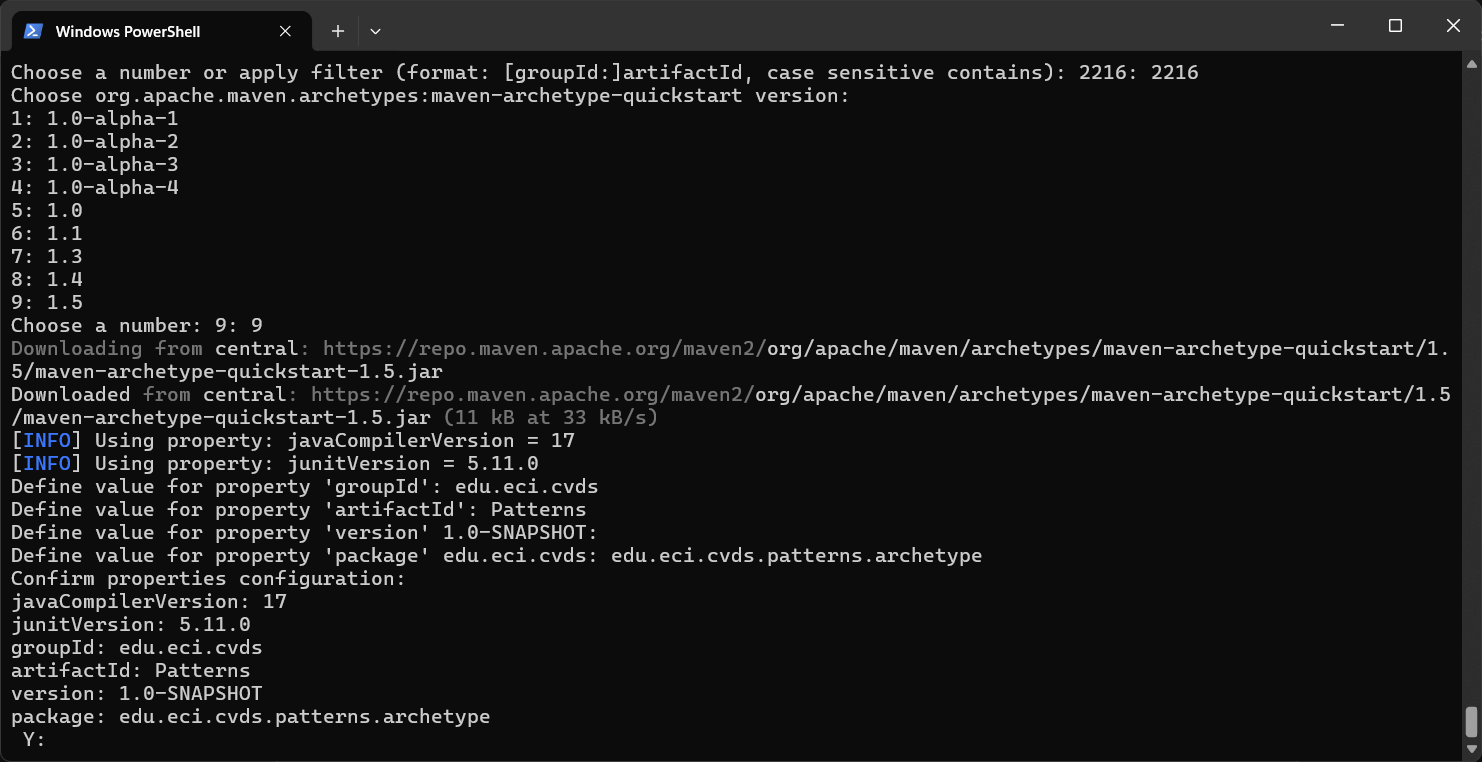
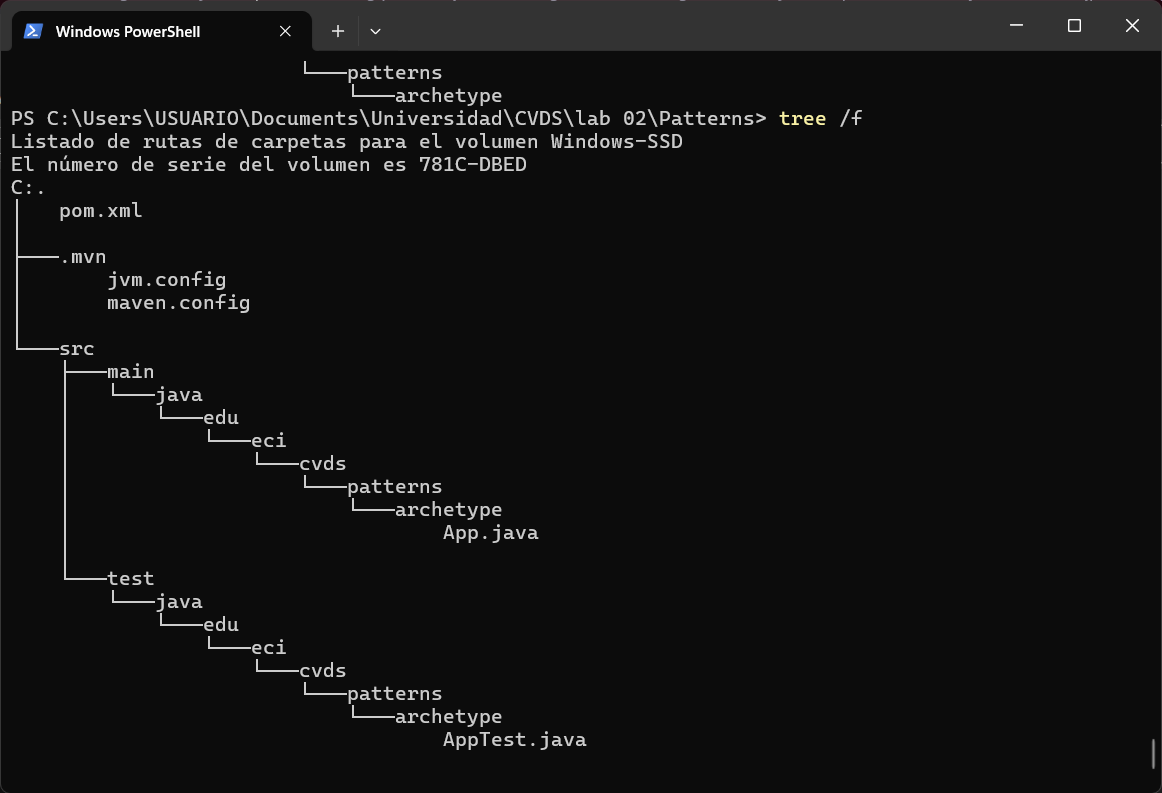
Cada fase se ejecuta en orden y depende de la anterior.

1. Son el conjunto de fases que se deben seguir para pasar un crear un proyecto, hay 3 tipos de ciclos;

* Default: Es el ciclo predeterminado que se encarga del control de la construcción, empaquetamiento y despliegue del proyecto
* Clean: Controla la limpieza del espacio de trabajo, es decir, limpia los archivos generados en compilaciones previas
* Site: Es el encargado de crear la página con la respectiva documentación del proyecto

1. Los plugins en Maven extienden sus funcionalidades y permiten ejecutar tareas específicas dentro del ciclo de vida.
2. Es el principal registro y repositorio de software para componentes, bibliotecas y marcos de Java, así como para lenguajes de Java Virtual Machine (JVM) y sirve para que los desarrolladores no tengan que buscar, descargar y configurar manualmente cada librería, si no solo añadirla al pom.xml

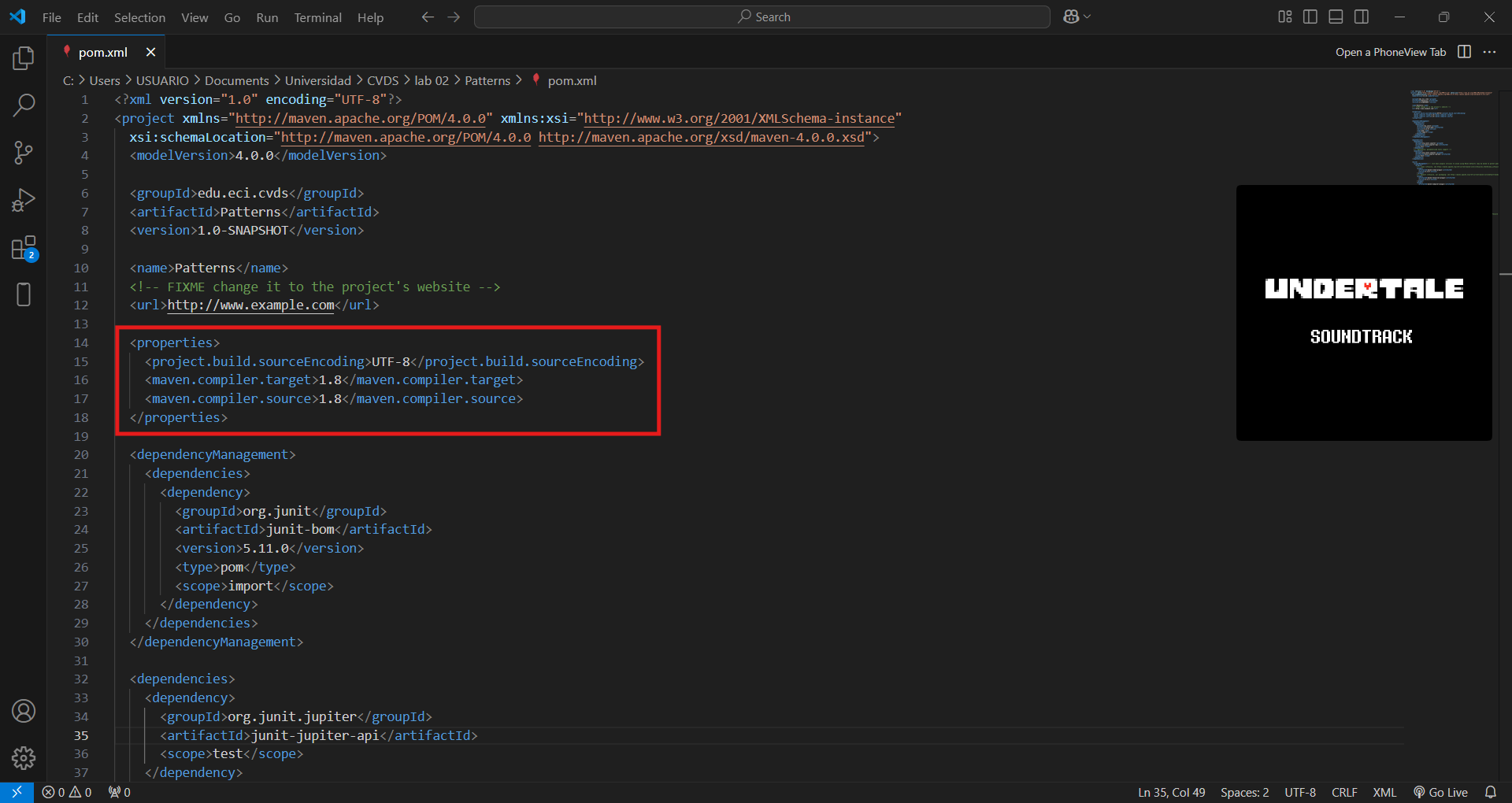
### CREAR UN PROYECTO CON MAVEN

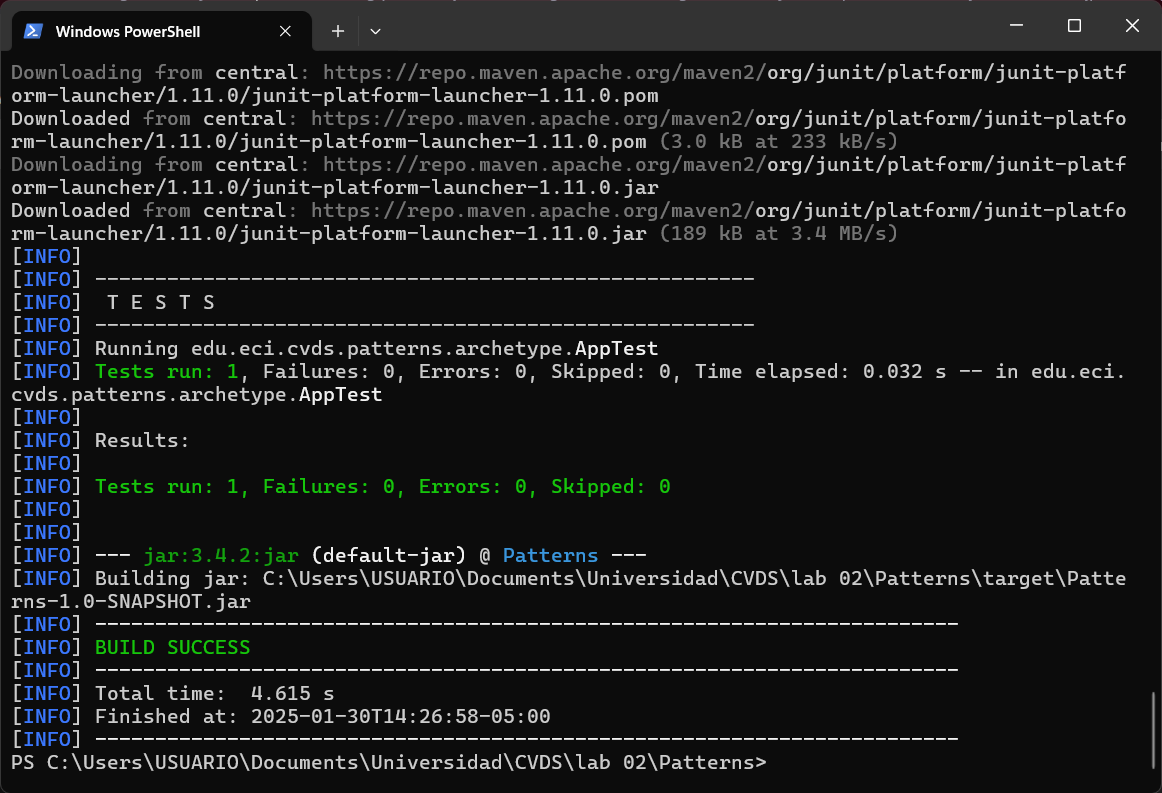
1. Se crea una carpeta donde va a reposar nuestro proyecto, para luego abrir la terminal y abrir la carpeta
2. Luego se ejecuta el comando mvn archetype:generate
3. Despues de ejecutado, nos pedira el arquetipo de proyecto maven, para este caso, usaremos 2216 que es el que generalmente se usa para proyectos java  
   
4. Luego seleccionaremos la version, la cual la sugerida es la ultima:
5. Luego daremos los parámetros solicitados en el laboratorio:
6. Y por ultimo comprobamos el tree que es exactamente igual al del laboratorio:

AJUSTAR ALGUNAS CONFIGURACIONES EN EL PROYECTO

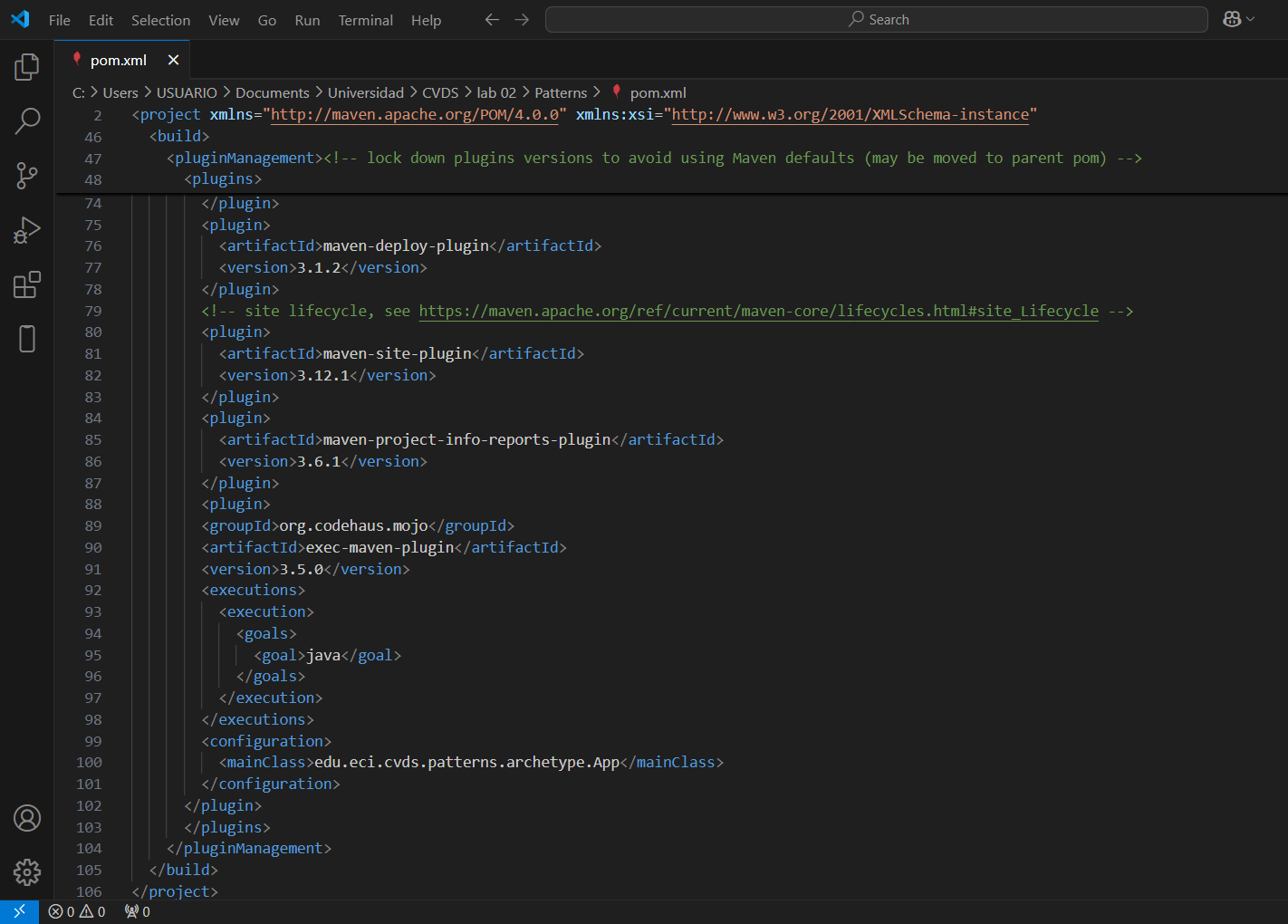
Para devolver la versión del JDK a java 8 tenemos que:

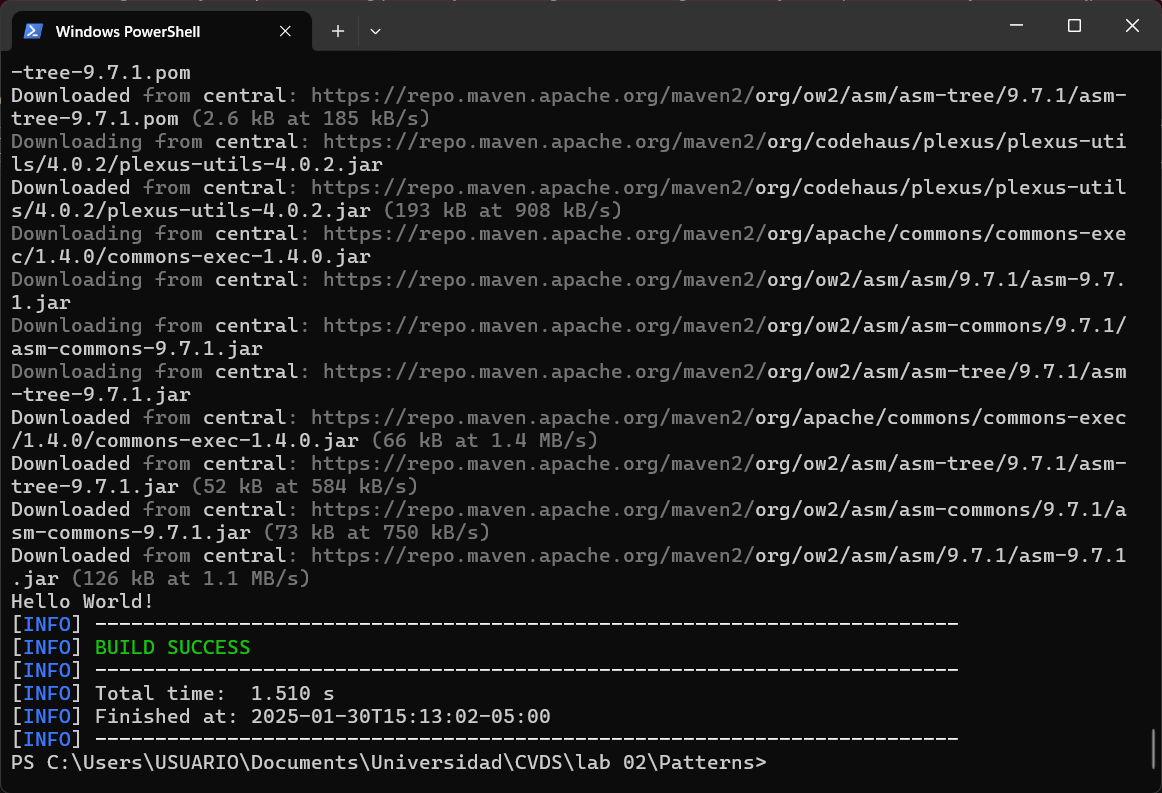
1. Editar el archivo pom.xml de la siguiente manera:



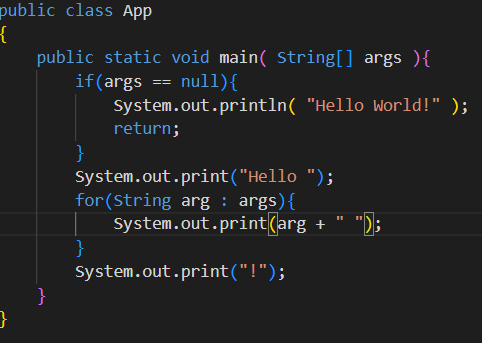
1. Luego ejecutamos el comando mvn package para compilar:

Preguntas:

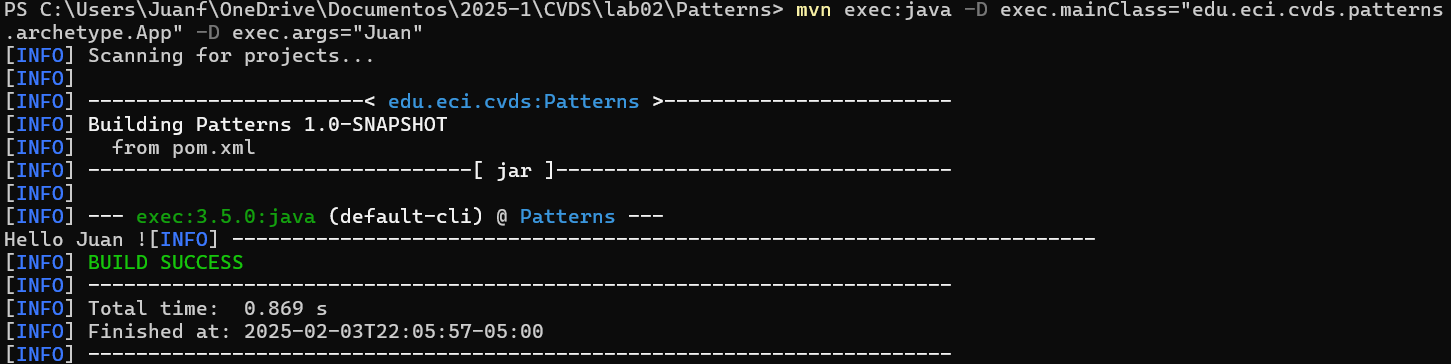
* El comando package sirve para compilar el código, ejecutar las pruebas unitarias, y empaquetar. Otros parámetros que se pueden enviar a este comando son, por ejemplo, clean, validate, compile, test, install, deploy, site, entre otros.
* Para esto tenemos que modificar el pom.xml añadiendo:
* Luego ejecutamos el comando mvn exec:java



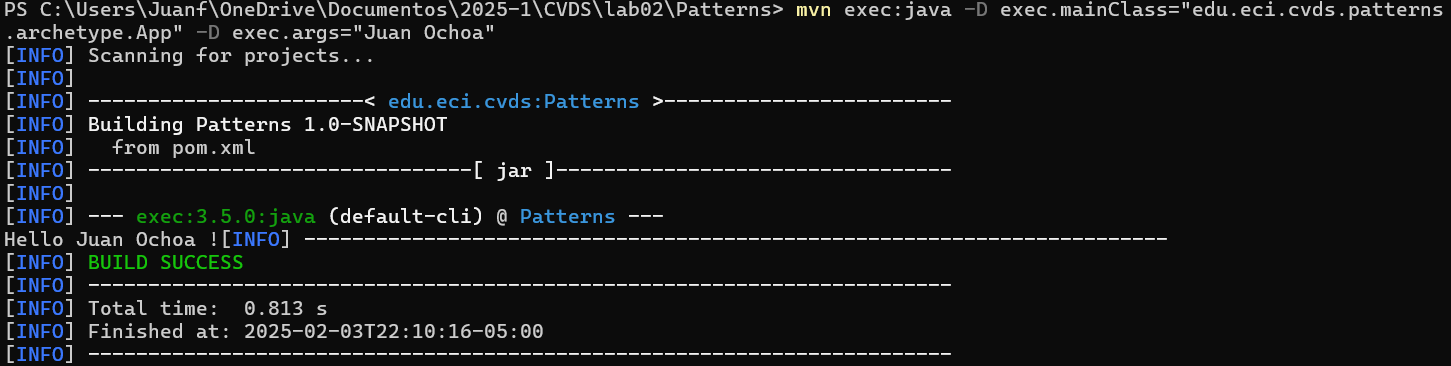
Pasamos a la ejecución del programa con argumentos. Para ello se cambiará el código del main:



Y se ejecutará con el siguiente comando y su respectiva respuesta:

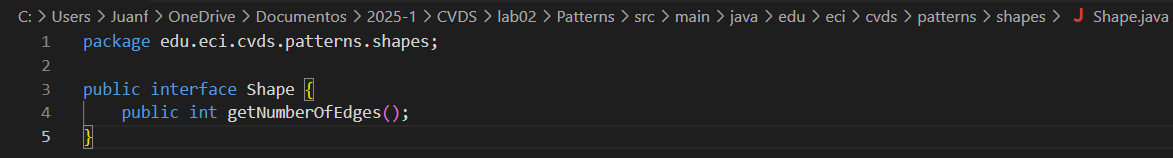


Y para que aparezca nombre y apellido, solo se le agregará un segundo argumento con el apellido:

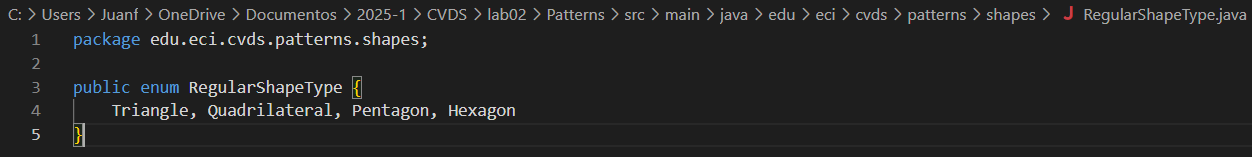


**HACER EL ESQUELETO DE LA APLICACIÓN**

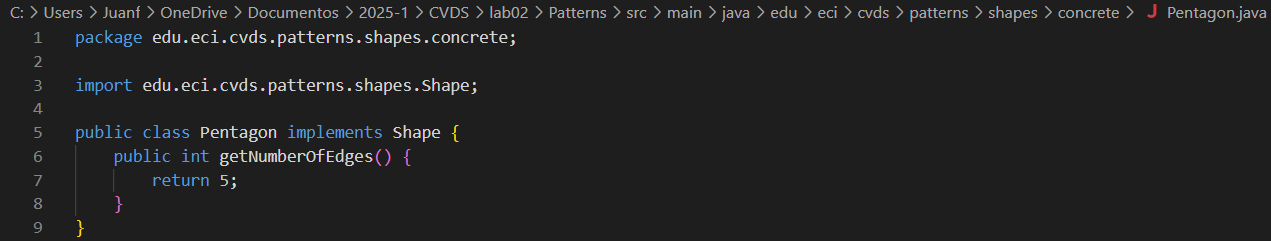
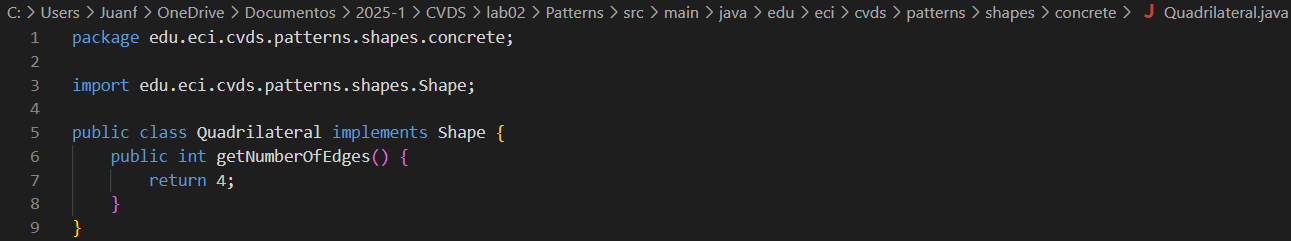
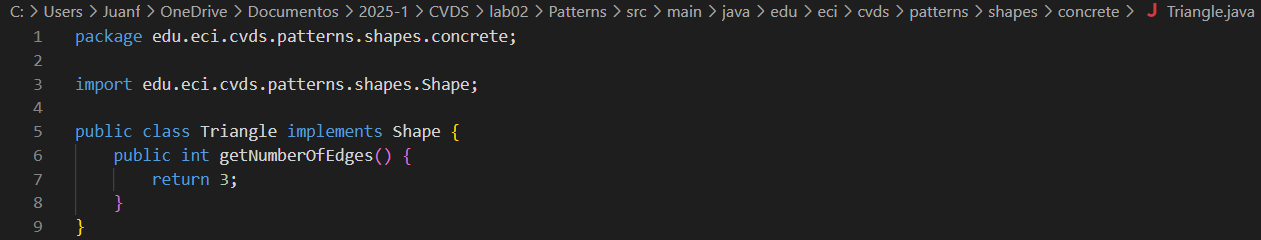
* Cree una interfaz llamada Shape.java:



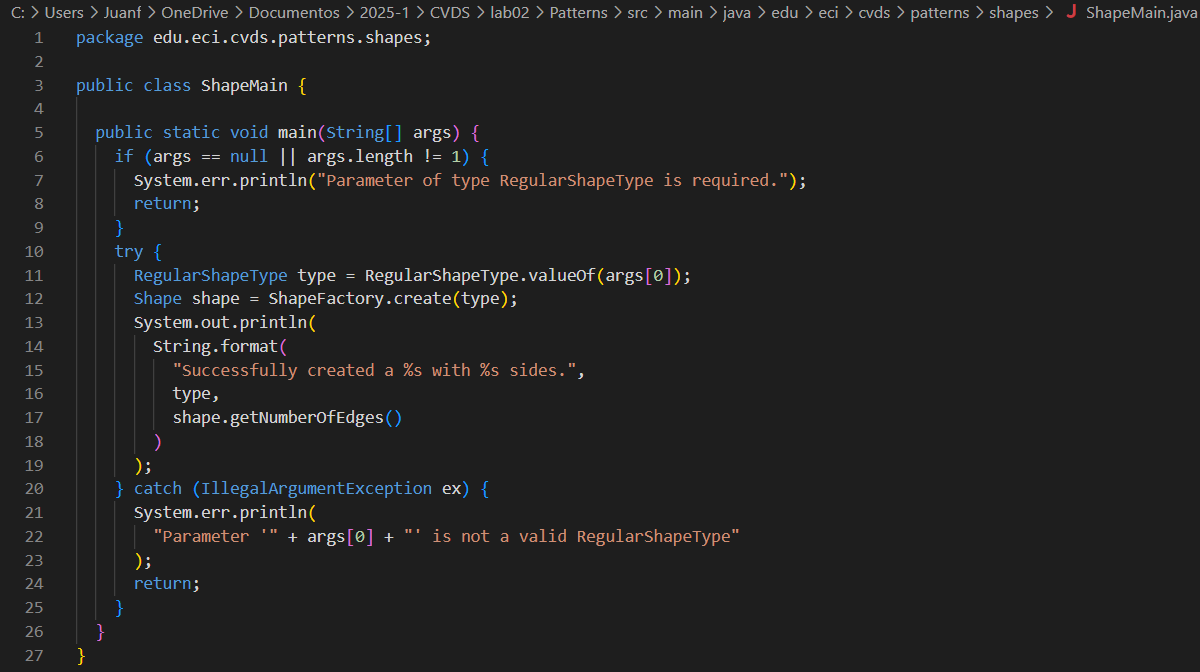
* Cree una enumeración llamada RegularShapeType.java:



* En el directorio src/main/java/edu/eci/cvds/patterns/shapes/concrete cree las diferentes clases (Triangle, Quadrilateral, Pentagon, Hexagon), que implementen la interfaz creada y retornen el número correspondiente de vértices que tiene la figura.



* Cree el archivo ShapeMain.java en el directorio src/main/java/edu/eci/cvds/patterns/shapes con el método main:



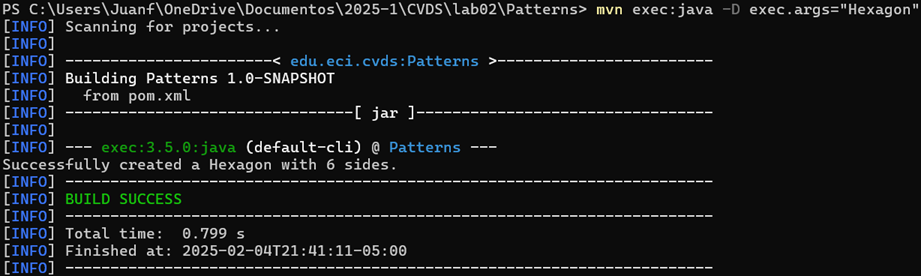
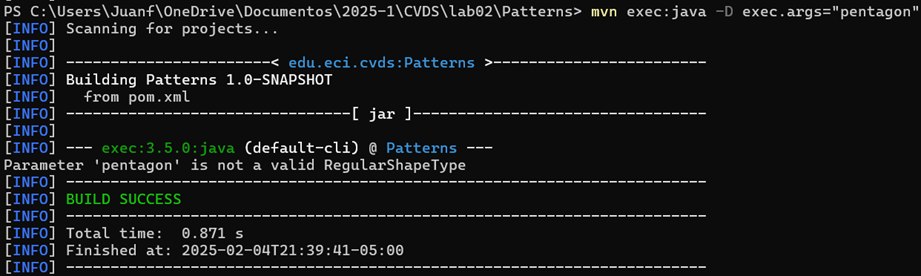
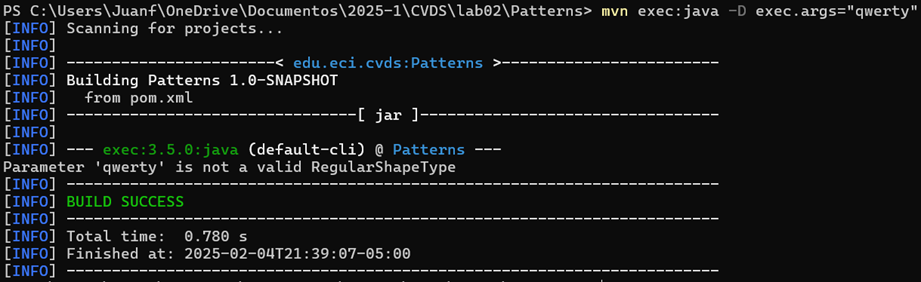
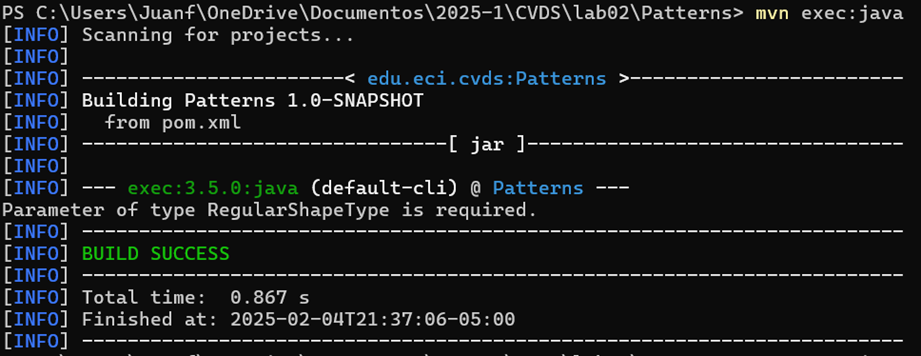
* ¿Cuál fábrica hiciste? y ¿Cuál es mejor?

Rta: La fábrica que implementamos fue la simple factory porque, al ser un proyecto sencillo, es fácil de implementar y en últimas, no es difícil hacer cambios para la extensibilidad, por ello también consideramos que es la mejor opción. Sin embargo, para proyectos más amplios, con mayor complejidad y con una cantidad mayor de clases por instanciar, la mejor opción sería abstract factory.



* ¿Cuál(es) de las instrucciones se ejecutan y funcionan correctamente y por qué?

Rta: Como se puede ver en los pantallazos, la única instrucción que funcionó correctamente fue la de “Hexagon” puesto que es la única que mantiene la sintaxis que se maneja en el switch-case de la clase de ShapeFactory.



* ¿Para qué sirve gitignore?

Rta: El archivo gitignore sirve para indicarle a gitHub que archivos deberían ser ignorados para ser subidos a algún repositorio por ser innecesarios o porque puedan llegar a generar conflicto con la línea main, así también se asegura que el repositorio este mucho mejor.