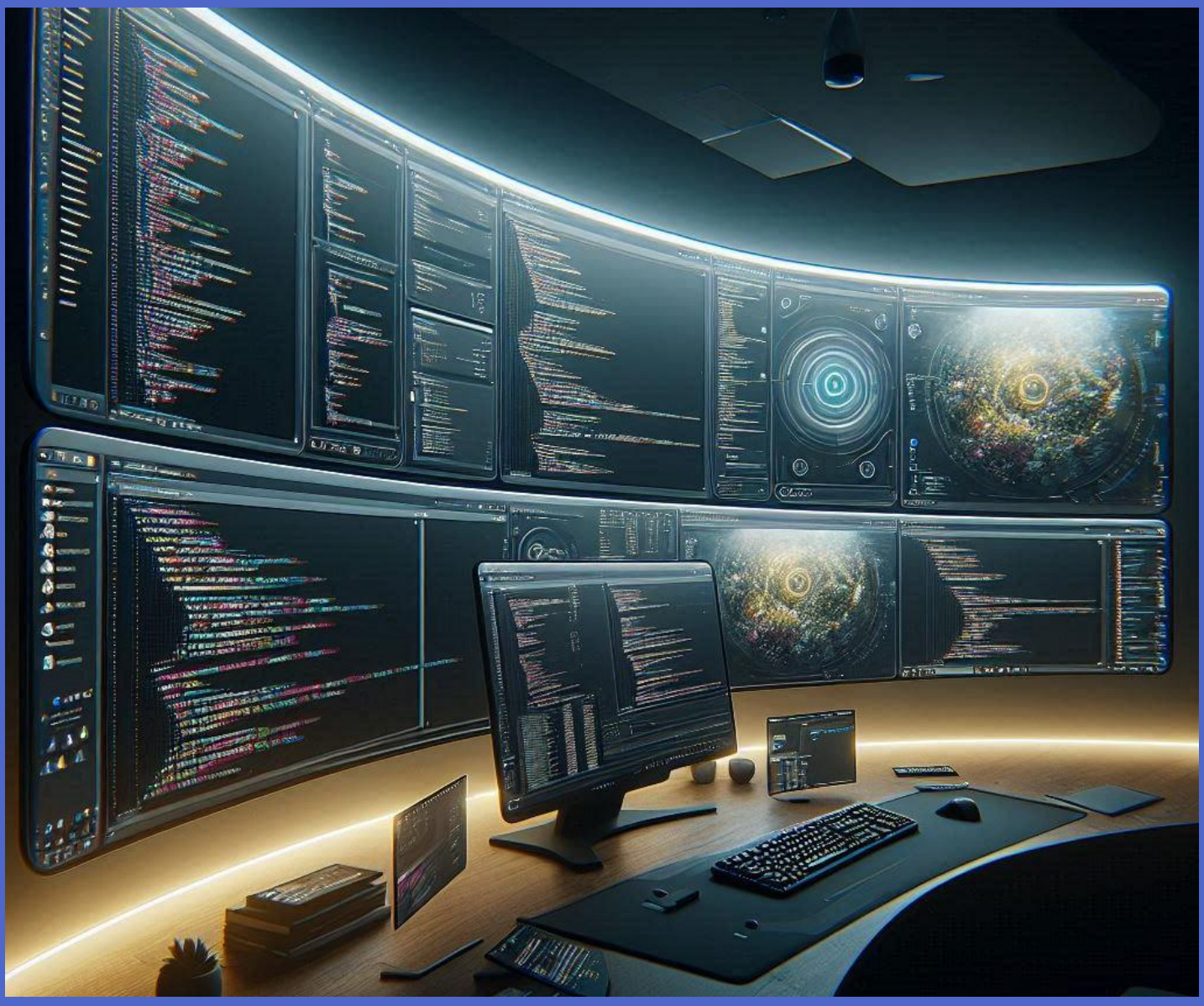
ENTORNOS DE DESARROLLO

CASO PRACTICO 1UD2



ALUMNO CESUR

24/25

Alejandro Muñoz de la Sierra

PROFESOR

Diego Tinedo Rodríguez

0 1
I N T R O D U C C I O N

En el desarrollo de software, seleccionar el entorno de desarrollo integrado (IDE) correcto es muy importante para mejorar la eficiencia, la productividad y la flexibilidad de los proyectos. Cada IDE tiene características diferentes que afectan la experiencia de desarrollo, incluyendo su facilidad de uso y la compatibilidad con varios lenguajes y herramientas.

Este caso práctico se centra en IntelliJ IDEA, que es conocida por su solidez y funciones avanzadas, comparándola con otras herramientas populares como Eclipse y Visual Studio Code. Se verá cómo instalarla, configurarla y evaluar algunas funciones clave, como la gestión de proyectos en distintos lenguajes y la creación de ejecutables.

Además, el análisis incluirá documentación del proceso, lo que ayudará a identificar las ventajas y desventajas de IntelliJ IDEA en comparación con otros IDE. Esto ofrece una visión práctica sobre cómo esta herramienta se ajusta a las demandas de proyectos reales, especialmente en entornos profesionales donde la calidad y la eficiencia son importantes.

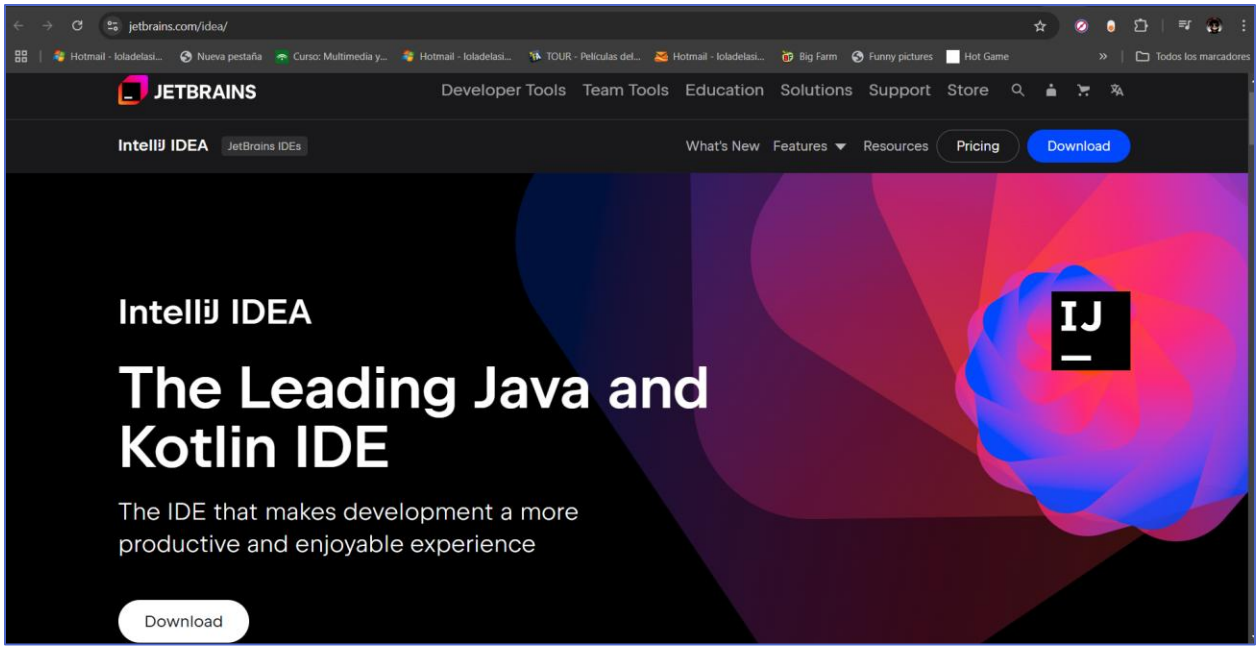
0 1

INSTALACIÓN Y CONFIGURACIÓN DE INTELLIJ IDEA

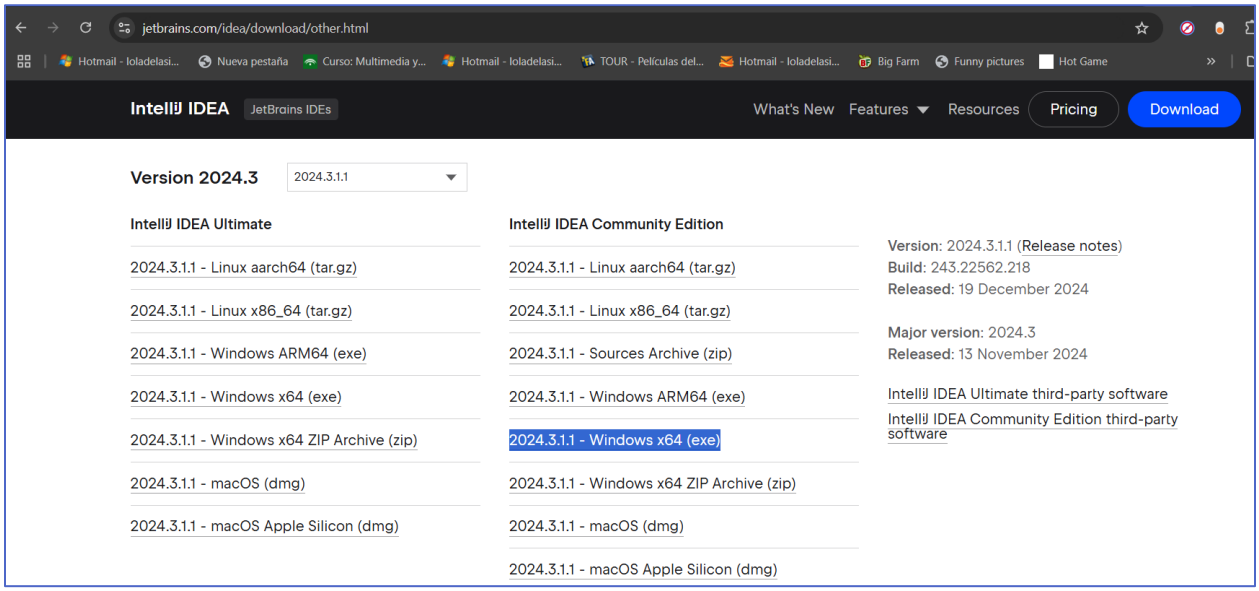
Pasos para la instalación y configuración:

Descarga del software:

- Visita la página oficial de JetBrains en: https://www.jetbrains.com/idea/.

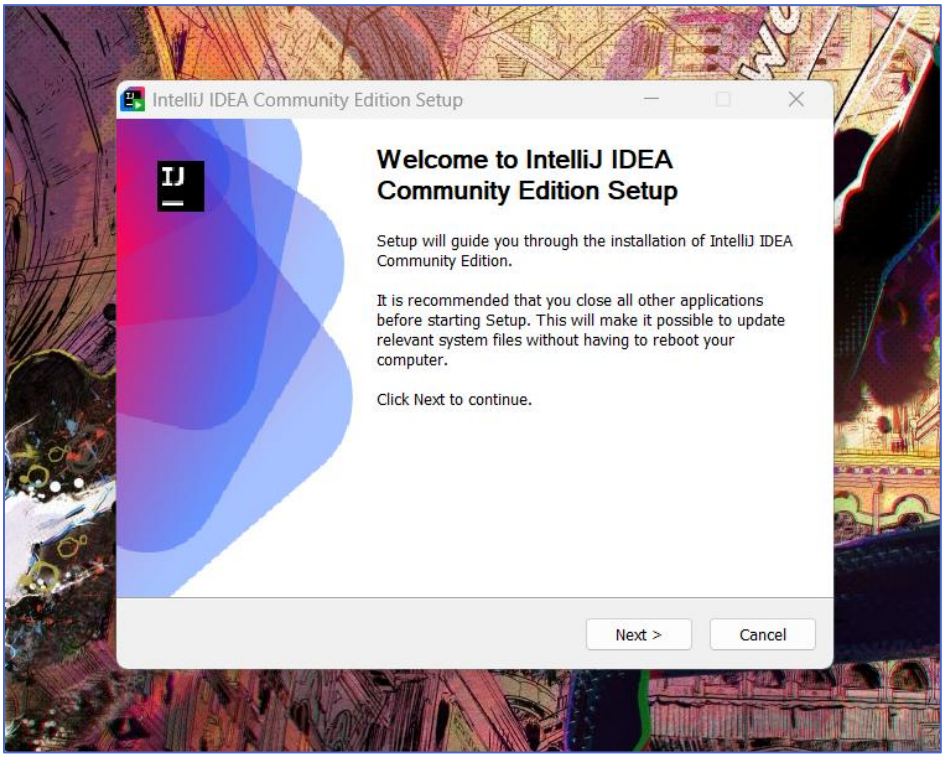


- Selecciona entre la versión **Community** (gratuita) o **Ultimate** (con prueba gratuita o licencia de pago).

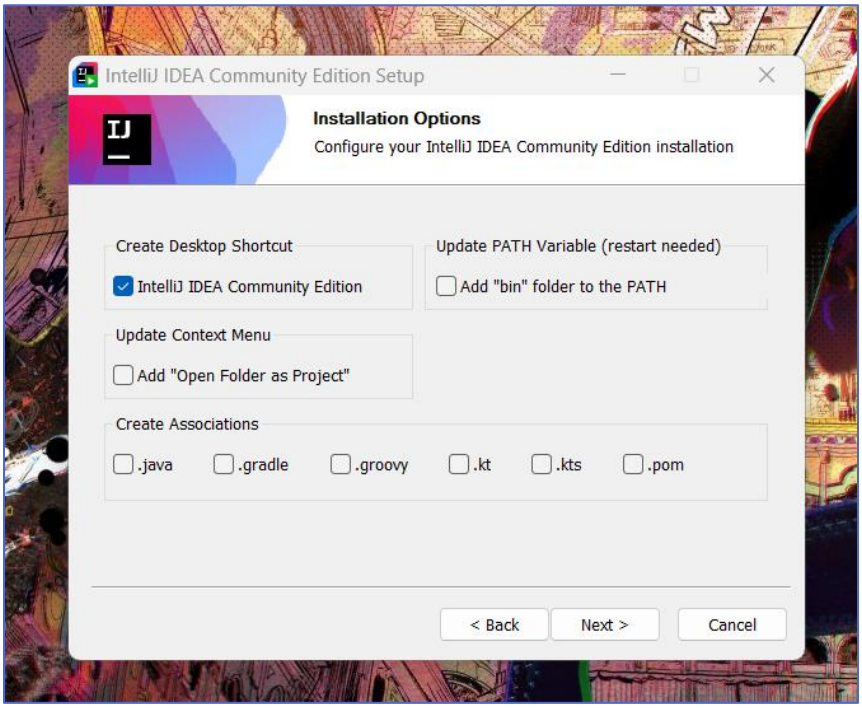


Instalación:

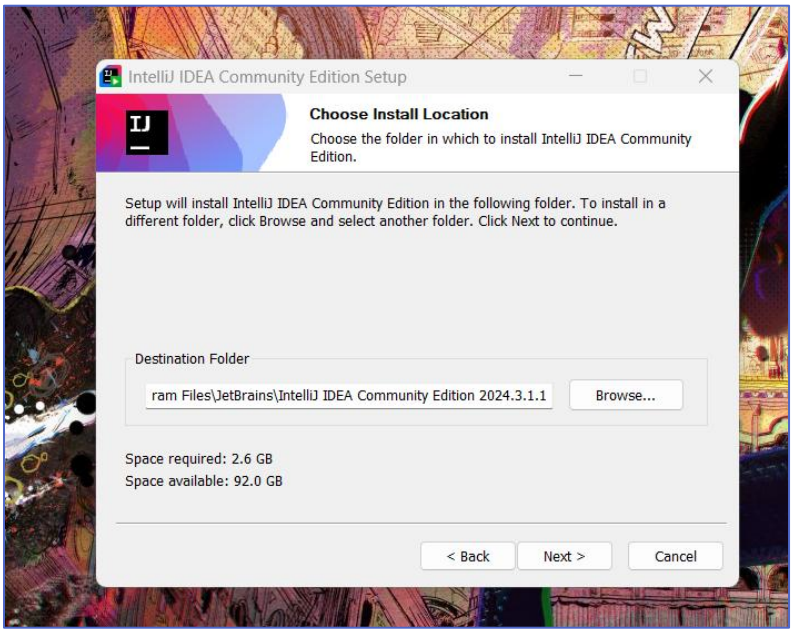
- Ejecuta el instalador y sigue estos pasos:

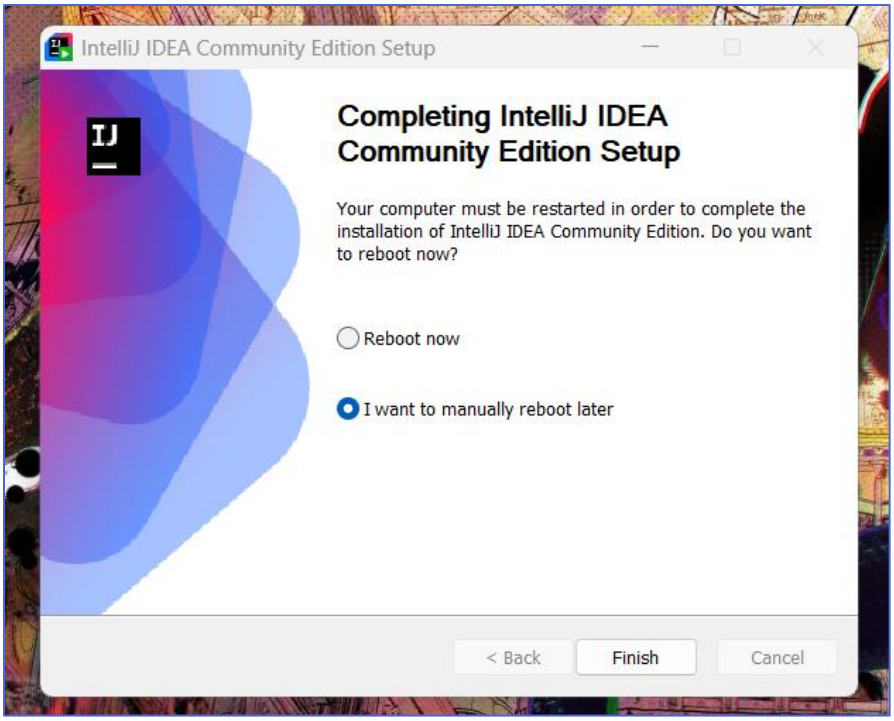


- Configura opciones personalizadas, como la asociación con archivos .java o .kt.



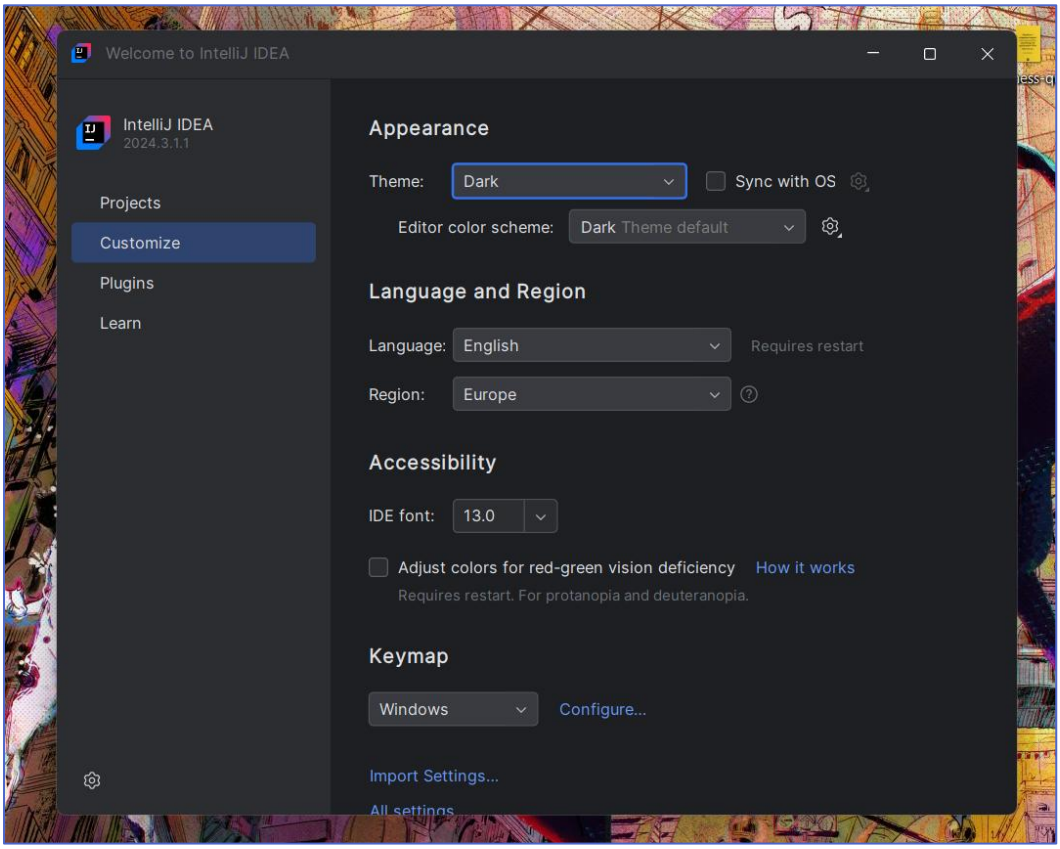
- Define el directorio de instalación y selecciona si deseas crear accesos directos en el escritorio o menú inicio.



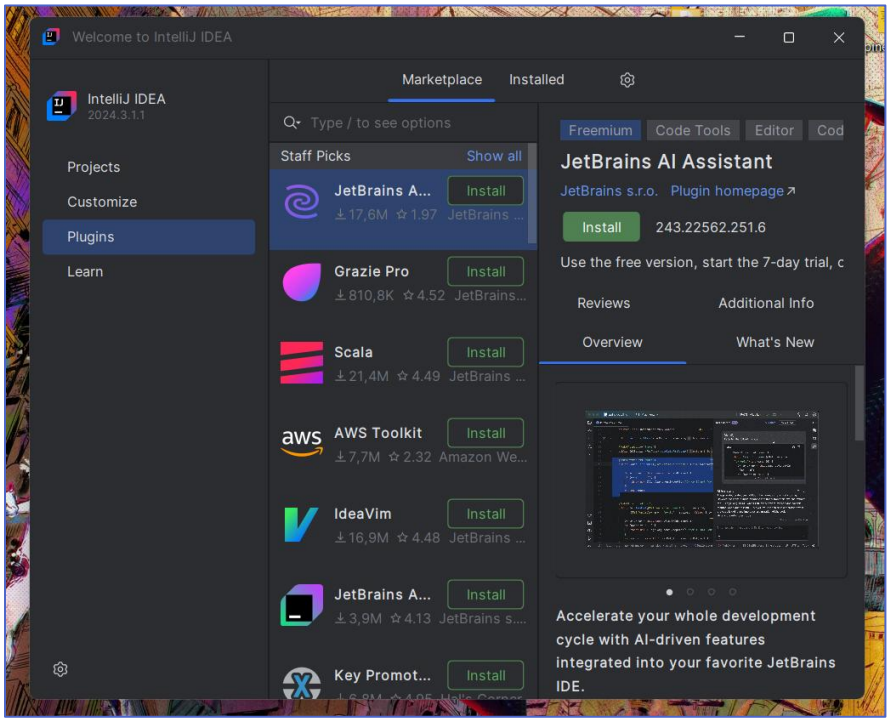


Configuración inicial:

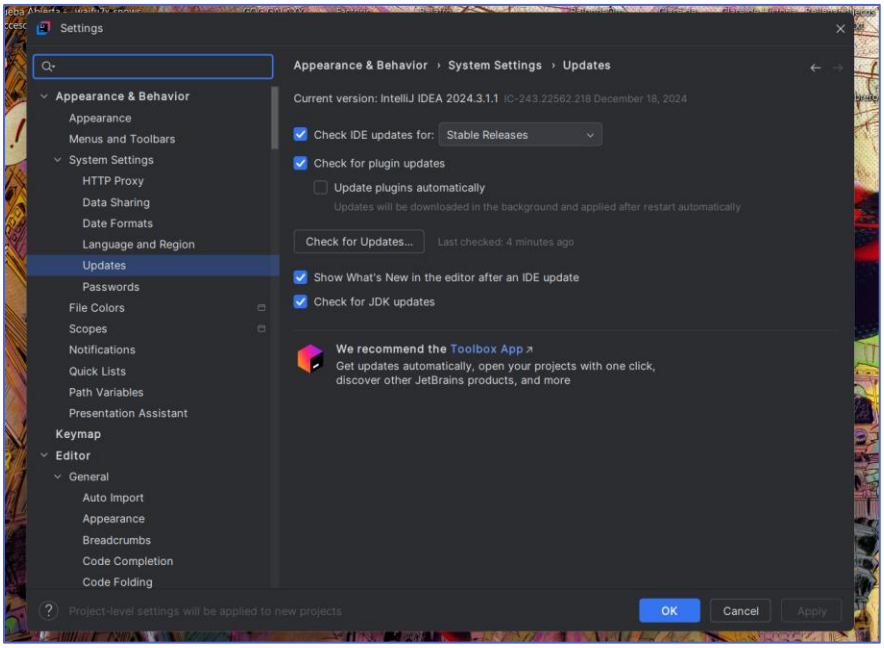
- En el primer inicio:
  - Escoge un tema visual (Dark o Light) según tu preferencia.



- Activa el soporte para lenguajes adicionales instalando plugins desde la configuración inicial o posteriormente:
  - **Kotlin** para proyectos JVM modernos.
  - **Python**, **JavaScript**, o **C++** según los requerimientos del proyecto.

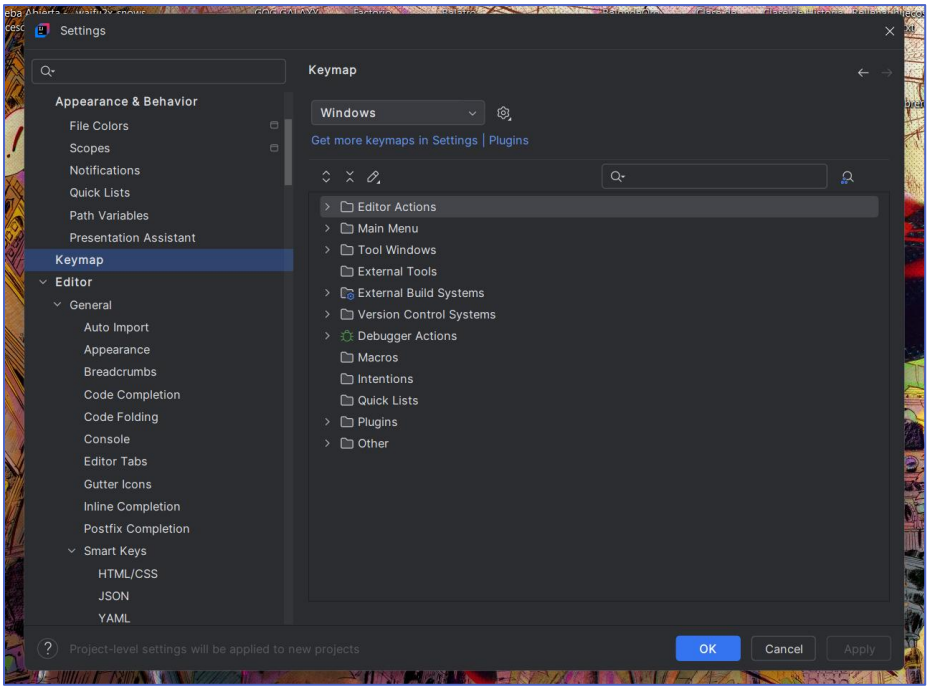


- Habilita la sincronización automática para recibir actualizaciones regulares.



Personalización del entorno:

- Configura accesos rápidos desde **Settings > Keymap**, adaptando combinaciones de teclas a tu flujo de trabajo.



0 2

GENERACIÓN DE EJECUTABLES EN INTELLIJ IDEA

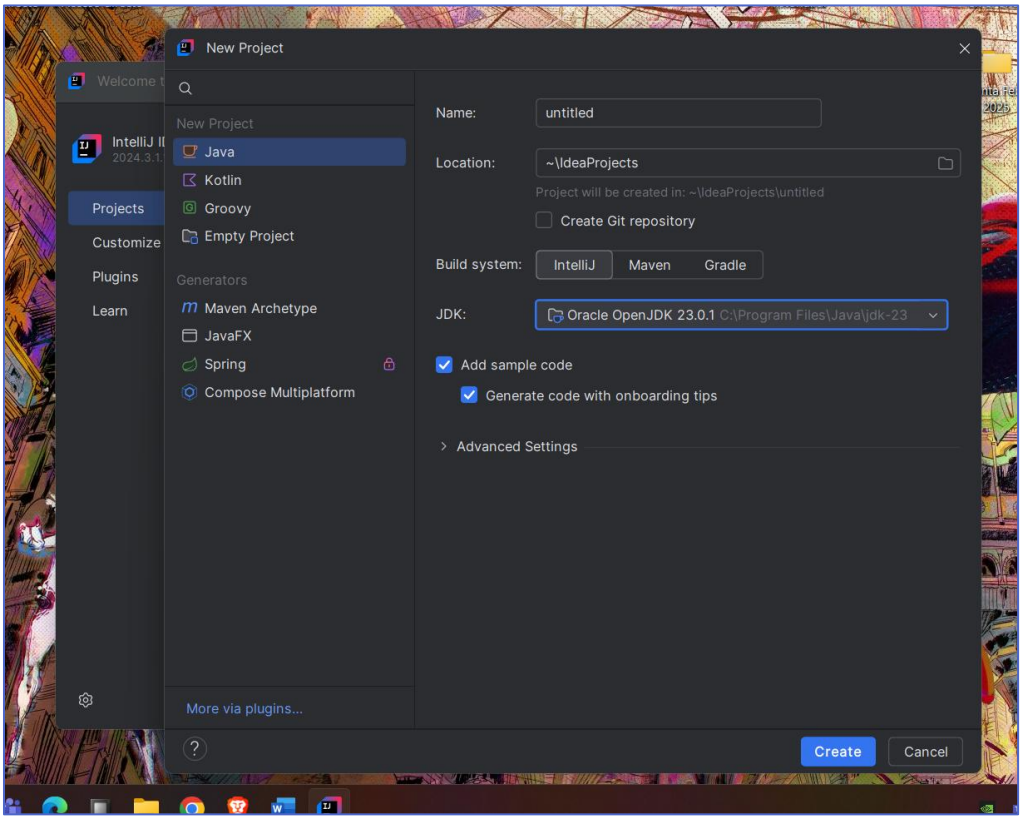
Objetivo:

Validar la flexibilidad de IntelliJ mediante la generación de ejecutables en diferentes lenguajes de programación, evaluando su integración con herramientas y flujos de trabajo comunes.

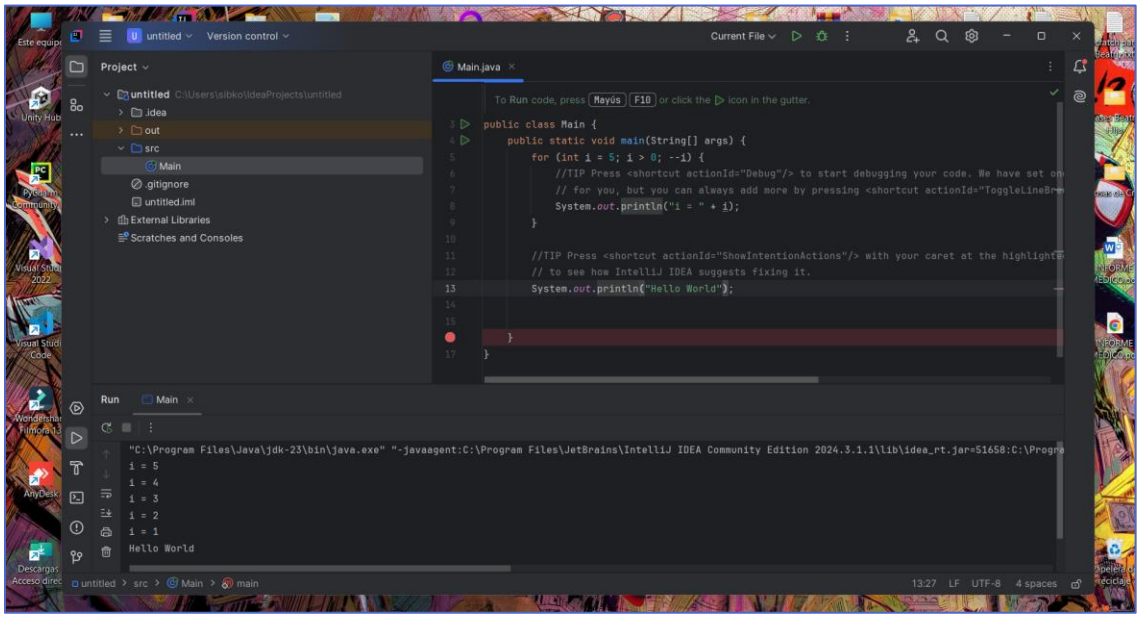
Ejemplos de pruebas realizadas:

Generación de un ejecutable en Java:

Crea un proyecto de tipo **Java Application**.



Escribe un programa simple (HelloWorld.java):



Configuración de los Artefactos

Abrir la configuración de artefactos:

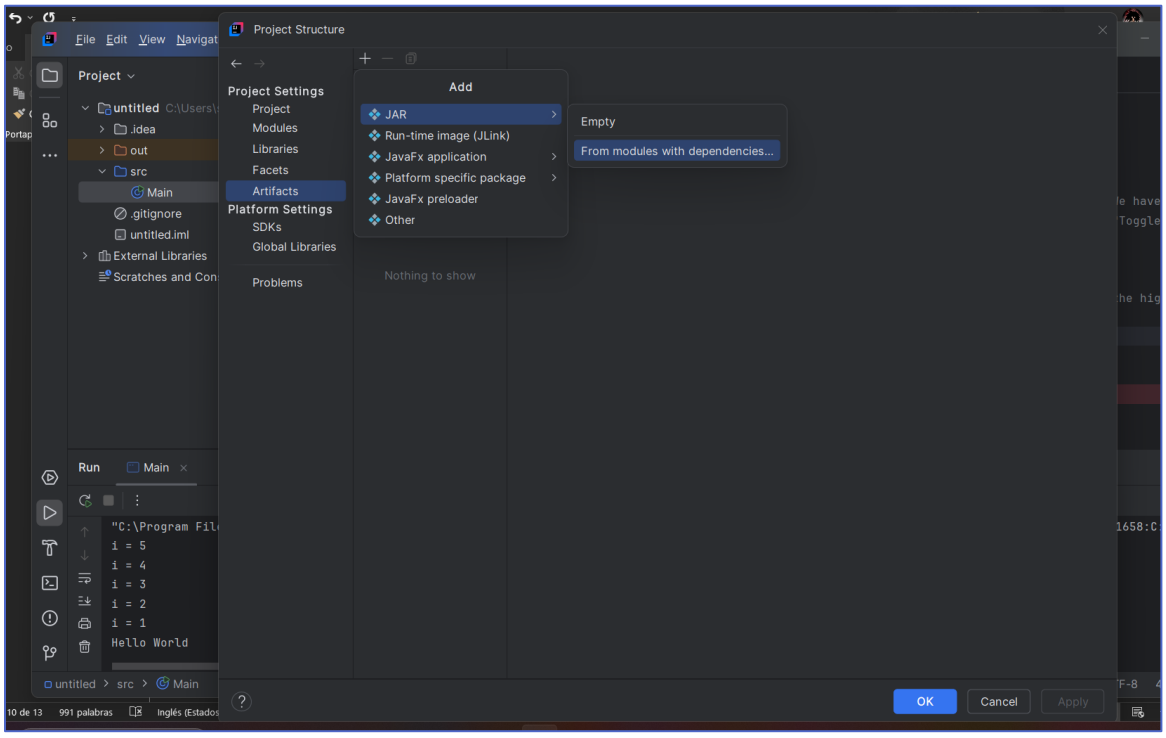
Ve al menú: File > Project Structure o presiona Ctrl + Alt + Shift + S.

En el panel de la izquierda, elige la opción Artifacts.

Crear un nuevo artefacto:

Haz clic en el botón + (en la parte superior derecha).

Selecciona JAR > From modules with dependencies.



Configurar el artefacto:

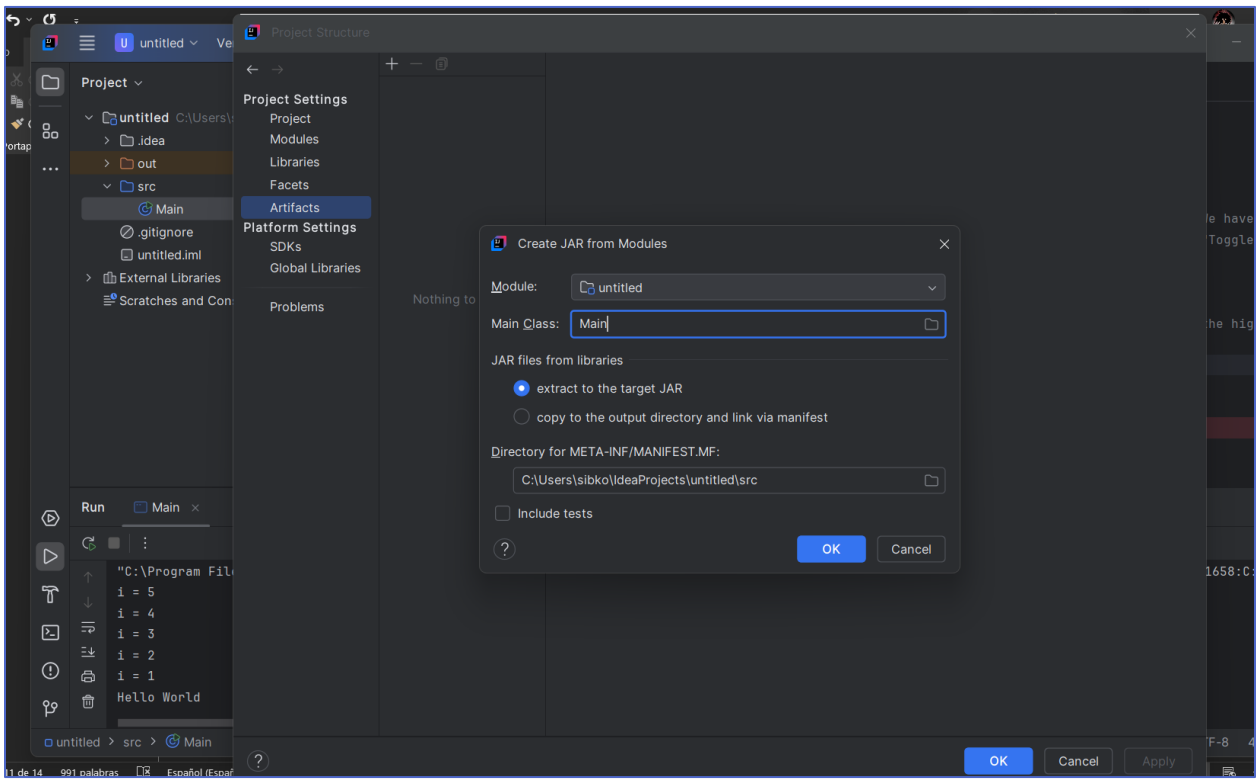
En el cuadro de diálogo:

Selecciona el módulo que tiene la clase principal.

Escoge la clase principal que tiene el método main().

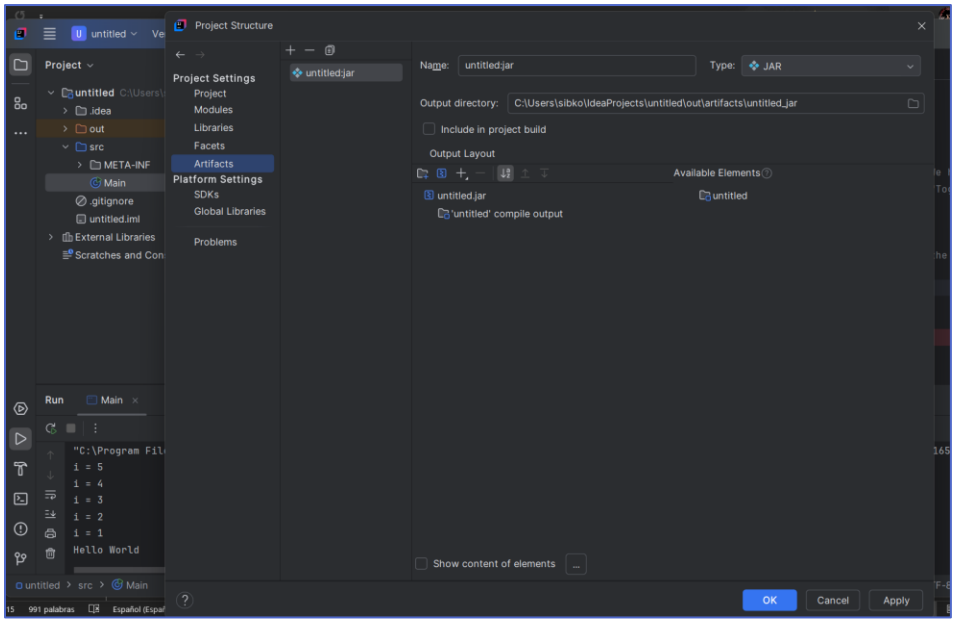
Asegúrate de que la opción Extract to the target JAR esté activada.

Haz clic en OK para guardar el artefacto.



Especificar la ubicación de salida:

En Output Directory, selecciona o escribe dónde deseas que esté el archivo .jar. Aplica los cambios y cierra la ventana de configuración.



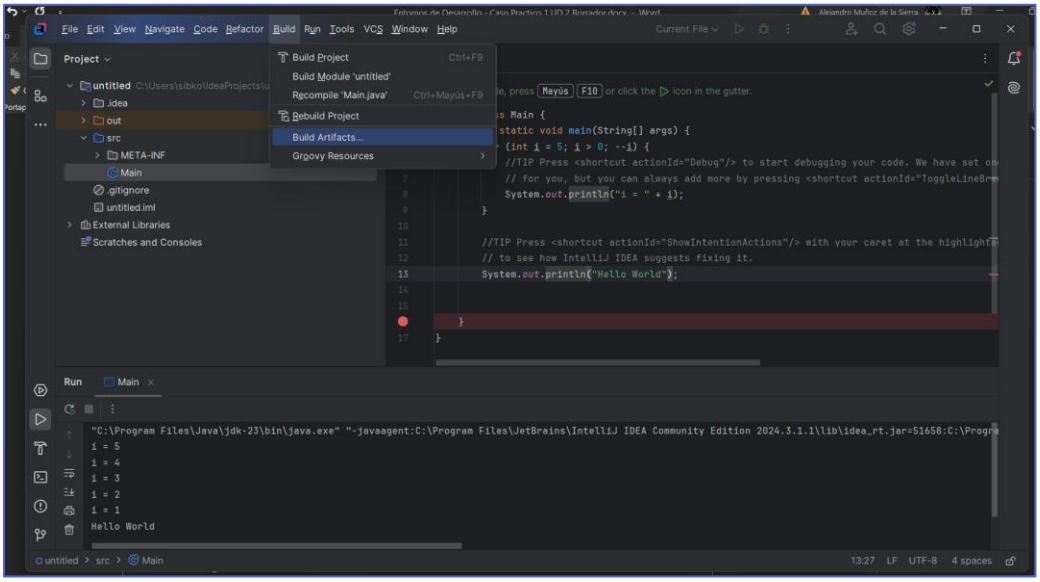
Generación del Archivo .jar

Ve al menú: Build > Build Artifacts.

Selecciona el artefacto que configuraste en la lista.

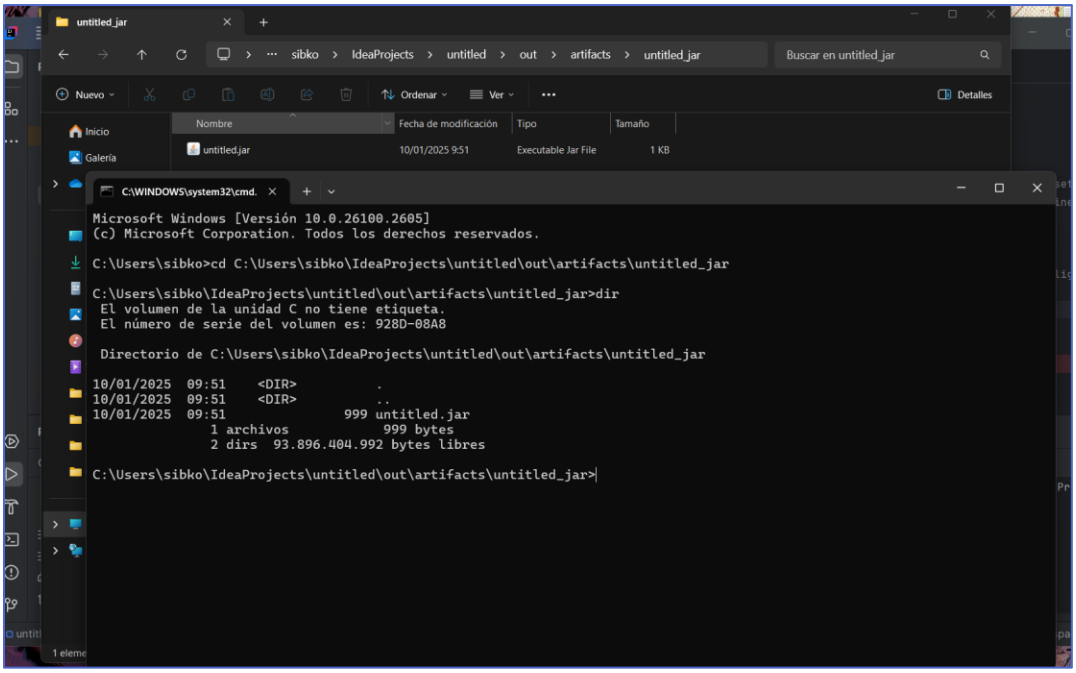
Haz clic en Build.

IntelliJ generará el archivo .jar en la carpeta que especificaste.



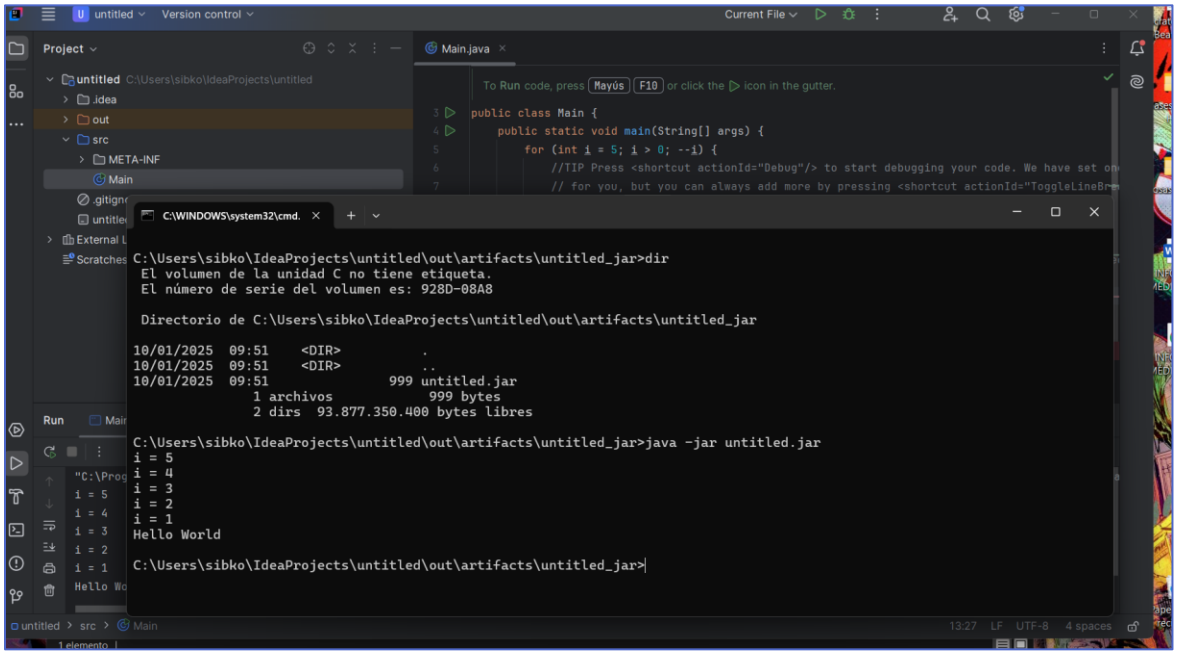
Ejecución del Archivo .jar

Abre una terminal o línea de comandos.
Navega a la carpeta donde está el archivo .jar.



Ejecuta el archivo con el siguiente comando:

java -jar tuarchivo.jar
Cambia tuarchivo.jar por el nombre de tu archivo.



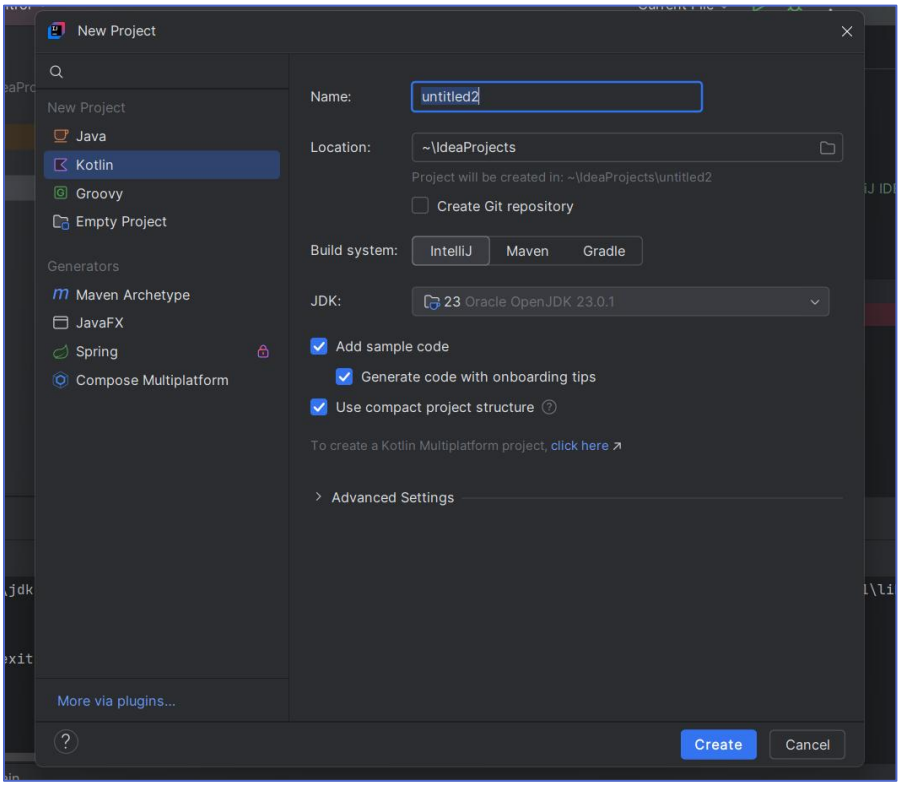
Consejos Adicionales:
Configuración del JDK:
Verifica que el JDK esté correctamente configurado. Esto se puede ver en File > Project Structure > SDKs.

Pruebas previas:
Antes de empaquetar, corre tu programa en IntelliJ para asegurarte de que todo funcione.

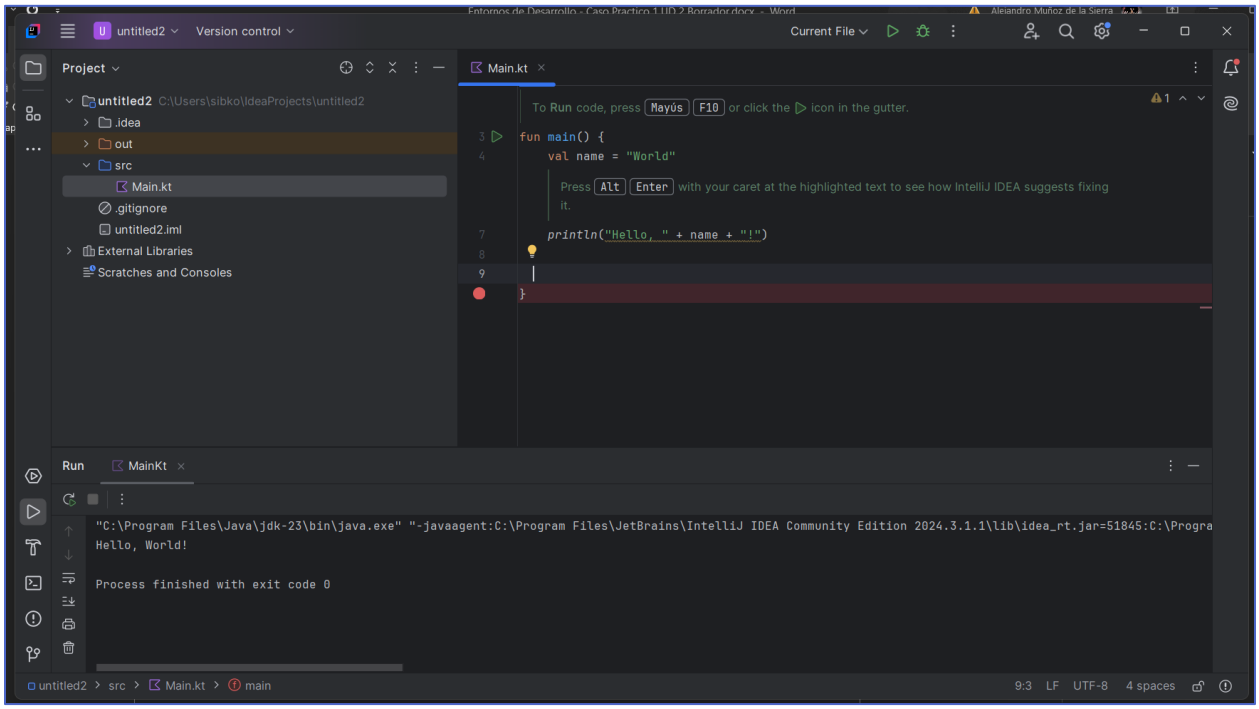
Errores comunes al ejecutar:
Si el archivo no corre, verifica que el JDK esté en la variable de entorno PATH.
Asegúrate de agregar todas las dependencias necesarias al configurar el artefacto.
Siguiendo estos pasos, habrás creado un archivo .jar que puedes compartir y ejecutar en cualquier sistema con Java instalado.

Generación de un ejecutable en Kotlin:

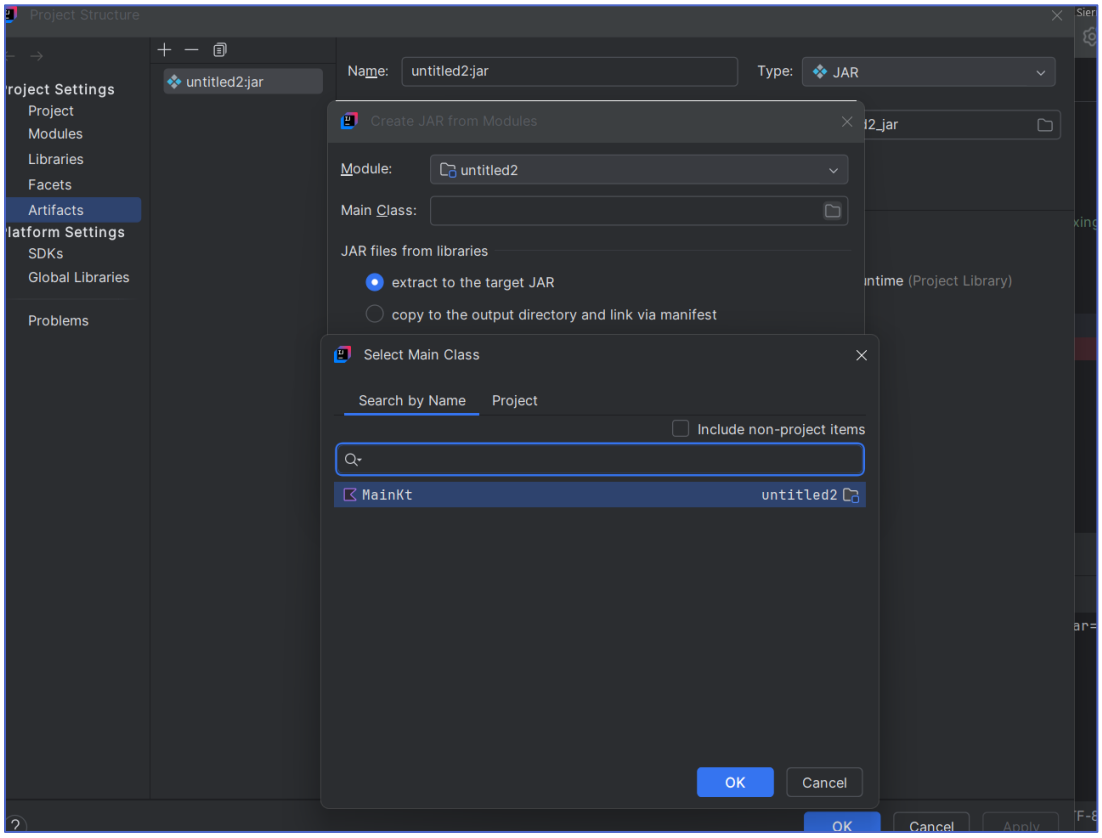
- Inicia un proyecto de tipo **Kotlin/JVM**.

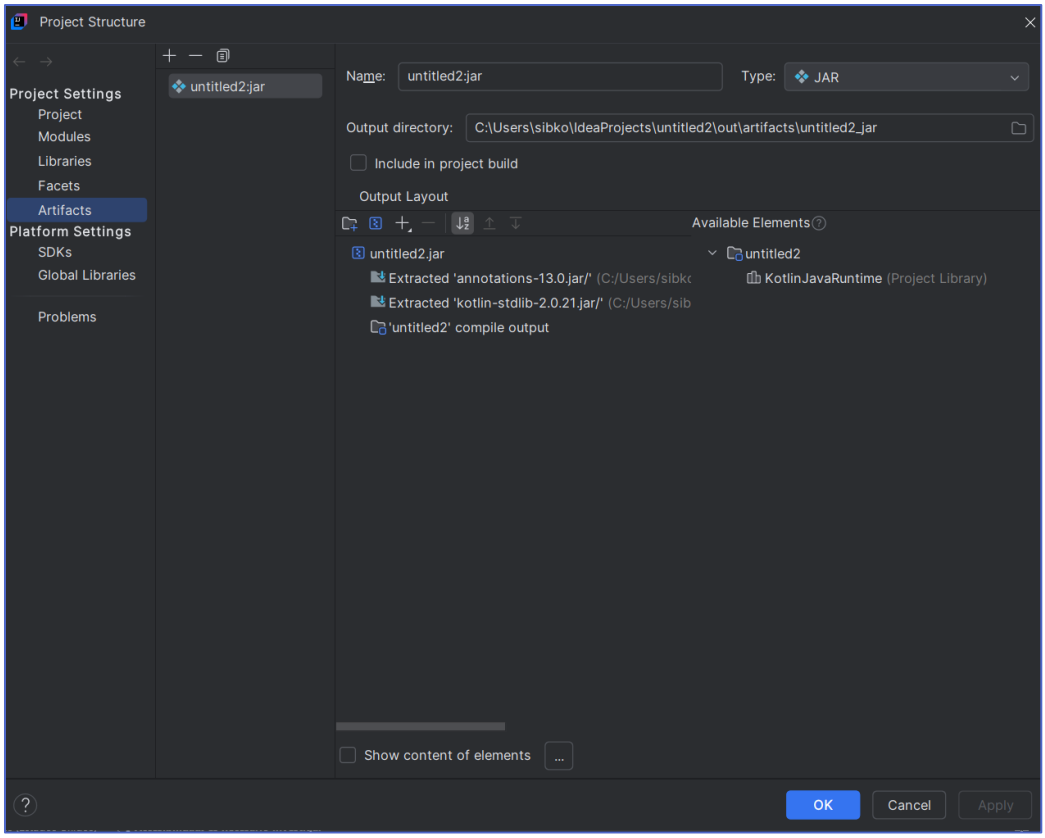


- Escribe un programa básico:

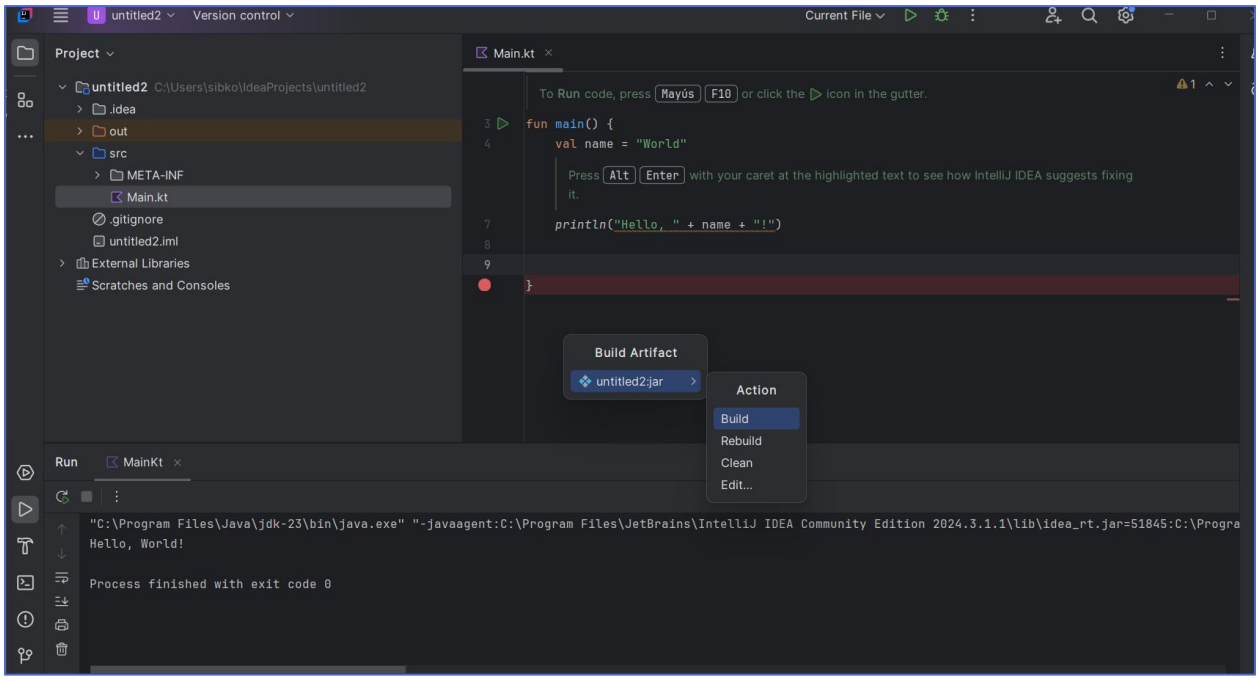


- Configuramos el proyecto con lo aprendido anteriormente

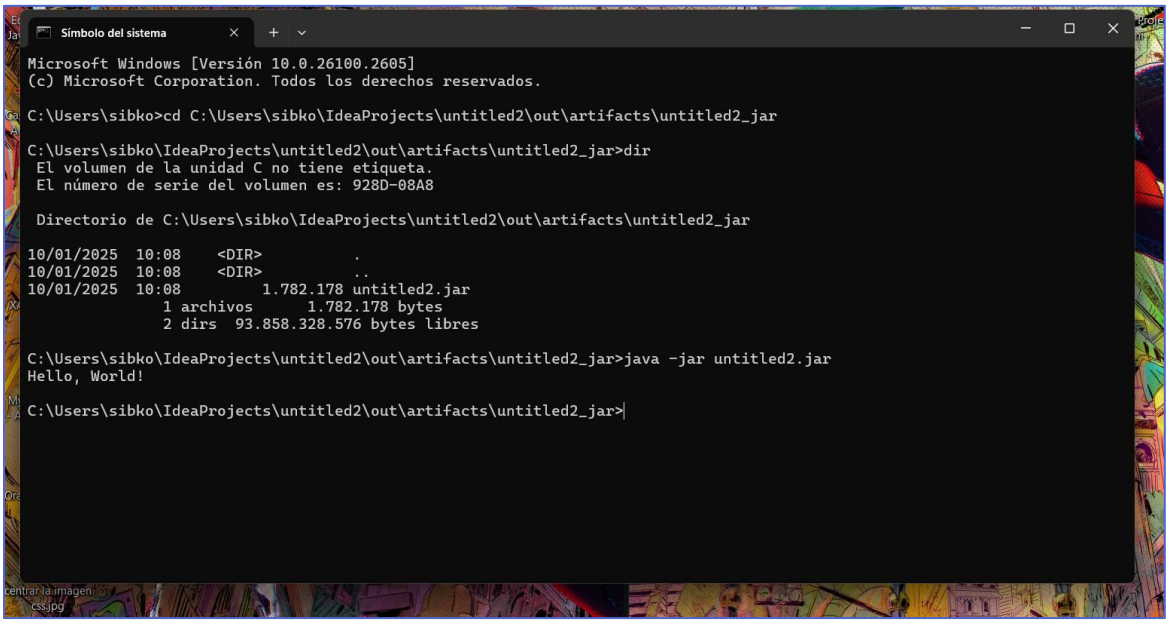


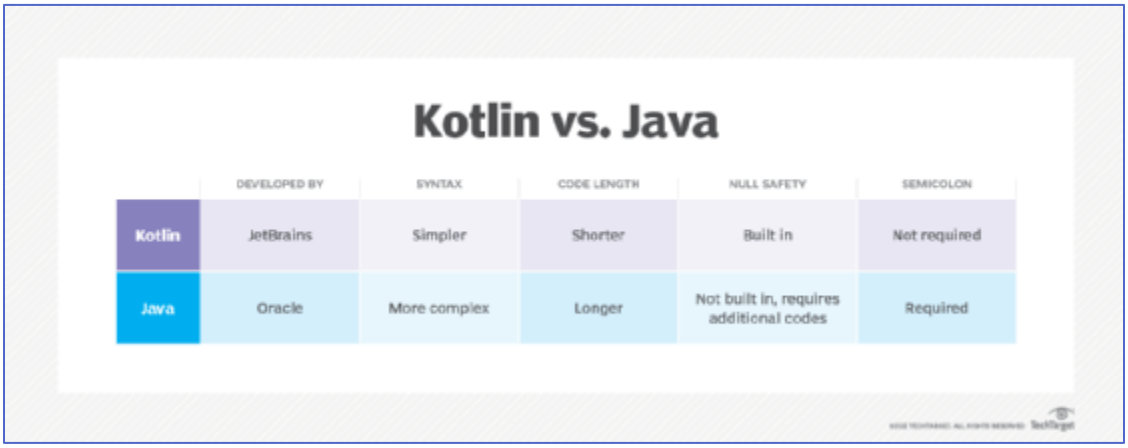


- Genera el archivo .jar desde **Build > Build Project**.



- Ejecutar el archivo .jar desde la consola de comandos





0 3

COMPARACIÓN CON OTROS IDES

Aquí se busca comparar IntelliJ con otras IDEs populares, como Eclipse y Visual Studio Code, para que se pueda ver ventajas y desventajas de cada una. Cada entorno tiene fortalezas, y la elección del mejor depende de lo que se necesite en un editor de código.

Ventajas de IntelliJ

Integración completa: IntelliJ proporciona una integración fluida con Kotlin, Java y otros lenguajes. Esto permite un desarrollo más eficiente sin requerir tantas herramientas externas.

Refactorización y análisis de código: Las herramientas de refactorización en IntelliJ son avanzadas, lo que facilita la mejora de la calidad del código.

Soporte para frameworks: IntelliJ tiene soporte para muchos frameworks populares, lo que puede ahorrar tiempo en proyectos grandes.

Interfaz intuitiva: Su interfaz de usuario es moderna y fácil de manejar, facilitando la navegación entre proyectos y archivos.

Inconvenientes de IntelliJ

Consumo de recursos: Puede ser más pesado que editores como Visual Studio Code, lo que puede ser problemático en máquinas con pocos recursos.

Curva de aprendizaje: Aunque su interfaz es intuitiva, algunas funciones avanzadas pueden ser difíciles de entender al principio.

Ventajas de Eclipse

Ligereza: Eclipse es más ligero comparado con IntelliJ, lo que lo convierte en una buena opción para equipos pequeños o proyectos menos complejos.
Personalización: Eclipse permite una alta personalización, adaptando el entorno a necesidades específicas.
Ecosistema amplio: Tiene un gran número de plugins y herramientas adicionales, permitiendo extender su funcionalidad según lo necesario.

Inconvenientes de Eclipse

Curva de aprendizaje: A pesar de ser ligero, la interfaz de Eclipse puede ser desordenada y algo difícil de usar al principio.
Integración limitada con ciertos lenguajes: Aunque soporta Java y otros lenguajes, no tiene la misma fluidez en la integración con Kotlin y lenguajes modernos como IntelliJ.

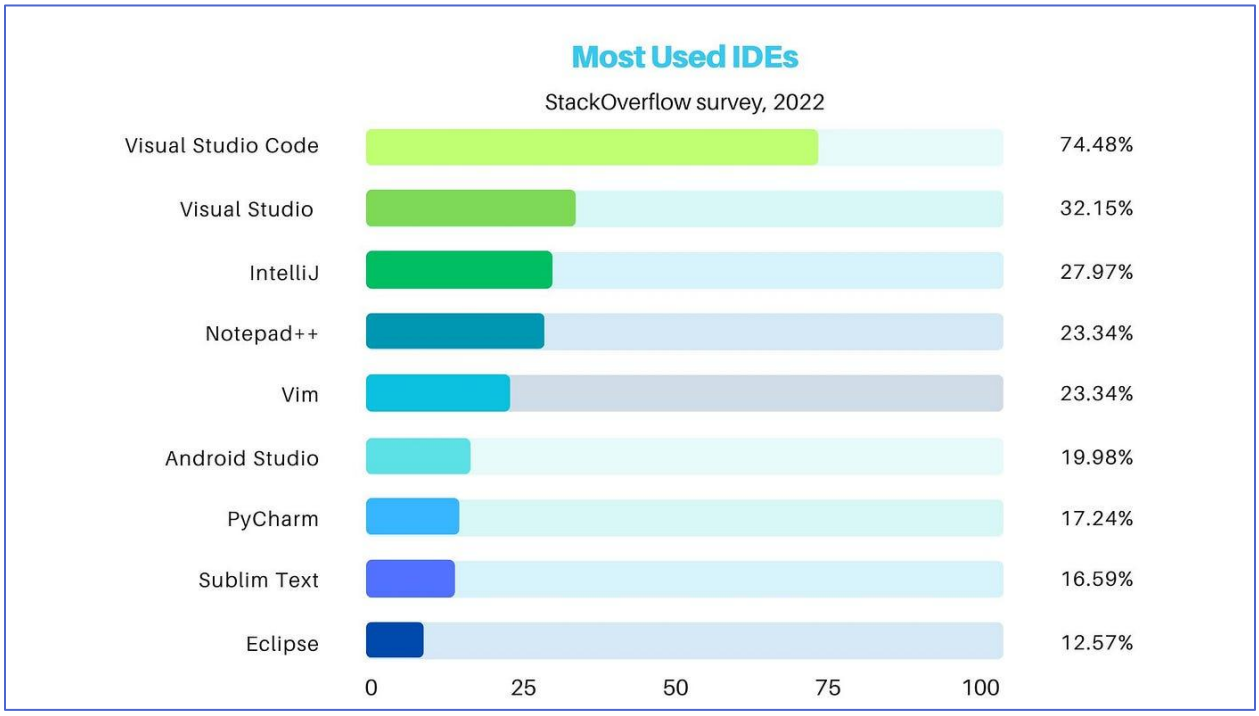
Ventajas de Visual Studio Code

Ligero y rápido: Visual Studio Code es rápido y consume pocos recursos, una gran opción para proyectos pequeños o en máquinas menos potentes.
Extensiones: Cuenta con muchas extensiones, permitiendo añadir soporte para diversos lenguajes y frameworks según las necesidades.
Interfaz sencilla: Su interfaz es limpia y minimalista, ideal para quienes prefieren una experiencia directa y sin distracciones.

Inconvenientes de Visual Studio Code

Funcionalidad limitada sin extensiones: Aunque es muy ligero, algunas funciones de IntelliJ o Eclipse requieren instalación de extensiones, añadiendo complejidad.
Menos herramientas integradas: A diferencia de IntelliJ, Visual Studio Code tiene menos herramientas nativas, así que algunas tareas pueden necesitar configuración adicional.

Aspecto	IntelliJ IDEA	Eclipse	Visual Studio Code
Interfaz de usuario	Intuitiva y moderna, ideal para productividad.	Más compleja y menos visual.	Minimalista y ligera.
Rendimiento	Alto, pero puede ser pesado para equipos con pocos recursos.	Más ligero, aunque con limitaciones.	Muy ligero y rápido.
Compatibilidad	Excelente soporte para Java, Kotlin, y frameworks como Spring.	Especializado en Java, extensiones limitadas.	Multilenguaje con gran cantidad de plugins.
Extensiones	Plugins integrados y fácil de gestionar.	Extensiones más limitadas.	Gran cantidad de extensiones disponibles.
Soporte	Documentación oficial y comunidad activa.	Menos soporte para tecnologías modernas.	Comunidad masiva, especialmente para web.



0 4

C O N C L U S I O N E S

Luego de terminar la instalación, configuración y comparación de IntelliJ IDEA, se ha visto que este IDE brilla en varios puntos importantes:

Interfaz fácil de usar: Su diseño ayuda a navegar y manejar proyectos, especialmente en desarrollos grandes.
Compatibilidad con diversos lenguajes: IntelliJ es muy efectivo en proyectos de Java y Kotlin, por su integración con herramientas como Gradle y JUnit.
Funciones avanzadas: Características como autocompletado, detección de errores en tiempo real y soporte para repositorios Git, aumentan la productividad.
Al compararlo con otros IDEs:

Eclipse es bueno por ser gratis y personalizable, pero su interfaz puede ser complicada para nuevos desarrolladores.
Visual Studio Code es bueno para proyectos pequeños y tiene buen soporte para muchos lenguajes, pero le faltan algunas características avanzadas que IntelliJ tiene de forma nativa.
Aunque el rendimiento de IntelliJ puede ser pesado en computadoras con pocos recursos, sus beneficios, como flexibilidad y soporte para herramientas modernas, lo hacen una opción adecuada para desarrollos complejos.

Este caso práctico subraya no solo la relevancia de elegir el IDE correcto según el contexto, sino también la importancia de documentar cada paso del proceso. Esto garantiza un desarrollo efectivo, organizado y en línea con los objetivos del proyecto en ambientes profesionales demandantes.

0 6

R E F E R E N C I A S

https://support.academicsoftware.eu/hc/es/articles/360006978997-C%C3%B3mo-instalar-IntelliJ-IDEA-Community-Edition

https://www.youtube.com/watch?v=VDgETn22rpo

https://docs.spongepowered.org/5.1.0/es/plugin/workspace/idea.html

https://platzi.com/tutoriales/1222-java-basico-2018/4706-como-crear-un-archivo-jar-en-intellij/

https://www.studocu.com/pe/document/universidad-tecnologica-del-peru/taller-de-programacion/creacion-de-archivos-jar/57513940

https://www.youtube.com/watch?v=d1CT7_WZGB8

https://diegosf.es/tecnologia/programacion/eligiendo-el-mejor-ide-entornos-desarrollo-integrado/

https://www.youtube.com/watch?v=3TXRJGEAL30

https://www.youtube.com/watch?v=wvWolacbrTU