



Programación Móvil



www.sena.edu.co

Programación Móvil

El desarrollo móvil es la actividad encaminada a la creación de aplicaciones o programas para dispositivos como los Smartphones y Tablets.

Esta actividad es llevada a cabo por programadores y diseñadores, quienes valiéndose de las herramientas como lenguajes de programación, APIs y SDKs, realizan aplicaciones para una plataforma móvil o para múltiples de ellas



Programación Móvil

A lo largo de los años han existidos sistemas operativos para plataformas móviles tales como:

- Android de Google
- Windows Phone de Microsoft,
- iOS de Apple,
- Blackberry OS de RIM, entre otros

Cabe resaltar que hay muchas de estas que han sido descontinuadas, actualmente solo hay dos plataformas que dominan totalmente el mercado: Android y iOS.

Para el desarrollo de aplicaciones específica se puede usar las plataformas de desarrollo nativo (Android Studio para Android y Xcode para IOS) o se puede hacer de tecnologías multiplataforma.

Programación Móvil

El desarrollo de aplicaciones móviles nativas implica escribir código específico de la plataforma para cada sistema operativo, utilizando los lenguajes y las herramientas de desarrollo proporcionados por las respectivas plataformas. Por ejemplo, las aplicaciones de Android suelen estar escritas en **Kotlin** o Java, mientras que las aplicaciones de iOS están programadas en Swift u Objective-C.

Por el contrario, el desarrollo multiplataforma implica escribir una única base de código que se puede compilar e implementar en múltiples plataformas. Los marcos multiplataforma populares incluyen Xamarin, React Native y Flutter, que utilizan lenguajes como C#, JavaScript y Dart, respectivamente.

Programación nativa VS multiplataforma

	Ventajas	Desventajas
Nativo	<ul style="list-style-type: none"> • Amplia Funcionalidad • Escalabilidad mejorada • Gran rendimiento y UX 	<ul style="list-style-type: none"> • Costosa • Desarrollo lento
Multiplataforma	<ul style="list-style-type: none"> • Menor costo • Desarrollo Más Rápido • Un solo código base 	<ul style="list-style-type: none"> • Menor rendimiento • Funcionalidad limitada • UX Limitada

Si bien el desarrollo multiplataforma puede proporcionar ahorros de costos y tiempos de desarrollo más rápidos, el desarrollo nativo sigue siendo la opción preferida para muchos desarrolladores debido a sus beneficios de rendimiento y al cumplimiento de las pautas de diseño de cada plataforma.

Android

Android es un sistema operativo desarrollado por Google para dispositivos móviles, como teléfonos inteligentes y tabletas.

Es el software que permite que los dispositivos funcionen, proporcionando una plataforma para que las aplicaciones se ejecuten y ofrezcan diversas funcionalidades, como navegación web, juegos, comunicación y más.

Android es conocido por su flexibilidad y personalización, permitiendo a los usuarios y desarrolladores adaptar la experiencia del dispositivo a sus necesidades y preferencias.



Android Desarrollo de aplicaciones

Para desarrollar aplicaciones en Android, se utilizan dos lenguajes principalmente

Kotlin: Es el lenguaje moderno recomendado por Google para el desarrollo de aplicaciones Android. Ofrece características avanzadas, seguridad nula en punteros, interoperabilidad perfecta con Java y es más conciso que Java, lo que puede mejorar la productividad del desarrollador.

Java: Es el lenguaje tradicional y más utilizado para desarrollar aplicaciones Android. Java es conocido por su robustez y es ampliamente soportado por la plataforma Android. Tiene una amplia documentación y comunidad de desarrolladores.

Ambos lenguajes son compatibles con Android Studio, el entorno de desarrollo integrado (IDE) oficial de Android.



Kotlin

Kotlin es un lenguaje de programación moderno y fácil de usar, creado por JetBrains, que se utiliza principalmente para desarrollar aplicaciones Android. Su sintaxis clara y concisa permite a los desarrolladores escribir y mantener código de manera más sencilla y eficiente, lo que mejora la productividad y la seguridad del desarrollo de aplicaciones.

En 2017, Google adoptó oficialmente a Kotlin como un lenguaje soportado para el desarrollo de aplicaciones Android, lo que marcó un hito importante en su evolución. Esta adopción significó que Kotlin se convertiría en una herramienta fundamental para los desarrolladores de Android, ya que cuenta con el respaldo de Google y está integrado en el ecosistema de desarrollo de Android, incluyendo Android Studio, el entorno de desarrollo integrado (IDE) principal para Android.


Instalación de Kotlin

Para poder trabajar con Kotlin debemos instalar:

- Primero debemos descargar e instalar el JDK (Java Development Kit).
- El entorno de desarrollo más extendido para el desarrollo en Kotlin es el IntelliJ IDEA. Podemos descargar la versión Community que es gratuita.

Instalación de Kotlin (Verificar si está instalado el JDK)


- En la línea de comandos, escribe:

 Copy code

```
java -version
```


- Si el JDK está instalado, deberías ver una salida similar a esta:

SCSS

 Copy code

```
java version "1.8.0_221"  
Java(TM) SE Runtime Environment (build 1.8.0_221-b11)  
Java HotSpot(TM) 64-Bit Server VM (build 25.221-b11, mixed mode)
```

- También puedes verificar la versión del compilador `javac` con:

 Copy code

```
javac -version
```

- Si el JDK no está instalado, verás un mensaje indicando que el comando no se reconoce.



Instalación de Kotlin (Instalar el JDK)

Ir a la página: <https://adoptium.net/es/>. Descargar y guardar la ultima versión del jdk

ADOPTIUM Home Marketplace Documentation FAQ Projects Further Information

OpenJDK Preconstruido
¡Binarios Gratis!

Java™ es el lenguaje de programación y la plataforma líderes en el mundo. El Grupo de Trabajo de Adoptium promueve y admite tiempos de ejecución de alta calidad, certificados por TCK y tecnología asociada para su uso en todo el ecosistema Java. Eclipse Temurin es el nombre de la distribución OpenJDK de Adoptium.

Descargar Temurin™ para Windows x64

↓ Última versión de LTS
jdk-17.0.7+7

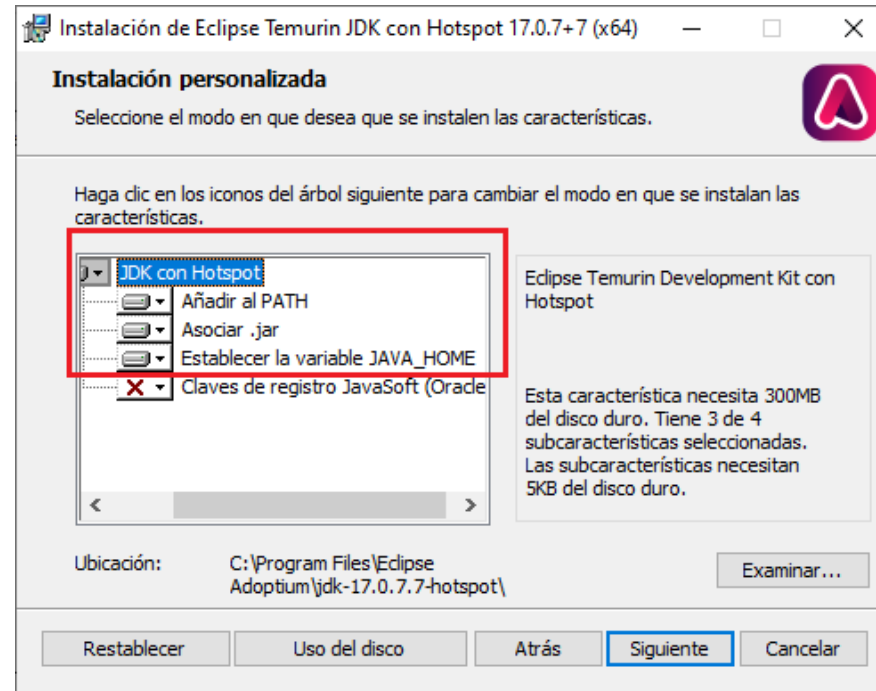
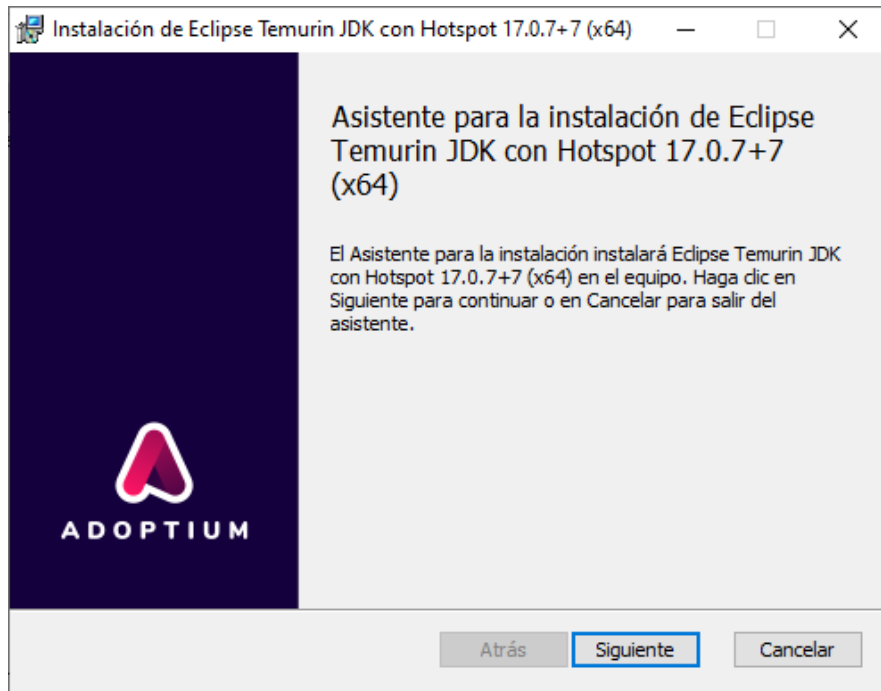
Otras plataformas y versiones ↗

Archivo de versión 📄

Cambiar idioma ▼

Instalación JDK de Java

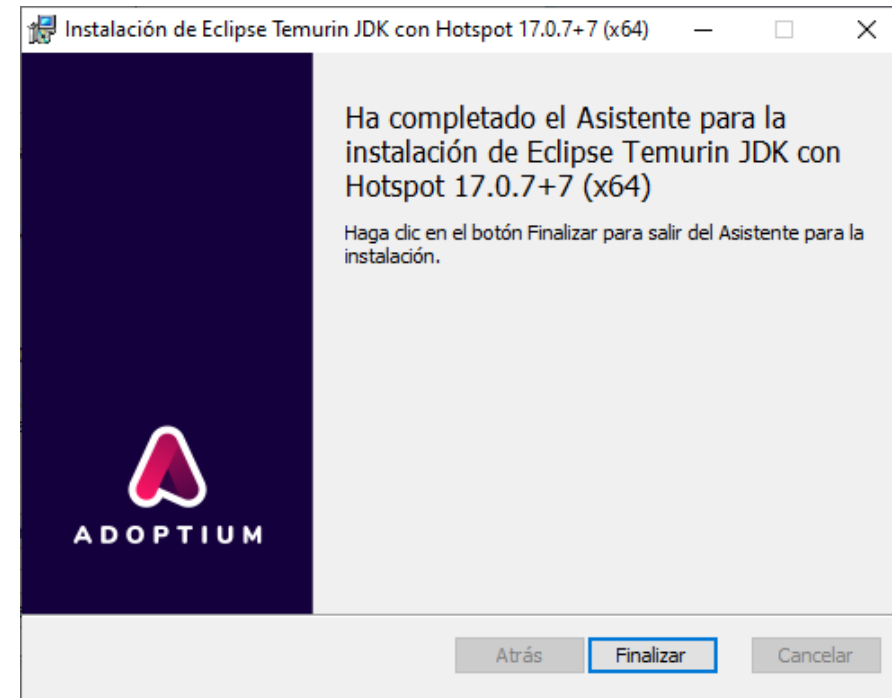
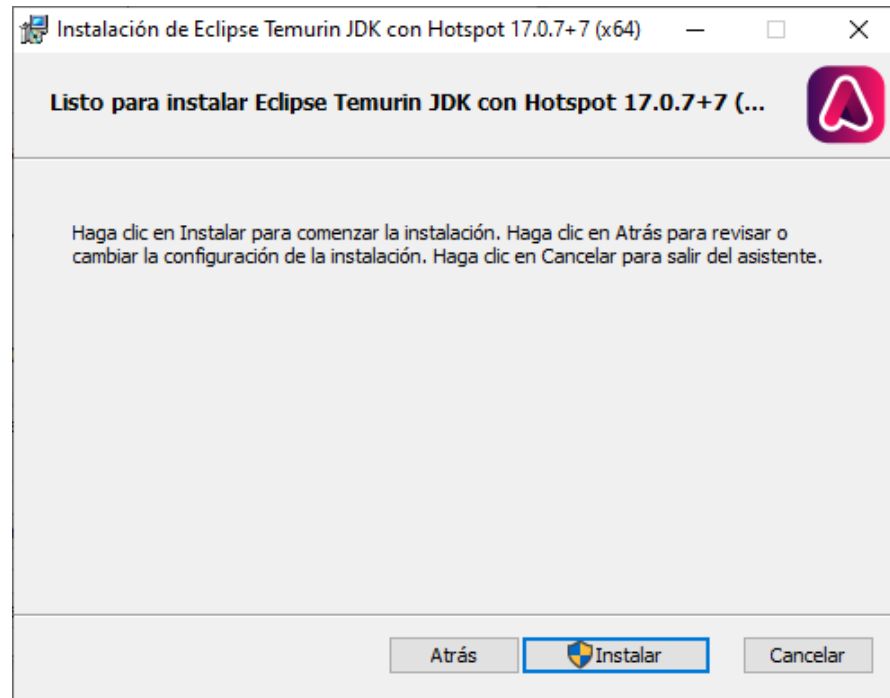
Proceder a la instalación



Verificar que estas tres opciones estén seleccionadas, como se muestra en la imagen.

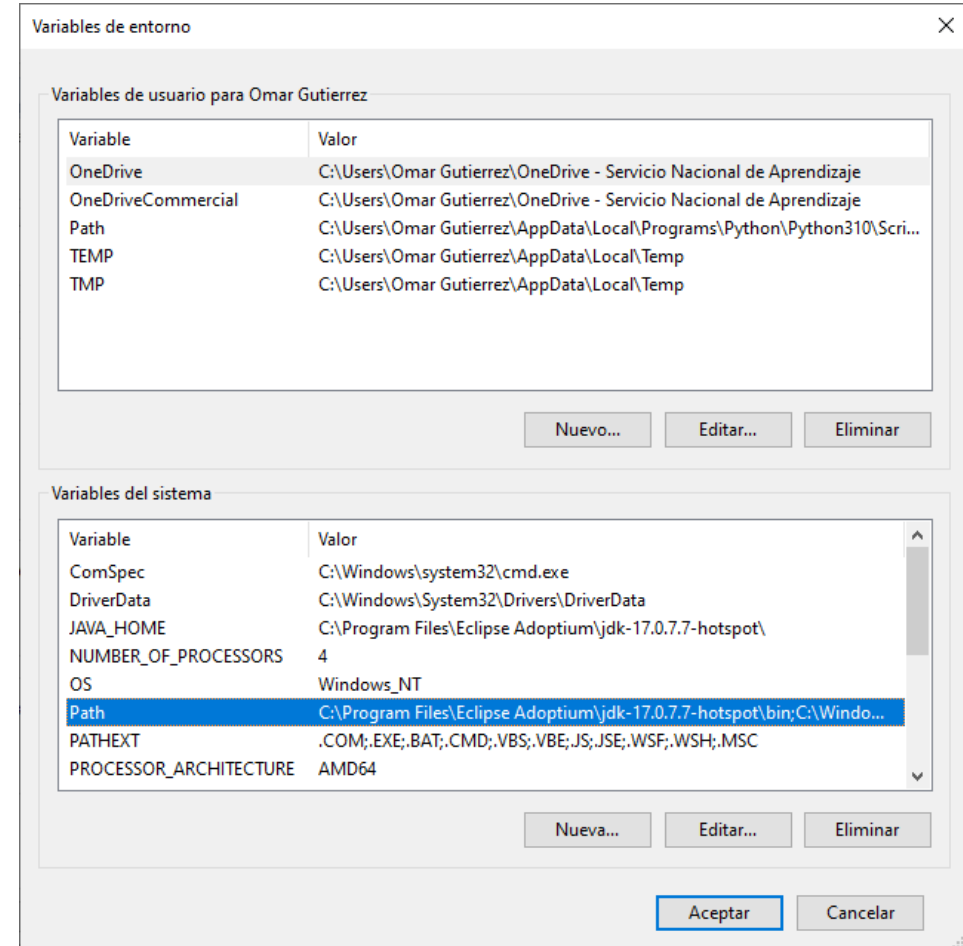
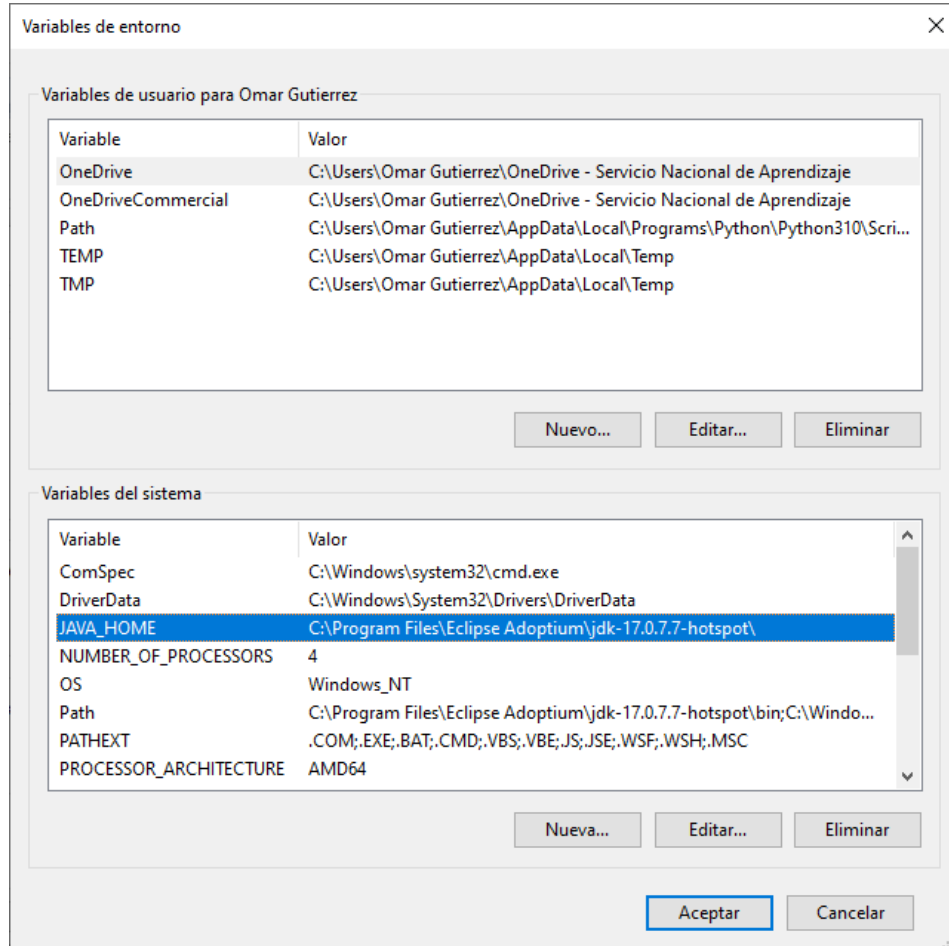
Instalación JDK de Java

Proceder a la instalación



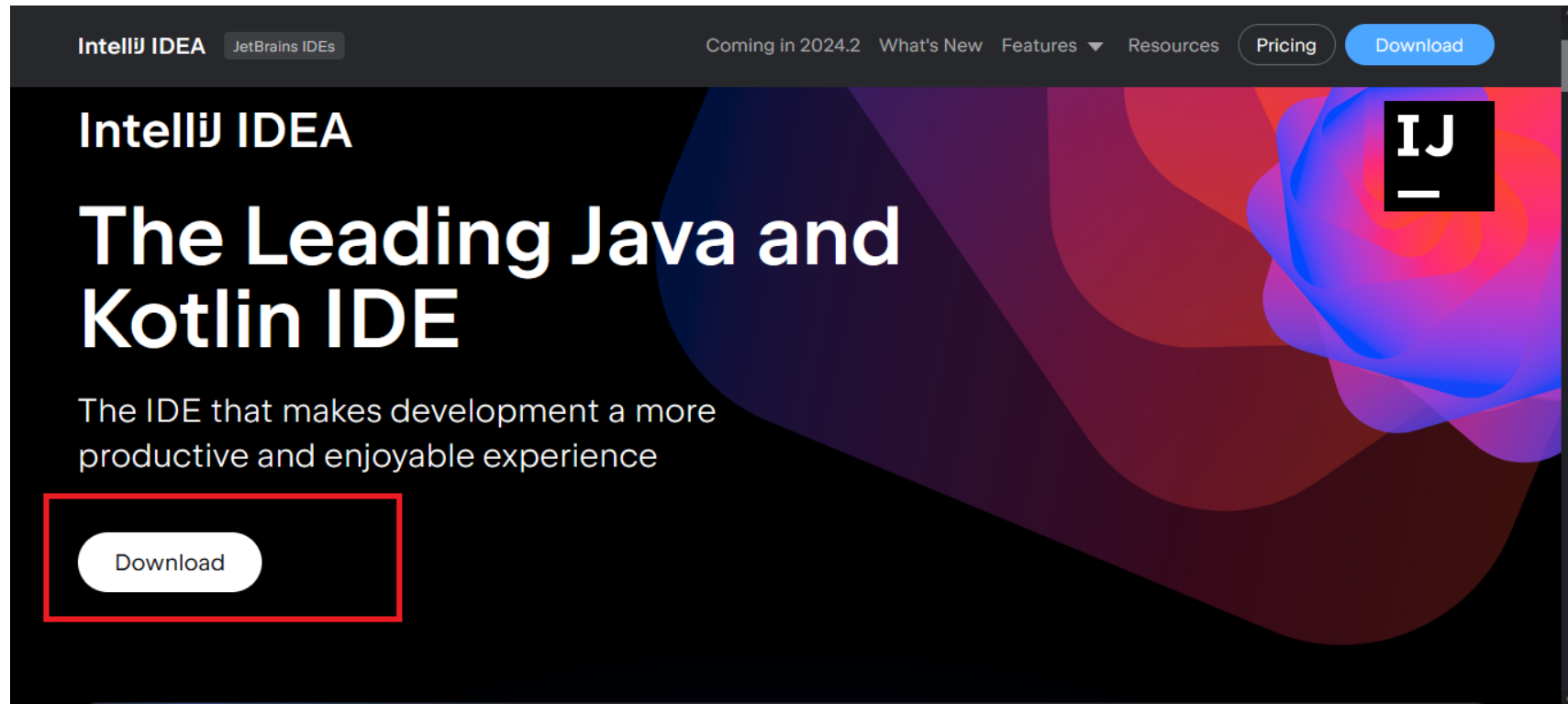
Instalación JDK de Java

Verificar variables de Entorno JAVA_HOME y Path



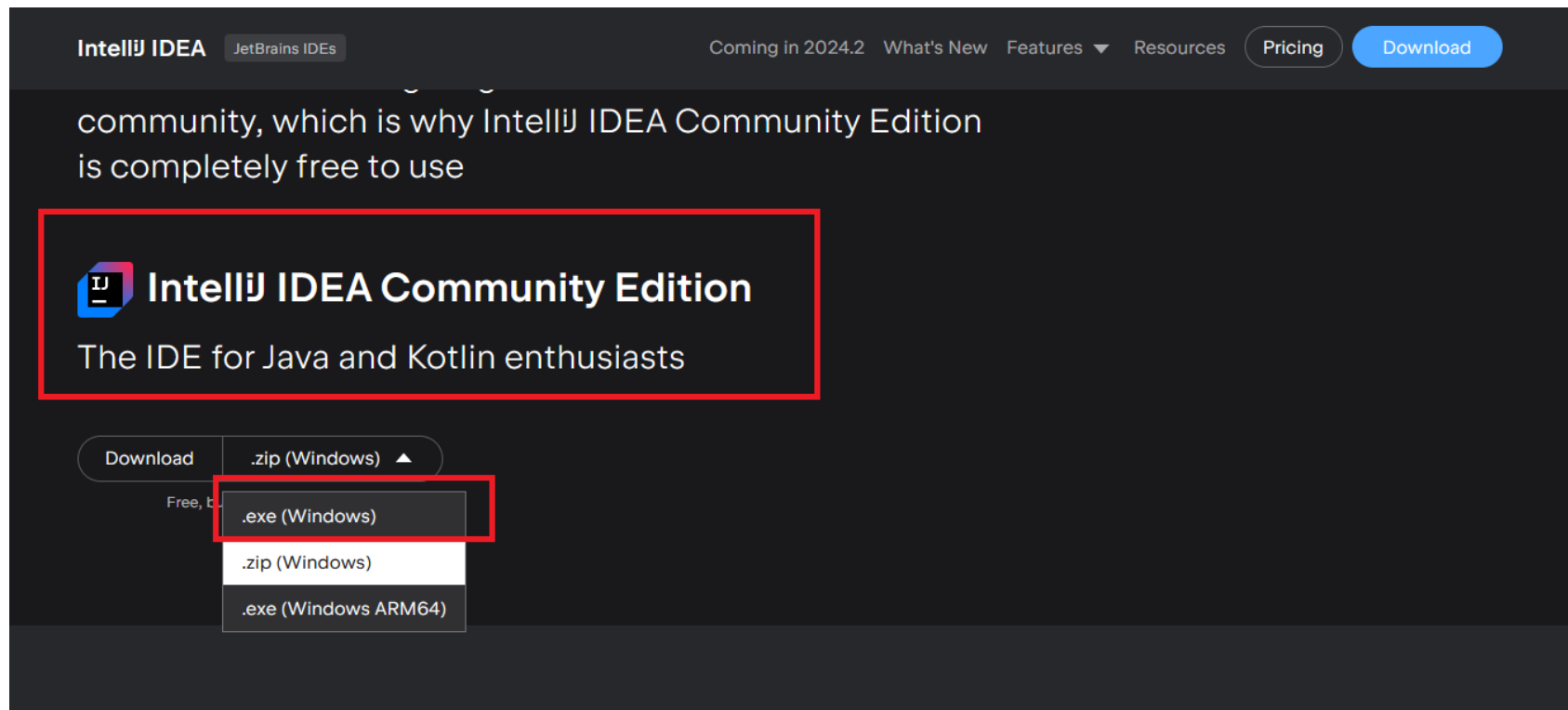
Instalación de Kotlin (IntelliJ IDEA)

Descargar el archivo de IntelliJ IDEA de: <https://www.jetbrains.com/idea/>

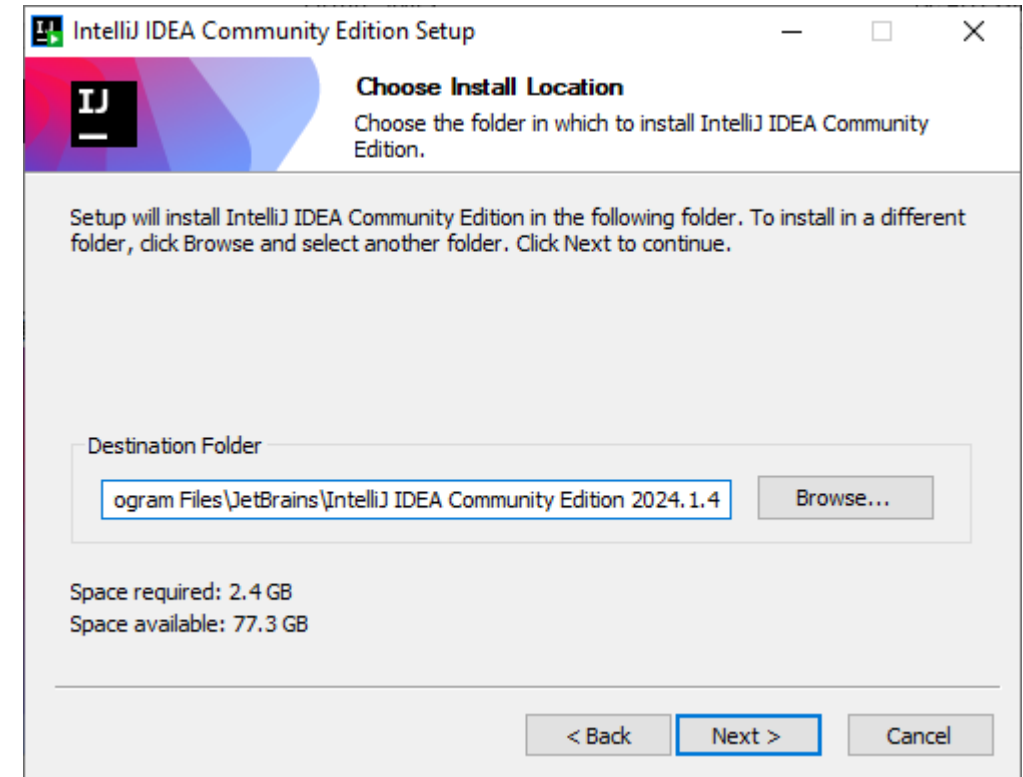


Instalación de Kotlin (IntelliJ IDEA)

Bajamos y ubicamos la versión Community y desplegamos en Download y escogemos **.exe (Windows)**

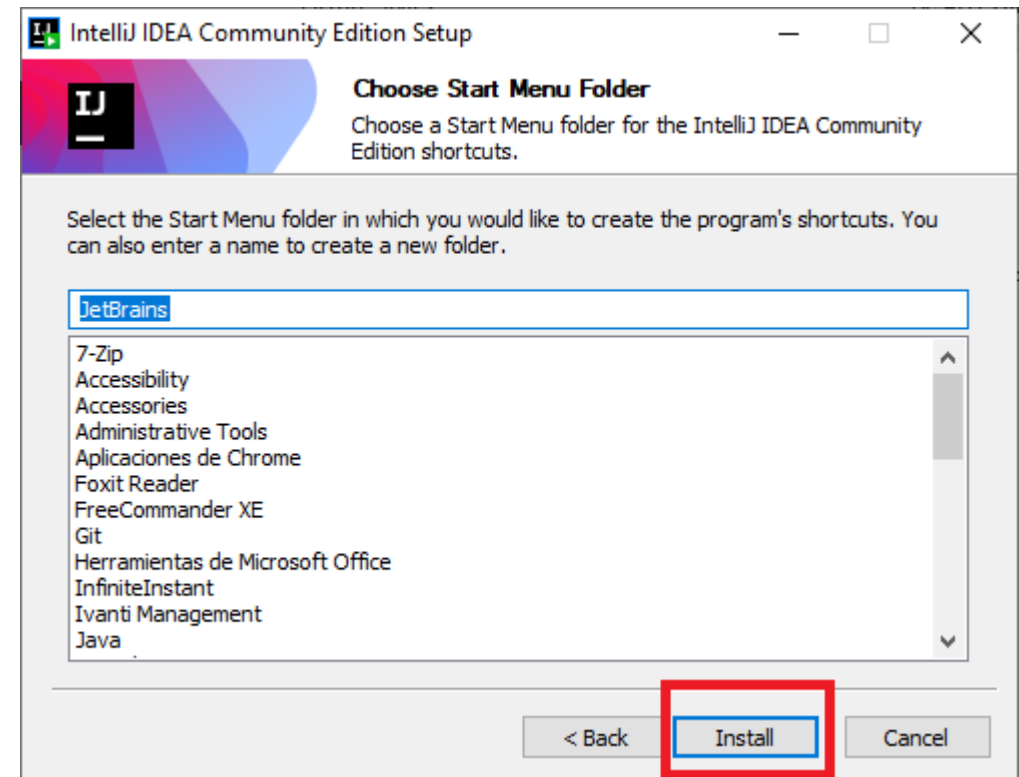
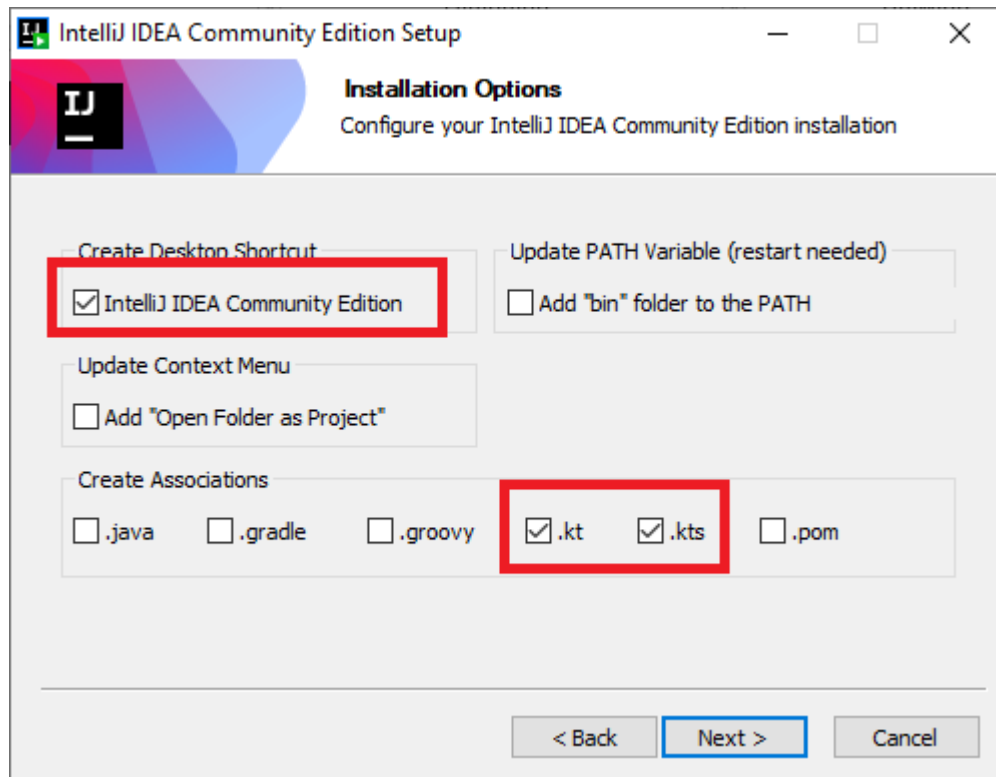


Instalación de Kotlin (IntelliJ IDEA)

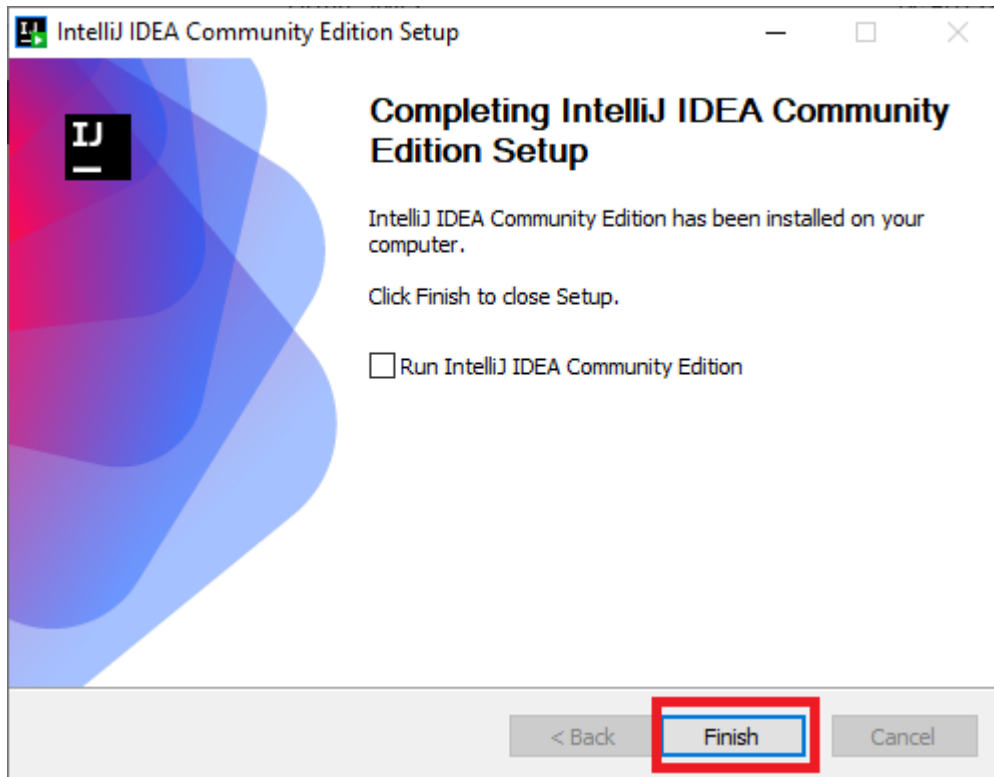


Instalación de Kotlin (IntelliJ IDEA)

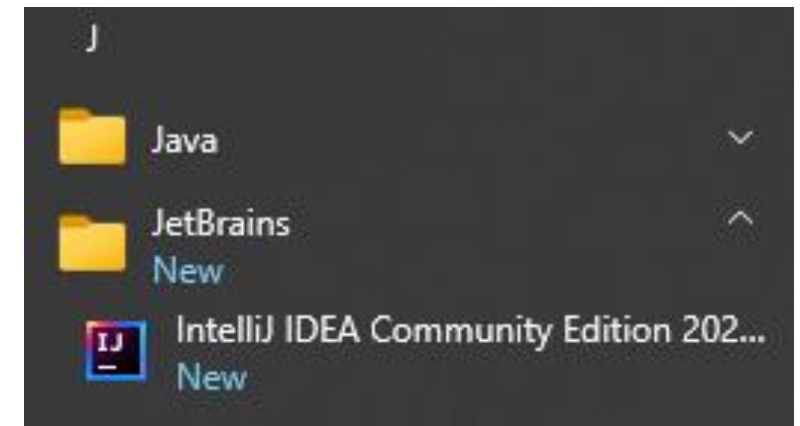
En **Create associations** seleccionaremos los **.kt** y **.kts** que es el formato de los archivos de kotlin



Instalación de Kotlin (IntelliJ IDEA)

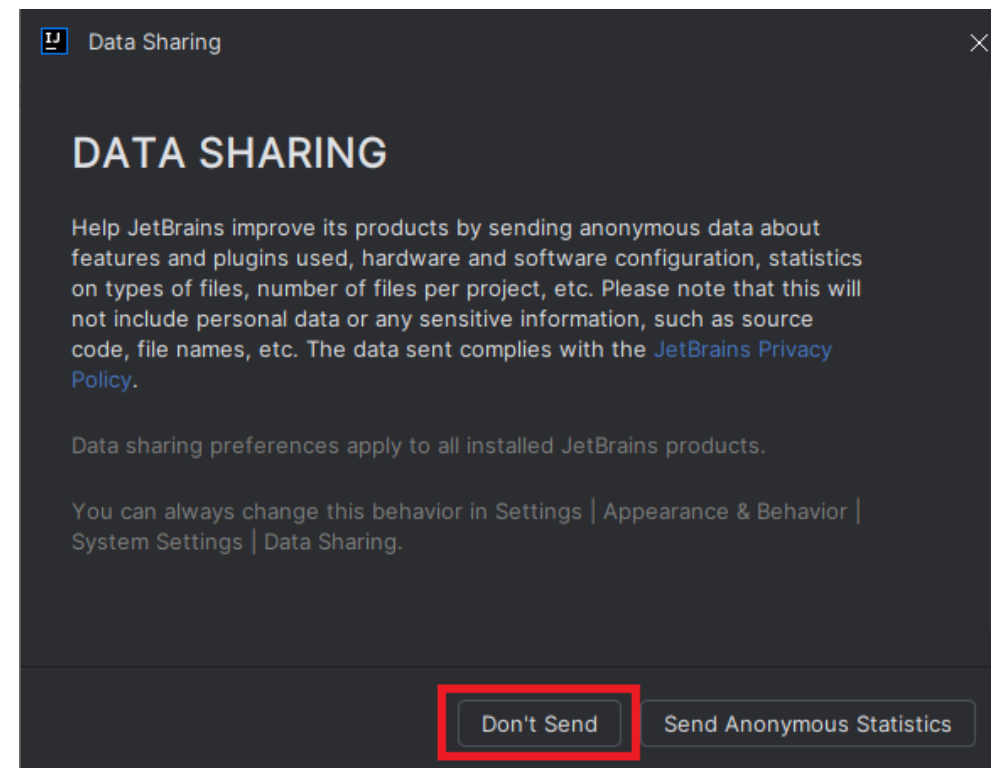
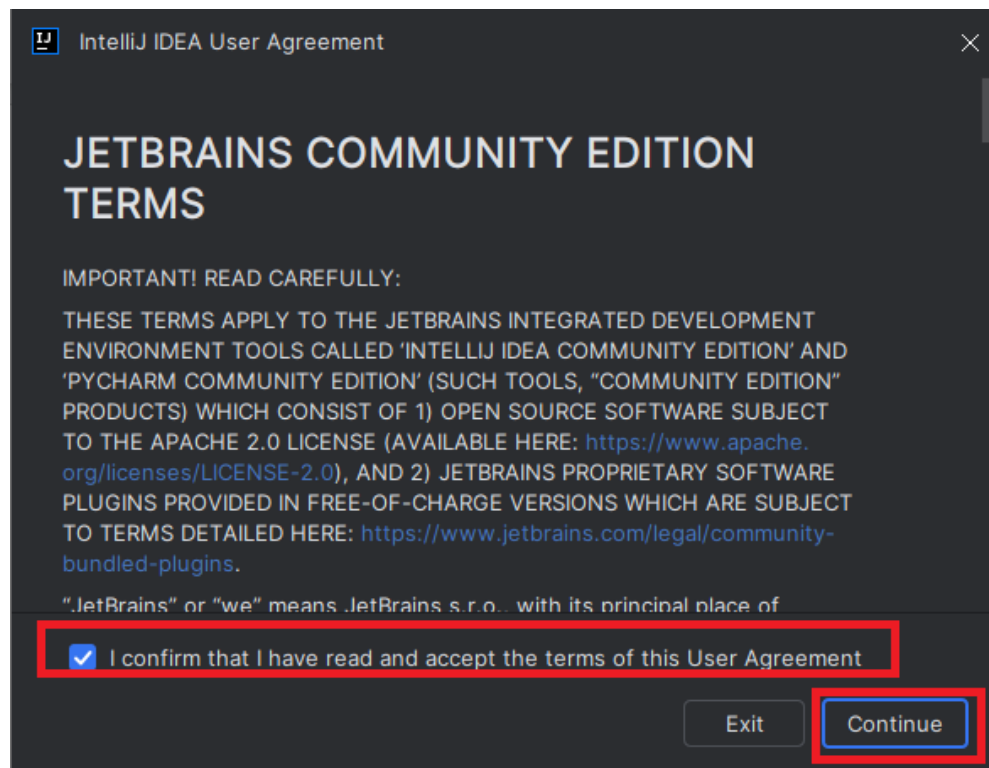


Ahora desde el menú de opciones de Windows podemos iniciar el entorno de desarrollo IntelliJ IDEA:



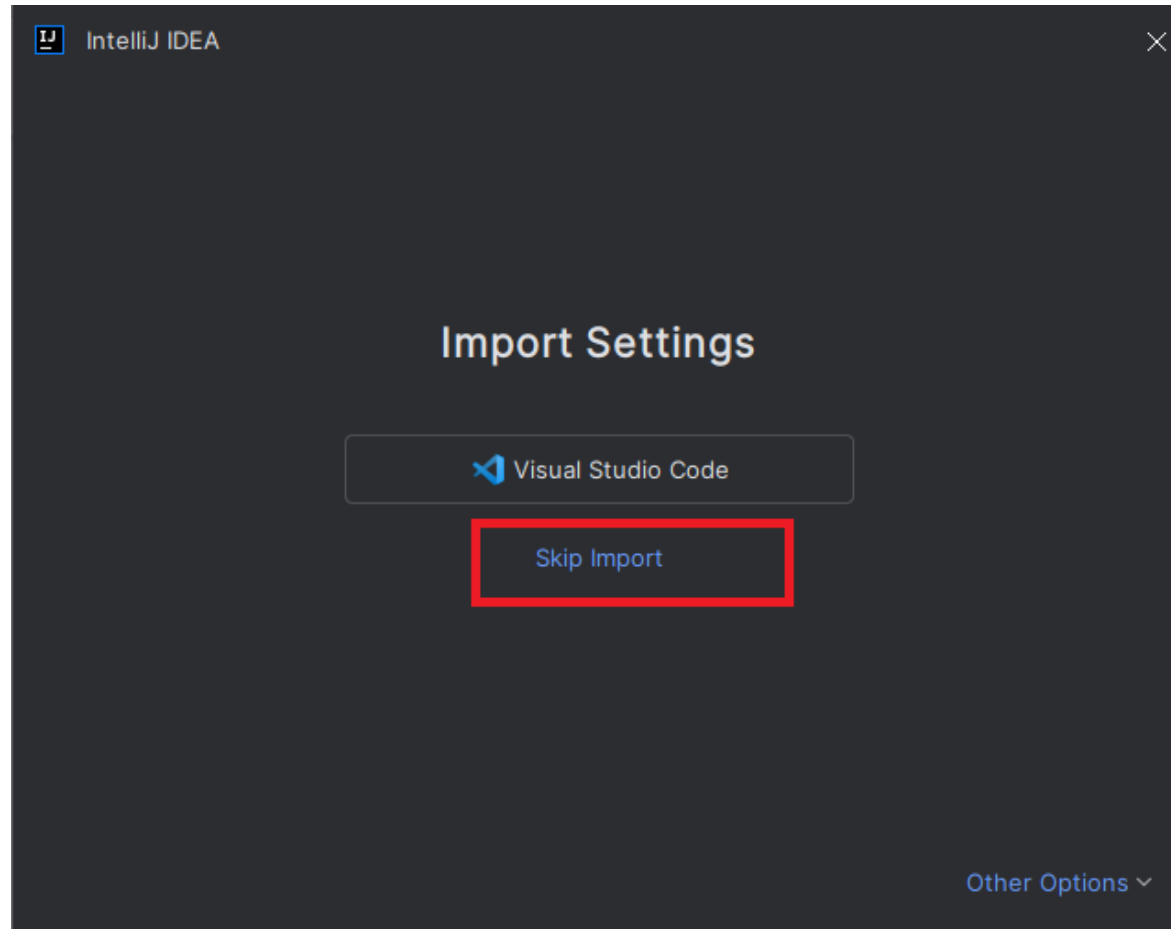
Instalación de Kotlin (IntelliJ IDEA)

La primera vez que lo ejecutamos aceptamos licencia, no compartimos data,



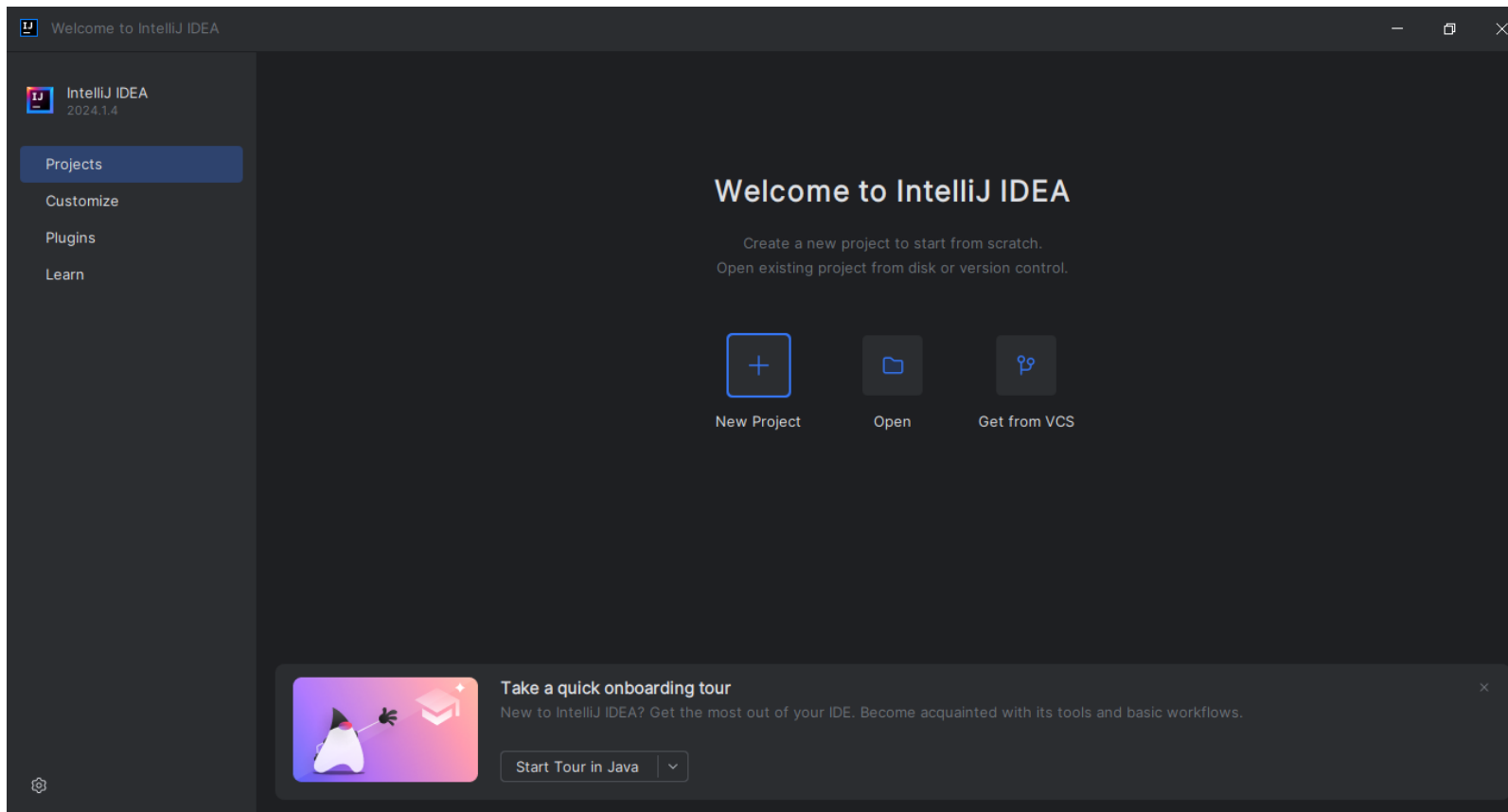
Instalación de Kotlin (IntelliJ IDEA)

La primera vez que lo ejecutamos nos pregunta si queremos importar alguna configuración previa (elegimos que no):



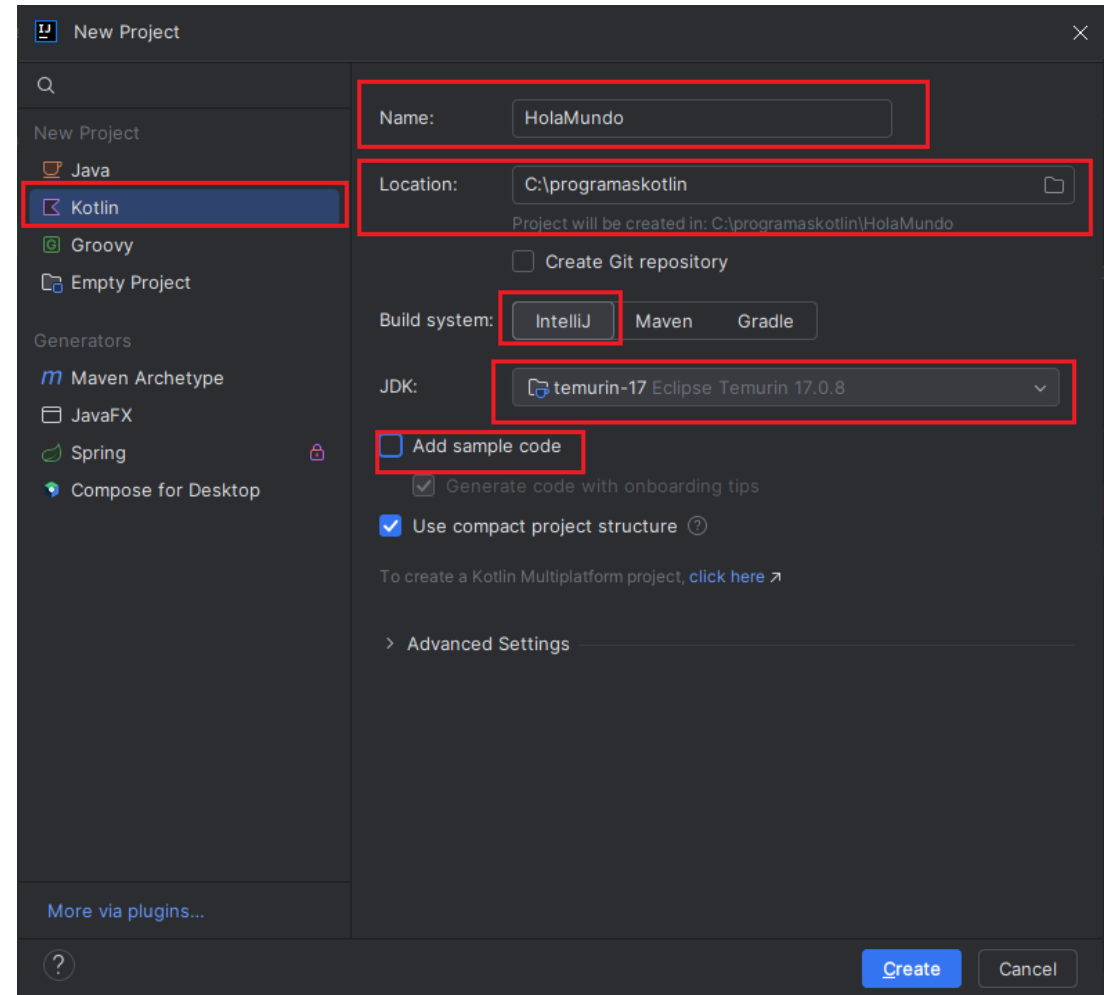
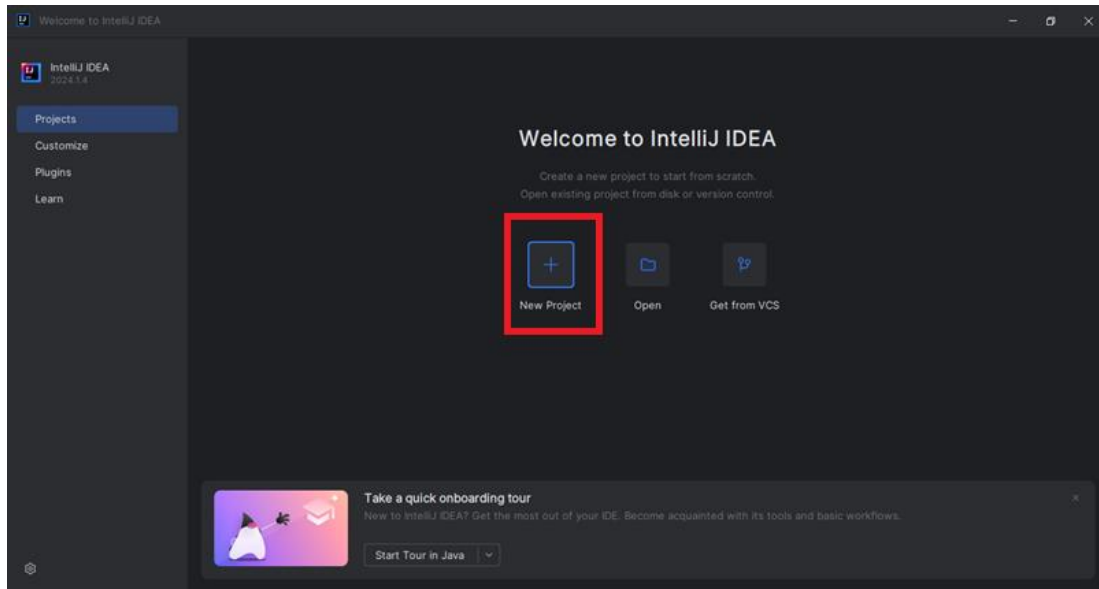
Instalación de Kotlin (IntelliJ IDEA)

Ahora si tenemos la pantalla que aparecerá siempre que necesitemos crear un nuevo proyecto o abrir otros existentes:



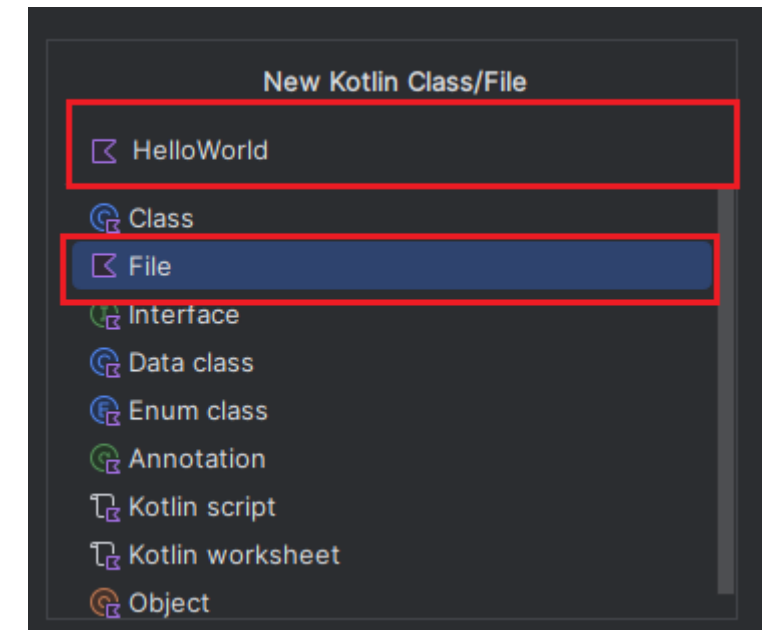
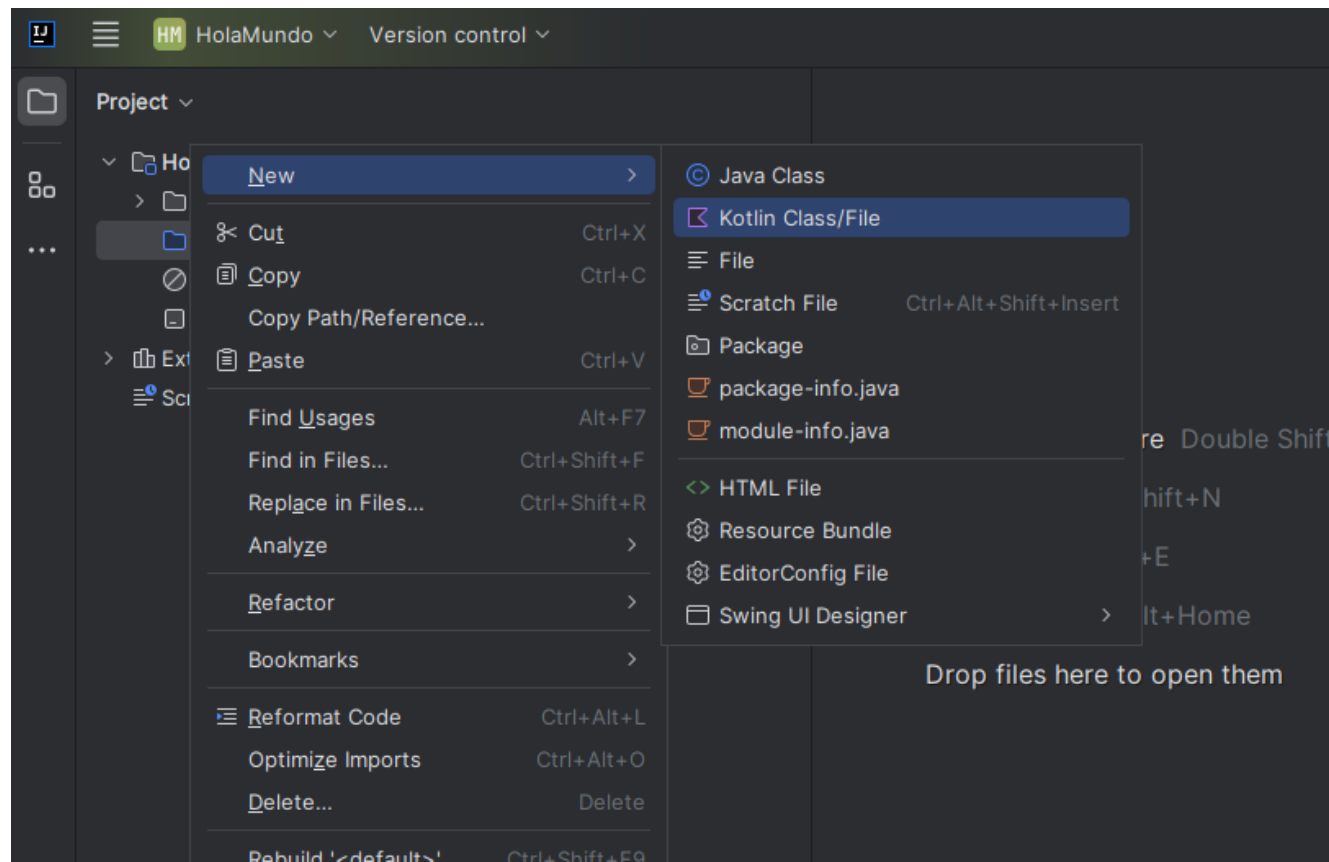


Crear un programa en Kotlin

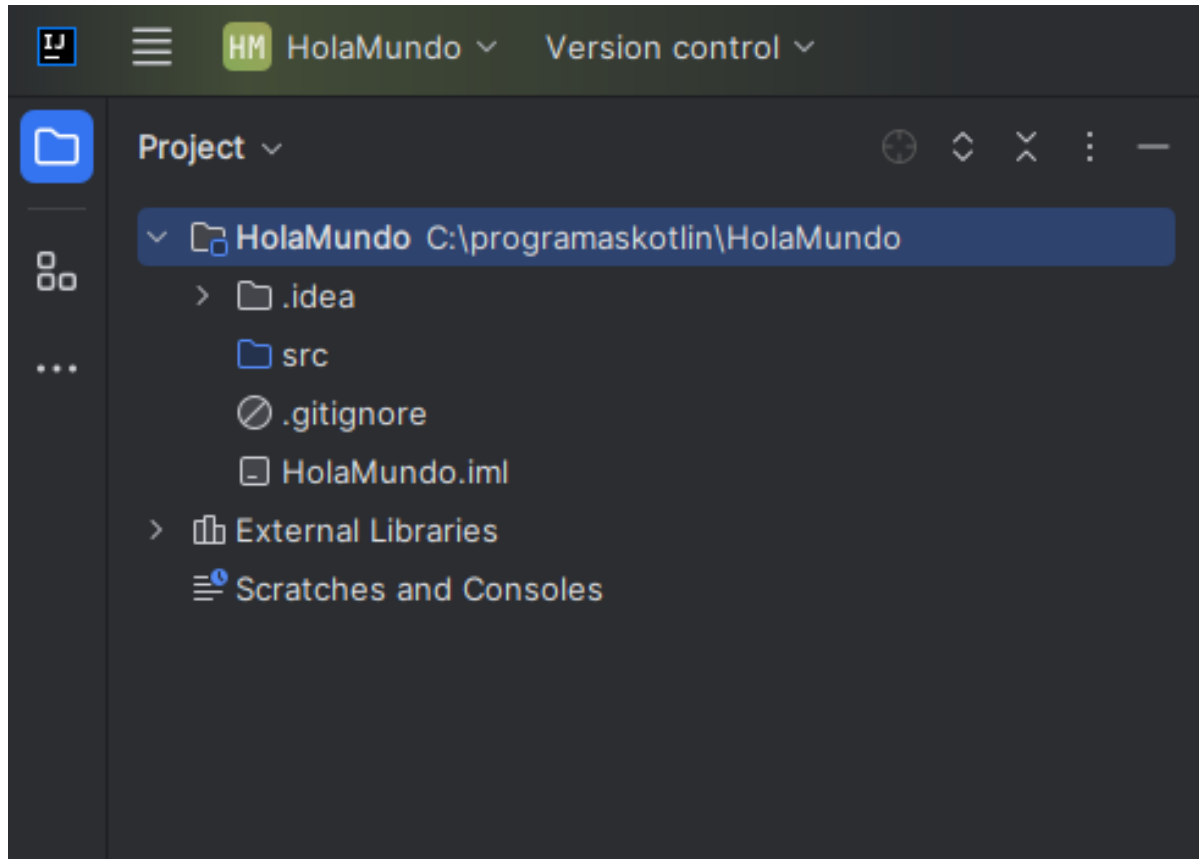


Crear un programa en Kotlin

Vamos a la carpeta src, botón secundario New>Kotlin File/Class, seguidamente nos saldrá una ventana donde daremos nombre **HelloWorld** y daremos doble clic sobre **File**



Crear un programa en Kotlin

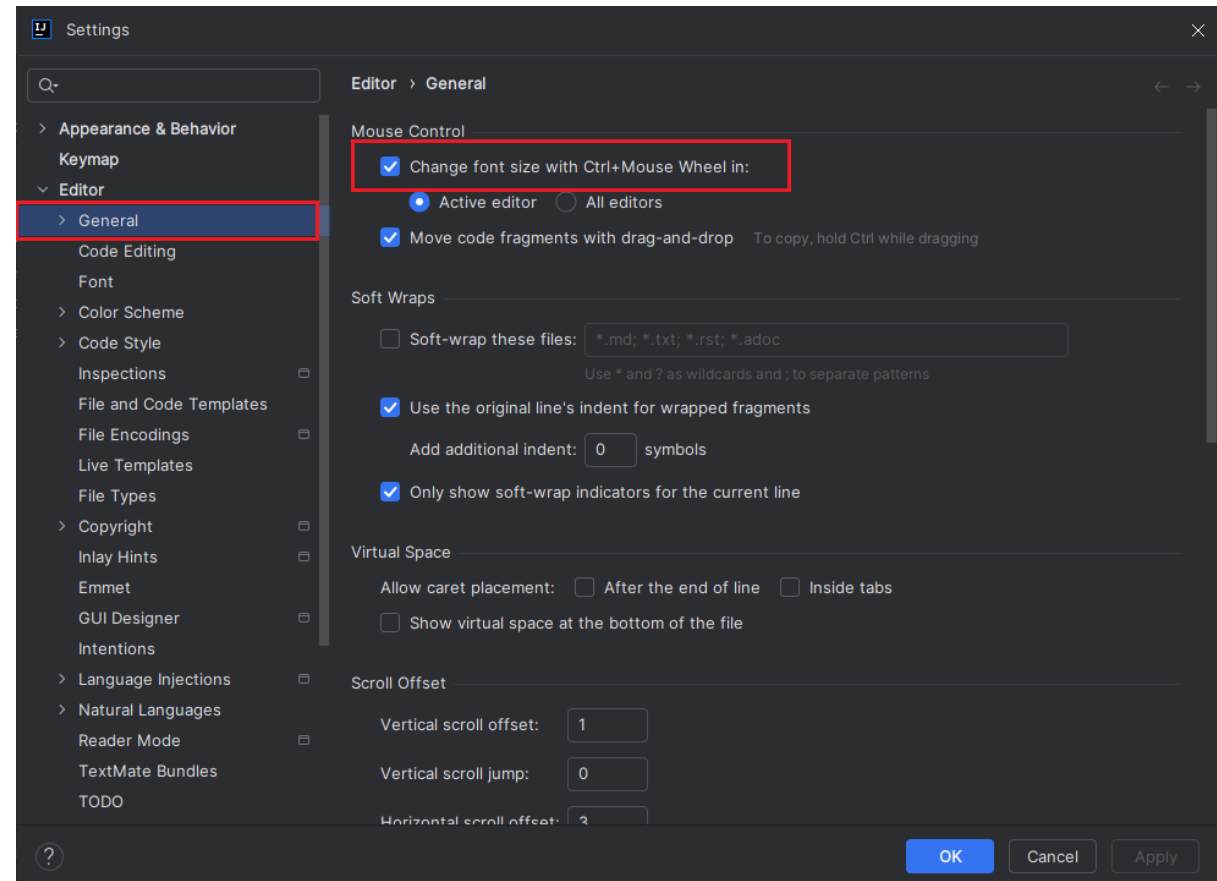
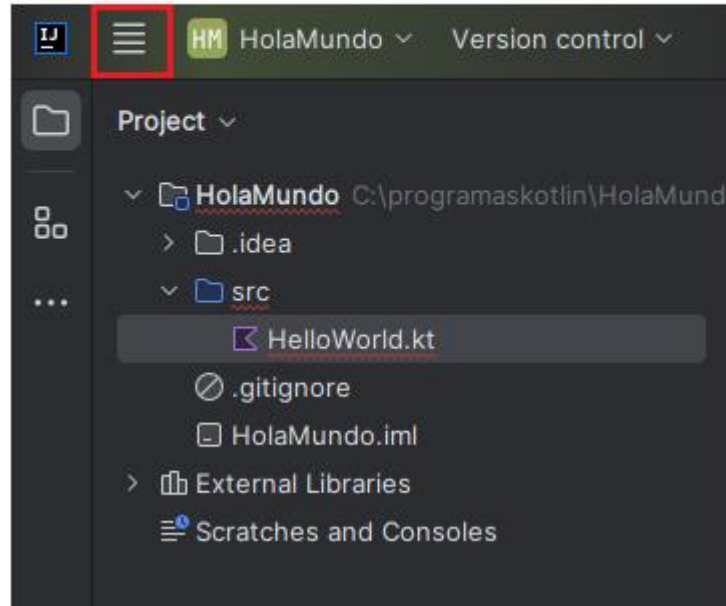


La carpeta `.idea` y el archivo `.iml` son ficheros de configuración del IDE. La carpeta `src` que será donde crearemos los ficheros y directorios para trabajar.

Activar Zoom con Ctrl+Rueda Ratón

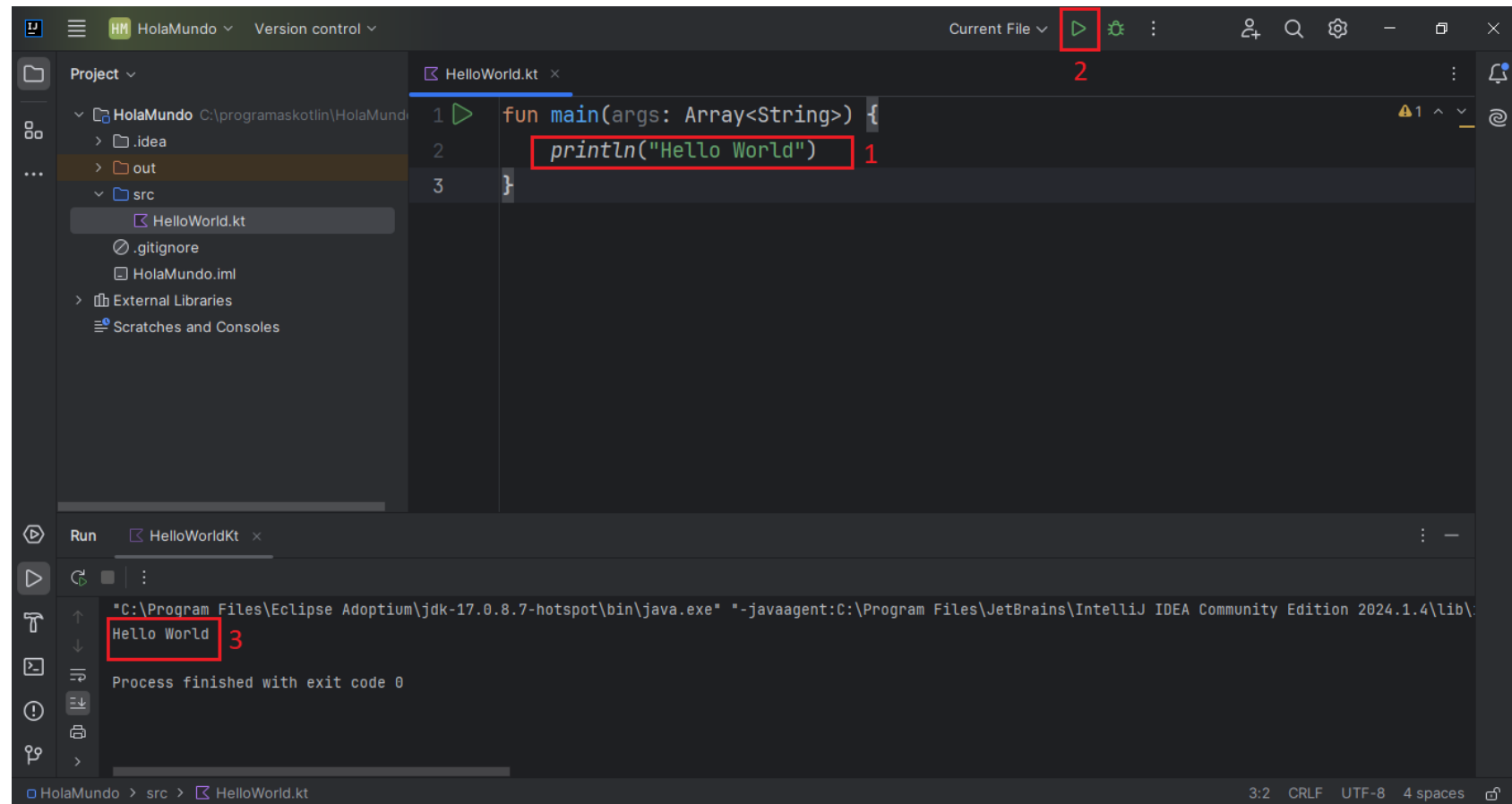
File – Settings – Editor – General – Mouse – «Change font size (Zoom) with Ctrl+Mouse Wheel

Activar Menús



Crear un programa en Kotlin

Dentro del método añadiremos la línea de código para imprimir en pantalla Hello World. También hay que destacar que Kotlin es sensible a mayúsculas y minúsculas.



The screenshot shows the IntelliJ IDEA IDE interface. On the left, the Project Explorer displays the 'HolaMundo' project structure, including the 'src' directory and the 'HelloWorld.kt' file. The main editor window shows the Kotlin code for 'HelloWorld.kt':

```
1 fun main(args: Array<String>) {  
2     println("Hello World")  
3 }
```

Annotations in the code: A red box highlights the `println("Hello World")` line, with a red '1' next to it. A red box highlights the green run button in the top toolbar, with a red '2' next to it.

At the bottom, the Run console shows the execution of the program. The command line is: `"C:\Program Files\Eclipse Adoptium\jdk-17.0.8.7-hotspot\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.1.4\lib\"`. The output is `Hello World`, which is highlighted with a red box and a red '3'. Below the output, it says 'Process finished with exit code 0'.



GRACIAS

Línea de atención al ciudadano: 01 8000 910270
Línea de atención al empresario: 01 8000 910682



www.sena.edu.co