

INFORME DE LABORATORIO 1

Introducción y Kotlin Esencial



PROGRAMACIÓN EN MÓVILES

Docente: Benjamin Pareja

Programa: Diseño y Desarrollo de
Software

2024 II



Developers

Alumno(s):	<i>Nina Mayta Milward Fernando</i>				Nota	
Grupo:	<i>D</i>	Ciclo: 4				
<i>Criterio de Evaluación</i>	<i>Excelente (4pts)</i>	<i>Bueno (3pts)</i>	<i>Requiere mejora (2pts)</i>	<i>No accept. (0pts)</i>	<i>Puntaje Logrado</i>	
Desarrolla adecuadamente los ejercicios teóricos					<i>3</i>	
Desarrolla adecuadamente los ejercicios prácticos					<i>6</i>	
Realiza una correcta instalación de Android Studio					<i>3</i>	
Realiza observaciones y conclusiones que aporten un opinión crítica y técnica					<i>3</i>	
Es puntual y redacta el informe adecuadamente sin copias de otros autores					<i>2</i>	
Evidencia avance en laboratorio					<i>3</i>	

Laboratorio 01: Introducción y Kotlin Esencial

I. Objetivo

- Comprender los diversos tipos de variables en Kotlin y cómo declararlas.
- Utilizar la función println para mostrar información en la consola en Kotlin.
- Crear funciones en Kotlin con argumentos y valores de retorno.
- Utilizar estructuras de control para tomar decisiones.

II. Seguridad

- No ingresar con líquidos, ni comida al aula de Laboratorio.
- Al culminar la sesión de laboratorio apagar correctamente la computadora y la pantalla, y ordenar las sillas utilizadas.

III. Equipos y Materiales

- Sistema Operativo Windows 10 o superior con conexión a la red del laboratorio
- Instalador de Android Studio

DESARROLLO

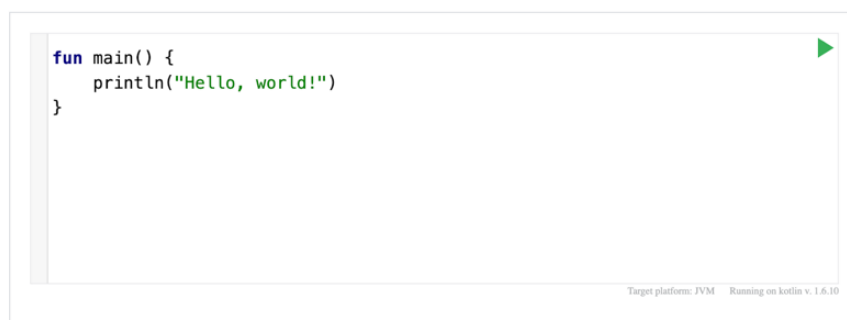
1. Hola Mundo en Kotlin


En un navegador web de tu computadora, abre [Playground de Kotlin](#).

Deberías ver una página web similar a esta imagen:

Kotlin Playground

Try Kotlin and practice what you've learned so far. Type your code in the window below, and click the button to run it!




When you click the run button, the code you wrote will be sent to a third-party server controlled by JetBrains to be compiled. [Learn more about how JetBrains collects and uses data.](#)  Kotlin is a registered trademark of the Kotlin Foundation.

Programación en Móviles

Ya existe un código predeterminado propagado en el editor de código. Estas tres líneas de código conforman un programa simple:

```
1 fun main() {  
2     println("Hello, world!")  
3 }
```

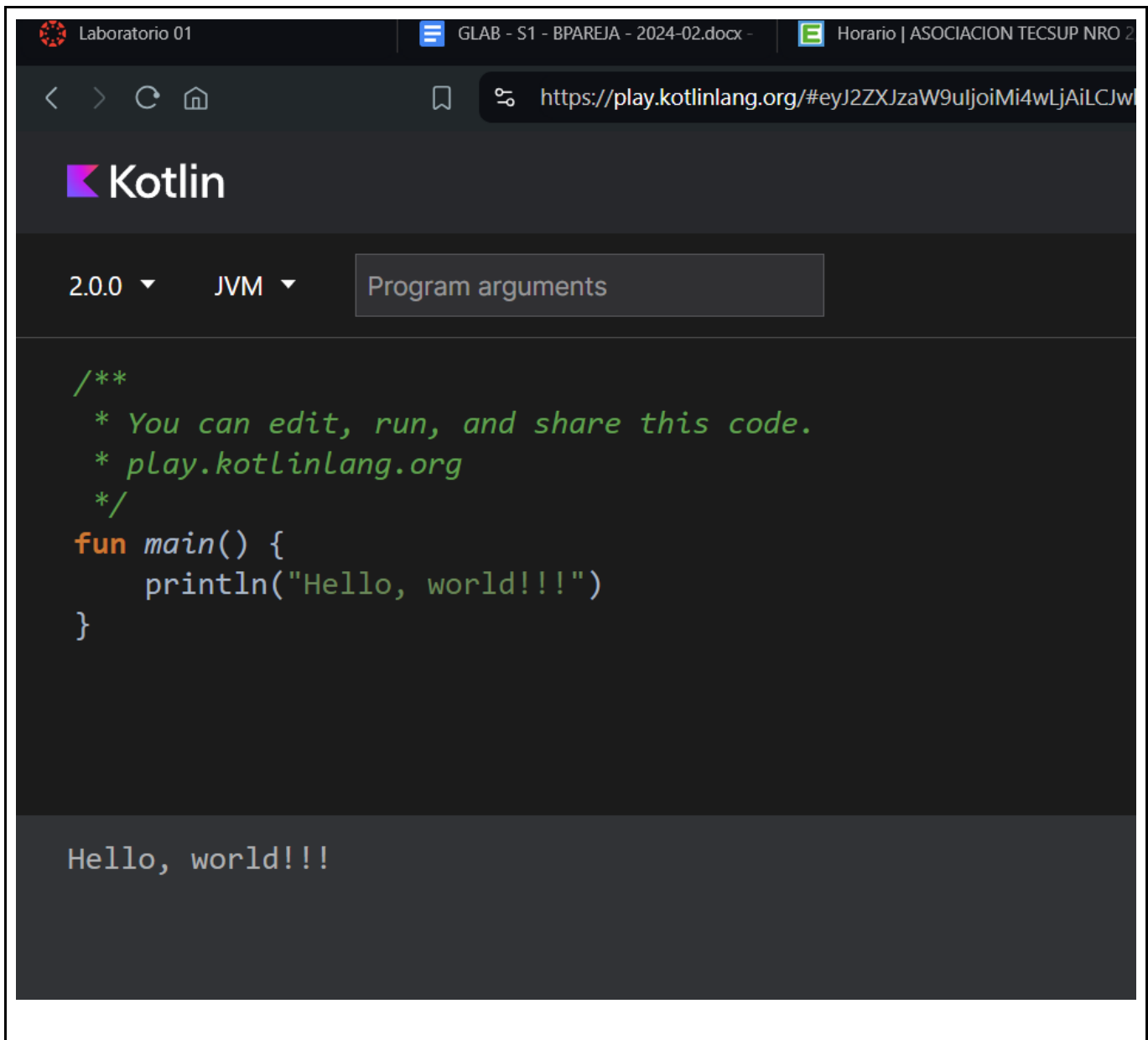
Haz clic en  RUN para ejecutar el programa.

Cuando haces clic en el botón Ejecutar, suceden muchas cosas. El objetivo del código en el lenguaje de programación Kotlin es que los humanos lo entiendan, de modo que las personas puedan leer, escribir y colaborar más fácilmente en programas de Kotlin. Sin embargo, la computadora no comprende este idioma de inmediato.

Necesitas algo llamado el *compilador de Kotlin*, que toma el código Kotlin que escribiste, lo analiza línea por línea y lo traduce a algo que la computadora puede comprender. Este proceso se denomina *compilación* de código.

Si tu código se compila correctamente, el programa se ejecutará. Cuando la computadora ejecuta tu programa.

Muestra el resultado del código ejecutado:



2. Cuerpo de la función

El *cuerpo de la función* contiene las instrucciones necesarias para lograr el propósito de la función. Para buscar el cuerpo de la función, busca las líneas de código encerradas entre llaves de apertura y cierre.

```
fun name ( inputs ) {
    

body


}
```

3. Explicación de un programa sencillo

Observa el programa sencillo que viste antes en el codelab.

```

      name inputs
      ↓   ↓
fun main() {
    println("Hello, world!") ← body
}

```

El programa contiene una función: la función principal. "main" es un nombre de función especial en Kotlin. Cuando escribes tu código en el Playground de Kotlin, debes escribirlo dentro de la función `main()` o llamarlo desde la función `main()`.

A continuación, se incluyen algunas de las recomendaciones de la guía de estilo para lo que aprendiste en Kotlin hasta el momento:

- Los nombres de las funciones deben seguir la convención de mayúsculas y minúsculas, y deben ser verbos o frases verbales.
- Cada sentencia debe estar en su propia línea.
- La llave de apertura debe aparecer al final de la línea donde comienza la función.
- Debe haber un espacio antes de la llave de apertura.

La sintaxis para declarar una función con un tipo de datos que se devuelve es la siguiente.

```

fun [name] () : [return type] {
    [body]
    [return statement]
}

```

Programación en Móviles

De forma predeterminada, si no se especifica un tipo de datos que se muestra, el predeterminado es `Unit`. `Unit` significa que la función no muestra ningún valor.

A fin de demostrar cómo una función puede mostrar un valor, deberás realizar la función `courseGreeting()` para que muestre una string, en lugar de simplemente imprimir el resultado.

```
1 fun main() {
2     courseGreeting()
3 }
4
5 fun courseGreeting(): String {
6     val nameGreeting = " Bienvenido al curso, Jaime!"
7     val semesterGreeting = " Estás en 4to semestre!"
8     return "$nameGreeting\n$semesterGreeting"
9 }
```



The screenshot shows a code editor with the following Kotlin code:

```
* play.kotlinlang.org
*/
fun main() {
    println(courseGreeting())
}

fun courseGreeting(): String {
    val nameGreeting = " Bienvenido al curso, Jaime!"
    val semesterGreeting = " Estás en 4to semestre!"
    return "$nameGreeting\n$semesterGreeting"
}
```

Below the code, the output is displayed in a console window:

```
Bienvenido al curso, Jaime!
Estás en 4to semestre!
```

Busca en la documentación y explica cómo agregar un parámetro y varios parametros a la función `courseGreeting()`:

Para agregar un parámetro a la función `courseGreeting()`, simplemente se debe incluir el nombre y el tipo de dato del parámetro entre los paréntesis de la función. Por ejemplo, si quisiera añadir un parámetro `studentName` de tipo `String`, lo haría de esta manera: `fun courseGreeting(studentName: String): String`.

Y si quisiera agregar varios parámetros, solo tendría que separar cada uno con una coma. Por ejemplo, si deseo añadir un parámetro adicional llamado `semester` de tipo `Int`, la firma de la función se vería así: `fun courseGreeting(studentName: String, semester: Int): String`.

¡Con estos cambios, sería capaz de personalizar aún más mi función `courseGreeting()` al incluir información específica del estudiante, como su nombre y el semestre en el que se encuentra! ¡Espero que esta explicación te haya sido de ayuda! ¡Si tienes más preguntas o necesitas más ayuda, no dudes en decírmelo!

```
/**
 * Agregar un parámetro studentName al saludo:
 */
fun main() {
    val studentName = "Luis"
    println(courseGreeting(studentName))
}

fun courseGreeting(studentName: String): String {
    val nameGreeting = "¡Bienvenido al curso, $studentName!"
    val semesterGreeting = "¡Estás en 4to semestre!"
    return "$nameGreeting\n$semesterGreeting"
}
```

```
¡Bienvenido al curso, Luis!
¡Estás en 4to semestre!
```

```
/**
 * Agregar parámetros studentName y semester al saludo
 */
fun main() {
    val studentName = "Maria"
    val semester = "2do"
    println(courseGreeting(studentName, semester))
}

fun courseGreeting(studentName: String, semester: String): String {
    val nameGreeting = "¡Bienvenido al curso, $studentName!"
    val semesterGreeting = "¡Estás en $semester semestre!"
    return "$nameGreeting\n$semesterGreeting"
}
```

```
¡Bienvenido al curso, Maria!
¡Estás en 2do semestre!
```

4. Tipos de datos

Programación en Móviles

Tipo de datos de Kotlin	Qué tipo de datos puede contener	Ejemplos de valores literales
String	Texto	"Add contact" "Search" "Sign in"
Int	Número entero	32 1293490 -59281
Double	Número decimal	2.0 501.0292 -31723.99999
Float	Número decimal (que es menos preciso que un Double). Tiene un f o F al final del número.	5.0f -1630.209f 1.2940278F
Boolean	true o false . Usa este tipo de datos cuando solo haya dos valores posibles. Ten en cuenta que true y false son palabras clave en Kotlin.	true false

Si necesitas actualizar el valor de una variable, declara la variable con la palabra clave de Kotlin **var**, en lugar de **val**.

- Palabra clave **val**: Úsala cuando esperes que el valor de la variable no cambie.
- Palabra clave **var**: Úsala cuando esperes que el valor de la variable pueda cambiar.

Ingresa este código en el Playground de Kotlin. ¿Puedes entender lo que sucede en cada línea de código?

```
1 fun main() {
2     val trip1: Double = 3.20
3     val trip2: Double = 4.10
4     val trip3: Double = 1.72
5     val totalTripLength: Double = 0.0
6     println("$totalTripLength miles left to destination")
7 }
```

```
* Agregar parámetros studentName y semester al saludo
*/
fun main() {
    val trip1: Double = 3.20
    val trip2: Double = 4.10
    val trip3: Double = 1.72
    val totalTripLength: Double = 0.0
    println("$totalTripLength miles left to destination")
}
```

```
0.0 miles left to destination
```

En la función se han creado cuatro variables de tipo Double con diferentes valores para representar las distancias de varios viajes. Posteriormente, se imprime el valor de la cuarta variable, totalTripLength, que se inicializa en 0.0

5. Ejercicios

Basandote en la [documentación oficial de kotlin](#), resuelve los siguientes ejercicios:

- ¿Puedes escribir una función `main()` que imprima estos mensajes en cuatro líneas separadas?

Use the val keyword when the value doesnt change.

Use the var keyword when the value can change.

When you define a function, define the parameters that can be passed to it.

When you call a function, you pass arguments for the parameters.

```

/* Agregar parametros studentname y semester al saludo */
fun main() {
    val message1 = "Use the val keyword when the value doesn't change."
    val message2 = "Use the var keyword when the value can change."
    val message3 = "When you define a function, define the parameters that can be passed to it."
    val message4 = "When you call a function, you pass arguments for the parameters."

    println(message1)
    println(message2)
    println(message3)
    println(message4)
}

```

Use the val keyword when the value doesn't change.
Use the var keyword when the value can change.
When you define a function, define the parameters that can be passed to it.
When you call a function, you pass arguments for the parameters.

Version Espanol

```

fun main() {
    val mensaje1 = "Utiliza la palabra clave val cuando el valor no cambia."
    val mensaje2 = "Utiliza la palabra clave var cuando el valor puede cambiar."
    val mensaje3 = "Al definir una función, especifica los parámetros que pueden pasarse a la misma."
    val mensaje4 = "Al llamar a una función, se pasan argumentos para los parámetros."

    println(mensaje1)
    println(mensaje2)
    println(mensaje3)
    println(mensaje4)
}

```

Utiliza la palabra clave val cuando el valor no cambia.
Utiliza la palabra clave var cuando el valor puede cambiar.
Al definir una función, especifica los parámetros que pueden pasarse a la misma.
Al llamar a una función, se pasan argumentos para los parámetros.

- b) Este programa imprime un mensaje que le notifica al usuario que recibió un mensaje de chat de un amigo:

```
fun main() {  
    println("New chat message from a friend")  
}
```

1. ¿Puedes determinar la causa raíz de los errores de compilación de este programa y corregirlos?
2. ¿El código usa los símbolos apropiados para indicar la apertura y el cierre de la cadena y el argumento de la función?

El error en el programa fue una comilla incorrecta en el mensaje que se quería imprimir. La solución fue corregir esa comilla para que la cadena de texto esté bien formada.

En cuanto a los símbolos apropiados, era importante usar comillas dobles (") para indicar el inicio y fin de la cadena de texto, y asegurarse de que el argumento de la función println() esté dentro de paréntesis.

Con la corrección, el código quedaría así de manera más simple y correcta:

```
fun main() {  
    println("New chat message from a friend")  
}
```

New chat message from a friend

Pista: Puedes usar Kotlin Playground a fin de ejecutar el código y ver los errores de compilación.

Después de corregir los errores, el programa debe compilarse sin problemas y, luego, imprimir este resultado:

New chat message from a friend

- c) En este ejercicio, escribirás un programa que realice operaciones matemáticas básicas y, luego, imprima el resultado.

Paso 1

La función `main()` contiene un error de compilación:

```
fun main() {
    val firstNumber = 10
    val secondNumber = 5

    println("$firstNumber + $secondNumber = $result")
}
```

¿Puedes corregir el error de modo que el programa imprima este resultado?

10 + 5 = 15

Se necesita asignar la suma de `firstNumber` y `secondNumber` a una nueva variable `result` antes de imprimirlo. Aquí está el código corregido:

```
*/
fun main() {
    val firstNumber = 10
    val secondNumber = 5
    val result = firstNumber + secondNumber

    println("$firstNumber + $secondNumber = $result")
}

10 + 5 = 15
```

Paso 2

El código funciona, pero la lógica para sumar dos números se encuentra dentro de la variable de resultado, lo que hace que el código sea menos flexible a la hora de volver a usarlo. En su lugar, puedes extraer la operación de suma en una función `add()` para que el código se pueda volver a usar. Para ello, actualiza el código con el que se muestra a continuación. Observa que el código ahora presenta una nueva variable `val`, llamada `thirdNumber`, e imprime el resultado de esta variable nueva con `firstNumber`.

```
fun main() {
    val firstNumber = 10
    val secondNumber = 5
    val thirdNumber = 8
```

Programación en Móviles

```
val result = add(firstNumber, secondNumber)
val anotherResult = add(firstNumber, thirdNumber)

println("$firstNumber + $secondNumber = $result")
println("$firstNumber + $thirdNumber = $anotherResult")
}

// Define add() function below this line
```

¿Puedes definir la función `add()` de modo que el programa imprima este resultado?

```
10 + 5 = 15
10 + 8 = 18
```

```
fun main() {
    val firstNumber = 10
    val secondNumber = 5
    val thirdNumber = 8

    val result = add(firstNumber, secondNumber)
    val anotherResult = add(firstNumber, thirdNumber)

    println("$firstNumber + $secondNumber = $result")
    println("$firstNumber + $thirdNumber = $anotherResult")
}

fun add(a: Int, b: Int): Int {
    return a + b
}

10 + 5 = 15
10 + 8 = 18
```

Paso 3

Ahora tienes una función reutilizable capaz de sumar dos números.

- ¿Puedes implementar la función `subtract()` de la misma manera en que implementaste la función `add()`? Modifica la función `main()` también para usar la función `subtract()`, de modo que puedas verificar que funcione como se espera.

```

fun main() {
    val firstNumber = 10
    val secondNumber = 5
    val thirdNumber = 8

    val addResult = add(firstNumber, secondNumber)
    val subtractResult = subtract(firstNumber, secondNumber)
    val anotherAddResult = add(firstNumber, thirdNumber)
    val anotherSubtractResult = subtract(firstNumber, thirdNumber)

    println("$firstNumber + $secondNumber = $addResult")
    println("$firstNumber - $secondNumber = $subtractResult")
    println("$firstNumber + $thirdNumber = $anotherAddResult")
    println("$firstNumber - $thirdNumber = $anotherSubtractResult")
}

fun add(a: Int, b: Int): Int {
    return a + b
}

fun subtract(a: Int, b: Int): Int {
    return a - b
}

10 + 5 = 15
10 - 5 = 5
10 + 8 = 18
10 - 8 = 2

```

Pista: Piensa en la diferencia entre la suma, la resta y otras operaciones matemáticas. Comienza a trabajar en el código de solución a partir de allí.

- d) En este programa, se muestra el clima de diferentes ciudades. Incluye el nombre de la ciudad, las temperaturas máxima y mínima del día, y las probabilidades de lluvia.

```

fun main() {
    println("City: Ankara")
    println("Low temperature: 27, High temperature: 31")
    println("Chance of rain: 82%")
    println()

    println("City: Tokyo")
    println("Low temperature: 32, High temperature: 36")
    println("Chance of rain: 10%")
    println()
}

```

```
println("City: Cape Town")
println("Low temperature: 59, High temperature: 64")
println("Chance of rain: 2%")
println()

println("City: Guatemala City")
println("Low temperature: 50, High temperature: 55")
println("Chance of rain: 7%")
println()
}
```

Hay muchas similitudes en el código que imprime el clima de cada ciudad. Por ejemplo, hay frases que se repiten varias veces, como "City:" y "Low temperature:". Los códigos similares y repetidos crean el riesgo de que se produzcan errores en tu programa. Puede que hayas escrito mal una de las ciudades o que hayas olvidado uno de los detalles del clima.

1. ¿Puedes crear una función que imprima los detalles del clima de una sola ciudad para reducir la repetición en la función `main()` y, luego, hacer lo mismo en las ciudades restantes?
2. ¿Puedes actualizar la función `main()` a fin de llamar a la función que creaste para cada ciudad y pasar los detalles apropiados del clima como argumentos?

```
fun main() {
    printWeatherDetails("Ankara", 27, 31, 82)
    printWeatherDetails("Tokyo", 32, 36, 10)
    printWeatherDetails("Cape Town", 59, 64, 2)
    printWeatherDetails("Guatemala City", 50, 55, 7)
}

fun printWeatherDetails(city: String, lowTemp: Int, highTemp: Int, chanceOfRain: Int) {
    println("City: $city")
    println("Low temperature: $lowTemp, High temperature: $highTemp")
    println("Chance of rain: ${chanceOfRain}%")
    println()
}
```

```
City: Ankara
Low temperature: 27, High temperature: 31
Chance of rain: 82%
```

```
City: Tokyo
Low temperature: 32, High temperature: 36
Chance of rain: 10%
```

```
City: Cape Town
Low temperature: 59, High temperature: 64
```


6. Instalación de Android Studio

En base a tu sistema operativo, realiza la instalación de android studio.

[Windows](#)

[macOS](#)

[Linux](#)

Documenta el proceso:

INSTALACION DEL ANDROID ESTUDIO EN UN SISTEMA WINDOWS

PASO 1

BUSCAR EN LA WEB

Buscar, en cambio, [DESCARGAR ANDROD ESTUDIO](#)



Android Developers

<https://developer.android.com> > [codelabs](#) > [basic-androi...](#)

Descarga e instala Android Studio

12 ene. 2024 — **Android Studio** es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de apps para Android creado y distribuido por Google. Un ...

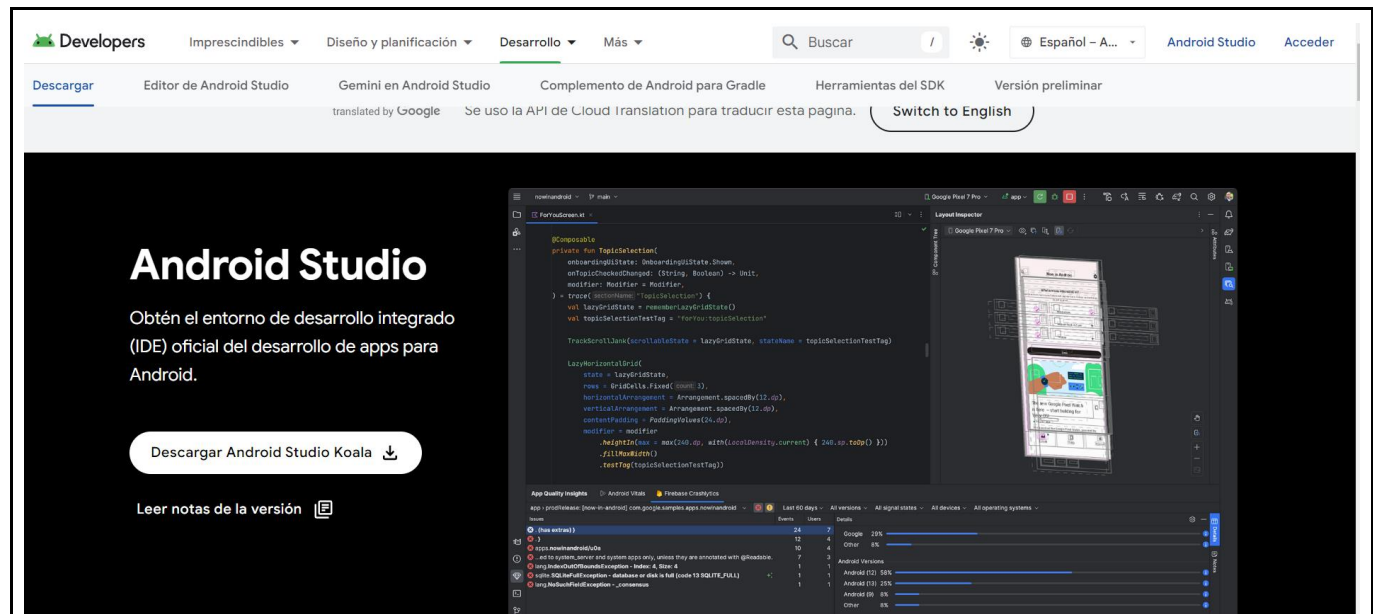
PASO 2

COMPLETAR EL TUTORIAL

The screenshot shows the 'Descarga e instala Android Studio' tutorial page. On the left, there is a list of steps: 1. Antes de comenzar, 2. Windows: Cómo verificar los requisitos del sistema, 3. Windows: Cómo descargar e instalar Android Studio, 4. macOS: Cómo verificar los requisitos del sistema, 5. macOS: Cómo descargar e instalar Android Studio, 6. Linux: Cómo verificar los requisitos del sistema, 7. Linux: Cómo descargar e instalar Android Studio, and 8. Conclusión. The 'Conclusión' step is currently selected. The main content area displays the title '8. Conclusión' followed by a congratulatory message: '¡Felicitaciones! Instalaste correctamente Android Studio. ¡Ya puedes avanzar al próximo paso!'. Below this, it says 'Si tuviste algún problema técnico durante los pasos de instalación, consulta la [guía de solución de problemas](#).' There is a 'Resumen' section with two bullet points: 'Un entorno de desarrollo integrado (IDE) es una colección de herramientas para desarrollar software.' and 'Android Studio es el IDE basado en IntelliJ IDEA que se usa en el desarrollo de Android.' A 'Más información' section follows with three links: 'Descarga Android Studio', 'Instala Android Studio', and 'Introducción a Android Studio'. At the bottom left, there is an 'Atrás' button.

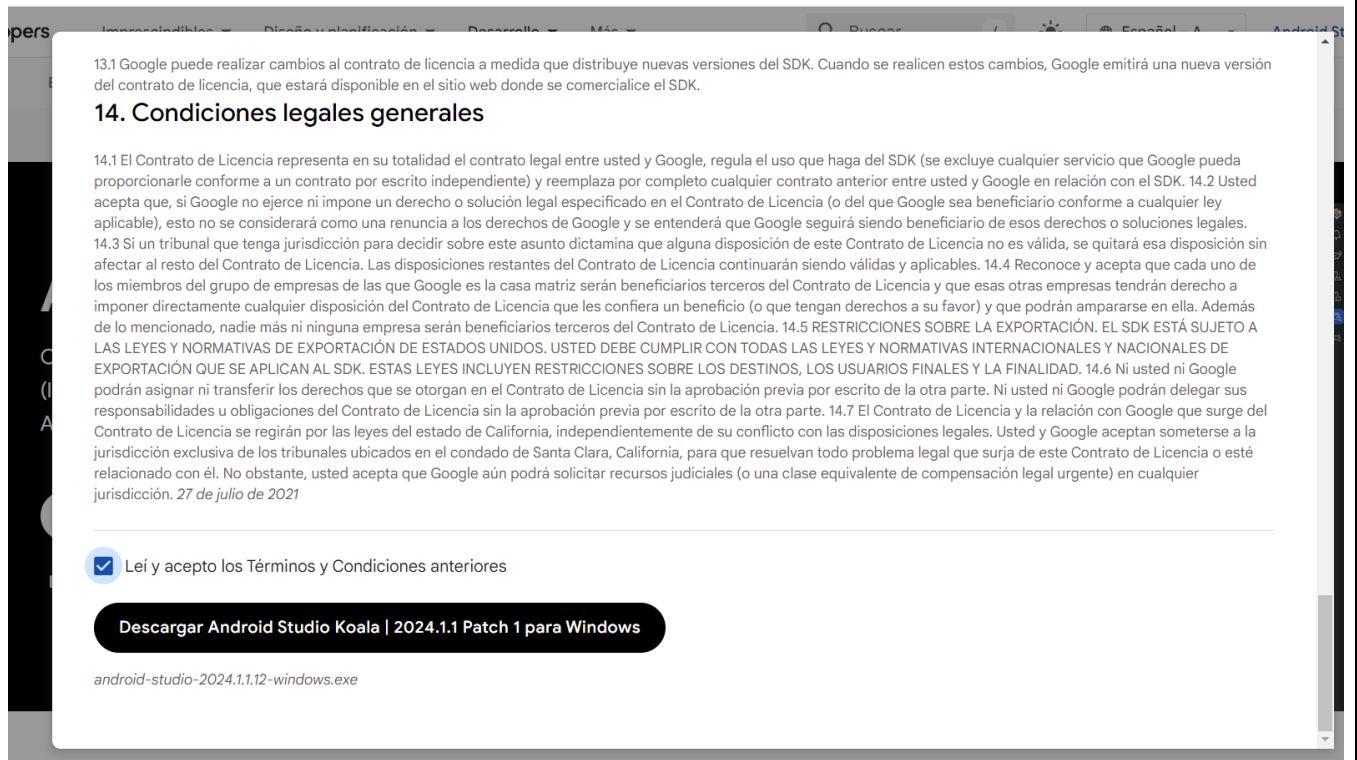
PASO 3

DESCARGAR



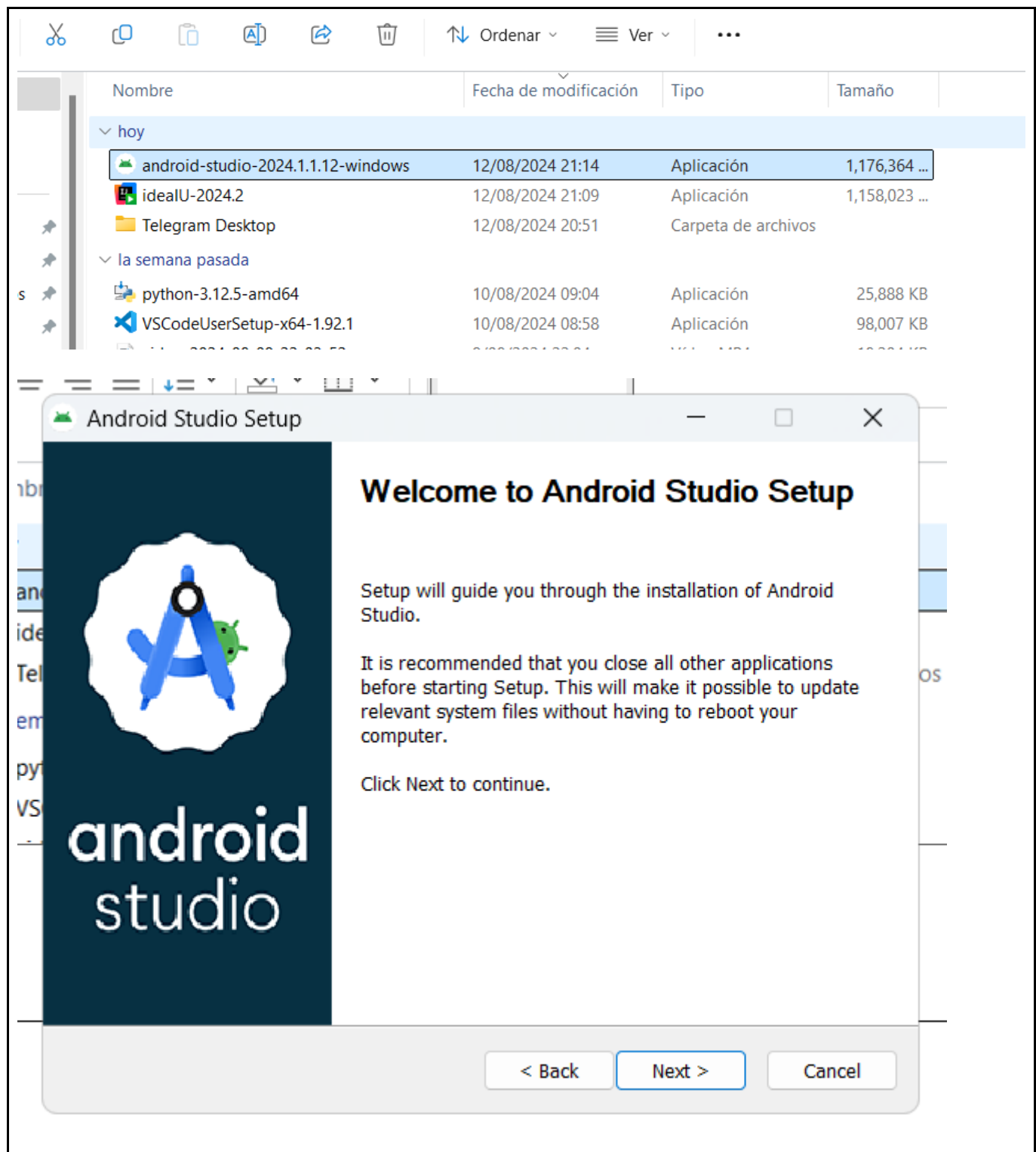
PASO 4

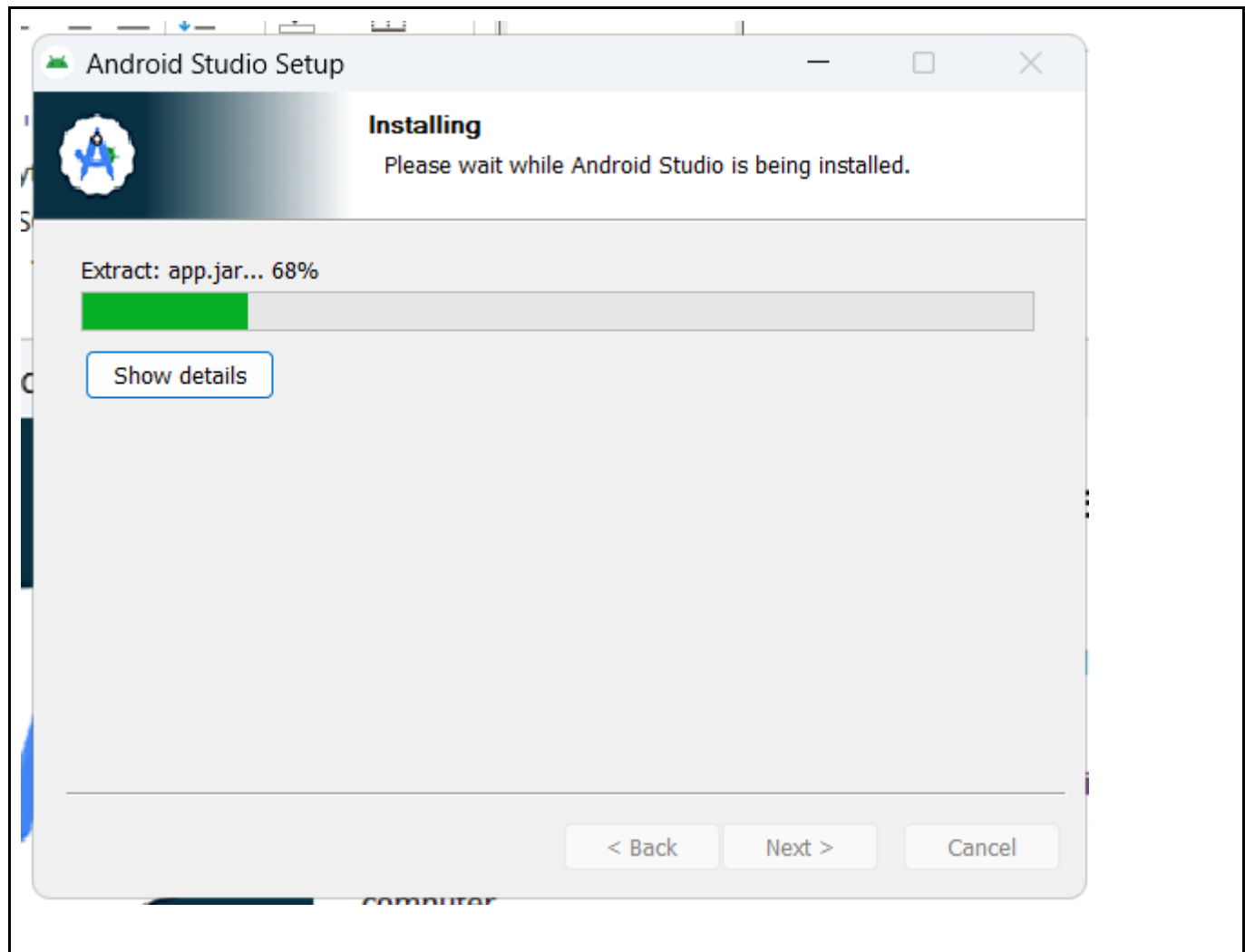
ACEPTAR LAS CONDICIONES OBLIGATORIAS

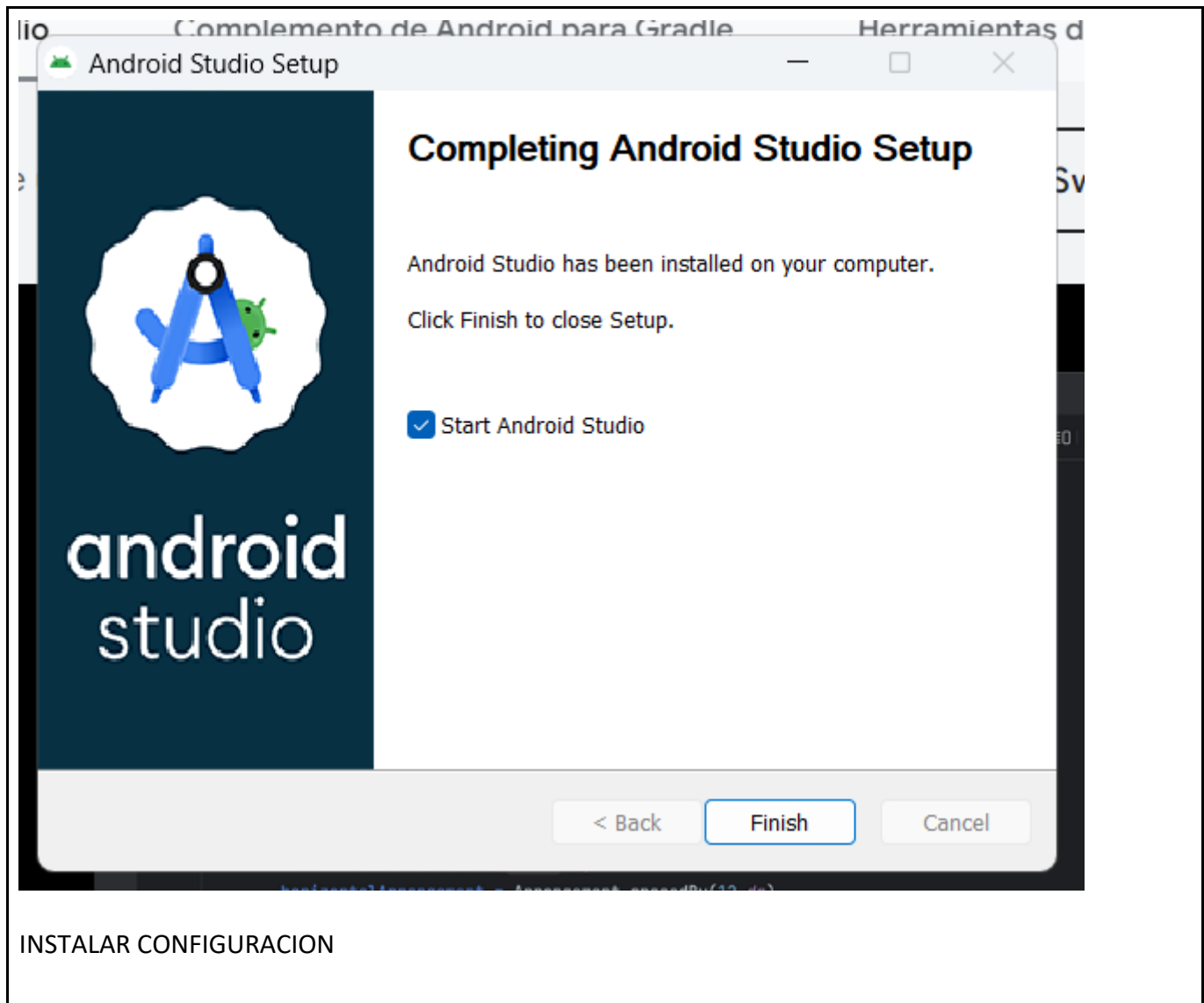


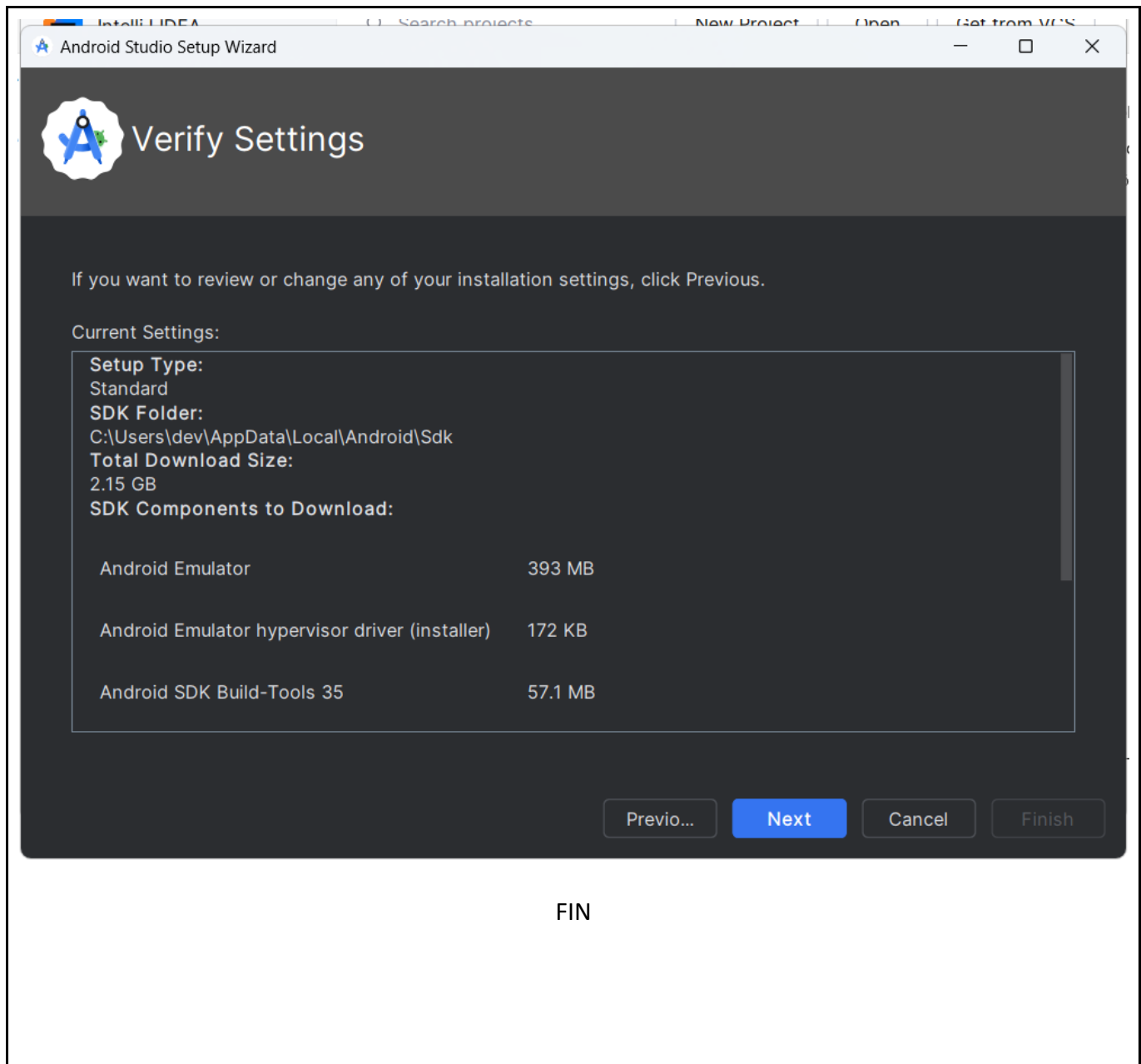
PASO 5

INSTALAR EL PAQUETE DESCARGADO









OBSERVACIONES (5 mínimo):

(Las observaciones son las notas aclaratorias, objeciones y problemas que se pudo presentar en el desarrollo del laboratorio)

- La instalación de Android Studio varía según el sistema operativo.
- Se requiere mucho espacio en disco para Android Studio.

Programación en Móviles

- Es necesaria una conexión a internet estable durante la instalación.
- Los equipos deben cumplir los requisitos mínimos para Android Studio.
- Se deben configurar las opciones del entorno de desarrollo tras la instalación.

CONCLUSIONES (5 mínimo):

(Las conclusiones son una opinión personal sobre tu trabajo, explicar cómo resolviste las dudas o problemas presentados en el laboratorio. Además de aportar una opinión crítica de lo realizado)

- Un tutorial estructurado facilita la instalación de Android Studio.
- Los ejercicios permiten aprender los conceptos básicos de Kotlin.
- Crear funciones para el código repetitivo mejora la eficiencia.
- La corrección de errores de compilación es esencial para la calidad del código.
- Los ejercicios muestran la versatilidad de Kotlin en diferentes escenarios.