MEDAC
Instituto Oficial de Formación Profesional

Tarea de Evaluación 3t

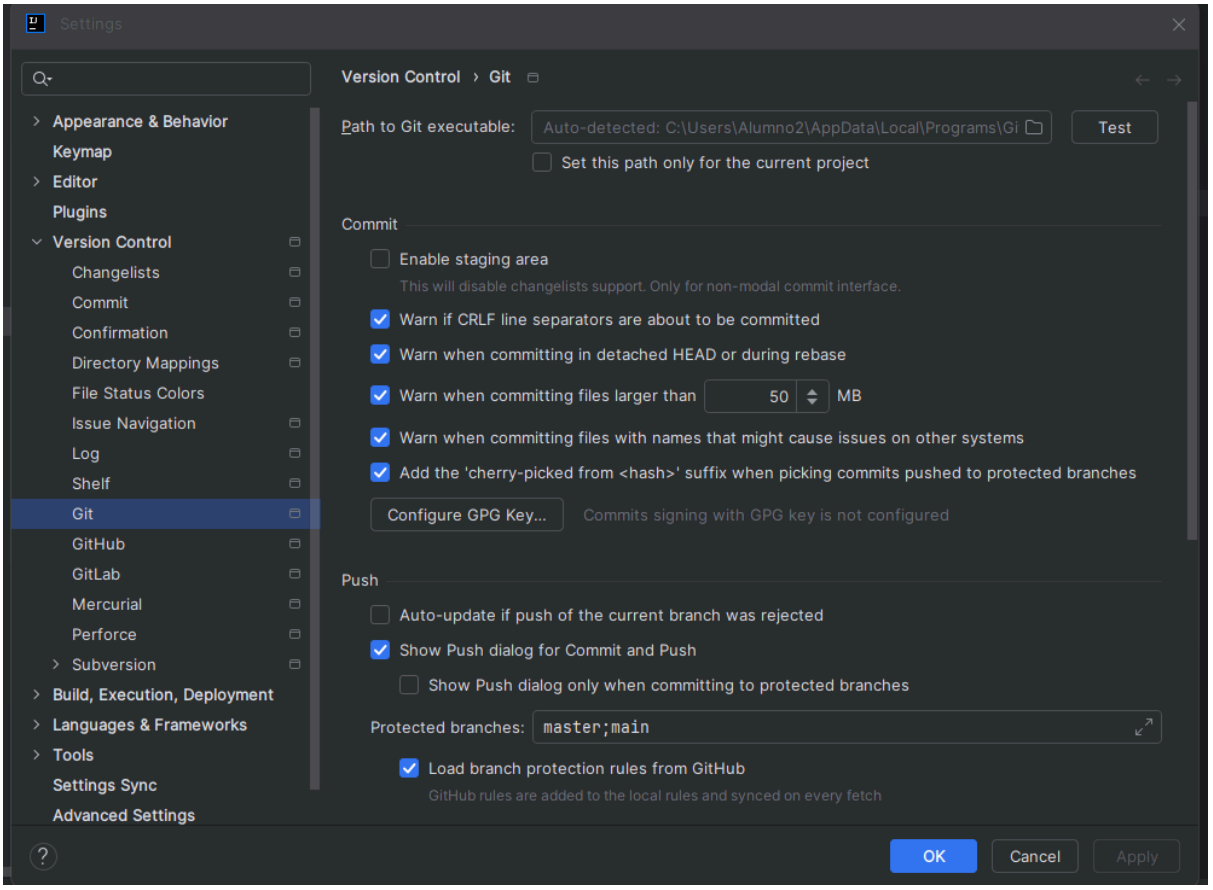
Alejandro Garcia-Mauriño Salas . Antonio Moro del Toro

1. Configurar Git y GitHub en IntelliJ IDEA

Git te ayuda a llevar un historial de cambios en tu código, y GitHub permite compartir y colaborar en proyectos.

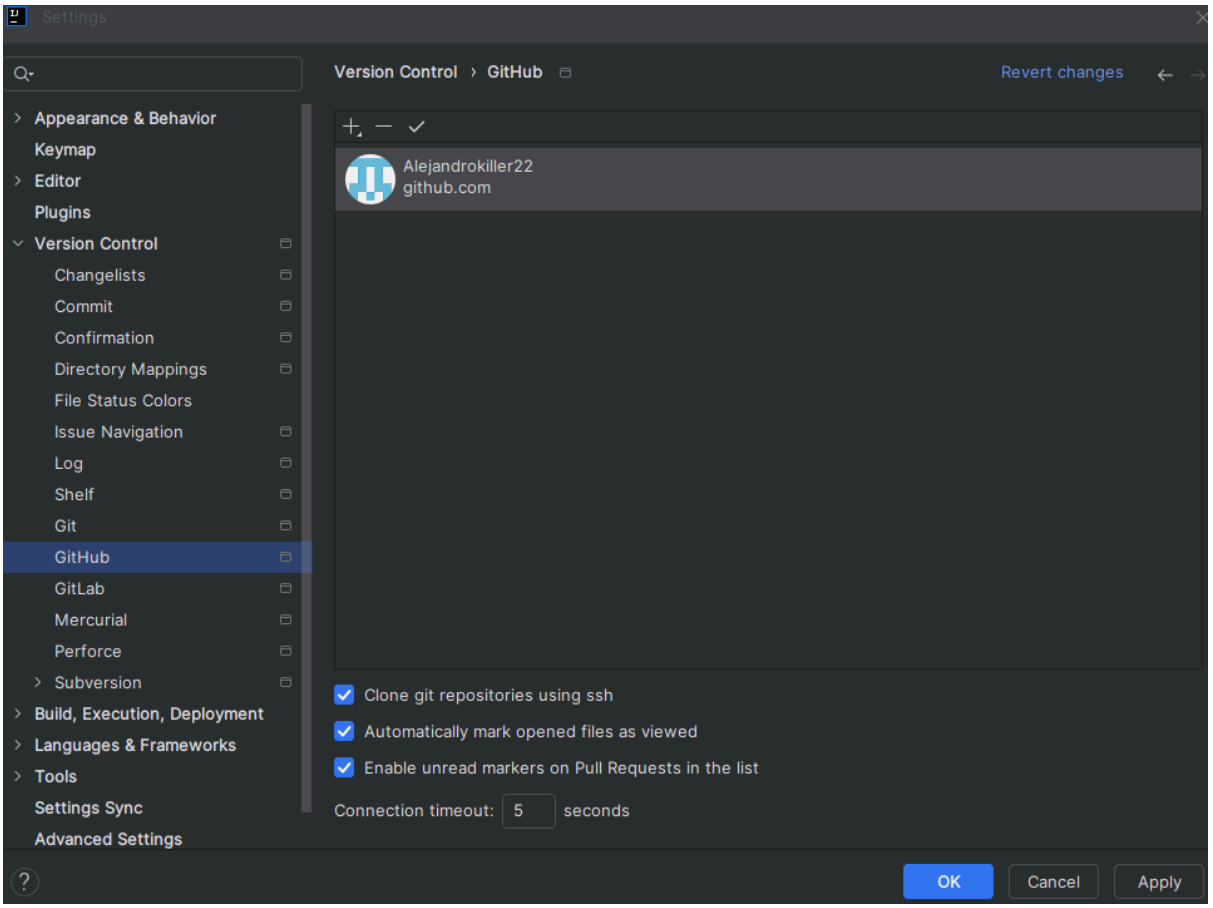
Activar Git en tu proyecto

Abre tu proyecto en IntelliJ IDEA. Ve a **VCS > Enable Version Control Integration...** Selecciona **Git** y presiona **OK**. Se creará la carpeta `.git`, lo que indica que tu proyecto está bajo control de versiones.



Conectar GitHub a IntelliJ

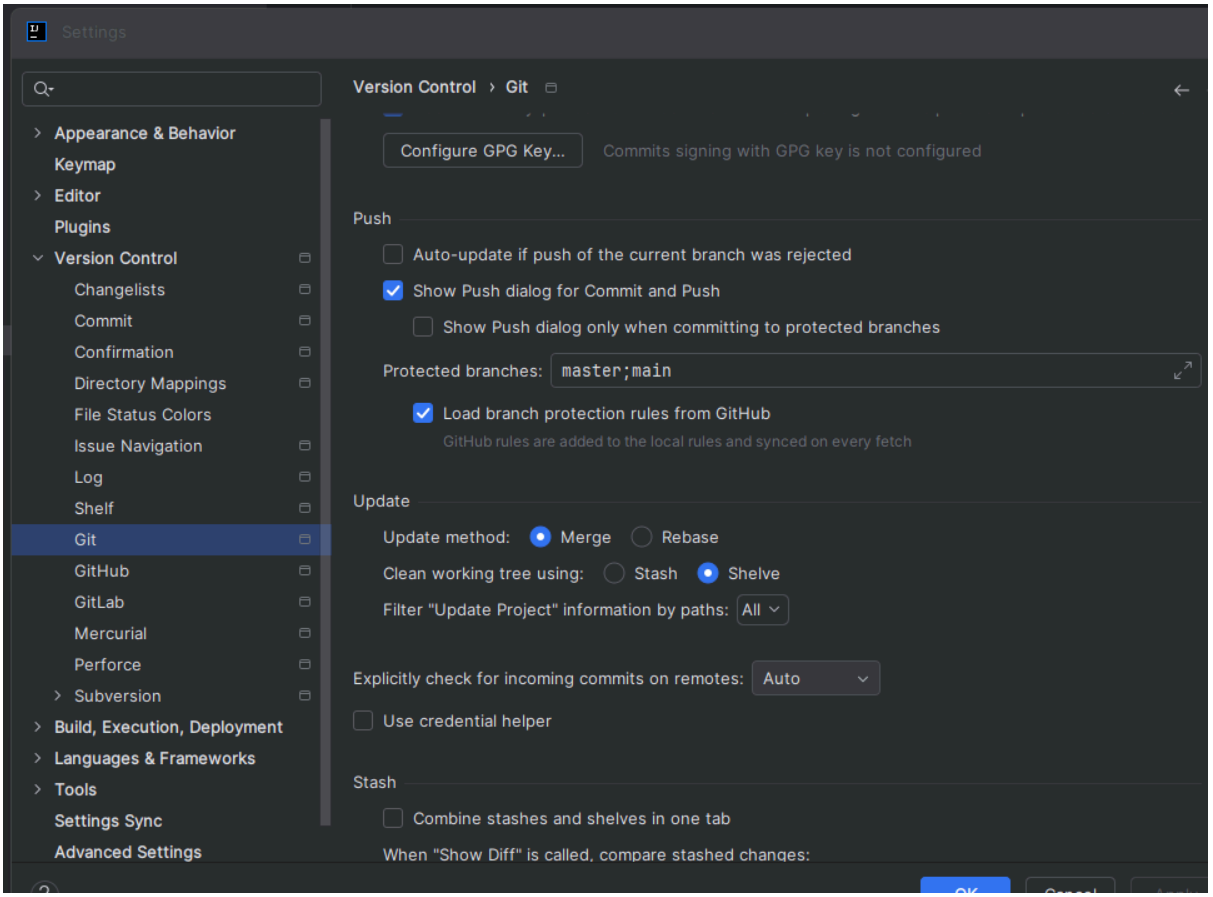
Ve a **File > Settings > Version Control > GitHub**. Presiona **+ Add Account** y selecciona **Log in via GitHub**. **Inicia sesión con tu cuenta** y autoriza IntelliJ.



Subir tu código a GitHub

Ve a **VCS > Git > Push**. Si no tienes un repositorio remoto, ve a **GitHub > New Repository** y crea uno. Copia la URL y usa el comando en la terminal de IntelliJ:

git remote add origin https://github.com/TuUsuario/NombreRepositorio.git



Usa `git push origin main` para subir tus archivos a GitHub.

El proyecto ahora está en GitHub y bajo control de versiones.

Commits frecuentes y descriptivos

Revisar los cambios

- Asegúrate de que las modificaciones en el archivo son correctas y necesarias para el proyecto.
- En este caso, se han agregado configuraciones relacionadas con **JUnit** (bibliotecas y documentación).

2️⃣ Añadir un mensaje descriptivo para el commit

En el cuadro de texto, escribe un mensaje claro sobre qué estás cambiando, por ejemplo:
Configuración de bibliotecas JUnit en IntelliJ IDEA

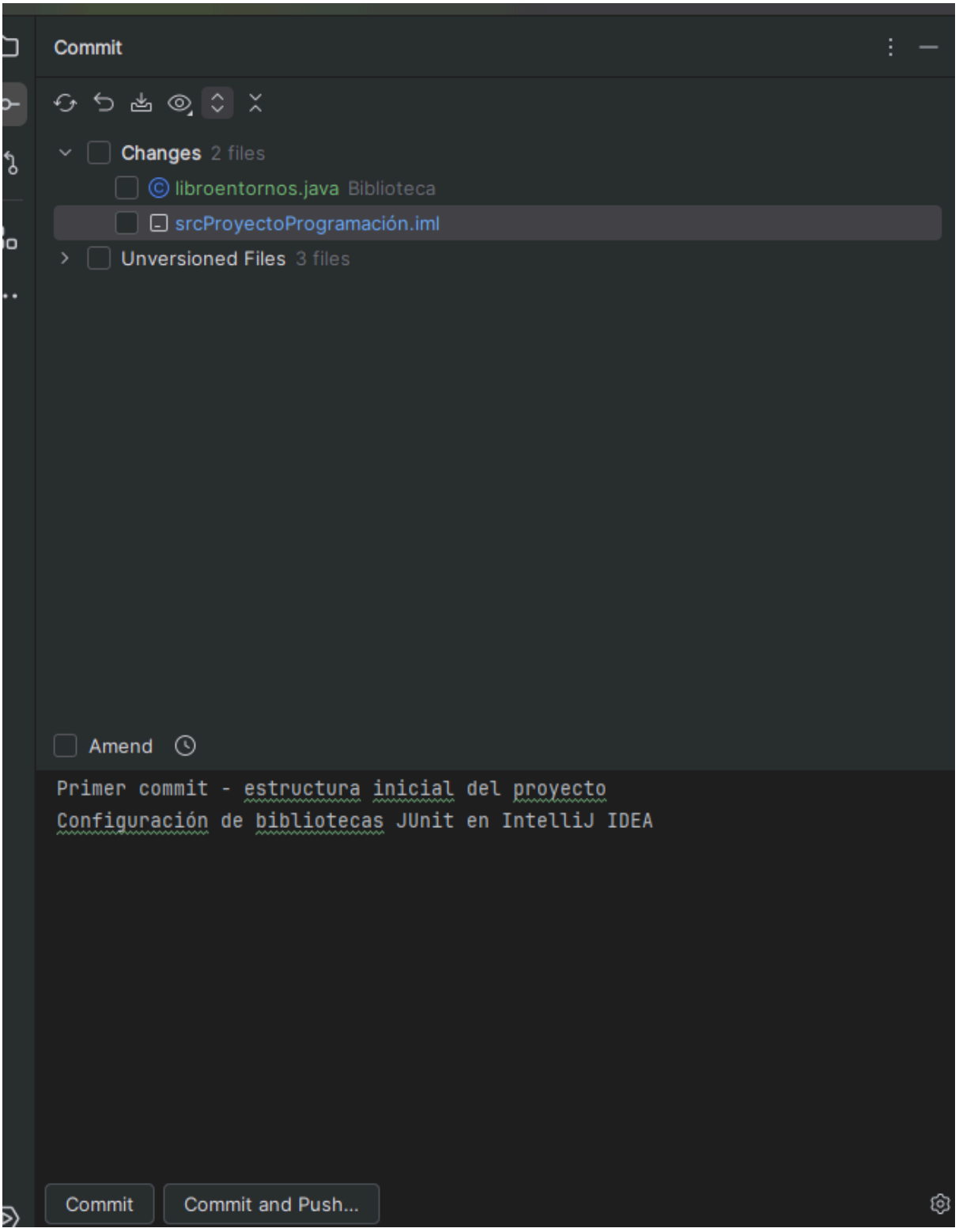
- Un mensaje claro ayuda a entender el propósito del cambio en el historial de versiones.

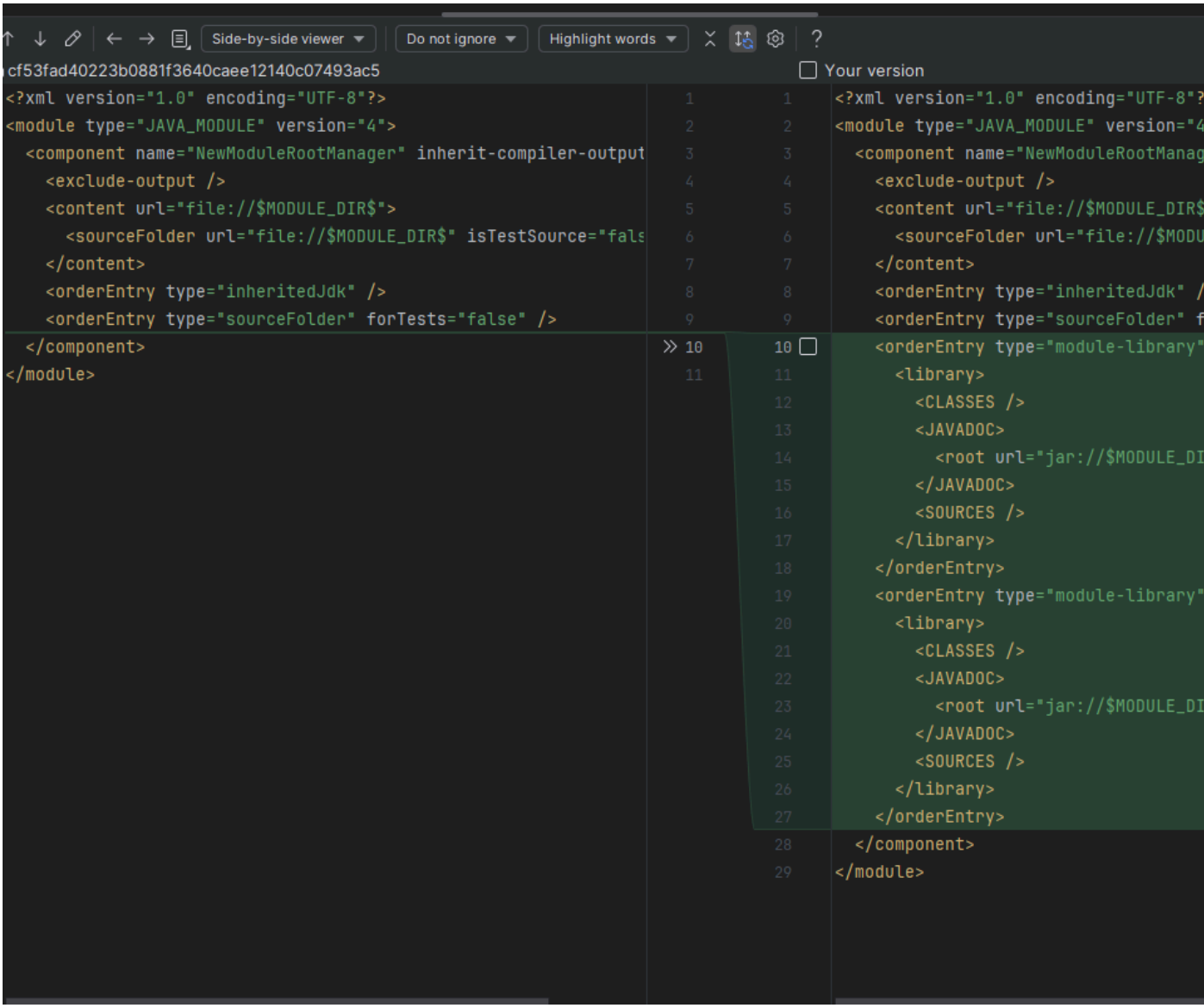
3️⃣ **Realizar el commit**

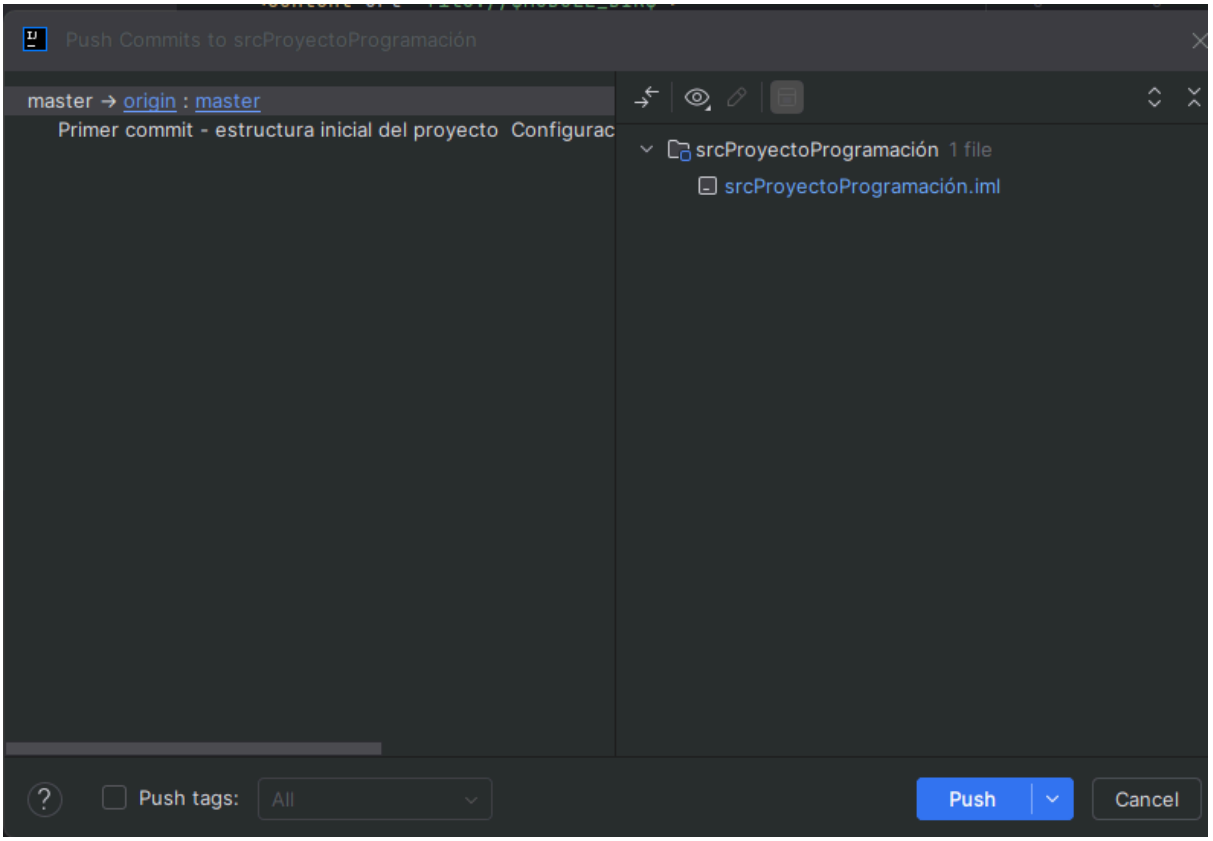
- Haz clic en **Commit** para guardar los cambios localmente.
- Si deseas subir los cambios a GitHub, haz **Commit and Push**.

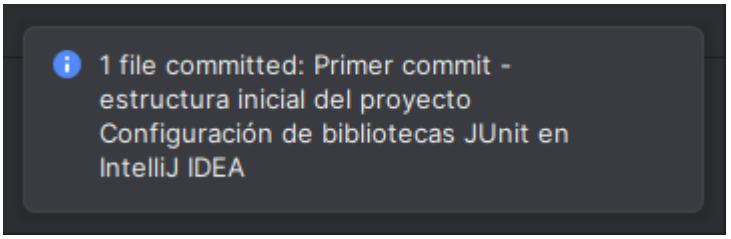
4️⃣ **Verificar en GitHub**

- Abre tu repositorio en GitHub y revisa si los cambios aparecen correctamente después del **push**.


Commit
Changes 2 files
libroentornos.java Biblioteca
srcProyectoProgramación.iml
Unversioned Files 3 files
Amend
Primer commit - estructura inicial del proyecto
Configuración de bibliotecas JUnit en IntelliJ IDEA
Commit
Commit and Push...


Side-by-side viewer
Do not ignore
Highlight words
cf53fad40223b0881f3640caee12140c07493ac5
Your version
<?xml version="1.0" encoding="UTF-8"?>
<module type="JAVA_MODULE" version="4">
<component name="NewModuleRootManager" inherit-compiler-output
<exclude-output />
<content url="file://$MODULE_DIR$">
<sourceFolder url="file://$MODULE_DIR$" isTestSource="fals
</content>
<orderEntry type="inheritedJdk" />
<orderEntry type="sourceFolder" forTests="false" />
</component>
</module>
<orderEntry type="module-library"
<library>
<CLASSES />
<JAVADOC>
<root url="jar://$MODULE_DI
</JAVADOC>
<SOURCES />
</library>
</orderEntry>





Configurar JUnit manualmente en IntelliJ IDEA

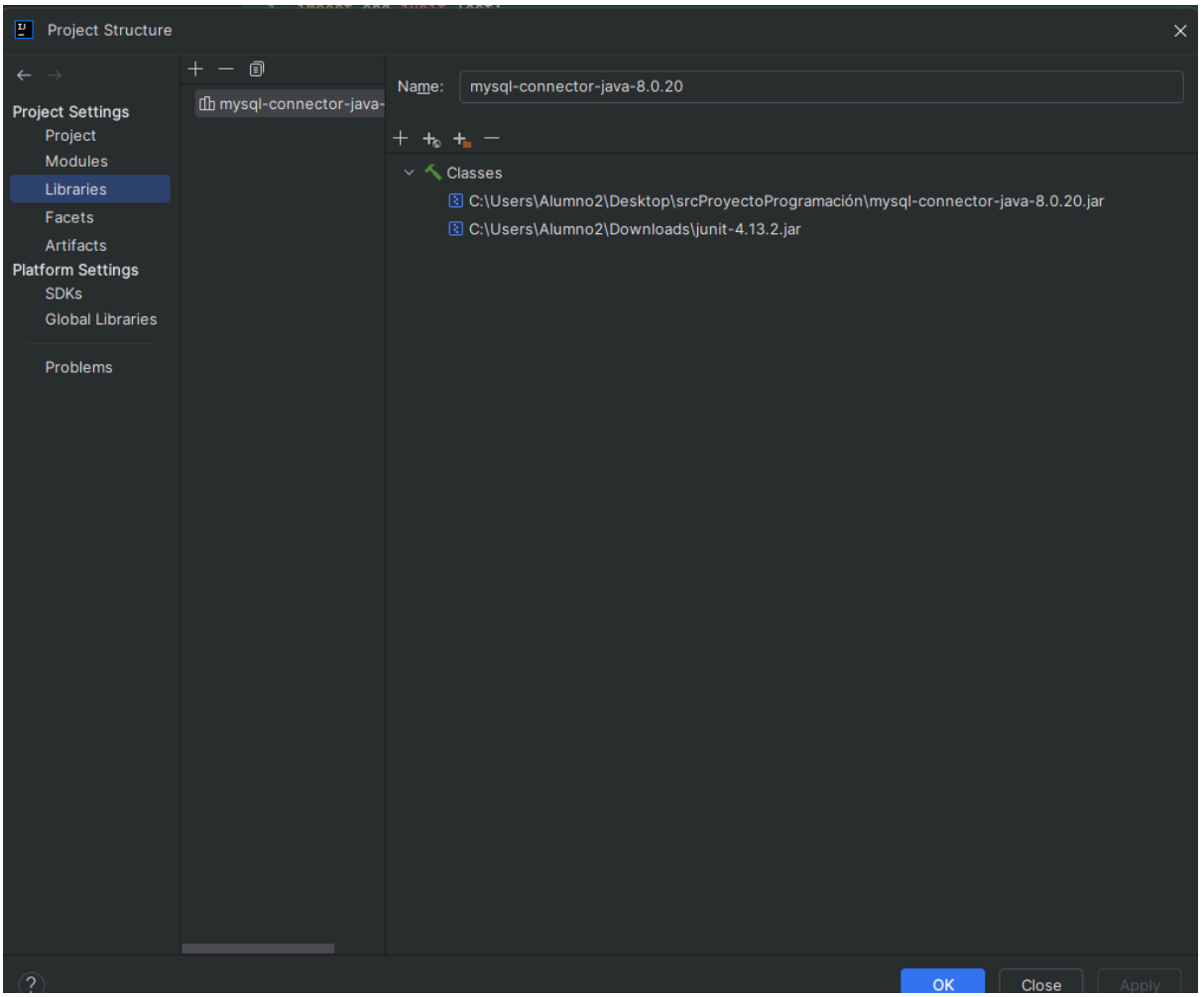
1️⃣ Descargar JUnit

- Ve a la página oficial de **JUnit**: https://junit.org/junit4/
- Descarga el archivo `junit-4.13.2.jar` (versión más reciente).

2️⃣ Añadir JUnit al proyecto

- Copia `junit-4.13.2.jar` en la carpeta `lib` dentro de tu proyecto.
- En IntelliJ, ve a **File > Project Structure > Libraries**.

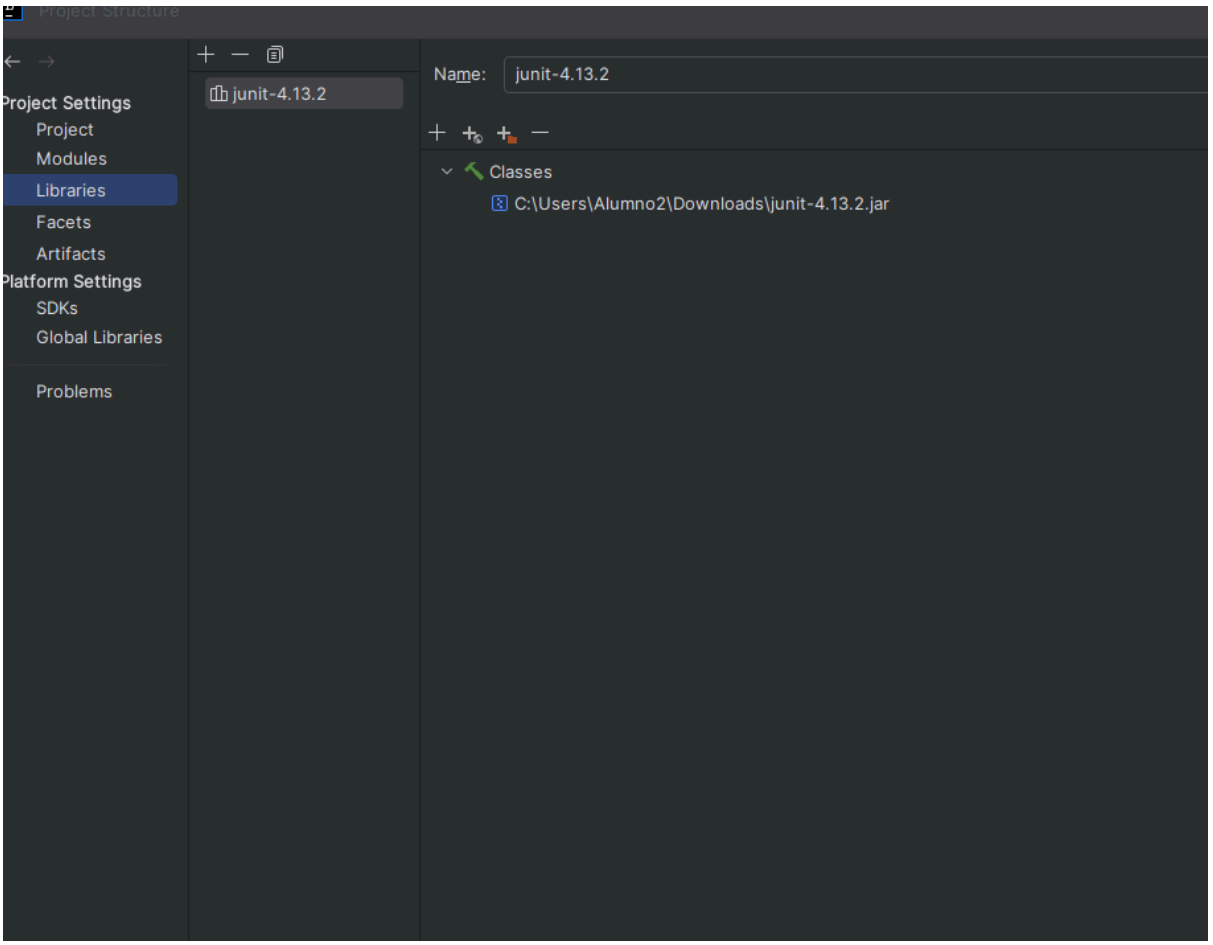
- Presiona **+ Add Library** y selecciona *Add JARs or Directories*.
- Busca el archivo `junit-4.13.2.jar` dentro de `lib`, selecciona **Apply and Close**.



Crear una clase de prueba en IntelliJ

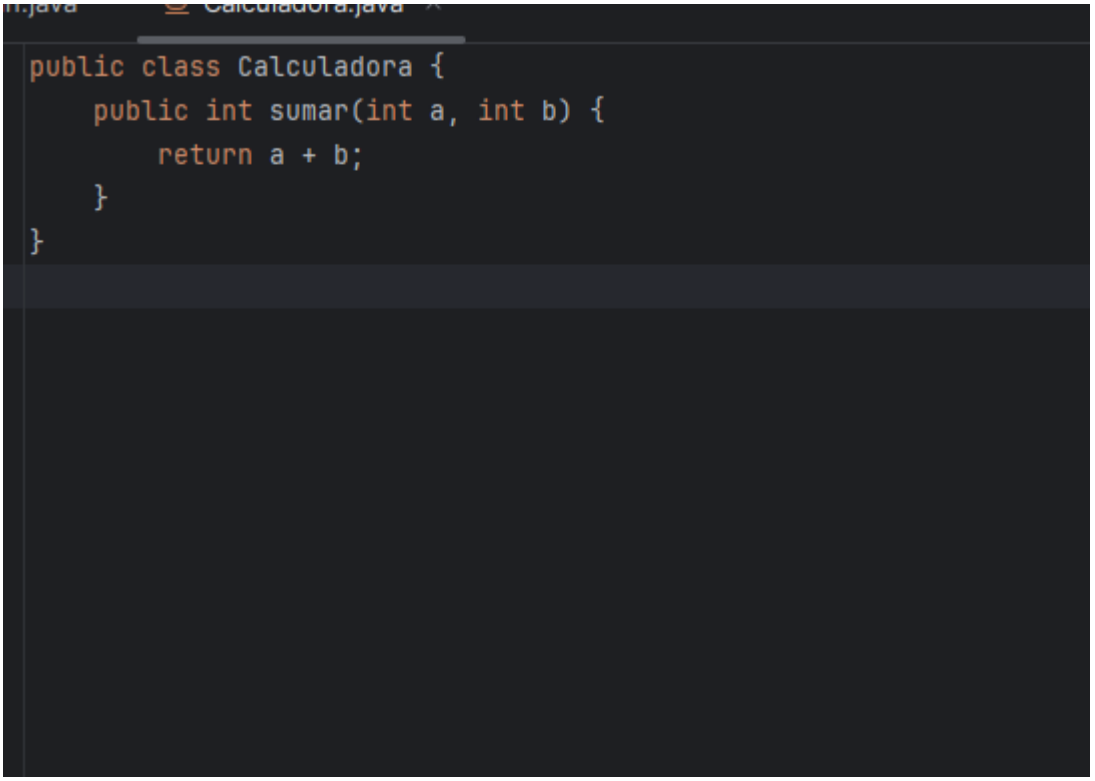
Abre IntelliJ IDEA. Ve a **File > New > Project**. Selecciona **Java** y presiona **Next**. Escribe un nombre para el proyecto (Ejemplo: `PruebasJUnit`). Presiona **Finish**.

- **Agregar JUnit al nuevo proyecto**



Ve a **File > Project Structure > Libraries**. Haz clic en **+ Add Library** > *Add JARs or Directories*. Busca y selecciona `junit-4.13.2.jar` (si no lo tienes, descárgalo aquí). Presiona **Apply and Close**.

Crear una clase de prueba fácil (`Calculadora.java`)



```java
public class Calculadora {
    public int sumar(int a, int b) {
        return a + b;
    }
}
```

Crear la prueba unitaria (`CalculadoraTest.java`)

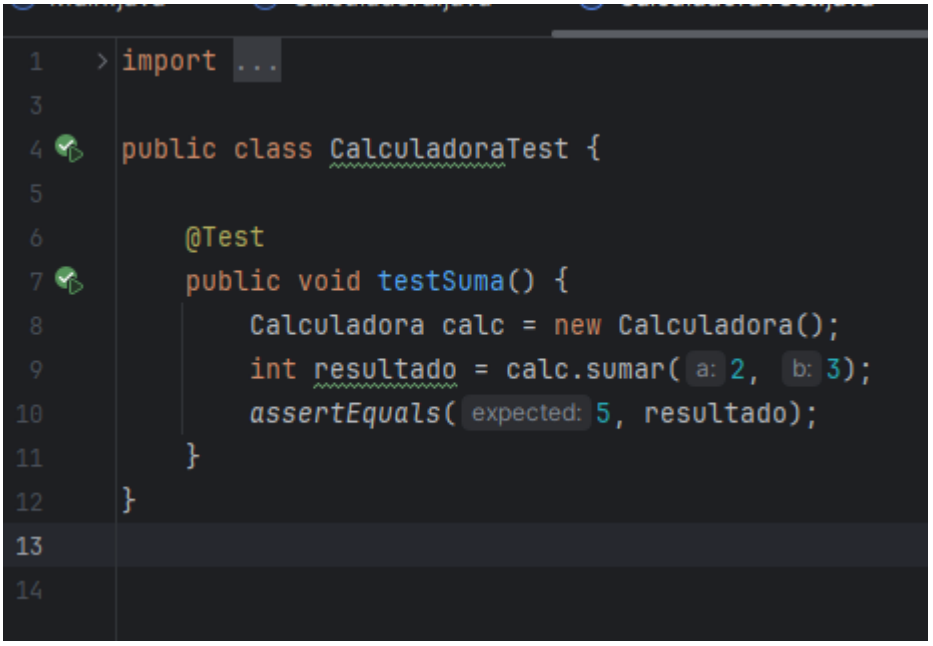
```java
import static org.junit.Assert.*;
import org.junit.Test;

public class CalculadoraTest {

    @Test
    public void testSuma() {
        Calculadora calc = new Calculadora();
        int resultado = calc.sumar(2, 3);
        assertEquals(5, resultado);
    }
}
```

Ejecutar la prueba en IntelliJ

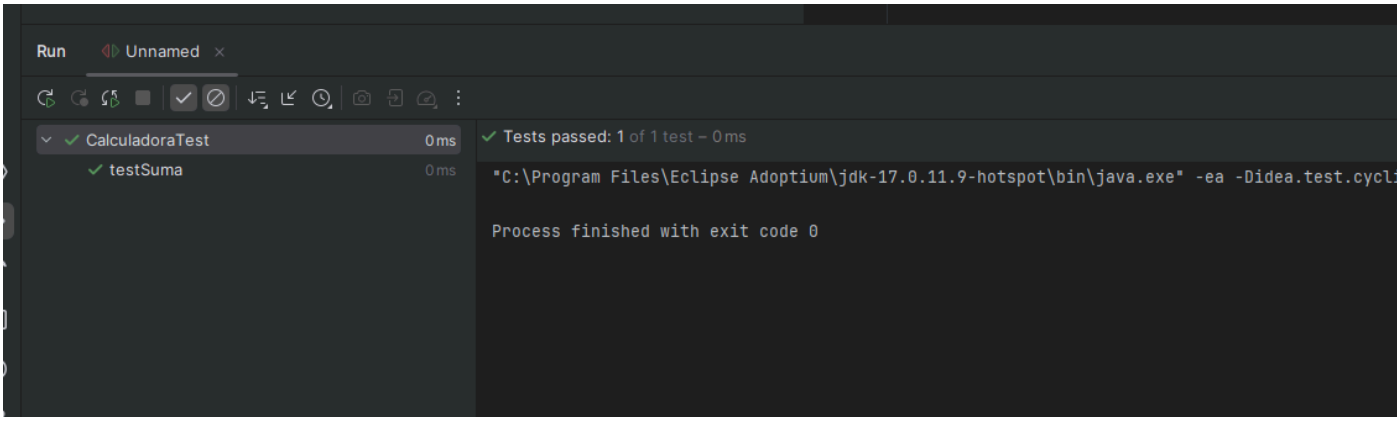
Haz clic derecho en `CalculadoraTest.java`. **Selecciona "Run 'CalculadoraTest'". Si todo está bien, verás verde indicando que la prueba pasó.**



```java
import ...

public class CalculadoraTest {

    @Test
    public void testSuma() {
        Calculadora calc = new Calculadora();
        int resultado = calc.sumar( a: 2,  b: 3);
        assertEquals( expected: 5, resultado);
    }
}
```



Prueba 1: Validar que la etiqueta `<img>` tenga el atributo `alt` correctamente

Main.java | Calculadora.java | CalculadoraTest.java | ImagenTest.java

```java
import static org.junit.Assert.*;
import org.junit.Test;

public class ImagenTest {
    @Test
    public void testImagenTieneAlt() {
        String imagenHTML = "<img src='logo.png' alt='Logo de la empresa'>";
        assertTrue(imagenHTML.contains("alt=")); // Verifica que la imagen tiene el atributo alt
    }
}

```

Prueba 2: Verificar que la etiqueta <meta> contenga el atributo charset="UTF-8"

Depuración y Refactorización

Durante el desarrollo del proyecto, se han utilizado las herramientas de depuración integradas en Eclipse e intelij para localizar y solucionar errores, especialmente en la conexión con la base de datos y la funcionalidad de inserción/búsqueda de libros.

Se han aplicado técnicas de refactorización para mejorar el código:

- **Extracción de métodos** para dividir funciones largas en partes más legibles.

- **Renombrado de variables y métodos** para mejorar la claridad del código (`btnGuardar`, `txtTitulo`, `insertarLibro()`...).

- **Eliminación de código duplicado** y limpieza de imports innecesarios.

- Mejora de la estructura de clases DAO para que cada clase tenga una única responsabilidad.

Calidad de Código

Durante el desarrollo de este proyecto, se ha aplicado una codificación limpia y estructurada siguiendo buenas prácticas de Java:

- Nombres significativos: Se han utilizado nombres descriptivos para clases como `LibroDAO`, `Ventana_Insertar`, `ConexionMySQL`, lo que facilita entender su propósito.

- Estructura organizada:

    - Interfaz gráfica con `Swing`: `JFrame`, `JPanel`, `JLabel`, `JTextField`, `JButton`, `JCheckBox`.

    - Acceso a base de datos mediante clases DAO.

    - Separación clara entre interfaz, lógica y acceso a datos.

- Estilo Java respetado:

    - Uso de `camelCase` para métodos y variables.

    - Clases con mayúscula inicial.

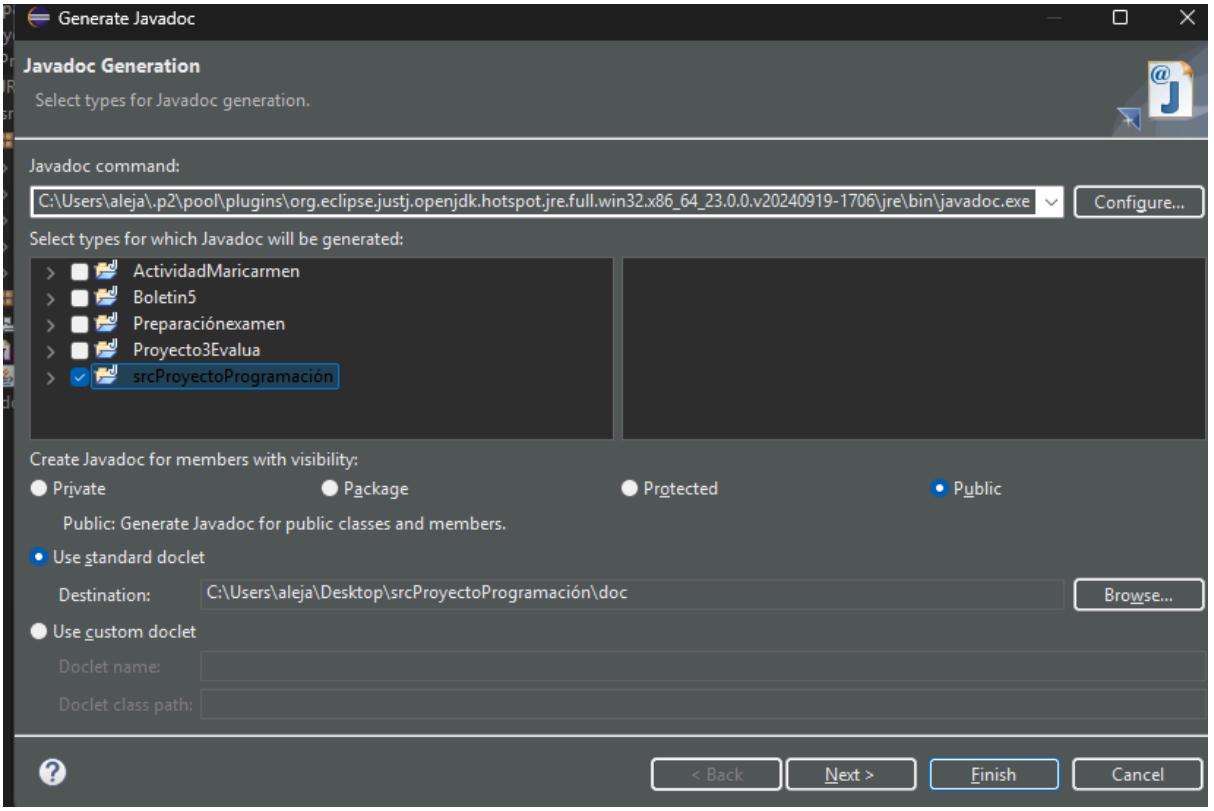
- Indentación consistente.

- Código legible:
  - Comentarios explicativos donde es necesario.
  - Métodos cortos, que realizan una sola función.
  - Eliminación de código innecesario o sin uso.

Generar JavaDoc en HTML

En Eclipse:

- Clic en `Project` > `Generate Javadoc...`

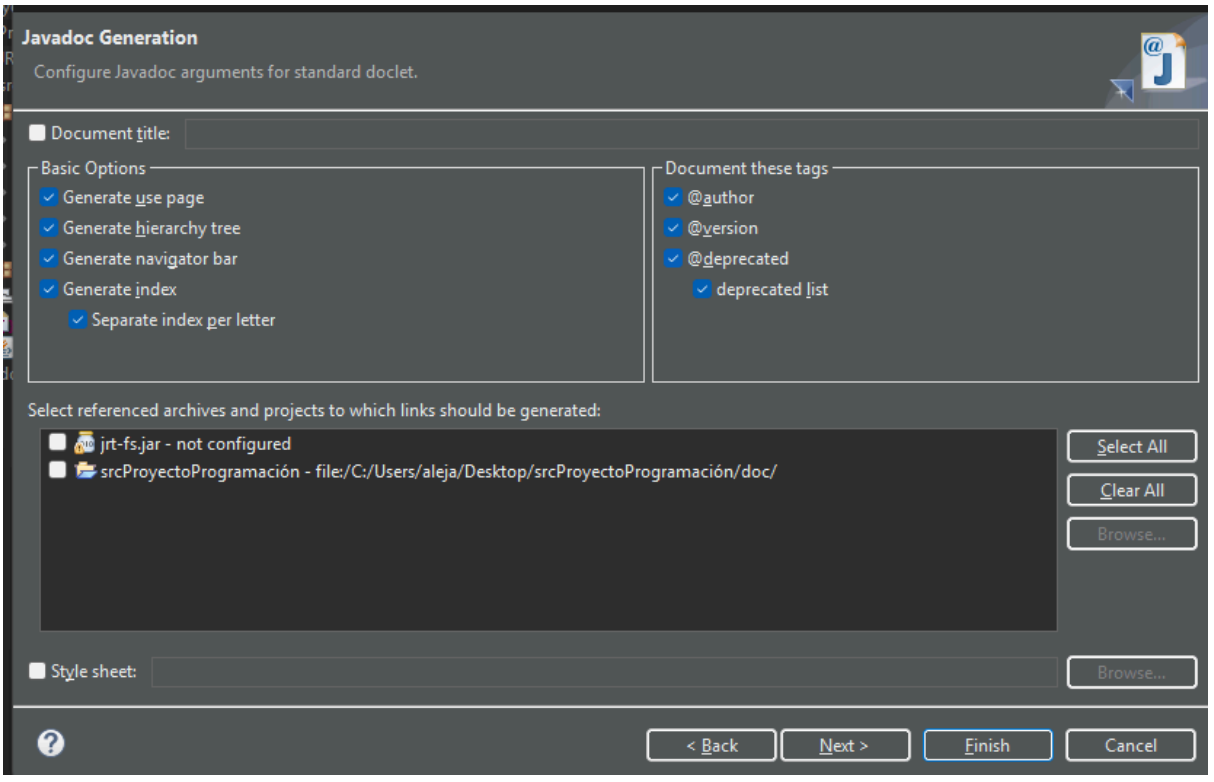


2. Selecciona las clases:

- Marca todas las que quieras documentar (por ejemplo `ConexionMySQL`, `LibroDAO`, etc.).

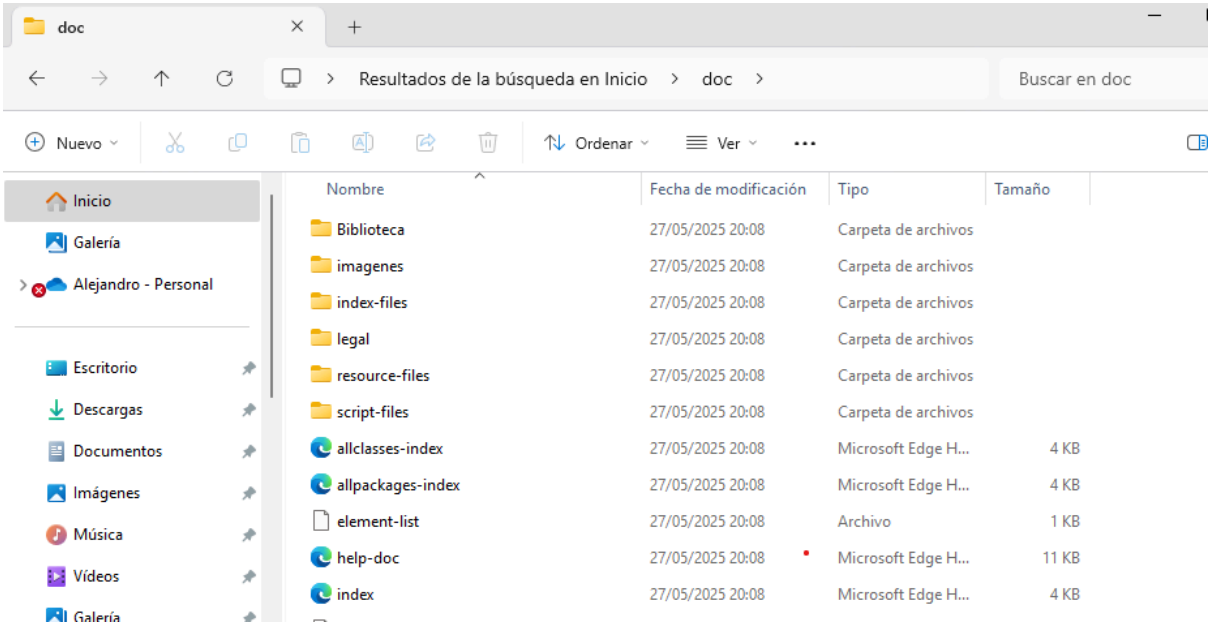
3. Configura la salida:

- Selecciona un directorio donde guardar (por ejemplo, dentro del proyecto una carpeta llamada `/doc`).



4. Dale a Finish.

5. Se generará una carpeta con archivos `.html`.


doc
Resultados de la búsqueda en Inicio
doc
Buscar en doc
Nuevo
Ordenar
Ver
Inicio
Galería
Alejandro - Personal
Escritorio
Descargas
Documentos
Imágenes
Música
Vídeos
Galería
Nombre
Fecha de modificación
Tipo
Tamaño
Biblioteca 27/05/2025 20:08 Carpeta de archivos
imagenes 27/05/2025 20:08 Carpeta de archivos
index-files 27/05/2025 20:08 Carpeta de archivos
legal 27/05/2025 20:08 Carpeta de archivos
resource-files 27/05/2025 20:08 Carpeta de archivos
script-files 27/05/2025 20:08 Carpeta de archivos
allclasses-index 27/05/2025 20:08 Microsoft Edge H... 4 KB
allpackages-index 27/05/2025 20:08 Microsoft Edge H... 4 KB
element-list 27/05/2025 20:08 Archivo 1 KB
help-doc 27/05/2025 20:08 Microsoft Edge H... 11 KB
index 27/05/2025 20:08 Microsoft Edge H... 4 KB