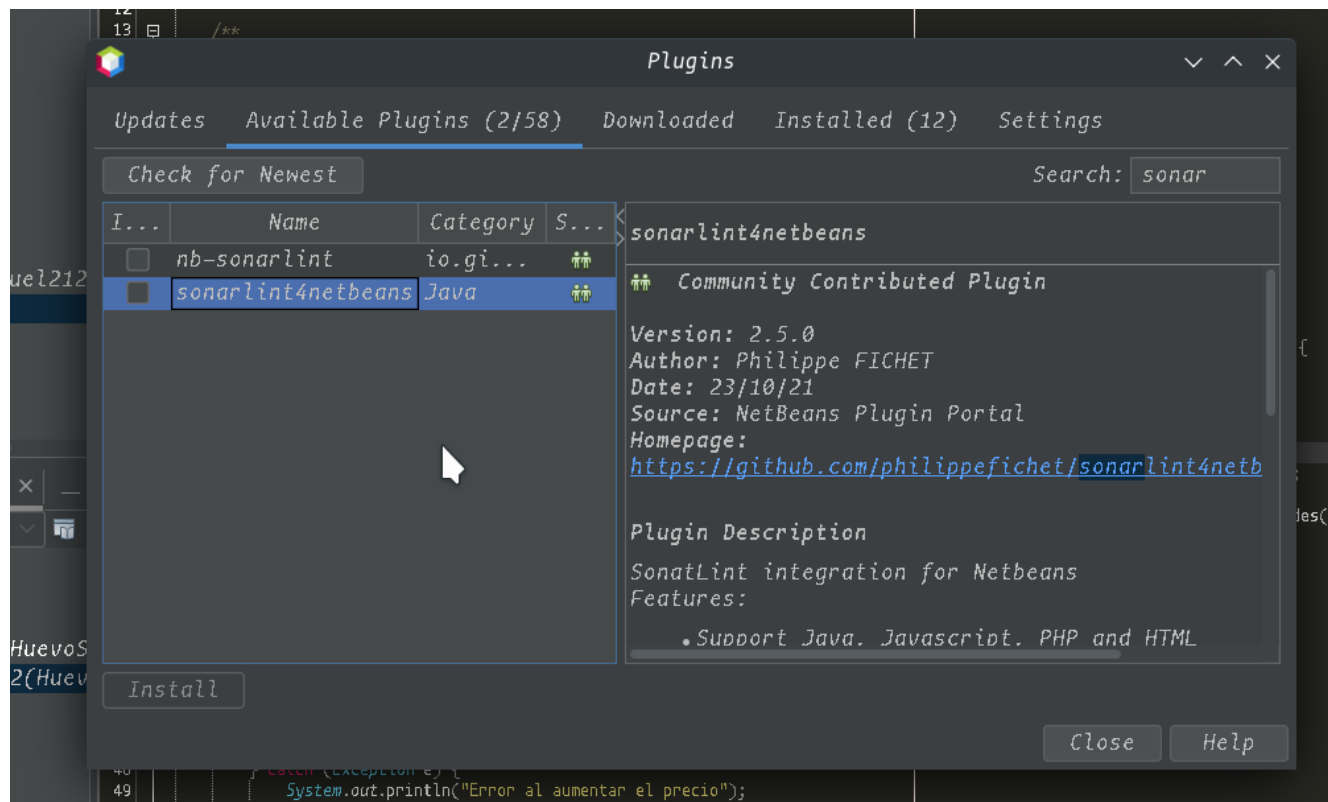
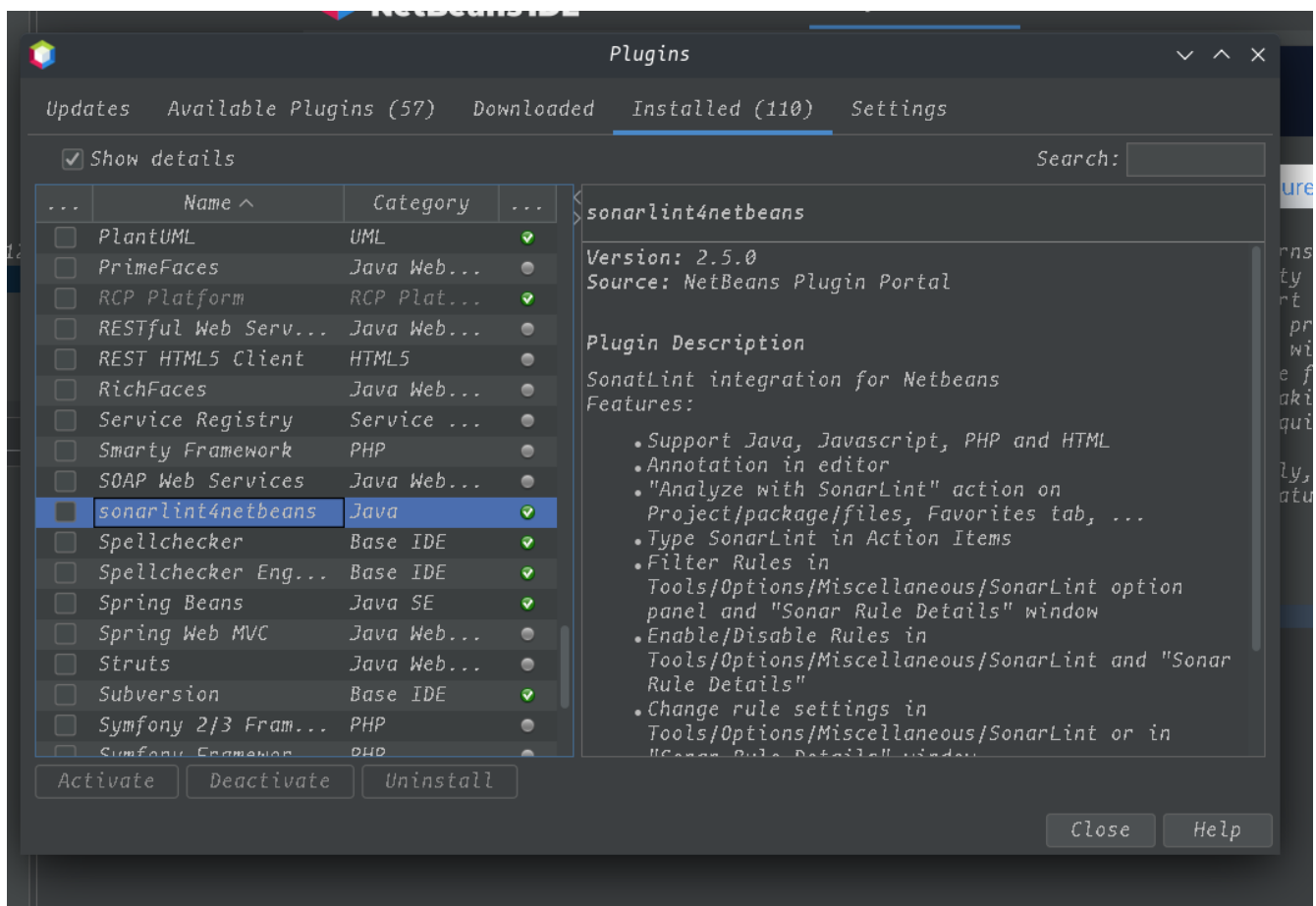


## Realizar un análisis de código.

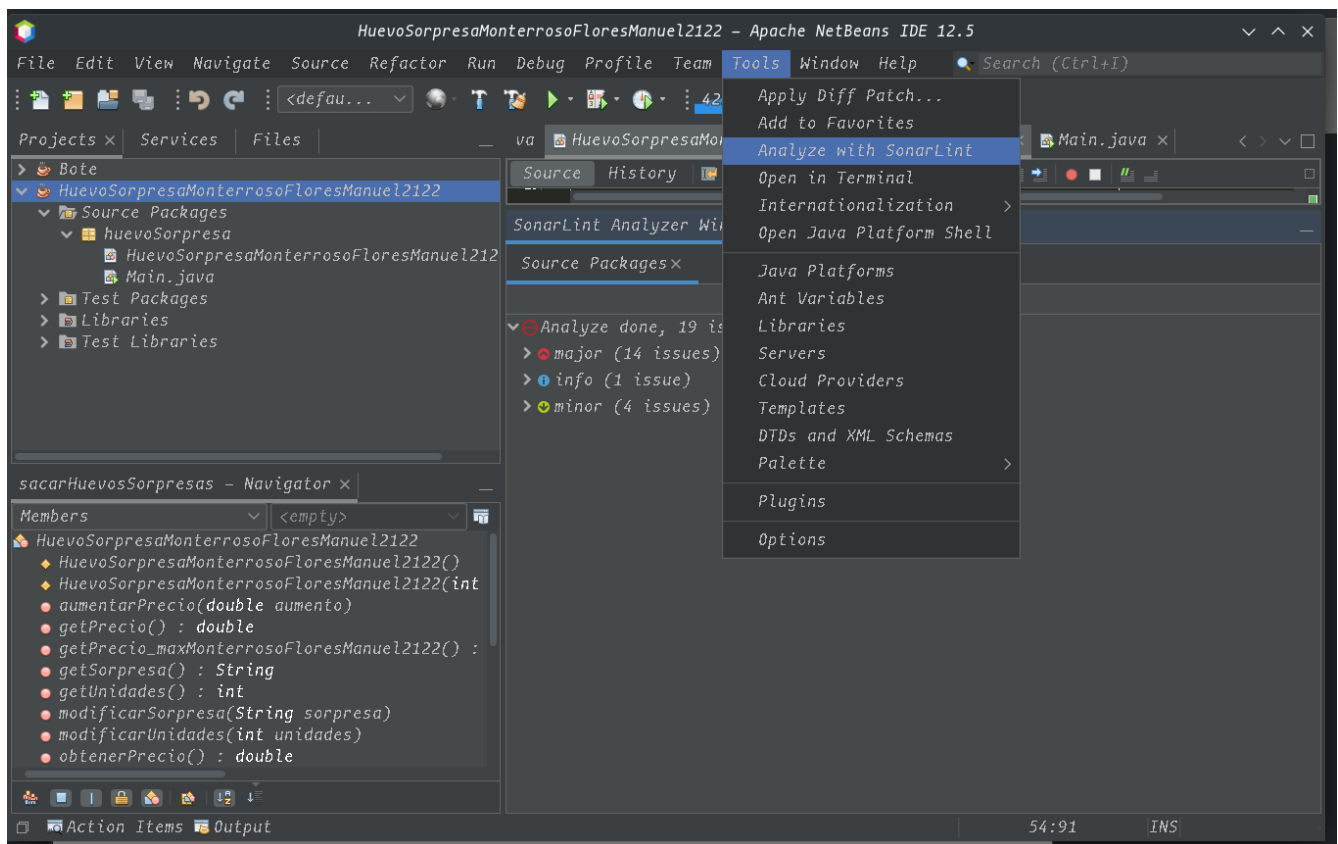
Instalación del Plugin SonarLint.



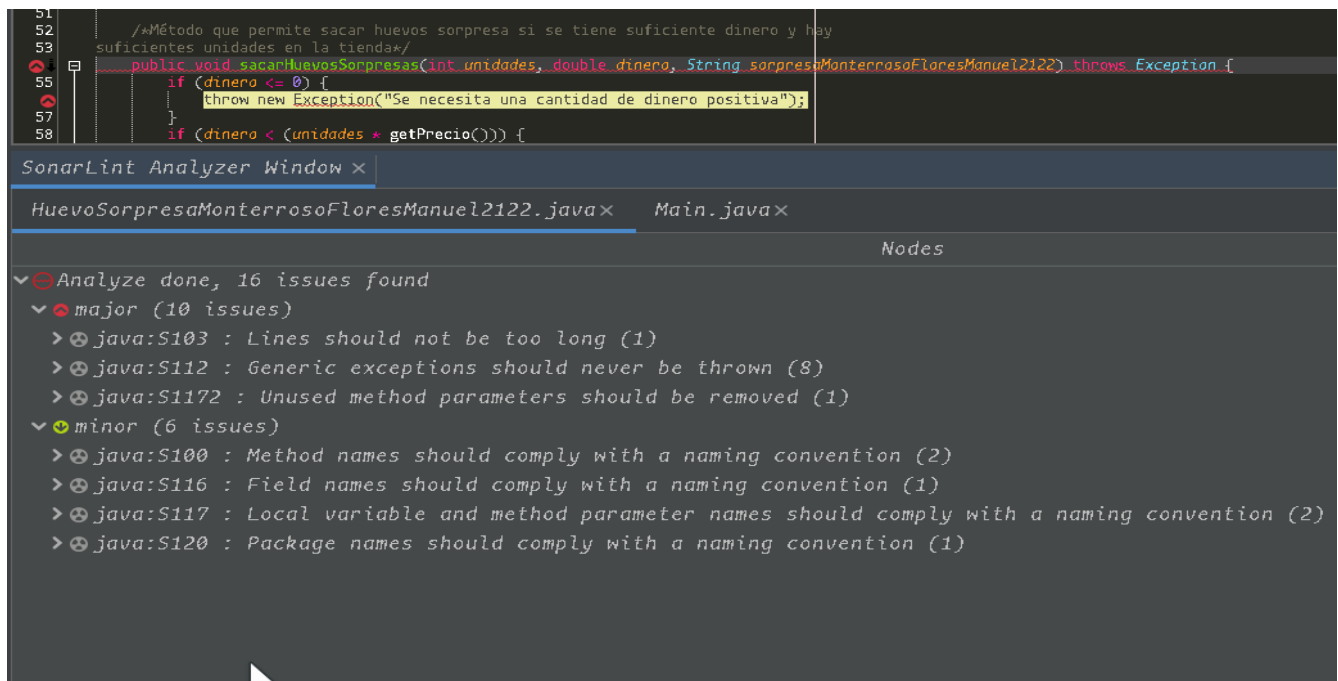
Plugin instalado.



Analizar código con el Pluguin.



Resultado del análisis primero con el plugin en la clase.



SonarLint es un plugin de analizador de código automático, el análisis automático reduce la complejidad para detectar problemas de base en el código, ya que los busca siguiendo una serie de reglas predefinidas. El plugin recibirá el código fuente de nuestro programa, lo procesará intentando averiguar la funcionalidad del mismo, y nos dará sugerencias, o nos mostrará posibles mejoras.

Esta herramienta basa su funcionamiento en detectar patrones, que son posibles errores en tiempo de ejecución, código que no se puede ejecutar nunca porque no se puede llegar a él, código que puede ser optimizado, expresiones lógicas que pueden ser simplificadas, malos usos del lenguaje, etc.

Tras el análisis del código nos da 10 reglas de carácter mayor en la clase, por ejemplo el error S1172 indica que nunca se ha usado un parámetro definido dentro de un método y que se puede borrar para limpiar código, de 6 reglas de clase menor.

También podemos observar que del error S103 (por líneas demasiado largas), sale un error antes de la modificación que nos pide la tarea.

Resultado del plugin en el primer análisis en Main.

```
51  /*Método que permite sacar huevos sorpresa si se tiene suficiente dinero y hay
52  suficientes unidades en la tienda*/
53  public void sacarHuevosSorpresas(int unidades, double dinero, String sorpresaMonterrosoFloresManuel2122) throws Exception {
54      if (dinero <= 0) {
55          throw new Exception("Se necesita una cantidad de dinero positiva");
56      }
57      if (dinero < (unidades * getPrecio())) {
58          // ...
59      }
60  }
```

SonarLint Analyzer Window x

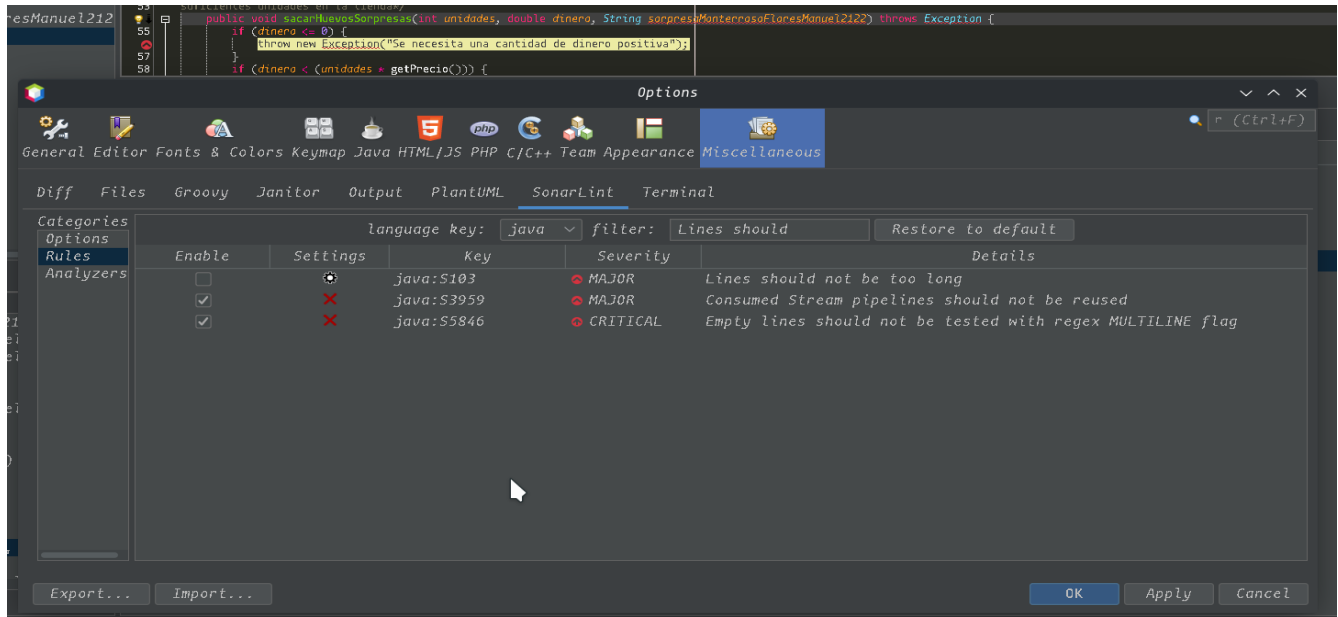
HuevoSorpresaMonterrosoFloresManuel2122.java x Main.java x

Nodes

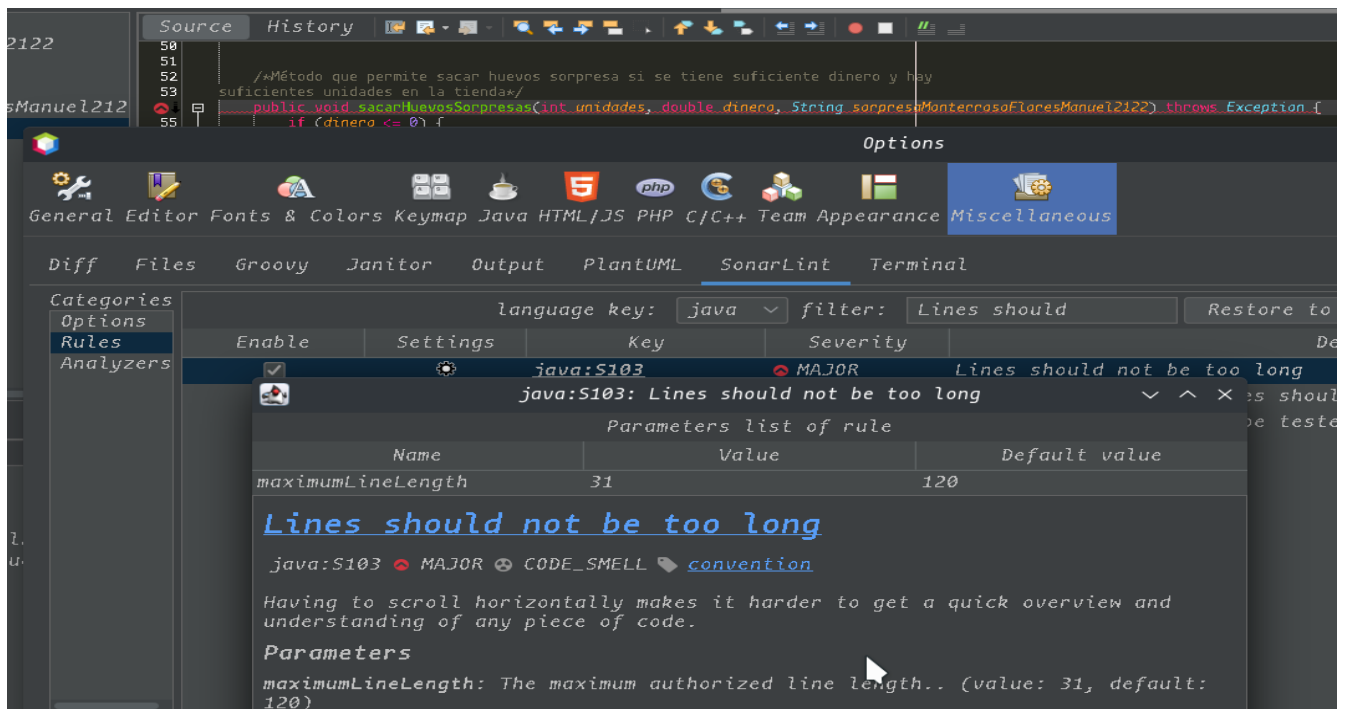
- ▼ Analyze done, 15 issues found
  - ▼ major (8 issues)
    - > java:S103 : Lines should not be too long (2)
    - > java:S106 : Standard outputs should not be used directly to log anything (6)
  - ▼ info (1 issue)
    - > java:S1135 : Track uses of "TODO" tags (1)
  - ▼ minor (6 issues)
    - > java:S100 : Method names should comply with a naming convention (2)
    - > java:S117 : Local variable and method parameter names should comply with a naming convention (3)
    - > java:S120 : Package names should comply with a naming convention (1)

Aquí podemos observar que en Main salen 8 advertencias de código mayor, 6 de código menor y 1 de información, de las cuales de mayor son 2 con el código de S103 (código de líneas demasiadas largas).

Buscando la opción de Lines should para modificarla.



Ahora vamos a modificar la regla de la longitud de las líneas que esta en 120 por 31 como longitud máxima en las líneas de código.



Como en la tarea pide que sean líneas mayores de 30 caracteres he puesto que salte el error a partir de que la línea tenga 31 caracteres.

Capturas con los nuevos análisis realizados con SonarLint con el cambio en la regla S103.

```
53 //suficientes unidades en la tienda*/
54 public void sacarHuevosSorpresas(int unidades, double dinero, String sorpresaMonterrosoFloresManuel2122) throws Exception {
55     if (dinero <= 0) {
56         throw new Exception("Se necesita una cantidad de dinero positiva");
57     }
58     if (dinero < (unidades * getPrecio())) {
59         throw new Exception("No tiene suficiente dinero");
60     }
61     if (unidades <= 0) {
62         throw new Exception("Es necesario indicar una cantidad positiva de huevos sorpresa");
63     }
64     if (this.getUnidades() <= unidades) {
65         throw new Exception("No hay suficientes huevos sorpresa en la tienda");
66     }
67     this.modificarUnidades(this.obtenerUnidades() - unidades);
68 }
69
70 //Método que permite aumentar el precio de venta de un huevo sorpresa, suma al precio actual
71
```

SonarLint Analyzer Window x

Main.java x    HuevoSorpresaMonterrosoFloresManuel2122.java x

Nodes

- ▼ Analyze done, 36 issues found
  - ▼ major (29 issues)
    - ⊗ java:S103 : Lines should not be too long (23)
    - ⊗ java:S106 : Standard outputs should not be used directly to log anything (6)
  - ▼ info (1 issue)
    - ⊗ java:S1135 : Track uses of "TODO" tags (1)
  - ▼ minor (6 issues)
    - ⊗ java:S100 : Method names should comply with a naming convention (2)
    - ⊗ java:S117 : Local variable and method parameter names should comply with a naming convention (3)
    - ⊗ java:S120 : Package names should comply with a naming convention (1)

```
50
51 //Método que permite sacar huevos sorpresa si se tiene suficiente dinero y hay
52 //suficientes unidades en la tienda*/
53 public void sacarHuevosSorpresas(int unidades, double dinero, String sorpresaMonterrosoFloresManuel2122) throws Exception {
54     if (dinero <= 0) {
55         throw new Exception("Se necesita una cantidad de dinero positiva");
56     }
57     if (dinero < (unidades * getPrecio())) {
58         throw new Exception("No tiene suficiente dinero");
59     }
60     if (unidades <= 0) {
61         throw new Exception("Es necesario indicar una cantidad positiva de huevos sorpresa");
62     }
63     if (this.getUnidades() <= unidades) {
64         throw new Exception("No hay suficientes huevos sorpresa en la tienda");
65     }
66     this.modificarUnidades(this.obtenerUnidades() - unidades);
67 }
68
69 //Método que permite aumentar el precio de venta de un huevo sorpresa, suma al precio actual
70
71
```

SonarLint Analyzer Window x

Main.java x    HuevoSorpresaMonterrosoFloresManuel2122.java x

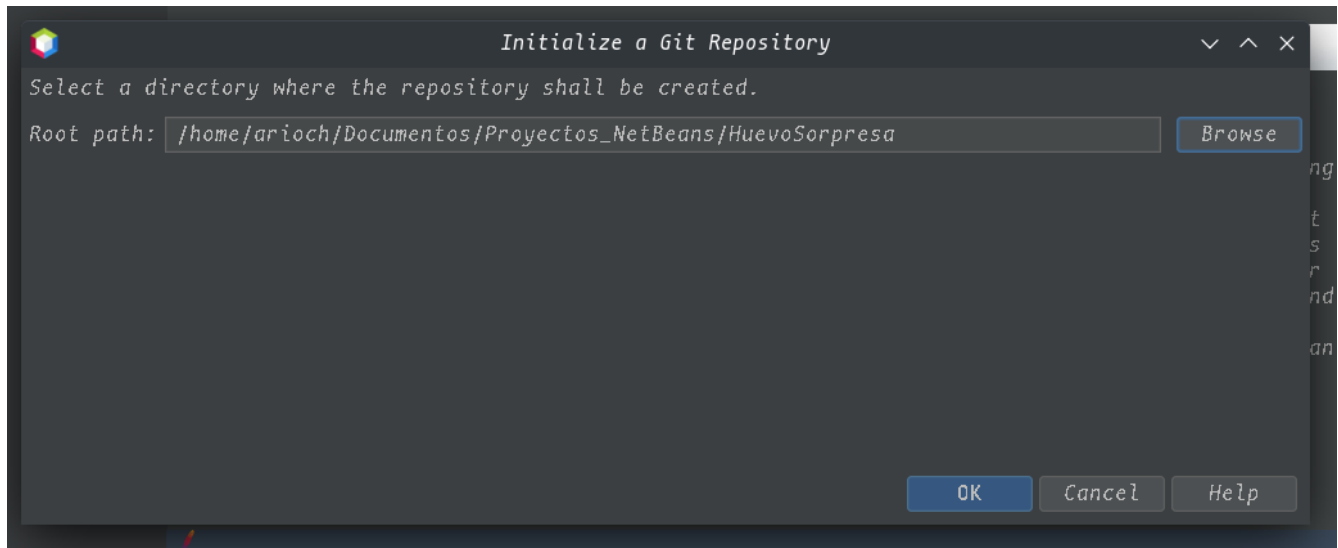
Nodes

- ▼ Analyze done, 72 issues found
  - ▼ major (66 issues)
    - ⊗ java:S103 : Lines should not be too long (57)
    - ⊗ java:S112 : Generic exceptions should never be thrown (8)
    - ⊗ java:S1172 : Unused method parameters should be removed (1)
  - ▼ minor (6 issues)
    - ⊗ java:S100 : Method names should comply with a naming convention (2)
    - ⊗ java:S116 : Field names should comply with a naming convention (1)
    - ⊗ java:S117 : Local variable and method parameter names should comply with a naming convention (2)
    - ⊗ java:S120 : Package names should comply with a naming convention (1)

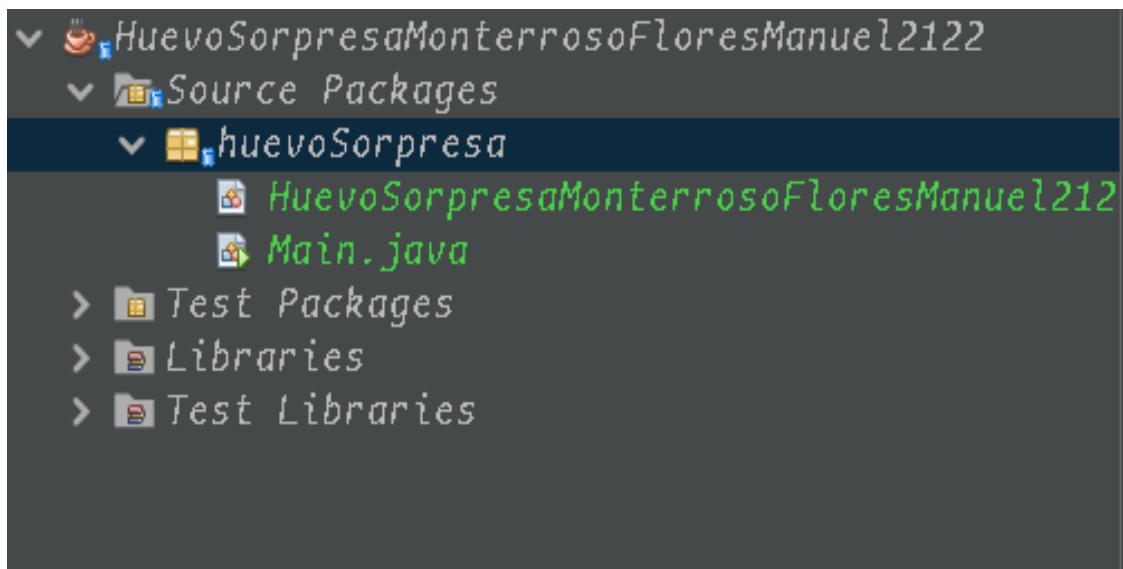
Ahora podemos ver como han aumentado el número de excepciones en la regla S103, en Main hemos pasado de 2 a 23 excepciones y en la clase hemos pasado de 1 a 57 excepciones.

## CONTROL DE VERSIONES.

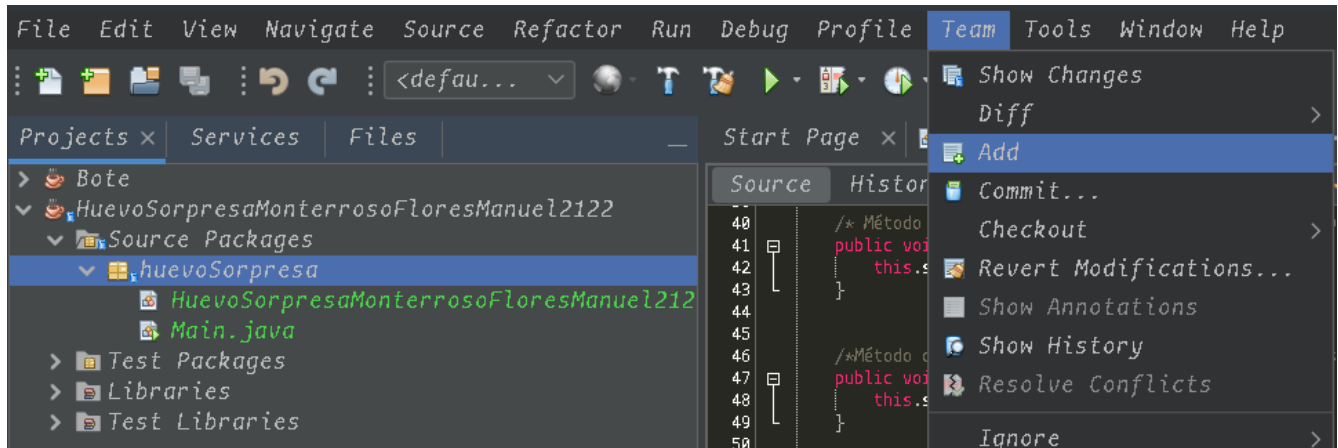
Iniciamos GIT con el proyecto HuevoSorpresa.



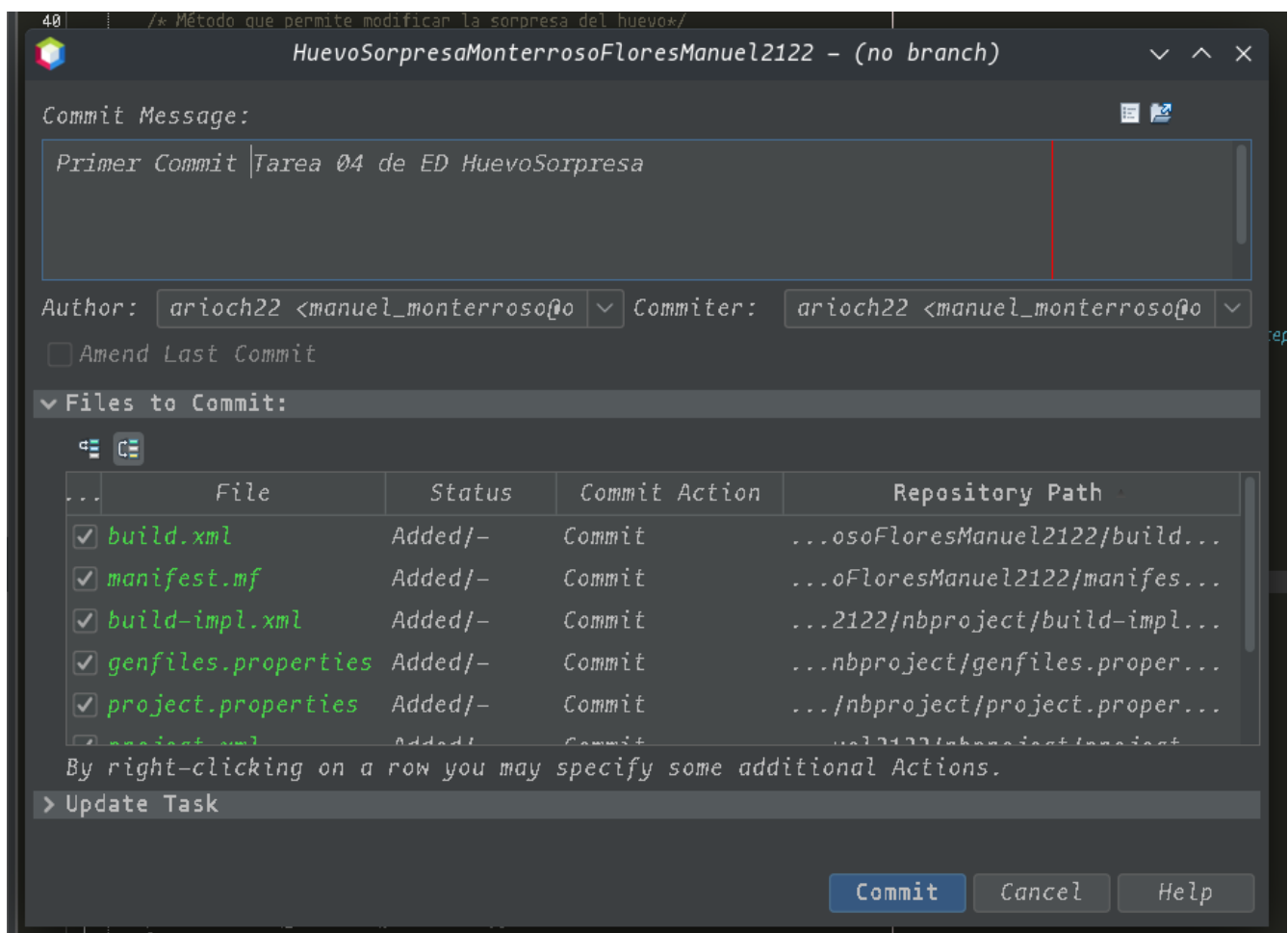
Proyecto ya inicializado.



Ahora vamos a realizar el ADD.

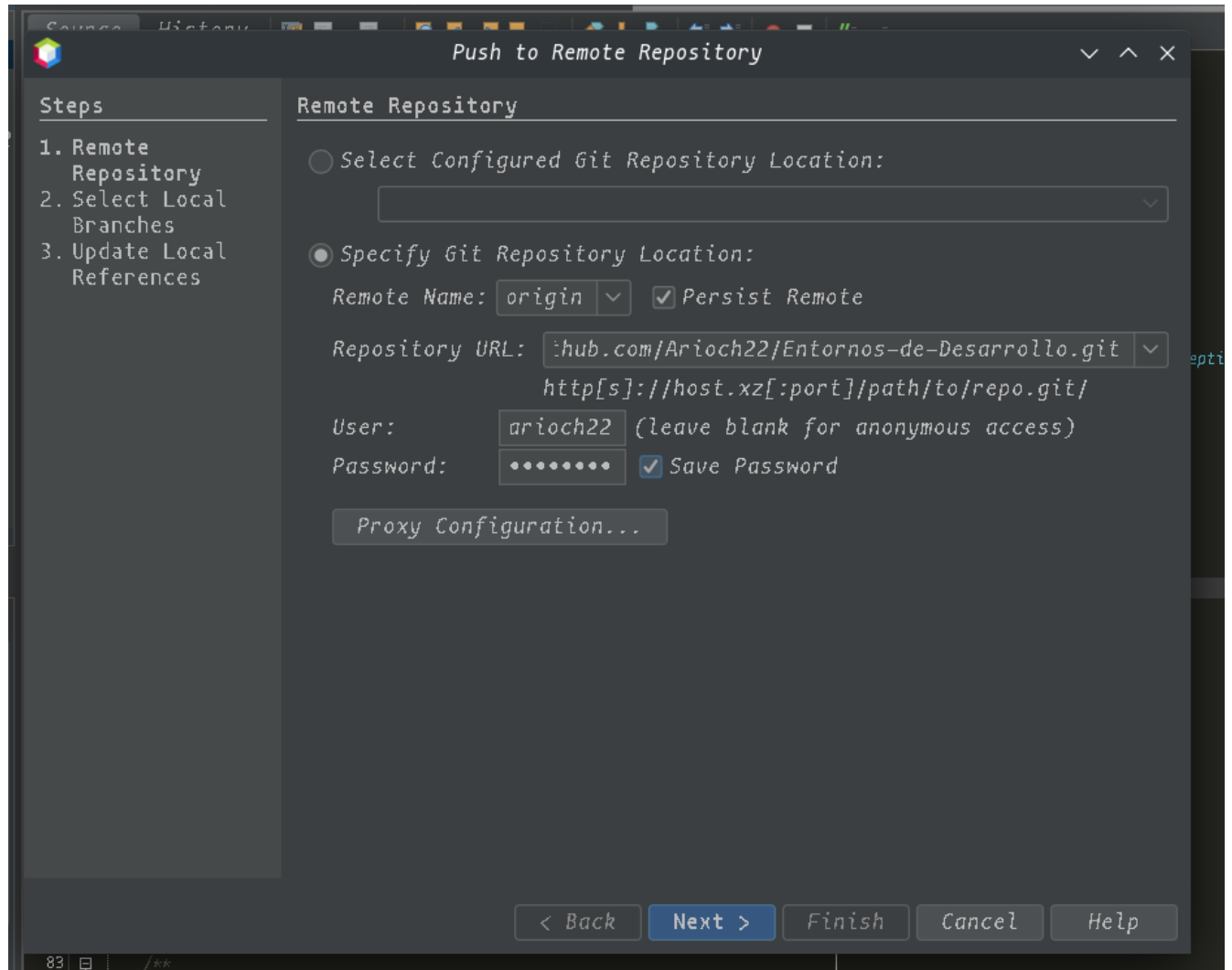


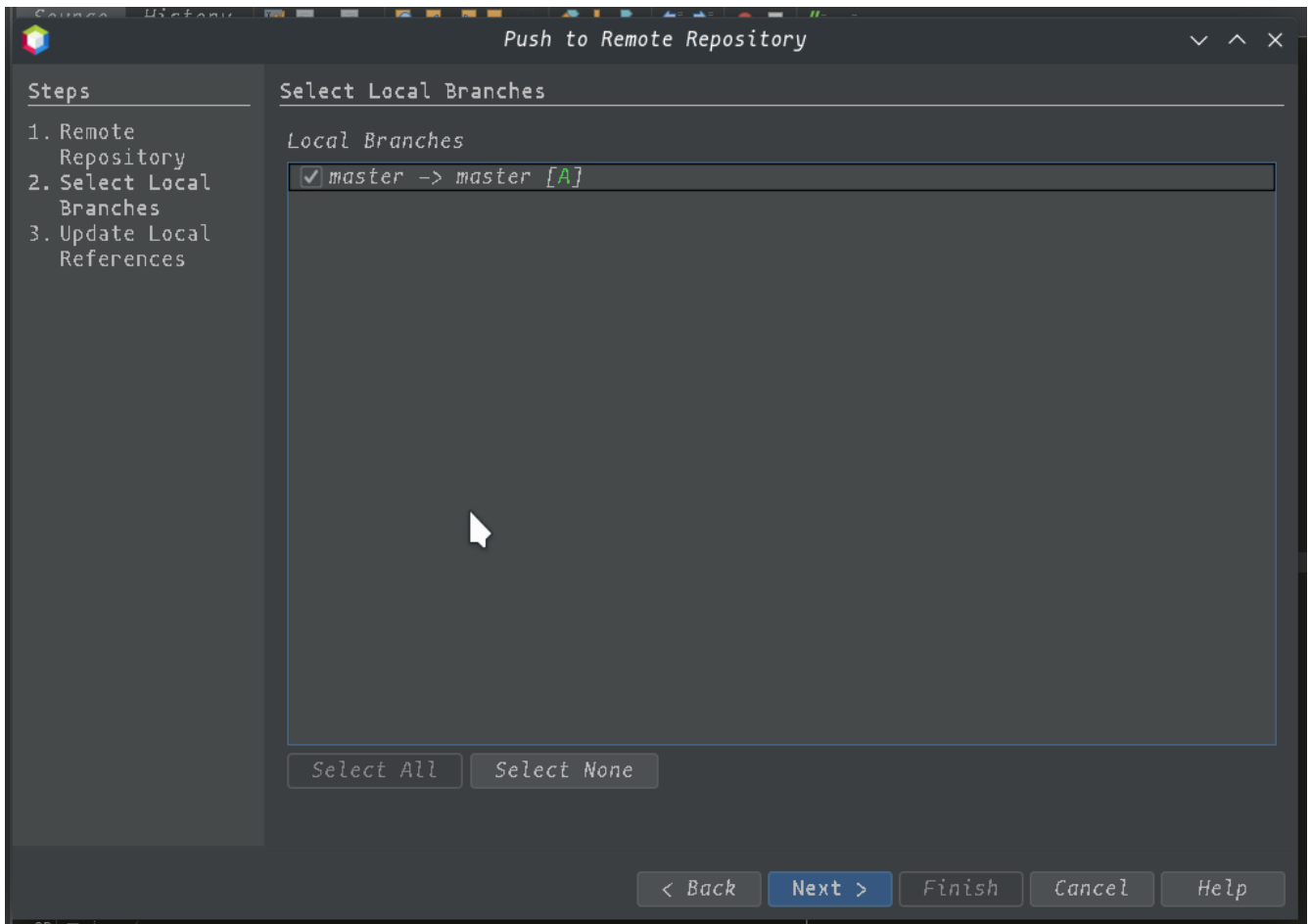
Ahora vamos a realizar el commit.

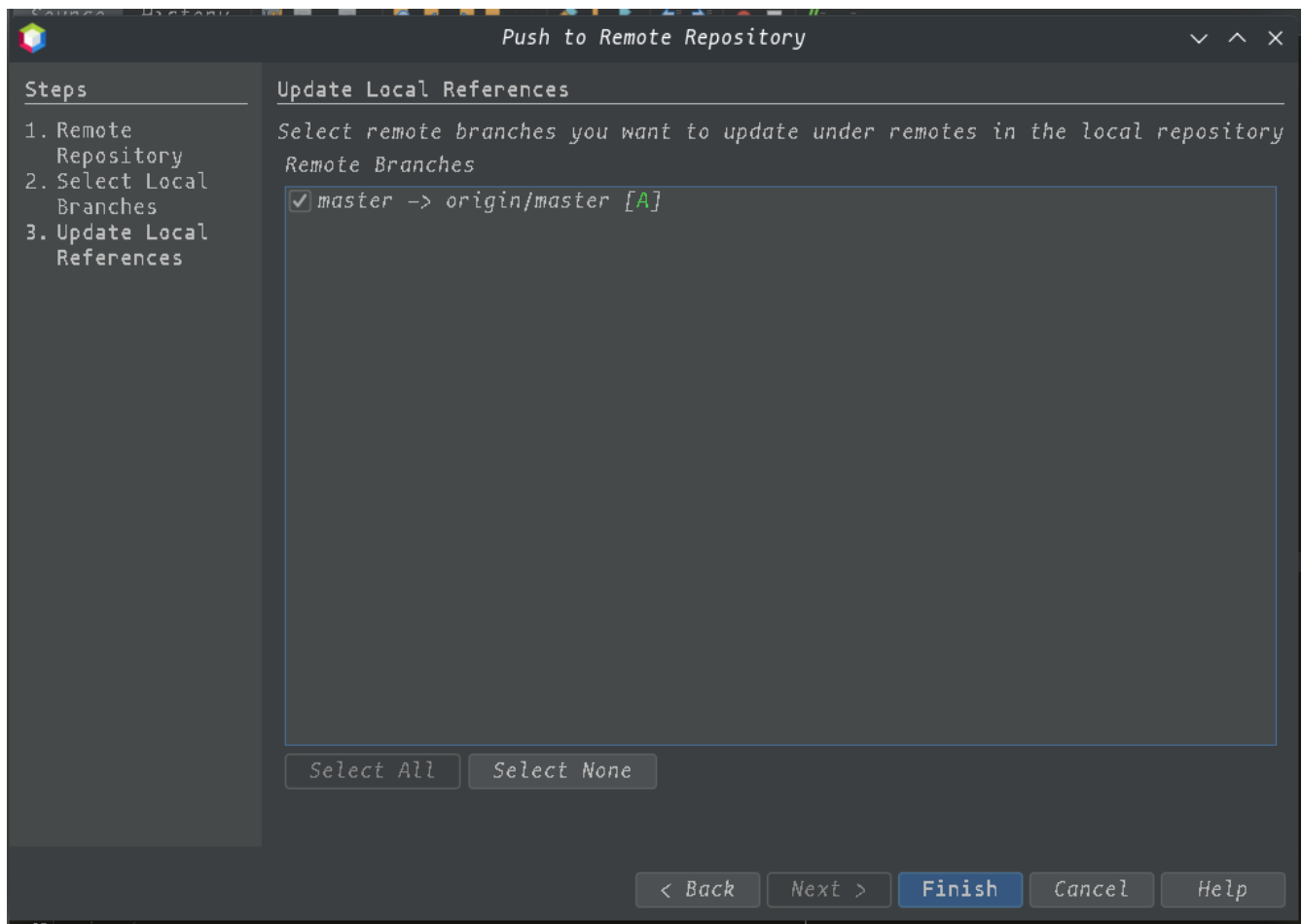





Realizando el Push para subir el proyecto del repositorio local a la nube.







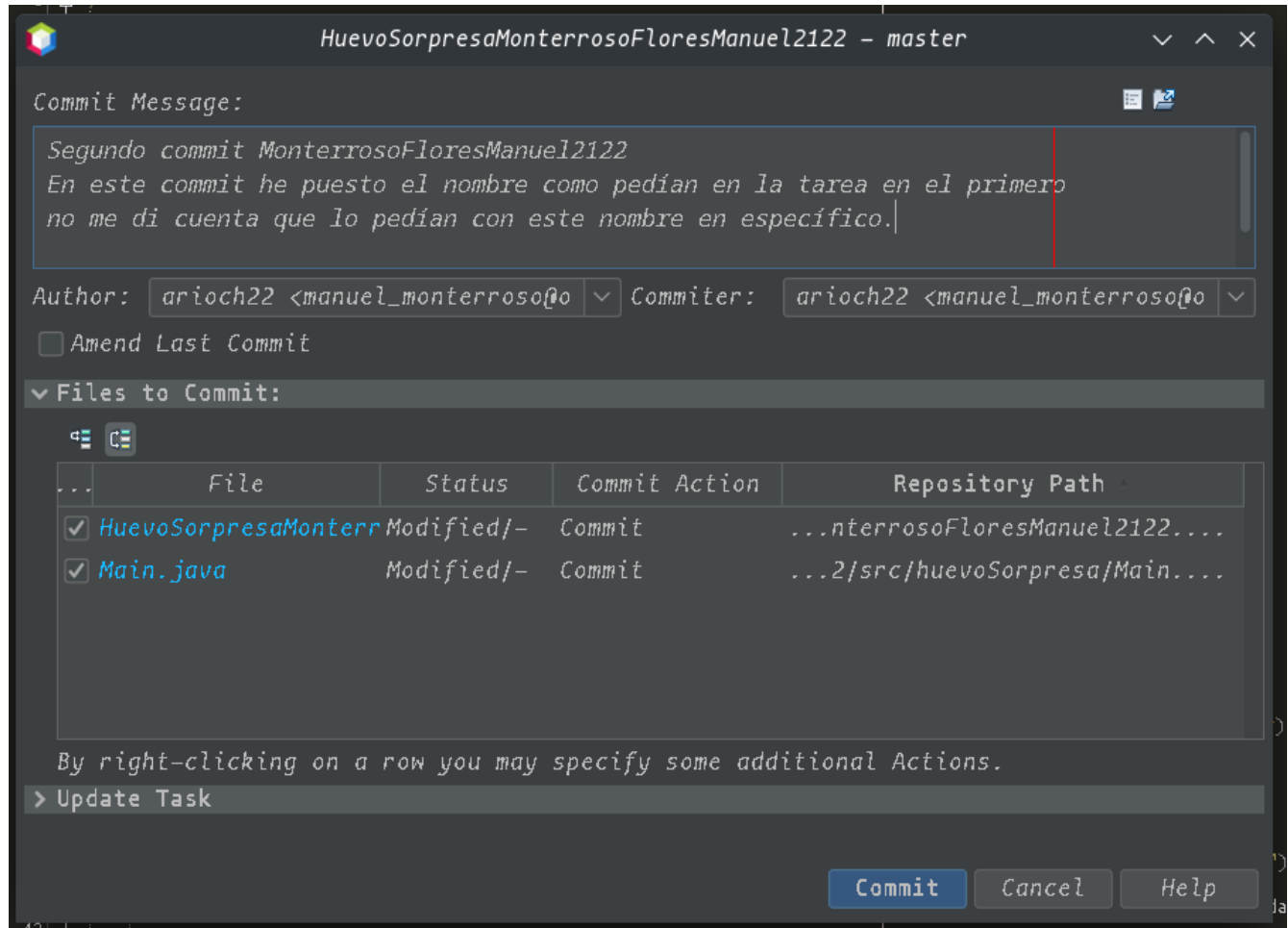
Proyecto subido a GitHub.

 master ▾ [Entornos-de-Desarrollo](#) / [HuevoSorpresaMonterrosoFloresManuel2122](#) /

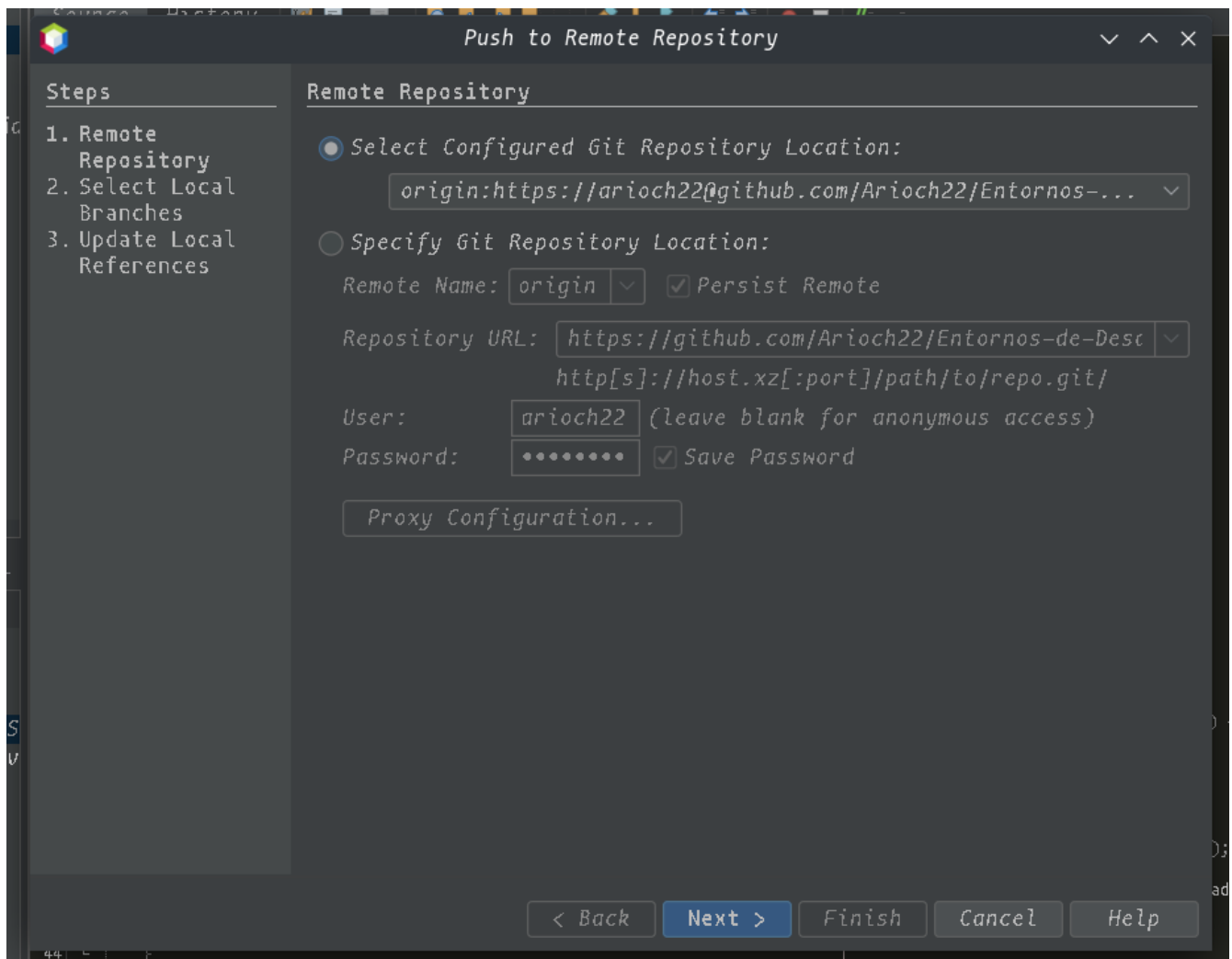
 <b>Arioch22</b> Primer Commit Tarea 04 de ED HuevoSorpresa		
..		
	nbproject	Primer Commit Tarea 04 de ED HuevoSorpresa
	src/huevoSorpresa	Primer Commit Tarea 04 de ED HuevoSorpresa
	build.xml	Primer Commit Tarea 04 de ED HuevoSorpresa
	manifest.mf	Primer Commit Tarea 04 de ED HuevoSorpresa

Tras general el JavaDoc como se ha producido cambios en el código ahora se podrá realizar el segundo commit que se pide en la tarea, lo primero será realizar el add como en el anterior punto y después realizar el commit.

Captura con el segundo commit realizandose.



Realizando el Push a la nube.



Proyecto ya actualizado a GitHub.

