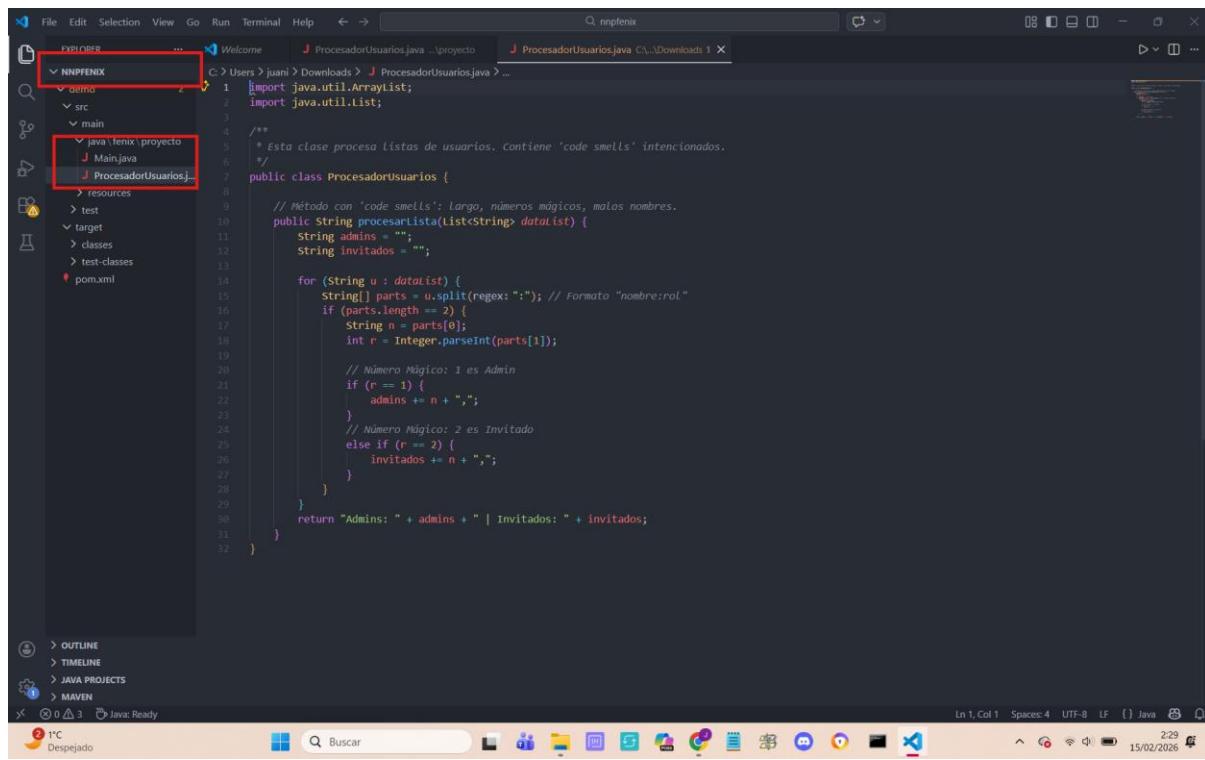


PROYECTO FÉNIX JUAN SILVESTRE DURO 1ºDAM

PASO 1 — PASO 1 — Crear el proyecto en el IDE Captura: Proyecto creado con la clase abierta.



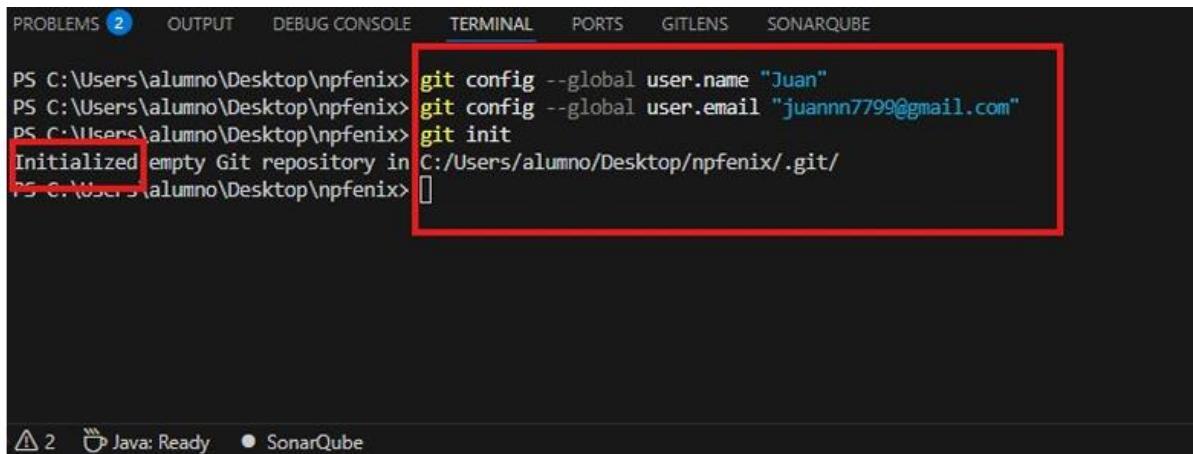
```
File Edit Selection View Go Run Terminal Help < - > Q npfpennix

EXPLORE C:\> Users > juan > Downloads > NPFPENIX > ProcesadorUsuarios.java > ...
Welcome J ProcesadorUsuarios.java ...> proyecto J ProcesadorUsuarios.java C:\...\Downloads\1 X

C:\> Users > juan > Downloads > NPFPENIX > demo > 1 import java.util.ArrayList;
      import java.util.List;
      ...
      /**
       * Esta clase procesa Listas de usuarios. Contiene 'code smells' intencionados.
       */
      public class ProcesadorUsuarios {
          ...
          // Método con 'code smells': Largo, números mágicos, malos nombres.
          public String procesarLista(List<String> dataList) {
              String admins = "";
              String invitados = "";
              ...
              for (String u : dataList) {
                  String[] parts = u.split(regex ":" );
                  if (parts.length == 2) {
                      String n = parts[0];
                      int r = Integer.parseInt(parts[1]);
                      ...
                      // Número Mágico: 1 es Admin
                      if (r == 1) {
                          admins += n + ",";
                      }
                      // Número Mágico: 2 es Invitado
                      else if (r == 2) {
                          invitados += n + ",";
                      }
                  }
              }
              return "Admins: " + admins + " | Invitados: " + invitados;
          }
      }
}

Outline Timeline Java Projects Maven
Ln 1, Col 1 Spaces: 4 UTF-8 LF Java 2:29 15/02/2026
Java Ready
IIC Despidojado Buscar 2:29 15/02/2026
```

PASO 2 — Inicializar repositorio Git local Captura: Git inicializado.



```
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SONARQUBE

PS C:\Users\alumno\Desktop\npfenix> git config --global user.name "Juan"
PS C:\Users\alumno\Desktop\npfenix> git config --global user.email "juannn7799@gmail.com"
PS C:\Users\alumno\Desktop\npfenix> git init
Initialized empty Git repository in C:/Users/alumno/Desktop/npfenix/.git/
PS C:\Users\alumno\Desktop\npfenix> []

Terminal 2 Java: Ready SonarQube
```

```

import java.util.ArrayList;
import java.util.List;

/*
 * Esta clase procesa listas de usuarios. Contiene 'code smells' intencionados.
 */
public class ProcesadorUsuarios {

    // Método con 'code smells': Largo, números mágicos, malos nombres.
    public String procesarLista(List<String> dataList) {
        String admins = "";
        String invitados = "";

        for (String u : dataList) {
            String[] parts = u.split(regex: ":");

            if (parts.length == 2) {
                String n = parts[0];
                int r = Integer.parseInt(parts[1]);

                // Número Mágico: 1 es Admin
                if (r == 1) {
                    admins += n + ",";
                }
                // Número Mágico: 2 es Invitado
                else if (r == 2) {
                    invitados += n + ",";
                }
            }
        }
        return "Admins: " + admins + " | Invitados: " + invitados;
    }
}

```

PASO 3 — Crear repositorio en GitHub creamos un nuevo repositorio para subir los push

PROYECTO-F-NIX Público

Configurar GitHub Copilot

Añadir colaboradores a este repositorio

Configuración rápida : si ya has hecho este tipo de cosas antes

...o crear un nuevo repositorio en la línea de comandos

```

echo "# PROYETO-F-NIX" >> README.md
git init
git add README.md
git commit -m "primer commit"
git branch -M main
git remote add origin https://github.com/Juansd79/PROYETO-F-NIX.git
git push -u origin main

```

...o enviar un repositorio existente desde la línea de comandos

PASO 4 — Conectar repositorio local con remoto se conecta para poder mostrar los cambios

PROYECTO-F-NIX Público

Acerca de
No se proporcionan descripción, sitio web ni temas.

Lanzamientos
No hay comunicados publicados
[Crear una nueva versión](#)

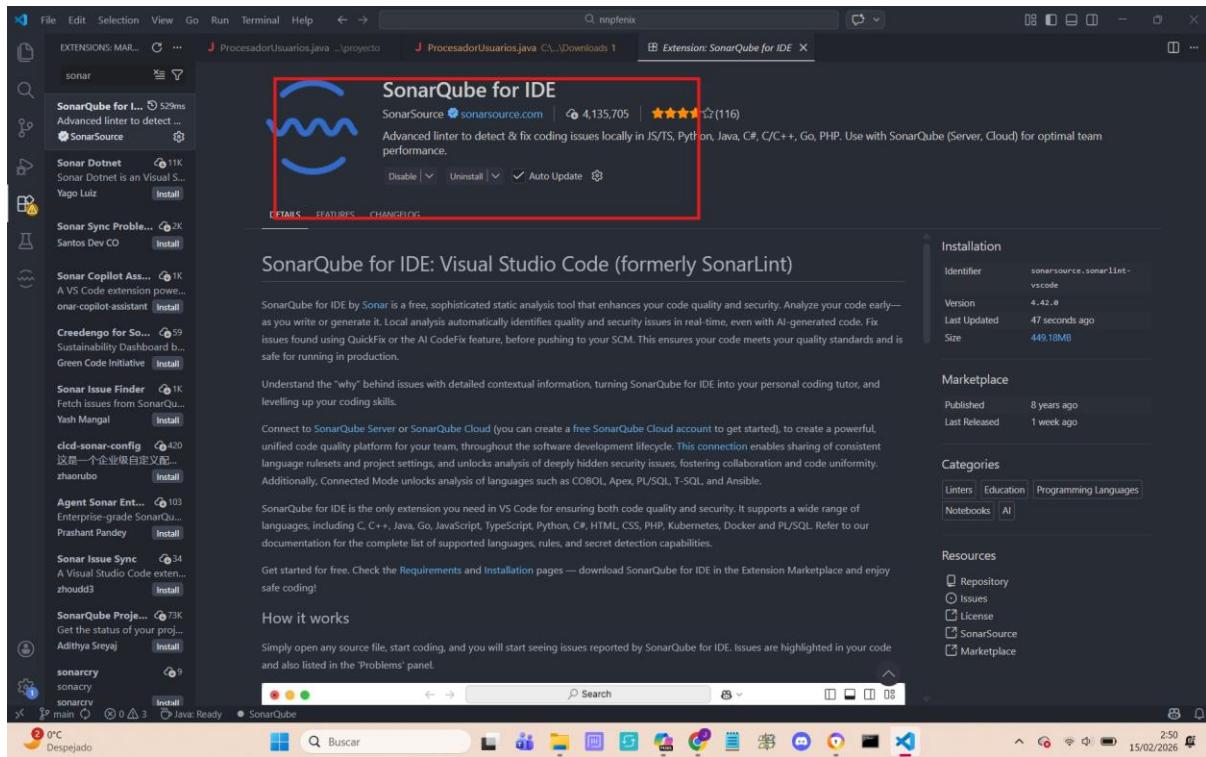
Paquetes
No hay paquetes publicados
[Publica tu primer paquete](#)

© 2026 GitHub, Inc. Términos Privacidad Seguridad Estado Comunidad Documentos Contacto Administrar cookies No comparta mi información personal

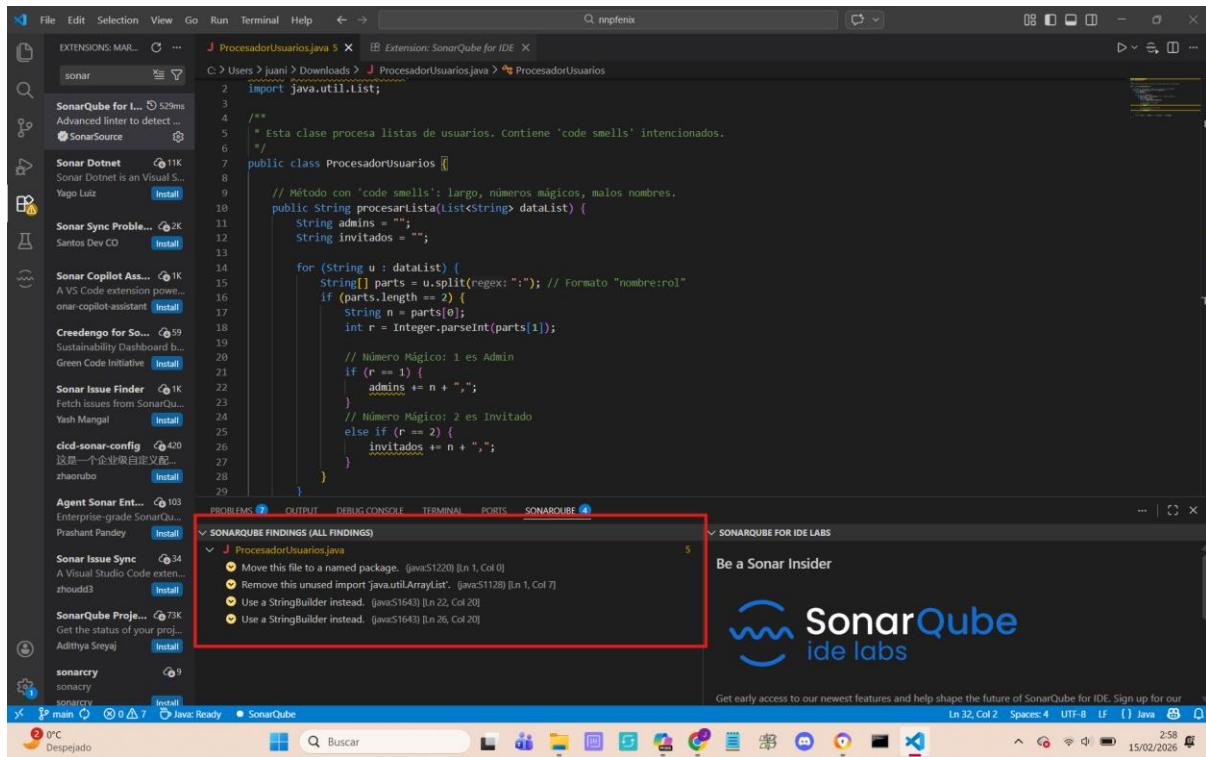
```
// Número Mágico: 1 es Admin
if (r == 1) {
    admins += n + ",";
}
// Número Mágico: 2 es Invitado
else if (r == 2) {
    invitados += n + ",";
}
```

git remote add origin https://github.com/Juansd79/PROYECTO-F-NIX.git

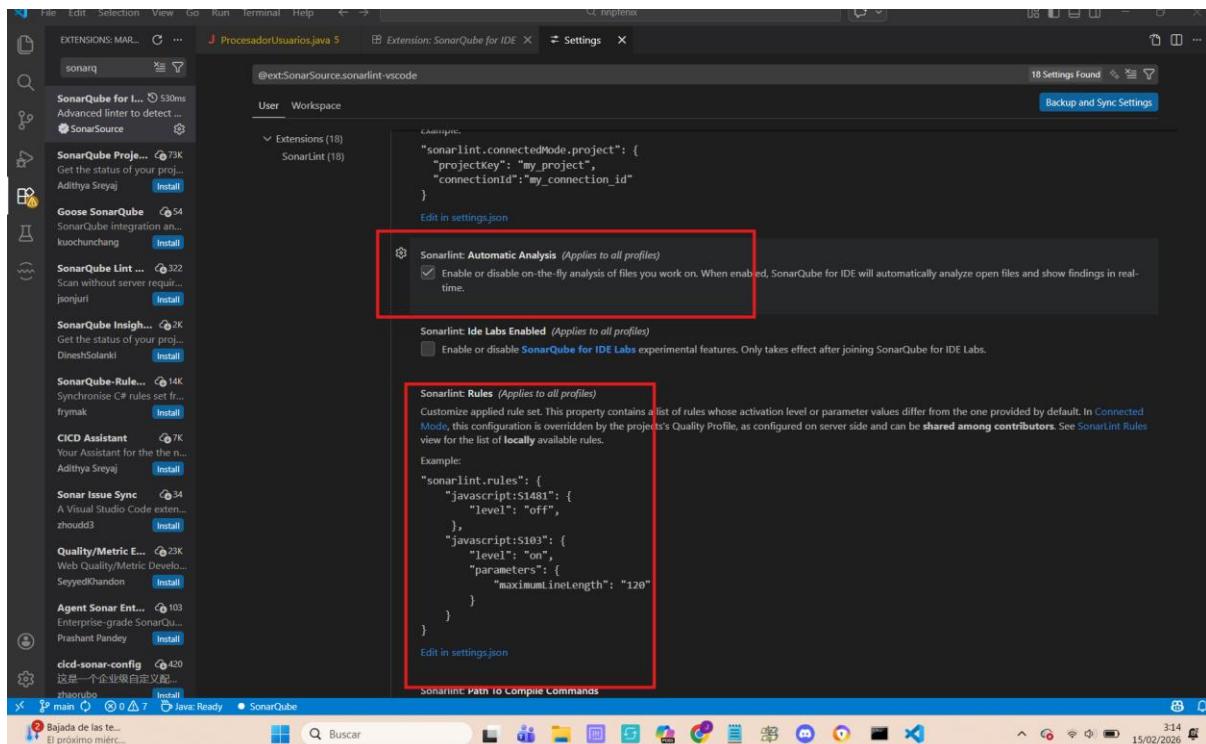
PASO 6 — Instalar SonarQube se instala para luego en settings entrar en sonarlint y acceder a las normas para activarlas



PASO 7 — Ejecutar análisis se ejecuta el análisis para ver los problemas



PASO 8 — Revisar reglas para comprobar que todo esta en orden y los tests pasan correctamente



The screenshot shows a Java test class named `ProcesadorUsuariosTEST`. The code imports `Assertions.assertEquals` and `Test` from `junit.jupiter`, and `ProcesadorUsuarios` and `List` from `com.example`. The class contains a single test method, `testProcesarListaComportamientoActual`, which creates a new instance of `ProcesadorUsuarios`, initializes a list of strings, and then asserts that the result of calling `procesarLista` on the list matches the expected output: "Admins: Ana,Eva, | Invitados: Luis,".

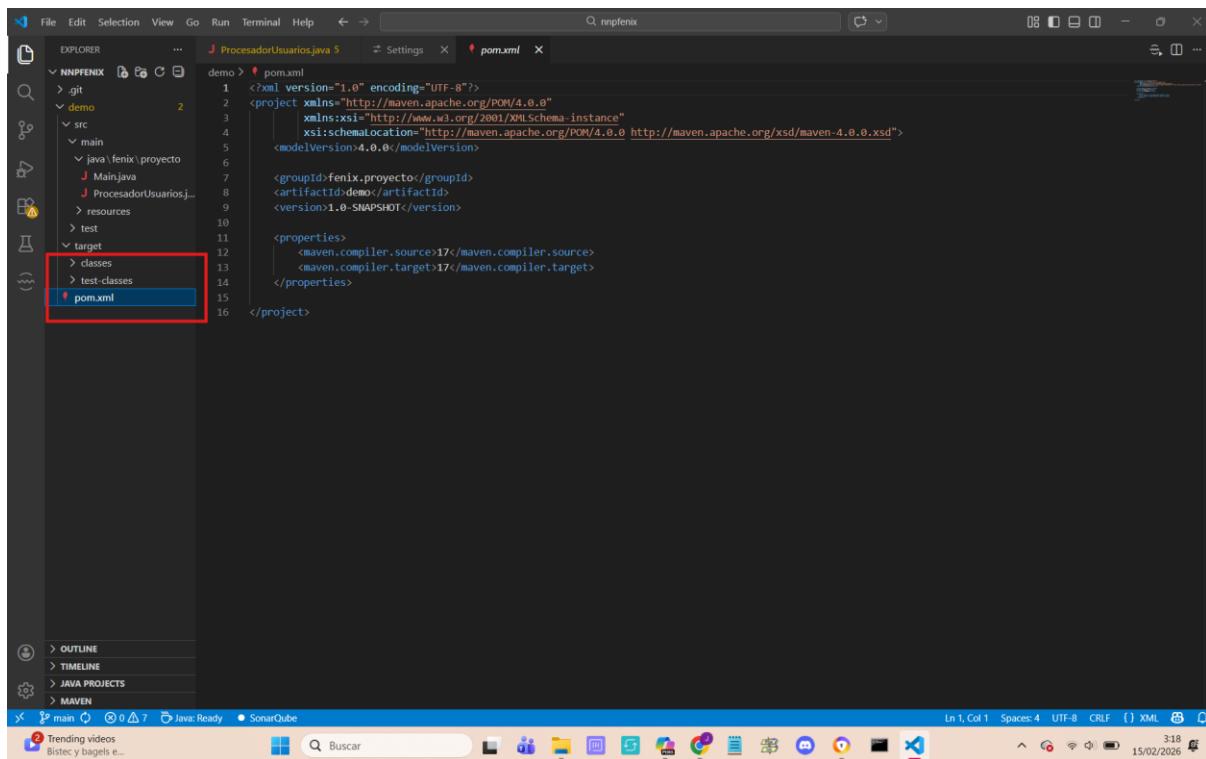
```
1 import static org.junit.jupiter.api.Assertions.assertEquals;
2 import org.junit.jupiter.api.Test;
3
4 import com.example.ProcesadorUsuarios;
5
6 import java.util.List;
7
8 public class ProcesadorUsuariosTEST {
9
10     @Test
11     void testProcesarListaComportamientoActual() {
12         ProcesadorUsuarios procesador = new ProcesadorUsuarios();
13
14         List<String> lista = List.of("Ana:1", "Luis:2", "Eva:1", "Juan:99");
15         String resultado = procesador.procesarLista(lista);
16
17         assertEquals("Admins: Ana,Eva, | Invitados: Luis,", resultado);
18     }
19 }
20
```

The bottom of the screen shows the VS Code interface with tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, TEST RESULTS (which is selected), PORTS, and SONARQUBE. The TEST RESULTS tab displays the execution results of the test:

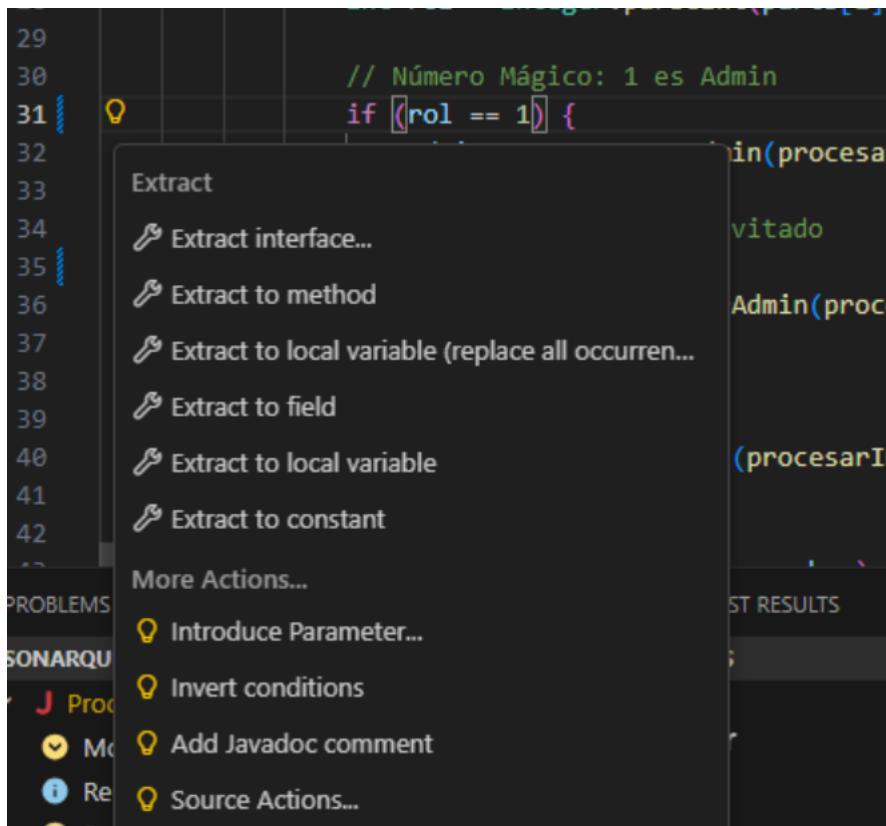
```
%TESTC 1 v2
%TSTTREE2,ProcesadorUsuariosTEST,true,1,false,1,ProcesadorUsuariosTEST,,[engine:junit-jupiter]/[class:ProcesadorUsuariosTEST]
%TSTTREE3,testProcesarListaComportamientoActual(ProcesadorUsuariosTEST),false,1,false,2,testProcesarListaComportamientoActual(),,[engine:junit-jupiter]/[class:ProcesadorUsuariosTEST]/[method:testProcesarListaComportamientoActual()]
%TESTS 3,testProcesarListaComportamientoActual(ProcesadorUsuariosTEST)
%TESTE 3,testProcesarListaComportamientoActual(ProcesadorUsuariosTEST)
```

A right-click context menu is open over the last test result, showing options like "Run", "Run All", and "Run Selection". A tooltip for "Activar Windows" is visible at the bottom right.

PASO 9 — Añadir JUnit al proyecto se añade junit al proyecto



PASO 13 — Extraer constantes escogemos los numeros y entramos en la opcion extract y damos a extract to constant



```

1 import java.util.ArrayList;
2 import java.util.List;
3
4 /**
5  * Esta clase procesa listas de usuarios. Contiene 'code smells' intencionados.
6 */
7 public class ProcesadorUsuarios {
8
9     private static final int _2 = 2;
10    private static final int _1 = 1;
11
12    // Método con 'code smells': largo, números mágicos, malos nombres.
13    public String procesarLista(List<String> dataList) {
14        String admins = "";
15        String invitados = "";
16
17        for (String u : dataList) {
18            String[] parts = u.split(":"); // Formato "nombre:rol"
19            if (parts.length == 2) {
20                String n = parts[0];
21                int r = Integer.parseInt(parts[1]);
22
23                // Número Mágico: 1 es Admin
24                if (r == _1) {
25                    admins += n + ",";
26                }
27                // Número Mágico: 2 es Invitado
28                else if (r == _2) {
29                    invitados += n + ",";
30                }
31            }
32        }
33
34        return "Admins: " + admins + " | Invitados: " + invitados;
35    }
36
37 }

```

The screenshot shows the Visual Studio Code interface with the Java project 'NPFPENIX' open. The file 'ProcesadorUsuarios.java' is the active editor. A specific block of code is highlighted with a red box:

```

private static final int _2 = 2;
private static final int _1 = 1;

```

PASO 14 — Renombrar variables utilizamos el comando f2 para cambiar los nombres de las variables rol n part

```

1 import java.util.ArrayList;
2 import java.util.List;
3
4 /**
5  * Esta clase procesa listas de usuarios. Contiene 'code smells' intencionados.
6 */
7 public class ProcesadorUsuarios {
8
9     private static final int _2 = 2;
10    private static final int _1 = 1;
11
12    // Método con 'code smells': largo, números mágicos, malos nombres.
13    public String procesarLista(List<String> Usuarios) {
14        String admins = "";
15        String invitados = "";
16
17        for (String u : Usuarios) {
18            String[] partes = u.split(":"); // Formato "nombre:rol"
19            if (partes.length == 2) {
20                String nombre = partes[0];
21                int rol = Integer.parseInt(partes[1]);
22
23                // Número Mágico: 1 es Admin
24                if (rol == _1) {
25                    admins += nombre + ",";
26                }
27                // Número Mágico: 2 es Invitado
28                else if (rol == _2) {
29                    invitados += nombre + ",";
30                }
31            }
32        }
33
34        return "Admins: " + admins + " | Invitados: " + invitados;
35    }
36
37 }

```

The screenshot shows the Visual Studio Code interface with the Java project 'NPFPENIX' open. The file 'ProcesadorUsuarios.java' is the active editor. The variable names 'n' and 'r' have been renamed to 'nombre' and 'rol' respectively, as indicated by the red boxes around them.

PASO 15 — Extract Method extraemos el método del bucle if

```
demo > src > main > java > fenix > proyecto > J ProcesadorUsuariosTest.java 5, U ● J Main.java 1 pom.xml ●

1 import java.util.ArrayList;
2 import java.util.List;
3
4 /**
5  * Esta clase procesa listas de usuarios. Contiene 'code smells' intencionados.
6  */
7 public class ProcesadorUsuarios {
8
9     private static final int _2 = 2;
10    private static final int _1 = 1;
11
12    // Método con 'code smells': largo, números mágicos, malos nombres.
13    public String procesarLista(List<String> Usuarios) {
14        String admins = "";
15        String invitados = "";
16
17        for (String u : Usuarios) {
18            String[] partes = u.split(":"); // Formato "nombre:rol"
19            if (partes.length == 2) {
20                String nombre = partes[0];
21                int rol = Integer.parseInt(partes[1]);
22
23                // Número Mágico: 1 es Admin
24                if (rol == _1) {
25                    admins += nombre + ",";
26                }
27                // Número Mágico: 2 es Invitado
28                else if (rol == _2) {
29                    invitados += nombre + ",";
30                }
31            }
32        }
33
34        return "Admins: " + admins + " | Invitados: " + invitados;
35    }
36
37    private String extracted(List<String> Usuarios, String admins, String invitados) {
38        for (String u : Usuarios) {
39            String[] partes = u.split(":");
40            if (partes.length == 2) {
41                String nombre = partes[0];
42                int rol = Integer.parseInt(partes[1]);
43
44                // Número Mágico: 1 es Admin
45                if (rol == _1) {
46                    admins += nombre + ",";
47                }
48                // Número Mágico: 2 es Invitado
49                else if (rol == _2) {
50                    invitados += nombre + ",";
51                }
52            }
53        }
54        return "Admins: " + admins + " | Invitados: " + invitados;
55    }
56
57 }
```

PASO 18 — Añadir Javadoc se añade los comentarios javadoc

```
C:\Users\alumno\Desktop\proyecto-fenix> mvn javadoc:javadoc
[INFO] Scanning for projects...
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/mojo/maven-metadata.xml
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-metadata.xml
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-metadata.xml (4 kB at 33 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/mojo/maven-metadata.xml (21 kB at 46 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-javadoc-plugin/maven-javadoc-plugin/3.2.0/maven-javadoc-plugin-3.2.0.jar
[INFO] ------------------------------------------------------------------------
[INFO] Building proyecto-fenix 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-javadoc-plugin:3.2.0:javadoc (default-cli) @ proyecto-fenix ---
[INFO]
```

The screenshot shows a Java IDE interface with several windows open. The main window displays a Java file named `ProcesadorUsuarios.java`. A red box highlights a Javadoc comment block:

```
 /**
 * Esta clase procesa listas de usuarios. Contiene 'code smells' intencionados.
 */
public class ProcesadorUsuarios {

    /**
     * Procesa una lista de strings y genera un resumen de roles.
     * @param usuarios Lista de strings en formato "nombre:rol".
     * @return Un String con los nombres de Admins e Invitados separados.
    */

    private static final int ROL_INVITADO = 2;
    private static final int ROL_ADMIN = 1;

    // Método con 'code smells': largo, números mágicos, malos nombres.
    public String procesarLista(List<String> usuarios) {
        String admins = "";
        String invitados = "";

        for (String usuario : usuarios) {
            String[] parts = procesarAdmin(procesarInvitado(usuario)).split(regex);
            if (parts.length == 2) {
                String nombre = parts[0];
                int rol = Integer.parseInt(parts[1]);

                // Número Mágico: 1 es Admin
                if (rol == ROL_ADMIN) {
                    admins += procesarAdmin(procesarInvitado(nombre)) + ",";
                }
                // Número Mágico: 2 es Invitado
                else if (rol == ROL_INVITADO) {
                    invitados += procesarAdmin(procesarInvitado(nombre)) + ",";
                }
            }
        }
        return "Admins: " + procesarAdmin(procesarInvitado(admins)) + " | Invitados: " + procesarAdmin(procesarInvitado(invitados));
    }
}
```

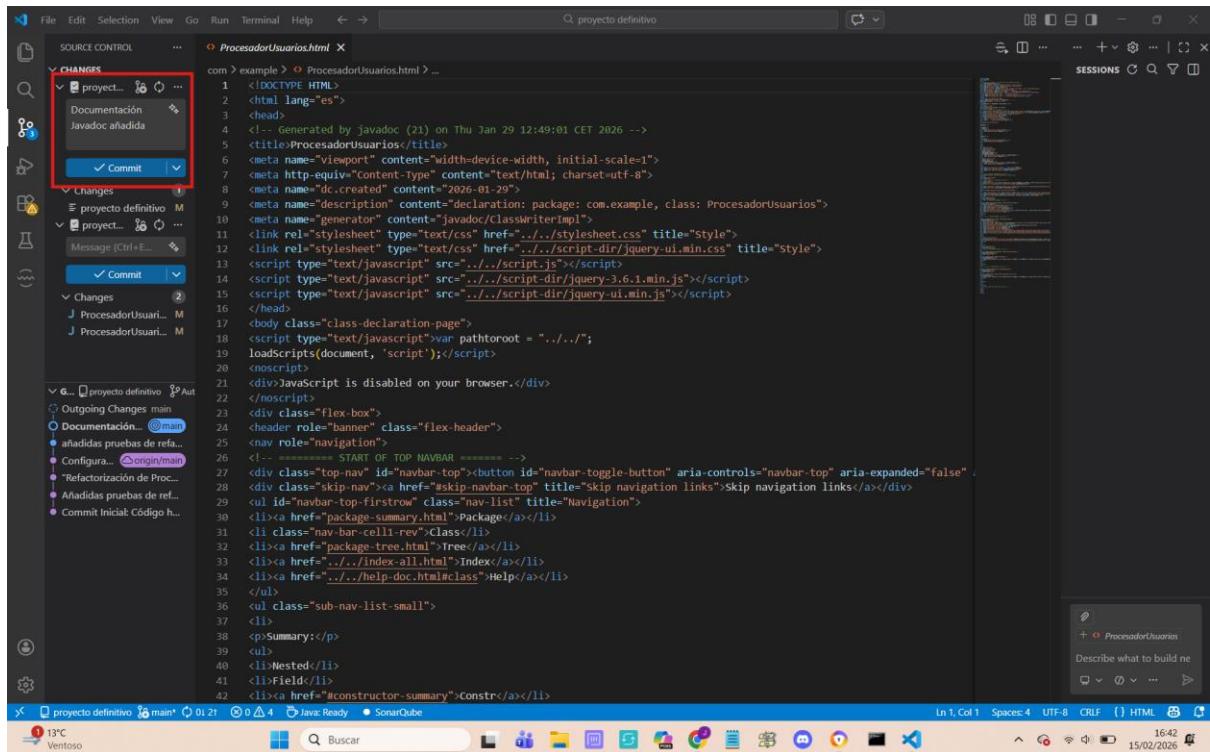
The left sidebar shows a project structure for a Java application named `NNPFENIX`, containing `demo`, `src`, `main`, `resources`, and `test` packages. The `test` package contains a `java` folder with a `ProcesadorUsuariosTest.java` file. The bottom status bar indicates the file is 49 lines long, the encoding is UTF-8, and the Java version is 1.8.

PASO 19 — Generar documentación se genera la documentacion html

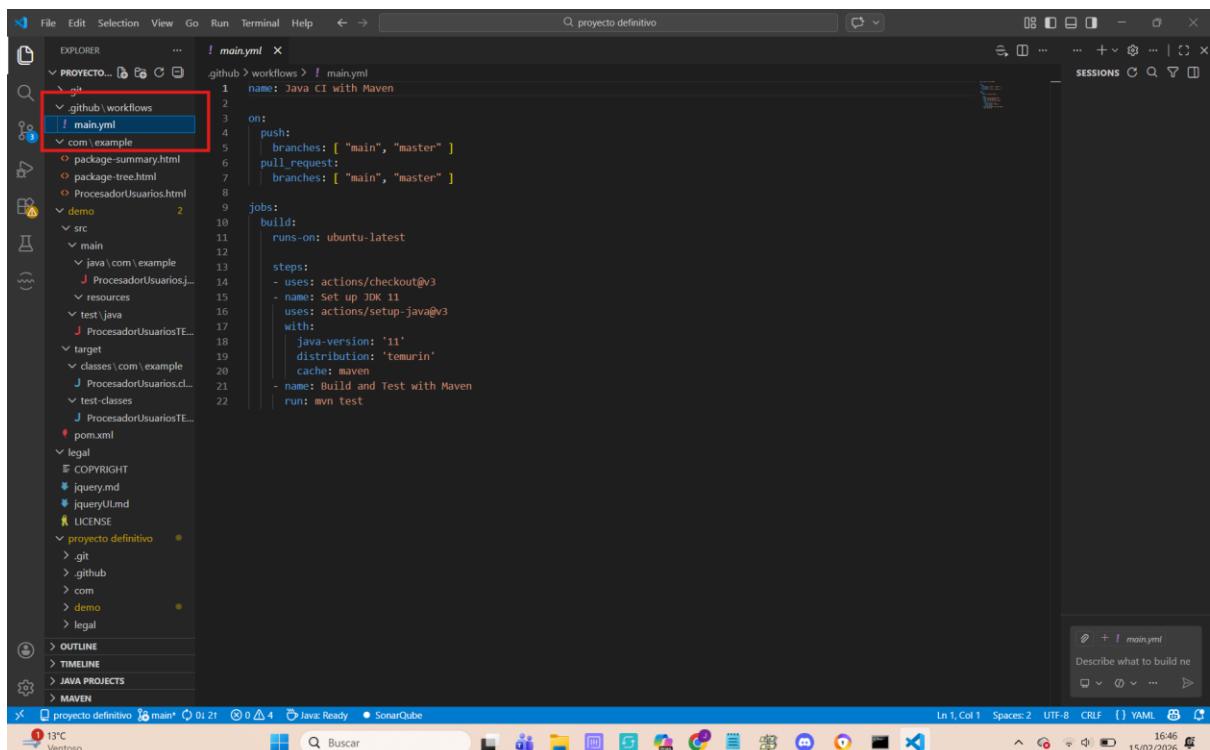
The screenshot shows the SonarQube interface with the following details:

- File Explorer:** On the left, it lists the project structure under "PROYECTO...". The file "ProcesadorUsuarios.html" is selected and highlighted with a red box.
- Code View:** The main area displays the content of "ProcesadorUsuarios.html". The code includes declarations for "viewport", "http-equiv", "dc.created", "generator", and various CSS links and script imports for jQuery and jQuery UI.
- Sessions:** A sidebar on the right titled "SESSIONS" shows a single session named "ProcesadorUsuarios".
- Bottom Navigation:** The footer includes tabs for "main", "JAVA PROJECTS", "MAVEN", and "OUTLINE". It also shows the status "Java Ready" and system information like "CPU 11%", "Memory 4%", and "Disk 15/16377 GB".

PASO 20 — Commit documentación hacemos un commit de la documentacion para que haya constancia



PASO 21 — Activar GitHub Actions entramos en github lo activamos y hacemos en archivo yaml



PASO 22 — Verificar que tenga: se comprueba en el archivo que corra el test con maven

The screenshot shows the GitHub Actions configuration file (`main.yml`) in the GitHub Actions tab of the VS Code Explorer. The file defines a workflow named "Java CI with Maven" that runs on pushes to the "main" and "master" branches. It uses an Ubuntu-latest runner and includes steps for checkout, setting up JDK 11, and running Maven tests. A red box highlights the step where Maven is run.

```
name: Java CI with Maven
on:
  push:
    branches: [ "main", "master" ]
  pull_request:
    branches: [ "main", "master" ]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Set up JDK 11
        uses: actions/setup-jdk@v3
        with:
          java-version: '11'
          distribution: 'temurin'
      - name: Build and Test with Maven
        run: mvn test
```