





# Análisis de Plataforma

## Estado Actual

Evaluación completa del estado actual de la plataforma, identificación de errores críticos en la lógica de código y implementación de pruebas automatizadas para garantizar la calidad del software



# Problemas Identificados



## Error

### Código

El método actualizarPeso resta 1 kg en lugar de usar el nuevo peso proporcionado por el usuario



## Impacto Usuario

Los usuarios visualizan pesos incorrectos generando resultados de salud erróneos y pérdida de confianza



## Falta

### Validación

Ausencia completa de pruebas automatizadas, JUnit, tests UI y procesos de CI/CD implementados

The background is a solid dark blue. It features two concentric circles in a lighter shade of blue, centered on the page. There are also several small, light blue dots scattered across the background, including one near the top left, one near the top right, one near the bottom right, and one near the bottom left.

# 69 kg

## **Peso Incorrecto**

Cuando el usuario ingresa 70 kg, el sistema muestra 69 kg debido al error en la lógica de actualización

# Solución Implementada



## Pruebas Unitarias

Implementación de JUnit para validar la lógica de actualización de peso correctamente



## Pruebas Funcionales

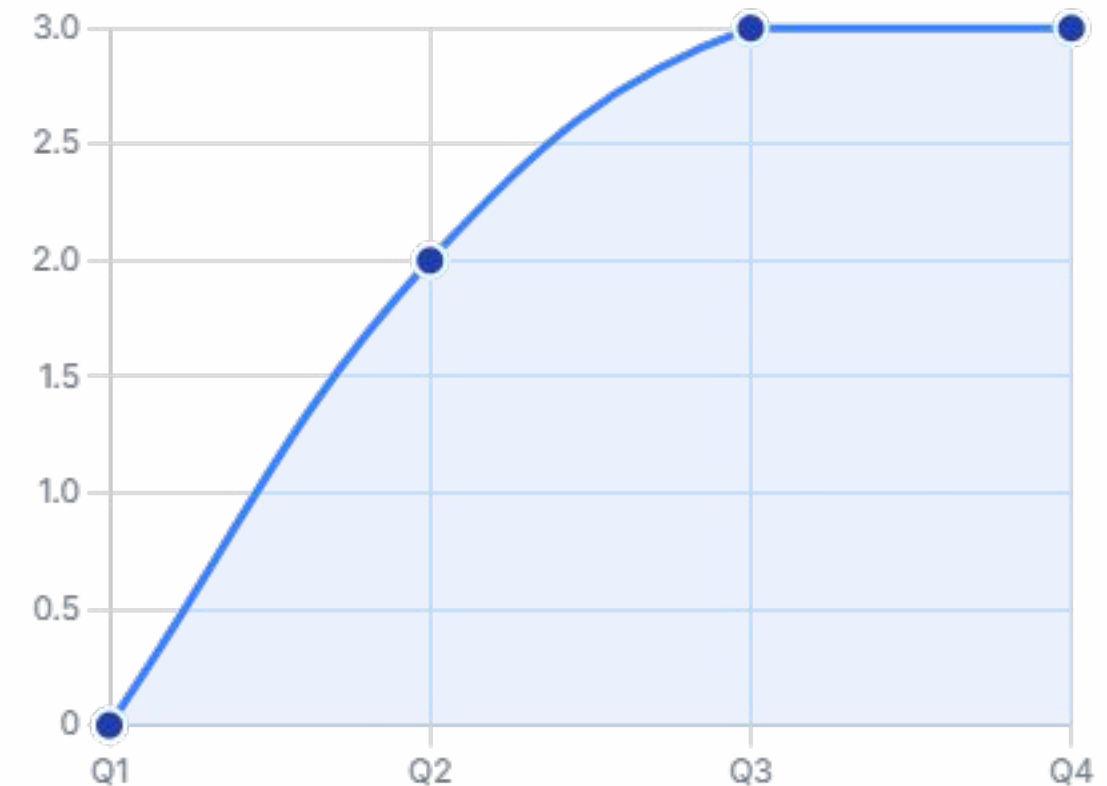
Selenium para simular el flujo completo del usuario en la aplicación



## Automatización CI/CD

GitHub Actions para ejecutar pruebas automáticamente en cada push y merge request

## Mejora Progresiva



# Pruebas JUnit

## Testing Unitario

Se implementaron pruebas unitarias con JUnit para validar el correcto funcionamiento del método actualizarPeso, detectando y corrigiendo el error que restaba 1kg al peso ingresado



## Características Implementadas

- ✓ Test testActualizarPesoCorrectamente
- ✓ Test mostrarInformacion
- ✓ Validación peso correcto
- ✓ Dependencias Maven configuradas
- ✓ Ejecución automatizada tests

2

Tests Creados

100%

Tests Pasando

0

Errores Detectados

# En Código

Test

```
src > test > java > UsuarioTest.java
You, 2 hours ago | 1 author (You)
1 import org.junit.jupiter.api.Test;
2 import static org.junit.jupiter.api.Assertions.*;
3
4 class UsuarioTest {
5
6     @Test
7     void testActualizarPesoCorrectamente() {
8         Usuario user = new Usuario("César", 70.0);
9         user.actualizarPeso(72.5);
10        assertEquals(72.5, user.getPeso());
11    }
12
13    @Test
14    void testMostrarInformacion() {
15        Usuario user = new Usuario("César", 70.0);
16        assertDoesNotThrow(user::mostrarInformacion);
17    }
18 }
19
```

Falla primera prueba

```
[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running UsuarioTest
Usuario: César, Peso Actual: 70.0 kg
[ERROR] Tests run: 2, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 0.079 s <<< FAILURE! -- in UsuarioTest
[ERROR] UsuarioTest.testActualizarPesoCorrectamente -- Time elapsed: 0.016 s <<< FAILURE!
org.opentest4j.AssertionFailedError: expected: <72.5> but was: <69.0>
    at org.junit.jupiter.api.AssertionFailureBuilder.build(AssertionFailureBuilder.java:151)
    at org.junit.jupiter.api.AssertionFailureBuilder.buildAndThrow(AssertionFailureBuilder.java:132)
    at org.junit.jupiter.api.AssertEquals.failNotEqual(AssertEquals.java:197)
    at org.junit.jupiter.api.AssertEquals.assertEquals(AssertEquals.java:70)
    at org.junit.jupiter.api.AssertEquals.assertEquals(AssertEquals.java:65)
    at org.junit.jupiter.api.Assertions.assertEquals(Assertions.java:885)
    at UsuarioTest.testActualizarPesoCorrectamente(UsuarioTest.java:10)
    at java.base/java.lang.reflect.Method.invoke(Method.java:569)
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)

[INFO] Results:
[INFO]
[ERROR] Failures:
[ERROR]   UsuarioTest.testActualizarPesoCorrectamente:10 expected: <72.5> but was: <69.0>
[INFO]
[ERROR] Tests run: 2, Failures: 1, Errors: 0, Skipped: 0
[INFO]
[INFO] BUILD FAILURE
[INFO]
[INFO] Total time: 2.314 s
[INFO] Finished at: 2025-07-28T18:00:18-04:00
```

Pasa 100%

```
[INFO] Running UsuarioTest
Usuario: César, Peso Actual: 70.0 kg
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.055 s -- in UsuarioTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 2.178 s
[INFO] Finished at: 2025-07-28T18:02:39-04:00
[INFO]
```

Se corrige el error

```
public void actualizarPeso(double nuevoPeso) {
    // ERROR: En lugar de asignar el nuevo peso, se está restando 1kg.
    this.peso = nuevoPeso;
}
```

# Pruebas Selenium



## 1 Flujo Usuario Completo

Se desarrolló una página HTML simple para simular la interfaz de usuario y ejecutar pruebas automatizadas con Selenium que validen el comportamiento completo del sistema

## 2 Automatización Navegador

Selenium WebDriver automatiza las interacciones del usuario en Chrome, verificando que los valores ingresados se procesen correctamente y se muestren sin errores en la interfaz

# En Código

## Test selenium

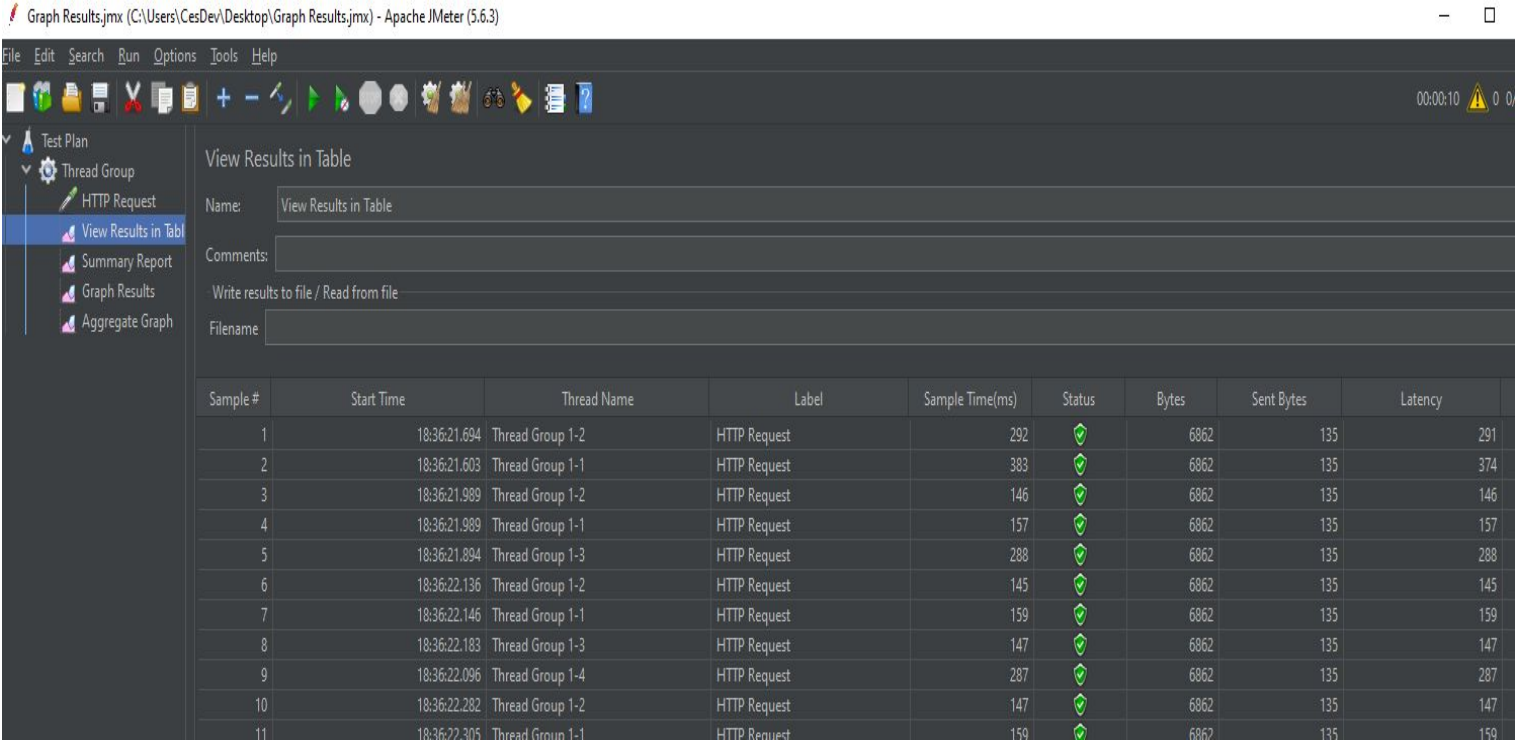
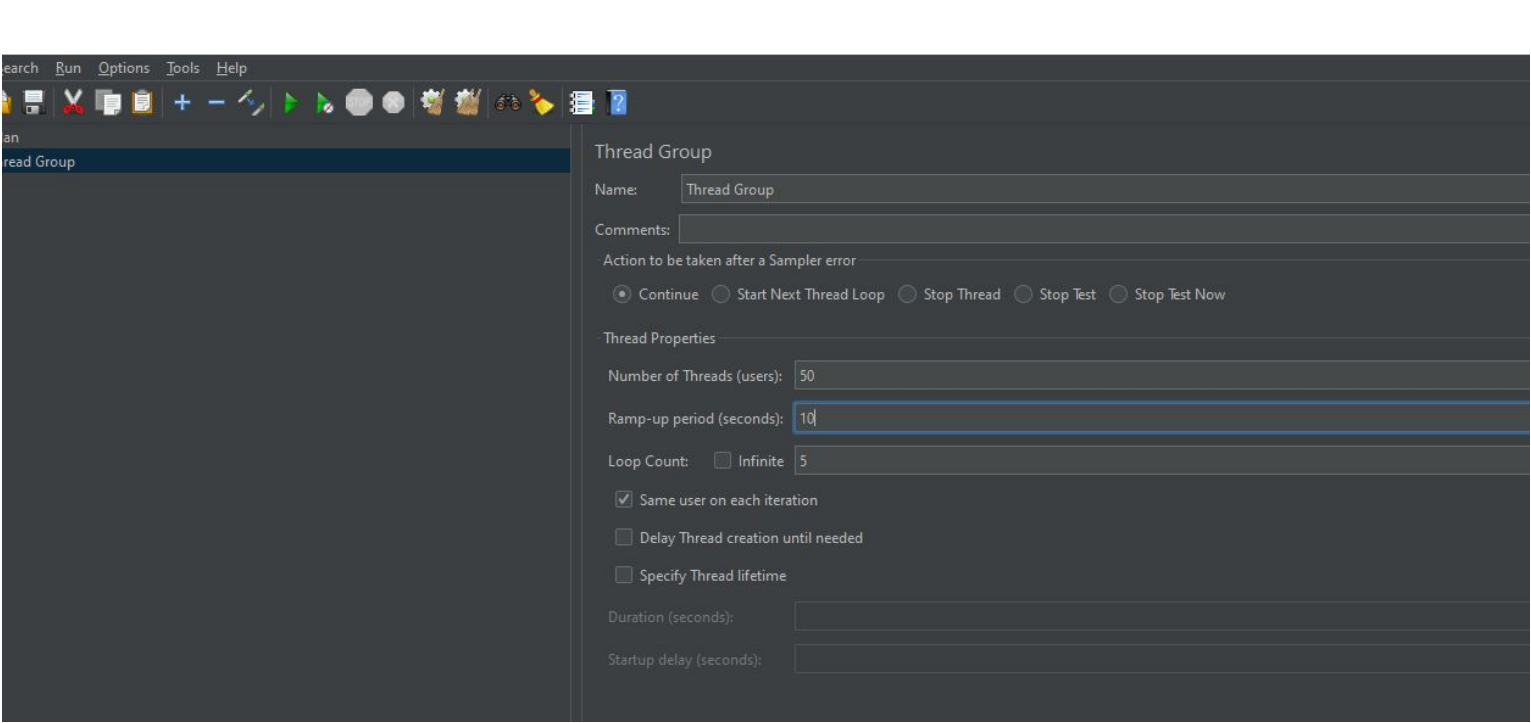
```
SeleniumTest.java selenium SeleniumTest.java .../java X Usuario.java index.html
src > test > java > SeleniumTest.java
1  import org.junit.jupiter.api.Test;
2  import static org.junit.jupiter.api.Assertions.assertEquals;
3  import org.openqa.selenium.By;
4  import org.openqa.selenium.WebDriver;
5  import org.openqa.selenium.chrome.ChromeDriver;
6
7  public class SeleniumTest {
8
9      @Test
10     public void testFlujoActualizacionPeso() {
11         // Opcional: define el path si chromedriver no está en PATH
12         // System.setProperty("webdriver.chrome.driver", "/ruta/a/chromedriver");
13
14         WebDriver driver = new ChromeDriver();
15         driver.get("http://localhost:8080");
16
17         driver.findElement(By.id("peso")).sendKeys("75.0");
18         driver.findElement(By.id("btn-actualizar")).click();
19
20         String pesoActual = driver.findElement(By.id("peso-actual")).getText();
21         assertEquals("75.0", pesoActual);
22
23         driver.quit();
24     }
25 }
26
```

```
[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running UsuarioTest
Usuario: César, Peso Actual: 70.0 kg
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.056 s -- in UsuarioTest
[INFO] Running SeleniumTest
Jul 28, 2025 6:06:31 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find CDP implementation matching 138
Jul 28, 2025 6:06:31 PM org.openqa.selenium.chromium.ChromiumDriver lambda$new$5
WARNING: Unable to find version of CDP to use for 138.0.7204.168. You may need to include a dependency on a specific version of the CDP using something similar to
`org.seleniumhq.selenium:selenium-devtools-v86:4.21.0` where the version ("v86") matches the version of the chromium-based browser you're using and the version
number of the artifact is the same as Selenium's.
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 3.028 s -- in SeleniumTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 5.575 s
[INFO] Finished at: 2025-07-28T18:06:32-04:00
[INFO]
[INFO] -----
```



# Pruebas de rendimiento

Configuración posible para JMETER, esto simulará 50 usuarios enviando solicitudes, repitiendo el test 5 veces.



# Pipeline CI/CD

## Setup Ambiente

Configuración de Java, Python y Chrome para ejecutar todas las pruebas automatizadas



## Ejecución Tests

Ejecución paralela de pruebas unitarias JUnit y pruebas funcionales con Selenium

## Análisis Código

Validación automática de calidad del código utilizando SonarQube para detectar code smells



## Reporte Resultados

Generación automática de reportes y notificaciones de fallas vía email

# En Código

Se crea un archivo yml con la configuración necesaria para las automatizaciones y se comenta en código cada paso, para mayor claridad de lo que está sucediendo.

```
.github > workflows > test.yml
You, 1 minute ago | 1 author (You)
1 | # Nombre del flujo de trabajo
2 | name: Run Tests and Analyze Code
3 |
4 | # Cuando se ejecuta este flujo de trabajo:
5 | on:
6 |   # Cuando hay un push (subida de código) a la rama main
7 |   push:
8 |     branches: [ main ]
9 |   # Cuando se crea un pull request hacia la rama main
10 |   pull_request:
11 |     branches: [ main ]
12 |
13 | jobs:
14 |   # Definición del trabajo que ejecutará este flujo
15 |   build-test-analyze:
16 |     # Define el sistema operativo del runner
17 |     runs-on: ubuntu-latest
18 |
19 |     steps:
20 |       # Paso 1: Clonar el repositorio en el runner
21 |       - name: Checkout code
22 |         uses: actions/checkout@v3
23 |
24 |       # Paso 2: Configurar Python en el runner
25 |       - name: Set up Python
26 |         uses: actions/setup-python@v4
27 |         with:
28 |           python-version: '3.x'
29 |
30 |       # Paso 3: Levantar un servidor HTTP para servir archivos estáticos (carpeta "resources")
31 |       - name: Levantar servidor HTTP estático
32 |         run: |
33 |           cd resources
34 |           python3 -m http.server 8080 &
35 |           sleep 5
36 |
37 |       # Paso 4: Configurar Java (versión 17)
38 |       - name: Set up Java
39 |         uses: actions/setup-java@v3
40 |         with:
41 |           distribution: 'temurin' # Usa la distribución Temurin
42 |           java-version: '17'     # Usa Java 17
43 |
44 |       # Paso 5: Cachear archivos de SonarQube (para acelerar análisis)
45 |       - name: Cache SonarQube packages
46 |         uses: actions/cache@v4
47 |         with:
48 |           path: ~/.sonar/cache
49 |           key: ${runner.os}-sonar
50 |           restore-keys: ${runner.os}-sonar
51 |
52 |       # Paso 6: Cachear dependencias de Maven
53 |       - name: Cache Maven packages
54 |         uses: actions/cache@v4
55 |         with:
56 |           path: ~/.m2
57 |           key: ${runner.os}-m2-${hashFiles('**/pom.xml')}
58 |           restore-keys: ${runner.os}-m2
59 |
```

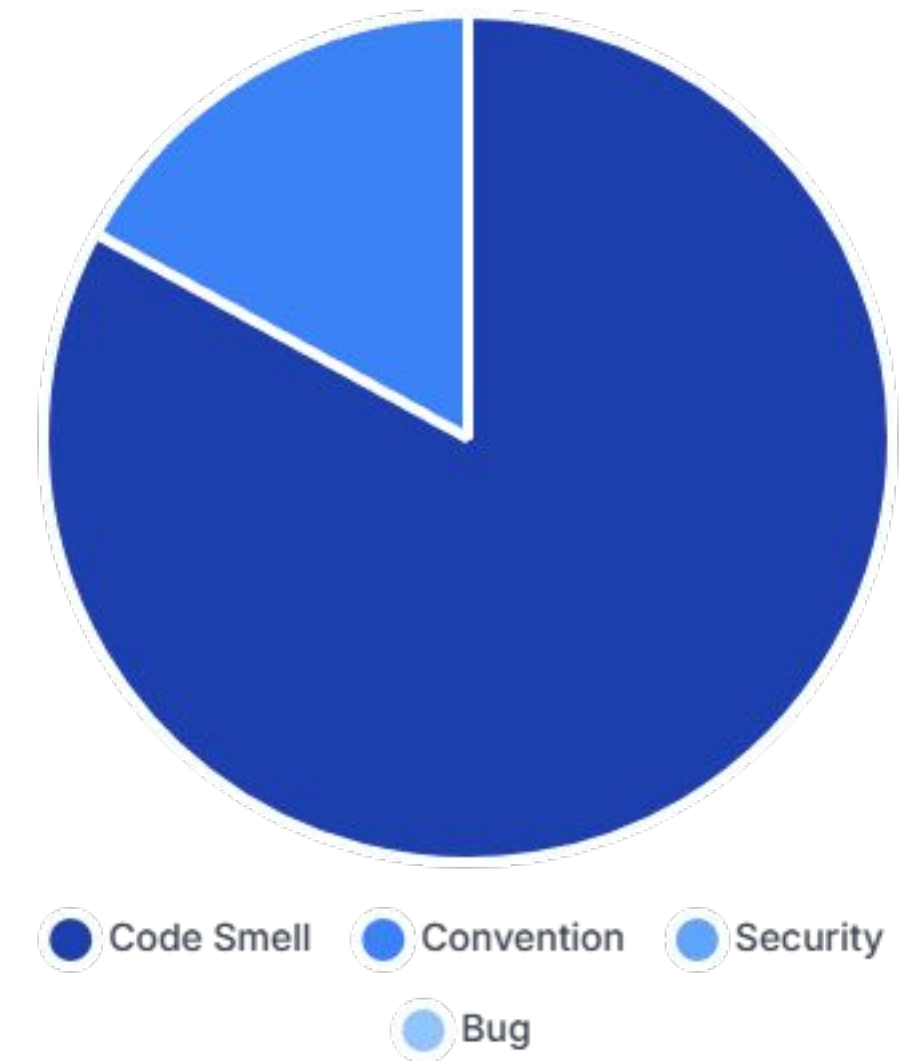
```
60 |
61 |   # Paso 7: Compilar y analizar
62 |   - name: Build and analyze
63 |     env:
64 |       SONAR_TOKEN: ${secrets.SONAR_TOKEN} # Se toma desde GitHub Secrets
65 |     run: |
66 |       mvn -B verify \
67 |         org.sonarsource.scanner.maven:sonar-maven-plugin:sonar \
68 |         -Dsonar.projectKey=CesarHerr_Ev_devOps_Adalid
69 |
70 |   # Paso 8: Instalar Chrome (requerido para Selenium)
71 |   - name: Set up Chrome
72 |     uses: browser-actions/setup-chrome@v1
73 |
74 |   # Paso 9: Mostrar versión de Chrome instalada
75 |   - name: Show Chrome version
76 |     run: chrome --version
77 |
78 |   # Paso 10: Obtener la versión de Chrome instalada (para usarla en el siguiente paso)
79 |   - name: Get Chrome version
80 |     id: get-chrome-version
81 |     run: |
82 |       CHROME_VERSION=$(google-chrome --version | grep -oP '\d+\.\d+\.\d+' || true)
83 |       echo "version=$CHROME_VERSION" >> $GITHUB_OUTPUT
84 |
85 |   # Paso 11: Instalar ChromeDriver (versión debe coincidir con Chrome)
86 |   - name: Set up ChromeDriver
87 |     uses: nanasess/setup-chromedriver@v2
88 |     with:
89 |       chromedriver-version: ${steps.get-chrome-version.outputs.version}
90 |
91 |   # Paso 13: Ejecutar pruebas Selenium
92 |   - name: Run Selenium tests
93 |     run: mvn test -Pselenium
94 |
```

## Calidad Código

- Detección automática de code smells y malas prácticas
- Análisis de mantenibilidad del código fuente
- Identificación de vulnerabilidades de seguridad potenciales
- Métricas de duplicación y complejidad del código

SonarQube identificó 6 issues menores relacionados principalmente con convenciones de código y estructura de archivos que no afectan la funcionalidad

**El proyecto mantiene una calidad general alta con issues de baja prioridad fácilmente corregibles**



## Distribución Issues

Clasificación de problemas detectados por SonarQube en el análisis de calidad

# En Código

Se añade a en el CI/CD la ejecución de pruebas vía SonarQube, para ello se realizan los siguientes pasos:

- Se crea una cuenta en <https://sonarcloud.io>
- Se conecta el repositorio sobre el que se va a trabajar
- Se crea un Token
- Se crea proyecto nuevo
- Se añade el token en github
- Se configura yml y pom.xml según lo indicado en sonarcloud

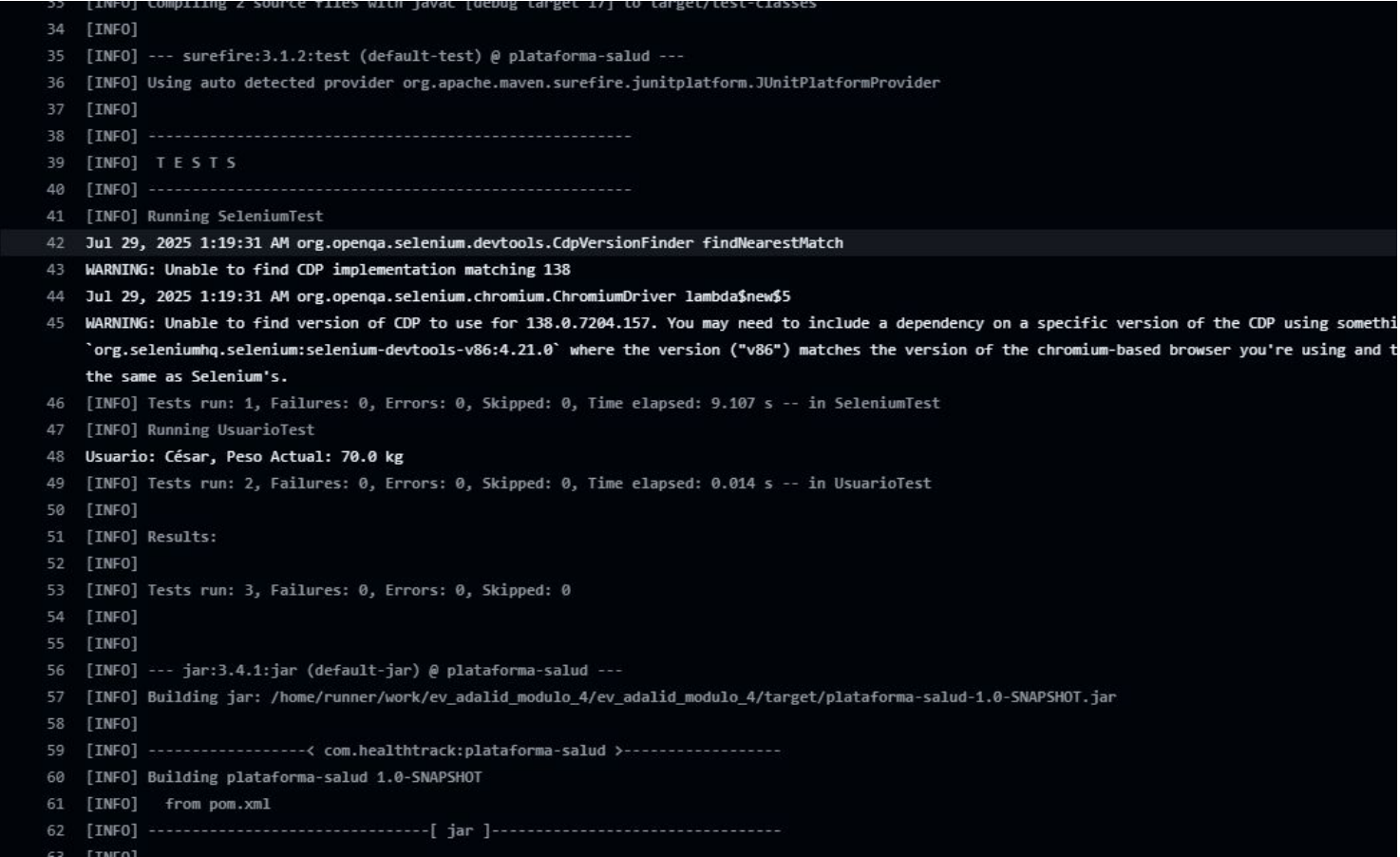
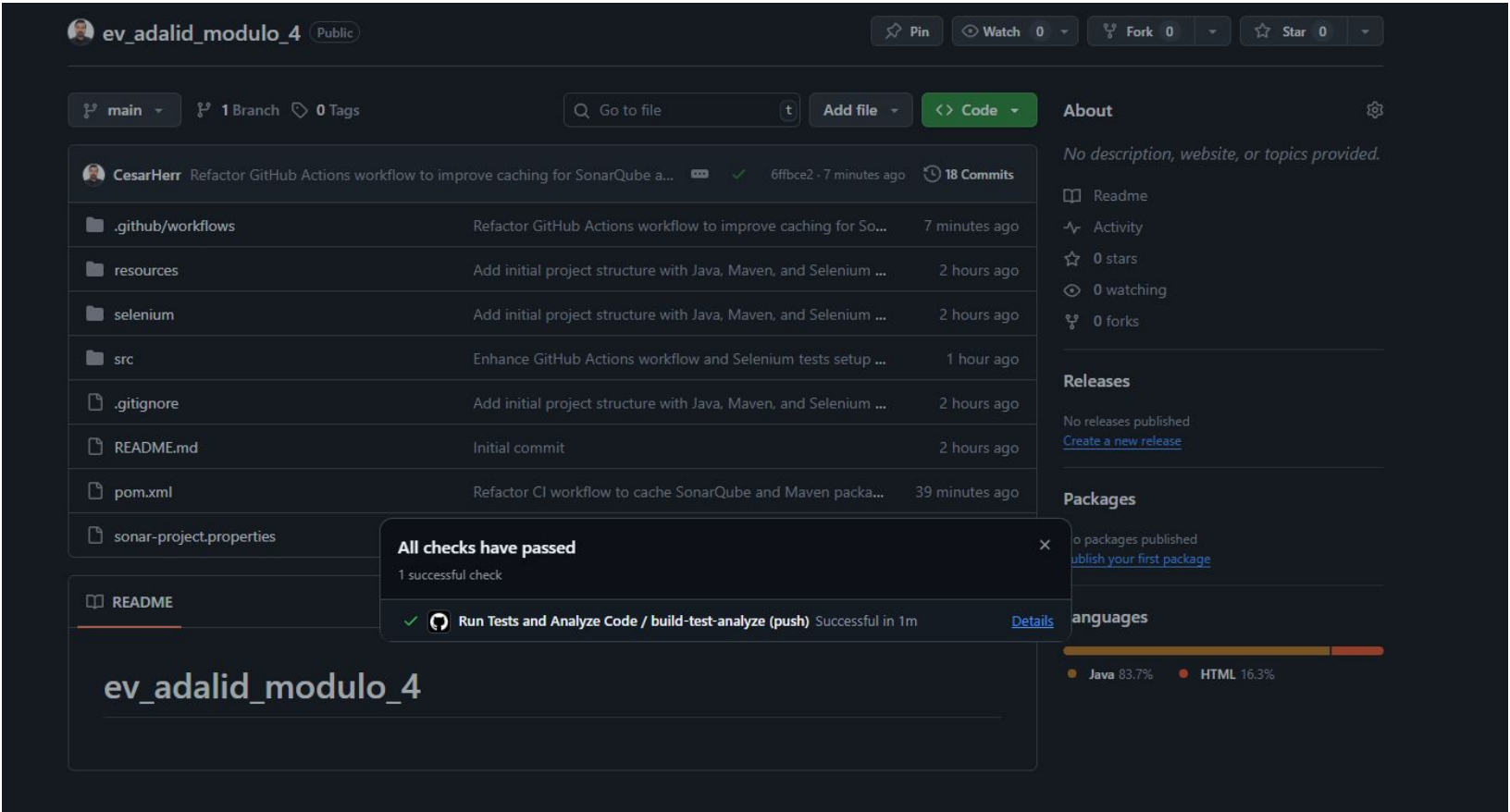
```
59 # Paso 7: Compilar y analizar
60 - name: Build and analyze
61   env:
62     SONAR_TOKEN: ${ secrets.SONAR_TOKEN }
63   run: mvn -B verify org.sonarsource.scanner.maven:sonar-maven-plugin:sonar -Dsonar.projectKey=CesarHerr_Ev_devOps_Adalid
64
65
```

```
1 <properties>
2   <maven.compiler.source>17</maven.compiler.source>
3   <maven.compiler.target>17</maven.compiler.target>
4   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
5   <junit.jupiter.version>5.10.0</junit.jupiter.version>
6   <selenium.version>4.21.0</selenium.version>
7   <sonar.organization>cesarherr</sonar.organization>
8   <sonar.host.url>https://sonarcloud.io</sonar.host.url>
9 </properties>
10 You, 2 hours ago • Add initial project structure with Java, Maven,...
11 <dependencies>
12   <!-- JUnit 5 -->
```



# Pruebas Generales de Ejecución Pipeline

Se realiza un commit, y se ejecuta el pipeline:



# Pruebas de Ejecución

← Run Tests and Analyze Code

✓

Refactor GitHub Actions workflow to improve caching for SonarQube and... #17

Summary

Jobs

✓ build-test-analyze

Run details

Usage

Workflow file

build-test-analyze

succeeded 7 minutes ago in 1m 4s

> ✓ Set up job

> ✓ Checkout code

> ✓ Set up Python

> ✓ Levantar servidor HTTP estático

> ✓ Set up Java

> ✓ Cache SonarQube packages

> ✓ Cache Maven packages

> ✓ Build and analyze

> ✓ Set up Chrome

> ✓ Show Chrome version

> ✓ Get Chrome version

> ✓ Set up ChromeDriver

> ✓ Run Selenium tests

> ✓ Post Cache Maven packages

> ✓ Post Cache SonarQube packages

> ✓ Post Set up Java

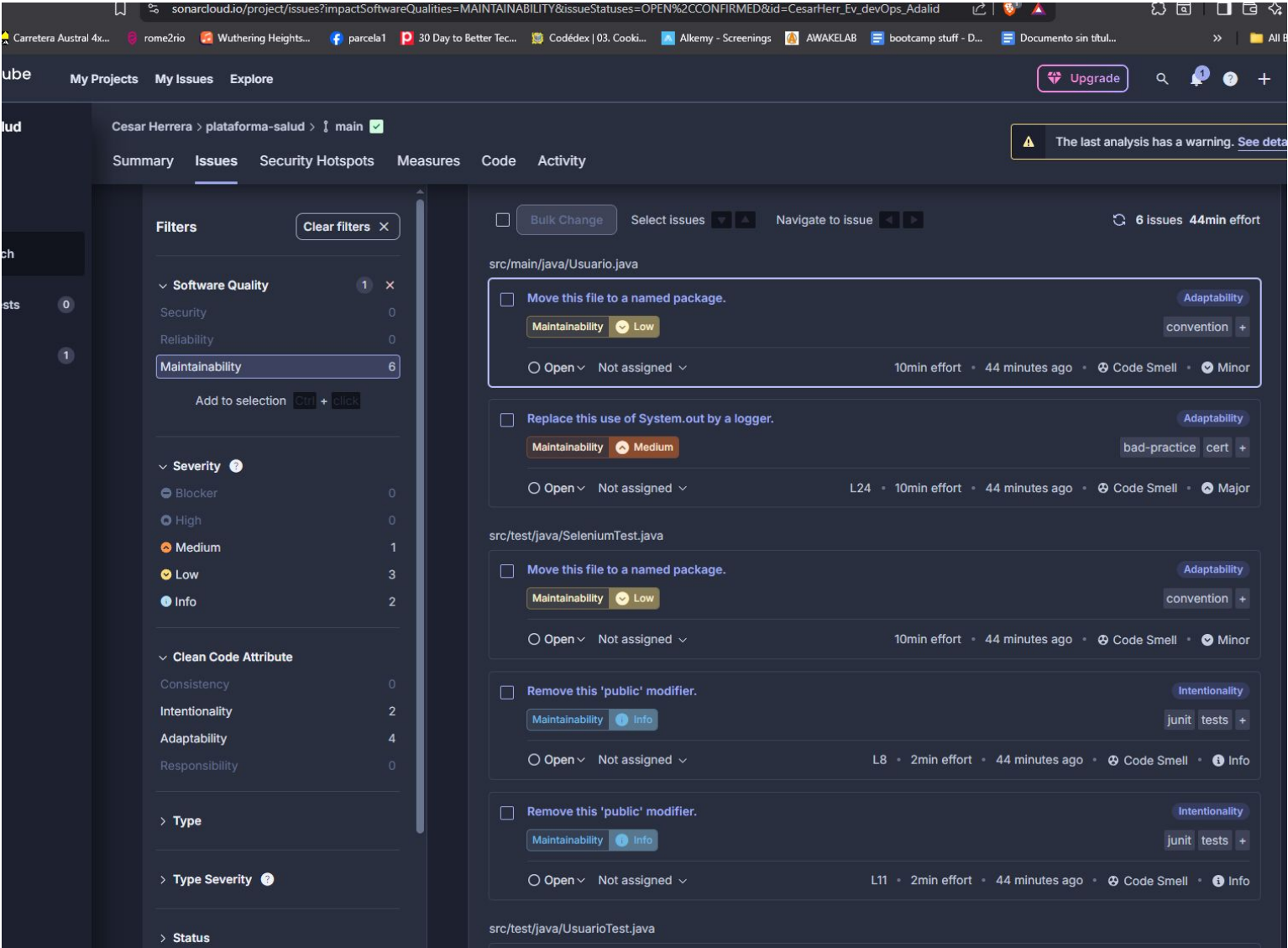
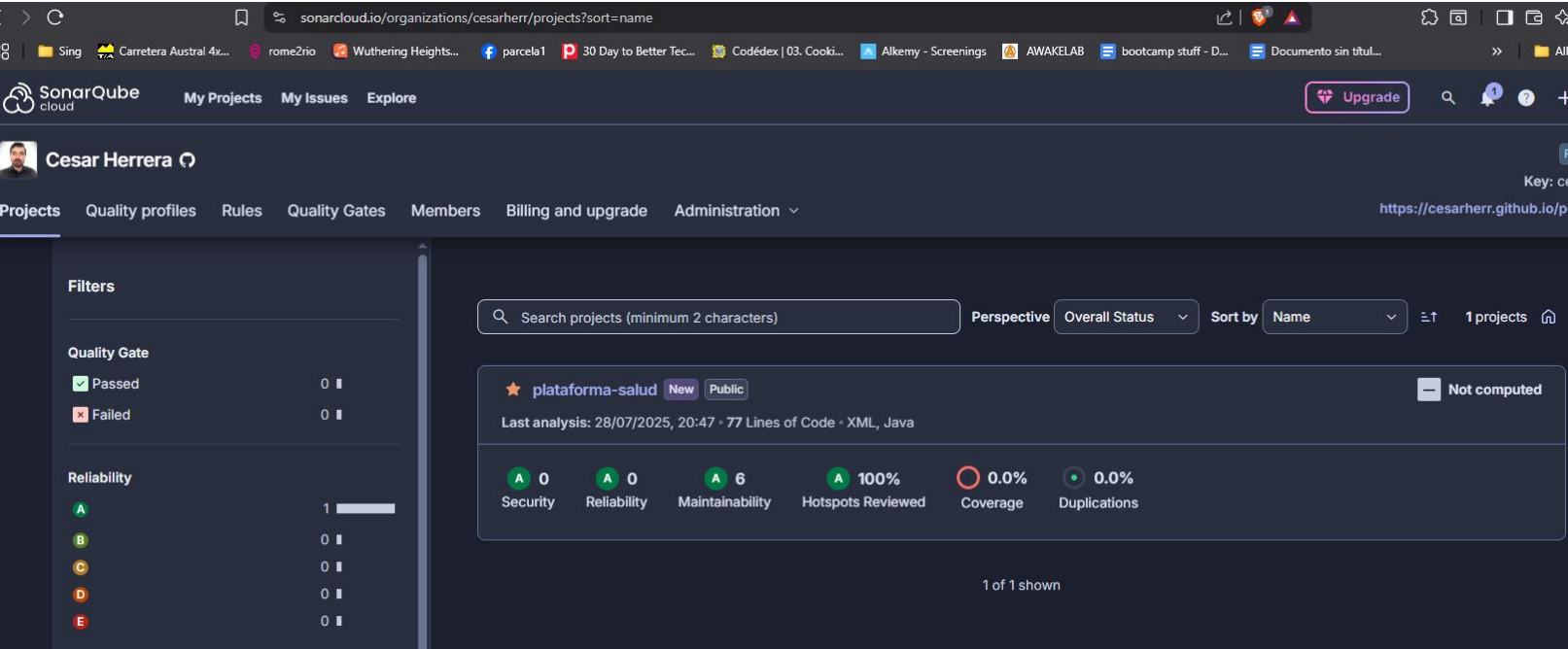
> ✓ Post Set up Python

> ✓ Post Checkout code

> ✓ Complete job

# Pruebas de Ejecución

SonarCloud







# Resultados Obtenidos

Resumen completo de los resultados obtenidos tras la implementación de pruebas automatizadas y procesos de validación continua en la plataforma

Tipo Prueba	Tests Creados	Tests Pasando	Errores Detectados	Tiempo Ejecución	Éxito	Estado
Pruebas Unitarias	2	2	0	0.014s	100 %	Exitoso
Pruebas Selenium	1	1	0	9.107s	100 %	Exitoso
Pipeline CI/CD	1	1	0	1m 4s	100 %	Activo
Análisis SonarQube	1	1	6	Auto	83%	Monitoreando
Total General	5	5	6	Auto	95%	Operativo

# Repositorio Github

[https://github.com/CesarHerr/ev\\_adalid\\_modulo\\_4](https://github.com/CesarHerr/ev_adalid_modulo_4)