Evaluación Módulo 4: Automatización de Pruebas en una Plataforma de Salud

🎯 Contexto:

La empresa HealthTrack ha desarrollado una plataforma web para el monitoreo del peso de los usuarios. La aplicación permite a los usuarios registrarse y actualizar su peso cada 48 horas. Sin embargo, la plataforma tiene un error crítico: cada vez que un usuario actualiza su peso, el sistema le resta automáticamente 1 kg en lugar de registrar el valor ingresado.

El problema surge porque no se han implementado pruebas unitarias, de integración, de regresión ni de rendimiento. Además, la empresa no tiene un pipeline de CI/CD que asegure la validación automática del código antes de su despliegue.

Como especialistas en automatización de pruebas, los estudiantes deberán evaluar el estado actual de la plataforma y proponer soluciones utilizando estrategias de pruebas unitarias, de integración, funcionales y de rendimiento. También deberán estructurar un pipeline de CI/CD para asegurar que estas pruebas se ejecuten automáticamente.

Código Base del Proyecto en Java

El siguiente código Java representa el error en la lógica del sistema. Actualmente, reduce 1 kg cada vez que un usuario actualiza su peso en lugar de registrar el valor ingresado.

```
public class Usuario {
    private String nombre;
    private double peso;

    public Usuario(String nombre, double peso) {
        this.nombre = nombre;
        this.peso = peso;
    }

    public String getNombre() {
        return nombre;
    }

    public double getPeso() {
        return peso;
    }

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

    public void mostrarInformacion() {
        System.out.println("Usuario: " + nombre + ", Peso Actual: " + peso + "
kg");
    }
}
```

🎯 **Desarrollo:**

- Analizar el problema en la plataforma de salud.

Al hacer revision del codigo nos encontramos con un problema básico en la estructura del código, en la cual el nuevo peso se deberia asignar directamente con el siguiente codigo:

```
public void actualizarPeso(double nuevoPeso) {
   this.peso = nuevoPeso; // Corrección implementada
}
```

Impacto: Esta falla genera **falsas tendencias de evolución en el peso del usuario**, lo que puede inducir a decisiones equivocadas relacionadas con su salud o incluso el abandono de la plataforma.

- Diseñar e implementar un plan de pruebas que contemple pruebas unitarias, funcionales, de regresión y de rendimiento.

Se recomienda implementar una batería de pruebas automatizadas para validar la funcionalidad, detectar fallas futuras y garantizar la calidad continua.

En pocas palabras utilizariamos el siguiente software:

Tipo de Prueba	Herramienta	Alcance Esperado
Unitarias	**JUnit**	Verificar lógica interna de métodos (`actualizarPeso`)
Funcionales (E2E)	**Selenium**	Simular interacción real del usuario: login, actualización de peso, etc.
Rendimiento	**JMeter**	Pruebas de carga concurrente sobre endpoints de actualización y login
Análisis de Calidad	**SonarQube**	Verifica duplicación, complejidad ciclomática, cobertura de pruebas

- Falta de procesos de validación y pruebas en el desarrollo actual.

Se recomienda agregar validaciones en el backend para prevenir el ingreso de valores negativos como peso:

```
if (nuevoPeso <= 0) {
   throw new IllegalArgumentException("El peso debe ser mayor a 0 kg.");
}
```

- Implementar reportes HTML de rendimiento con JMeter (`-e -o`) para visualizar métricas en cada prueba.

- Integrar SonarQube en el ciclo de desarrollo para evaluar métricas de calidad técnica con cada cambio.
- Crear un pipeline de CI/CD con GitHub Actions que ejecute todas las pruebas (JUnit, Selenium, JMeter) en cada commit a la rama principal.

Tipo de Prueba	Herramienta	Alcance Esperado
Unitarias	**JUnit**	Verificar lógica interna de métodos (`actualizarPeso`)
Funcionales (E2E)	**Selenium**	Simular interacción real del usuario: login, actualización de peso, etc.

```
[INFO] Changes detected - recompiling the module! :source
[INFO] Compiling 2 source files with javac [debug release 17] to target/test-classes
[INFO]
[INFO] --- surefire:3.2.5:test (default-test) @ healthtrack ---
[INFO] Using auto detected provider org.apache.maven.surefire.junitplatform.JUnitPlatformProvider
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running test.java.com.UsuarioTest
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.023 s -- in test.java.com.UsuarioTest
[INFO] Running com.healthtrack.test.FlujoUsuarioFirefoxTest
19:43:55.737 [main] INFO io.github.bonigarcia.wdm.WebDriverManager -- Using geckodriver 0.36.0 (resolved driver for Firefox 140)
19:43:55.765 [main] INFO io.github.bonigarcia.wdm.WebDriverManager -- Exporting webdriver.gecko.driver as /home/vantage/.cache/sele
odriver/linux64/0.36.0/geckodriver
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.928 s -- in com.healthtrack.test.FlujoUsuarioFirefoxTest
jul. 08, 2025 7:43:58 P. M. org.openqa.selenium.os.ExternalProcess$Builder lambda$start$0
ADVERTENCIA: failed to copy the output of process 189672
java.io.IOException: Stream closed
        at java.base/java.io.BufferedInputStream.getBufIfOpen(BufferedInputStream.java:168)
        at java.base/java.io.BufferedInputStream.read(BufferedInputStream.java:334)
        at java.base/java.io.InputStream.transferTo(InputStream.java:782)
        at org.openqa.selenium.os.ExternalProcess$Builder.lambda$start$0(ExternalProcess.java:209)
        at java.base/java.lang.Thread.run(Thread.java:840)

[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  4.944 s
[INFO] Finished at: 2025-07-08T19:43:58-04:00
[INFO] ------------------------------------------------------------------------
➜  healthtrack git:(main)
```

Rendimiento	**JMeter**	Pruebas de carga concurrente sobre endpoints de actualización y login

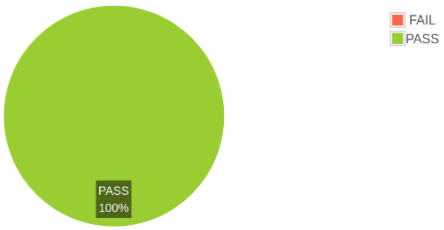
Test and Report information

Source file	"resultados.jtl"
Start Time	"7/8/25, 7:37 PM"
End Time	"7/8/25, 7:37 PM"
Filter for display	""

APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
1.000	**500 ms**	**1 sec 500 ms**	**Total**
1.000	500 ms	1 sec 500 ms	HTTP Request-1
1.000	500 ms	1 sec 500 ms	HTTP Request-0
1.000	500 ms	1 sec 500 ms	HTTP Request

Requests Summary



Análisis de Calidad **SonarQube**