

## Análisis de la calidad del producto

### 1. INTRODUCCIÓN

En este informe se va a analizar la calidad del producto desarrollado durante la primera semana del segundo sprint del proyecto integrado. Se realiza a fecha de 3 de noviembre de 2024, habiéndose desarrollado una de las dos interfaces nuevas que se requerían (Registrar Descuento En Marca), así como la implementación del código fuente de Registrar Descuento casi en su totalidad. Por lo que se tiene material suficiente para analizar lo que se ha realizado e introducir cambios antes del fin del sprint.

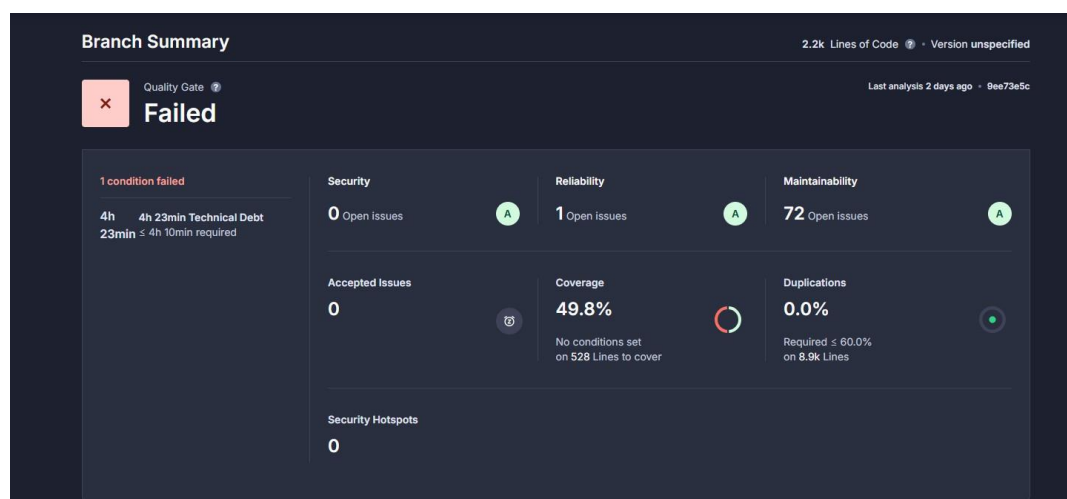
### 2. ANÁLISIS DE LA CALIDAD

En la *captura 2.1* podemos observar el resumen del análisis realizado por la herramienta SonarQube, vemos que no hay issues de Security, solo uno de Reliability y 72 de Maintainability. Además, no hay duplicaciones de código, pero la cobertura es inferior al 50%.

Como observamos el Quality Gate no está en Passed, esto es debido a que tenemos más de 4h y 10 minutos (máximo establecido) de deuda técnica.

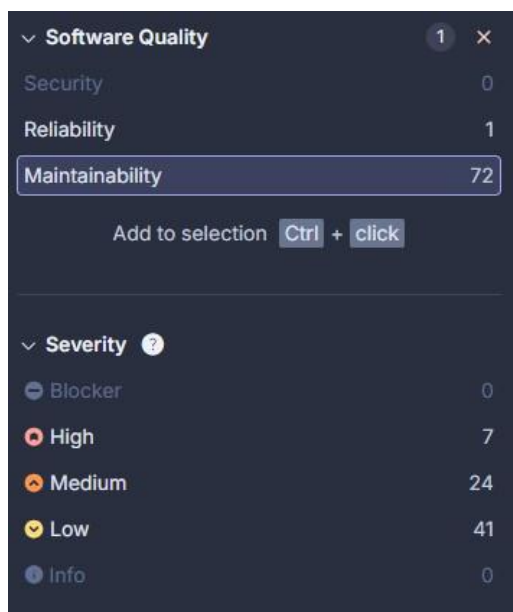
En cuanto a los issues detectados vemos que en su mayoría son fallos de mantenibilidad del código.

Para corregir completamente el proyecto deberíamos emplear 4 horas y 23 min.



*Captura 2.1 Imagen inicial del análisis*

En las capturas 2.2 y 2.3 que podemos observar debajo, se nos muestra el detalle de la severidad de los fallos de cada tipo. Aunque en Maintainability hay bastantes issues, solo 7 son de severidad alta por lo que el resto no serán muy difíciles de solventar ni nos llevarán demasiado tiempo. Vemos también que la mayoría son de severidad baja. En el caso de Reliability el único issue es de severidad media. Con esto podemos deducir que, aunque hay fallos de calidad no son excesivamente graves por lo que se pueden solventar y reconducir la situación.



This screenshot shows the 'Software Quality' panel with 'Maintainability' selected. It lists 72 issues. Below, the 'Severity' breakdown shows 0 Blocker, 7 High, 24 Medium, 41 Low, and 0 Info issues.

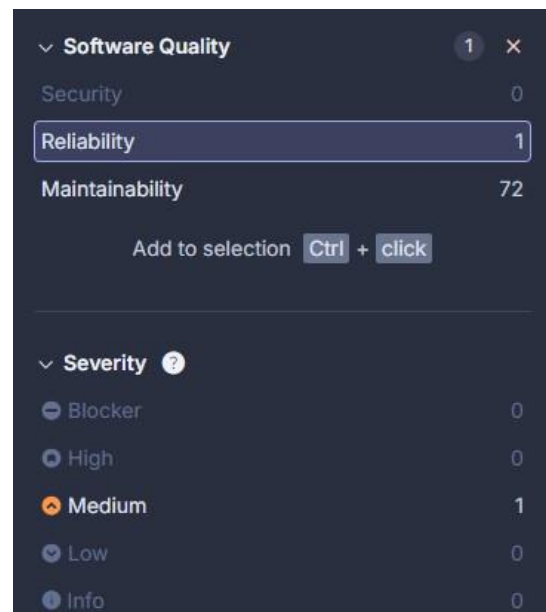
Category	Count
Security	0
Reliability	1
Maintainability	72

Add to selection **Ctrl** + **click**

Severity	Count
Blocker	0
High	7
Medium	24
Low	41
Info	0

*Captura 2.2 Detalle fallos Mantenibilidad*



This screenshot shows the 'Software Quality' panel with 'Reliability' selected. It lists 1 issue. Below, the 'Severity' breakdown shows 0 Blocker, 0 High, 1 Medium, 0 Low, and 0 Info issues.

Category	Count
Security	0
Reliability	1
Maintainability	72

Add to selection **Ctrl** + **click**

Severity	Count
Blocker	0
High	0
Medium	1
Low	0
Info	0

*Captura 2.3 Detalle fallos Reliability*

En el caso de Maintainability algunos de los fallos más graves son:

- 1- Add a nested comment explaining why this method is empty, throw an UnsupportedOperationException or complete the implementation.: esto es debido a que existen metodos vacios y esta situacion acarrea que estos métodos no tengan funcionalidad alguna, de momento.
- 2- This class is part of 3 cycles containing 4 classes: nos dice que hay un ciclo entre el MainView y RegistrarDescuento .

De los de severidad media los que más se repiten son:

- 1- Eliminar bloques o líneas de código comentadas: tenemos varias líneas comentadas ya que estamos probando cosas, además de alguna cosa que en el código proporcionado inicialmente ya estaba comentado.
- 2- Make this anonymous inner class lambda.

De los de low:

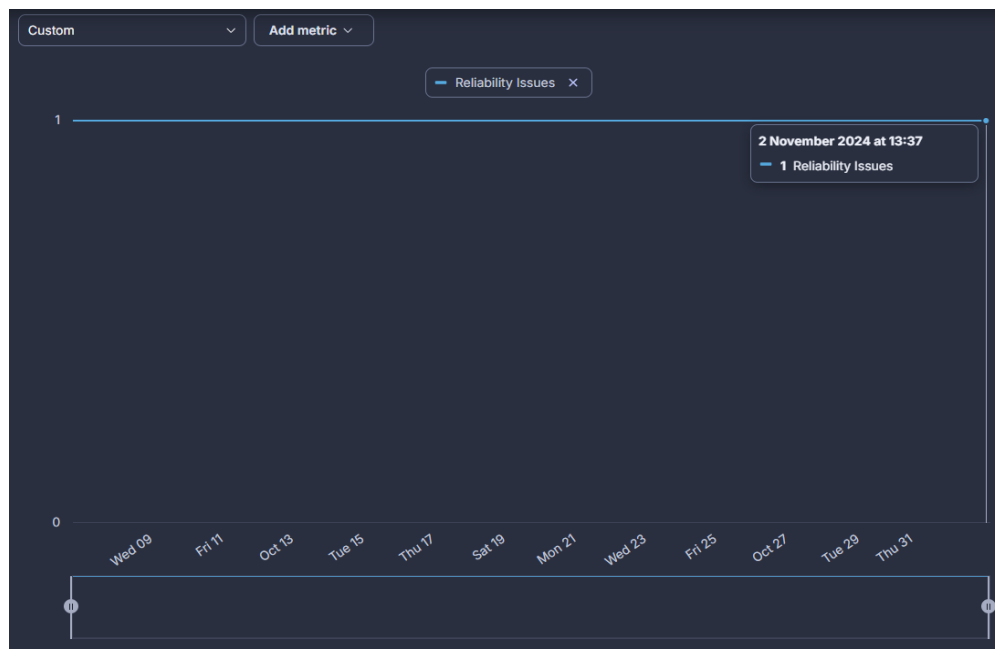
- 1- Eliminar import que no se están utilizando: en varias clases hay imports que no se están utilizando por lo que son innecesarios y deberían ser eliminados.

El fallo de Reliability es “Remove this field injection and use constructor injection instead”: es una mejora que nos propone en la clase MainView que no ha sido desarrollada por nuestro equipo, pero nos dice que podríamos eliminar la directiva de inject y hacer un constructor inyectivo en su lugar.

En las siguientes capturas 2.4 y 2.5 vemos la evolución de los diferentes issues a lo largo del desarrollo del proyecto. Se puede observar desde el sprint pasado ya existían varios fallos de mantenibilidad, pero estos han ido aumentando a lo largo de este sprint ya que se ha ido desarrollando más código, en cierto modo es normal que aumenten ya que es prácticamente imposible desarrollar código nuevo perfecto. El resto de issues de momento se mantienen.



Captura 2.4 Evolución security y maintainability issues



Captura 2.5 Evolución issues reliability

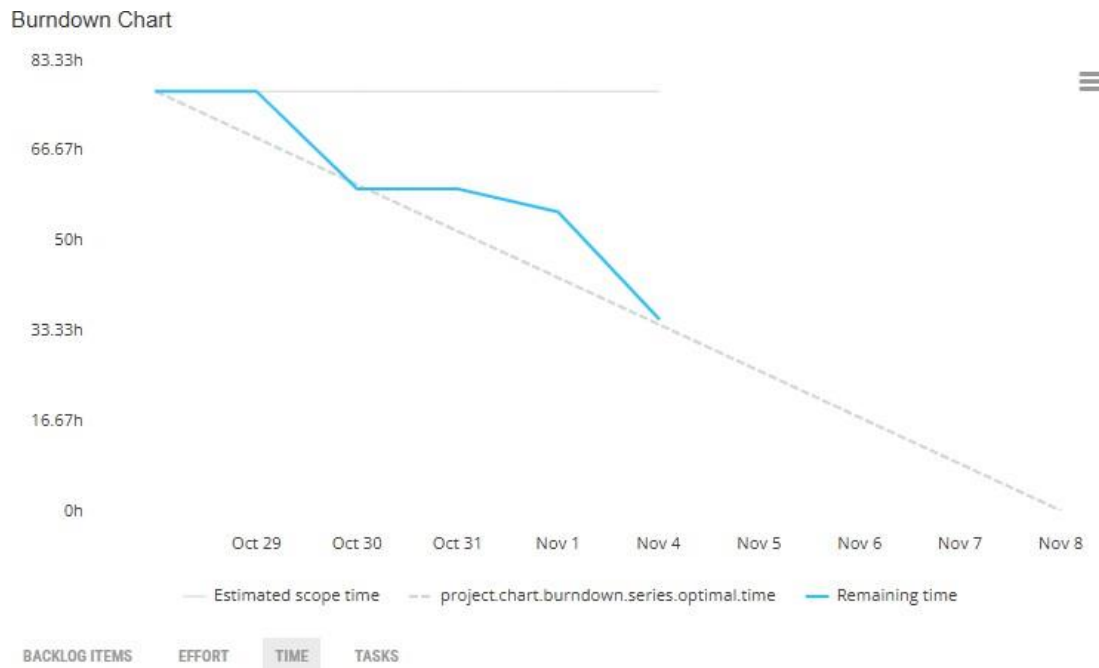
Finalmente, en la captura 2.6, observamos que el código mantiene una ausencia de duplicación de líneas, siendo un dato positivo. Sin embargo, la complejidad ciclomática ha aumentado debido a la incorporación de nuevas clases y métodos, lo cual era esperado. Aunque un aumento moderado en la complejidad es normal durante el desarrollo, es importante monitorear este valor para evitar que el código se vuelva más difícil de mantener y probar en futuras iteraciones.



Captura 2.6 Evolución líneas duplicadas y complejidad ciclomática

### 3. EVOLUCIÓN CARGA DE TRABAJO

En el gráfico del Sprint Burndown Chart (Captura 3.1) podemos observar cómo ha evolucionado la cantidad de trabajo pendiente a lo largo del sprint. La línea azul representa el progreso real, mientras que la línea gris discontinua muestra la trayectoria ideal. A medida que avanza el sprint, vemos que el equipo está logrando reducir las tareas de manera constante y se mantiene bastante cerca de la línea ideal. Esto indica que, se espera finalizar todas las tareas a tiempo. Cabe comentar que, aunque el progreso es adecuado, es importante prestar atención a los últimos días del sprint, ya que cualquier incremento en tareas no previstas o bloqueos podrían afectar la finalización. En resumen, el equipo está mantenido un buen ritmo de trabajo durante el sprint, y, de continuar con esta tendencia, se podrá cumplir con los objetivos planteados.



Captura 3.1 Sprint burndown chart

#### 4. PLAN DE ACCIÓN PARA LA MEJORA DEL CÓDIGO

Prioridad	Nombre	Severidad	Localización	Effort
1	Make this anonymous inner class a lambda	Medium	RegistrarDescuentoView, RegistrarView	40 minutos
2	Extract this nested try block into a separate method	Medium	RegistrarPresenter	20 minutos
3	Eliminar imports sin usar	Low	Varias clases	28 minutos
				<b>1h y 28 min</b>

No se ha decidido resolver los issues de severidad alta ya que estos saltan debido a que hay funciones vacías, pero estas están vacías ya que no están completadas todas las interfaces nuevas que tiene que tener la aplicación porque no está terminada la implementación, una vez estas estén finalizadas ya podemos implementar estas funciones correctamente y ya no saltará más este tipo de error, pero si queremos que de momento no salte podemos simplemente comentar que por ahora esa funcionalidad está vacía. En el caso del segundo issue de severidad alta no se va a resolver ya que no hay forma de hacerlo, debido a que para que la aplicación funcione correctamente es necesario las llamadas en ambas clases. Además, aunque si se ha incluido eliminar imports que no se usan, puede que parte de ellos sí que sean necesarios ya que por ejemplo las clases de prueba están aún sin implementar por lo que necesitarán varios de estos imports, entonces no será necesario eliminarlos y el effort se reducirá.

#### 5 IMPLEMENTACIÓN PLAN DE MEJORA

Para ejecutar el plan de acción diseñado, se incluirán dos nuevas tareas, una de 2h y la otra de 0h ya que serán realizada por la misma persona, pero debe haber una tarea por cada historia de usuario, el nombre de esta tarea será "Implementación del plan de acción".