Análisis de la calidad del producto

1. INTRODUCCIÓN

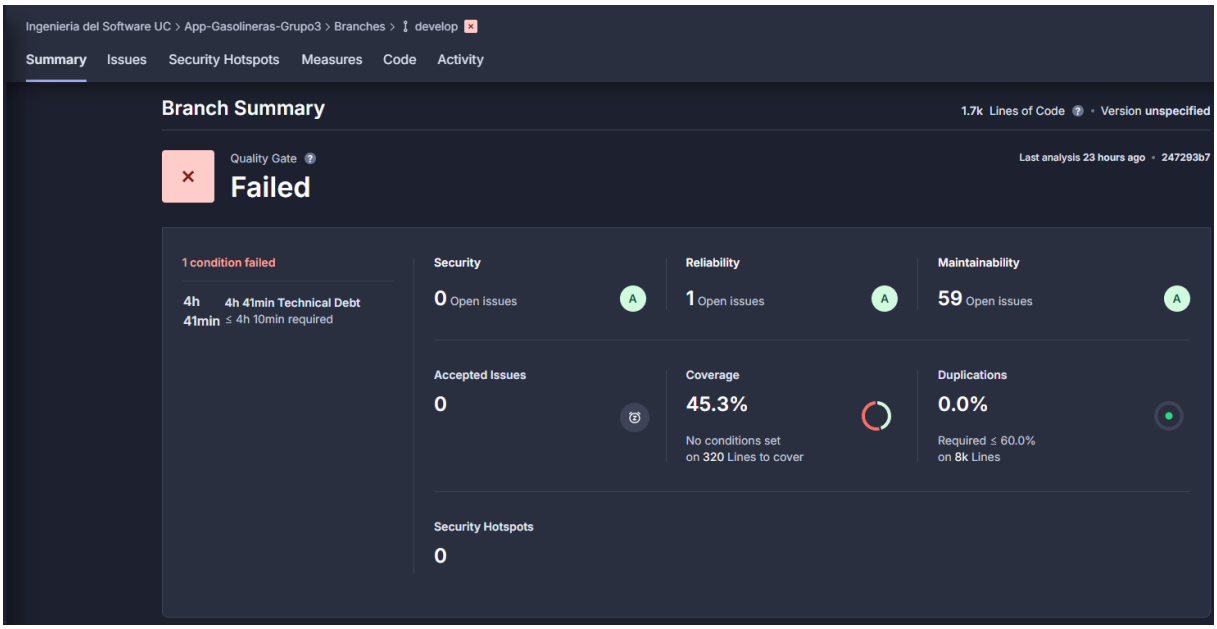
En este informe se va a analizar la calidad del producto desarrollado durante la primera semana del primer sprint del proyecto integrado. Se realiza a fecha de 19 de octubre de 2024, habiéndose desarrollado las dos interfaces nuevas que se requerían, así como la implementación del código fuente de Consultar Repostaje 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 seguridad, solo uno de realibity y 59 de maintainability. No hay duplicaciones de código, pero se la cobertura es inferior al 50%, por lo que esto hace que el quality gate no esté en *Passed*, esto es debido a que aún no ha sido implementado ningún caso de prueba.

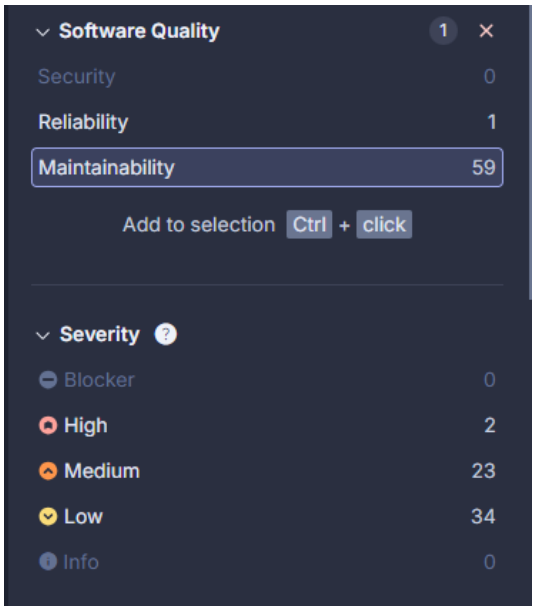
En cuanto a los issues detectados vemos que en su mayoría son fallos de mantenibilidad del código.

Para corregir completamente el proyecto deberíamos emplear 4 horas y 41 min.

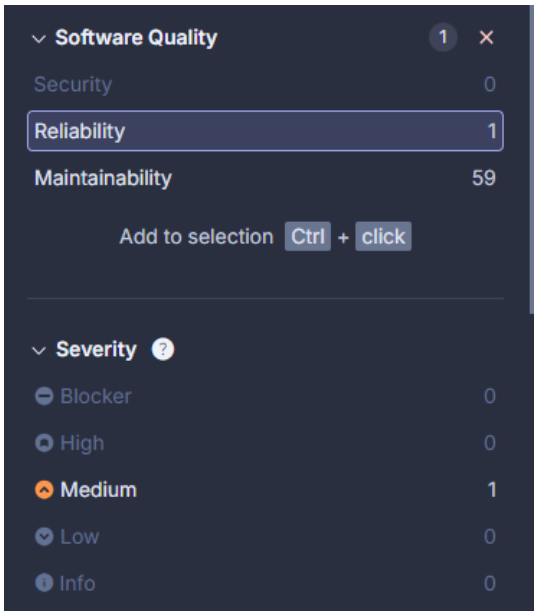


Captura 2.1 Resumen análisis SonarQube

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 2 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.



Captura 2.2 Detalle fallos mantenibilidad



Captura 2.3 Detalle fallos realibity

En el caso de maintainability los fallos graves son:

1. Make the enclosing method "static" or remove this set: que es debido a que cuando hemos creado la base de datos el atributo le hemos declarado como static pero la inicializamos dentro del onCreate de una clase por lo que cada vez que reiniciemos la aplicación perderemos todos los datos.
2. This class is part of 2 cycles containing 3 classes: nos dice que hay un ciclo entre el MainView y ConsultarRepostaje .

De los de severidad media los que mas 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 porporcionado inicialmente ya estaba comentado.
2. Make this annonymous inner class a lambda.
3. Add a private constructor to hide the implicit public one: en varias clases nos propone añadir un constructor privado.

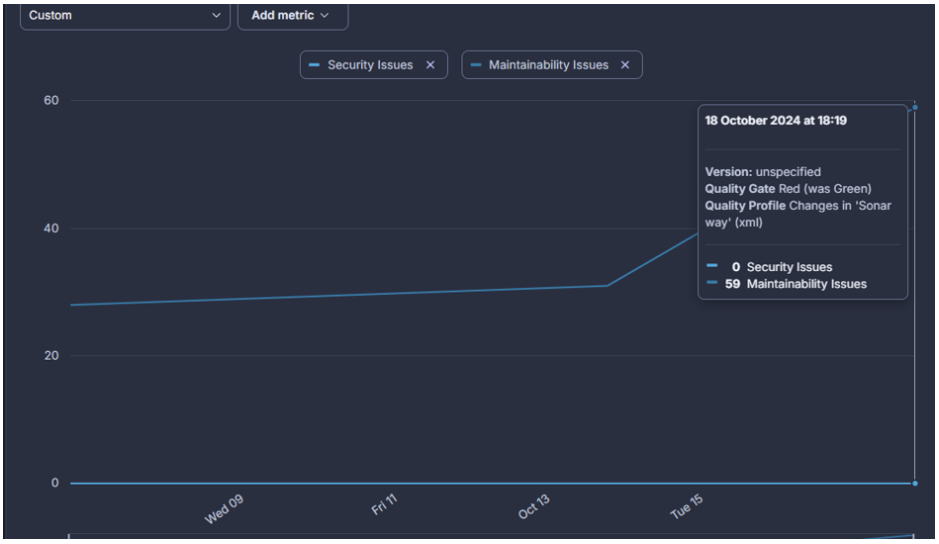
De los de low:

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.

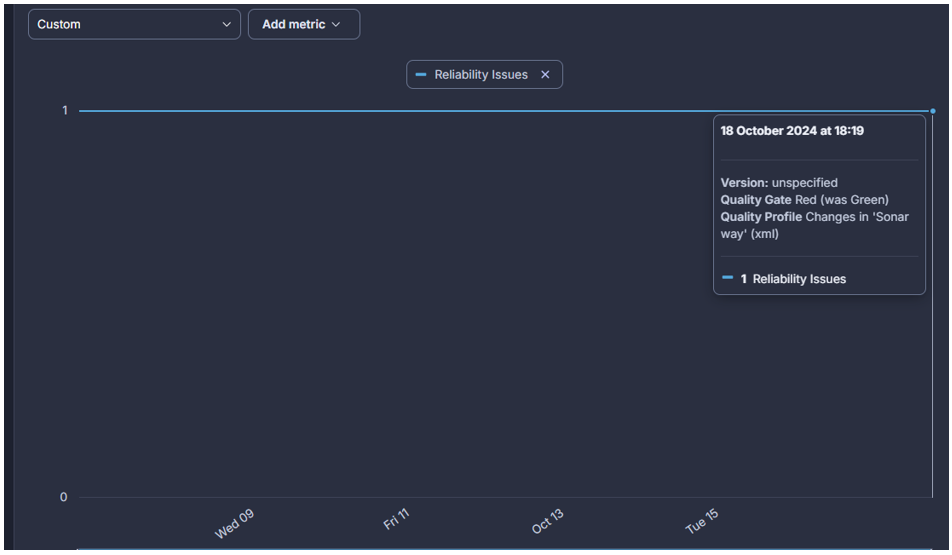
2. Extract this nested code block into a method: nos dice de extraer varios bloques de código cada uno a un método nuevo para luego que el código sea más fácil de comprender.

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 que cuando se nos entregó la primera versión de la aplicación el 14 de octubre ya existían varios fallos de mantenibilidad, pero estos han ido aumentando 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.

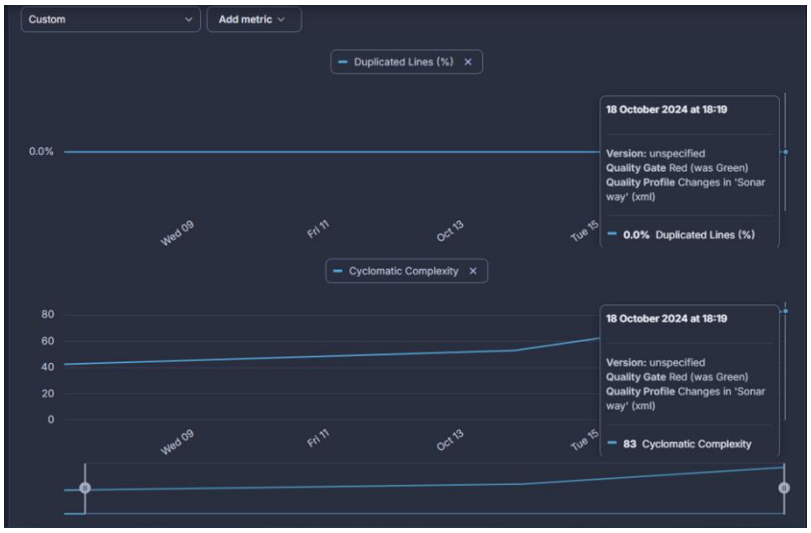


Captura 2.4 Evolución security y maintainability issues



Captura 2.5 Evolución issues reliability

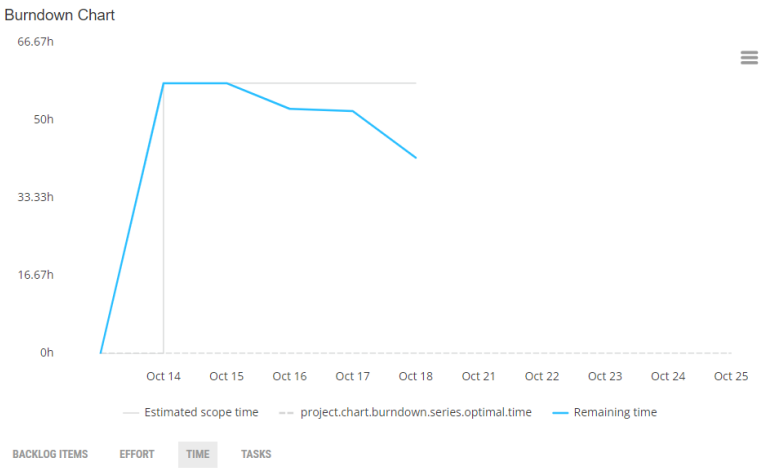
Finalmente, en la captura 2.6, observamos que el código mantiene una ausencia de duplicación de líneas, siendo un dato positivo. Sin embargo, la complejidad ciclomática ha aumentado debido a la incorporación de nuevas clases y métodos, lo cual era esperado. Aunque un aumento moderado en la complejidad 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 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 burdown chart

4. PLAN DE ACCIÓN PARA LA MEJORA DEL CÓDIGO

Prioridad	Nombre	Severidad	Localización	Effort
1	Catch exception instead of Trowable	Medium	ConsultarRepostaje	20 minutos
2	Make this annonymous inner class a lambda	Medium	ConsultarRepostaje, RegistrarView,	25 minutos
3	Add a private constructor to hide the implicit public one	Medium	Utils, RepositoriesModule, GasolinerasService	15 minutos
4	Extract this nested code block into a methos	Low	RepostajesArrayAdapter, y GasolinerasArrayAdapter	90 minutos
5	Eliminar imports sin usar	Low	Varias clases	18 minutos
				2h y 48 min

No se ha decidido resolver los issues de severidad alta ya que, uno de ellos cambiará debido a que la creación de la base de datos en esa clase es provisional, por lo que cuando se arregle lo citado este issue desaparecerá. 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.

No se incluye el issue de eliminar código comentado ya que parte de ello desaparecerá cuando se termine la implementación ya que son cosas que aún no funcionan del todo o que finalmente se sustituirán por otras. Además, aunque si se ha incluido eliminar imports que no se usan, puede que parte de ellos si 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 cada una de hora y media que las dos implicarán ejecutar el citado plan, pero será realizado por dos integrantes del grupo para equilibrar de mejor manera la carga de trabajo.