INFORME DE CALIDAD DE PRODUCTO

1. INTRODUCCIÓN

Este informe tiene como objetivo evaluar la calidad del producto desarrollado durante la primera semana del primer sprint del proyecto integrado. La revisión se realiza con fecha 19 de octubre de 2025. Durante este periodo se han trabajado las historias de usuario **Listar contenido deseado** y **Buscar contenido por nombre**, incluyendo el desarrollo de sus interfaces correspondientes.

El funcionamiento de **Listar contenido deseado** se encuentra prácticamente finalizado, mientras que el de **Buscar contenido por nombre** está completamente implementado. Esto proporciona una base sólida para analizar los progresos alcanzados e introducir posibles mejoras antes del cierre del sprint.

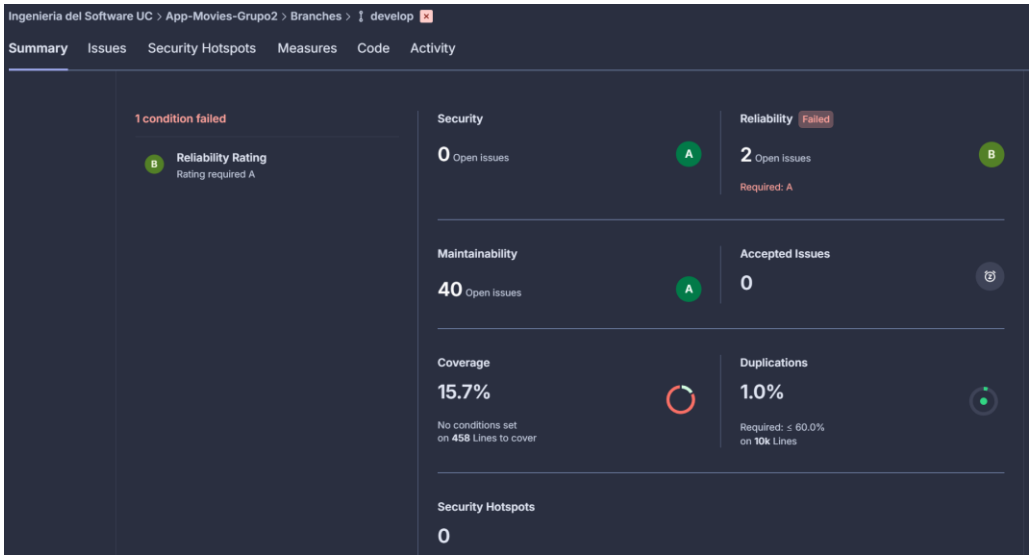
2. ANALISIS DE CALIDAD DE PRODUCTO

En la captura 2.1 se muestra el resumen del análisis realizado mediante la herramienta **SonarQube** sobre el código desarrollado durante esta primera semana. Los resultados indican que no se han detectado errores de seguridad (**security**), mientras que se han identificado 2 incidencias de fiabilidad (**reliability**) y 40 problemas de mantenibilidad (**maintainability**).

El análisis refleja un 1 % de código duplicado, causado principalmente por 102 líneas repartidas en las clases **MainView.java** (49 líneas) y **WishlistView.java** (53 líneas).

La cobertura de código se sitúa en un 15,7 %, un valor inferior al umbral recomendado. No obstante, este resultado se considera provisional, ya que las pruebas aún no han sido implementadas.

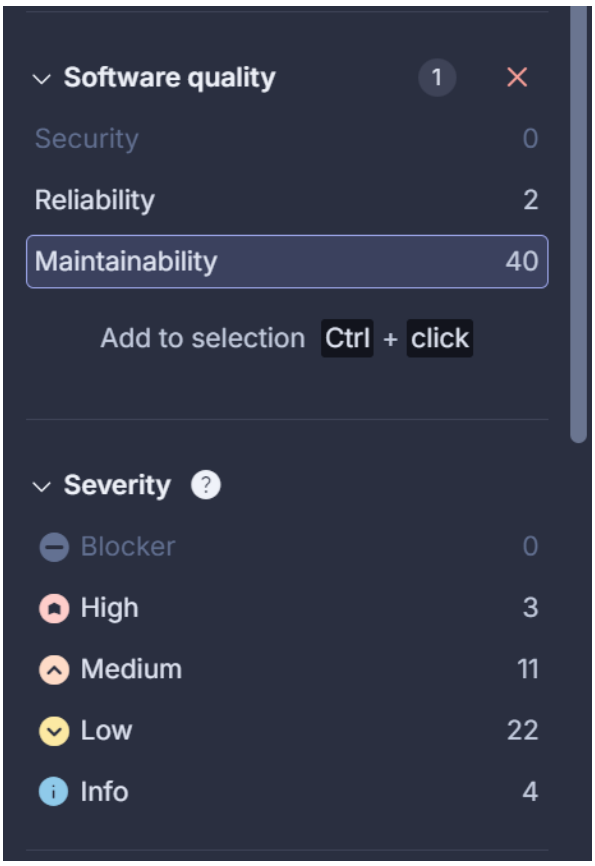
En conjunto, se estima que serían necesarias aproximadamente **3 horas y 44 minutos** de trabajo para resolver completamente las incidencias reportadas.

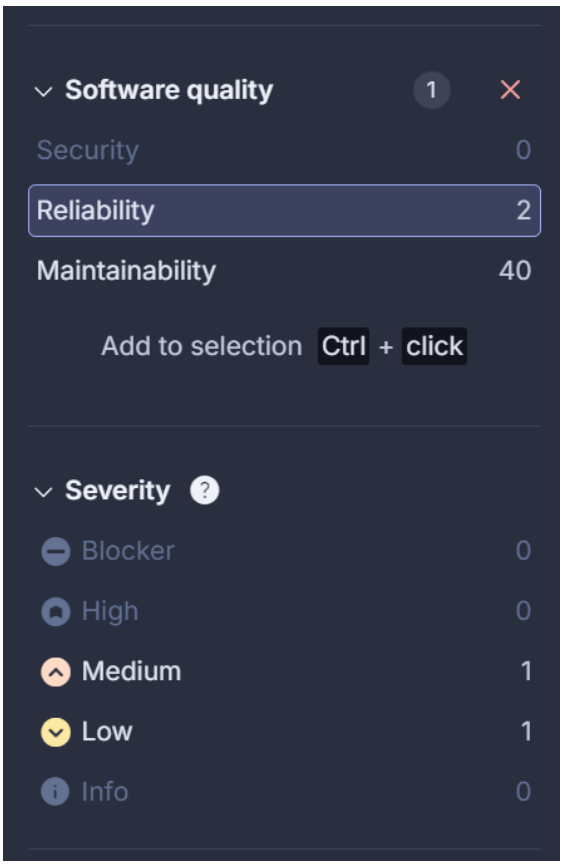


Captura 2.1 Resumen del análisis

En la captura 2.2 se puede observar la severidad de los fallos de mantenibilidad. Aunque se han detectado varios fallos, solo tres son de severidad alta, por lo que no se prevé que su corrección requiera mucho tiempo.

Respecto a la fiabilidad, como se observa en la captura 2.3, se han detectado dos fallos: uno de severidad media y otro de baja. Esto indica que, aunque existen fallos de calidad, no son críticos y se pueden corregir sin un alto coste en tiempo.





Captura 2.2 Fallos mantenibilidad desglosados *Captura 2.2 Fallos fiabilidad desglosados*

Fallos graves de mantenibilidad:

1. **Add a nested comment explaining why this method is empty, throw an UnsupportedOperationException or complete the implementation:** En la clase **SeriesDB**, el método seriesDB() aún no se ha implementado.
2. **Make the enclosing method "static" or remove this set:** En la clase **MoviesApp**, se asigna this a una variable estática (instance) dentro de un método no estático.
3. **Rename this class:** Hay que renombrar la clase **DetailsView**.
4. Los fallos más repetidos son de **severidad baja**, como eliminar imports innecesarios o constantes no utilizadas.

Fallos de fiabilidad:

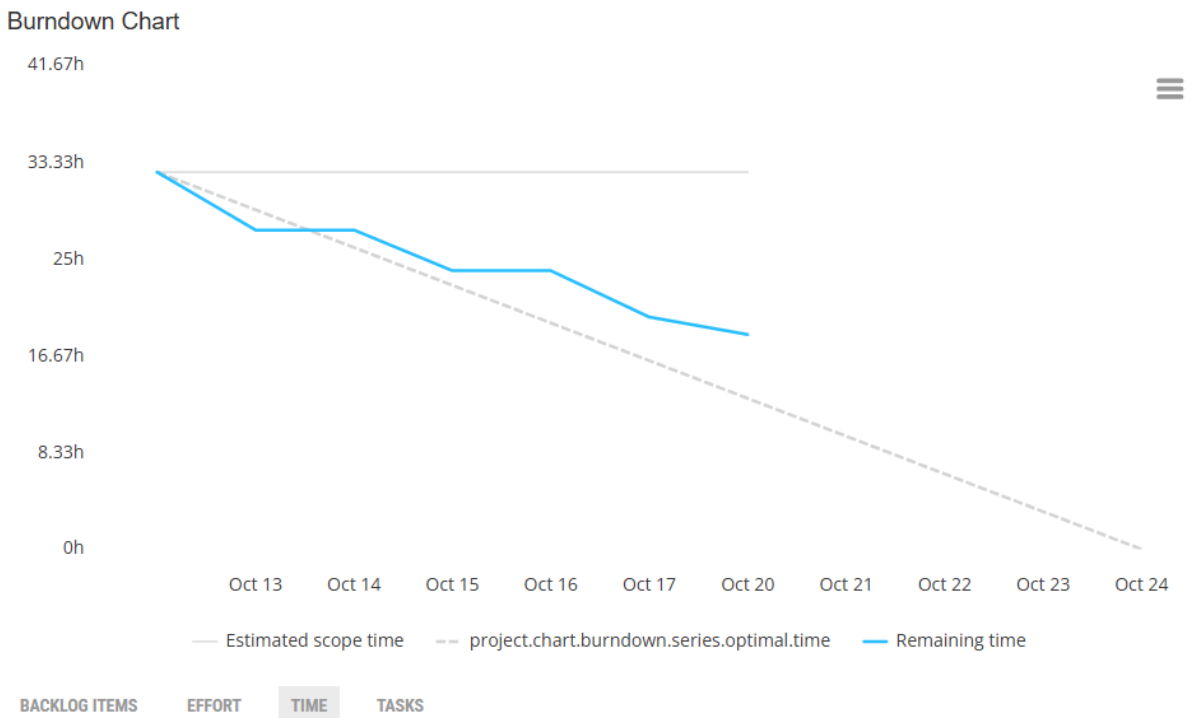
1. **Remove this field injection and use constructor injection instead:** Se sugiere eliminar la directiva @Inject y usar inyección mediante constructor.
2. **Cast one of the operands of this addition operation to a "double":** En el método obtenerPuntuacionSumaria de la clase **Utils**, tanto 1 como voteCount son valores enteros. Al sumarlos directamente, el resultado también se trata como número entero antes de pasarlo a Math.log10(), lo que puede causar una ligera pérdida de precisión.

3. EVOLUCIÓN CARGA DE TRABAJO

En el **Sprint Burndown Chart** (captura 3.1) se aprecia la evolución de la carga de trabajo a lo largo del sprint. La línea azul representa el tiempo restante real para completar las tareas, mientras que la línea gris discontinua muestra la trayectoria ideal.

Durante los primeros días, el equipo redujo significativamente el trabajo pendiente, indicando un inicio productivo y buena organización. En la parte media del gráfico se observa cierta estabilización, por la resolución de tareas más complejas. Hacia el final del periodo, la línea real se mantiene cercana a la ideal, evidenciando planificación y adaptación efectivas.

En términos generales, la evolución refleja una ejecución sólida y controlada, con reducción constante de la carga pendiente y demostrando madurez en la planificación. Si esta tendencia se mantiene en próximos sprints, será posible cumplir objetivos y mejorar la eficiencia global, reduciendo la deuda técnica acumulada.



Captura 3.1 Sprint Burndown Chart

4. PLAN DE ACCIÓN CORRECTIVA

Prioridad	Nombre del problema	Severidad	Localización	Esfuerzo estimado	Acción Correctiva
1	Cast one of the operands of this addition operation to a "double".	Low	Utils	5 min	Declarar argumento (voteCount) como double.
2	Add a nested comment explaining why this method is empty, throw an UnsupportedOperationException or complete the implementation.	High	SeriesDB	5 min	Borrar método inútil seriesDB().
3	Make the enclosing method "static" or remove this set.	High	MoviesApp	20 min	Quitar setter de Lombok.
4	Rename this class.	High	DetailsView	5 min	Cambiar nombre a DetailsSeriesView.

5	Add the "@Override" annotation above this method signature.	Medium	MoviesApp	5 min	Añadir @Override previo a método onCreate().
6	Replace this assert with a proper check.	Medium	MainPresenter	5 min	Reemplazar assert por condición explícita.
7	Extract this nested try block into a separate method.	Medium	MainPresenter	20 min	Meter bloque try en nuevo método readWishlist.
8	Remove this unused "context" private field.	Medium	SeriesAdapter y WhislistAdapter	10 min	Borrar variable no usada context.
9	Rename "wishlist" which hides the field declared at line 17.	Medium	WishlistPresenter	5 min	Cambiar nombre a localWishlist.
10	Add a private constructor to hide the implicit public one.	Medium	Utils y RepositoriesModule	10 min	Añadir constructor sugerido por Sonar.
11	Remove this commented out code.	Medium	Themes.xml y backup_rules.xml	10 min	Borrar código comentado.
12	Remove unused imports.	Low	Varias clases	8 min	Borrar imports no usados.
TOTAL ESTIMADO PLAN DE ACCIÓN CORRECTIVA				1 hora y 48 min	

Se ha asignado **prioridad 1** al primer problema, ya que bloqueaba la compilación, impidiendo la ejecución correcta del proyecto. Se han abordado todos los problemas de **alta y media severidad**, mientras que algunas tareas de baja prioridad también se han ejecutado para mantener el código limpio y ordenado. No obstante, algunos errores de baja prioridad no se han solucionado para no sumar demasiado tiempo. El problema relacionado con la anotación @Inject, aunque de severidad media, se ha dejado sin modificar siguiendo la recomendación del IDE.

5. IMPLEMENTACIÓN PLAN ACCIÓN CORRECTIVA

La implementación del plan se ha gestionado mediante la creación de una única tarea en **ScrumDesk** denominada **“Implementar Plan de Acción Correctiva”**, asignando un responsable encargado de aplicar todas las correcciones definidas.