INFORME SPRINT 1: CALIDAD DE PRODUCTO

1. Introducción

El objetivo de este documento es analizar la calidad del producto software AppMovies-Grupo1, centrando el estudio en las historias de usuario desarrolladas durante el Sprint 1. Para ello, se ha empleado la herramienta SonarQube, que permite identificar y categorizar los problemas de calidad del código, así como estimar el esfuerzo necesario para su corrección.

El análisis se ha realizado sobre el commit 96093a34285dcfa4747deae58a633230dcb47c25 correspondiente a la rama *develop*, una vez integrada la primera versión funcional del Sprint.

A partir de los resultados obtenidos, se propone un plan de acción orientado a la resolución de las incidencias más relevantes, con el objetivo de mejorar la mantenibilidad, fiabilidad y seguridad del sistema, y asegurar una evolución continua y controlada en los próximos Sprints.

2. Análisis inicial de calidad (SonarQube y Scrumdesk)

En este apartado se presentan las capturas obtenidas de SonarCloud y Scrumdesk, que ofrecen una visión general del estado actual de la calidad del código y del progreso del Sprint. Estas imágenes permiten identificar de un vistazo los principales indicadores de seguridad, fiabilidad, mantenibilidad y avance temporal del desarrollo.

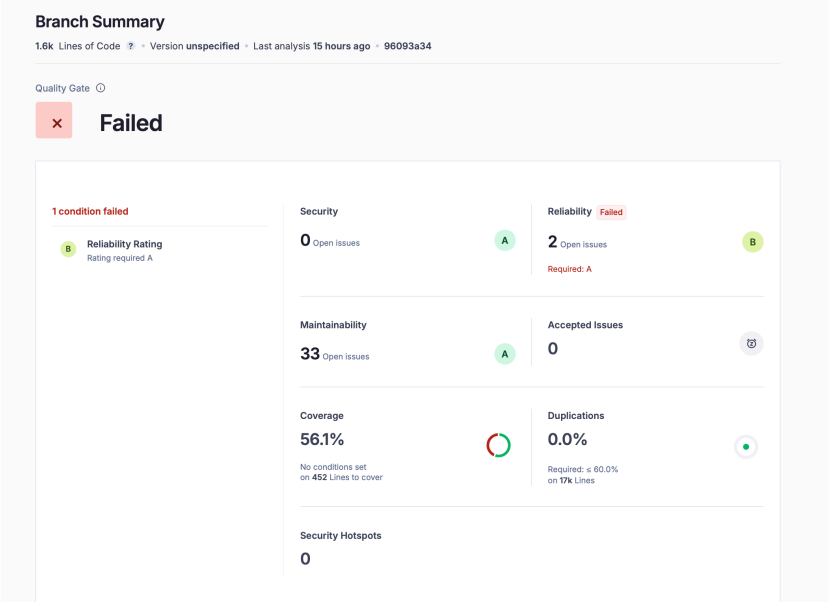


Figura 1. Resumen del análisis

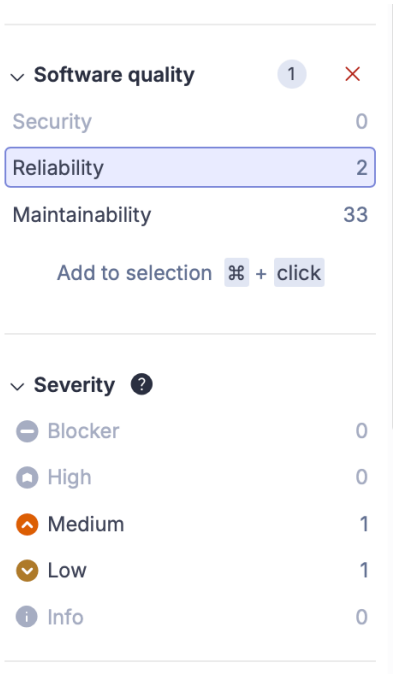


Figura 2. Severidad de issues – Reliability

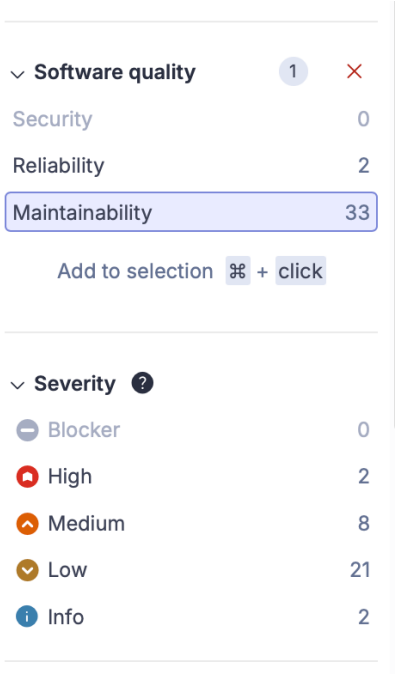
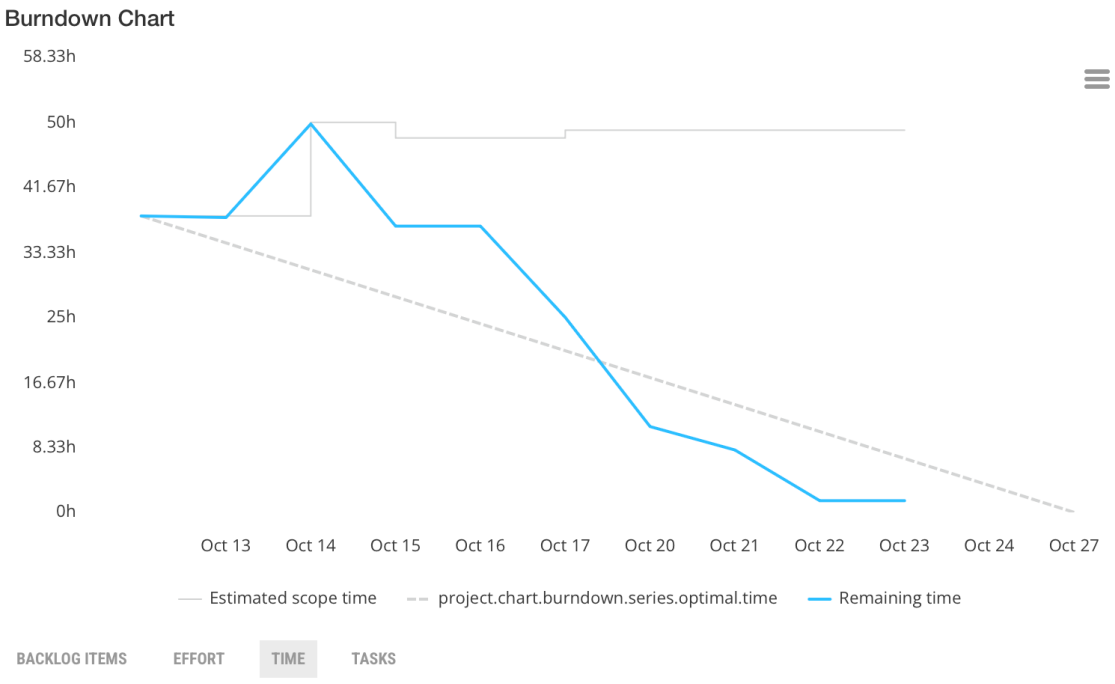


Figura 3. Severidad de issues – Maintainability

Gráfica 1. Número de issues

Gráfica 2. Porcentaje de código repetido y complejidad ciclomática



Gráfica 3. Burndown Chart

3. Análisis del estado del producto

A continuación, llevaremos a cabo un análisis detallado de la calidad del producto, con el objetivo de determinar cuáles son los problemas de calidad principales, y dónde están localizados.

Al analizar el proyecto podemos detectar rápidamente que el Quality Gate no se ha pasado. Este fallo se debe principalmente por el atributo de Reliability, a pesar de no encontrarse muchos problemas, solamente dos, estos hacen que la calificación de fiabilidad sea “B”, con lo cual no alcanza el mínimo exigido para pasar el Quality Gate, es necesario que se evalúe como “A”. Aunque en este análisis no marca más condiciones fallidas, podemos prestar atención a la cobertura alcanzada, ya que ahora mismo solo se ha logrado cubrir poco más del cincuenta por ciento, lo que indica que una parte significativa del código aún no está verificada mediante pruebas automatizadas. Esto puede suponer un mayor riesgo de errores al realizar cambios o introducir nuevas funcionalidades.

En las figuras 2 y 3 se puede observar como la mayoría de los problemas de calidad se concentran en la mantenibilidad del código, con un total de 33 issues, de los cuales dos son de severidad alta y ocho de severidad media. En cuanto a la fiabilidad, los dos problemas encontrados tienen severidad media y baja, teniendo solo esto en cuenta podríamos asumir una menor urgencia aunque no importancia de su solución. Sin embargo tras el anterior análisis correspondiente a la Figura 1, hay que tener en cuenta que aunque los problemas no sean críticos sí que afectan notoriamente a la calificación final, no alcanzando el mínimo exigido.

En conjunto, los resultados reflejan que la prioridad inmediata debería centrarse en resolver las incidencias de fiabilidad para cumplir con los criterios mínimos de calidad, y a continuación abordar las refactorizaciones relacionadas con mantenibilidad para mejorar la calidad global del producto.

Sobre el aspecto de seguridad, verificamos que en la Gráfica 1 se mantiene como una línea recta constante, indicando que no se ha encontrado ninguna vulnerabilidad y que el código es seguro.

La complejidad ciclomática, presenta un aumento (Gráfica 2) desde aproximadamente 40 hasta superar los 150, algo esperado ya que se han incorporado nuevas funcionalidades complejas al código pero que habrá que solucionar mediante refactorizaciones para evitar que el código se vuelva difícil de mantener y comprender.

En la Gráfica 3, correspondiente al Burndown Chart se observa una tendencia descendiente representando el tiempo restante del sprint. Al inicio aparece un aumento del esfuerzo estimado, pero esto se debe a la incorporación de tareas que se realiza durante los dos primeros días del sprint. A partir del 15 de octubre la curva comienza a descender, acercándose a la tendencia de la línea discontinúa, que marca el ritmo óptimo. En general se puede decir que el tiempo y esfuerzo ha sido gestionado de manera adecuada.

4. Plan de Acción

En este apartado se presenta el plan de acción que se deberá llevar a cabo antes de que finalice el sprint. Dado que el Burndown Chart indica que el tiempo del sprint se encuentra prácticamente agotado, el plan de acción será el que se hubiera realizado en el supuesto caso de disponer de 50 minutos adicionales de trabajo. Aunque solo se tendrá en cuenta la incidencia cuyo problema es necesario resolver para que el proyecto satisfaga los criterios de calidad de producto mínimos definidos por la organización.

El plan de mejora priorizará las incidencias más relevantes detectadas por SonarCloud. Para empezar, asignaremos la máxima prioridad a los dos problemas de fiabilidad, que afectan directamente al Quality Gate. Seguidamente se priorizará la corrección de los defectos más severos (nivel alto y nivel medio) con el fin de facilitar el desarrollo a futuro de la aplicación. Al no haberse detectado problemas de seguridad estos no se consideran.

A continuación se detallan las incidencias a corregir (de más a menos prioridad):

- src/.../movies/activities/details/DetailsPresenter.java (línea 73) Cast one of the operands of this addition operation to a "double".
    - Problema de fiabilidad de severidad baja, afecta al Quality Gate.
    - Esfuerzo estimado por instancia: 5min
    - Garantiza cálculos más precisos y evita errores.
    -
- src/.../unican/movies/activities/main/MainView.java (líneas 46-47) Remove this field injection and use constructor injection instead.
    - Potencial problema de fiabilidad de severidad media.
    - Esfuerzo estimado por instancia: 5min
    - Para garantizar una inicialización más segura, mejorar la fiabilidad del componente y facilitar las pruebas unitarias.

- src/.../unican/movies/activities/main/MainPresenter.java (línea 109) Define a constant instead of duplicating this literal "\\s*\\(\\d+\\)$" 5 times. Y (línea 116) Define a constant instead of duplicating this literal "%s (%d)" 4 times.
    - Problema de mantenibilidad de severidad alta.
    - Esfuerzo estimado por instancia: 12 y 10 min
    - Favorece la legibilidad y reutilización.

- src/.../unican/movies/activities/main/IMainContract.java (línea 33), src/main/res/values-night/themes.xml (línea 5) y src/main/res/xml/backup_rules.xml (línea 12) Remove this commented out code.
    - Problema de mantenibilidad de severidad media.
    - Esfuerzo estimado por instancia: 5 min
    - Mantener el código limpio y legible.
    -

Finalmente, teniendo en cuenta la limitación del tiempo disponible, como ya se ha indicado anteriormente, se incluye en la lista de tareas de Scrumdesk una nueva tarea “*Resolución plan de acción*” que recoja la resolución de la incidencia marcada como más prioritaria.