Informe de calidad – MyFuel (Grupo 1)

Fecha de realización: Durante el Sprint 3, el día 2024/11/13

Hash del commit: [c3be433]

Índice:

Resumen del análisis

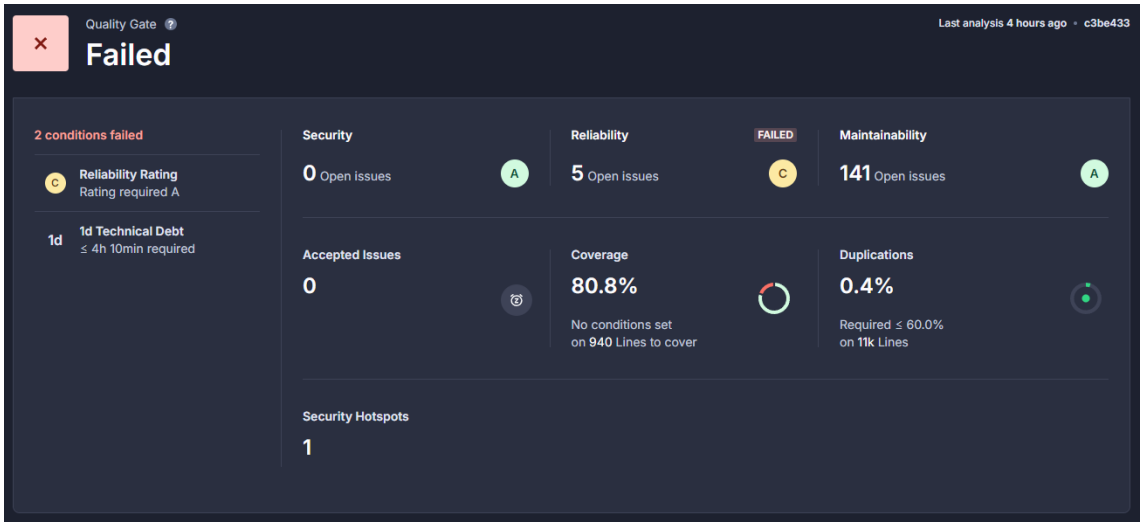


Ilustración 1: Vista general de SonarCloud.

Se puede ver que, al pasar SonarQube, el código que se ha implementado no cumple con los estándares de calidad. Analizando globalmente, vemos que los puntos más importantes a mejorar para pasar el Quality Gate son los siguientes:

- Mejorar la fiabilidad (Reliability) a 'A', ya que actualmente posee una fiabilidad de 'C', debido a la existencia de 5 Open Issues que hay que solventar.
    - Solución: eliminar los open issues que bajan la fiabilidad.

- Mejorar la Mantenibilidad del código, con la incorporación de las últimas funcionalidades se han incrementado los problemas de mantenibilidad respecto al último análisis (SP02-QAReport.pdf), generando una deuda técnica de más de 1 día (unas 9 horas de trabajo) para solucionar dichos problemas. SonarQube nos dice que dicha deuda debe ser igual o inferior a 4 horas y 10 minutos para poder pasar el Quality Gate.
    - Solución: Reducir la deuda técnica al menos al número de horas marcado por SonarCloud.

En lo referente al resto de medidas principales que determinan el Quality Gate, podemos ver que los valores son correctos y no generan problemas, entrando en detalle sobre estos:

- Security (A): hay cero vulnerabilidades.
- Coverage: se comprueba 80.8% del código cubierto con pruebas, un incremento de más del 10% respecto último análisis (SP02-QAReport.pdf).
- Duplications: en este caso, vemos que hay un 0.4% de duplicaciones de código añadidas, respecto al último análisis (SP02-QAReport.pdf), donde no había código duplicado.

Severidad de los issues de cada elemento principal

Security

No existen problemas se seguridad detectados.

Reliability

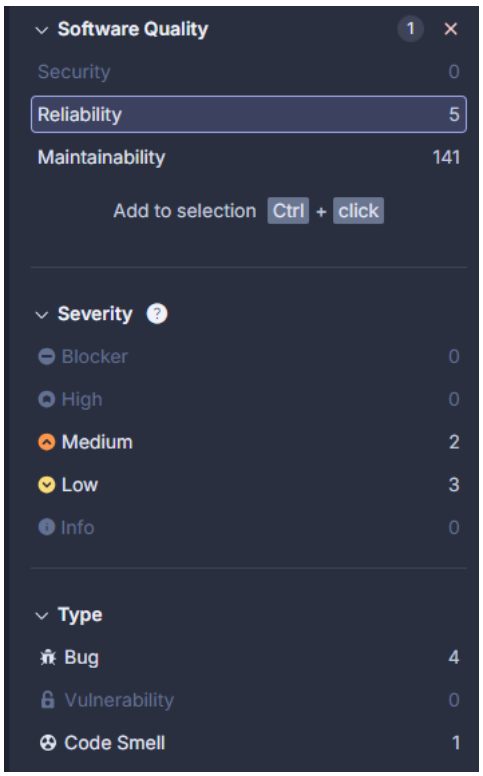


Ilustración 2: Severidad de los issues de Reliability.

En cuanto a fiabilidad, hayun total de 5 problemas detectados:

- Severidad Medium: 2.
- Severidad Baja: 3.

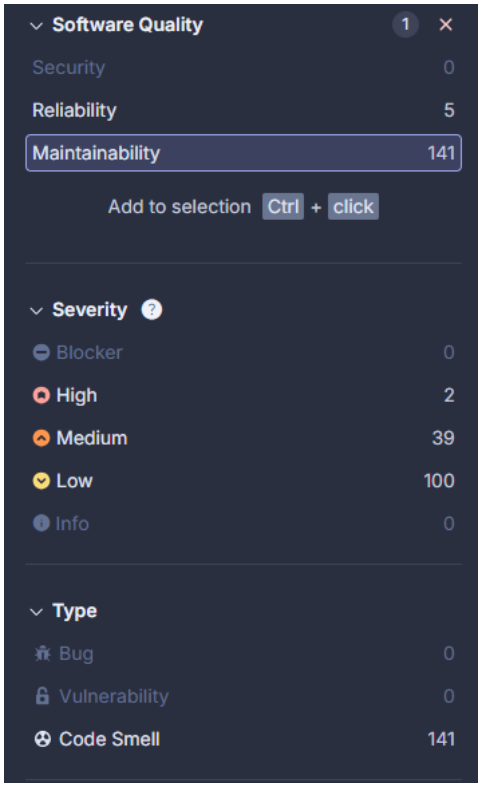
Sin embargo, se ve que 4 de ellos son bugs. Se les dará prioridad a la hora de solucionarlos respecto al resto de problemas que se detecten de mantenibilidad que no sean bugs y al code smell detectado en esta sección, esto debido a que los bugs pueden generar errores en la ejecución de la aplicación.

A continuación, se exponen los issues principales de Reliability ordenados de mayor a menor siguiendo el criterio anterior categorizado por SonarCloud.

Nombre	Clase	Severidad	Tipo	Deuda Técnica
Make "dateFormat" an instance variable.	Converters.java	Medium.	Bug.	15 min.
"name" is marked "@NonNull" but is not initialized in this constructor.	InterestPoint.java	Low.	Bug.	45 min (3 errores iguales de 15 minutos).
Remove this field injection and use constructor injection instead.	MainView.java	Medium.	Code Smell.	5 min.
			TOTAL:	1h 5min

Tabla 1: Principales Issues Reliability.

Maintainability



En cuanto a fiabilidad, hay un total de 141 problemas detectados:

- Severidad High: 2
- Severidad Medium: 39.
- Severidad Baja: 100.

Se ve que todos ellos son Code Smell, por ende, las prioridades a la hora de mostrar los issues principales se establecerán en base a la severidad.

Ilustración 3: Severidad de issues de Maintanability.

A continuación, se exponen los issues principales de Maintanability, teniendo en cuenta serán ordenados de mayor a menor respecto a su nivel de severidad y respecto al tiempo de deuda técnica categorizado por SonarCloud. Se priorizará la severidad para el caso de los hard, y luego, entre los medium y low se priorizará la deuda técnica de cada issue, mostrando errores de severidad low sobre aquellos medium cuyo effort sea bajo, que no se mostrarán como prioritarios.

Nombre	Clase	Severidad	Deuda Técnica
This class is part of one cycle containing 2 classes.	MainView.java	Hard.	0 min.
"This file "MockRepositories.java" should be located in "es/unican/gasolineras/utils" ...	MockRepositories.java	Hard.	5 min.
Define and throw a dedicated exception instead of using a generic one.	MainPresenter.java	Medium.	20 min.
Update this method so that its implementation is not identical to "closeFiltersPopUp" on line 423.	MainView.java	Medium.	15 min.
Remove this use of "getParcelable"; it is deprecated.	DetailsView.java	Low.	15 min.
Extract this nested code block into a method.	GasolinerasArrayAdapter .java	Low.	50 min (5 errores iguales de 10 min).
Extract this nested code block into a method.	PointsArrayAdapter.java	Low.	50 min (5 errores

			iguales de 10 min).
This block of commented-out lines of code should be removed.	SeekBarActions.java MainPresenterTest.java MainPresenterITest.java	Medium.	50 min (10 errores iguales de 5 min).
		TOTAL:	3,25 min

Tabla 2: Principales Issues Maintanability.

Por otro lado, en SonarCloud hay una serie de problemas de Mantenibilidad de grado medio que se deben ignorar debido a que, por fallos en Github Actions, los test no pasan si no se comentan las líneas donde se comprueban los toast. Se ha decidido comentar dichas líneas para que pasen los test al integrar en develop. Por ende, todos aquellos issues que sean de la forma “This block of commented-out lines of code should be removed” que se encuentren en una clase de test y que comenten un toast serán omitidos.

Estos son, en total, 5 issues * 5 minutos = 25 minutos.

Además, se detectan 50 issues respecto a la importación de librerías innecesarias, a las cuales se les asigna un 1min de effort a cada una, resultando en 50 minutos totales de esfuerzo. Sin embargo, estas librerías pueden ser fácilmente eliminadas mediante la opción “Sort import” de Android Studio, lo que provoca que no necesitemos más de 2min para solventar todos estos problemas.

Gráficas de issues por tipos

A continuación, se mostrarán gráficas del crecimiento de los issues clasificadas por tipos a lo largo del ciclo de vida de la aplicación, se separan en dos graficas las métricas de mantenibilidad respecto a las de fiabilidad y seguridad para visualizar mejor el crecimiento de los datos.

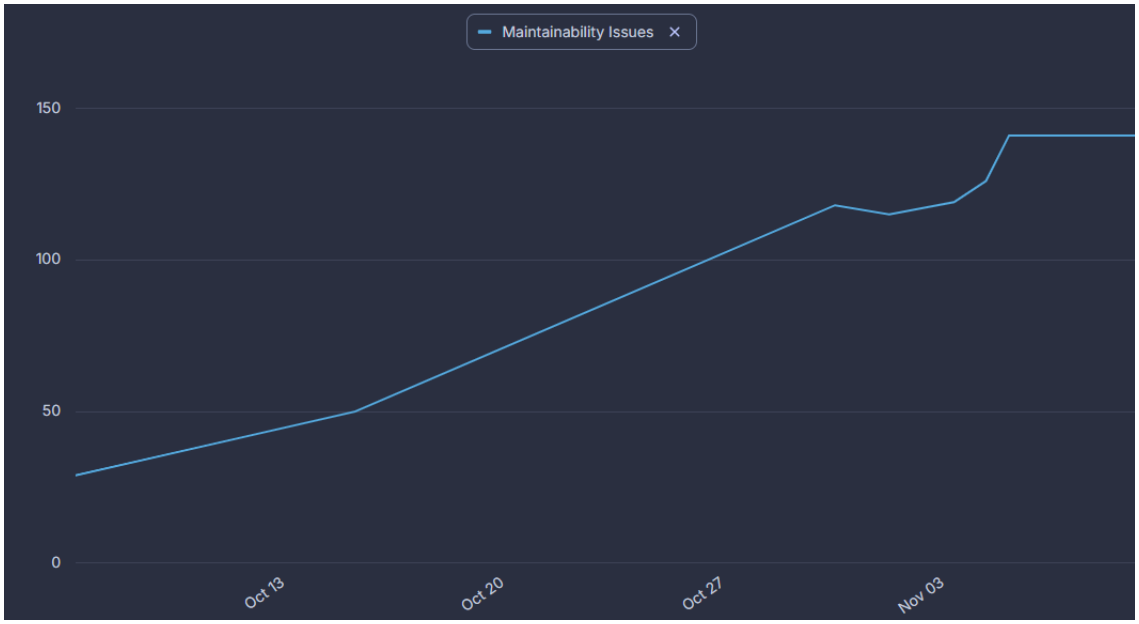


Ilustración 4: Gráfico progresión problemas de Mantenibilidad.

Ilustración 5: Gráfico progresión problemas de Fiabilidad y Seguridad.

Se ve que, a lo largo de la primera semana de noviembre (Sprint 2), disminuyen los problemas de mantenibilidad ligeramente debido a la ejecución del plan de acción del último análisis (SP02-QAReport.pdf), y seguido de esto, con la incorporación de nuevas funcionalidades vuelven a aumentar significativamente los issues en comparación con el decremento previo. En cambio, los problemas de fiabilidad aumentan ligeramente, y con las últimas incorporaciones crecen a más del doble.

Porcentaje de código repetido y complejidad ciclomática

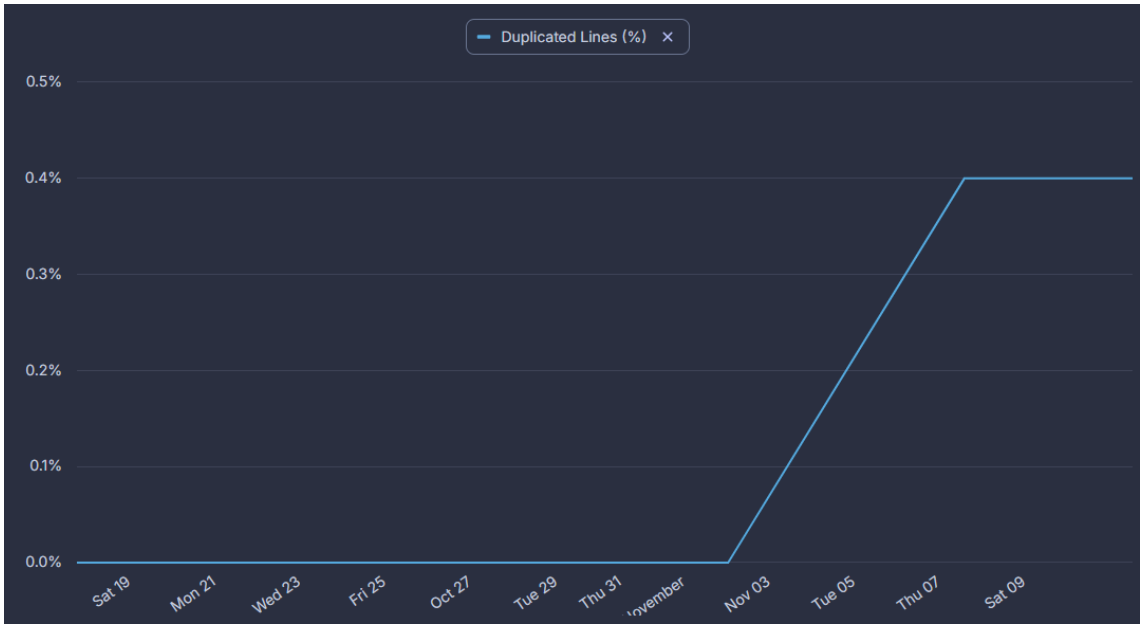


Ilustración 6: Gráfico que muestra la evolución del % de código repetido durante la duración del proyecto

Ilustración 7: Gráfico que muestra la evolución de la complejidad de los algoritmos presentes durante el proyecto

En el gráfico de código repetido se puede ver que en la última integración en "develop" de cara al Sprint II, ha incrementado el número de líneas de código repetidas, siendo un porcentaje del 0,4% de líneas repetidas en todo el código. Un número muy reducido teniendo en cuenta la cantidad de líneas de código total, que son unas 11.000.

En cuanto a la complejidad ciclomática, se aprecia como en la primera semana de noviembre (última semana del Sprint II), ha incrementado la complejidad, sin embargo, el incremento ha sido inferior al de semanas anteriores (Sprint I y comienzo del Sprint II). De hecho, en la primera semana del Sprint II se observa cómo se trató de corregir levemente la complejidad. En los primeros días de este Sprint III se mantiene la complejidad. Por ende,

donde más hay que refactorizar en cuanto a complejidad serían en los desarrollos realizados en el Sprint I (del 14 al 25 de octubre) y Sprint II (del 28 de octubre al 8 de noviembre).

Burndown Chart

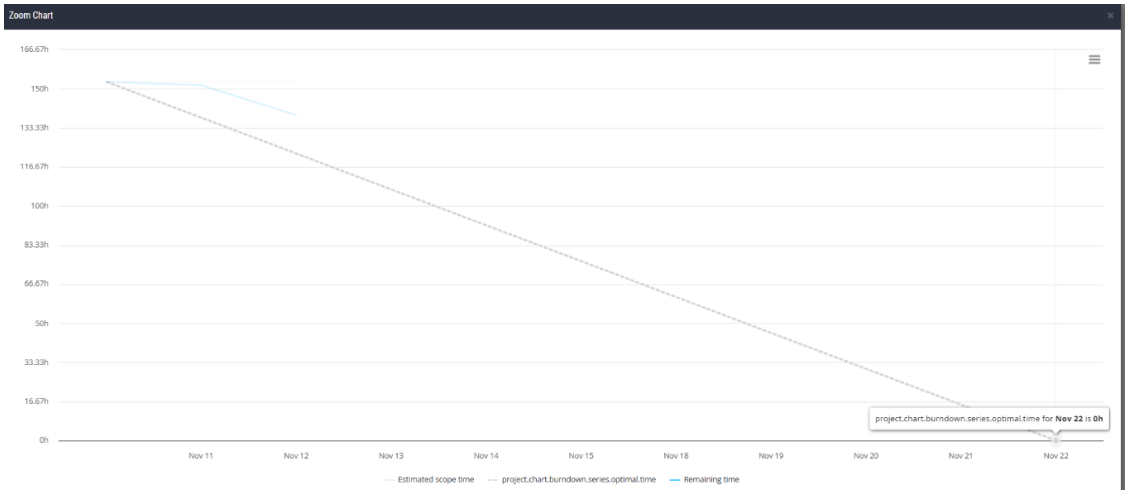


Ilustración 9: Gráfico de ScrumDesk que indica el avance del sprint actual respecto a una evolución ideal

Disclaimer: esta es la única captura que se ha podido obtener del Burndown Chart, ya que ha habido una serie de problemas con ScrumDesk y no está generando las gráficas correctamente. La captura ha sido realizada en el mismo día en que se ha comenzado el informe.

En este gráfico se puede ver cómo al inicio del Sprint hay un cierto atraso temporal, si el equipo sigue la misma trayectoria de la línea azul, se acabaría el proyecto con un día de atraso. Por ende, se valora que, en dicho momento, la ejecución de un plan de acción resolviendo todos los problemas descritos no sería viable. Sin embargo, el día 18/11/2024, tras la primera semana del Sprint, se ha analizado (sin el Burndown Chart, ya que no está disponible), el progreso en tareas, y se ha visto que, muchas de las tareas ya estaban realizadas. Por ello, se ha decidido resolver todos los issues identificados en el apartado "*Severidad de los issues de cada elemento principal*", asignando la tarea del plan de acción a los miembros más desocupados en esta semana del Sprint.

Plan de acción

Con el objetivo de pasar el Quality Gate y teniendo en cuenta el tiempo libre del que se dispone para implementar las mejoras estimado en base al Burndown Chart anterior, se planean realizar las siguientes acciones en el siguiente orden:

1. Solucionar los issues de Reliability mencionados en la Tabla 1 (en este punto la deuda técnica se reducirá a 1h 5min).
2. Solucionar los issues de Maintanability mencionados en la Tabla 2 (en este punto la deuda técnica se reducirá a 1h 5min + 3h 25min = 4h 30min).
3. Descontar de la deuda técnica lo relativo a los toast mencionado debajo de la Tabla 2 (en este punto la deuda técnica se reducirá a 4h 30min + 25min = 4h 55min).
4. Eliminar los imports innecesarios mencionados debajo de la Tabla 2 (en este punto la deuda técnica se reducirá a 4h 55min + 50min = 5h 45min)

De esta forma, se reducirá la deuda técnica a unas 9h – 5h 45min = 3h 15min, pasando de esta forma el Quality Gate, ya que en el proceso también se han solucionado los problemas de Reliability.

Es imperativo recordar que los fallos relativos a los toast no se descuentan automáticamente del Quality Gate, por lo que, si no se realizan todos los cambios solicitados, es posible que este Quality Gate visualmente no pase en SonarCloud aunque si haya pasado en la teoría.

Nótese que, para llevar a cabo este plan de acción, se ha de añadir una tarea en este sprint para realizar los cambios solicitados llamada “Ejecutar plan de acción calidad”.

Ejecución del plan de acción

Dentro de las issues definidas algunas se han quedado fuera debido a que son errores que sonar no interpreta bien o se ha determinado que la corrección puede causar más problemas y se han dejado para futuras mejoras con menos issues.

Lista de las issues que no se han corregido:

- Remove this use of "getParcelable"; it is deprecated. En DetailsView.java Se ha omitido debido a que requiere implementar Parcelable en Gasolinera.
- "Remove this field injection and use constructor injection instead." En MainView.java no es un error y es parte de la implementación que no detecta bien sonar. Por eso se deja como está.

Además de los issues planificados para solucionar durante el proceso se han corregido varios issues sencillos detectados con SonarLint:

Nombre	Clase	Severidad	Deuda Técnica
Add a private constructor to hide the implicit public one.	Converters,java	Major	5 min
Complete the task associated to this TODO comment. Corrección: Menciones a la opción TODOS la cual se han puesto comillas y se solución.	MainPresenter.java	Info	4x0 min
Remove useless curly braces around statement Corrección: Listeners que llaman a 1 metodo y se soluciona con una lambda	MainView.java	Minor	10x5 min
This block of commented-out lines of code should be removed.	MainPresenter.java MainPresenterITest.java	Info	11x5 min
Unused "private" methods should be removed	FuelTypesUITest.java	Info	2 min
Utility classes should not have public constructors	InterestPointValidator.java Utils.java	Major	2x5 min
		TOTAL:	1h 52 min

Tabla 3: Issues detectados con SonarLint

- Se ha corregido también el SecurityHotspot localizado en la clase Converters.java con nombre Insecure Configuration, la solución ha sido quitar que se pinte el stackTrace.
- Se ha eliminado la clase LimitPricesEnum, la cual ya no se usaba y tenía 4 issues.

Análisis de calidad del producto al final del Sprint

Tras haber aplicado el plan de acción, y haber integrado en Develop los cambios, se ha analizado en SonarCloud el proyecto después de haber incorporado además todas las funcionalidades. En la siguiente imagen se analizará el Quality Gate.

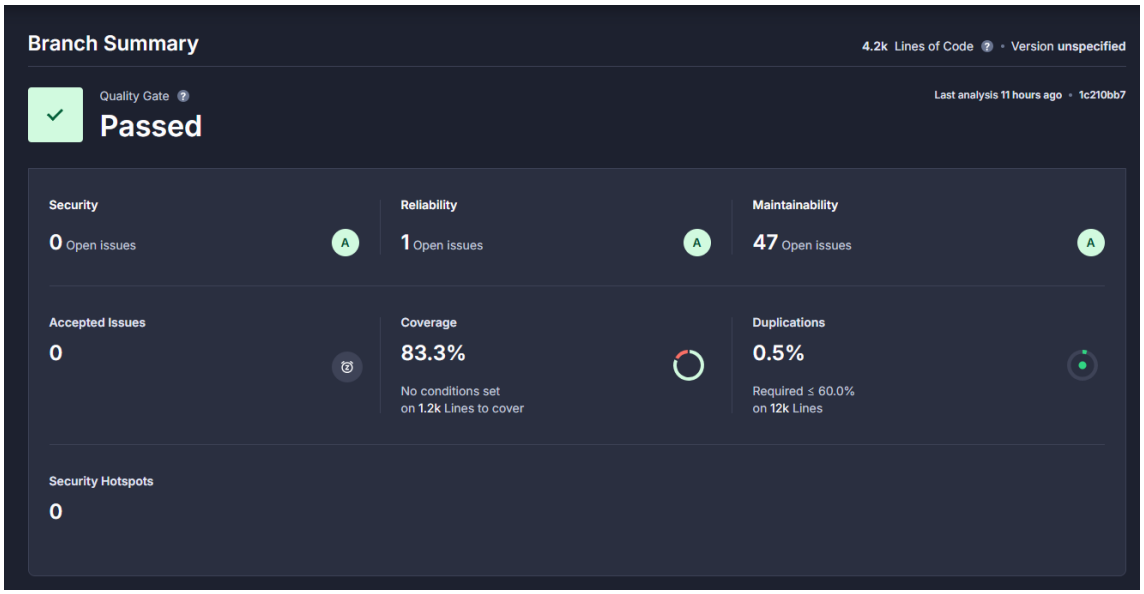


Ilustración 8: Análisis general de la calidad del commit 1c210bb7 en rama develop.

En este caso, ser ve como se ha pasado el Quality Gate, teniendo únicamente 1 problema de Reliability y una calificación de A, y habiendo solucionado los problemas de Maintanability reduciendo las horas de deuda técnica que ya no sobrepasan el límite de 4h establecido.