**Lint Report**

* Nombre del proyecto: Acme ANS
* Grupo: C2.051
* Repositorio: https://github.com/danjimdia1/Acme-ANS
* Responsable: Manuel Jesús Sánchez García
* Fecha: 29/06/2025

**Resumen ejecutivo**

Este informe tiene como objetivo presentar un análisis preliminar de la calidad del código fuente del proyecto utilizando la herramienta SonarLint. A través de esta herramienta, se busca identificar posibles problemas de estilo, estructuras complejas o malas prácticas que puedan afectar a la legibilidad, mantenibilidad o funcionamiento del sistema. A lo largo del informe se explicará la metodología seguida, las clases que han sido analizadas y los resultados obtenidos, en caso de que se detecten incidencias. Además, se evaluará si es necesario aplicar correcciones o si la implementación actual puede considerarse adecuada según las reglas establecidas por SonarLint.

**Capítulo 1: Análisis de código con SonarLint**

Clase AirlineManager



SonarLint nos indica que deberíamos sobreescribir el método equals. En este caso no sería necesario porque AirlineManager extiende hasta la clase AbstractEntity que ya implementa el equals.

También nos indica que debemos renombrar el paquete, que actualmente tiene el nombre de “airlineManager”, aunque por convención debería llamarse “airlinemanager”, lo cuál no es un code smell ni problema aparente.

Clase Flight



Nos indica que deberíamos sobreescribir el método equals. En este caso no sería necesario porque Flight extiende hasta la clase AbstractEntity que ya implementa el equals.

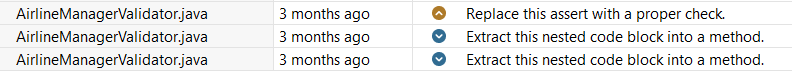
Clase Leg



Nos indica que deberíamos sobreescribir el método equals. En este caso no sería necesario porque Flight extiende hasta la clase AbstractEntity que ya implementa el equals.

Además también nos indica que en el método getDuration deberíamos retornar el valor directamente en vez de incluirlo primero en una variable y después retornarlo, no es un code smell ni error aparente.

Clase AirlineManagerValidator



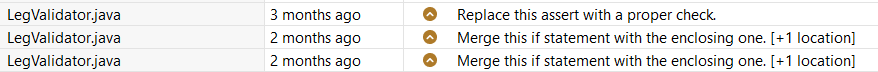
Aquí nos indica que reemplazemos el assert pero viene del framework así que no debemos preocuparnos por ello. También nos indica que podemos extraer condiciones a métodos separados, aunque creo que no es conveniente pues cada bloque condicional indica una restricción del AirlineManager y sólo es necesario leer el código para entender perfectamente que se está haciendo.

Clase FlightValidator



Igual que en el anterior, ese assert viene del framework así que no debemos preocuparnos por ello.

Clase LegValidator



Igual que en el anterior, ese assert viene del framework así que no debemos preocuparnos por ello.

Además se nos indica que que combines condicionales, aunque si bien podría combiar en un solo if, va a tener la misma complejidad y no considero que sea una mala práctica a la hora de escribir una condición anidada, únicamente preferencia del desarrollador.

Clase AirlineManagerFlightController

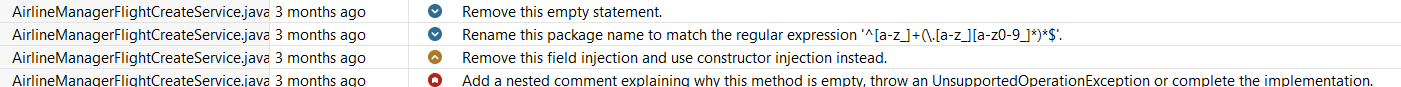
A close up of a screen

AI-generated content may be incorrect.

Igual que en archivos anteriores nos indica que el nombre del paquete no sigue la convención estándar, no es muy relevante.

También SonarLint se queja de la inyección de dependencias por no usar un constructor para inyectar los servicios, lo cual carece de relevancia.

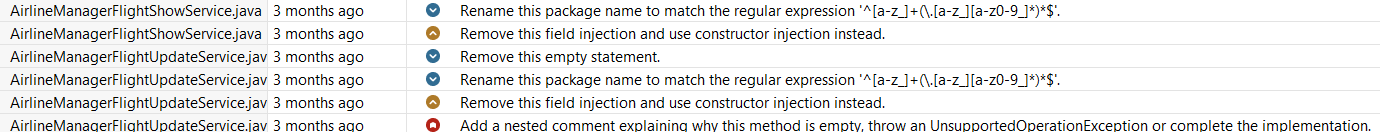
Clase AirlineManagerFlightCreateService



Aquí SonarLint se queja de que el método validate está vació y que tiene una sentencia vacía, nos indica que lo comentemos. Lo podemos obviar puesto que el mejor modo de comentar un código es mediante el propio código, y por tanto aquí nos indica que este servicio no tiene una lógica específica de validación. también nos vuelve a indicar el mal olor del nombre d ellos paquetes pero considero que no es relevante, al igual con la inyección de dependencias.

Para los demás servicios de AirlineManager, Show, List, Publish y Update se nos indica lo mismo que en el servicio de Create.

A screenshot of a computer

AI-generated content may be incorrect. 

Para la clase AirlineManagerLegController se nos indica lo mismo que para las clase AirlineManagerFlightController

A screenshot of a computer

AI-generated content may be incorrect.

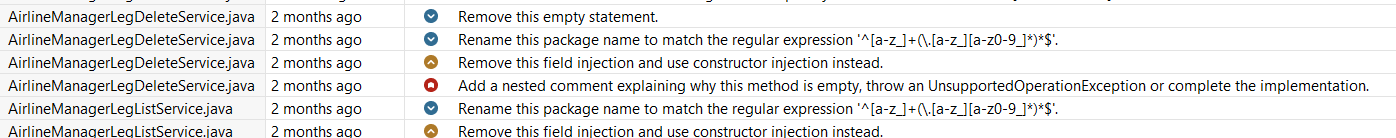
Clase AirlineManagerLegCreateService

A close up of a text

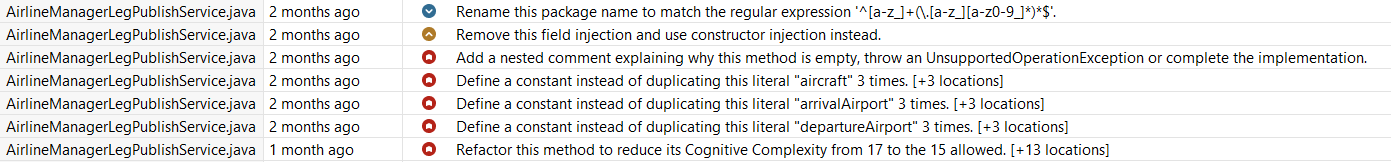
AI-generated content may be incorrect.

SonarLint tiene quejas similares a servicios anteriores. Además nos indica que que definamos una constante para varios literales duplicados, se podría hacer si estos literales fueran cambiando a lo largo del proyecto sin embargo no es el caso y no tiene sentido ya que aun creando una constante global para estos literales habría que seguir repitiéndolos varias veces. También nos indica que la complejidad cognitiva del método authorise es alta, lo cual es normal debido a que es una función critica del sistema con varias validaciones. Seguido a esto, a pesar de tener una complejidad cognitiva alta en mi opinión la función es bastante legible pues separa detalladamente los bloques necesarios para comprobar cada atributo de navegación y arroja al final el resultado de la autorización. Se podría tratar de cambiar pero al ser una función compleja y muy importante no sería del todo recomendable.

Los servicios de AirlineManagerLegListService, AirlineManagerLegShowService y AirlineManagerLegDelteService tiene indicaciones similares a las mencionadas anteriormente



Los servicios AirlineManagerLegupdateService y AirlineMangerLegPublishService tienen indicaciones similares a la de la clase AirlineManagerLegCreateService

 A close up of a text

AI-generated content may be incorrect.

**Conclusión**

Tras analizar el proyecto con SonarLint, la mayoría de los avisos detectados son recomendaciones menores que no afectan al funcionamiento del sistema, como nombres de paquetes o variables intermedias. Algunos métodos presentan una complejidad cognitiva algo alta, pero está justificada por la lógica que manejan y su estructura es clara.

En general, el código es legible y está bien organizado. No se considera necesario aplicar cambios importantes por ahora, aunque se recomienda revisar ciertos métodos en el futuro si su lógica crece. La calidad del código es adecuada para el estado actual del proyecto.