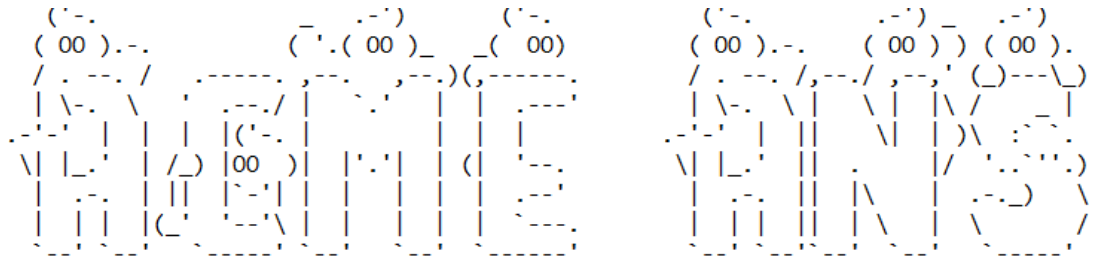
REPORTE DE LINT

Acme-ANS-D04



Repositorio: https://github.com/FranciscoFernandezN/Acme-ANS

Creado por el grupo C1.022, del G1

Participantes	
Nombres	Correos
Fernández Noguerol, Francisco	frafernog@alum.us.es

24 de mayo de 2025

Índice

Resumen ejecutivo

Este es el proyecto del grupo C1.022 sobre Acme AirNav Solutions, S.A. (Acme ANS, S.A. abreviado), la cual es una compañía ficticia especializada en ayudar a aeropuertos a organizar y coordinar sus operaciones a partir de soluciones desarrolladas en software. La logística de los vuelos (la programación de los vuelos, la organización de reservas y de tripulación, etc.) se gestionan mediante el desarrollo de un WIS.

Con esto, para el correcto funcionamiento de la aplicación, se deberá escribir un reporte de testing donde se describan los valores de cobertura del código junto con los valores correspondientes al rendimiento de la aplicación, ambos, previos y posteriores a realizar mutaciones en el código.

Tabla de revisiones

Número	Fecha	Descripción
v1.0.0	24/05/2025	Versión finalizada del documento para el entregable 4

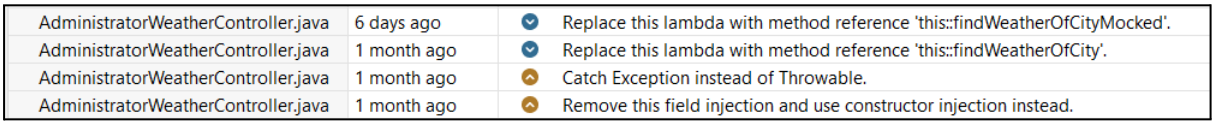
Introducción

Para el contenido de este documento, se han tenido en cuenta todos los resultados de SonarLint para los requisitos individuales del estudiante 1, tanto de aquellos requisitos obligatorios como opcionales.

Para cada entrada relevante de SonarLint, explicaré porqué pienso que no debe ser cambiado, o en su defecto, porqué lo he cambiado.

Contenidos

Cambios AdministratorWeatherController:



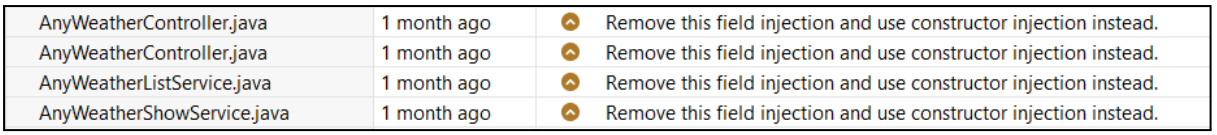
AdministratorWeatherController.java	6 days ago	Replace this lambda with method reference 'this::findWeatherOfCityMocked'.
AdministratorWeatherController.java	1 month ago	Replace this lambda with method reference 'this::findWeatherOfCity'.
AdministratorWeatherController.java	1 month ago	Catch Exception instead of Throwable.
AdministratorWeatherController.java	1 month ago	Remove this field injection and use constructor injection instead.

Figura 1: Resultados SonarLint para AdministratorWeatherController

Las dos primeras entradas del análisis hacen referencia a la forma de acceder a un método sin parámetros de un objeto en Java. No es relevante pero sí mejora la estética y legibilidad del código, por lo que he realizado el cambio.

Las otras dos recomendaciones hacen referencia a partes del código globalmente usadas en Acme-Framework, como es la notación @Autowired o el “cazar” la excepción Throwable a la hora de poblar la base de datos, en vez de usar Exception, por lo que ambas recomendaciones son inofensivas y no se cambiarán.

Cambios AnyWeather:

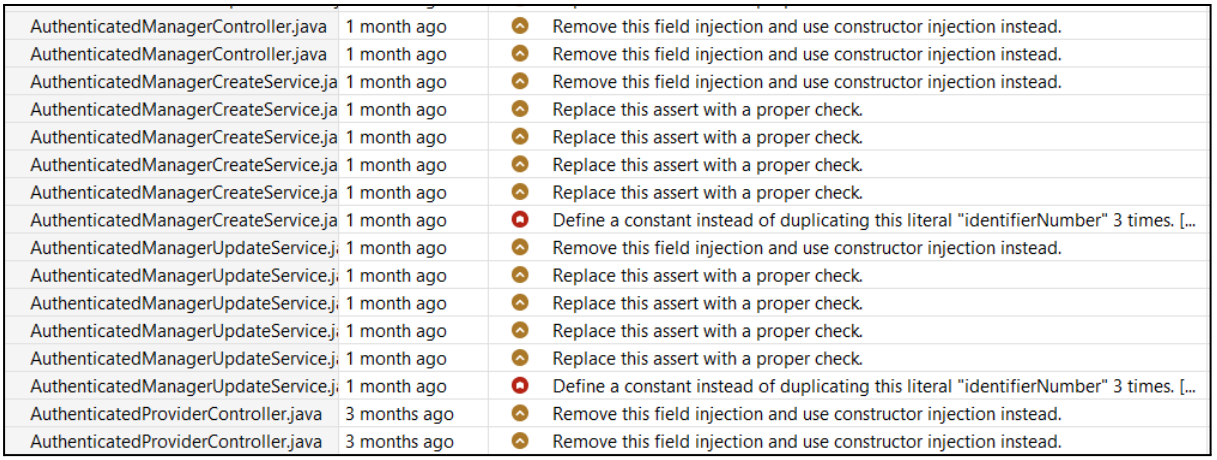


AnyWeatherController.java	1 month ago	Remove this field injection and use constructor injection instead.
AnyWeatherController.java	1 month ago	Remove this field injection and use constructor injection instead.
AnyWeatherListService.java	1 month ago	Remove this field injection and use constructor injection instead.
AnyWeatherShowService.java	1 month ago	Remove this field injection and use constructor injection instead.

Figura 2: Resultados SonarLint para las features de Weather

En este caso, todas las recomendaciones de SonarLint tienen que ver con la notación @Autowired, por lo que se pueden considerar como inofensivas.

Cambios AuthenticatedManager:



AuthenticatedManagerController.java	1 month ago	Remove this field injection and use constructor injection instead.
AuthenticatedManagerController.java	1 month ago	Remove this field injection and use constructor injection instead.
AuthenticatedManagerCreateService.ja	1 month ago	Remove this field injection and use constructor injection instead.
AuthenticatedManagerCreateService.ja	1 month ago	Replace this assert with a proper check.
AuthenticatedManagerCreateService.ja	1 month ago	Replace this assert with a proper check.
AuthenticatedManagerCreateService.ja	1 month ago	Replace this assert with a proper check.
AuthenticatedManagerCreateService.ja	1 month ago	Replace this assert with a proper check.
AuthenticatedManagerCreateService.ja	1 month ago	Define a constant instead of duplicating this literal "identifierNumber" 3 times. [...
AuthenticatedManagerUpdateService.j	1 month ago	Remove this field injection and use constructor injection instead.
AuthenticatedManagerUpdateService.j	1 month ago	Replace this assert with a proper check.
AuthenticatedManagerUpdateService.j	1 month ago	Replace this assert with a proper check.
AuthenticatedManagerUpdateService.j	1 month ago	Replace this assert with a proper check.
AuthenticatedManagerUpdateService.j	1 month ago	Replace this assert with a proper check.
AuthenticatedManagerUpdateService.j	1 month ago	Define a constant instead of duplicating this literal "identifierNumber" 3 times. [...
AuthenticatedProviderController.java	3 months ago	Remove this field injection and use constructor injection instead.
AuthenticatedProviderController.java	3 months ago	Remove this field injection and use constructor injection instead.

Figura 3: Resultados SonarLint para las features de Authenticated y Manager

En este caso, vemos como se reporta la necesidad de eliminar asserts innecesarios, tras analizar bien el origen de este, se ha llegado a la conclusión que venían a raíz

de unas líneas que originalmente eran del proyecto Hello World, por lo que han sido eliminadas y por tanto, corregido ese error.

Por otro lado, se vuelve a repetir el error de usar @Autowired, junto con un error que también será recurrente en el resto del documento, que consiste en la no duplicación de literales, la cual se produce a la hora de poner los atributos de un objeto en forma de String para su método unbind, bind, etc. Como es irracional poner una constante para estos literales, esta entrada de SonarLint también será omitida.

A partir de este momento, estos dos errores, al ser tan frecuentes, serán omitidos de la explicación.

Cambios Flight:

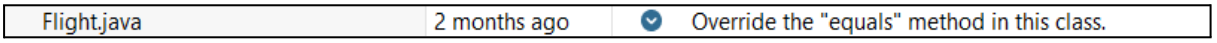


Figura 4: Resultados SonarLint para la entidad de Flight

Este error viene provocado por no sobreescribir el método equals por defecto de Java, como sabemos que esto lo gestiona el framework internamente, podemos omitir este error para esta y todas las futuras entidades que aparezcan.

A partir de este momento, solo aparecerán aquellas clases en las que no se repitan estos mismos fallos, para los cuales, se modificarán o no en función de la explicación previa.

Cambios ManagerDashboard:

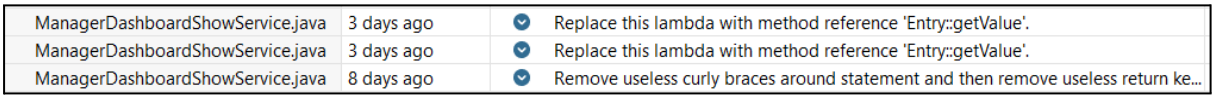


Figura 5: Resultados SonarLint para el dashboard del Manager

Estos errores, son en parte, parecidos a los errores anteriores:

- Los dos primeros, corresponden con la nomenclatura de Java para acceder a funciones, pero en este caso, están dentro de un objeto de tipo Stream, y si se usara esa nomenclatura, Java pierde la traza sobre qué tipo de objetos está usando, y por lo tanto, no es posible modificar estos dos avisos.
- El último de ellos, corresponde con el uso de un corchete a la hora de definir un objeto de tipo Function para mejorar su legibilidad. Si bien, de vista al código, no es muy bonito ya que añade líneas “innecesarias”, de cara a su comprensión es mucho más legible y por lo tanto, no supone un error que deba ser cambiado.

Cambios ManagerFlightCreate:

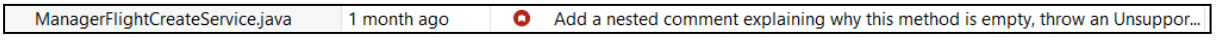


Figura 6: Resultados de SonarLint para la creación de vuelos para Managers

Otro error común que detecta SonarLint es que el método validate está vacío, y se

debería de explicar el porqué de ello. Como en Acme-Jobs esta praxis es recurrente, este aviso es totalmente inofensivo, ya que se da por hecho que si el método validate está vacío, implica que no se debe validar nada por la propia naturaleza del requisito.

Cambios ManagerFlightList:

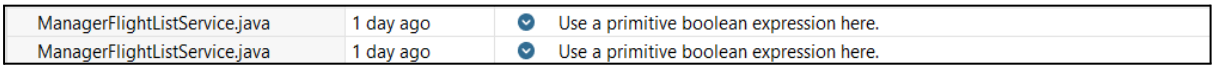


Figura 7: Resultados de SonarLint para el listado de vuelos para Managers

Este error reporta que, en una condición de tipo <condición>?<result1>:<result2>, la condición no debe ser el propio booleano, sino que debe ser: booleano == condición_booleano. Esto, no es relevante para el proyecto y además, empeora la legibilidad del código, por lo que no supone un problema para el proyecto.

El resto de errores mostrados por SonarLint en otras clases, formaban parte de los errores inofensivos de los que se han estado hablando por todo el documento, por lo que recuerdo que han sido omitidos. Como nota, no se han tenido que modificar más elementos del código ya que no se ha repetido ningún aviso de los mencionados anteriormente como necesarios por cambiar.

Conclusiones

Como conclusión, el reporte Lint proporciona cierta información sobre el proyecto y propone algunas ideas útiles para mejorar la calidad estática del código, aunque, en la mayoría de casos, estos cambios no suponen una mejora clara.

En este proyecto, al usarse mucho la notación @Autowired y usar los métodos de Acme-Framework para *bindear* los datos, surgen muchos avisos por parte de SonarLint, pero que pueden ser omitidos sin ningún tipo de problema.

Bibliografía

Intencionalmente en blanco