***DISEÑO Y PRUEBAS II***

***TESTING REPORT***

**C2.018**

**Pablo Espinosa Naranjo**

[**pabespnar@alum.us.es**](mailto:pabespnar@alum.us.es)

**https://github.com/davidg43/Acme-SF-D**

**Student 5**



***Indice***

1. Resumen ejecutivo 3
2. Introducción 4
3. Contenido
   1. Pruebas funcionales 5 - 34
   2. Análisis de desempeño 35 - 40
4. Conclusiones 41
5. Bibliografía 42

|  |  |  |
| --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** |
| 30/06/2024 | 1.0 | Versión inicial |
| 02/07/2024 | 1.1 | Testing funcional |
| 04/07/2024 | 1.2 | Análisis de desempeño |
| 05/07/2024 | 2.0 | Versión final |

Tabla de versiones

|  |  |  |
| --- | --- | --- |
| **Fecha** | **Numero de revisión** | **Descripción** |
| 30/06/2024 | 1 | D04 (Julio) |
| 02/07/2024 | 2 | D04 (Julio) |
| 04/07/2024 | 3 | D04 (Julio) |
| 05/07/2024 | 4 | D04 (Julio) |
| 07/07/2024 | 5 | D04 (Julio) |

Tabla de revisiones

***Resumen Ejecutivo***

En este documento se describe de forma detallada el proceso de testeo de nuestro proyecto, Acme-SF. Este proceso es necesario para poder garantizar el cumplimiento de todos los requisitos funcionales solicitados, y asegurar que estos prevalezcan en el futuro.

Este proceso incluye distintos procedimientos, como la adición de suficientes datos de ejemplo, la comprobación exhaustiva del cumplimiento de todos los requisitos solicitados, la prevención de la aplicación en distintos casos de hacking, o la evaluación de rendimiento y tiempo de ejecución por medio de distintas métricas y herramientas estadísticas en diferentes sistemas.

En resumen, se busca asegurar el cumplimiento de las condiciones que debe satisfacer la aplicación por medio de distintos métodos, y posteriormente, recoger todo este proceso en este documento, y plasmar todo su desarrollo al detalle.

***Introducción***

En un informe de testing se realiza una explicación de como se han desarrollado las pruebas de la aplicación, en este caso Acme-SF.

En este documento se tratan específicamente los test referentes a los requisitos 6 y 7 del estudiante 5 del grupo C2.018, que giran en torno a las entidades “Code-Audit” y “Audit-Record”.

Para el desarrollo de este documento se han seguido exhaustivamente los pasos y puntos indicados en el documento “Annexes”, siendo estos: una portada con la información relevante, un índice, una tabla de versiones del documento, una tabla de revisiones, un resumen ejecutivo, una introducción, un contenido dividido en el análisis y explicación de las pruebas desarrolladas y en el análisis de desempeño, unas conclusiones y por último, una bibliografía.

Los apartados de mayor importancia del documento son los dos referidos al contenido. En el primero de ellos se especifican las pruebas realizadas, el porcentaje de código que ha sido cubierto con las mismas y se justifica el código que no ha sido ejecutado o ha sido ejecutado parcialmente. En el segundo, se muestran gráficas y análisis estadísticos que comparan el tiempo de ejecución de la aplicación en distintos ordenadores, así como el tiempo de ejecución antes y después de la adición de índices en nuestras entidades, en una búsqueda de mejorar el rendimiento.

***Contenido***

***Pruebas funcionales***

Para todos los distintos servicios que ofrece la aplicación en torno a las entidades “Code-Audit” y “Audit-Record”.

Es necesario mencionar que las pruebas han sido realizadas en la rama “July-Student5” de manera individual, una vez fueron corregidos los cambios pertinentes de la entrega de junio. En esta rama se encuentran todas las correcciones realizadas.

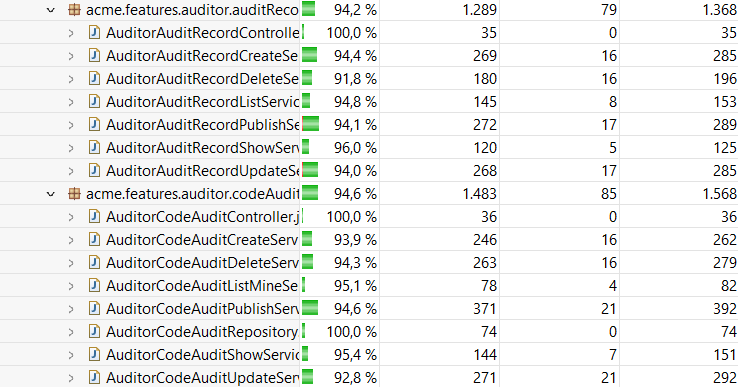
Destaco esto ya que, aunque no fue mi caso, varios de mis compañeros experimentaron errores al ejecutar el Tester Replayer en sus pruebas, dando errores “FAILED TO”. Entendemos que esto se dio debido a actualizaciones de código del resto de compañeros, sobre todo de los .csv, produciendo así modificaciones en la base de datos. Debido a esto, por si en esta entrega sí se vieran afectadas mis pruebas, aclaro que estos test fueron realizados y grabados correctamente con las herramientas proporcionadas por la asignatura y puede ser comprobado su correcto funcionamiento en la rama indicada anteriormente. Por tanto, además, solo se vera afectada la cobertura de los test, no su funcionamiento ni ejecución en el analyser.

Por último, también debe aclararse que los “FAILED TO” relacionados con banner se deben a los errores controlados de “Payload” de los banners mencionados.

Tras esta aclaración, procedemos con las pruebas. Estas están divididas en:

* “Tests.safe”: Pruebas positivas y negativas dentro de un uso normal de la aplicación
* “Tests.hack”: Intentos varios de hackeo en la aplicación

Quedando así:



A continuación, pasamos a enumerar todas las pruebas que han sido realizadas, así como hablar del código ejecutado totalmente y parcialmente.

***Code Audit***

***List-Mine***

***Tests.safe***

* Listado de todos los Code Audit que pertenecen a un usuario
* Listado de Code Audits vacío

***Tests.hack***

* Intentar acceder a un listado de Code Audits sin estar loggeado
* Intentar acceder a un listado de Code Audits loggeado, pero no como auditor

***Código ejecutado***

******

* Assert object != null: Linea obligatoria que sirve para verificar que el objeto no sea nulo.

***Show***

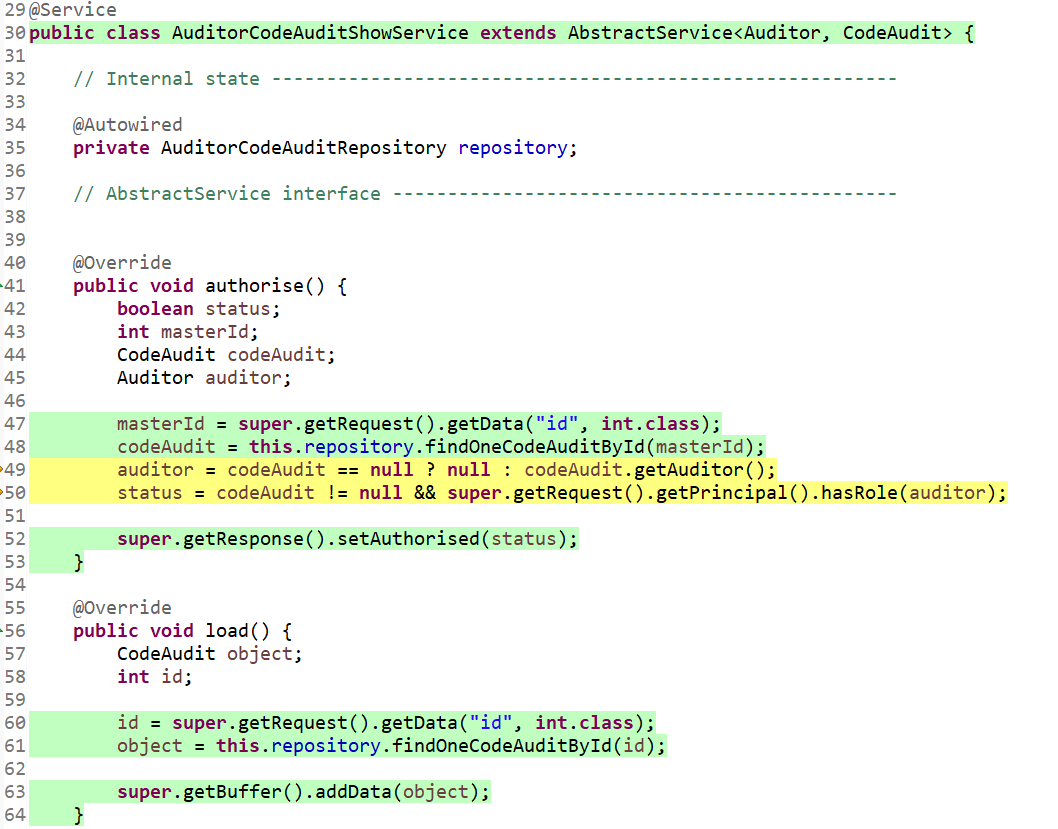
***Tests.safe***

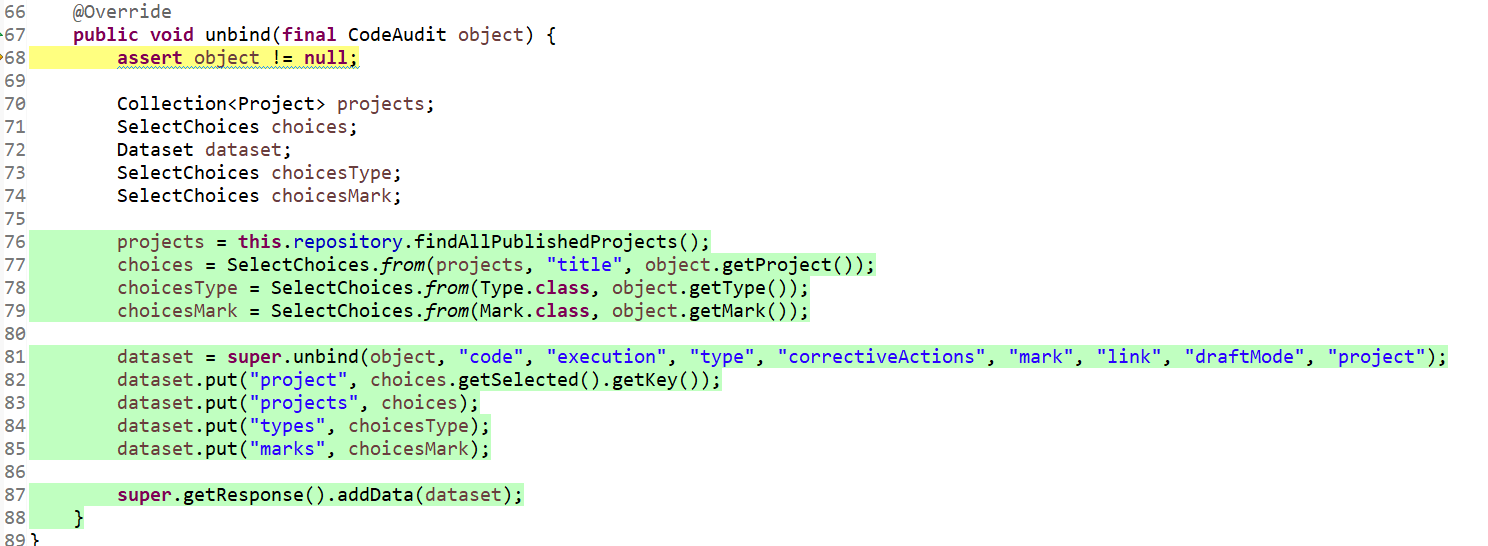
* Enseña los datos de un code Audit que pertenece al usuario
* Enseña los datos de un code audit publicado que pertenece al usuario

***Tests.hack***

* Intentar acceder a los datos de un code Audit sin estar loggeado
* Intentar acceder a los datos de un code Audit loggeado, pero no como auditor
* Intentar acceder a los datos de un code Audit loggeado como auditor, pero a los de uno que no te pertenece
* Intentar acceder a los datos de un code Audit inexistente

***Código ejecutado***

******

******

******

* Assert object != null: Linea obligatoria que sirve para verificar que el objeto no sea nulo.
* Condiciones de authorise: Se han ejecutado todas las alternativas por separado, faltando ciertas combinaciones concretas de valores sin cumplir combinadas. No es refactorizable, ya que todos los requisitos son necesarios para que asegurar un uso correcto y seguro de la aplicación

***Create***

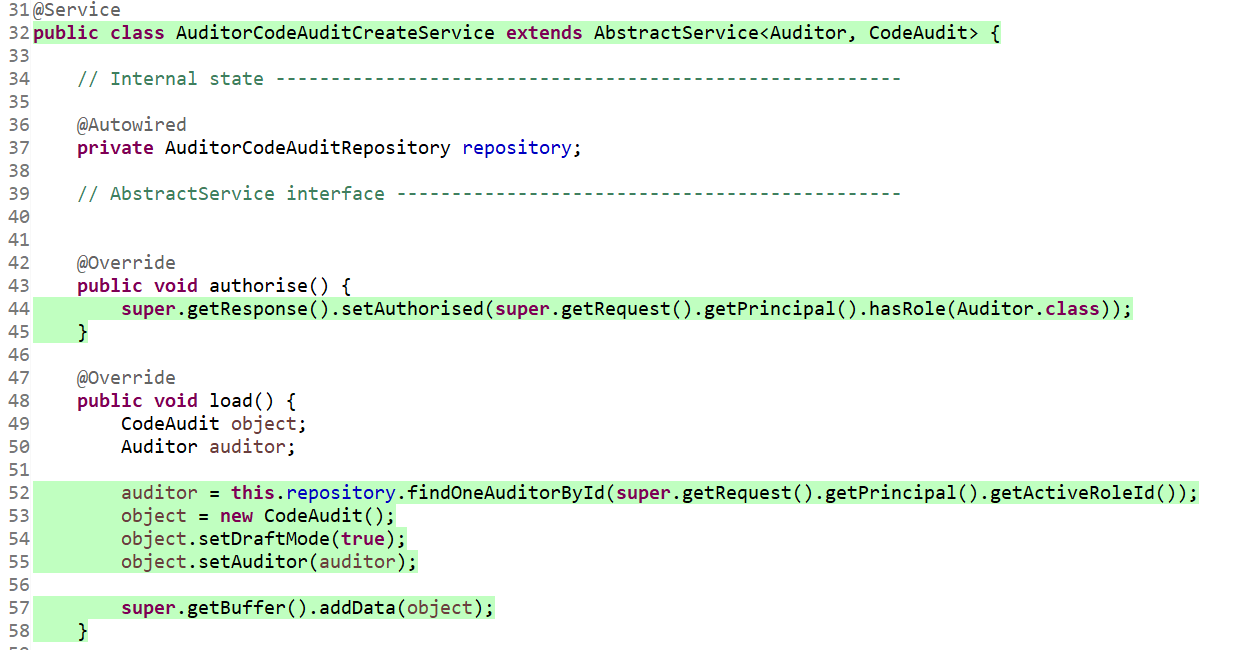
***Tests.safe***

* Crear un code Audit con los datos vacíos
* Crear un code Audit con variedad de datos erróneos en sus atributos
* Crear un code Audit con datos límite válidos e inválidos para sus atributos
* Crear un code Audit con variedad de datos válidos en sus atributos

***Tests.hack***

* Intentar crear un code Audit sin estar loggeado
* Intentar crear un code Audit loggeado, pero no como auditor
* Intentar crear un code Audit loggeado como auditor, pero probando a editar valores que serían ineditables por pantalla
* Intentar crear un code Audit loggeado como auditor, pero dando valores invalidos a atributos “choice”

***Código ejecutado***

******

******

******

******

* Assert object != null: Linea obligatoria que sirve para verificar que el objeto no sea nulo.

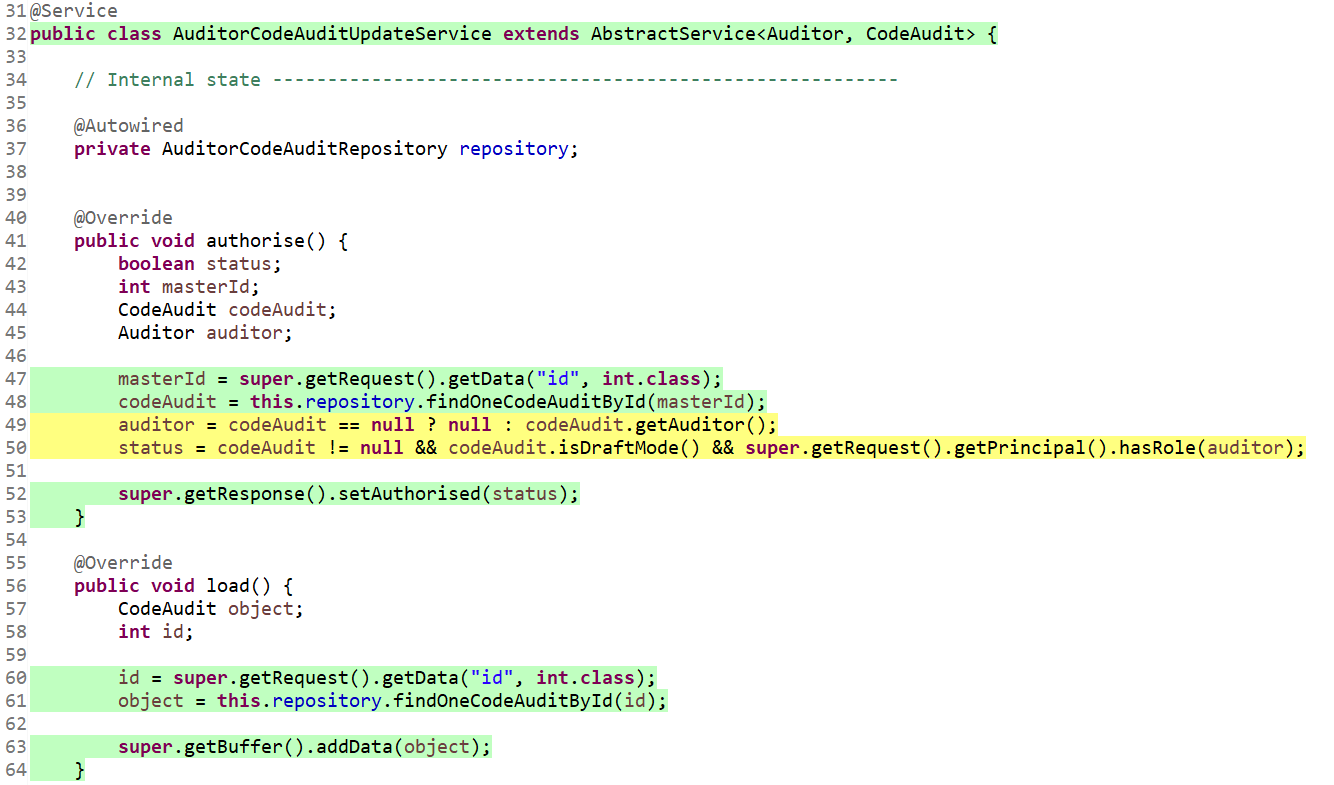
***Update***

***Tests.safe***

* Actualizar un code Audit con los datos vacíos
* Actualizar un code Audit con variedad de datos erróneos en sus atributos
* Actualizar un code Audit con variedad de datos límite válidos e inválidos para sus atributos
* Actualizar un code Audit con variedad de datos validos en sus atributos

***Tests.hack***

* Intentar actualizar un code Audit sin estar loggeado
* Intentar actualizar un code Audit loggeado, pero no como auditor
* Intentar actualizar un code Audit loggeado como auditor, pero en un code Audit que no te pertenece
* Intentar actualizar un code Audit loggeado como auditor, pero en un code Audit inexistente
* Intentar actualizar un code Audit loggeado como auditor, pero probando a editar valores que serían ineditables por pantalla
* Intentar actualizar un code Audit loggeado como auditor, pero dando valores invalidos a atributos “choice”
* Intentar actualizar un code Audit loggeado como auditor, pero un code Audit publicado

***Código ejecutado*** ******

******

******

* Assert object != null: Linea obligatoria que sirve para verificar que el objeto no sea nulo.
* Condiciones de authorise: Se han ejecutado todas las alternativas por separado, faltando ciertas combinaciones concretas de valores sin cumplir combinadas. No es refactorizable, ya que todos los requisitos son necesarios para que asegurar un uso correcto y seguro de la aplicación
* Validate de Project: Linea que sirve para evitar un hackeo al atributo proyect. Esta línea ha sido ejecutada en su totalidad en el servicio de create, donde tiene el mismo sentido su presencia, y funcionalidad en la aplicación, no teniendo variación ninguna en su resultado con respecto al anterior.

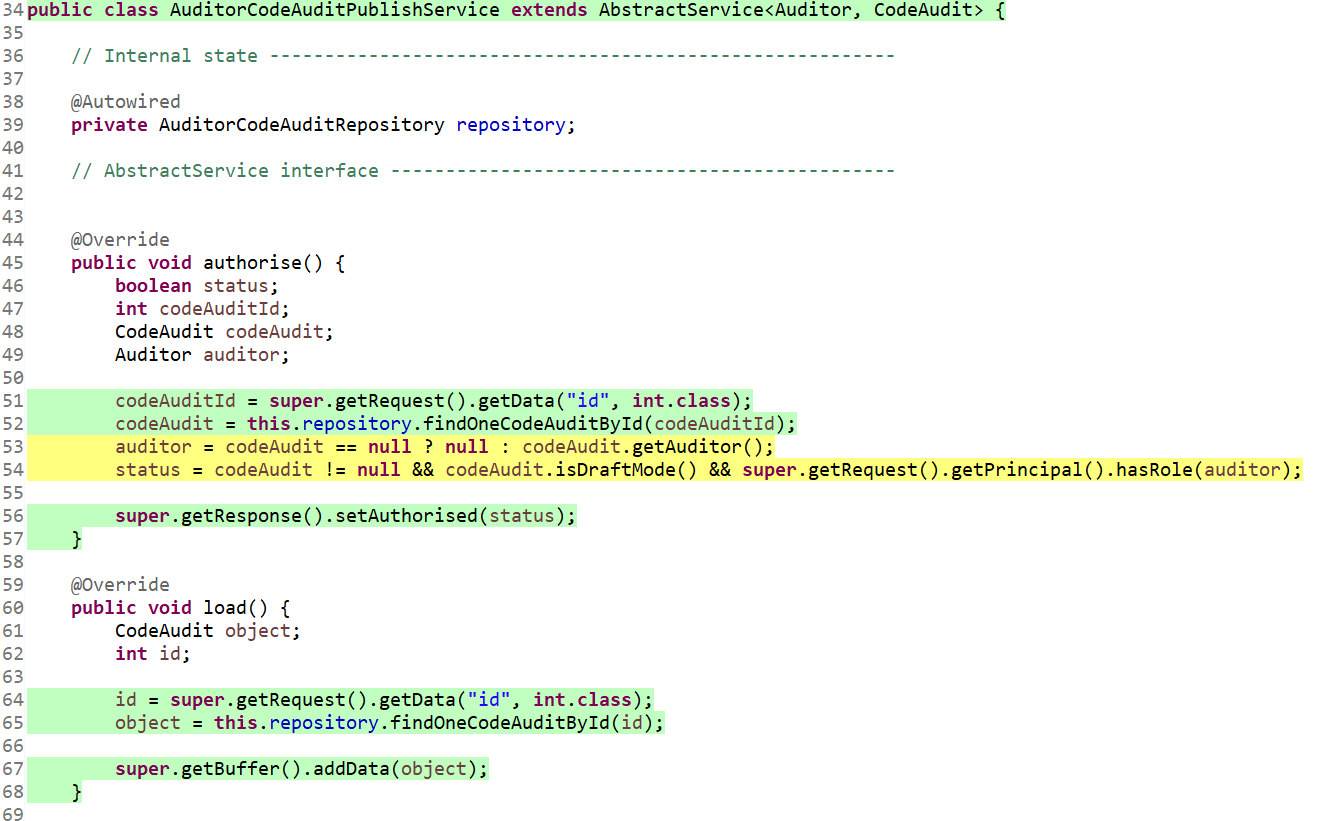
***Publish***

***Tests.safe***

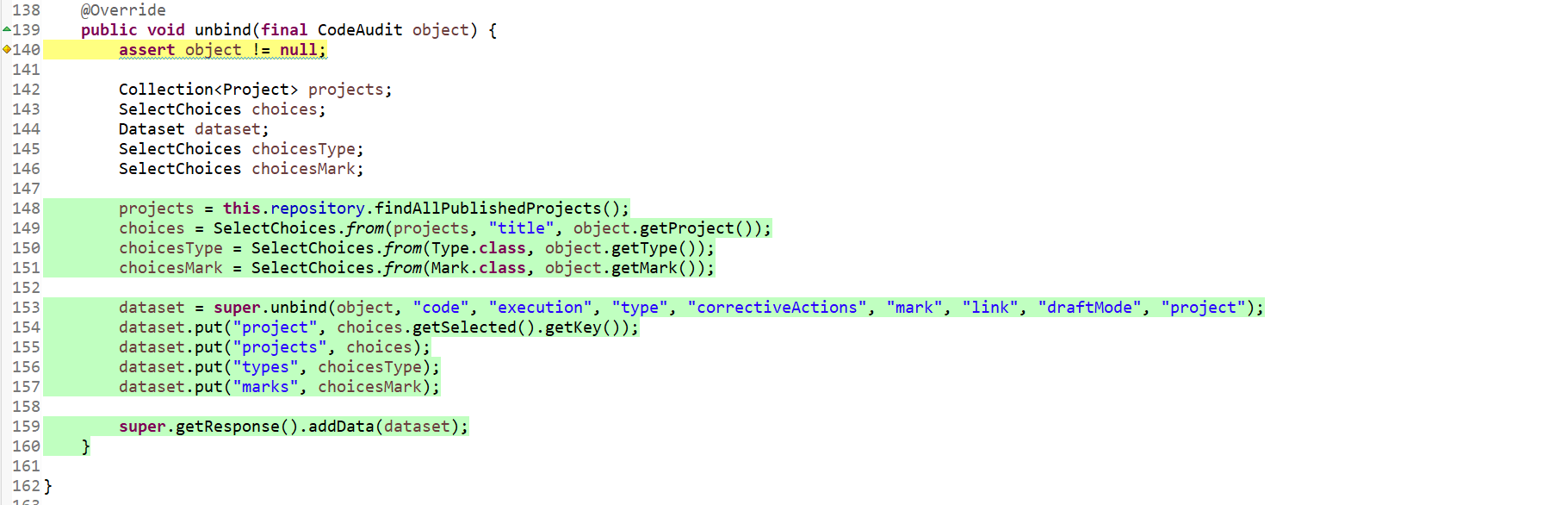
* Publicar un code Audit con los datos vacíos
* Publicar un code Audit con variedad de datos erróneos en sus atributos
* Publicar un code Audit con variedad de datos límite válidos e inválidos para sus atributos
* Publicar un code Audit con variedad de datos validos en sus atributos
* Intentar publicar un code Audit en condiciones en las que no es posible

***Tests.hack***

* Intentar publicar un code Audit sin estar loggeado
* Intentar publicar un code Audit loggeado, pero no como auditor
* Intentar publicar un code Audit loggeado como auditor, pero en un code Audit que no te pertenece
* Intentar publicar un code Audit loggeado como auditor, pero en un code Audit inexistente
* Intentar publicar un code Audit loggeado como auditor, pero probando a editar valores que serían ineditables por pantalla
* Intentar publicar un code Audit loggeado como auditor, pero dando valores invalidos a atributos “choice”
* Intentar publicar un code Audit loggeado como auditor, pero un code Audit publicado

***Código ejecutado***

****** 

******



* Assert object != null: Linea obligatoria que sirve para verificar que el objeto no sea nulo.
* Condiciones de authorise: Se han ejecutado todas las alternativas por separado, faltando ciertas combinaciones concretas de valores sin cumplir combinadas. No es refactorizable, ya que todos los requisitos son necesarios para que asegurar un uso correcto y seguro de la aplicación.
* Validate de code: Esta validación comprueba que un código no pueda usarse si ya es usado por otro code Audit, teniendo en cuenta que este puede ser el mismo que era anteriormente. No se ha comprobado una de las 4 alternativas (Que si se comprobó en update donde también esta implementado), que es que code sea blank, y que este tenga el mismo valor que tenía anteriormente, caso el cual por supuesto es contemplado y en el que la aplicación responde correctamente.
* Validate en mark: Este comprueba que la nota no puede valer F o F\_MINUS. Ambos casos son comprobados, y sale subrayado ya que espera un error en el propio buffer, que no puede llegar a darse ya que mark se actualiza automáticamente.
* Validate de Project: Linea que sirve para evitar un hackeo al atributo proyect. Esta línea ha sido ejecutada en su totalidad en el servicio de créate, donde tiene el mismo sentido su presencia, y funcionalidad en la aplicación, no teniendo variación ninguna en su resultado con respecto al anterior.
* Validate de \*: Este validate sirve para asegurar que un code Audit no pueda publicarse si no posee ningún Audit record. No se ejecuta totalmente debido a que el buffer no ha experimentado ningún error al probar este caso, pero es una línea que, como puede verse, cumple la funcionalidad requerida y, por tanto, no es refactorizable. Además, como puede verse en la captura, esta línea funciona correctamente en otro “Validate”, siendo que esta no da ningún problema.

***Delete***

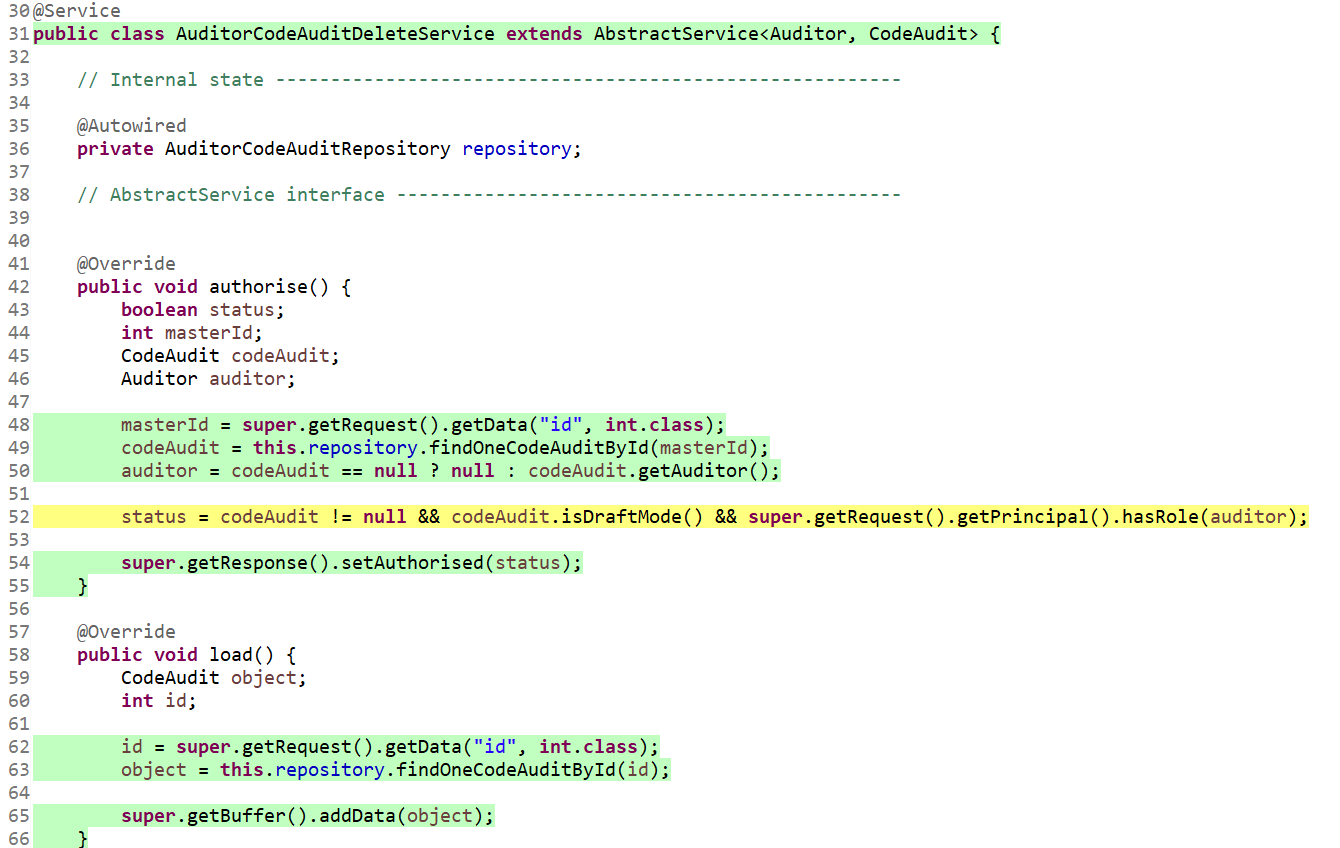
***Tests.safe***

* Eliminar un code Audit perteneciente al usuario
* Intentar eliminar un code Audit en condiciones en las que no es posible

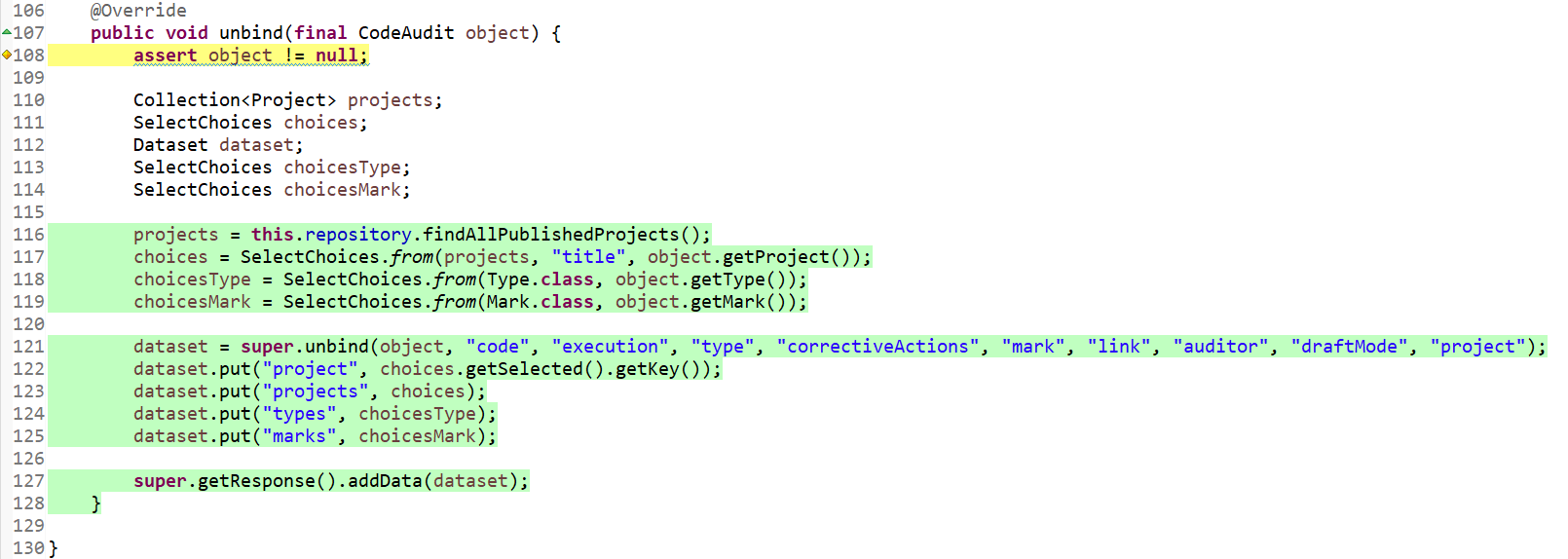
***Tests.hack***

* Intentar eliminar un code Audit sin estar loggeado
* Intentar eliminar un code Audit loggeado, pero no como auditor
* Intentar eliminar un code Audit loggeado como auditor, pero en un code Audit que no te pertenece
* Intentar eliminar un code Audit loggeado como auditor, pero en un code Audit inexistente
* Intentar eliminar un code Audit loggeado como auditor, pero un code Audit publicado

***Código ejecutado***

******

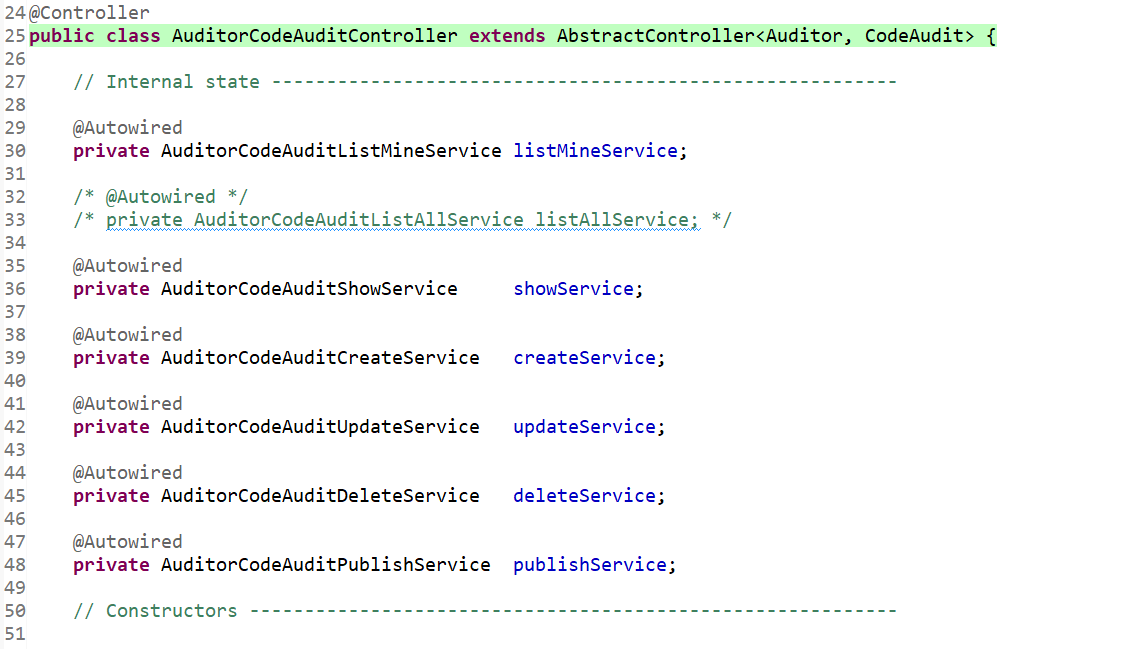
******

******

******

* Assert object != null: Linea obligatoria que sirve para verificar que el objeto no sea nulo.
* Condiciones de authorise: Se han ejecutado todas las alternativas por separado, faltando ciertas combinaciones concretas de valores sin cumplir combinadas. No es refactorizable, ya que todos los requisitos son necesarios para que asegurar un uso correcto y seguro de la aplicación

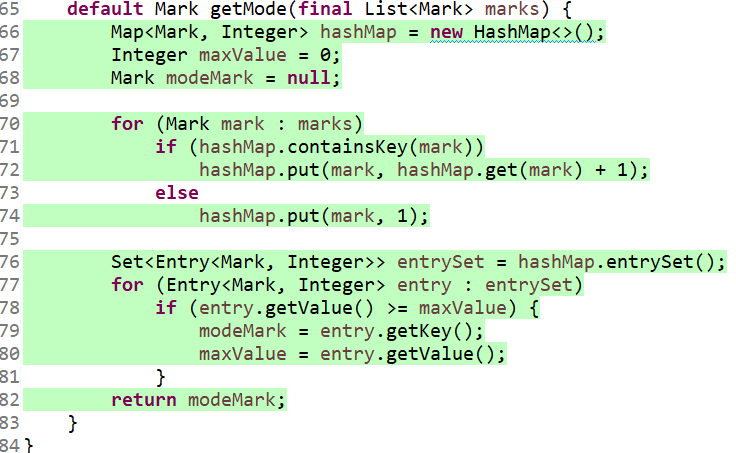
***Extra: Controlador***

****** ******

******

Se observa cómo se ejecuta todo el código correctamente

***Extra: Repositorio***

******



Se ejecuta el código entero, siendo esta una función aparte de las querys que sirve para calcular la moda de las mark de un code-audit al ser publicado.

***Audit Record***

***List***

***Tests.safe***

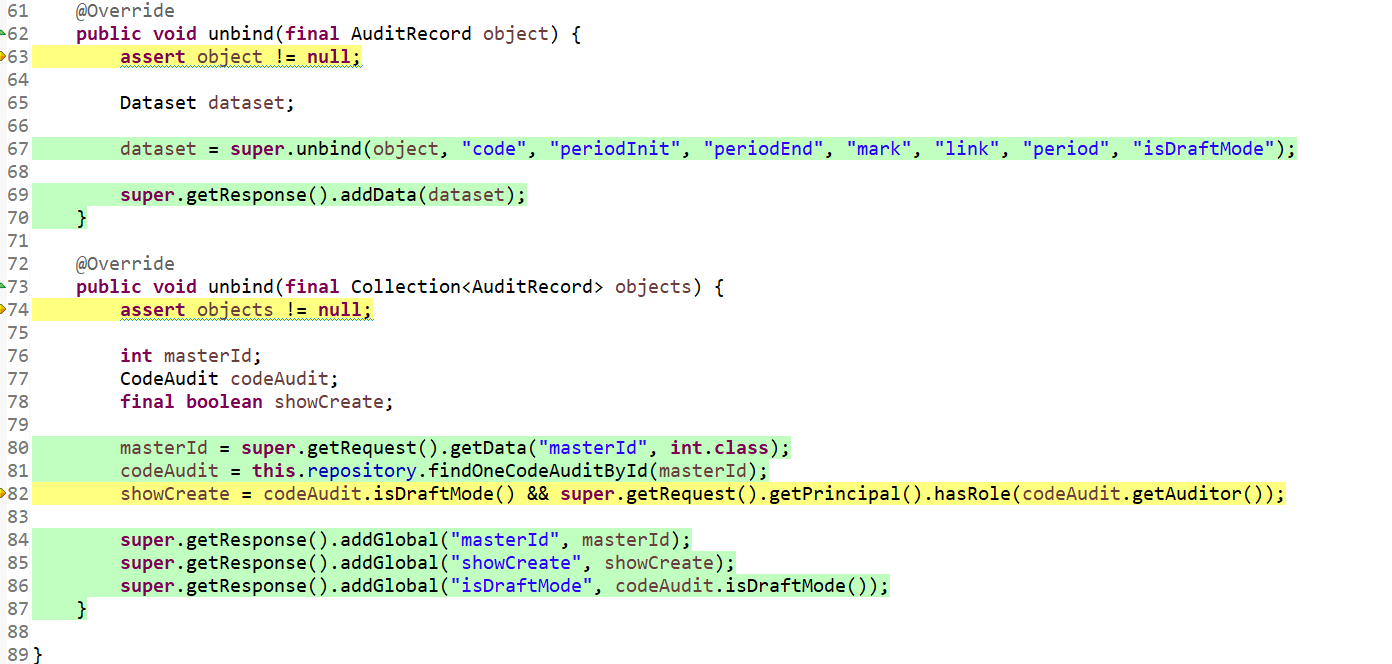
* Listado de todos los Audit Record de un Code Audit que pertenece a un usuario
* Listado de Audit Record vacío

***Tests.hack***

* Intentar acceder a un listado de Audit Record sin estar loggeado
* Intentar acceder a un listado de Audit Record loggeado, pero no como auditor
* Intentar acceder a un listado de Audit Record loggeado como auditor, pero de un code Audit del que el auditor no es propietario
* Intentar acceder a un listado de Audit Record loggeado como auditor, pero de un code Audit inexistente

***Código ejecutado***

******

******

******

* Assert object ¡= null: Linea obligatoria que sirve para verificar que el objeto no sea null
* showCreate: Linea que sirve para indicar cuando se mostrará por pantalla el botón de showCreate. Se ejecuta parcialmente ya que, si las condiciones no se dan, este no será mostrado, pero es necesario para el funcionamiento del servicio, por lo que no debe refactorizarse

***Show***

***Tests.safe***

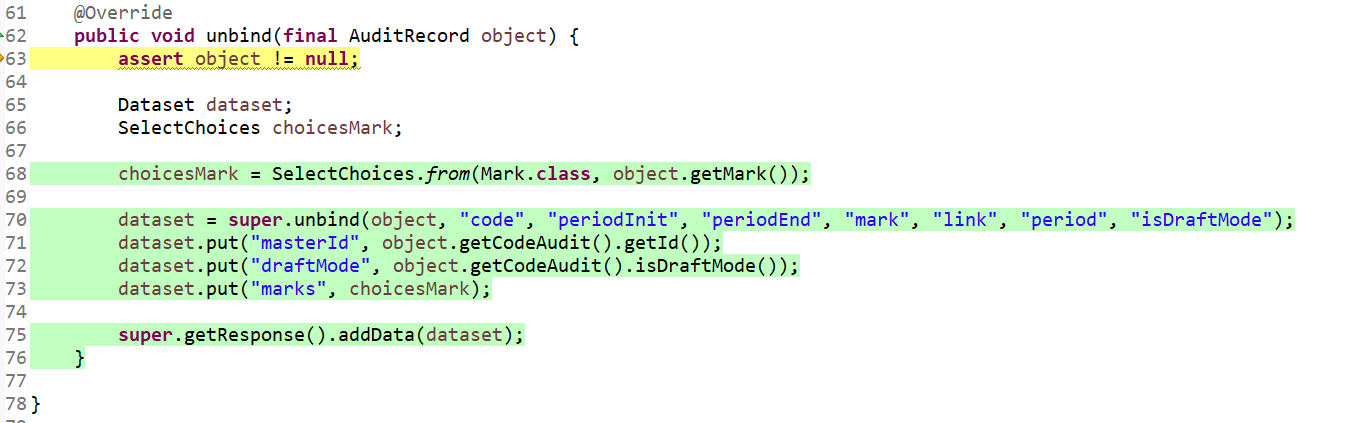
* Enseñar los datos de un Audit Record perteneciente al usuario
* Enseñar los datos de un Audit Record publicado perteneciente al usuario

***Tests.hack***

* Intentar acceder a los datos de un Audit Record sin estar loggeado
* Intentar acceder a los datos de un Audit Record loggeado, pero no como auditor
* Intentar acceder a los datos de un Audit Record loggeado como auditor, pero a los de uno que no te pertenece
* Intentar acceder a los datos de un Audit Record inexistente

***Código Ejecutado***

******

******



* Assert object ¡= null: Linea obligatoria que sirve para verificar que el objeto no sea nulo.
* Condiciones de authorise: Se han ejecutado todas las alternativas por separado, faltando ciertas combinaciones concretas de valores sin cumplir combinadas. No es refactorizable, ya que todos los requisitos son necesarios para que asegurar un uso correcto y seguro de la aplicación

***Create***

***Tests.safe***

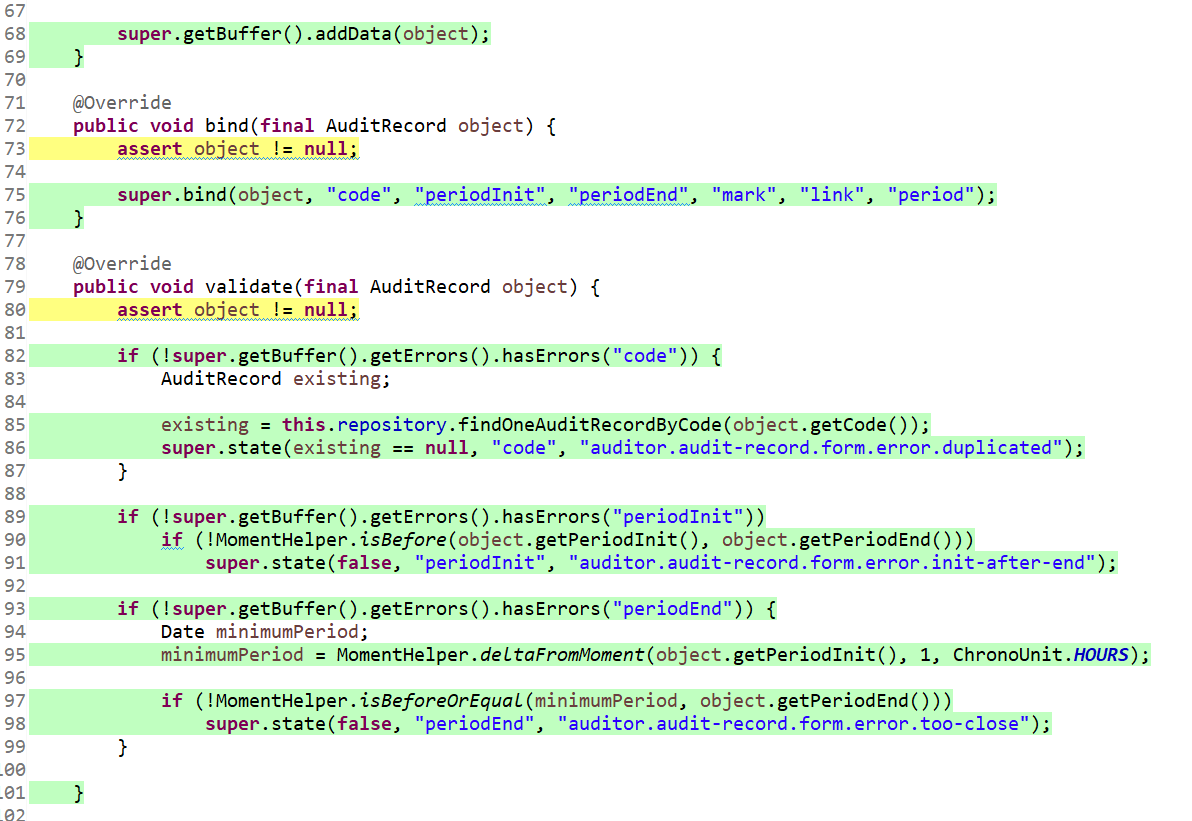
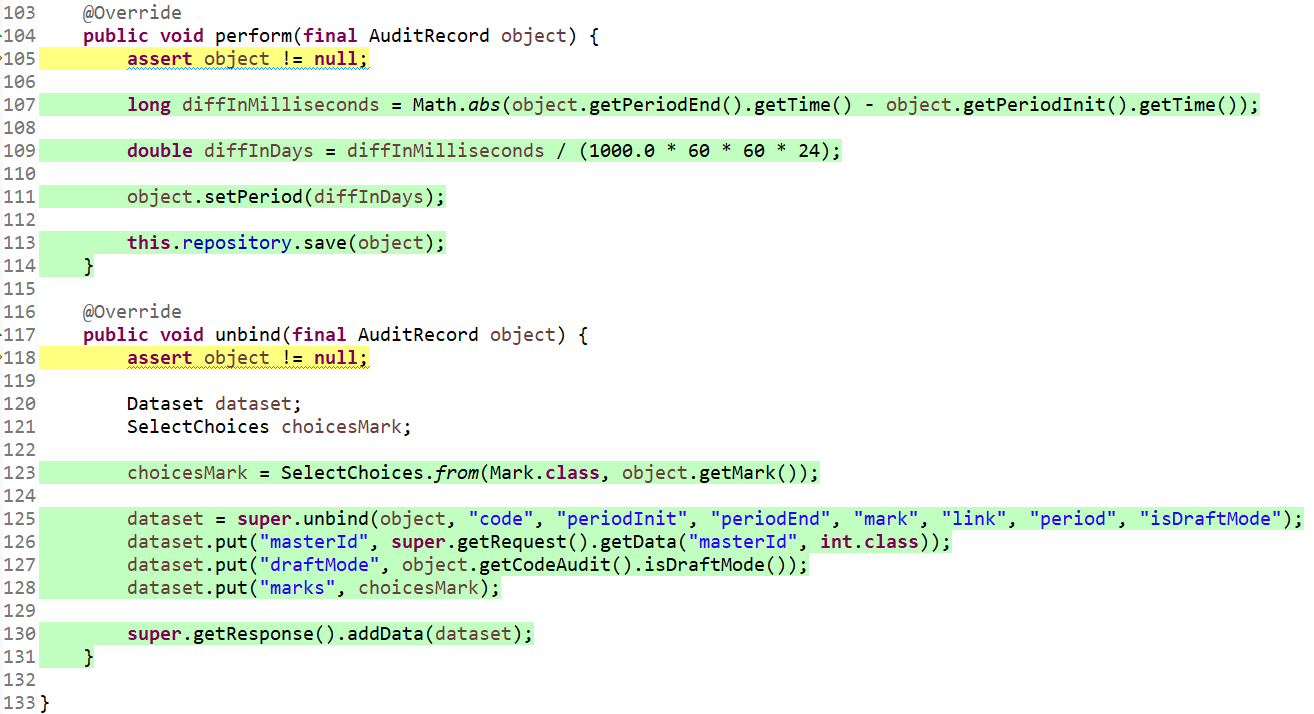
* Crear un Audit Record con los datos vacíos
* Crear un Audit Record con variedad de datos erróneos en sus atributos
* Crear un Audit Record con datos límite válidos e inválidos para sus atributos
* Crear un Audit Record con variedad de datos en sus atributos

***Tests.hack***

* Intentar crear un Audit Record sin estar loggeado
* Intentar crear un Audit Record loggeado, pero no como auditor
* Intentar crear un Audit Record loggeado como auditor, pero en un code Audit que no es el del usuario
* Intentar crear un Audit Record loggeado como auditor, pero en un code Audit publicado
* Intentar crear un Audit Record loggeado como auditor, pero probando a editar valores que serían ineditables por pantalla
* Intentar crear un Audit Record loggeado como auditor, pero dando valores invalidos a atributos “choice”

***Código Ejecutado***

******

****** ******



* Assert object ¡= null: Linea obligatoria que sirve para verificar que el objeto no sea nulo.
* Condiciones de authorise: Se han ejecutado todas las alternativas por separado, faltando ciertas combinaciones concretas de valores sin cumplir combinadas. No es refactorizable, ya que todos los requisitos son necesarios para que asegurar un uso correcto y seguro de la aplicación

***Update***

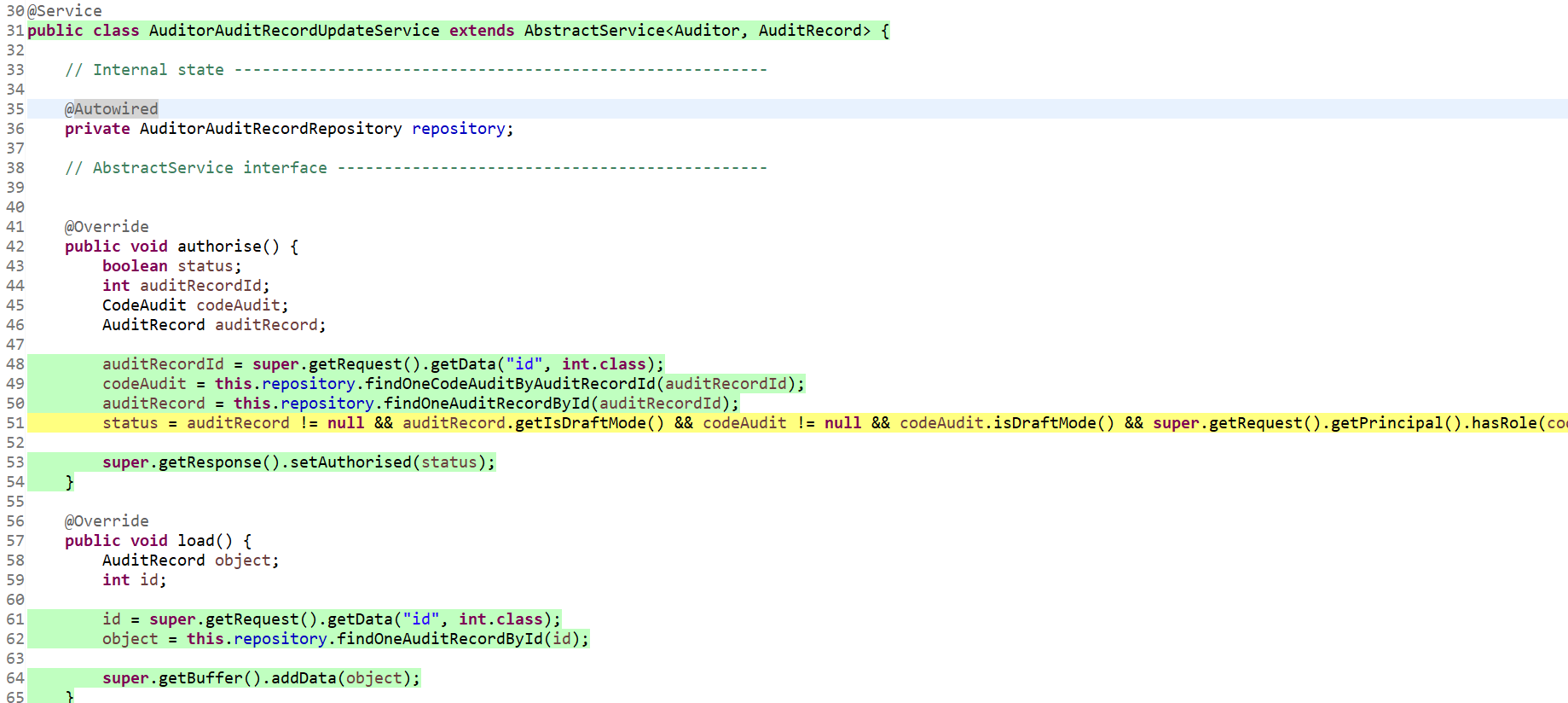
***Tests.safe***

* Actualizar un Audit Record con los datos vacíos
* Actualizar un Audit Record con variedad de datos erróneos en sus atributos
* Actualizar un Audit Record con variedad de datos límite válidos e inválidos para sus atributos
* Actualizar un Audit Record con variedad de datos validos en sus atributos

***Tests.hack***

* Intentar actualizar un Audit Record sin estar loggeado
* Intentar actualizar un Audit Record loggeado, pero no como auditor
* Intentar actualizar un Audit Record loggeado como auditor, pero en un Audit Record que no te pertenece
* Intentar actualizar un Audit Record loggeado como auditor, pero en un Audit Record inexistente
* Intentar actualizar un Audit Record loggeado como auditor, pero probando a editar valores que serían ineditables por pantalla
* Intentar actualizar un Audit Record loggeado como auditor, pero dando valores invalidos a atributos “choice”
* Intentar actualizar un Audit Record loggeado como auditor, pero un Audit Record publicado

***Código Ejecutado***

******

******

******

******



* Assert object ¡= null: Linea obligatoria que sirve para verificar que el objeto no sea nulo.
* Condiciones de authorise: Se han ejecutado todas las alternativas por separado, faltando ciertas combinaciones concretas de valores sin cumplir combinadas. No es refactorizable, ya que todos los requisitos son necesarios para que asegurar un uso correcto y seguro de la aplicación

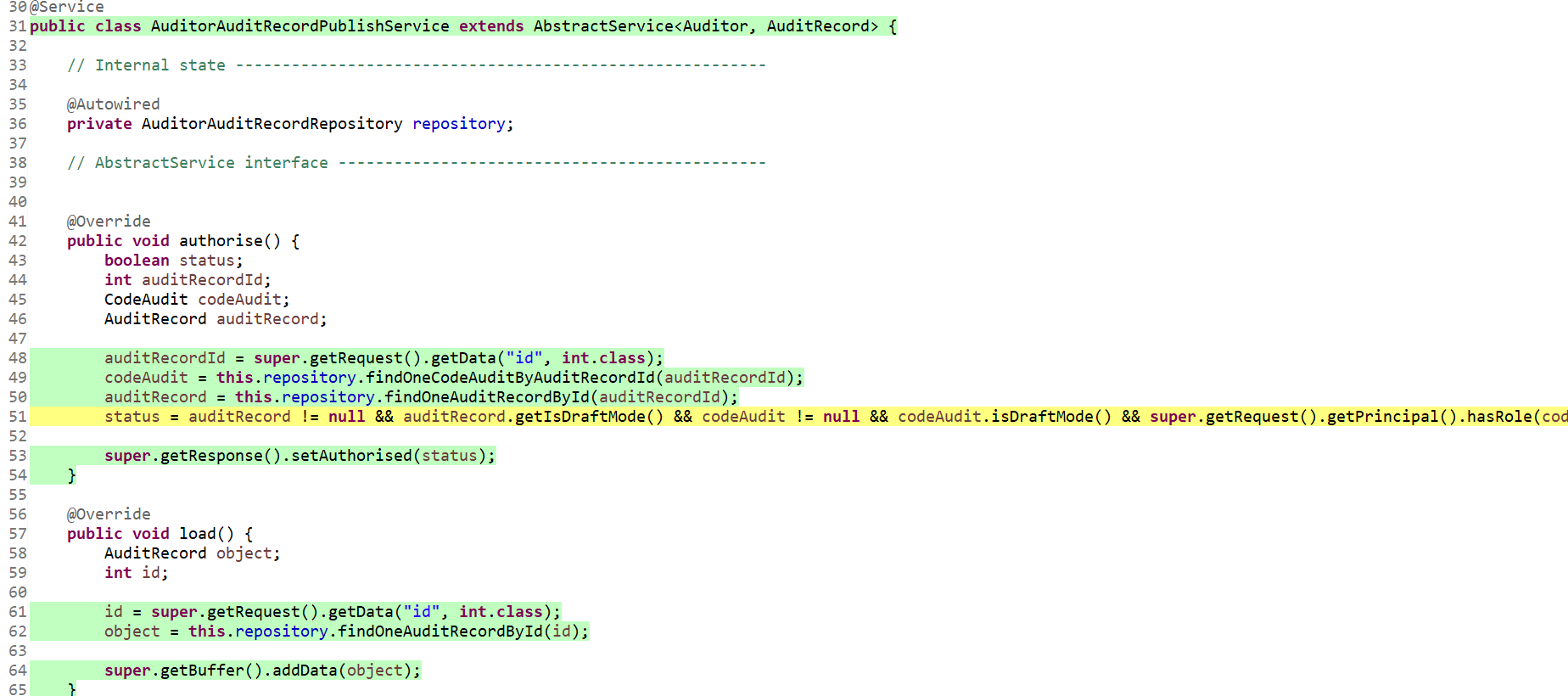
***Publish***

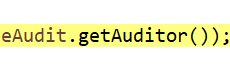
***Tests.safe***

* Publicar un Audit Record con los datos vacíos
* Publicar un Audit Record con variedad de datos erróneos en sus atributos
* Publicar un Audit Record con variedad de datos límite válidos e inválidos para sus atributos
* Publicar un Audit Record con variedad de datos validos en sus atributos

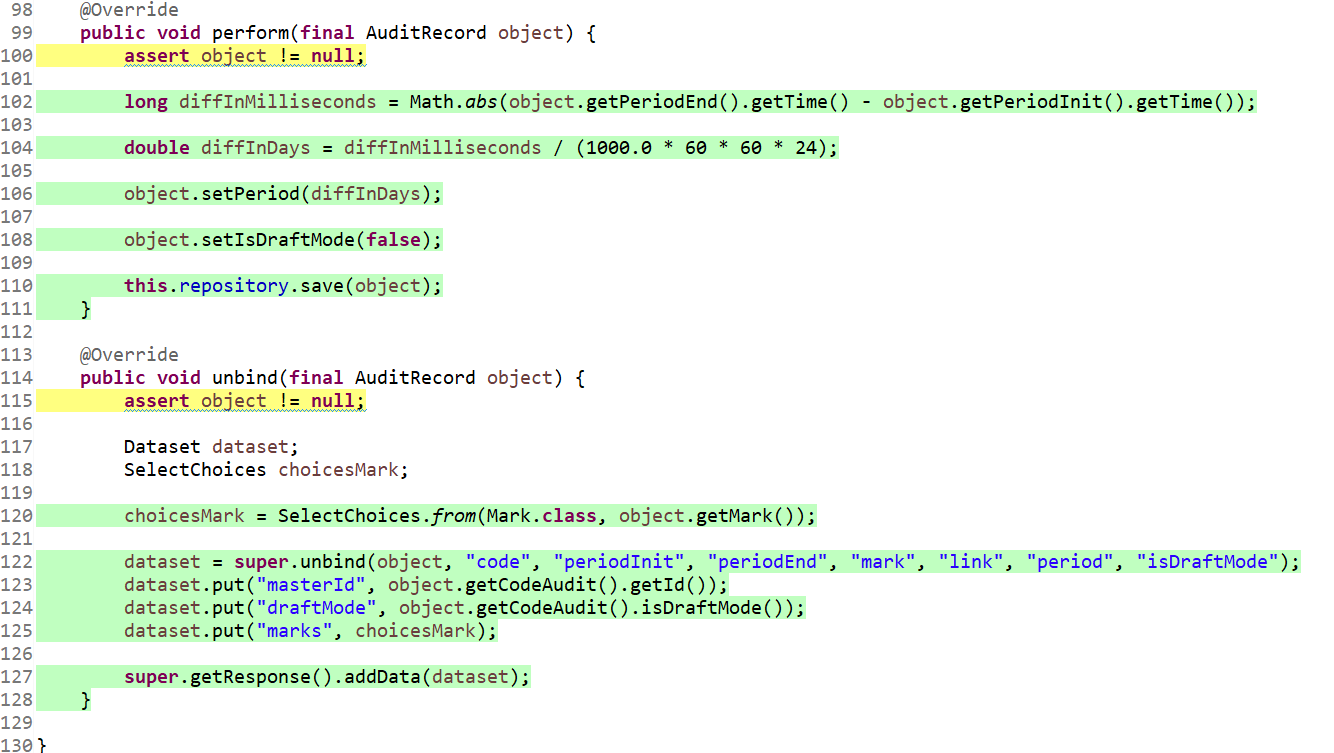
***Tests.hack***

* Intentar publicar un Audit Record sin estar loggeado
* Intentar publicar un Audit Record loggeado, pero no como auditor
* Intentar publicar un Audit Record loggeado como auditor, pero en un Audit Record que no te pertenece
* Intentar publicar un Audit Record loggeado como auditor, pero en un Audit Record inexistente
* Intentar publicar un Audit Record loggeado como auditor, pero probando a editar valores que serían ineditables por pantalla
* Intentar publicar un Audit Record loggeado como auditor, pero dando valores invalidos a atributos “choice”
* Intentar publicar un Audit Record loggeado como auditor, pero un Audit Record publicado

***Código Ejecutado***

******

******

******



* Assert object ¡= null: Linea obligatoria que sirve para verificar que el objeto no sea nulo.
* Condiciones de authorise: Se han ejecutado todas las alternativas por separado, faltando ciertas combinaciones concretas de valores sin cumplir combinadas. No es refactorizable, ya que todos los requisitos son necesarios para que asegurar un uso correcto y seguro de la aplicación
* Validate de code: Como se indico antes esta validación comprueba que un código no pueda usarse si ya es usado por otro code Audit, teniendo en cuenta que este puede ser el mismo que era anteriormente, y ambas posibilidades son necesarias, por lo que no es refactorizable

***Delete***

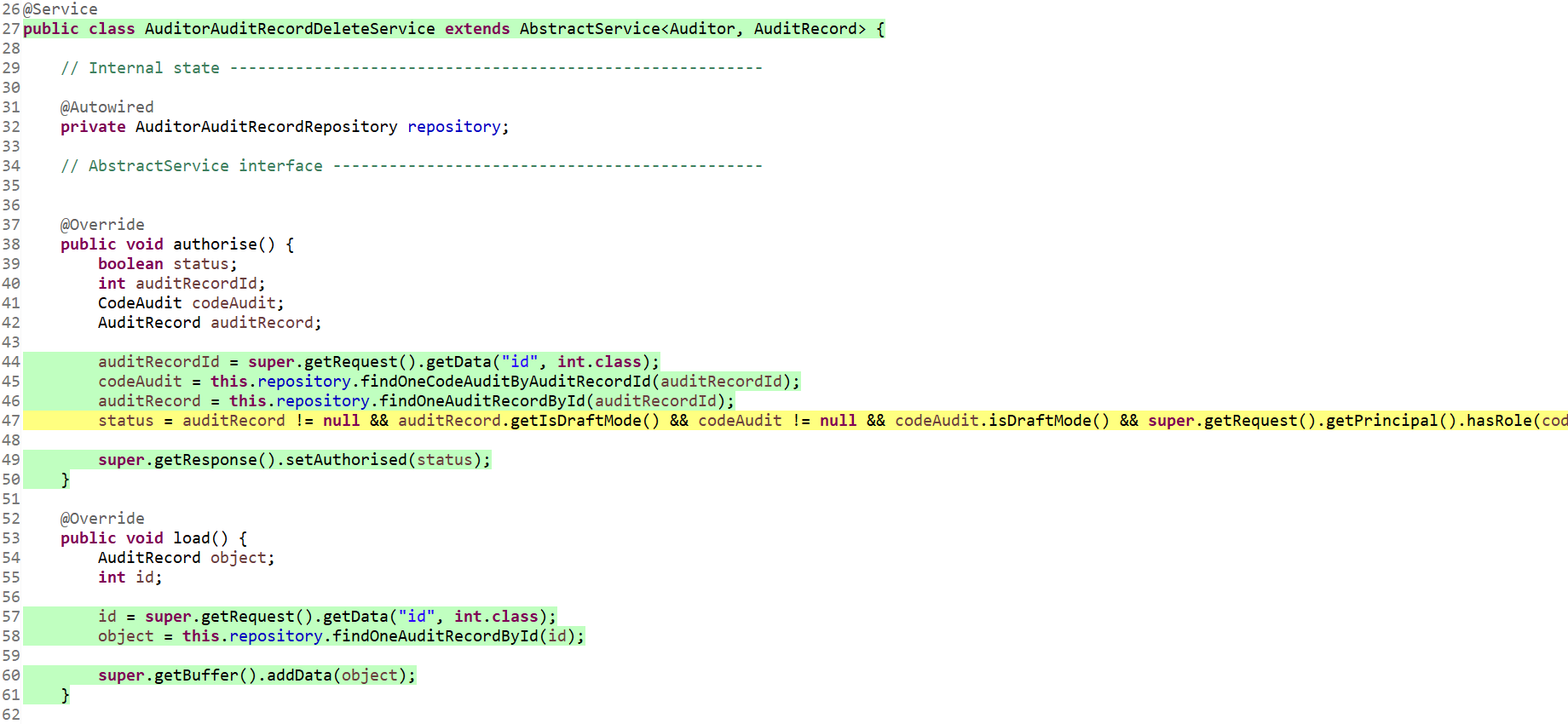
***Tests.safe***

* Eliminar un Audit Record perteneciente al usuario

***Tests.hack***

* Intentar eliminar un Audit Record sin estar loggeado
* Intentar eliminar un Audit Record loggeado, pero no como auditor
* Intentar eliminar un Audit Record loggeado como auditor, pero en un Audit Record que no te pertenece
* Intentar eliminar un Audit Record loggeado como auditor, pero en un Audit Record inexistente
* Intentar eliminar un Audit Record loggeado como auditor, pero un Audit Record publicado

***Código Ejecutado***

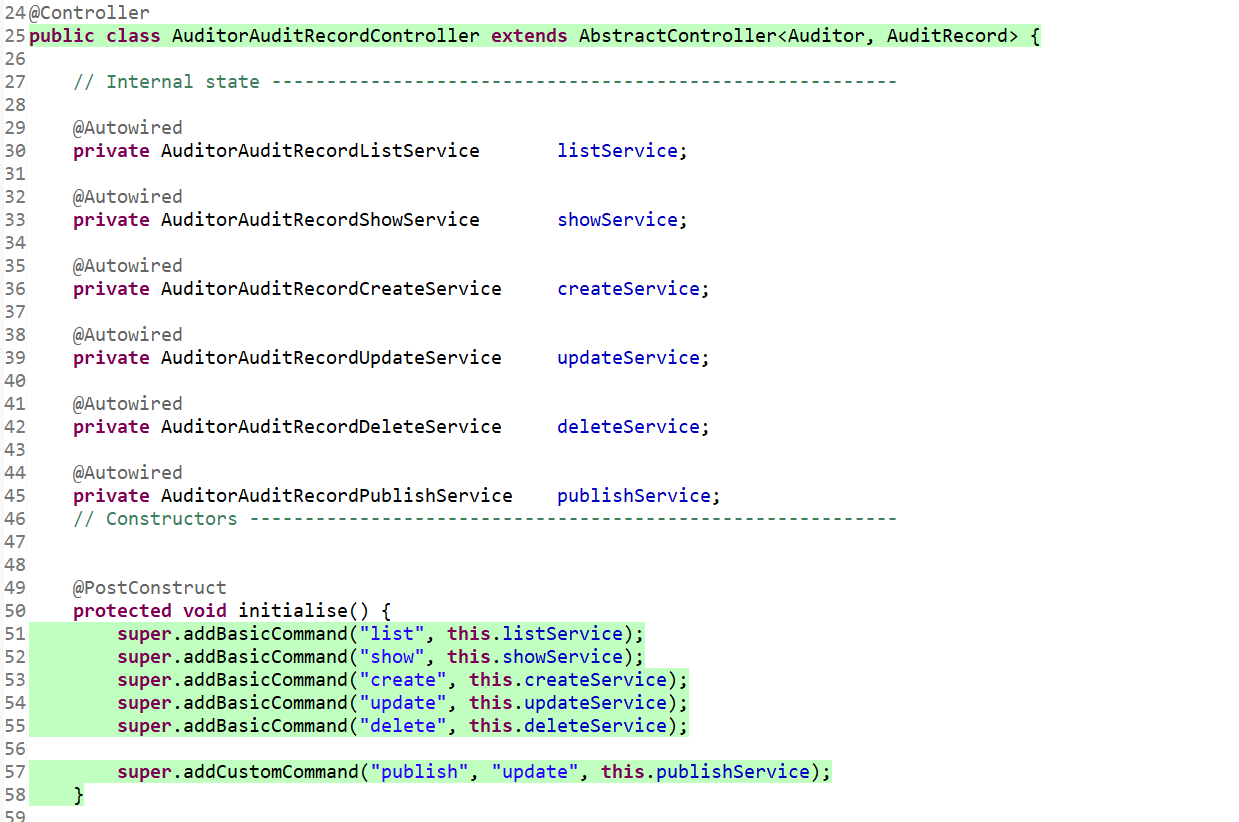
******

******

******

* Assert object ¡= null: Linea obligatoria que sirve para verificar que el objeto no sea nulo.
* Condiciones de authorise: Se han ejecutado todas las alternativas por separado, faltando ciertas combinaciones concretas de valores sin cumplir combinadas. No es refactorizable, ya que todos los requisitos son necesarios para que asegurar un uso correcto y seguro de la aplicación

***Extra: Controlador***

******



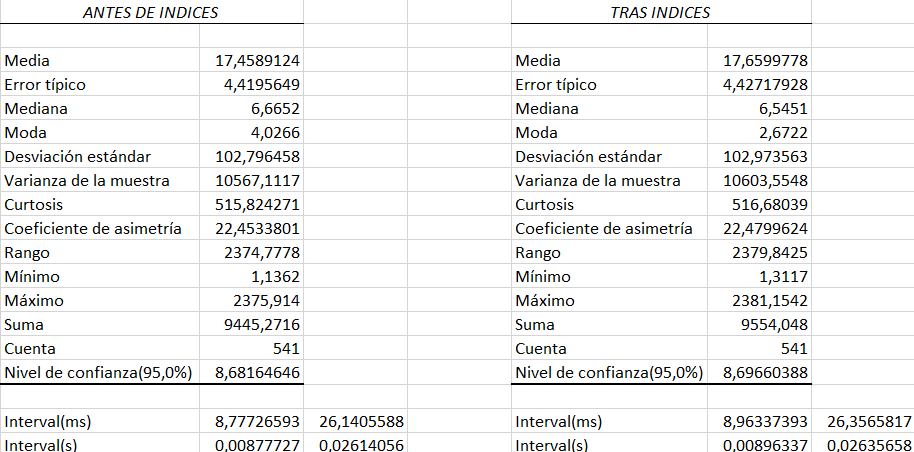
Se observa cómo se ejecuta todo el código correctamente

***Analisis de desempeño***

En primer lugar es necesario introducir sobre que trata este análisis de desempeño realizado. Se han llevado a cabo varias pruebas con fin de analizar los distintos rendimientos entorno a la traza de los test de “Code-Audit” y “Audit-Record”. En una de ellas, no se utilizaron los índices necesarios para optimizar las consultas SQL. En otra prueba, sí se aplicaron los índices mencionados anteriormente. Estas dos se realizaron desde mi equipo de trabajo habitual. Tras esto, volvimos a realizar una tercera prueba en el ordenador de uno de los miembros del grupo (con índices), ya que los resultados pueden variar dependiendo del equipo en el que se ejecuten.

**Comparación antes y después de los índices:**

En este apartado se ha comparado el resultado de los datos obtenidos antes (imagen 1) y después (imagen 2) de añadirse índices.



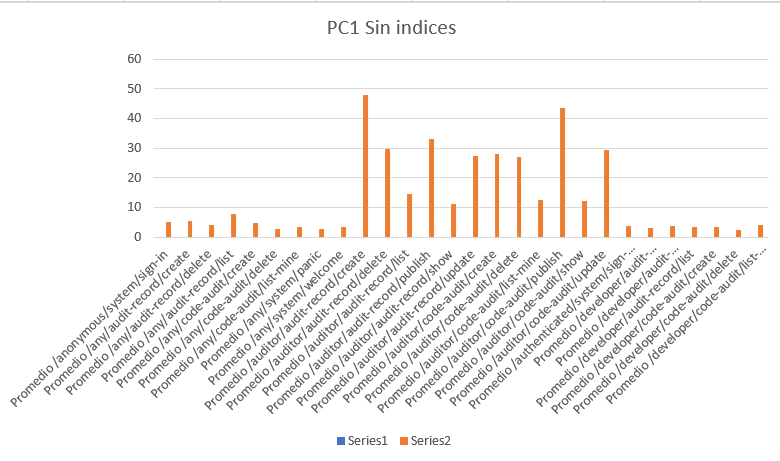
Estos son los resultados obtenidos utilizando la herramienta de "Estadística Descriptiva".

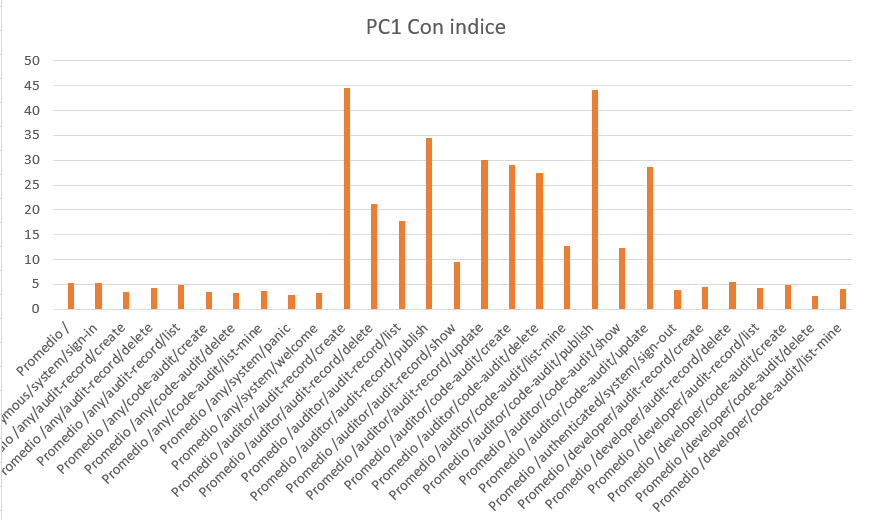
Como aclaración antes del análisis, es necesario indicar que el tamaño de la varianza es tan amplio debido a que las consultas “administrator/system/shut-down” han sido con amplia diferencia las de mayor duración a la hora de ejecutarse, lo que ha hecho que se distorsionasen los resultados con respecto a las demás. Ante esto, se han suprimido en el gráfico para que puedan observarse correctamente, pero han permanecido en el análisis estadístico y en la comparativa de prestaciones para mostrar la realidad de los resultados.

En primer lugar, podemos observar como la media del conjunto sin índices (17,45) es bastante similar que la que posee los índices (17,65), siendo levemente superiores en el primer caso.

Si seguimos comparando los datos de mediana (S.I – 6,66, C.I – 6,54), varianza (S.I - 10567,11, C.I – 10606,55), desviación estándar (S.I – 102,72, C.I – 102,97), curtosis (S.I – 515,82, C.I – 516,68) e intervalos son todos muy similares, siendo muy levemente favorables en el conjunto sin índices, por lo que todo hace parecer que el rendimiento permanece constante.

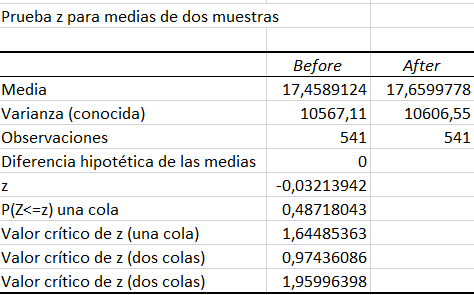
Observamos ahora los histogramas:





En ambos histogramas se observa una forma similar, mostrando una relación comparable entre las barras. Se observa una leve variación en alguna de las consultas, pero, aún con todo esto, no podemos concluir que estos no hayan supuesto una variación en su rendimiento significativa.

Procedemos al último cálculo estadístico:

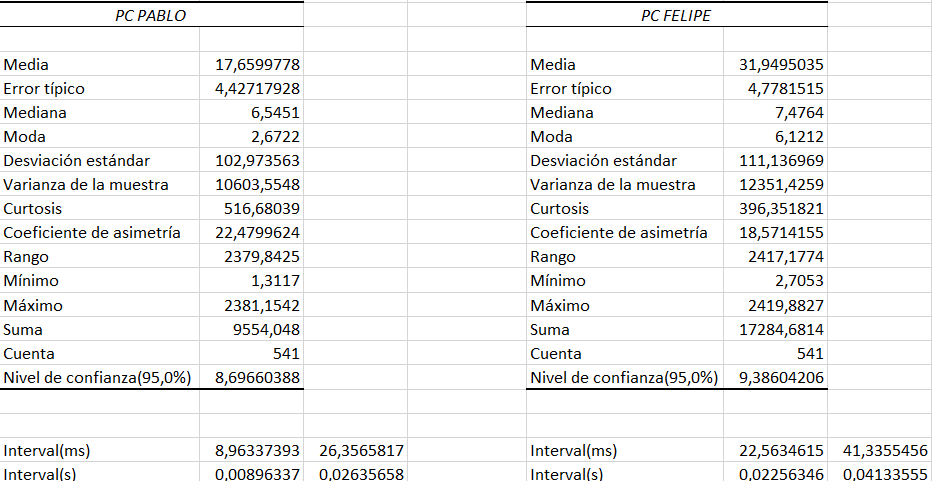


Hemos realizamos un cálculo estadístico de prueba Z para medias de dos muestras, siendo la columna before la correspondiente a las pruebas antes de añadir los índices, y after tras ser añadidos.

Nos enfocamos en el primer valor “Valor critico de z (dos colas)”, que es de 0,97. Comparamos con alfa (1 – alfa). Por tanto, alfa es 0.05. Como el valor critico no se encuentra en el intervalo [0,0.5] podemos concluir que:  
  
**No hay una diferencia significativa estadísticamente entre los conjuntos desde una perspectiva global.** Esto quiere decir que, aunque hay diferencias, estas no suponen una mejora significativa en el rendimiento.

**Comparación de dos ordenadores distintos optimizados**:

En este apartado se ha comparado las prestaciones de mi ordenador (PC1) con las de mi compañero, felsolagu (PC2).



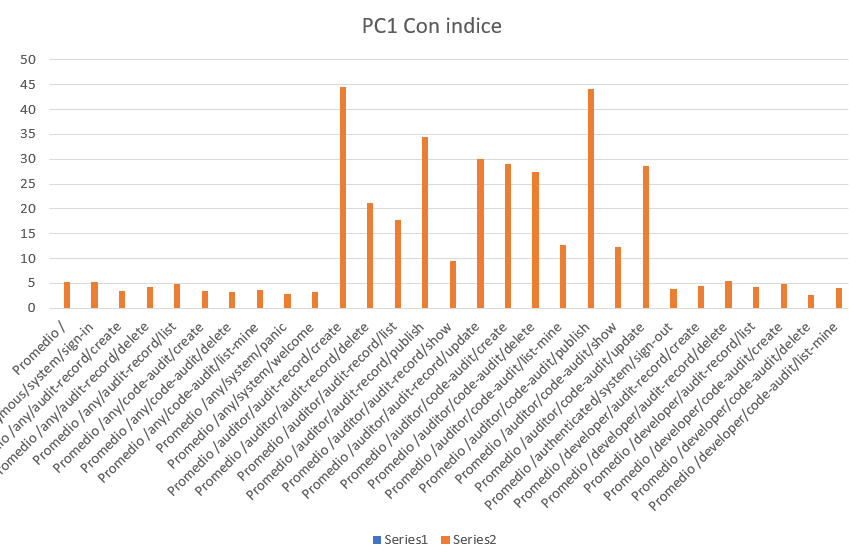
Estos son los resultados obtenidos utilizando la herramienta de "Estadística Descriptiva".

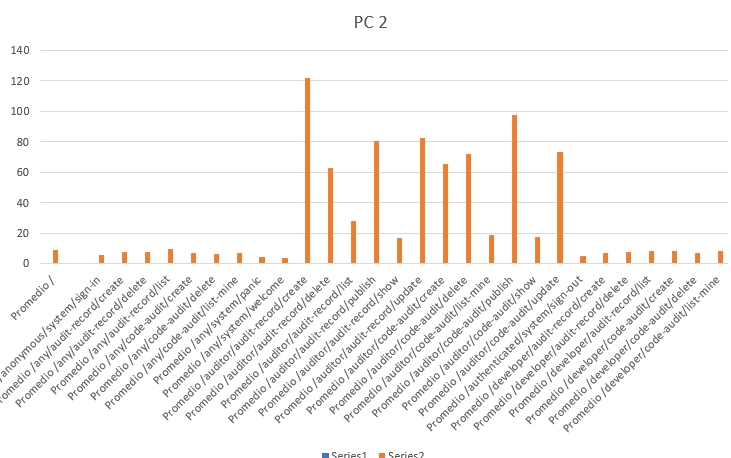
Se observa como hay una gran diferencia entre los valores de media de mi PC (17.65) y de mi compañero (31.94) lo que parece indicar que el rendimiento es mayor en mi equipo.

Si seguimos observando valores como la mediana (PC 1 = 6.54, PC 2 = 7.47), varianza (PC1 – 10606.55, PC2 – 12351.42), desviación estándar (PC1 – 102.97, PC2 – 111.13), curtosis (PC1 – 516,68, PC2 – 396.35) o los intervalos, podemos observar que tienen una gran diferencia entre ellos, sugiriendo la siguiente conclusión:

Mi PC parece poseer un mejor rendimiento que el de mi compañero en el tiempo de ejecución de las pruebas, por lo que debe tener mejores prestaciones.

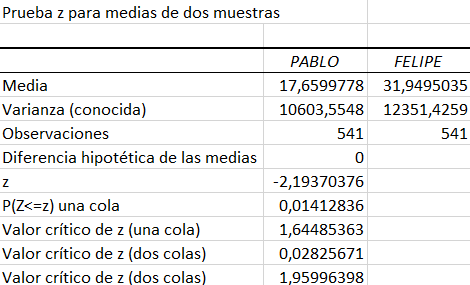
Pasamos ahora a la observación de los histogramas:





Si entramos en detalle, podemos observar que hay una diferencia enorme entre los tiempos de ejecución de las distintas consultas en mi equipo y el de mi compañero, llegando a duplicarse en algunos de estas. Podemos confirmar así los datos de estadística descriptiva que se ofrecían anteriormente, en los que veíamos como el tiempo de ejecución promedio es mucho menor en mi ordenador.

Por último, realizamos un cálculo estadístico de prueba Z para medias de dos muestras:



Con este cálculo buscamos entender si hay una diferencia significativa entre las medias de los datos de ambos equipos.

Nos fijamos en el “Valor crítico de z (dos colas)”. Este es 0.028 aproximadamente. Debemos compararlo con alfa, cuyo valor es “(1 – nivel de confianza)”. El nivel de confianza es de 95%, por lo que alfa es 0.05.

Podemos ver como nuestro valor crítico está dentro del intervalo [0, 0.05], por lo que sacamos como conclusión lo siguiente:  
  
**Existe una diferencia estadísticamente significativa** entre los dos conjuntos, siendo mi PC como mencionamos anteriormente el que ofrece un mejor rendimiento.

***Conclusiones***

En el desarrollo del proceso de testing nos hemos visto envueltos en un desarrollo amplio de test, gracias a los cuales hemos podido probar el correcto funcionamiento de nuestro sistema, asegurando que este es de calidad. También nos ha servido para lograr detectar errores, desde el más simple al más crítico, de los cuales no éramos conscientes. Y, además, han servido para, tras editar ciertas partes de los servicios, asegurar que el sistema seguía funcionando correctamente y no había ningún tipo de conflicto con nuevas implementaciones en ellos.

Así, se puede garantizar la solidez del proyecto, que cumple con los requisitos solicitados por el cliente, garantizando el funcionamiento pretendido y la seguridad necesaria para este.

Por otro lado, el uso de herramientas estadísticas e índices, y la comparación de la ejecución de los test en dos sistemas diferentes, han servido para realizar una evaluación objetiva del rendimiento, observando si el uso de estos índices realmente tenía un impacto significativo. También hemos podido observar como varía el rendimiento al cambiar de un equipo de trabajo a otro.

Como conclusión, hemos podido vernos inmersos en el desarrollo de una fase de testing de una manera compleja, pudiendo observar su importancia para que el producto final sea de mayor calidad para el cliente.

***Bibliografía***

Intencionalmente en blanco.