***DISEÑO Y PRUEBAS II***

***TESTING REPORT***

**C2.018**

**David Gavira Serrano**

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

**Luis Garcia Parras**

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

**Felipe Solis Agudo**

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

**Pablo Espinosa Naranjo**

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

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



|  |  |  |
| --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** |
| 01/07/2024 | 1.0 | Versión inicial |
| 03/07/2024 | 1.1 | Testing funcional |
| 04/07/2024 | 1.2 | Analisis 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

***Contenido***

1. Resumen ejecutivo 3
2. Introducción 4
3. Contenido
   1. Pruebas funcionales 5 - 13
   2. Análisis de desempeño 14 - 17
4. Conclusiones 18
5. Bibliografía 19

***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 al requisito 11 de los “groupal requitement” del grupo C2.018, que giran en torno a las entidad “Banner”.

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 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 una búsqueda de mejorar el rendimiento.

***Contenido***

***Pruebas funcionales***

Para todos los distintos servicios que ofrece la aplicación en torno a la entidad “Banner”.

Estas pruebas se han desarrollado en la rama “Banner-Test”, tras ser corregidos los cambios solicitados para julio. En esta rama se encuentran todas las correcciones realizadas.

Destacamos esto por si se dieran errores “FAILED TO”, y los test no ofrecieran una cobertura real en la rama entregada al realizar el Tester Replayer. Esto nos paso a varios miembros del grupo en el anterior entregable, y entendemos que fue debido a la actualización del sample-data antes de hacer el merge, afectando al resultado directo de los test. Debido a esto, ante la posibilidad de que esto ocurra, queremos dejar en claro que en la rama “Banner-Test” se puede comprobar la cobertura real, si se diesen los fallos mencionados.

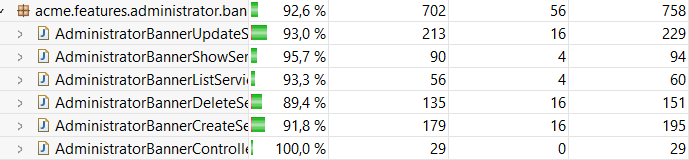
Además, solo se verá afectada la cobertura de los test, no su funcionamiento ni ejecución en el analyser.

Por último, también debe aclararse que los “FAILED TO” relacionados con banner se deben a los errores controlados de “Payload” de los banners mencionados. (No de la propia entidad, sino del randomicer en el footer)

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.

***Banner***

***List***

***Tests.safe***

* Listado de todos los banners, loggeado como administrador

***Tests.hack***

* Intentar acceder al listado de banners sin estar loggeado
* Intentar acceder al listado de banners loggeado, pero no como administrador

***Código ejecutado***

******

******

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

***Show***

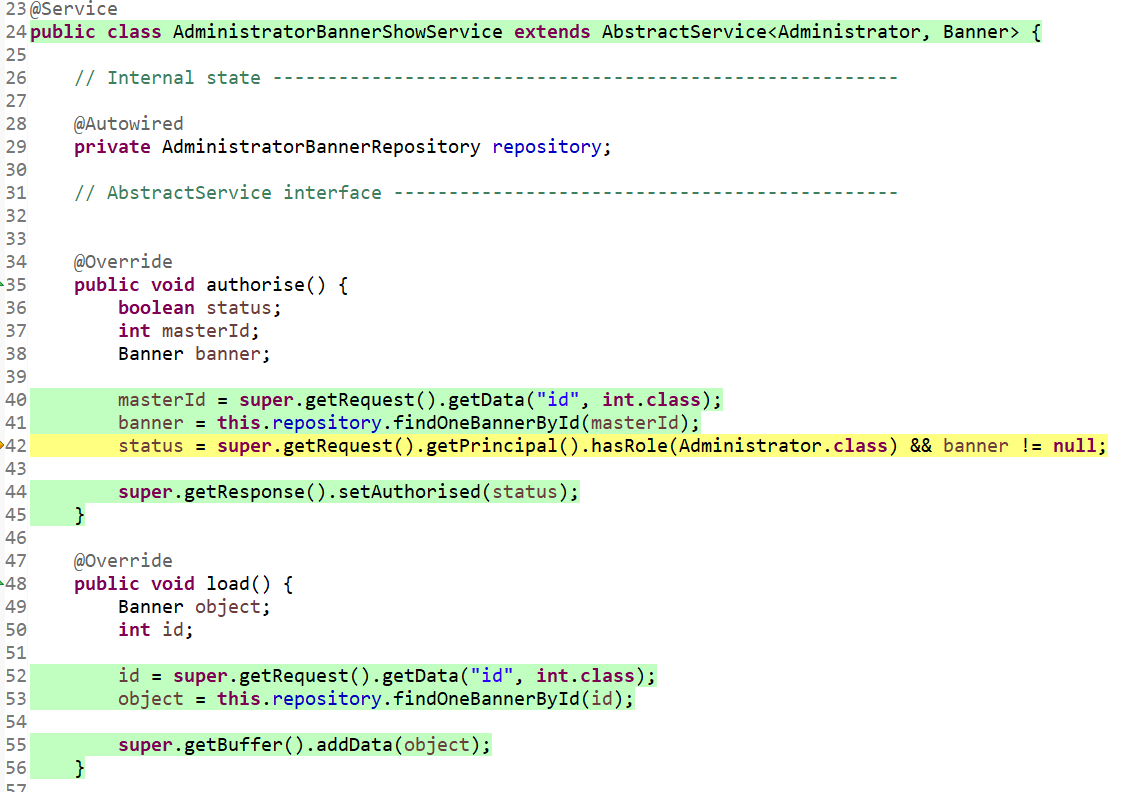
***Tests.safe***

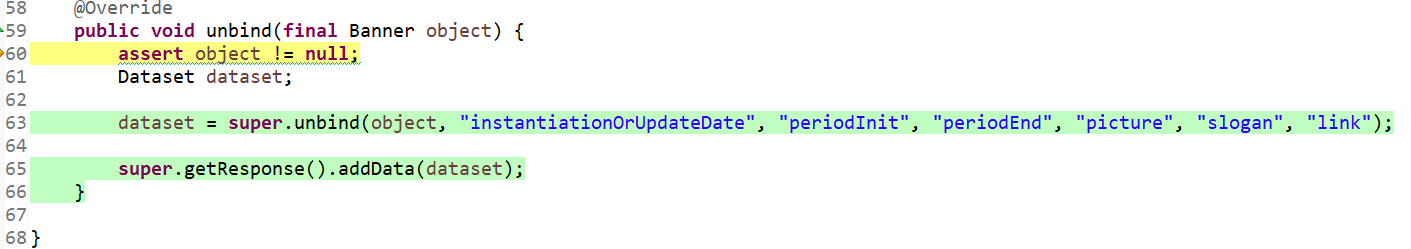
* Enseña los datos de un Banner

***Tests.hack***

* Intentar acceder a los datos de un banner sin estar loggeado
* Intentar acceder a los datos de un banner loggeado, pero no como administrador
* Intentar acceder a los datos de un banner loggeado como administrador, pero a uno 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***

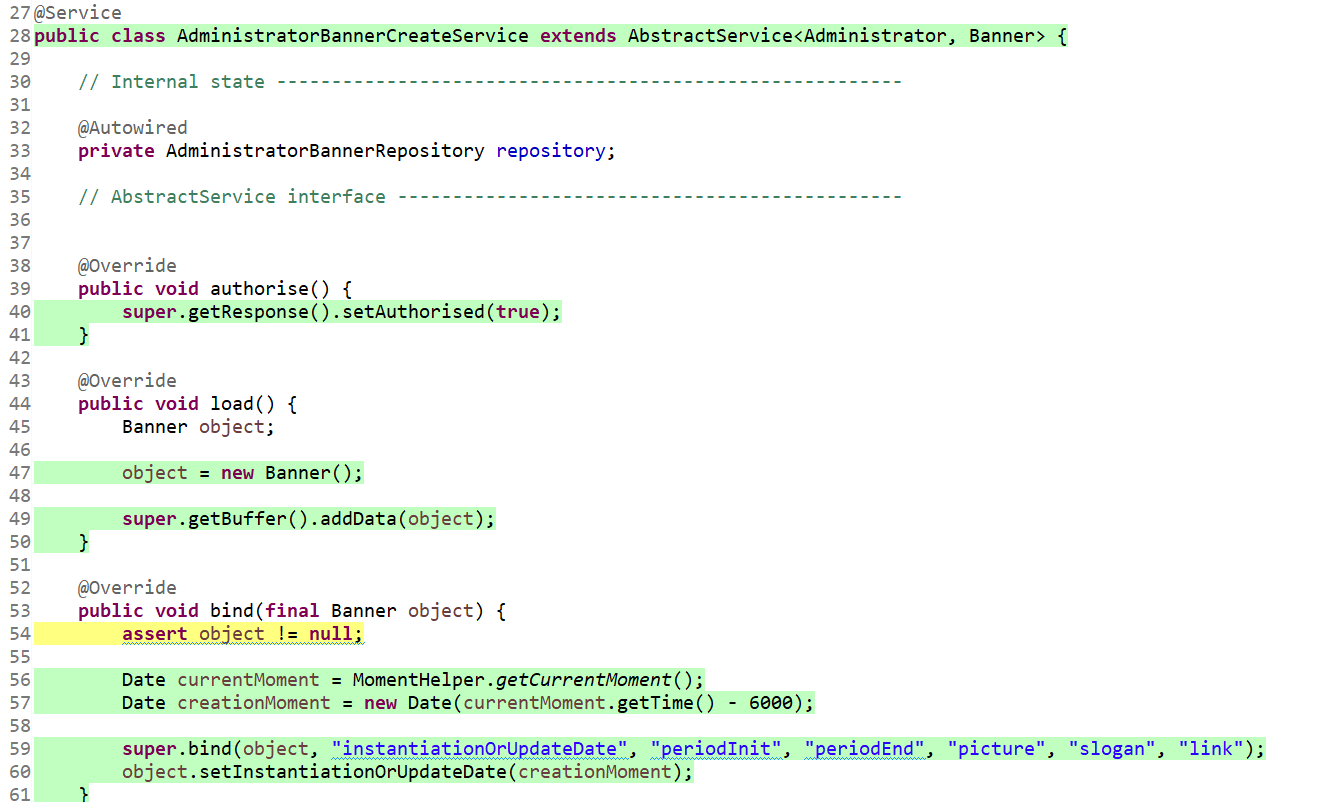
***Tests.safe***

* Crear un Banner con los datos vacíos
* Crear un Banner con variedad de datos erróneos en sus atributos
* Crear un Banner con datos límite válidos para sus atributos
* Crear un Banner con variedad de datos válidos en sus atributos

***Tests.hack***

* Intentar crear un Banner sin estar loggeado
* Intentar crear un Banner loggeado, pero no como administrador
* Intentar crear un Banner loggeado como administrador, pero probando a editar valores que serían ineditables por pantalla

***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: Se ejecutan los tres errores por separado, cumpliéndose las distintas restricciones de formato fecha. Sale como recorrida parcialmente ya que falta alguna combinación de las restricciones que, en el caso remoto de que se diesen a la vez, sería contemplada y solventada. Todos sus parámetros son necesarios, por lo que no pueden eliminarse para refactorizar.

***Update***

***Tests.safe***

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

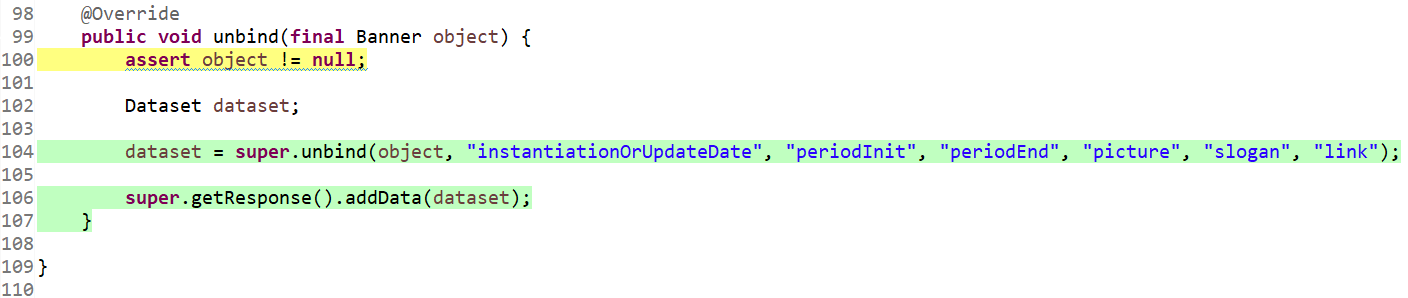
***Tests.hack***

* Intentar actualizar un Banner sin estar loggeado
* Intentar actualizar un Banner loggeado, pero no como administrador
* Intentar actualizar un Banner loggeado como administrador, pero probando a editar valores que serían ineditables por pantalla
* Intentar actualizar un Banner loggeado como administrador, pero un banner 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
* Validate: Se ejecutan los tres errores por separado, cumpliéndose las distintas restricciones de los atributos de tipo “date”, que están relacionados entre sí. Sale como “recorrida parcialmente” debido a que falta alguna combinación de las restricciones que, en el caso remoto de que se diesen a la vez, sería contemplada y solventada. Todos sus parámetros son necesarios, por lo que no pueden eliminarse para refactorizar.

***Delete***

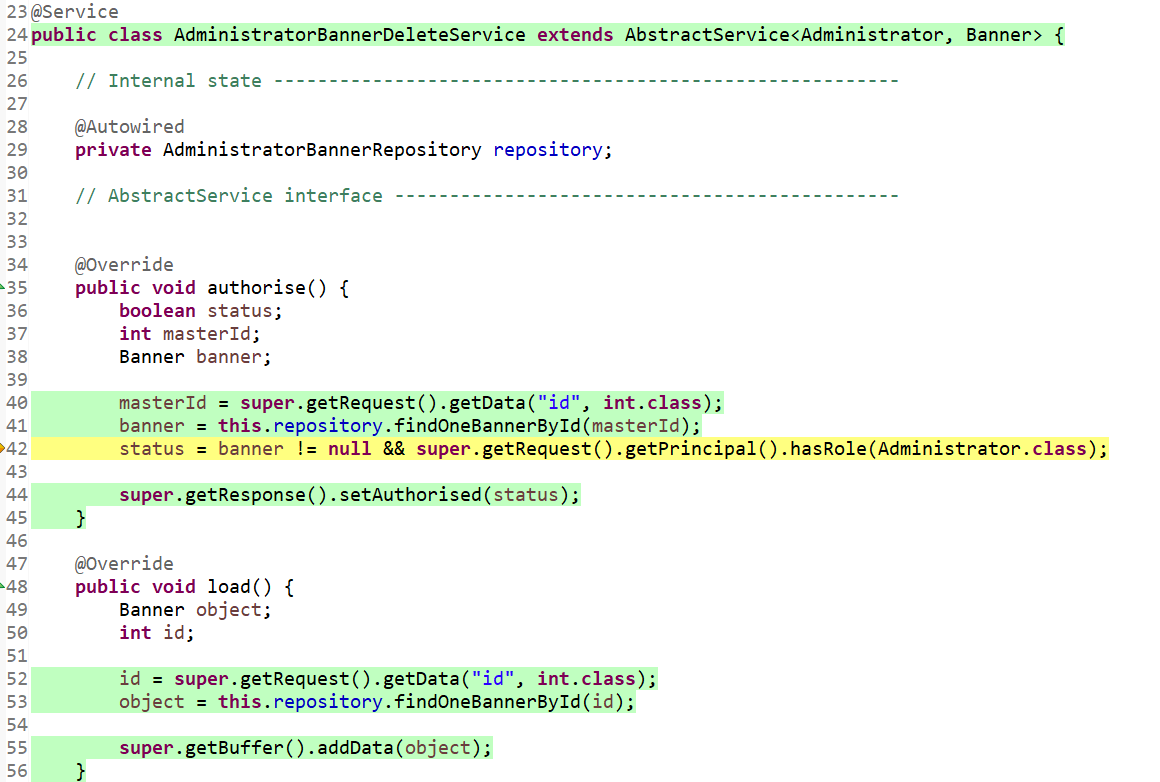
***Tests.safe***

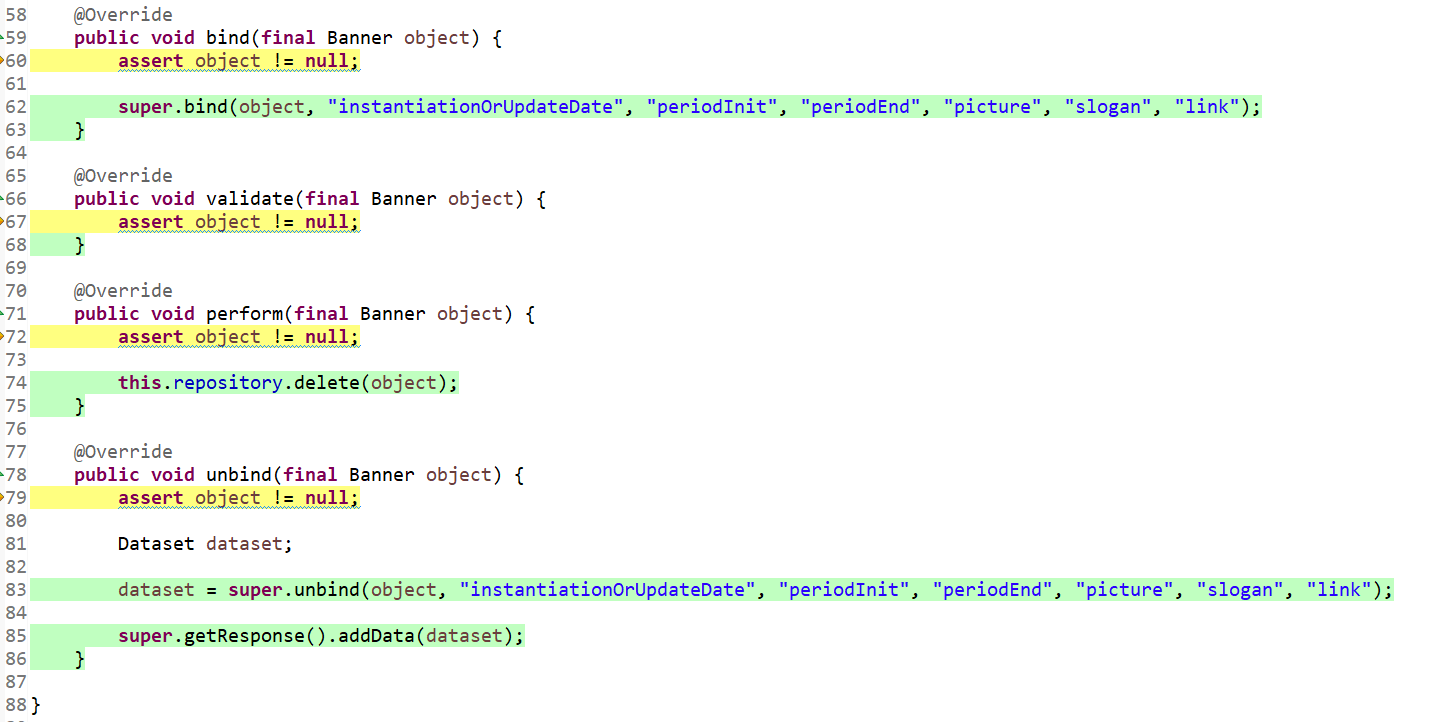
* Eliminar un Banner

***Tests.hack***

* Intentar eliminar un Banner sin estar loggeado
* Intentar eliminar un Banner loggeado, pero no como administrador
* Intentar eliminar un Banner loggeado como administrador, pero un banner inexistente

***Código ejecutado***

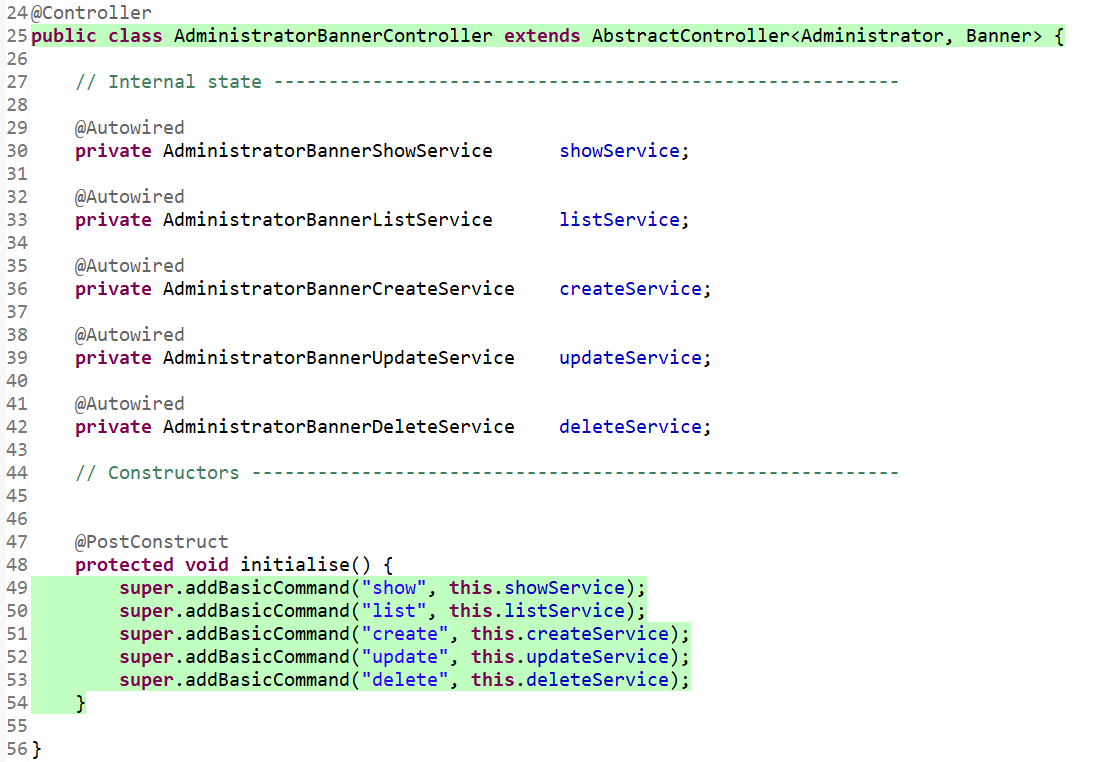






* Assert object != null: Linea obligatoria que sirve para verificar que el objeto no sea nulo.
* Condiciones de authorise: Se han ejecutado todas las alternativas por separado, faltando ciertas combinaciones concretas de valores sin cumplir combinadas. No es refactorizable, ya que todos los requisitos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

***Extra: Controlador***





Se observa cómo se ejecuta todo el código correctamente

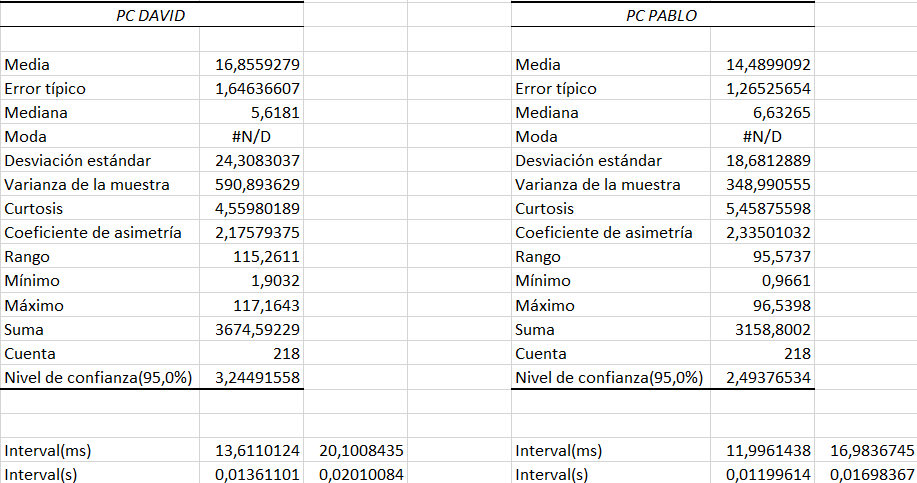
***Analisis de desempeño***

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 sus distintos rendimientos. 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.

En primer lugar es necesario introducir sobre que trata este análisis de desempeño realizado. Para desarrollar este análisis, hemos comparado el rendimiento ofrecido por dos ordenadores de dos miembros del equipo a la hora de ejecutar la traza referente a los test de “Banner”, observando así si hay una diferencia de rendimiento significativa estadísticamente.

**Comparación de dos ordenadores distintos optimizados**:

En este apartado se han comparado las prestaciones del ordenador del student-1, davgavser (PC1), y del student-5, pabespnar (PC2).

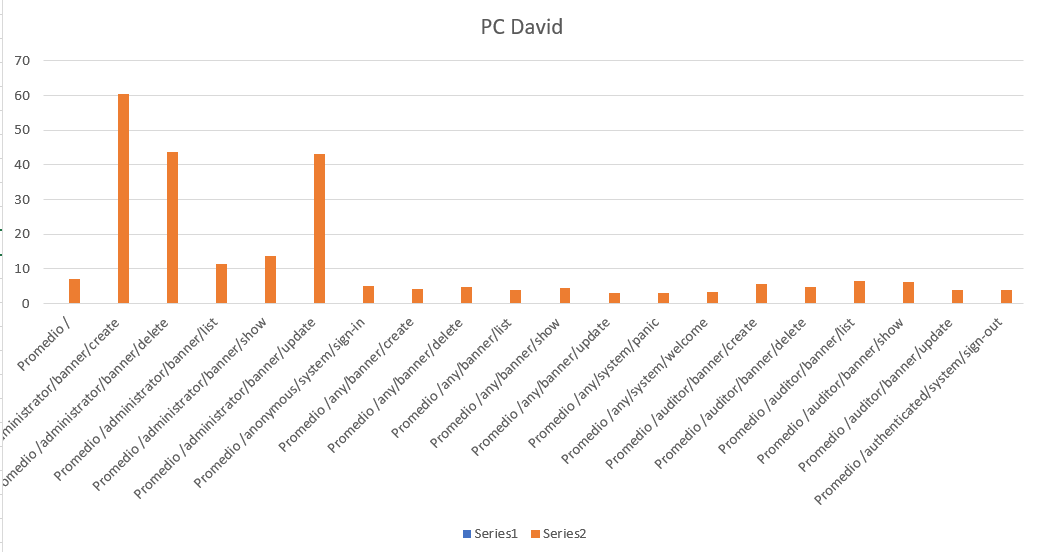


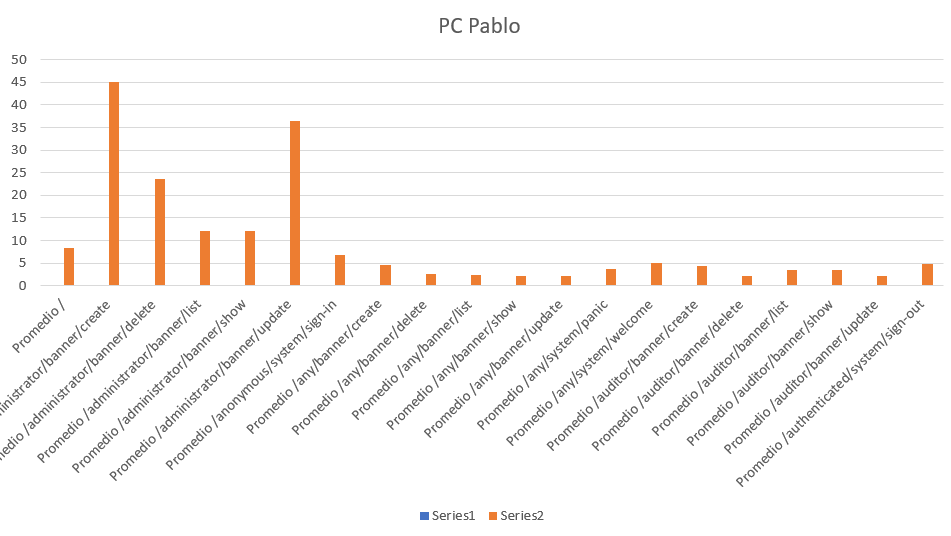
Estos son los resultados obtenidos usando la herramienta de “Estadística Descriptiva”

Se observa como no hay una gran diferencia entre las medias de ambos equipos (PC1 – 16.85, PC2 – 14.48), siendo levemente inferior el del PC de pabespnar, por lo que parece ofrecer un mayor rendimiento.

Si seguimos observando valores obtenidos, como la mediana (PC1 – 5.61, PC2 – 6.63), desviación estándar (PC1 – 24.3, PC2 – 18.68), varianza (PC1 – 590.89, PC2 – 348.99), curtosis (PC1 – 4.55, PC2 – 5.45), o sus intervalos, podemos ver como no existe una gran diferencia entre ninguno de estos. Aún así, es pronto para garantizar que sean o no significativos los cambios de rendimiento.

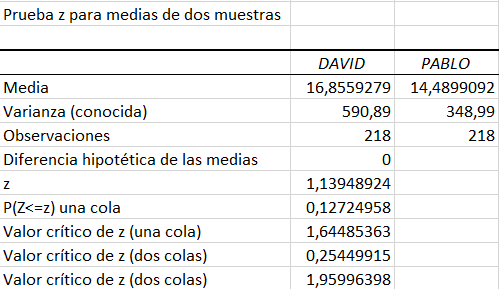
Pasamos a ver los histogramas:





Si entramos en detalle, podemos ver como en general, el PC de Pablo tarda menos tiempo promedio en realizar las distinas consultas, por lo que podemos suponer que este posee mayor rendimiento, aunque minimamente.

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 0,25 aproximadamente. Se debe comparar 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 entonces como nuestro valor crítico está fuera del intervalo [0, 0.05],por lo que llegamos a la siguiente conclusión:  
  
**No existe una diferencia estadísticamente significativa** entre los dos conjuntos, siendo esto bastante parejos, aunque con mejor rendimiento levemente del PC de pabespnar.

***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