Escuela Técnica Superior de Ingeniería Informática

Testing Report D04

Group C2.020 | Diseño y Pruebas II | 08/07/2024

Fecha	Versión	Autor
07/07/2024	1.0	Diego González Quintanilla

Miembros:

- Diego González Quintanilla (diegonqui@alum.us.es)

Repositorio de Github: https://github.com/DP2-C1-020/Acme-SF-D04

Contenido

Resumen ejecutivo..... 3
Introducción..... 4
Contenido: Pruebas Funcionales..... 5
Pruebas de Rendimiento..... 17
Conclusiones..... 21
Bibliografía..... 22

Resumen ejecutivo

El documento pretende dar a conocer y presentar un análisis de los resultados de realizar las pruebas necesarias para garantizar la ausencia de bugs en las implementaciones desarrolladas por el Student 4, Diego González Quintanilla, asociadas a los requisitos #6 y #7 del documento de requisitos proporcionado por el cliente. El informe se dividirá en dos capítulos: **Pruebas Funcionales** y **Pruebas de Rendimiento**.

Introducción

El objetivo del presente informe es garantizar al cliente la calidad del software realizado, mediante la muestra de las pruebas ejecutadas y su posterior análisis. Para la organización del informe, se han seguido las pautas del documento "Annexes: On your reports", que explica cómo se deberá realizar la división de contenidos del informe. Por tanto, se procederá a dividir el documento en dos secciones:

- Un primer capítulo llamado **Pruebas Funcionales**, en el que se ahondará en todos los casos de prueba positivos, negativos y de "*hacking"* llevados a cabo por el equipo de Testing para probar todas las características y funcionalidades relacionadas con los requisitos #6 y #7 propuestos por el cliente, haciendo hincapié en la efectividad de los mismos a la hora de detectar errores cuya solución condujo a un aumento de calidad del producto.

- Un segundo capítulo denominado **Pruebas de Rendimiento,** en el que se proporcionarán gráficos de los tiempos de ejecución de las instrucciones ejecutadas por las pruebas funcionales y el intervalo de confianza de los mismos en dos equipos diferentes. Además, se generará un contraste de hipótesis sobre qué equipo tiene mayor capacidad a la hora de ejecutar los tests.

Además de estos dos apartados principales, se adjunta una conclusión del equipo de Testing tras haber completado el análisis de las pruebas, así como una bibliografía con aquellos enlaces y documentos consultados que han servido de ayuda para la realización de este informe de pruebas.

Contenido: Pruebas Funcionales

En esta sección, se presentarán los casos de prueba realizados por el equipo de Testing y la efectividad de los mismos a la hora de resolver errores. Para cada característica, se han realizado casos de prueba **positivos y negativos**, almacenados en archivos *".safe"*, y casos de prueba de **hacking**, en archivos *".hack"*. Los casos positivos y negativos exploran acciones legales que pueden ser realizadas por el usuario a la hora de usar la aplicación, dando resultados positivos en caso de no encontrar problemas de validación, y resultados negativos en caso de hacer saltar un error cuando se introduce información incorrecta. Para facilitar la comprensión del documento, se usarán los siguientes signos para diferenciar los casos de prueba:

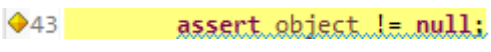
- +: Casos positivos.
- +-: Casos positivos y negativos.
- !: Casos de hacking.

A continuación, se procede a listar los casos de prueba agrupados por características implementadas en los requisitos #6 y #7. En adición, se indicará tanto la efectividad de cada prueba y el "*coverage*" de la función a desgranar.

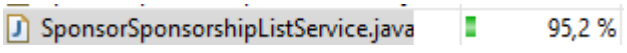
Comenzamos con las funcionalidades relacionadas con la **entidad Sponsorship** (requisito #6):

List Sponsorship:

- **Descripción de pruebas:**
  - list.safe (+): Se realiza una petición GET a la página principal, se inicia sesión con el Sponsor-01 y se accede al listado de Sponsorships del Sponsor-01 satisfactoriamente. Se repite el mismo proceso con el Sponsor-02.
  - list-wrong-role.hack (!): Se realiza una petición GET a la página principal y se trata de acceder a la url "/sponsor/sponsorship/list" insatisfactoriamente.
- **Efectividad**:
  - list.safe (+): Baja, porque no se encontraron bugs, debido a la sencillez de la funcionalidad, ya que simplemente muestra datos de la BBDD al Sponsor correspondiente.
  - list-wrong-role.hack (!): Baja, porque no se encontraron bugs, debido a la seguridad del código, que no permite el acceso a la lista de Sponsorships a ningún otro rol que no sea Sponsor.
- **Coverage:**
  - La cobertura de la funcionalidad List Sponsorship es de un 95.2 %. La única línea que no se cubre del todo es:

    Esto es porque el objeto nunca será nulo, ya que entonces se trataría de un fallo en la recuperación de datos de la base de datos. Por lo tanto, incurriríamos en un error en la lógica del software. En conclusión, no puede ser probada.

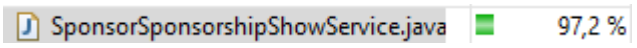
Show Sponsorship:

- **Descripción de pruebas:**
  - show.safe (+): Se realiza una petición GET a la página principal, se inicia sesión con el Sponsor-01, se accede al listado de Sponsorships del Sponsor-01 y se ingresa en los detalles del Sponsorship-02 satisfactoriamente. Se repite el mismo proceso con el Sponsor-02 y el Sponsorship-01.
  - show-wrong-role.hack (!): Se realiza una petición GET a la página principal y se trata de acceder a la url "/sponsor/sponsorship/show?id=149" con el rol Any insatisfactoriamente.
  - show-wrong-user.hack (!): Se realiza una petición GET a la página principal, se inicia sesión con el Sponsor-02 y se trata de acceder a la url "/sponsor/sponsorship/show?id=149" insatisfactoriamente.
- **Efectividad**:
  - show.safe (+): Baja, porque no se encontraron bugs, debido a la sencillez de la funcionalidad, ya que simplemente muestra datos de la BBDD al Sponsor correspondiente.
  - show-wrong-role.hack (!): Baja, porque no se encontraron bugs, debido a la seguridad del código, que no permite el acceso a los detalles de un Sponsorship a ningún otro rol que no sea Sponsor.
  - show-wrong-user.hack (!): Baja, porque no se encontraron bugs, debido a la seguridad del código, que no permite el acceso a los detalles de un Sponsorship ajeno a ningún otro Sponsor que no sea el correspondiente.
- **Coverage:**
  - La cobertura de la funcionalidad Show Sponsorship es de un 97.2 %. Las únicas líneas que no se cubren del todo son:

```
assert sponsorship != null;
```

```
status = sponsorship != null && super.getRequest().getPrincipal().hasRole(sponsorship.getSponsor());
```

Se remite a la misma explicación dada en List Sponsorship sobre la no nulidad de un objeto.



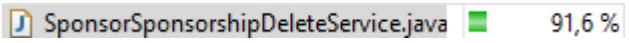
Delete Sponsorship:

- **Descripción de pruebas:**
  - delete.safe (+): Se realiza una petición GET a la página principal, se inicia sesión con el Sponsor-01, se accede al listado de Sponsorships del Sponsor-01, se ingresa en los detalles del Sponsorship-02 y se borra satisfactoriamente.
  - delete-wrong-role.hack (!): Se realiza una petición GET a la página principal y posteriormente un POST a "/sponsor/sponsorship/delete" con los datos del Sponsorship-02 en la request. El resultado es insatisfactorio.
  - delete-wrong-user.hack (!): Se realiza una petición GET a la página principal, se inicia sesión con el Sponsor-02 y posteriormente un POST a "/sponsor/sponsorship/delete" con los datos del Sponsorship-02 (perteneciente a Sponsor-01) en la request. El resultado es insatisfactorio.
  - delete-published.hack (!): Se realiza una petición GET a la página principal, se inicia sesión con el Sponsor-04 y posteriormente un POST a "/sponsor/sponsorship/delete" con los datos del Sponsorship-04 (que está publicado) en la request. El resultado es insatisfactorio.
- **Efectividad:**
  - delete.safe (+): Baja, porque no se encontraron bugs, debido a la sencillez de la funcionalidad, ya que simplemente borra el Sponsorship correspondiente.
  - delete-wrong-role.hack (!): Baja, porque no se encontraron bugs, debido a la seguridad del código, que no permite el acceso al borrado de un Sponsorship a ningún otro rol que no sea Sponsor.
  - delete-wrong-user.hack (!): Baja, porque no se encontraron bugs, debido a la seguridad del código, que no permite el acceso al borrado de un Sponsorship ajeno a ningún otro Sponsor que no sea el correspondiente.
  - delete-published.hack (!): Baja, porque no se encontraron bugs, debido a la seguridad del código, que no permite el acceso al borrado de un Sponsorship publicado.
- **Coverage:**
  - La cobertura de la funcionalidad Delete Sponsorship es de un 91.6 %. Las únicas líneas que no se cubren del todo son:

4x `assert sponsorship != null;`

```
status = sponsorship != null && sponsorship.isDraftMode() && super.getRequest().getPrincipal().hasRole(sponsorship.getSponsor());
```

Se remite a la misma explicación dada en List Sponsorship sobre la no nulidad de un objeto.



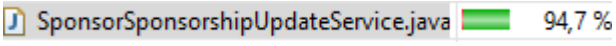
Update Sponsorship:

- **Descripción de pruebas:**
  - update.safe (+-): Se realiza una petición GET a la página principal, se inicia sesión con el Sponsor-01, se accede al listado de Sponsorships del Sponsor-01, se ingresa en los detalles del Sponsorship-02 y se prueban todas las combinaciones posibles de datos haciendo saltar validaciones. El resultado es satisfactorio.
  - update-wrong-role.hack (!): Se realiza una petición GET a la página principal y posteriormente un POST a "/sponsor/sponsorship/update" con los datos del Sponsorship-04 en la request. El resultado es insatisfactorio.
  - update-wrong-user.hack (!): Se realiza una petición GET a la página principal, se inicia sesión con el Sponsor-02 y posteriormente un POST a "/sponsor/sponsorship/update" con los datos del Sponsorship-02 (perteneciente a Sponsor-01) en la request. El resultado es insatisfactorio.
  - update-published.hack (!): Se realiza una petición GET a la página principal, se inicia sesión con el Sponsor-04 y posteriormente un POST a "/sponsor/sponsorship/update" con los datos del Sponsorship-04 (que está publicado) en la request. El resultado es insatisfactorio.
- **Efectividad:**
  - update.safe (+-): Baja, porque no se encontraron bugs, debido a la sencillez de la funcionalidad, ya que simplemente actualiza el Sponsorship correspondiente.
  - update-wrong-role.hack (!): Baja, porque no se encontraron bugs, debido a la seguridad del código, que no permite el acceso a la actualización de un Sponsorship a ningún otro rol que no sea Sponsor.
  - update-wrong-user.hack (!): Baja, porque no se encontraron bugs, debido a la seguridad del código, que no permite el acceso a la actualización de un Sponsorship ajeno a ningún otro Sponsor que no sea el correspondiente.
  - update-published.hack (!): Baja, porque no se encontraron bugs, debido a la seguridad del código, que no permite el acceso a la actualización de un Sponsorship publicado.
- **Coverage:**
  - La cobertura de la funcionalidad Update Sponsorship es de un 94.7 %. Las únicas líneas que no se cubren del todo son:

4x `assert sponsorship != null;`

```
status = sponsorship != null && sponsorship.isDraftMode() && super.getRequest().getPrincipal().hasRole(sponsorship.getSponsor());
```

Se remite a la misma explicación dada en List Sponsorship sobre la no nulidad de un objeto.



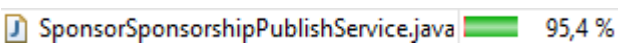
Publish Sponsorship:

- **Descripción de pruebas:**
  - publish.safe (+-): Se realiza una petición GET a la página principal, se inicia sesión con el Sponsor-04, se accede al listado de Sponsorships del Sponsor-01, se ingresa en los detalles del Sponsorship-02 y se prueban todas las combinaciones posibles (validaciones, Invoices no publicados, suma de Quantities no suficientes...) haciendo saltar validaciones. El resultado es satisfactorio.
  - publish-wrong-role.hack (!): Se realiza una petición GET a la página principal y posteriormente un POST a "/sponsor/sponsorship/publish" con los datos del Sponsorship-02 en la request. El resultado es insatisfactorio.
  - publish-wrong-user.hack (!): Se realiza una petición GET a la página principal, se inicia sesión con el Sponsor-02 y posteriormente un POST a "/sponsor/sponsorship/publish" con los datos del Sponsorship-02 (perteneciente a Sponsor-01) en la request. El resultado es insatisfactorio.
  - publish-published.hack (!): Se realiza una petición GET a la página principal, se inicia sesión con el Sponsor-04 y posteriormente un POST a "/sponsor/sponsorship/publish" con los datos del Sponsorship-04 (que está publicado) en la request. El resultado es insatisfactorio.
- **Efectividad:**
  - publish.safe (+-): Baja, porque no se encontraron bugs, debido a la sencillez de la funcionalidad, ya que simplemente publica el Sponsorship correspondiente.
  - publish-wrong-role.hack (!): Baja, porque no se encontraron bugs, debido a la seguridad del código, que no permite el acceso a la publicación de un Sponsorship a ningún otro rol que no sea Sponsor.
  - publish-wrong-user.hack (!): Baja, porque no se encontraron bugs, debido a la seguridad del código, que no permite el acceso a la publicación de un Sponsorship ajeno a ningún otro Sponsor que no sea el correspondiente.
  - publish-published.hack (!): Baja, porque no se encontraron bugs, debido a la seguridad del código, que no permite el acceso a la publicación de un Sponsorship publicado.
- **Coverage:**
  - La cobertura de la funcionalidad Publish Sponsorship es de un 95.4 %. Las únicas líneas que no se cubren del todo son:

4x `assert sponsorship != null;`

```
status = sponsorship != null && sponsorship.isDraftMode() && super.getRequest().getPrincipal().hasRole(sponsorship.getSponsor());
```

Se remite a la misma explicación dada en List Sponsorship sobre la no nulidad de un objeto.

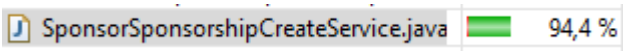


Create Sponsorship:

- **Descripción de pruebas:**
  - create.safe (+-): Se realiza una petición GET a la página principal, se inicia sesión con el Sponsor-01, se accede al listado de Sponsorships del Sponsor-01, se ingresa en el formulario de creación y se prueban todas las combinaciones posibles haciendo saltar validaciones. Resultado satisfactorio.
  - create-wrong-role.hack (!): Se realiza una petición GET a la página principal y posteriormente un POST a "/sponsor/sponsorship/create" con datos de un Sponsorship nuevo en la request. El resultado es insatisfactorio.
- **Efectividad:**
  - create.safe (+-): Alta, ya que esta prueba permitió descubrir que no se estaba realizando la validación correspondiente a las divisas permitidas en el sistema en relación al atributo Amount de tipo Money, por lo que se procedió a corregir esta validación en los servicios de Create, Update y Publish.
  - create-wrong-role.hack (!): Baja, porque no se encontraron bugs, debido a la seguridad del código, que no permite el acceso a la creación de un Sponsorship a ningún otro rol que no sea Sponsor.
- **Coverage:**
  - La cobertura de la funcionalidad Create Sponsorship es de un 94.4 %. Las únicas líneas que no se cubren del todo son:

    4x `assert sponsorship != null;`

    Se remite a la misma explicación dada en List Sponsorship sobre la no nulidad de un objeto.



A continuación, se procederá a explicar los tests de las funcionalidades relacionadas con la **entidad Invoice** (requisito #7):

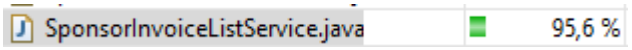
List Invoice:

- **Descripción de pruebas:**
  - list.safe (+): Se realiza una petición GET a la página principal, se inicia sesión con el Sponsor-01, se accede al listado de Sponsorships, se ingresa en los detalles del Sponsorship-02 y se finaliza accediendo a la lista de Invoices asociada satisfactoriamente. Se repite el mismo proceso con el Sponsor-02.
  - list-wrong-role.hack (!): Se realiza una petición GET a la página principal y se trata de acceder a la url "/sponsor/invoice/list" con el id del Sponsorship-01 en la request. El resultado es insatisfactorio.
  - list-wrong-user.hack (!): Se realiza una petición GET a la página principal, se inicia sesión con el Sponsor-02 y se trata de acceder a la url "/sponsor/invoice/list" con el id del Sponsorship-01 en la request. El resultado es insatisfactorio.
- **Efectividad**:
  - list.safe (+): Baja, porque no se encontraron bugs, debido a la sencillez de la funcionalidad, ya que simplemente muestra datos de la BBDD al Sponsor correspondiente.
  - list-wrong-role.hack (!): Baja, porque no se encontraron bugs, debido a la seguridad del código, que no permite el acceso a la lista de Invoices a ningún otro rol que no sea Sponsor.
  - list-wrong-user.hack (!): Baja, porque no se encontraron bugs, debido a la seguridad del código, que no permite el acceso a la lista de Invoices a ningún otro Sponsor que no sea el Sponsor correspondiente.
- **Coverage:**
  - La cobertura de la funcionalidad List Invoice es de un 95.6 %. Las únicas líneas que no se cubren del todo son:

```
assert object != null;
```

```
status = sponsorship != null && super.getRequest().getPrincipal().hasRole(sponsorship.getSponsor());
```

Se remite a la misma explicación dada en List Sponsorship sobre la no nulidad de un objeto.



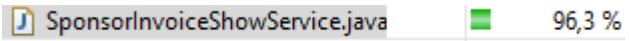
Show Invoice:

- **Descripción de pruebas:**
  - show.safe (+): Se realiza una petición GET a la página principal, se inicia sesión con el Sponsor-01, se accede al listado de Sponsorships del Sponsor-01, se ingresa en los detalles del Sponsorship-02, se accede a la lista de Invoices asociada y se termina entrando a los detalles del Invoice-02 satisfactoriamente. Se repite el mismo proceso con el Sponsor-02, Sponsorship-01 e Invoice-03.
  - show-wrong-role.hack (!): Se realiza una petición GET a la página principal y se trata de acceder a la url "/sponsor/invoice/show?id=165" con el rol Any insatisfactoriamente.
  - show-wrong-user.hack (!): Se realiza una petición GET a la página principal, se inicia sesión con el Sponsor-02 y se trata de acceder a la url "/sponsor/sponsorship/show?id=165" insatisfactoriamente.
- **Efectividad**:
  - show.safe (+): Baja, porque no se encontraron bugs, debido a la sencillez de la funcionalidad, ya que simplemente muestra datos de la BBDD al Sponsor correspondiente.
  - show-wrong-role.hack (!): Baja, porque no se encontraron bugs, debido a la seguridad del código, que no permite el acceso a los detalles de un Invoice a ningún otro rol que no sea Sponsor.
  - show-wrong-user.hack (!): Baja, porque no se encontraron bugs, debido a la seguridad del código, que no permite el acceso a los detalles de un Invoice ajeno a ningún otro Sponsor que no sea el correspondiente.
- **Coverage:**
  - La cobertura de la funcionalidad Show Invoice es de un 96.3 %. Las únicas líneas que no se cubren del todo son:

```
assert invoice != null;
```

```
status = invoice != null && super.getRequest().getPrincipal().hasRole(invoice.getSponsorship().getSponsor());
```

Se remite a la misma explicación dada en List Sponsorship sobre la no nulidad de un objeto.

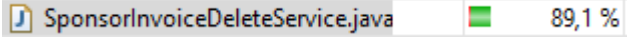


Delete Invoice:

- **Descripción de pruebas:**
  - delete.safe (+): Se realiza una petición GET a la página principal, se inicia sesión con el Sponsor-01, se accede al listado de Sponsorships del Sponsor-01, se ingresa en los detalles del Sponsorship-02, se accede a la lista de Invoices relacionados, se termina entrando en los detalles del Invoice-02 y se borra satisfactoriamente.
  - delete-wrong-role.hack (!): Se realiza una petición GET a la página principal y posteriormente un POST a "/sponsor/invoice/delete" con los datos del Invoice-02 en la request. El resultado es insatisfactorio.
  - delete-wrong-user.hack (!): Se realiza una petición GET a la página principal, se inicia sesión con el Sponsor-02 y posteriormente un POST a "/sponsor/invoice/delete" con los datos del Invoice-02 (perteneciente a Sponsor-01) en la request. El resultado es insatisfactorio.
  - delete-published.hack (!): Se realiza una petición GET a la página principal, se inicia sesión con el Sponsor-01 y posteriormente un POST a "/sponsor/invoice/delete" con los datos del Invoice-12 (que está publicado) en la request. El resultado es insatisfactorio.
- **Efectividad:**
  - delete.safe (+): Baja, porque no se encontraron bugs, debido a la sencillez de la funcionalidad, ya que simplemente borra el Invoice correspondiente.
  - delete-wrong-role.hack (!): Baja, porque no se encontraron bugs, debido a la seguridad del código, que no permite el acceso al borrado de un Invoice a ningún otro rol que no sea Sponsor.
  - delete-wrong-user.hack (!): Baja, porque no se encontraron bugs, debido a la seguridad del código, que no permite el acceso al borrado de un Invoice ajeno a ningún otro Sponsor que no sea el correspondiente.
  - delete-published.hack (!): Baja, porque no se encontraron bugs, debido a la seguridad del código, que no permite el acceso al borrado de un Invoice publicado.
- **Coverage:**
  - La cobertura de la funcionalidad Delete Invoice es de un 89.1 %. Las únicas líneas que no se cubren del todo son:

    4x `assert invoice != null;`

    Se remite a la misma explicación dada en List Sponsorship sobre la no nulidad de un objeto.



Update Invoice:

- **Descripción de pruebas:**
  - update.safe (+-): Se realiza una petición GET a la página principal, se inicia sesión con el Sponsor-01, se accede al listado de Sponsorships del Sponsor-01, se ingresa en los detalles del Sponsorship-02, se accede a su vez a la lista de Invoices asociados, se entra en los detalles del Invoice-02 y se prueban todas las combinaciones posibles de datos haciendo saltar validaciones. El resultado es satisfactorio.
  - update-wrong-role.hack (!): Se realiza una petición GET a la página principal y posteriormente un POST a "/sponsor/invoice/update" con los datos del Invoice-03 en la request. El resultado es insatisfactorio.
  - update-wrong-user.hack (!): Se realiza una petición GET a la página principal, se inicia sesión con el Sponsor-01 y posteriormente un POST a "/sponsor/invoice/update" con los datos del Invoice-03 (perteneciente a Sponsor-02) en la request. El resultado es insatisfactorio.
  - update-published.hack (!): Se realiza una petición GET a la página principal, se inicia sesión con el Sponsor-01 y posteriormente un POST a "/sponsor/invoice/update" con los datos del Invoice-12 (que está publicado) en la request. El resultado es insatisfactorio.
- **Efectividad:**
  - update.safe (+-): Baja, porque no se encontraron bugs, debido a la sencillez de la funcionalidad, ya que simplemente actualiza el Invoice correspondiente.
  - update-wrong-role.hack (!): Baja, porque no se encontraron bugs, debido a la seguridad del código, que no permite el acceso a la actualización de un Invoice a ningún otro rol que no sea Sponsor.
  - update-wrong-user.hack (!): Baja, porque no se encontraron bugs, debido a la seguridad del código, que no permite el acceso a la actualización de un Invoice ajeno a ningún otro Sponsor que no sea el correspondiente.
  - update-published.hack (!): Baja, porque no se encontraron bugs, debido a la seguridad del código, que no permite el acceso a la actualización de un Invoice publicado.
- **Coverage:**
  - La cobertura de la funcionalidad Update Invoice es de un 93.4 %. Las únicas líneas que no se cubren del todo son:

    4x `assert invoice != null;`

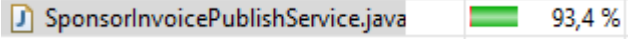
    Se remite a la misma explicación dada en List Sponsorship sobre la no nulidad de un objeto.

Publish Invoice:

- **Descripción de pruebas:**
  - publish.safe (+-): Se realiza una petición GET a la página principal, se inicia sesión con el Sponsor-01, se accede al listado de Sponsorships del Sponsor-01, se ingresa en la lista de Invoices asociados para finalmente entrar en los detalles del Invoice-02 y se actualiza probando todas las combinaciones posibles haciendo saltar validaciones. El resultado es satisfactorio.
  - publish-wrong-role.hack (!): Se realiza una petición GET a la página principal y posteriormente un POST a "/sponsor/invoice/publish" con los datos del Invoice-02 en la request. El resultado es insatisfactorio.
  - publish-wrong-user.hack (!): Se realiza una petición GET a la página principal, se inicia sesión con el Sponsor-02 y posteriormente un POST a "/sponsor/invoice/publish" con los datos del Invoice-02 (perteneciente a Sponsor-01) en la request. El resultado es insatisfactorio.
  - publish-published.hack (!): Se realiza una petición GET a la página principal, se inicia sesión con el Sponsor-01, se accede a la lista de Sponsorships, luego a los detalles del Sponsorship-13, a su lista asociada de Invoices para finalmente entrar en los detalles del Invoice-12. Posteriormente se realiza un POST a "/sponsor/invoice/publish" con los datos del Invoice-12 (que está publicado) en la request. El resultado es insatisfactorio.
- **Efectividad:**
  - publish.safe (+-): Baja, porque no se encontraron bugs, debido a la sencillez de la funcionalidad, ya que simplemente publica el Invoice correspondiente.
  - publish-wrong-role.hack (!): Baja, porque no se encontraron bugs, debido a la seguridad del código, que no permite el acceso a la publicación de un Invoice a ningún otro rol que no sea Sponsor.
  - publish-wrong-user.hack (!): Baja, porque no se encontraron bugs, debido a la seguridad del código, que no permite el acceso a la publicación de un Invoice ajeno a ningún otro Sponsor que no sea el correspondiente.
  - publish-published.hack (!): Baja, porque no se encontraron bugs, debido a la seguridad del código, que no permite el acceso a la publicación de un Invoice publicado.
- **Coverage:**
  - La cobertura de la funcionalidad Publish Invoice es de un 93.4 %. Las únicas líneas que no se cubren del todo son:

    4x `assert invoice != null;`

    Se remite a la misma explicación dada en List Sponsorship sobre la no nulidad de un objeto.

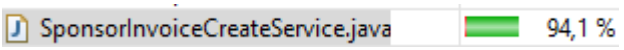


Create Invoice:

- **Descripción de pruebas:**
  - create.safe (+-): Se realiza una petición GET a la página principal, se inicia sesión con el Sponsor-01, se accede al listado de Sponsorships del Sponsor-01, seguidamente se entra en los detalles del Sponsorship-02 para acabar accediendo a la lista de Invoices asociados. Aquí, se ingresa en el formulario de creación y se prueban todas las combinaciones posibles haciendo saltar validaciones. Resultado satisfactorio.
  - create-wrong-role.hack (!): Se realiza una petición GET a la página principal y posteriormente un POST a "/sponsor/invoice/create" con datos de un Invoice nuevo en la request. El resultado es insatisfactorio.
  - create-wrong-user.hack (!): Se realiza una petición GET a la página principal, se inicia sesión con el Sponsor-02 y posteriormente un POST a "/sponsor/sponsorship/create" con datos de un Invoice nuevo y el id del Sponsorship-13 (perteneciente al Sponsor-01) en la request. El resultado es insatisfactorio.
- **Efectividad:**
  - create.safe (+): Baja, porque no se encontraron bugs, debido a la sencillez de la funcionalidad, ya que simplemente crea un Invoice nuevo.
  - create-wrong-role.hack (!): Baja, porque no se encontraron bugs, debido a la seguridad del código, que no permite el acceso a la creación de un Invoice a ningún otro rol que no sea Sponsor.
  - create-wrong-user.hack (!): Baja, porque no se encontraron bugs, debido a la seguridad del código, que no permite el acceso a la creación de un Invoice nuevo asociado a un Sponsorship ajeno al del Sponsor correspondiente.
- **Coverage:**
  - La cobertura de la funcionalidad Create Invoice es de un 94.1 %. Las únicas líneas que no se cubren del todo son:

    4x `assert sponsorship != null;`

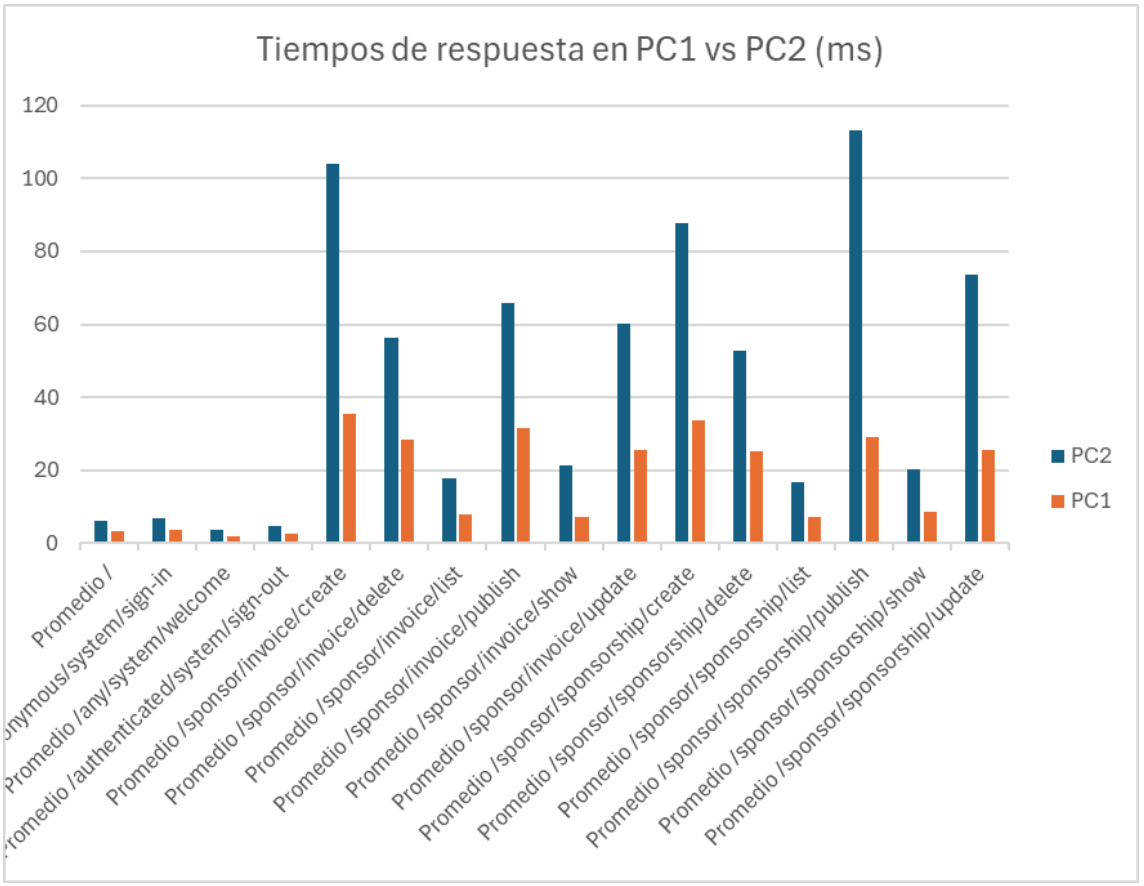
    Se remite a la misma explicación dada en List Sponsorship sobre la no nulidad de un objeto.

Pruebas de Rendimiento

En este capítulo del informe se presentarán los resultados de haber ejecutado los tests en dos equipos diferentes (PC1 y PC2), así como la comparativa del PC1 tras haber mejorado el rendimiento del código con el uso de índices.

Por tanto, se comienza con los datos de rendimiento del PC1 vs. PC2:



Esta gráfica muestra que las prestaciones del PC1 son mucho mayores, pues realiza las requests en un tiempo considerablemente menor a las realizadas por el PC2 al lanzar el mismo test suite. Además, como puede comprobarse en la siguiente página, el intervalo de confianza del 95% es más bajo en el PC1 que en el PC2. Por tanto, en este apartado se concluye que el PC1 es más potente a la hora de ejecutar los tests que el PC2.

Estadísticas de PC1					
			Interval (ms)	10,1495266	14,1287707
Media	12,1391486		Interval (s)	0,01014953	0,01412877
Error típico	1,01230753				
Mediana	5,2945				
Moda	#N/D				
Desviación estándar	21,1376017				
Varianza de la muestra	446,798206				
Curtosis	53,1363353				
Coeficiente de asimetría	6,13867436				
Rango	239,6382				
Mínimo	1,1787				
Máximo	240,8169				
Suma	5292,6688				
Cuenta	436				
Nivel de confianza(95,0%)	1,98962204				

Estadísticas del PC2					
Media	32,0406764		Interval (ms)	32,04067638	36,0055303
Error típico	2,01729341		Interval (s)	0,032040676	0,03600553
Mediana	11,26805				
Moda	#N/D				
Desviación estándar	42,122323				
Varianza de la muestra	1774,2901				
Curtosis	2,48197554				
Coeficiente de asimetría	1,70492593				
Rango	236,6384				
Mínimo	2,4972				
Máximo	239,1356				
Suma	13969,7349				
Cuenta	436				
Nivel de confianza(95,0%)	3,96485389				

Por otro lado, se ha realizado una mejora en los índices de las tablas de las entidades Invoice y Sponsorship, con el fin de mejorar los tiempos de ejecución de los tests. A continuación, se muestran las estadísticas antes y después de la creación de los índices:

Estadísticas de PC1 antes de índices					
			Interval (ms)	10,1495266	14,1287707
Media	12,1391486		Interval (s)	0,01014953	0,01412877
Error típico	1,01230753				
Mediana	5,2945				
Moda	#N/D				
Desviación estándar	21,1376017				
Varianza de la muestra	446,798206				
Curtosis	53,1363353				
Coeficiente de asimetría	6,13867436				
Rango	239,6382				
Mínimo	1,1787				
Máximo	240,8169				
Suma	5292,6688				
Cuenta	436				
Nivel de confianza(95,0%)	1,98962204				

Estadísticas de PC1 después de índices					
Media	13,1174216		Interval (ms)	13,11742156	14,6792781
Error típico	0,79466305		Interval (s)	0,013117422	0,01467928
Mediana	6,97215				
Moda	2,8625				
Desviación estándar	16,5930517				
Varianza de la muestra	275,329363				
Curtosis	15,2957519				
Coeficiente de asimetría	3,17101781				
Rango	144,9577				
Mínimo	1,6153				
Máximo	146,573				
Suma	5719,1958				
Cuenta	436				
Nivel de confianza(95,0%)	1,56185653				

Tras el cálculo de estos datos, se procede a hacer el z-test para medias de las dos muestras anteriores. Los resultados siguen en la siguiente página.

Prueba z para medias de dos muestras		
	before	*after*
Media	12,1391486	13,11742156
Varianza (conocida)	446,798206	275,3293632
Observaciones	436	436
Diferencia hipotética de las medias	0	
z	-0,76014479	
P(Z<=z) una cola	0,22358402	
Valor crítico de z (una cola)	1,64485363	
Valor crítico de z (dos colas)	0,44716804	
Valor crítico de z (dos colas)	1,95996398	

Contraste de hipótesis

La conclusión final que se puede sacar tras realizar el z-test a ambas muestras es que el p-value (valor crítico de z (dos colas)) se encuentra en el intervalo [alpha, 1.00], siendo alpha = 1 - nivel de confianza(95%) = 0.05. Por tanto, si p-value se encuentra dentro del intervalo [0.05, 1], quiere decir que los cambios llevados a cabo implementando los índices no se han traducido en ninguna mejora significante. Los tiempos de ejecución pueden ser diferentes pero globalmente son los mismos.

Conclusiones

La realización de pruebas funcionales es absolutamente crucial para la detección de errores que han podido escaparse a la vista del desarrollador, pues no es el encargado final de probar todas las funcionalidades de la aplicación y garantizar los requisitos del cliente. Es por eso, que se debe dar a conocer el proceso de cómo estos errores son detectados, para mejorar la confianza del cliente en el producto desarrollado.

Las pruebas de rendimiento son una parte importante a la hora de llevar a cabo el desarrollo de un producto y su posterior puesta en producción en dispositivos ajenos al de los desarrolladores. Es por ello que probar el sistema y la ejecución de sus pruebas en dispositivos ajenos a los desarrolladores es de suma importancia, pues puede dar lugar a oportunidades de mejora, como la refactorización del código o el arreglo de errores de compatibilidad con otros sistemas que lo ejecuten.

Para concluir el informe de pruebas se quiere destacar la importancia del testing a la hora de garantizar la seguridad y calidad de un producto software, por lo que se quiere transmitir al cliente un mensaje de confianza en el trabajo realizado por el equipo de desarrolladores.

Bibliografía

Documentos *"Annexes: On your reports"* , *"S02 - Performance testing"* y *"S01 - Formal testing"* en Enseñanza Virtual:

- https://ev.us.es/ultra/course