Universidad de Sevilla

Escuela Técnica Superior de Ingeniería Informática

**Documentación de la entrega D04**

**Testing**



Grado en Ingeniería Informática – Ingeniería del Software  
 Diseño y Pruebas 2

Curso 2023 – 2024

|  |  |
| --- | --- |
| **Fecha** | **Versión** |
| 27/05/2024 | v1r1 |

|  |  |  |
| --- | --- | --- |
| **Grupo de prácticas: G6-64** | | |
| **Autores por orden alfabético** | **Rol** | **Correo electrónico** |
| Aguayo Orozco, Sergio - 25604244T | Desarrollador | ahydul1@gmail.com |
| García Lama, Gonzalo - 47267072W | Desarrollador, Tester | gongarlam@alum.us.es |
| Huecas Calderón, Tomás - 17476993Y | Desarrollador | tomhuecal@alum.us.es |
| Fernández Pérez, Pablo - 54370557Y | Desarrollador,  Analista | pablofp.33@gmail.com |
| Youssafi Benichikh, Karim - 28823709V | Desarrollador, operador, manager | karyouben@alum.us.es |

## Repositorio: https://github.com/karyouben/Acme-SF-D04

**Índice de contenido**

[1. Resumen ejecutivo3](#_Toc158967746)

[2. Control de versiones4](#_Toc158967747)

[3. Introducción5](#_Toc158967748)

[4. Contenido6](#_Toc158967749)

[5. Conclusiones6](#_Toc158967750)

[6. Bibliografía19](#_Toc158967751)

# 1. Resumen ejecutivo

El documento de pruebas está dividido en dos apartados. El primero está centrado en pruebas funcionales y el segundo está centrado en pruebas de rendimiento.

# 2. Control de versiones

|  |  |  |
| --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** |
| 20/05/2024 | v1r0 | Creado el documento |
| 27/05/2024 | v2r0 | Terminado el documento |
|  |  |  |
|  |  |  |

# 3. Introducción

A continuación, mostramos los resultados de las pruebas. El primer apartado incluye un listado con los casos de prueba implementados, agrupados por implementación, con descripción sobre su efectividad a la hora de encontrar errores.

El segundo apartado incluye información sobre el rendimiento de la aplicación. Gráficas e intervalos de confianza 95% tomados en dos ordenadores distintos, además de un contraste de hipótesis de confianza respecto a qué ordenador es más potente.

# 4. Contenido

No se ha implementado ningún índice porque el framework crea los índices para id y los atributos de navegación automáticamente, no hay otro tipo de índice que haya que implementar según los querys de los repositorios relevantes al proyecto, historias de usuarios o asociaciones.

Además, en el créate al haber probado mas de 50 datos en solo esa feature al ejecutar todos los test a veces eclipse se ralentiza o se queda pillado y no corre bien el créate, por lo que habría que ejecutarlo de nuevo

Si hay algún tipo de error puede ser porque algún compañero haya tocado sus csv. Mi rama es karyouben y funciona perfectamente todo ahí:

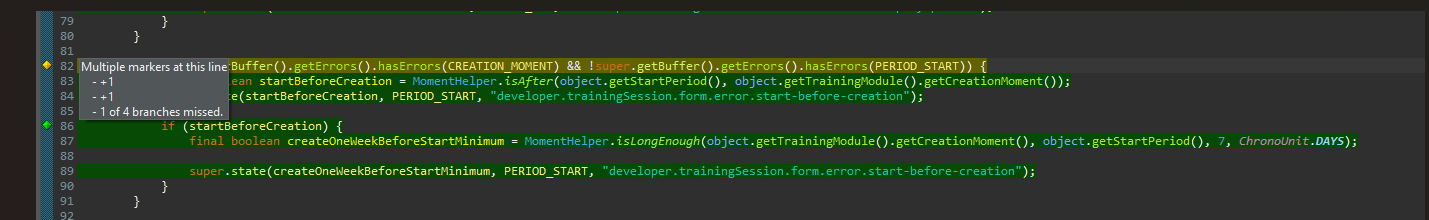
<https://github.com/karyouben/Acme-SF-D04.git>

Se añaden también los csv usados para la metodología de datos probados y hacking en todos los tests:  
(hoja de cálculo de la metodología usada en testing)  
<https://docs.google.com/spreadsheets/d/1AjAxVLVrZ2YukrSqX_RwuzNsiU_QYZ8VvQnXHhI2X7Y/edit?usp=sharing>

(hoja de cálculo de la metodología usada en hacking)  
<https://docs.google.com/spreadsheets/d/1dCT-zC0ePqIXSqy1jAUUZAcXpZhy07yZKieTTbzIRjw/edit#gid=0>

**Pruebas Funcionales**

**TrainingModule**

****

En las restricciones de fechas del creation/update del trainning sección sale el coverage amarillo porque una de las ramas no se puede probar, pues el creation moment es una fecha que se crea por defecto por el sistema y dichas restricciones están hecha para un doble check del mismo.

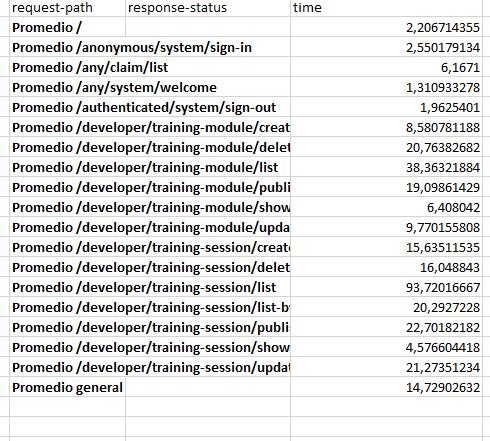
|  |  |
| --- | --- |
| Create.safe | Se inicia sesión como developer1 y se prueba a crear “Módulos de entrenamiento “de todas las formas que permite el sistema, comprobando los límites y los mensajes de error. No se detecta ningún error.  Se sigue la metodología de clase de ir probando todos los campos en blanco, los errores de cada campo y después probar un gran número de datos positivos, en mi caso he probado mas de 30 datos aproximadamente asegurando los límites. |
| Delete.safe | Se inicia sesión como developer1 y se elimina un proyecto que tiene hijos publicados para que salte la restricción y otro que tiene hijos no publicados el cual se borra satisfactoriamente. |
| List.safe | Se inicia sesión como developer1 y se listan sus “Módulos de entrenamiento” en inglés y en español. No se detecta ningún error. |
| Update.safe | Se inicia sesión como developer1 y se prueba a actualizar “Módulos de entrenamiento “en especial el de id 650 de todas las formas que permite el sistema, comprobando los límites y los mensajes de error. No se detecta ningún error.  Se sigue la metodología de clase de ir probando todos los campos en blanco, los errores de cada campo y después probar un gran número de datos positivos, en mi caso he probado más de 30 datos aproximadamente asegurando los límites. |
| Publish.safe | Se inicia sesión como developer1 y se publica un “Modulo de entrenamiento”. Se prueba a publicar el “Modulo de entrenamiento” cuando no tiene “trainning Sessións” asociadas para que salte la restricción, y cuando hay algún training sección sin publicar para que también, salte finalmente se prueba a publicar un “training Module” con todos sus hijos publicados y se publica satisfactoriamente. No se detecta ningún error.  Como el publish también actualiza los datos, se ha seguido la misma metodología que en el créate/update, de crear todos los valores posibles tanto positivos como negativos siguiendo la metodología de clase. |
| Show.safe | Se inicia sesión como developer1, se listan su training modules y se muestra cada training Module uno a uno. No se detecta ningún error. |
| Delete.hack | Primero se hace get de /developer/trainingModule/delete con id 656 (¿id=656), con id 1503 con usuario anónimo. Luego se inicia sesión como developer2 y se hace lo mismo. Finalmente se inicia sesión como developer1 y se hace hacking para id 656 (publicado) y el id 1583. No se detecta ningún problema. |
| List.hack | Se hace /developer/trainingModule/list con usuario anónimo. No se detecta ningún problema. |
| Update.hack | Se hace get de /developer/trainingModule/update con id 650 (no está publicado) y con id 6503 (no existe) con usuario anónimo. Se hace lo mismo como developer2 y se crea un trainingModule para hackear con la consola para el id 656 (del developer1) y 6503. Finalmente se inicia sesión como developer1 y se intenta actualizar el proyecto con id 656 (ya publicado) y 6503 (no existe). No se detecta ningún problema. |
| Publish.hack | Se hace get de /developer/trainingModule/publish con id 656 (no está publicado) y con id 6503 (no existe) con usuario anónimo. Se hace lo mismo como developer2 y se crea un proyecto para hackear con la consola para el id 656 (del developer1) y 6503 Finalmente se inicia sesión como developer1 y se intenta actualizar el proyecto con id 656 (ya publicado) y con un id inexistente 6503. No se detecta ningún problema. |
| Show.hack | Se hace get /developer/trainingModule/show con id 656 y 1654 con usuario anónimo. Se inicia sesión como developer2 y se hace lo mismo. No se detecta ningún problema |

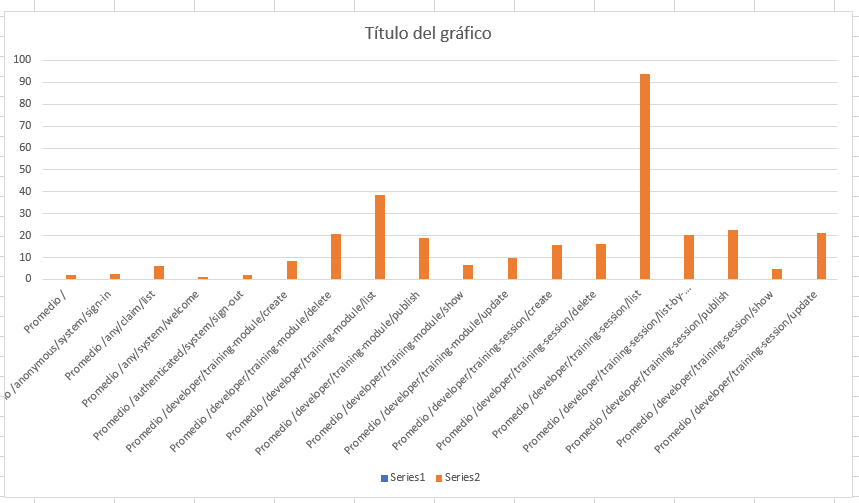
**Training Sessión**

|  |  |
| --- | --- |
| Create.safe | Se inicia sesión como developer1 y se prueba a crear training sessions de todas las formas que permite el sistema, comprobando los límites y los mensajes de error. No se detecta ningún error.  Se sigue la metodología de clase de ir probando todos los campos en blanco, los errores de cada campo y después probar un gran número de datos positivos, en mi caso he probado más de 30 datos aproximadamente asegurando los límites. |
| Delete.safe | Se inicia sesión como developer1 y se elimina un training sesión en draft mode |
| List.safe | Se inicia sesión como developer1 y se listan los trainings sessions. No se detecta ningún error. |
| List-by-trainingModule.safe | Se inicia sesión como developer1 y se listan los trainings sessions asociadas a cada trainingModule. Se listan todos en ingles y algunos en español. |
| Publish.safe | Se inicia sesión como developer1 y se publican y se crean varios trainnings sessions, probando todos los límites de pruebas.  Se sigue la metodología de clase de ir probando todos los campos en blanco, los errores de cada campo y después probar un gran número de datos positivos, en mi caso he probado más de 30 datos aproximadamente asegurando los límites. |
| Show.safe | Se inicia sesión como developer1, se listan su trainning sessions y se muestra cada una. No se detecta ningún error. |
| Update.safe | Se inicia sesión como developer1 y se prueba a actualizar los datos de varios trainings modules de todas las formas que permite el sistema, comprobando los límites y los mensajes de error. No se detecta ningún error.  Se sigue la metodología de clase de ir probando todos los campos en blanco, los errores de cada campo y después probar un gran número de datos positivos, en mi caso he probado más de 30 datos aproximadamente asegurando los límites. |
| List.hack | Se hace /developer/training-session/list con usuario anónimo. No se detecta ningún problema. |
| List-by-trainingModule.hack | developer/training-session/list-by-training-module?trainingModuleId=656 y 6743 con usuario anónimo. Luego se hace lo mismo iniciado sesión como developer2. No se detecta ningún problema. |
| Publish.hack | /developer/training-module/publish con ids 751 y 7224 tanto con usuario anónimo como siendo developer2. Además, se crea un training session siendo developer2 para hackear usando la consola pretendiendo publicar los ids 751 (del developer1), 7224 (no existe) y 126 (no publicado del developer2). No se detecta ningún problema. |
| Delete.hack | /developerr/training-session/delete con ids 724 y 16342 tanto con usuario anónimo como siendo developer2. Además, se crea un training sesión siendo developer2 para hackear usando la consola pretendiendo eliminar los ids 724 (del developer1), 16342 (no existe) y 722 (no publicado del developer1). No se detecta ningún problema. |
| Show.hack | Se hace get /developer/training-session/show con id 721 y 1673 siendo usuario anónimo. Se inicia sesión como developer2 y se hace lo mismo. No se detecta ningún problema |
| Update.hack | Se hace get /developer/training-session/update con ids 656 y 6503 tanto con usuario anónimo como siendo developer2. Se crea un training session para hackear usando la consola e intentar actualizar los ids 650, 656 y 6503. No se detectó ningún problema. |

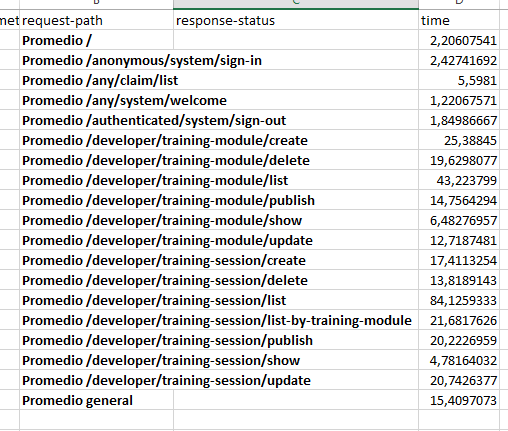
**Pruebas de Rendimiento**

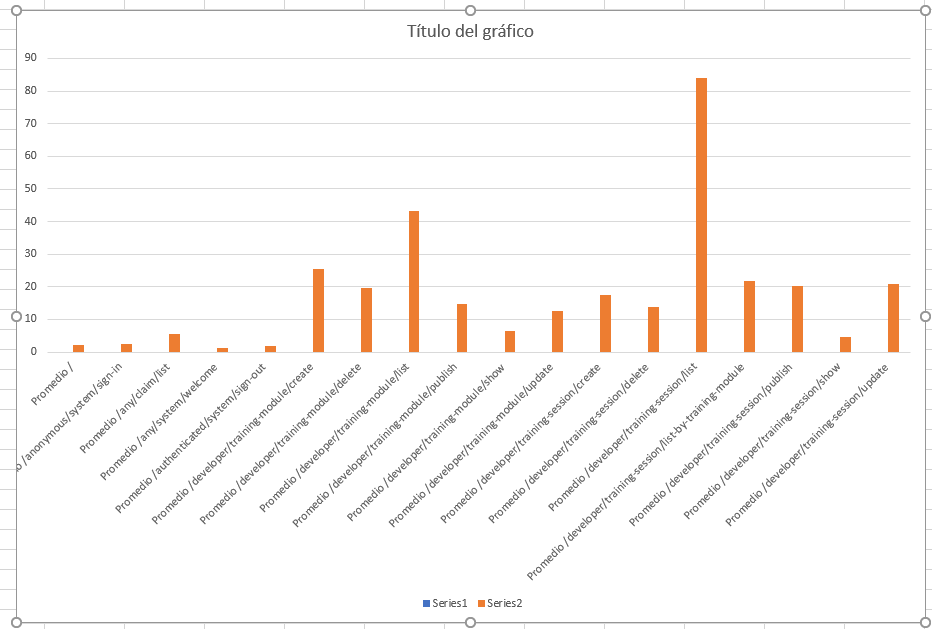
**PC A**

****

****

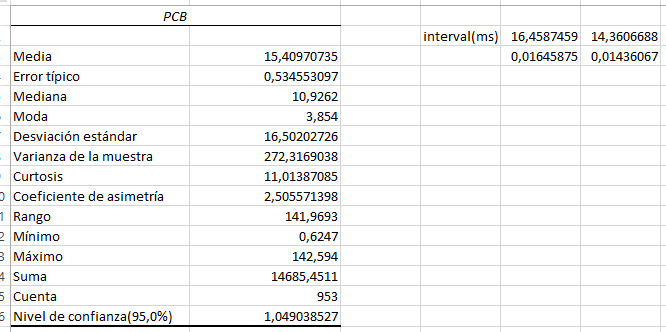
**PC B**

****

****

El PC B por lo general tiene tiempos más altos como se puede observar en la comparativa de las dos gráficas.

# 



# El PCA tiene intervalo de confianza 95% (15,66, 13,79) y el PCB tiene intervalo (16,45, 14,36). Son intervalos más que aceptables.

A continuación, calculamos la hipótesis de contraste con 95% confianza para intentar averiguar que ordenador es más potente:

# 

Como podemos observar, el valor P es menor que 0.2 Por esta razón sabemos que comparar las medias de los tiempos es una buena manera de averiguar que ordenador es más potente. En este caso PCA es mejor al tener una media de tiempos menor. Ese ordenador tiene un RYZEN 9000 y una 4090, el cual es bastante moderno, mientras que PCA tiene una CPU algo más antigua y bastante menos potente un RYZEN 7000, además de tener en cuenta que el PCA es un pc de sobremesa y PCB es un portátil.

# 5. Conclusiones

Este documento recoge las pruebas realizadas los cuales han servido para encontrar algún error en el código y calcular el rendimiento de nuestro código compara con dos ordenadores.

# 6. Bibliografía

Intencionadamente en blanco