INFORME DE PRUEBAS

GRUPO C1.02.03

Autores:

- **Alejandro Campano Galán** – alecamgal1@alum.us.es
- **Juan Jesús Campos Garrido** – juacamgar2@alum.us.es
- **David Cortabarra Romero** – davcorrom@alum.us.es
- **Alejandro García Sánchez-Hermosilla** – alegarsan11@alum.us.es
- **Pablo Mera Gómez** – pabmergom@alum.us.es

Repositorio: https://github.com/Alejandrocg024/Acme-L3-D04

Fecha del documento: 18/05/2023

Tabla de contenidos.

Resumen general..... 3
Tabla de revisión. 4
Introducción. 5
Contenido. 6
Conclusiones. 11
Bibliografía. 12

Resumen general.

En este documento se va a hablar sobre todas las pruebas del sistema, que han sido agrupadas por funcionalidad sobre entidades, resultando en diferentes tipos de casos de prueba: casos de prueba positivos, casos de prueba negativos y casos de prueba de hacking, algunos de estos no existen en realidad dado que no se necesita autorización para realizar operaciones en algunas entidades. En total el sistema tiene 13 pruebas.

Además de esto también analizamos el rendimiento de nuestra aplicación sobre la ejecución de las pruebas y hemos concluido que nuestra aplicación puede servir cualquier petición en un intervalo de 0.009927 y 0.01110 segundos.

Tabla de revisión.

Fecha	Versión	Descripción
27/04/2023	0.1	Creación del documento
18/05/2023	0.2	Contenido y conclusión

Introducción.

Después de haber acabado de implementar toda la funcionalidad y realizar pruebas de manera informal para comprobar el correcto funcionamiento la misma, ahora toca desarrollar de manera formal un conjunto de pruebas que se realicen de manera automatizada para que se comprueben que no hay fallos en la implementación, según lo que hemos aprendido, vamos a realizar las pruebas según 3 tipos de casos diferentes: casos de pruebas positivas, donde comprobamos que el sistema puede realizar operaciones y se comporta como debería; casos de pruebas negativas donde se realizan operaciones legales pero las cuales contiene errores manejables y el sistema encuentra fallos en la realización de dicha operación, y por ultimo casos de prueba de hacking en donde el usuario realiza operaciones ilegales donde el usuario no este autorizado para realizar. En este documento se registrarán todas las pruebas con estos casos.

Además, aparte de que nuestra aplicación funcione correctamente, el cliente espera que este funcionamiento sea en un rango de tiempo determinado, y por ello tras hablar de las pruebas vamos a realizar un análisis sobre el tiempo que ha llevado terminar cada prueba y de esta forma determinaremos el intervalo en el que todas las peticiones se pueden realizar.

Contenido.

Conjunto de pruebas realizadas:

Para los peeps:

List:

- Test100Positive()
  - Descripción: Sin iniciar sesión, accedemos al listado de los “peeps”, una vez en el listado, ordenamos y comprobamos que los “peeps” que aparecen tienen los datos que esperábamos
- Test101Positive()
  - Descripción: Iniciamos sesión como “lecturer1” y accedemos al listado de los “peeps”, una vez en el listado, comprobamos que el primer peep tiene los datos que esperábamos. Repetimos el proceso con los usuarios “assistant1”, “student1”, “auditor1”, “company1”, “administrator” y sin loguearnos. Esta prueba comprueba que cualquier usuario puede acceder al listado
-
  - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test200Negative()
  - No hay pruebas negativas para los listados.
- Test300Hacking()
  - No hay pruebas de hacking porque los peeps pueden ser listados por cualquier usuario de la aplicación.

Show:

- Test100Positive()
  - Descripción: Iniciamos sesión como “lecturer1” y accedemos al listado de los “peeps”, una vez en el listado, ordenamos y uno a uno vamos pulsando sobre el primer peep para ver sus detalles y comprobamos que sus datos son los que esperábamos. Repetimos el proceso con los usuarios “assistant1”, “student1”, “auditor1”, “company1”, “administrator” y sin loguearnos. Esta prueba sirve para comprobar que todos los usuarios del sistema pueden acceder los detalles de un peep
  - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test100Positive()
  - Descripción: Sin iniciar sesión accedemos al listado de los “peeps”, una vez en el listado, ordenamos y uno a uno vamos pulsando sobre los distintos “peeps” para ver sus detalles y comprobamos que sus datos son los que esperábamos.
  - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test200Negative()
  - No hay pruebas negativas a la hora de mostrar los detalles de un peep.

- Test300Hacking()
    - No hay pruebas de hacking porque los peeps pueden ser vistos por cualquier usuario de la aplicación.

Publish:

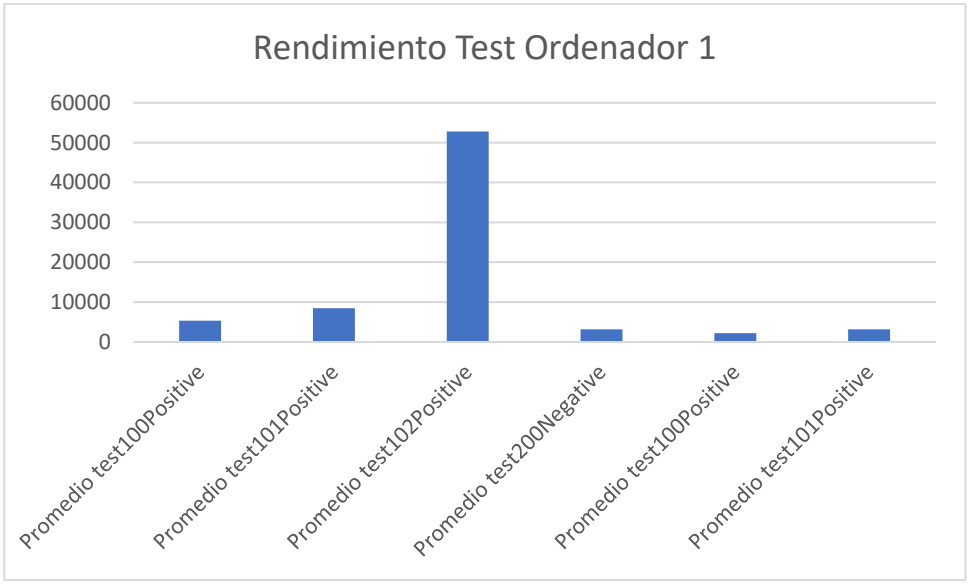
- Test100Positive()
    - Descripción: Iniciamos sesión como “lecturer1” y accedemos al listado de los “peeps”, una vez en el listado, pulsamos en el botón de crear un peep, rellenamos el formulario y comprobamos que se ha creado correctamente. Vamos repitiendo el proceso creando peeps siempre con los mismos datos, pero variando un campo para probar distintos valores válidos, por ejemplo, para las entradas de texto probamos con javascript, HTML, palabras con otros signos, mensajes de longitud uno, de longitud dos, de longitud igual al máximo para cada campo, de longitud igual al máximo menos uno. Para los emails se probarán emails correctos, que podrían ser del dominio de la universidad, de Google, de varios dominios y en cuanto a enlaces probaremos varios tipos. Todos valores aceptados por el sistema.
    - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test102Positive()
    - Descripción: Iniciamos sesión como “lecturer1” y accedemos al listado de los “peeps”, una vez en el listado, pulsamos en el botón de crear un peep, rellenamos el formulario sin poner un Nick, después comprobamos que se ha creado con los datos correctos. La función de esta prueba es comprobar que si estamos logueados e intentamos crear un peep, en Nick se rellenará solo con nuestros datos a pesar de que podamos cambiarlo si queremos.
    - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test102Positive()
    - Descripción: Iniciamos sesión como “lecturer1” y accedemos al listado de los “peeps”, una vez en el listado, pulsamos en el botón de crear un peep, rellenamos el formulario y comprobamos que se ha creado correctamente. Repetimos el proceso con los usuarios “assistant1”, “student1”, “auditor1”, “company1”, “administrator” y sin loguearnos. La función de esta prueba es comprobar que cualquier usuario puede crear un peep.
    - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test200Negative()
    - Descripción: Iniciamos sesión como “lecturer1” y accedemos al listado de los “peeps”, una vez en el listado, pulsamos en el botón de crear un peep, rellenamos el formulario siempre con los mismos datos, pero variando un campo para probar distintos valores no aceptados por el sistema, por ejemplo, para las entradas de texto probamos con cadenas superiores al máximo, con el formulario vacío. Para los emails se probarán palabras y

cadenas que no son emails y en cuanto a enlaces probaremos palabras y enlaces de longitud superior a 256 caracteres. Tras cada iteración comprobamos que el peep no se ha podido crear.

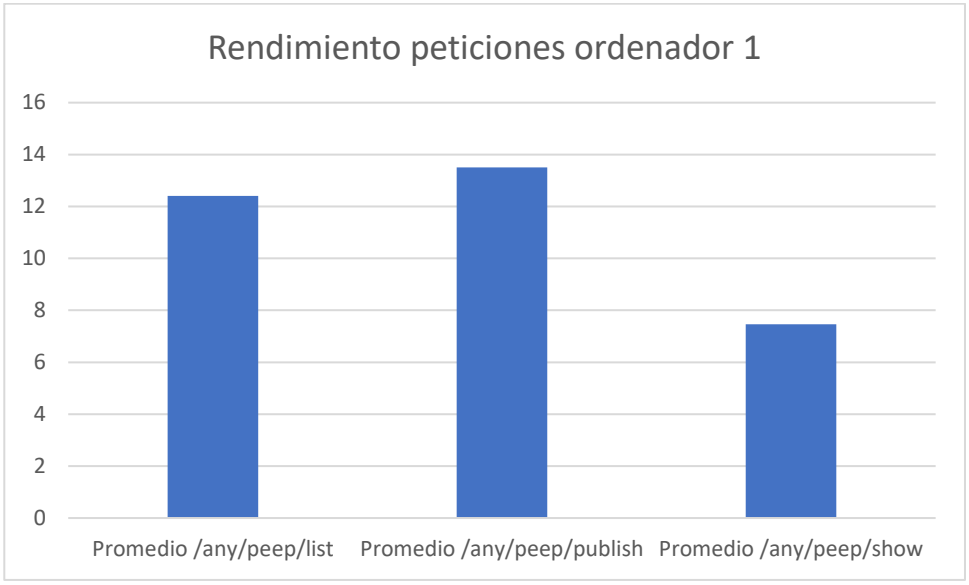
  - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test300Hacking()
  - No hay pruebas de hacking porque los peeps pueden ser creados por cualquier usuario de la aplicación.

Después de explicar de que tratan las diferentes pruebas previamente, vamos a ver cuál es el rendimiento del sistema y de sus pruebas para estas pruebas. Para poder observar el rendimiento a nivel global vamos a analizar diferentes aspectos de las pruebas y de las peticiones realizadas al sistema. Haremos el análisis de rendimiento con dos PC de diferentes componentes.

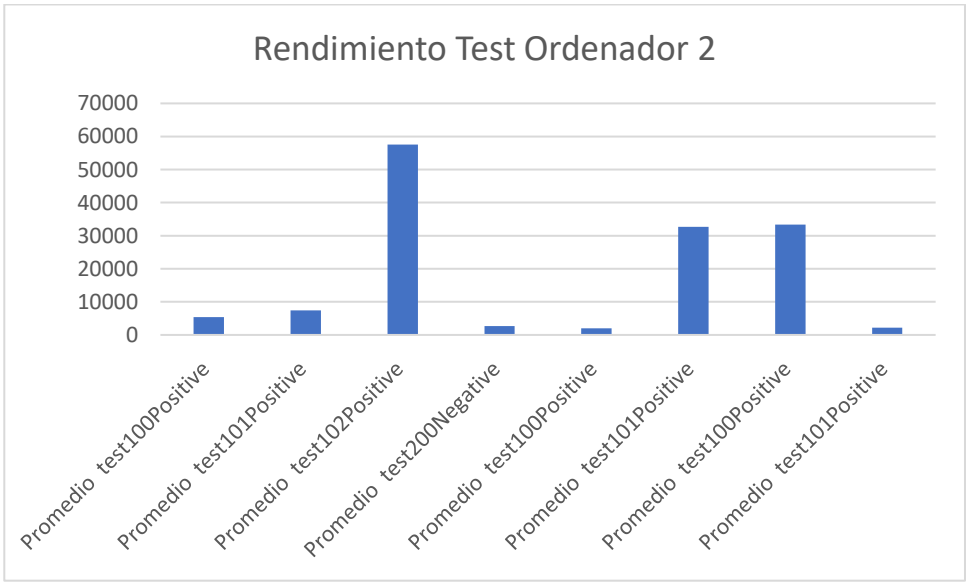
El primer ordenador cuyos componentes principales son, un procesador AMD Ryzen 7 5800H de 3.2 GHz y 16 Gigas de RAM, el tiempo que ha tardado en ejecutarse cada prueba puede verse en la siguiente gráfica:



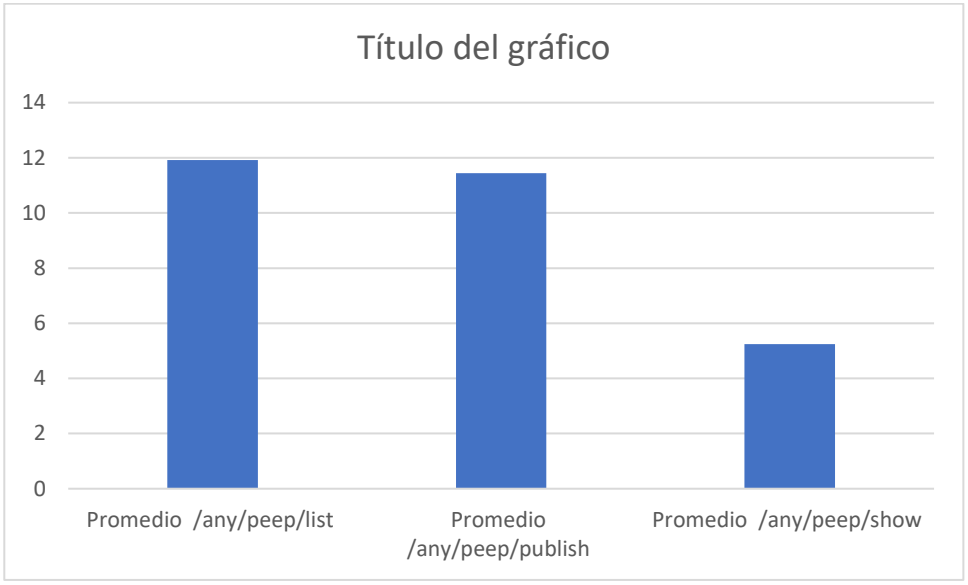
En cuanto a la velocidad con la que se han servido las peticiones en este ordenador, queda está gráfica:



El segundo ordenador tiene como componentes principales, un procesador Intel Cores i5-9300H 2.5 GHz y 16 Gigas de RAM, el tiempo que ha tardado en ejecutarse cada prueba puede verse en esta gráfica:



En cuanto a la velocidad con la que se han servido las peticiones en este ordenador, resulta en esta gráfica:



Tras haber comentado esto lo que nos quedaría por hacer son los cálculos para poder definir un rango de confianza en el cual podamos asegurar a el cliente que de media nuestra aplicación servirá todas las peticiones en un tiempo dentro de ese rango, para calcular este rango hemos utilizado la siguiente tabla de Excel:

Columna1					
			Interval (ms)	9,927939435	11,10595887
Media	10,51694915		Interval (s)	0,009927939	0,011105959
Error típico	0,299490323				
Mediana	10				
Moda	12				
Desviación es	5,634876801				
Varianza de l	31,75183656				
Curtosis	12,22976978				
Coeficiente c	2,416892916				
Rango	49				
Mínimo	2				
Máximo	51				
Suma	3723				
Cuenta	354				
Nivel de conf	0,589009717				

Conclusiones.

Para concluir este informe pues tenemos que analizar la efectividad final que ha resultado de realizar las pruebas del sistema, gracias a estas, hemos podido comprobar que la funcionalidad se ha implementado correctamente dado que no hemos encontrado fallo alguno en las pruebas.

Además de la eficiencia también podemos analizar los tiempos de ejecución que nos ha ayudado a demostrar a nuestro cliente que todas las peticiones se ejecutan de forma correcta en un tiempo dado entre un rango de valores que está determinado mediante un intervalo de confianza.

Bibliografía.

Intencionadamente vacío.