Universidad de Sevilla  
Escuela Técnica Superior de Ingeniería Informática

TESTING REPORT - D04



**C3.04.02**

**Repositorio**: [https://github.com/josrojrom1/C1.04.02](https://github.com/josrojrom1/C1.04.02%C3%A7)

**Autor:**

Ybarra Manrique, Miguel, migybaman[@alum.us.es](mailto:josrojrom1@alum.us.es)

**Fecha:** 22/10/2023

**Tabla de contenidos**

[**Resumen**](#_4hmpm5uc85lj) **2**

[**Tabla de revisión**](#_90otg4vxm3ka) **2**

[**Introducción**](#_58vtmc9pw6vo) **2**

[**Contenido**](#_tvy5edcq3b5t) **3**

[**Bibliografía**](#_ysgwrb1dc6k3) **11**

**Resumen**

En este documento se va a llevar a cabo un desglose detallado a cerca del

*testing suite* completo obtenido por el *coverage* de las pruebas unitarias *JUnit* realizadas sobre el proyecto en su versión final, en concreto sobre las funcionalidades relacionadas con *Peep*.

**Tabla de revisión**

|  |  |  |
| --- | --- | --- |
| **Versión** | **Fecha** | **Descripción** |
| V1 | 25/05/2023 | Testing report D04 |

**Introducción**

Nuestro requisito al que aplicar el *testing suite* es el **Requisito #21**, el cual enuncia lo siguiente:

***Operations by any principals on peeps:***

***-List the peeps and show their details.***

***-Publish a peep. By default, the nick must be kept blank if the principal is anonymous; otherwise, it must be filled with the principal’s full name; anyway, it can be changed at will.***

**Testing funcional**

**Funcionalidad:** *Peep*

**Rol:** *Any*

**Test:** List Peep with Rol Any

**Ruta:** *“acme>testing>any>peep>AnyPeepListTest.java”*

Para el **list** de *Peeps* se desarrollaron dos casos distintos:

* En el primer caso se comprobaba que para el usuario *Anonymous* se mostrasen todos los *Peeps* creados anteriormente en los archivos iniciales de la base de datos. Para ello se creó el fichero *.csv* con todos los *Peeps* creados y se comprueba que en el listado aparecen los atributos elegidos.
* En el segundo caso se comprueba para un usuario *Authenticated*, para el usuario *lecturer1*. Este caso se lleva a cabo de la misma manera que el anteriormente mencionado, se utiliza el mismo fichero *.csv* de casos positivos, porque se muestran los mismos *Peeps* a todos los *Principals*.

Con estos casos de testing para el listado no se han detectado ningún tipo de bugs.

**Test:** Show Peep with Rol Any

**Ruta:** *“acme>testing>any>peep>AnyPeepShowTest.java”*

Para el test ***show***de *Peeps* se desarrolló el siguiente caso:

* Para comprobar que la información del *Peep* se mostrase adecuadamente se creó el fichero *.csv* con un caso de *show* positivo. Para llevarlo a cabo de manera correcta, una vez que se accede al listado, este se ordena con orden ascendente en el campo *moment*, así nos aseguramos de que el primer elemento de la lista sea el que intentamos comprobar. Una vez ordenado se accede y se comprueba que los campos tienen los datos correctos.

**Test:** Publish Peep with Rol Any

**Ruta:** *“acme>testing>any>peep>AnyPeepPublishTest.java”*

Para el **publish** de Peeps se desarrollaron los siguientes casos:

* El primer caso es el test positivo de *publish Peep* donde se siguen los siguientes pasos siendo un usuario con Rol *Any*: Click en *“Peeps > listar Peeps”* y de nuevo click en *publish*, después se introducen todos los campos pertinentes del formulario con las entradas creadas en el fichero *.csv* pertinente. Al final se comprueba que no existen errores al publicar el *Peep*.
* El segundo caso es el test positivo de *publish Peep* donde se siguen los siguientes pasos siendo un usuario con Rol *Any*: Se inicia sesión con un usuario, click en *“Peeps > listar Peeps”* y de nuevo click en *publish*, después se introducen todos los campos pertinentes del formulario con las entradas creadas en el fichero *.csv* pertinente. Al final se comprueba que no existen errores al publicar el *Peep*. Este se diferencia del anterior en que se comprueba que el nick haya podido ser editado
* El tercer caso es el test negativo. En este test se repiten los mismos pasos que en el test anterior pero esta vez en el fichero *.csv* se introducen entradas erróneas comprobando que efectivamente se activan las validaciones correspondientes y que en ningún caso podemos publicar un *Peep*.
* El cuarto caso es el test negativo. Este es prácticamente el mismo caso que el anterior, pero se diferencian en que en este último se hace el sign in para poder comprobar que no se meta la entrada de nick de manera errónea.

**Testing de rendimiento**

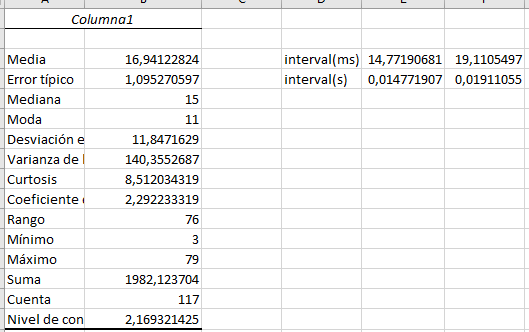
En la siguiente gráfica se puede analizar el rendimiento de cada petición http que se han hecho durante la ejecución de los tests. Como podemos comprobar, las peticiones fueron servidas en un promedio de menos de 8 milisegundos.

[ Eje Y en milisegundos | Eje X promedio de cada test por funcionalidad/entidad/rol ]

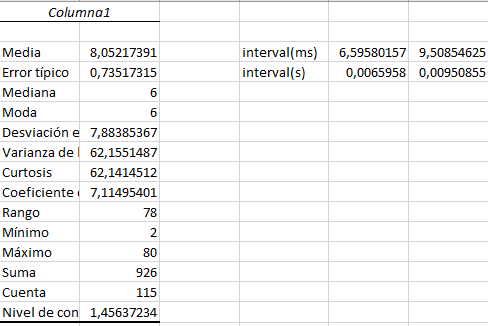
En la siguiente gráfica podemos ver el tiempo promedio en milisegundos que tardaron todos los tests en ejecutarse.

[ Eje Y en milisegundos | Eje X promedio de cada test individual ]

Usando la herramienta excel, tal y como viene explicado en la teoría, para analizar los datos, con el ordenador de Miguel Ybarra obtenemos los siguientes datos:



Si comparamos estos datos con los tests ejecutados con el ordenador de José Joaquín Rojas:



Podemos llegar a la conclusión de que el equipo de José Joaquín ejecutó los tests de manera mucho más rápida, ahorrándose casi la mitad del tiempo que tardó en el equipo de Miguel Ybarra.

# Bibliografía

* Documento 08 Annexes, de los contenidos de la plataforma virtual de esta asignatura.
* Documentos del lesson L04 - formal testing