Testing report D04

Grupo C1.03.10 - Group

https://github.com/DP2-2023/Acme-L3-D04

María del Carmen García García

Email: margargar81@alum.us.es

Francisco García de la Vega García

Email:fragargar15@alum.us.es

Ignacio Jordano de Castro

Email: ignjorde@alum.us.es

Javier Ulecia García

Email: javulegar@alum.us.es

Gabriel Vidal Tévar

Email: gabvidtev@alum.us.es

26/05/2023



Grado en Ingeniería del Software – Diseño y Pruebas II

Índice

[**Resumen ejecutivo 3**](#_52n8aip1s8vt)

[**Historial de versiones 3**](#_470a83r18fw2)

[**Introducción 4**](#_in8nh87qsr73)

[**Contenidos 4**](#_ww27ibfd8h)

[Pruebas funcionales 4](#_dv6fjdl1uvgj)

[Pruebas de rendimiento 5](#_2qjv9w4zqqlg)

[Resultados PC 1 5](#_8jxoh91ff6i6)

[Resultados PC 2 6](#_n3i9vxgdxx8f)

[Discusión de los resultados 7](#_4ayiqam9750r)

[**Conclusiones 7**](#_byyylknpky9l)

[**Bibliografía 7**](#_hrt1r1nxhyj)

# Resumen ejecutivo

En este informe se incluyen un resumen de los tests funcionales realizados sobre la parte grupal, y los resultados de los tests de rendimiento.

# Historial de versiones

| **Versión** | **Fecha** | **Descripción** |
| --- | --- | --- |
| 1.0 | 26/05/2023 | Versión inicial |

# Introducción

A continuación se resumen las pruebas implementadas para cada feature del requisito #21, incluyendo listado, publicación y visualización de Peeps.

Además, se realiza un estudio del rendimiento del sistema durante dichos tests, donde se contrastan los resultados obtenidos en dos equipos distintos.

Finalmente analizamos los resultados, y establecemos una serie de conclusiones acerca del rendimiento del sistema, y de las pruebas realizadas en general.

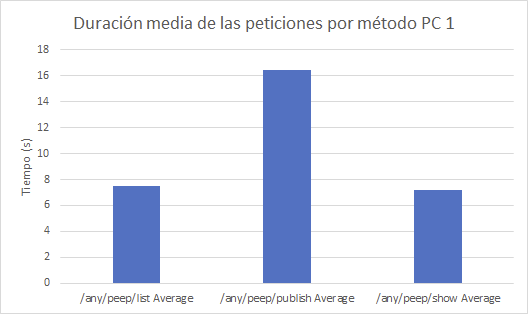
# Contenidos

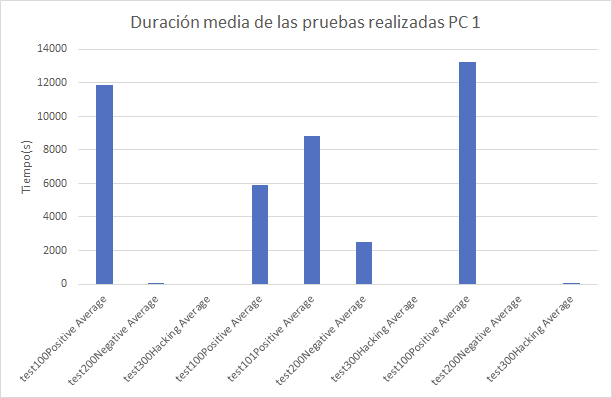
## Pruebas funcionales

| **Funcionalidad** | **Tests** |
| --- | --- |
| Listado de Peeps | * Positivas: Comprobación de que cada registro que debería ser visible en el listado es en efecto visible, con acceso desde los roles de usuario cualquiera, administrador, y lecturer. * Negativas: No aplica * Hacking: No aplica |
| Publicación de Peeps | * Positivas:   + Creación de peeps con valores aceptados desde el rol de usuario cualquiera. Comprobación de que el peep creado esta en el listado, y se puede mostrar con los campos adecuados.   + Creación de peeps con valores aceptados (usando el nick por defecto) desde el rol de usuario cualquiera. Comprobación de que el peep creado está en el listado, y se puede mostrar con los campos adecuados. * Negativas: Creación de peeps con valores negativos desde el rol de usuario cualquiera:   + Valores nulos   + Valores fuera de rango (número de caracteres)   + Contenido de spam   + Formato de email erróneo   + Formato de url erróneo   + Combinación de errores * Hacking: No aplica |
| Mostrar Peep | * Positivas: Mostrar los detalles de cada registro que debería ser visible. Comprobación de que los datos concuerdan con los esperados. * Negativas: No aplica * Hacking: No aplica |

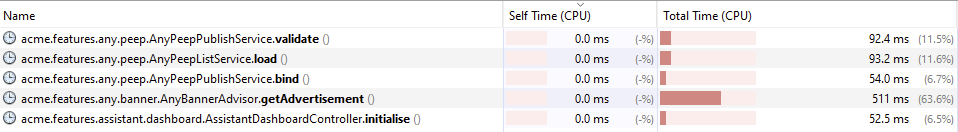
## Pruebas de rendimiento

### Resultados PC 1

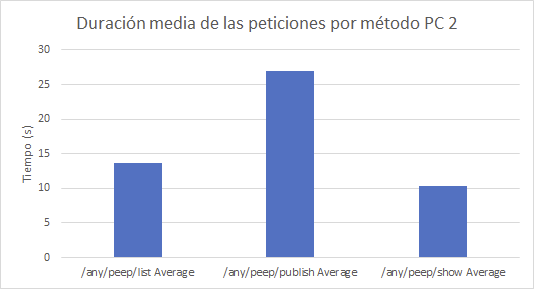


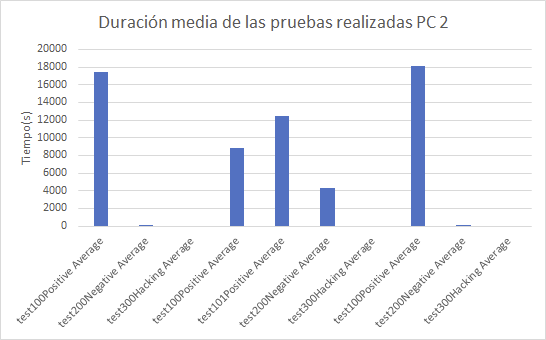


| **Interval (ms)** | 6.435694 | 13.98551 |
| --- | --- | --- |
| **Interval (s)** | 0.006436 | 0.013986 |

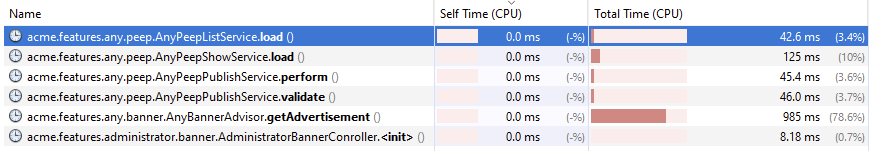


### Resultados PC 2





| **Interval (ms)** | 11.1883 | 19.15952 |
| --- | --- | --- |
| **Interval (s)** | 0.011188 | 0.01916 |



### Discusión de los resultados

En ambos PC se verifica inmediatamente el requisito de rendimiento establecido. Los tiempos medios por petición por feature son menores a 1s para un 95% de las peticiones. Es decir, el intervalo de confianza se encuentra dentro del rango aceptable, con ambos extremos dentro del intervalo, para ambos PCs.

Este fue el resultado desde un primer momento, sin necesidad de cambios en la implementación.

En el caso de los tests de monitorización, encontramos que el método más costoso es el de *getAdvertisements()*. Tras un estudio de la implementación, e intentos de reducir las consultas realizadas a la base de datos, o acelerarlas (uso de otras clases para generación de números aleatorios, y uso de índices aleatorios sobre la lista de banners en memoria, en lugar de Pageable) fuimos incapaces de mejorar el rendimiento. Concluimos que el cuello de botella no se debe a la implementación, y que probablemente se debe a las peticiones externas necesarias para obtener las imágenes del banner. En cualquier caso, el tiempo acumulado en *getAdvertisements()* a lo largo de todos los tests es menor a un segundo en el PC con peor rendimiento.

# Conclusiones

Las pruebas funcionales implementadas nos han permitido asegurarnos de que el sistema cumple su especificación.

El estudio realizado nos ha familiarizado métodos para el análisis del rendimiento, y en ese sentido ha sido exitoso. Además, hemos encontrado que la implementación realizada cumple los requisitos de rendimiento establecidos.

# Bibliografía

*Intencionadamente en blanco*