Diseño y pruebas 2

Testing report D04

Grupo C1.057

**Realizado por:**

*Alcantara Bernal, Pablo (*[*pabalcber@alum.us.es*](mailto:pabalcber@alum.us.es)*)*

Fecha: 26/05/2025

**Tabla de contenidos**

[**Tabla de contenidos 2**](#_heading=h.tkvs4ggqslwd)

[**Resumen ejecutivo 3**](#_heading=h.851b9rui3uz3)

[**Tabla de revisiones 3**](#_heading=h.u9zmxm1yanuv)

[**Introducción 3**](#_heading=h.n0dthyatragt)

[**Contenido 3**](#_heading=h.47idirgk1t71)

[**Conclusiones 3**](#_heading=h.lrup02dmhw13)

[**Bibliografía 3**](#_heading=h.iw0zwvo246bv)

**Resumen ejecutivo**

En este documento se resume el proceso y análisis de resultados del testing de la aplicación en la entrega D04. Para ello, se ha puesto a prueba el proyecto siguiendo los pasos descritos en las clases de la asignatura

**Tabla de revisiones**

| Número | Fecha | Autor | Descripción |
| --- | --- | --- | --- |
| 1 | 26/05/2025 | Pablo Alcántara Bernal | Creación del documento inicial |

**Introducción**

El contenido de este informe se va a dividir en dos partes: el testing funcional, en el que hablaremos de los casos de prueba implementados, agrupados por feature, y el performance testing.

**Contenido**

* Testing funcional

Para el testing funcional, se han dividido los tipos de pruebas en dos: pruebas “seguras”, es decir, acciones que un usuario podría intentar hacer legalmente, y pruebas de hacking, acciones no legales que un usuario podría intentar hacer.

* Pruebas seguras:

Tanto para las features de flight como las de leg, tenemos: list (listado), show (mostrar), create (crear), update (actualizar), delete (update) y publish (publicar).

Para el listado, sencillamente se comprueban todas las listas y todas las páginas de dichas listas.

Para el show, se comprueban todas las vistas de los datos.

Para el create, se comprueba uno por uno cada atributo, estando los demás vacíos, y se prueba con todos los datos erróneos posibles. Finalmente, se envía el formulario con los datos correctos y se comprueba si ha funcionado.

El proceso de update es similar al anterior.

Para delete y publish, lo que se hace es comprobar si funcionan las excepciones y las condiciones definidas en ellos para que se realicen las acciones. Por ejemplo, un flight no se puede borrar si tiene alguna leg publicada, y un flight no se puede publicar si no tiene todas sus legs publicadas.

* Pruebas de hacking

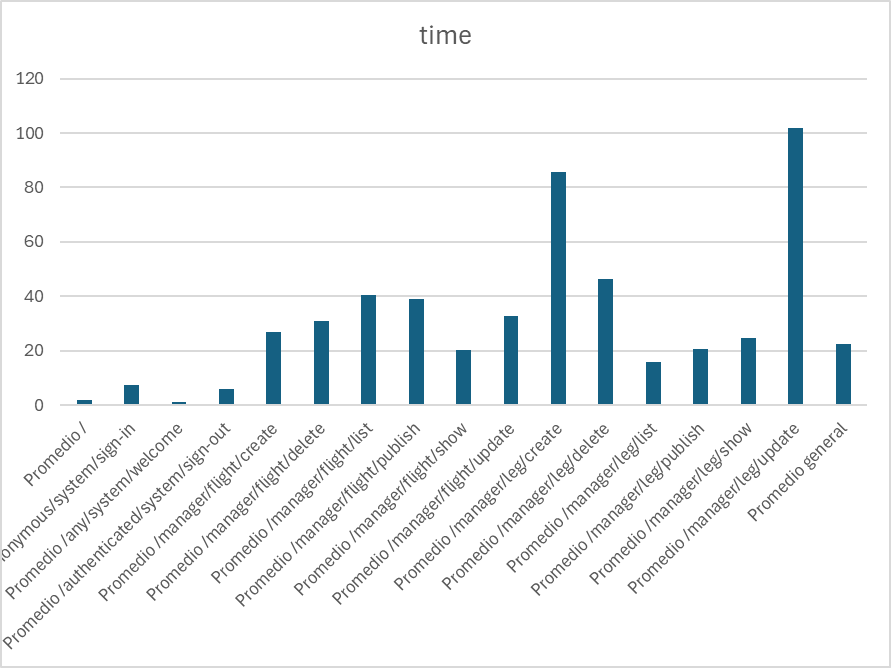
Para list y show, sencillamente se intenta acceder a estos recursos sin tener la autenticación necesaria para ellos.

Para create y update, se intenta incluir elementos que no se pueden incluir de forma autorizada, como atributos read-only, y se intenta acceder o enviar formularios sin tener la autenticación necesaria.

Para delete y publish, se intenta utilizar estos recursos incluso cuando el sistema no deja usarlos. Por ejemplo, se intenta borrar o publicar algo ya borrado o inexistente.

De todas estas pruebas, la que más me ayudó fue con el create de leg, que me mostró un bug en la validación entre fechas de salida y llegada. Por suerte, fue sencillo de solucionar, debido a que ese fallo ya había sido detectado en una entrega anterior y corregido en el update de la misma entidad.

* Performance testing



**Conclusiones**

El testing es clave para poder detectar errores que, en principio, suponemos que directamente no ocurren, ya que es muy probable que al programar no hayamos contemplado esa posibilidad. Es por ello que es necesario todo este proceso antes de entregar una aplicación a su cliente, para dar la mayor garantía posible de que el sistema es confiable y a prueba de errores.

**Bibliografía**

Intencionalmente en blanco.