**Testing Report Student #2**

**C2.019**

**Ignacio Martínez Díaz**

| **Grupo** | C2.019 |
| --- | --- |
| **Repositorio** | <https://github.com/adolfoborrego/Acme-ANS> |
| **Student #2** | **ID**: 77873179D  **UVUS**: SSK0456  **Nombre**: Martínez Díaz, Ignacio  **Roles**: Analyst |
| **Student #4** | **ID**: 52077055H  **UVUS**: TCP2748  **Nombre**: Sánchez Carmona, Germán  **Roles**: Developer |
| **Student #5** | **ID**: 54794337B  **UVUS**: CFV7375  **Nombre**: Regidor García, Miguel  **Roles**: Tester |

Índice

[**1. Introducción 2**](#_jhuf8yv9iz9o)

[1.1. Propósito del documento 2](#_vhw0ph3g883v)

[**2. Functional Testing 2**](#_2o7winp4n81o)

[2.1. Introducción 2](#_u0mptw7qxets)

[2.2. Cobertura de las pruebas 3](#_miyy5k1dkxh7)

[2.4. Conclusiones 6](#_l9jap5oujti4)

[2.4.1 Booking 6](#_kt34o7lfj3il)

[2.4.2 Passenger 6](#_hgxn21b2pkge)

[2.4.3 PassengerBooking 6](#_3eyns7o1toq)

[**3. Performance Testing 7**](#_e8k0vhxse4bq)

[3.1. Introducción 7](#_okqbw7wqfat6)

[3.2. Gráficos de eficiencia medios 7](#_hvholuf2b67l)

[3.3. Estadísticas descriptivas 9](#_tjedc5gtr4xl)

[3.4. Hipótesis y conclusiones 9](#_tzzfw1l1kz1z)

[4. Historial de versiones 10](#_mj8fdm2hfhu)

[**5. Bibliografía 10**](#_u8bxkmmvpl1i)

# 

# **1. Introducción**

## **1.1. Propósito del documento**

El propósito de este documento es presentar de forma estructurada y rigurosa los resultados obtenidos durante el proceso de pruebas del sistema desarrollado por el equipo. El informe tiene como objetivo principal verificar que las funcionalidades implementadas cumplen correctamente con los requisitos definidos y que el sistema ofrece un comportamiento estable, seguro y eficiente tanto a nivel funcional como de rendimiento.

A través de este documento se detalla la ejecución de pruebas funcionales, diferenciando entre versiones inseguras (.hack) y versiones protegidas (.safe), así como el análisis del rendimiento del sistema en diferentes condiciones, incluyendo comparativas estadísticas e intervalos de confianza. Este análisis permite no solo identificar errores o vulnerabilidades, sino también justificar decisiones técnicas tomadas durante el desarrollo para mejorar la calidad final del producto.

# **2. Functional Testing**

## **2.1. Introducción**

Para comprobar el correcto funcionamiento de las entidades **Booking**, **Passenger** y **PassengerBooking**, correspondientes a los requisitos del segundo estudiante, mediante el uso de tester record.

Para cada entidad se comprobó que todas las operaciones que gestionaban dichas entidades funcionaran de acuerdo a lo que se pedía. También se comprobó que ningún tipo de **GET/POST** hacking fuera posible.

## **2.2. Cobertura de las pruebas**

A continuación, se presenta una tabla detallada con el nivel de cobertura de pruebas alcanzado, tanto para el paquete **Booking** como para los paquetes **Passenger** y **PassengerBooking**.

| **Paquete** | **Cobertura** |
| --- | --- |
| customer.booking | 95,7% |
| customer.passenger | 94,0% |
| customer.passengerBooking | 94,4% |

Más detalle sobre Booking:

| **Servicio** | **Cobertura** |
| --- | --- |
| CustomerBookingListService | 100,0% |
| CustomerBookingShowService | 96,9% |
| CustomerBookingCreateService | 94,2% |
| CustomerBookingUpdateService | 94,1% |
| CustomerBookingPublishService | 96,6% |

Más detalle sobre Passenger:

| **Servicio** | **Cobertura** |
| --- | --- |
| CustomerPassengerListService | 97,0% |
| CustomerPassengerShowService | 96,1% |
| CustomerPassengerCreateService | 90,8% |
| CustomerPassengerUpdateService | 89,8% |
| CustomerPassengerPublishService | 96,4% |

| **Servicio** | **Cobertura** |
| --- | --- |
| CustomerPassengerBookingCreateService | 94,1% |

Más detalle sobre PassengerBooking:

**2.3. Casos de prueba**

Para Booking

| **Caso de prueba** | **Descripción** | **Eficacia** |
| --- | --- | --- |
| list.safe | Comprueba que el listado funciona correctamente. | No se detectó ningún fallo. |
| list.hack | Comprueba que no se puedan hacer GET hacking en el listado. | No se detectó ningún fallo |
| show.safe | Comprueba que se muestran las propiedades de booking correctamente. | No se detectó ningún fallo |
| show.hack | Comprueba que no se pueda acceder a un booking desde un customer no autorizado. | No se detectó ningún fallo |
| create.safe | Comprueba que se pueda crear correctamente un booking, contemplando así las validaciones de cada propiedad. | No se detectó ningún fallo |
| create.hack | Comprueba que el que esté creando el booking sea un customer, y que el vuelo asociado esté publicado y en el futuro. | No se detectó ningún fallo |
| update.safe | Comprueba que se puedan actualizar los datos de un booking, teniendo en cuenta las validaciones de dichos datos. | Saltó un panic al crear un booking sin seleccionar vuelo. |
| update.hack | Comprueba que el que esté actualizando el booking sea el customer de dicho booking, y que el vuelo asociado esté publicado y en el futuro (en caso de que se actualice). | No se detectó ningún fallo |
| publish.safe | Comprueba que se publique correctamente el booking, comprobando que tenga pasajeros y los últimos cuatro dígitos de la tarjeta. | No se detectó ningún fallo |
| publish.hack | Comprueba que el customer del booking que se va a publicar sea el mismo que el que lo está publicando. | No se detectó ningún fallo |

Para Passenger:

| **Caso de prueba** | **Descripción** | **Eficacia** |
| --- | --- | --- |
| list.safe | Comprueba que el listado funciona correctamente. | No se detectó ningún fallo |
| list.hack | Comprueba que no se puedan hacer GET hacking en el listado, y en caso de ser el listado de los passenger de un booking en concreto que el customer de ese booking sea correcto. | No se detectó ningún fallo |
| show.safe | Comprueba que se muestran correctamente los datos de passenger. | No se detectó ningún fallo |
| show.hack | Comprueba que el customer que accede a un passenger sea el mismo que el customer asociado a dicho passenger. | No se detectó ningún fallo |
| create.safe | Comprueba que se crea un passenger correctamente, validando cada una de las propiedades. | No se detectó ningún fallo |
| create.hack | Comprueba que el que crea un passenger es un customer. | No se detectó ningún fallo |
| update.safe | Comprueba que se puede actualizar un passenger, validando cada una de las propiedades. | No se detectó ningún fallo |
| update.hack | Comprueba que el passenger no esté publicado, y que el customer esté autorizado para modificar dicho passenger. | No se detectó ningún fallo |
| publish.safe | Comprueba que se publica el passenger sin problemas. | No se detectó ningún fallo |
| publish.hack | Comprueba que el customer esté autorizado a publicar dicho passenger. | No se detectó ningún fallo |

Para PassengerBooking:

| **Caso de prueba** | **Descripción** | **Eficacia** |
| --- | --- | --- |
| create.safe | Comprueba que se asocia un passenger a un booking correctamente. | No se detectó ningún fallo |
| create.hack | Comprueba que ambos pertenezcan al mismo customer, que el passenger esté publicado y que el booking no. | No se detectó ningún fallo |

## **2.4. Conclusiones**

### **2.4.1 Booking**

La entidad Booking ha sido sometida a pruebas funcionales, tanto en condiciones normales (.safe) como frente a intentos de acceso indebido (.hack). Los resultados reflejan una implementación robusta y segura, con una cobertura del 95,7% en su paquete. Los servicios específicos alcanzan coberturas destacables, como el CustomerBookingListService con un 100% y otros superando el 93%.

El único fallo detectado fue en la prueba update.safe, donde se lanzó un error en la creación de un booking con un flight nulo, indicando una posible falta de validación de de valor inexistente. Aun así, los mecanismos de seguridad y control de acceso han respondido correctamente en todos los casos .hack.

En resumen, la entidad Booking está bien implementada, siendo funcional, segura y con buena cobertura de pruebas.

### **2.4.2 Passenger**

La entidad Passenger también ha mostrado un comportamiento sólido frente a los distintos escenarios de prueba. Con una cobertura global del 94,0%, los servicios clave como CustomerPassengerListService (97%) y CustomerPassengerPublishService (96,4%) muestran una cobertura alta, aunque servicios como el de actualización (CustomerPassengerUpdateService) presentan una cobertura algo menor (89,8%), lo cual deja margen para pequeñas mejoras.

Todas las pruebas funcionales se superaron sin detectar fallos, lo que indica una correcta implementación de las validaciones en operaciones CRUD. Las pruebas de seguridad demostraron que los mecanismos de autorización funcionan correctamente, impidiendo accesos o modificaciones no permitidas.

En resumen, la entidad Passenger cumple con los requisitos funcionales y de seguridad, y cuenta con una cobertura de pruebas muy aceptable.

### **2.4.3 PassengerBooking**

En el caso de PassengerBooking, los dos casos de prueba (create.safe y create.hack) se ejecutaron exitosamente, sin detectarse fallos. La cobertura del servicio principal (CustomerPassengerBookingCreateService) es del 94,1%, lo que demuestra que la lógica de asociación entre pasajeros y bookings está bien cubierta.

En resumen, aunque la funcionalidad de esta entidad es más pobre, su comportamiento es correcto y seguro, con un nivel de cobertura y validación satisfactorios.

# **3. Performance Testing**

## **3.1. Introducción**

El propósito de este apartado es analizar el rendimiento del sistema mediante pruebas de tipo performance testing. A diferencia de las pruebas funcionales, cuyo objetivo principal es verificar el comportamiento correcto del sistema, las pruebas de rendimiento se centran en medir el tiempo de respuesta de las distintas funcionalidades bajo condiciones controladas.

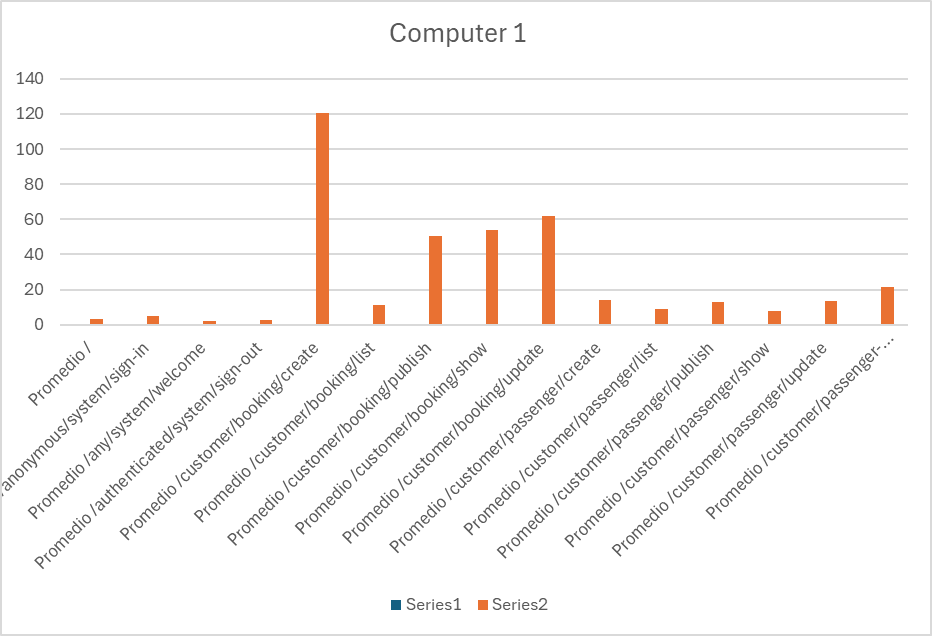
Para llevar a cabo este análisis, se han reutilizado los casos de prueba funcionales ya existentes, reproduciéndolos (replay) en dos equipos portátiles diferentes. De este modo, se ha obtenido un conjunto de datos reales de ejecución a partir de los cuales se ha podido evaluar la eficiencia del sistema y comparar el rendimiento relativo de ambos entornos.

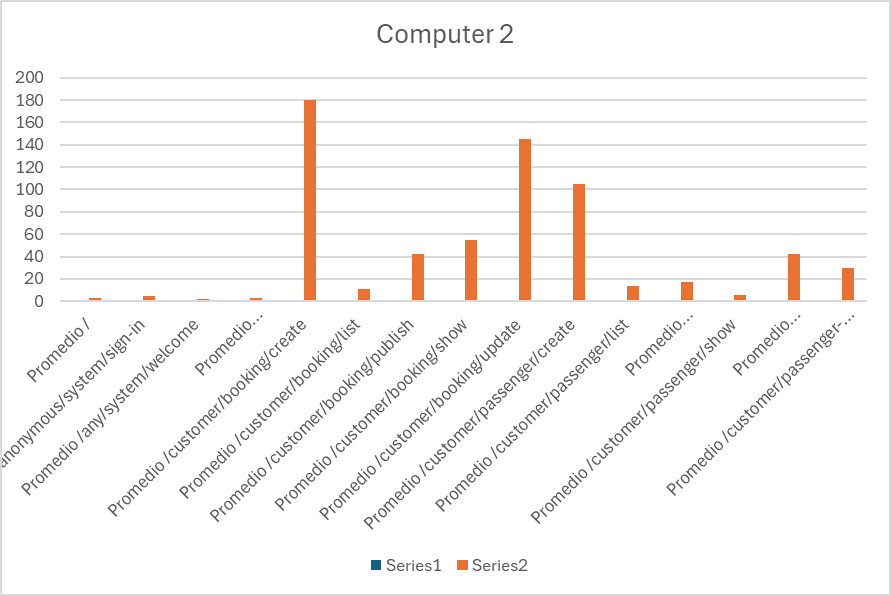
El análisis se ha realizado siguiendo la metodología propuesta, que incluye el cálculo de intervalos de confianza para el tiempo medio de respuesta y la aplicación de una prueba Z para dos muestras independientes, con el objetivo de determinar si las diferencias observadas entre ambos portátiles son estadísticamente significativas.

En las siguientes secciones se presentan los resultados obtenidos, tanto en forma de gráficos de eficiencia media como de análisis estadísticos que permiten extraer conclusiones fundamentadas sobre el rendimiento de la aplicación.

## **3.2. Gráficos de eficiencia medios**

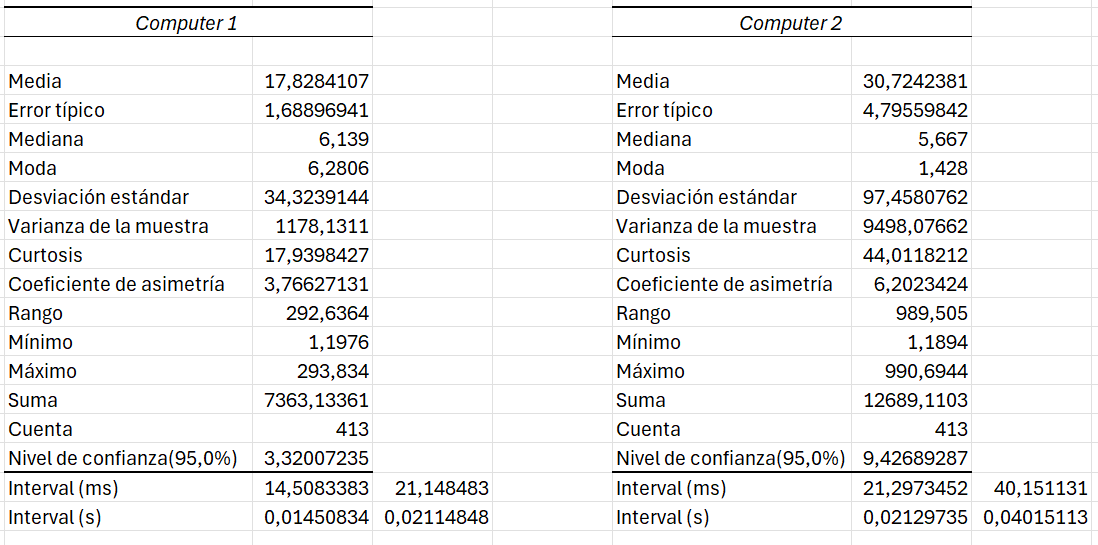
Los gráficos muestran los tiempos medios de respuesta (en milisegundos) al ejecutar las pruebas funcionales del sistema en dos equipos distintos: primero mi portatil propio y segundo el portatil de uno de mis compañeros de grupol. Se observa que, en general, el ordenador propio ofrece un rendimiento bastante superior, con menores tiempos de respuesta en la mayoría de las operaciones.





## **3.3. Estadísticas descriptivas**

Como paso previo al contraste de hipótesis, se ha realizado un análisis estadístico descriptivo de los tiempos de respuesta registrados al ejecutar los casos de prueba en ambos portátiles. Este análisis permite obtener una visión general del comportamiento del sistema en cada equipo y calcular el intervalo de confianza para el tiempo medio de respuesta.

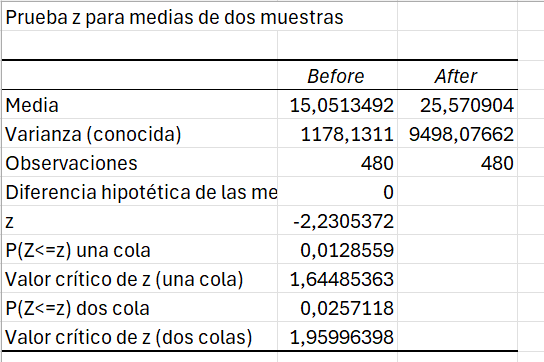


Ambos portátiles cumplen con el supuesto de que la media de respuesta se mantiene muy por debajo del segundo (1 s), cumpliendo así el requisito de rendimiento utilizado como referencia.

## 

## **3.4. Hipótesis y conclusiones**

Tras obtener los intervalos de confianza y las estadísticas descriptivas de ambos portátiles, se ha realizado un contraste de hipótesis mediante una prueba Z para dos muestras independientes, con el objetivo de determinar si la diferencia entre los tiempos medios de respuesta es estadísticamente significativa. La prueba Z se ha aplicado sobre los conjuntos de datos recogidos en Ordenador 1 y Ordenador 2, considerando un nivel de significación del 5 % (α = 0.05).



Dado que el valor de P(Z<=z) dos colas se encuentra en el intervalo (α,1], podemos afirmar que no existen grandes diferencias en el rendimiento entre ambos portátiles.

# **4. Historial de versiones**

| **Versión** | **Fecha** | **Descripción** |
| --- | --- | --- |
| 0.0 | 25/05/2025 | Versión inicial. |
| 1.0 | 26/05/2025 | Versión final. |
| 2.0 | 02/07/2025 | Versión actualizada para second call. |

# **5. Bibliografía**

Intentionally Blank