Testing Report Student #2
C1.019
Ignacio Martínez Díaz

Grupo	C1.019
Repositorio	https://github.com/adolfoborrego/Acme-ANS
Student #1	**ID**: 29584665H **UVUS**: XXB5458 **Nombre**: Borrego González, Adolfo Agustín **Roles**: Project Manager
Student #2	**ID**: 77873179D **UVUS**: SSK0456 **Nombre**: Martínez Díaz, Ignacio **Roles**: Analyst
Student #3	**ID**: 12830191D **UVUS**: PVL1690 **Nombre**: Mir Ceballos, Miguel **Roles**: Operator
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

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	95,5%
customer.passenger	94,0%
customer.passengerBooking	94,4%

Más detalle sobre Booking:

Servicio	Cobertura
CustomerBookingListService	100,0%
CustomerBookingShowService	96,9%
CustomerBookingCreateService	93,9%
CustomerBookingUpdateService	93,7%
CustomerBookingPublishService	96,5%

Más detalle sobre Passenger:

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

Más detalle sobre PassengerBooking:

Servicio	Cobertura
CustomerPassengerBookingCreateService	94,1%

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.	Se detectó un fallo al poner un locatorCode ya existente.
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.	No se detectó ningún fallo
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 95,5% 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 create.safe, donde se lanzó un error en la creación de un booking con un locatorCode ya existente, indicando una posible falta de validación de unicidad. 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.

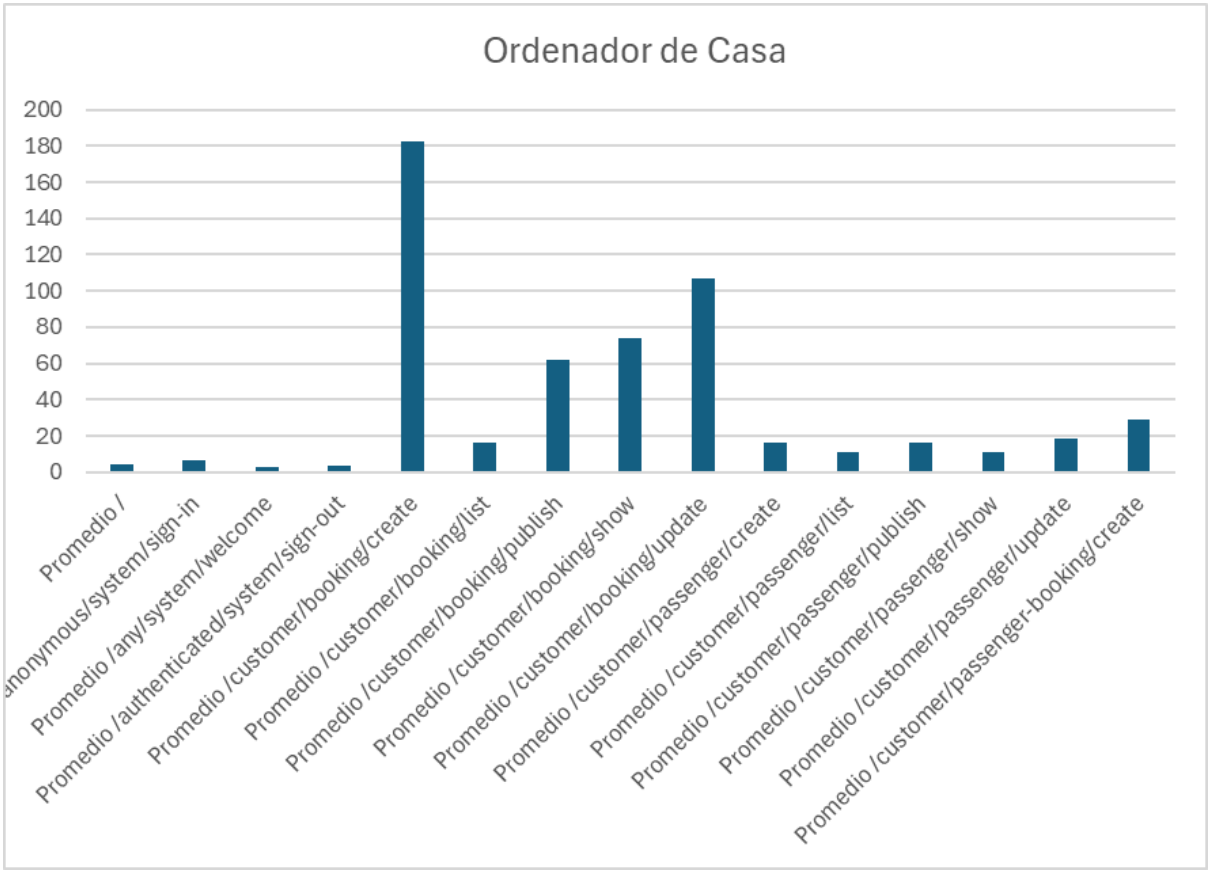
Para llevar a cabo este análisis, se han reutilizado los casos de prueba funcionales ya existentes, reproduciéndolos (replay) en dos equipos portátiles diferentes. De este modo, se ha obtenido un conjunto de datos reales de ejecución a partir de los cuales se ha podido evaluar la eficiencia del sistema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

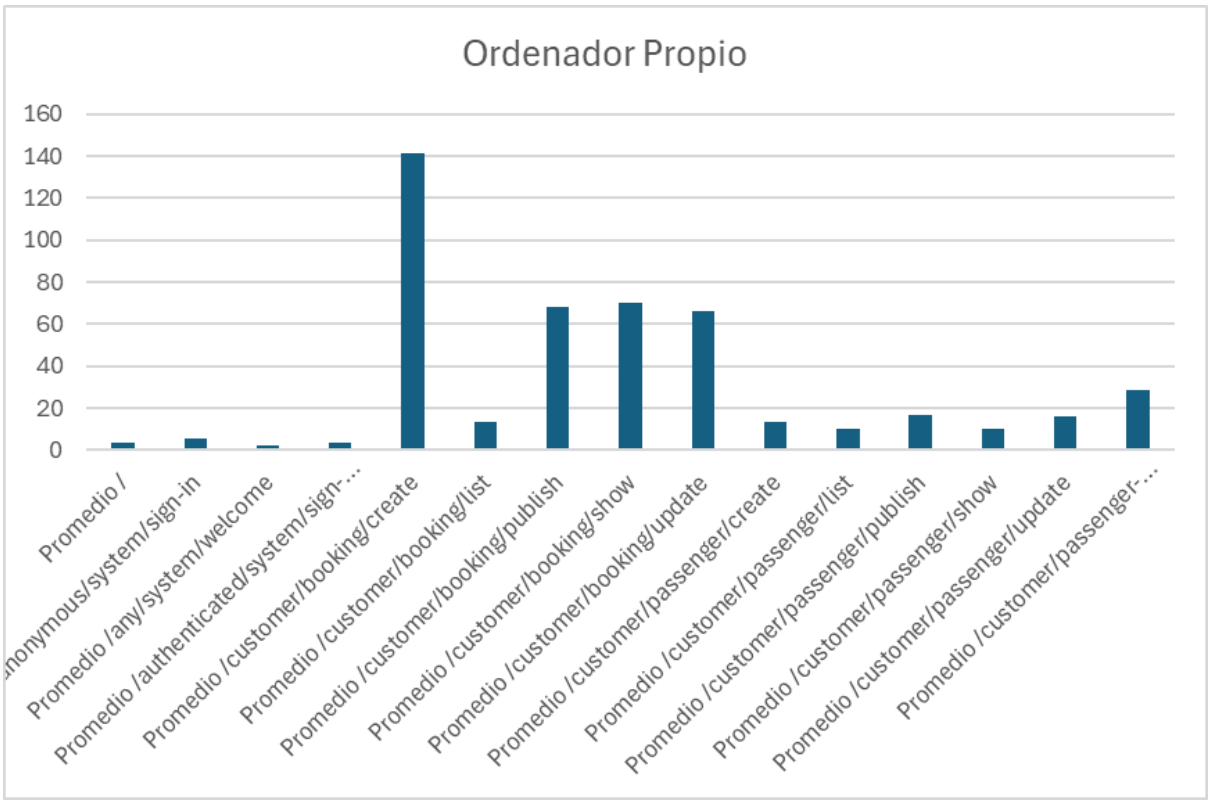
El análisis se ha realizado siguiendo la metodología propuesta, que incluye el cálculo de intervalos de confianza para el tiempo medio de respuesta y la aplicación de una prueba Z para dos muestras independientes, con el objetivo de determinar si las diferencias observadas entre ambos portátiles son 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: un ordenador de casa y mi ordenador personal. Se observa que, en general, el ordenador propio ofrece un rendimiento ligeramente superior, con menores tiempos de respuesta en la mayoría de las operaciones.


Ordenador de Casa
200
180
160
140
120
100
80
60
40
20
0
Promedio /
nonymous/system/sign-in
Promedio /any/system/welcome
Promedio /authenticated/system/sign-out
Promedio /customer/booking/create
Promedio /customer/booking/list
Promedio /customer/booking/publish
Promedio /customer/booking/show
Promedio /customer/booking/update
Promedio /customer/passenger/create
Promedio /customer/passenger/list
Promedio /customer/passenger/publish
Promedio /customer/passenger/show
Promedio /customer/passenger/update
Promedio /customer/passenger-booking/create


Ordenador Propio
160
140
120
100
80
60
40
20
0
Promedio /
nonymous/system/sign-in
Promedio /any/system/welcome
Promedio /authenticated/system/sign-...
Promedio /customer/booking/create
Promedio /customer/booking/list
Promedio /customer/booking/publish
Promedio /customer/booking/show
Promedio /customer/booking/update
Promedio /customer/passenger/create
Promedio /customer/passenger/list
Promedio /customer/passenger/publish
Promedio /customer/passenger/show
Promedio /customer/passenger/update
Promedio /customer/passenger-...

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.

Before			*After*		
Media	23,8228018		Media	12,5922371	
Error típico	2,54647486		Error típico	1,16723674	
Mediana	7,9666		Mediana	3,0958	
Moda	#N/D		Moda	1,9722	
Desviación es	49,8354954		Desviación es	22,8432732	
Varianza de la	2483,5766		Varianza de la	521,815132	
Curtosis	14,7568174		Curtosis	14,3871364	
Coeficiente d	3,66297333		Coeficiente d	3,50887876	
Rango	363,4286		Rango	167,062	
Mínimo	1,4574		Mínimo	1,2764	
Máximo	364,886		Máximo	168,3384	
Suma	9124,1331		Suma	4822,8268	
Cuenta	383		Cuenta	383	
Nivel de confi	5,00686234		Nivel de confi	2,29501329	
Intervalo(ms)	18,8159395	28,8296642	Intervalo(ms)	10,2972238	14,8872504
Intervalo(s)	0,01881594	0,02882966	Intervalo(s)	0,01029722	0,01488725

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).

Prueba z para medias de dos muestras		
	Before	*After*
Media	20,2994357	16,7697254
Varianza (conocida)	2483,5766	521,815132
Observaciones	468	468
Diferencia hipotética de las medias	0	
z	1,39287274	
P(Z<=z) una cola	0,08182914	
Valor crítico de z (una cola)	1,64485363	
Valor crítico de z (dos colas)	0,16365828	
Valor crítico de z (dos colas)	1,95996398	

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.

5. Bibliografía

Intentionally Blank