TESTING REPORT

Raquel Garcia Hortal – Student 2

Resumen ejecutivo:

Este documento recoge de forma completa y estructurada el conjunto de pruebas realizadas para verificar el correcto funcionamiento y la seguridad de las funcionalidades implementadas por el Estudiante #2. En concreto, se han evaluado las operaciones relacionadas con reservas de vuelos, gestión de pasajeros y la vinculación entre ambos conceptos mediante registros de reserva.

Las pruebas funcionales han contemplado tanto escenarios válidos como intentos de manipulación mediante el uso de herramientas de desarrollo del navegador (DevTools), prestando especial atención a los casos borde, restricciones de validación, control de acceso y comportamiento ante entradas no autorizadas. Se ha seguido un enfoque exhaustivo que asegura el cumplimiento de la normativa vigente, en particular la Ley Orgánica 3/2018 sobre protección de datos.

Finalmente, el documento también presenta una evaluación del rendimiento, comparando tiempos de ejecución entre diferentes entornos y ofreciendo conclusiones claras sobre la eficiencia de la solución.

Tabla de revisiones

Revisión	Fecha	Descripción
1.0	25/05/2025	Tests de bookings
1.1	26/05/2025	Test de Passengers
2.0	26/05/2026	Fin del documento

Introducción:

El propósito de este informe es documentar y justificar todas las pruebas realizadas sobre las funcionalidades asignadas al Estudiante #2 en el desarrollo del sistema. Dichas funcionalidades forman parte del rol Customer y abarcan principalmente la creación, edición, publicación y visualización de reservas (Bookings), así como la gestión de pasajeros (Passengers).

A lo largo del informe se presentan pruebas positivas (funcionalidades que deben aceptarse), negativas (datos inválidos o situaciones incorrectas) y de hacking (intentos de manipular o acceder sin autorización a través de DevTools o modificando parámetros).

Estas pruebas permiten asegurar la fiabilidad del sistema, su seguridad frente a usos indebidos, y el cumplimiento de los requisitos funcionales establecidos en el proyecto, tanto desde una perspectiva técnica como legal.

Atención:

Las funcionalidades están implementadas siempre desde la entidad Booking, lo cual implica que no existe una interfaz directa para gestionar BookingRecord por separado en la aplicación. En otras palabras, no hay un formulario, botón o URL dedicado a crear, listar, mostrar o eliminar BookingRecord.

Esta entidad intermedia (BookingRecord) se gestiona de forma implícita dentro de otras funcionalidades, en concreto:

- En la creación y actualización de reservas (Booking), se seleccionan pasajeros y, como consecuencia, se crean o actualizan automáticamente los registros de BookingRecord.
- En la funcionalidad de gestión de pasajeros (Passenger), también se tiene en cuenta su asociación a reservas a través de BookingRecord, por ejemplo, para comprobar si un pasajero puede modificarse o no.

Por este motivo, no se ha desarrollado una batería específica de pruebas funcionales o de hacking para BookingRecord, ya que sus efectos y restricciones se validan indirectamente a través de las pruebas sobre Booking y Passenger

Pruebas funcionales

1. **Operaciones de customers sobre bookings**
   - **1.1. List**
     - ***1.1.1. Safe Cases***

Se ha probado al listado de los bookings desde Customer1, Customer2 y Customer3, actualizando la página continuamente para confirmar que todo se renderiza perfectamente.

- Detección de bugs: No se detectó ningún bug.

     - ***1.1.2. Hacking cases***

Se ha probado a entrar al listado con la url de bookings usando realms, en este caso, administrator1 y member1.

- Detección de bugs: No se detectó ningún bug.

   - **1.2. Show**
     - **1.2.1. Safe Cases**

Se ha probado al show de cada booking desde Customer1, Customer2 y Customer3, actualizando la página continuamente para confirmar que todo se renderiza perfectamente.

- Detección de bugs: No se detectó ningún bug.

     - **1.2.2. Hack cases**

Se ha probado a entrar al listado con la url de show bookings usando realms, en este caso, administrator1 y member1.

- Detección de bugs: No se detectó ningún bug.

1.3. Update

1.3.1. *Safe cases*

Se ha probado con:

- Editar una booking con todos los datos vacíos.
- Editar una booking con un locatorCode en blanco.
- Editar una booking con un locatorCode duplicado (ya existe en otra booking del mismo customer).
- Editar una booking con un locatorCode inválido (menos de 6 caracteres, más de 8, minúsculas o símbolos).
- Editar una booking con purchaseMoment en el futuro y vacío.
- Editar una booking con travelClass nulo.
- Editar una booking con lastNibble en blanco o nulo (válido).
- Editar una booking con lastNibble inválido (letras o más/menos de 4 dígitos).

- Detección de bugs: No se detectó ningún bug.

1.3.2. Hack cases

Se ha probado con:

- Realm incorrecto: intentar acceder a /customer/booking/update?id=X como administrator1 y member1.
- Usuario incorrecto: intentar modificar una booking que pertenece a otro customer.
- Acción ilegal: intentar modificar una booking publicada (violación de isDraftMode).
- Atributo de navegación: modificar el campo Class Type con un valor que no existe.
- Campo read-only: cambiar el valor de Flight a un vuelo no publicado

- Detección de bugs: No se detectó ningún bug.

1.4. Create

1.4.1. Safe cases

Se ha probado a crear una booking con parámetros inválidos como:

- Formulario en blanco
- locatorCode:
  - Vacío → locatorCode = ""
  - Muy corto → locatorCode = "A1B2"
  - Muy largo → locatorCode = "ABCDEFGH1"
  - Con minúsculas → locatorCode = "abc123" (debe ser mayúsculas)
  - Con caracteres raros → locatorCode = "AB#12@"
  - Duplicado → usar un código ya registrado en la base de datos
- PurchaseMoment:
  - Futuro → una fecha como más tarde que 01/01/2025
  - Vacío → sin rellenar
  - Formato no válido ("ayer")
- lastNibble incorrecto
  - Menos de 4 dígitos → "12"
  - Más de 4 dígitos → "12345"
  - Letras incluidas → "12A4"
  - Campo vacío debe ser válido (porque es @Optional)
- flight : no seleccionar ningún vuelo → null

Crear una booking para customer2 con un código que ya tiene customer1.

Y por último una con datos válidos para validar que se creaba.

- Detección de bugs: No se detectó ningún bug.

1.4.2. Hack cases

Se han probado los siguientes casos:

- Realm incorrecto: crear booking como administrator1 o member1.
- Campo read-only: forzar el campo Price a que no sea readOnly
- Atributo de navegación: añadir al desplegable de flight un vuelo que aún no ha sido publicado.
- Atributo de navegación: añadir al desplegable de travel Class un valor que no está permitido.

- Detección de bugs: No se detectó ningún bug.

1.5. Publish

1.5.1. Safe cases

Se ha probado con customer1 los siguientes casos:

- Publicar una booking con pasajeros sin publicar..
- Publicar una booking sin pasajeros
- Publicar una booking sin valor en lastnibble.
- Publicar una booking con valor inválido en lastnibble.
- Publicar una boooking sin todos los pasajeros publicados.

- Detección de bugs: No se detectó ningún bug.

1.5.2. Hack cases

Se ha probado:

- Realm incorrecto: intentar publicar como administrator1 y member1.
- Usuario incorrecto: intentar publicar una booking ajena.
- Acción ilegal: intentar volver a publicar una booking ya publicada.
- Campo read-only: manipular el campo Vuelo para añadir un vuelo que no existe el antes de enviar el formulario.

2. Operations of customers on passengers

2.1. List

2.1.1. Safe cases

Se ha probado al listado de los pasajeros desde Customer1 y Customer2 actualizando la página continuamente para confirmar que todo se renderiza perfectamente.

- Detección de bugs: No se detectó ningún bug.

2.1.2. Hack cases

Se ha probado a:

- Entrar al listado con la url de bookings usando realms, en este caso, administrator1 y member1.
- Entrar en el listado de los pasajeros de una booking de otro customer.

- Detección de bugs: No se detectó ningún bug.

2.2. Show

2.2.1. Safe cases

Se ha probado al listado de cada pasajero desde Customer1 y Customer2 actualizando la página continuamente para confirmar que todo se renderiza perfectamente.

- Detección de bugs: No se detectó ningún bug.

2.2.2. Hack cases

Se ha probado a:

- Entrar al show de un pasajero de una booking usando realms, en este caso, administrator1 y member1.
- Entrar en el show de un pasajero de otro customer.

2.3. Update

2.3.1. Safe cases

Se ha probado a actualizar un pasajero con los siguientes valores inválidos o incorrectos, esperando que el sistema muestre los errores correspondientes:

- Formulario completamente vacío.
- FullName: vacío, con un único carácter, y con más de 255 caracteres.
- Email: con formato inválido (sin @, sin dominio, espacios…), y vacío.
- Passport: con menos de 6 o más de 9 caracteres, caracteres inválidos (símbolos, minúsculas), vacío y duplicado con otro pasajero.
- Birth: con fecha futura o no indicada.
- SpecialNeeds: con más de 50 caracteres

- Detección de bugs: No se detectó ningún bug.

2.3.2. Hack cases

Se ha probado a crear un pasajero con:

- Realm incorrecto: intentar actualizar como administrator1 y member1.
- Usuario incorrecto: intentar actualizar un pasajero en una booking ajena.
- Actualizar un pasajero ajeno a esa booking.

- Detección de bugs: No se detectó ningún bug.

2.4. Create

2.4.1. Safe cases

Se ha probado a crear un pasajero con:

- Formulario entero vacio
- Nombre de más de 255 caracteres, con 1 carácter y en blanco,
- Email invalido y vacío.
- Pasaporte inválido, vacío y repetido.
- Cumpleaños futuro.
- Necesidades especiales con 0 caracteres y 51 carcateres.

- Detección de bugs: No se detectó ningún bug

2.4.2. Hack cases

Se ha probado a crear un pasajero con:

- Realm incorrecto: intentar crear como administrator1 y member1.
- Usuario incorrecto: intentar crear un pasajero en una booking ajena

- Detección de bugs: No se detectó ningún bug

2.5. Publish

2.5.1. Safe cases

Se ha probado a publicar un pasajero con los siguientes valores inválidos o incorrectos, esperando que el sistema muestre los errores correspondientes:

- Formulario completamente vacío.
- FullName: vacío, con un único carácter, y con más de 255 caracteres.
- Email: con formato inválido (sin @, sin dominio, espacios...), y vacío.
- Passport: con menos de 6 o más de 9 caracteres, caracteres inválidos (símbolos, minúsculas), vacío y duplicado con otro pasajero.
- Birth: con fecha futura o no indicada.
- SpecialNeeds: con más de 50 caracteres

2.5.2. Hack cases

Se ha probado a publicar un pasajero con:

- Realm incorrecto: intentar publicar como administrator1 y member1.
- Usuario incorrecto: intentar publicar un pasajero en una booking ajena

- Detección de bugs: No se detectó ningún bug

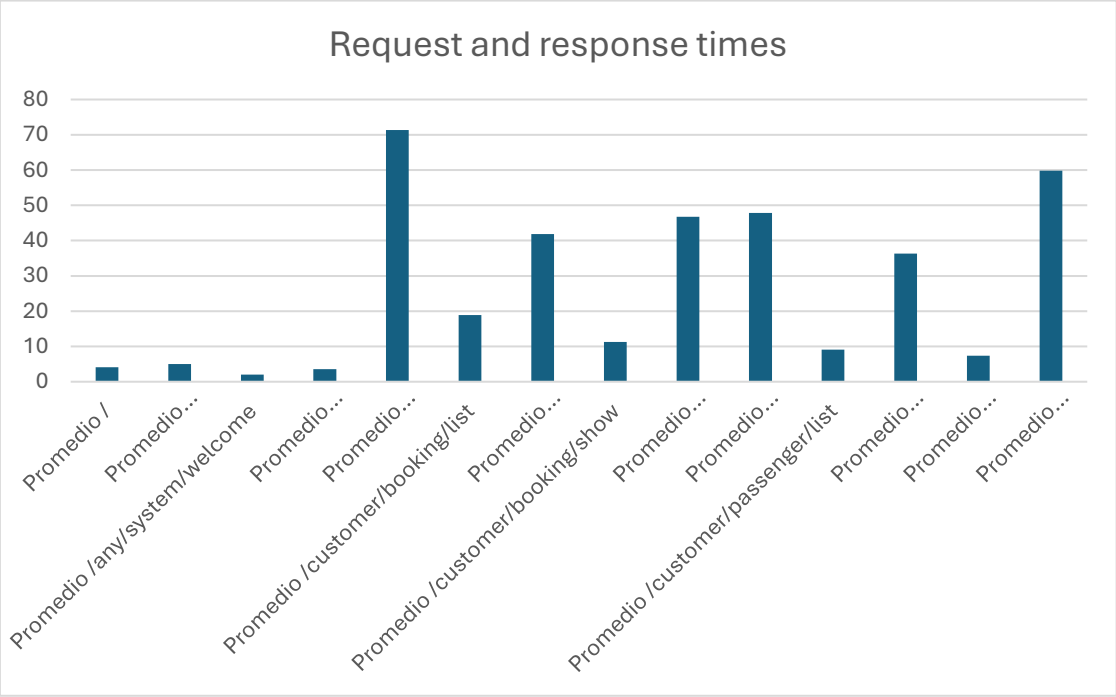
Pruebas performance:

En esta sección se detallan las pruebas de rendimiento realizadas con el objetivo de evaluar el comportamiento del sistema bajo diferentes niveles de carga y condiciones operativas. Estas pruebas permiten identificar posibles cuellos de botella, tiempos de respuesta excesivos o degradación del servicio.

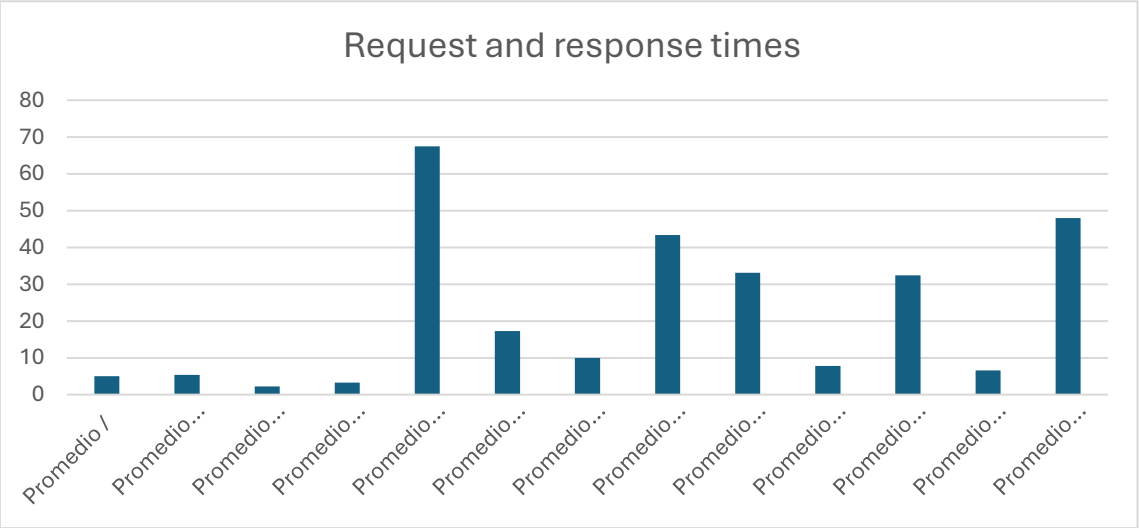
Se ha usado las trazas con y sin índices en las entidades

Lo primero que calculamos fue los “Requests and Response times”, gracias a las herramientas de record y replay que nos proporciona Acme-Framework. Este calculó se realizó promediando los tiempos por cada petición distinta, de tal manera que el gráfico que se obtuvo para cada PC fueron los siguientes:

Sin índices:



Con índices:



A primera vista, los resultados muestran que los tiempos de ambos dispositivos son bastante parecidos. Sin embargo, para obtener una visión más precisa y profunda, vamos a comparar los intervalos de confianza utilizando las herramientas estadísticas que ofrece Excel:

Sin índices:

	A	B	C	D	E	F
1	*Columna1*					
2				Interval(ms)	11,57672015	14,9214833
3	Media	13,24910173		Interval(s)	0,01157672	0,01492148
4	Error típico	0,851732485				
5	Mediana	6,0192				
6	Moda	1,7736				
7	Desviación estándar	22,07942951				
8	Varianza de la muestra	487,5012074				
9	Curtosis	15,95429868				
10	Coeficiente de asimetría	3,538692627				
11	Rango	203,1453				
12	Mínimo	1,0621				
13	Máximo	204,2074				
14	Suma	8903,396364				
15	Cuenta	672				
16	Nivel de confianza(95,0%)	1,672381579				

Utilizando nuevamente el analizador de datos de Excel, se obtiene que la amplitud del intervalo de confianza al 95 % en este segundo entorno es de 3,34 ms. Por tanto, el intervalo de confianza para la media del tiempo de respuesta queda comprendido entre: [11,58 ms – 14,92 ms] o, en segundos: [0,0116 s – 0,0149 s]

Con índices:

Columna1					
			Interval(ms)	10,6836937	13,7649182
Media	12,224306		Interval(s)	0,01068369	0,01376492
Error típico	0,78462326				
Mediana	5,8443				
Moda	3,1008				
Desviación es	20,3397596				
Varianza de la	413,70582				
Curtosis	19,4088054				
Coeficiente de	3,86383066				
Rango	180,3933				
Mínimo	0,8652				
Máximo	181,2585				
Suma	8214,7336				
Cuenta	672				
Nivel de confia	1,54061224				

Utilizando el analizador de datos de Excel, obtenemos que la amplitud del intervalo de confianza al 95 % es de aproximadamente 3,08 ms. Esto nos permite calcular el siguiente intervalo de confianza para el tiempo medio de respuesta: [10,68 ms – 13,76 ms], o lo que es lo mismo, en segundos: [0,0107 s – 0,0138 s].

A simple vista, los tiempos de respuesta en ambos dispositivos son bastante similares. No obstante, el segundo entorno muestra una media.

Continuemos evaluando las pruebas z para comprobar si ha habido cambios significativos en cuanto a rendimiento entre un PC y otro. Como resultado obtenemos lo siguiente:

Prueba z para medias de dos muestras		
	Before	*After*
Media	12,9758748	11,9994845
Varianza (conocida)	487,501207	413,70582
Observaciones	658	658
Diferencia hipotética de las me	0	
z	0,83430358	
P(Z<=z) una cola	0,20205497	
Valor crítico de z (una cola)	1,64485363	
Valor crítico de z (dos colas)	0,40410994	
Valor crítico de z (dos colas)	1,95996398	

Tras realizar una prueba z para comparar las medias de los tiempos de ejecución en ambos entornos (Before y After), obtenemos un valor de **z = 0,8343** y un **p-value (una cola) = 0,20**.

Como este valor p **es mucho mayor que 0,05**, no se puede rechazar la hipótesis nula de que ambos entornos tienen el mismo rendimiento. Es decir, **no existen diferencias estadísticamente significativas** entre los tiempos de ejecución de ambos dispositivos.

Conclusión:

El proceso de verificación y validación ha demostrado que el sistema funciona correctamente bajo una amplia variedad de condiciones. Las pruebas funcionales han cubierto escenarios tanto típicos como excepcionales, garantizando la estabilidad y robustez de las funcionalidades implementadas. La detección y prevención de posibles errores, incluidos ataques mediante manipulación del cliente (DevTools), refuerzan la solidez del sistema frente a usos indebidos.

En cuanto a las pruebas de rendimiento, el análisis de tiempos de ejecución muestra una respuesta estable del sistema en distintos entornos. El cálculo de los intervalos de confianza ha delimitado el comportamiento temporal de forma precisa, y la prueba z ha confirmado que no existen diferencias estadísticamente significativas entre los equipos evaluados. Por tanto, se concluye que el rendimiento del sistema es consistente y no depende del entorno hardware utilizado.