**Testing report**

**Acme Ans C3**

**Student02**

**Group C3.X01**

**Student:**

* Olga Cantalejo Gómez ([olgcangom@alum.us.es](mailto:olgcangom@alum.us.es))

**Repository:** [**https://github.com/olgacangom/Acme-ANS-C3**](https://github.com/olgacangom/Acme-ANS-C3)

**Date:** 15/10/2025

Contenido

[1. Resumen ejecutivo 3](#_Toc211461644)

[2. Tabla de versiones 4](#_Toc211461645)

[3. Introducción 5](#_Toc211461646)

[4. Testing functional 6](#_Toc211461647)

[4.1 Booking 6](#_Toc211461648)

[4.2 Booking Record 7](#_Toc211461649)

[4.3 Passenger 9](#_Toc211461650)

[5. Rendimiento del testing 11](#_Toc211461651)

[5.1 Resultados sin índices 11](#_Toc211461652)

[5.2 Resultados con índices 12](#_Toc211461653)

[5.3 Comparación final 13](#_Toc211461654)

[6. Conclusión 14](#_Toc211461655)

[7. Bibliografía 15](#_Toc211461656)

# Resumen ejecutivo

Este informe presenta los resultados obtenidos tras la ejecución de pruebas funcionales y de rendimiento sobre las operaciones de gestión de reservas y pasajeros destinadas a los usuarios con rol **Customer**. Las funcionalidades evaluadas incluyen la consulta del listado de registros, reservas y pasajeros (list), la visualización detallada de cada entidad (show), la creación de nuevas reservas, pasajeros y registros (create), su modificación (update), la publicación de reservas y pasajeros (publish) y la eliminación de ellos y de registros (delete), todas ellas sujetas a restricciones de autorización y validación contextual.

Las pruebas funcionales se han diseñado con un enfoque exhaustivo, abarcando tanto escenarios válidos como situaciones de error y accesos no autorizados, con el objetivo de verificar la robustez, seguridad y fiabilidad del sistema. Para ello, se han considerado casos positivos, negativos y de tipo hacking, siguiendo una metodología formal de verificación y validación del comportamiento esperado en cada operación.

Además, se han llevado a cabo pruebas de rendimiento, registrando los tiempos de respuesta y analizando los resultados mediante técnicas estadísticas. Se ha calculado un intervalo de confianza del 95% para comparar el comportamiento del sistema en distintos entornos y detectar posibles diferencias significativas en términos de eficiencia.

# Tabla de versiones

|  |  |  |
| --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** |
| 09/10/2025 | 0.0 | Creación del documento plantilla |
| 15/10/2025 | 1.0 | Actualización del documento |

# Introducción

Como integrante del grupo C3.X01, afronto la cuarta y última fase del proyecto **Acme-ANS**, centrada en el proceso de testing. Esta etapa representa un punto crítico en el desarrollo, ya que permite validar la calidad, seguridad y fiabilidad del sistema antes de su posible despliegue en un entorno real. El objetivo principal es verificar de forma exhaustiva las funcionalidades asignadas al rol de comprador (Customer).

El informe se estructura en dos secciones principales: una dedicada al análisis de las pruebas funcionales y otra enfocada en la evaluación del rendimiento. Para garantizar una validación completa, se han diseñado distintos tipos de pruebas que abarcan casos positivos (utilizando valores límite y cercanos al límite), negativos (con entradas no válidas) y simulaciones de ataques (para interceptar intentos de acceso no autorizado mediante peticiones GET, POST, UPDATE, DELETE Y PUBLISH).

# Testing functional

## 4.1 Booking

Las pruebas se realizaron siguiendo la metodología descripta en las dispositivas teóricas, con el objetivo de alcanzar el mayor nivel de cobertura possible.

* **Create:** Se testearon múltiples combinaciones de atributos al crear neuvas reservas, incluyendo valores inválidos, con el fin de comprobar que se generaban los mensajes de error esperados. En el apartado de hacking, se intentó crear registros sin autenticación, reutilizar identificadores ya existentes, entre otros casos. La funcionalidad demostró un comportamiento correcto en todos los intentos, obteniendo una cobertura del 100%.
* **List:** Se verificó la funcionalidad de visualización del listado de reservas, asegurando que el sistema respondiera correctamente ante accesos válidos. En las pruebas de seguridad, se simularon intentos de acceso no autorizado: desde usuarios no autenticados, usuarios incorrectos dentro de un realm válido, y realms inválidos. Todos los casos fueron gestionados correctamente por el sistema, alcanzando una cobertura del 100%.
* **Show:** Esta funcionalidad fue evaluada mediante la consulta de información detallada de una reserva específica. Para validar la robustez del sistema, se realizaron pruebas de acceso indebido, como el uso de credenciales de roles no autorizados y la consulta de reservas inexistentes. El sistema respondió adecuadamente en todos los escenarios, logrando una cobertura del 100%.
* **Update:** Las pruebas de modificación incluyeron variaciones en los atributos, tanto válidas como inválidas, para verificar la gestión de errores. También se evaluaron intentos de actualización sin autenticación, sobre reservas inexistentes y con usuarios de distinto rol. El sistema manejó correctamente cada situación, alcanzando una cobertura del 100%.
* **Publish:** La funcionalidad de publicación fue evaluada en el contexto de reservas que ya contaban con al menos un pasajero asociado, cumpliendo con los requisitos del sistema. Se realizaron pruebas con reservas válidas en modo borrador, así como intentos de publicación sin pasajeros, con locatorCode ya existente, etc.. También se incluyeron pruebas de hacking, como el acceso sin autenticación, el uso de identificadores manipulados y la simulación de roles no autorizados. En todos los casos, el sistema respondió correctamente, mostrando los mensajes de error apropiados o bloqueando el acceso según correspondía. La funcionalidad demostró ser robusta y segura, alcanzando una cobertura del 100%.
* **Delete:** Se testearon múltiples escenarios de eliminación de registros de reserva, incluyendo intentos válidos sobre reservas en modo borrador y casos inválidos como la eliminación de registros ya publicados, inexistentes o pertenecientes a otros clientes. Las pruebas de seguridad incluyeron accesos sin autenticación, manipulación de parámetros y simulación de roles no autorizados. Además, se verificó que el sistema no ejecutara la eliminación al acceder directamente a la URL, sino que esperara la confirmación explícita del usuario. La funcionalidad respondió correctamente en todos los casos, gestionando adecuadamente los errores y restricciones, y alcanzando una cobertura del 100%.

Gráfico, Gráfico de barras

El contenido generado por IA puede ser incorrecto.

## 4.2 Booking Record

Las pruebas se realizaron siguiendo la metodología descripta en las dispositivas teóricas, con el objetivo de alcanzar el mayor nivel de cobertura possible.

* **Create:** Las pruebas de creación se centraron en validar la correcta asociación entre una reserva y un pasajero. Se testearon múltiples combinaciones de atributos, incluyendo casos inválidos como reservas sin modo borrador, pasajeros ajenos al cliente, y duplicación de relaciones ya existentes. También se realizaron pruebas de hacking, como intentos de creación sin autenticación, manipulación de identificadores y uso de roles no autorizados. Aunque la funcionalidad demostró un comportamiento robusto en todos los escenarios relevantes, quedó un pequeño margen sin cubrir, alcanzando una cobertura del 97%.

Esto es debido al authorise()

**Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.**

Aunque el método authorise() presenta una cobertura alta, se ha detectado que la instrucción authorised = false; dentro del bloque catch (Throwable e) no ha sido ejecutada en ningún caso de prueba. Esto se debe a que, durante la ejecución de las pruebas funcionales y de hacking, no se ha producido ninguna excepción en el bloque try, lo que impide alcanzar una cobertura del 100%.

La aplicación no permite simular manualmente una excepción en esta ruta sin alterar el comportamiento interno del sistema, lo que limita la posibilidad de forzar artificialmente la ejecución del bloque catch.

* **List:** La funcionalidad de listado de relaciones entre reservas y pasajeros fue evaluada verificando que el sistema mostrara correctamente todas las asociaciones existentes para el cliente autenticado. Se realizaron pruebas de acceso válido y también intentos de acceso indebido, incluyendo usuarios no autenticados, clientes ajenos y roles no autorizados. En todos los casos, el sistema respondió adecuadamente, mostrando únicamente los datos permitidos o bloqueando el acceso según correspondiera. La funcionalidad alcanzó una cobertura del 100%.
* **Show:** Se testearon consultas detalladas de relaciones específicas entre reservas y pasajeros. Las pruebas incluyeron accesos válidos, intentos de visualización de relaciones inexistentes, y simulaciones de acceso por parte de usuarios no autorizados. El sistema gestionó correctamente cada situación, mostrando los datos esperados o denegando el acceso cuando correspondía. La cobertura obtenida fue del 100%.
* **Delete:** La funcionalidad de eliminación fue evaluada asegurando que solo se permitiera borrar relaciones entre reservas y pasajeros cuando la reserva asociada estuviera en modo borrador. Se testearon eliminaciones válidas, intentos sobre reservas ya publicadas, relaciones inexistentes y accesos indebidos. Además, se verificó que el sistema no ejecutara la eliminación al acceder directamente a la URL, sino que esperara la confirmación explícita del usuario. El sistema respondió correctamente en todos los casos, alcanzando una cobertura del 100%.

Imagen que contiene Gráfico

El contenido generado por IA puede ser incorrecto.

## 4.3 Passenger

Las pruebas se realizaron siguiendo la metodología descripta en las dispositivas teóricas, con el objetivo de alcanzar el mayor nivel de cobertura possible.

* **Create:** Las pruebas de creación se centraron en validar la correcta incorporación de nuevos pasajeros al sistema. Se testearon múltiples combinaciones de atributos, incluyendo valores inválidos, para comprobar que se generaban los mensajes de error esperados. También se realizaron pruebas de hacking, como intentos de creación sin autenticación, manipulación de identificadores y uso de roles no autorizados. El sistema demostró un comportamiento robusto, gestionando correctamente cada situación y alcanzando una cobertura del 100%.
* **List:** La funcionalidad de listado de pasajeros fue evaluada verificando que el sistema mostrara correctamente todos los registros asociados al cliente autenticado. Se realizaron pruebas de acceso válido y también intentos de acceso indebido, incluyendo usuarios no autenticados, clientes ajenos y roles no autorizados. En todos los casos, el sistema respondió adecuadamente, mostrando únicamente los datos permitidos o bloqueando el acceso según correspondiera. La funcionalidad alcanzó una cobertura del 100%.
* **Show:** Se testearon consultas detalladas de información de pasajeros específicos. Las pruebas incluyeron accesos válidos, intentos de visualización de registros inexistentes, y simulaciones de acceso por parte de usuarios no autorizados. El sistema gestionó correctamente cada situación, mostrando los datos esperados o denegando el acceso cuando correspondía. La cobertura obtenida fue del 100%.
* **Update:** Las pruebas de modificación incluyeron variaciones en los atributos del pasajero, tanto válidas como inválidas, para verificar la gestión de errores. También se evaluaron intentos de actualización sin autenticación, sobre registros inexistentes y con usuarios de distinto rol. El sistema manejó correctamente cada situación, mostrando los mensajes de validación correspondientes y bloqueando accesos indebidos. La cobertura obtenida fue del 99,3%.

Esto es debido al authorise()

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Se ha verificado la lógica principal del método y se ha ejecutado en la mayoría de los casos, aunque podría existir algún escenario puntual que no haya sido cubierto al 100% durante las pruebas.

* **-Delete:** La funcionalidad de eliminación fue evaluada asegurando que solo se permitiera borrar pasajeros en modo borrador y que no estuvieran asociados a reservas publicadas. Se testearon eliminaciones válidas, intentos sobre reservas ya publicadas, registros inexistentes y accesos indebidos. Además, se verificó que el sistema no ejecutara la eliminación al acceder directamente a la URL, sino que esperara la confirmación explícita del usuario. El sistema respondió correctamente en todos los casos, alcanzando una cobertura del 100%.
* **Publish:** La funcionalidad de publicación fue evaluada en el contexto de pasajeros que ya habían sido creados por el cliente autenticado. Se realizaron pruebas con registros válidos en modo borrador, así como intentos de publicación de pasajeros ya publicados o pertenecientes a otros clientes. También se incluyeron pruebas de hacking, como el acceso sin autenticación y el uso de roles no autorizados. En todos los casos, el sistema respondió correctamente, mostrando los mensajes de error apropiados o bloqueando el acceso según correspondía. La funcionalidad alcanzó una cobertura del 100%.

Gráfico, Gráfico de barras

El contenido generado por IA puede ser incorrecto.

# Rendimiento del testing

## 5.1 Resultados sin índices

Tabla

El contenido generado por IA puede ser incorrecto.

## 5.2 Resultados con índices

Se ha añadido a la entidad Booking el siguiente índice:

@Table(indexes = {

@Index(columnList = "draftMode"), @Index(columnList = "draftmode,customer\_id")

})

Se ha añadido a la entidad Booking Record el siguiente índice:

@Table(indexes = {

@Index(columnList = "booking\_id, passenger\_id")

})

Tabla

El contenido generado por IA puede ser incorrecto.

Comentario: No he podido generar las gráficas, ya que excel me generaba error.

## 5.3 Comparación final

Tabla

El contenido generado por IA puede ser incorrecto.

Tabla

El contenido generado por IA puede ser incorrecto.

# Conclusión

La fase de pruebas constituye un pilar esencial en el desarrollo de sistemas informáticos, ya que permite verificar que la aplicación cumple con los requisitos funcionales y de rendimiento esperados. En este estudio comparativo, se han analizado los tiempos de respuesta con y sin índices, aplicando métricas estadísticas y una prueba Z para determinar la significancia de las diferencias observadas.

El valor obtenido fue z = 0.0246, con un valor p (dos colas) = 0.9803, claramente superior al nivel de significación α = 0.05. Dado que |z| < 1.9599 y p > 0.05, no se rechaza la hipótesis nula.

Por tanto, no existen diferencias estadísticamente significativas entre los tiempos antes y después de los cambios, lo que indica que las modificaciones no produjeron una mejora apreciable en el rendimiento del sistema.

Estas conclusiones refuerzan la importancia de realizar pruebas estructuradas y cuantificables, no solo para validar la funcionalidad del sistema, sino también para identificar variaciones de rendimiento que puedan afectar la experiencia del usuario. Además, este tipo de análisis permite tomar decisiones informadas sobre la infraestructura tecnológica más adecuada para garantizar eficiencia, estabilidad y escalabilidad en distintos escenarios de uso.

# Bibliografía

Intencionalmente en blanco.