

**Student#2 Testing Report**

C1.046

<https://github.com/DP2-C1-046/Acme-ANS-D04>

<https://github.com/orgs/DP2-C1-046/projects/1/views/1?filterQuery=rafcarmed>

|  |  |
| --- | --- |
| Apellidos, Nombre | Email |
| Caro Medina, Rafael David | rafcarmed@alum.us.es |

**Tabla de contenidos**

[Introducción 3](#_Toc199164132)

[1. Pruebas funcionales o formales (Formal testing) 4](#_Toc199164133)

[1.1 Bookings ( Reservas ) 4](#_Toc199164134)

[1.1.1 Feature customer/booking/list 4](#_Toc199164135)

[1.1.2 Feature customer/booking /show 4](#_Toc199164136)

[1.1.3 Feature customer/booking/create 4](#_Toc199164137)

[1.1.4 Feature customer/booking/update 5](#_Toc199164138)

[1.1.5 Feature customer/booking/publish 5](#_Toc199164139)

[1.1.6 Feature customer/booking/delete 6](#_Toc199164140)

[1.2 Passengers (Pasajeros) 6](#_Toc199164141)

[1.2.1 Feature customer/passenger/list 6](#_Toc199164142)

[1.2.2 Feature customer/passenger/show 6](#_Toc199164143)

[1.2.3 Feature customer/passenger/create 7](#_Toc199164144)

[1.2.4 Feature customer/passenger/update 7](#_Toc199164145)

[1.2.5 Feature customer/passenger/publish 7](#_Toc199164146)

[1.3 BookingRecords 7](#_Toc199164147)

[1.3.1 Feature customer/booking-record/create 8](#_Toc199164148)

[2. Pruebas de rendimiento (Performance testing) 8](#_Toc199164149)

[2.1 Método aplicado al cálculo de rendimiento 8](#_Toc199164150)

[2.2 Resultados obtenidos 8](#_Toc199164151)

[2.2.1 Cobertura de líneas de código y ramas. Analyser 9](#_Toc199164152)

[2.2.2 Gráfica de tiempos de respuesta sin índices 11](#_Toc199164153)

[2.2.3 Creación de índices 12](#_Toc199164154)

[2.2.4 Gráfica de tiempos de respuesta con índices 12](#_Toc199164155)

[2.2.3 Comparativa entre ejecuciones de la Test Suite : Z-Test 13](#_Toc199164156)

[2.3 Contraste de hipótesis 15](#_Toc199164157)

# Introducción

El presente documento tiene como objetivo describir y documentar la creación de una Test Suite para la aplicación Acme-ANS con el fin de realizar las pruebas formales (Formal Testing) y de Rendimiento (Performance Testing) necesarias para garantizar una calidad razonable de la aplicación a entregar al cliente en el cuarto y último entregable de la asignatura.

# Pruebas funcionales o formales (Formal testing)

En primer lugar, comentaré que las entidades relativas al Studen#2 que se han diseñado e implementado, y con las que se ha trabajado durante el desarrollo del proyecto han sido Bookings (Reservas de vuelos), Passengers (Pasajeros) y BookingsRecords (la entidad intermedia que implementa el objeto conglomerado que conforma una reserva completa en la vida real y asocia pasajeros a las reservas).

Los Casos de Prueba han sido generados para el principal Customer1 al ser el que tiene mayor número de datos de ejemplo (sample data).

Se definen los casos de prueba generados como:

Casos de prueba **Positivos**: comprueban o validan que peticiones legales con datos o entradas válidas resultan en éxito

Casos de prueba **Negativos**: comprueban o validan que peticiones legales con datos o entradas inválidas resultan en error (error controlado por la aplicación que informa al usuario)

Casos de prueba de **Hacking**: comprueban o validan que peticiones ilegales no son autorizadas. Una petición ileal es aquella que no se genera desde la interfaz de usuario de la aplicación a través de un botón, un enlace o un submit, sino pegando una URL en el navegador o manipulando los datos a través de las Herramientas del Desarrollador del navegador.

## 1.1 Bookings (Reservas)

Se detallarán los Casos de Prueba generados y ejecutados para la entidad Bookings de Student#2.

### 1.1.1 Feature customer/booking/list

* Casos positivos:
  + Visualización del listado de Reservas correspondientes al Customer logado (opción del menú principal de Customer->List my bookings).
* Casos negativos:
* Casos de hacking:
  + Varios intentos de visualización del listado de Reservas desde un rol distinto a Customer, por ejemplo, Manager o Anonymous. Se debe desautorizar el acceso a la visualización. En el caso de “Right Realm/Wrong User”, otro Customer, se muestra el listado de Reservas correspondientes a ese usuario.

### 1.1.2 Feature customer/booking /show

* Caso positivo:
  + Acceso al formulario de detalles de una Reserva del principal propietario.
* Caso negativo:
* Casos de hacking:
  + Varios intentos de visualización del formulario de Reservas de Customer1 desde distintos Realms ( Manager, Anonymous ), mismo Real pero distinto User, por ejemplo, Customer2 o Customer3.

### 1.1.3 Feature customer/booking/create

* Casos positivos:
  + Creación de un Reserva con la comprobación previa de todas las restricciones y todos los límites de rangos en los distintos atributos, según la metodología recomendada, y finalmente la creación con valores correctos.
* Casos negativos:
  + Intento de creación de una Reserva con todos los campos del formulario vacíos.
  + Iteración por cada uno de los campos del formulario o atributos de Reservas, dejando el resto de campos vacíos, para comprobar todos los límites y rangos del atributo en cuestión, tantos valores inválidos como sea posible, con los valores recomendados en Sample-Data.xls
* Casos de hacking:
  + Intento de creación de una Reserva desde un Realm incorrecto ( Manager, Anonymous , ..,. ) o desde un realm Customer pero con otro usuario distinto al propietario de la reserva.

### 1.1.4 Feature customer/booking/update

La feature Update permite actualizar cualquier dato referente a una Reserva, siempre y cuando ésta no se haya publicado previamente (se encuentre en modo Borrador). En caso de poder actualizarse se podrán modificar tanto sus datos como los pasajeros que están asociados a ella.

Los casos de prueba implementados han sido:

* Caso positivo:
  + Actualizar cada campo del formulario de una reserva que se encuentra en modo Borrador, con valores válidos para ese campo, desde el Customer propietario de la reserva.
* Casos negativos:
  + Intento de actualización de una Reserva con todos los campos del formulario vacíos.
  + Actualizar cada campo del formulario de una reserva que se encuentra en modo Borrador, con valores inválidos para ese campo, desde el Customer propietario de la reserva.
* Casos de hacking:
  + Intento de actualizar una reserva por un realm distinto de Customer, o un Customer distinto de Customer1, propietario de la reserva. Como acción no autorizada, intento de actualización, por parte del propietario, de una reserva que ya se encuentra publicada, o modificación, a través de las Herramientas del Desarrollador, de los valores de los desplegables de Flight, que seleccionan el vuelo asociado a la reserva, o del TravelClass que especifica la clase de vuelvo seleccionada en el desplegable.

### 1.1.5 Feature customer/booking/publish

La feature Publish permite publicar una reserva, que se encuentra actualmente en estado Borrador, una vez ésta cumple con todos los criterios de restricciones y validaciones especificadas en los requisitos. Una vez publicada una reserva no se podrá modificar.

Los casos de prueba implementados han sido:

* Caso positivo:
  + Publicar una reserva, por parte de su propietario, que se encuentra en estado borrador y con todos sus campos válidos.
* Casos negativos:
  + Publicar una Reserva con todos los campos del formulario vacíos.
  + Publicar una reserva, por parte de su propietario, que se encuentra en estado borrador y con alguno de sus campos inválidos.
  + Publicar una reserva, por parte de su propietario, que se encuentra en estado borrador y sin los 4 últimos dígitos de la tarjeta de crédito introducidos.
  + Publicar una reserva, por parte de su propietario, que se encuentra en estado borrador y sin ningún pasajero asociado.
  + Publicar una reserva, por parte de su propietario, que se encuentra en estado borrador y con alguno de sus pasajeros asociados en estado Borrador.
* Casos de hacking:
  + Intento de publicar una reserva por un realm distinto de Customer, o un Customer distinto de Customer1, propietario de la reserva. Como acción no autorizada, intento de publicación, por parte del propietario, de una reserva que ya se encuentra publicada, o modificación, a través de las Herramientas del Desarrollador, de los valores de los desplegables de Flight, que seleccionan el vuelo asociado a la reserva, o del TravelClass que especifica la clase de vuelvo seleccionada en el desplegable.

### 1.1.6 Feature customer/booking/delete

Como en los requisitos de Student#2 no se especifica ninguna feature para eliminar una reserva, un pasajero o deasociar a un pasajero de una reserva no he desarrollado ninguna feature que implemente estas funcionalidades.

## 1.2 Passengers (Pasajeros)

A continuación, se relacionan los Casos de Prueba generados y ejecutados para la entidad Passenger de Student#2.

### 1.2.1 Feature customer/passenger/list

* Casos positivos:
  + Visualización del listado de Pasajeros correspondientes al Customer logado (opción del menú principal de Customer->List passengers in my bookings o Customer->List all my passengers).
* Casos negativos:
* Casos de hacking:
* Varios intentos de visualización del listado de Pasajeros desde un rol distinto a Customer, por ejemplo, Manager o Anonymous. Se debe desautorizar el acceso a la visualización. En el caso de “Right Realm/Wrong User”, otro Customer, se muestra el listado de Pasajeros correspondientes a ese usuario.

### 1.2.2 Feature customer/passenger/show

* Caso positivo:
  + Acceso al formulario de detalles de un Pasajero por parte del principal propietario, desde la lista de pasajeros del menú principal.
* Caso negativo:
* Casos de hacking:
  + Varios intentos de visualización del formulario de un Pasajero de Customer1 desde distintos Realms ( Manager, Anonymous ), mismo Real pero distinto User, por ejemplo, Customer2 o Customer3.

### 1.2.3 Feature customer/passenger/create

* Casos positivos:
  + Creación de un Pasajero con la comprobación previa de todas las restricciones y todos los límites de rangos en los distintos atributos, según la metodología recomendada, y finalmente la creación con valores correctos.
* Casos negativos:
  + Intento de creación de un Pasajero con todos los campos del formulario vacíos.
  + Iteración por cada uno de los campos del formulario o atributos de Pasajeros, dejando el resto de campos vacíos, para comprobar todos los límites y rangos del atributo en cuestión, tantos valores inválidos como sea posible, con los valores recomendados en Sample-Data.xls
* Casos de hacking:
  + Intento de creación de un Pasajero desde un Realm incorrecto ( Manager, Anonymous , ...) o desde un realm Customer pero con otro usuario distinto al propietario del pasajero.

### 1.2.4 Feature customer/passenger/update

La feature Update de pasajeros permite actualizar cualquier dato referente a un Pasajero, siempre y cuando éste no se haya publicado previamente (se encuentre en modo Borrador).

Los casos de prueba implementados han sido:

* Caso positivo:
  + Actualizar cada campo del formulario de un pasajero que se encuentra en modo Borrador, con valores válidos para ese campo, desde el Customer propietario de la reserva.
* Casos negativos:
  + Intento de actualización de un Pasajero con todos los campos del formulario vacíos.
  + Actualizar cada campo del formulario de un pasajero que se encuentra en modo Borrador, con valores inválidos para ese campo, desde el Customer propietario del pasajero.
* Casos de hacking:
  + Intento de actualizar un pasajero por parte de un realm distinto de Customer, o un Customer distinto de Customer1, propietario del pasjero. Como acción no autorizada, intento de actualización, por parte del propietario, de un pasajero que ya se encuentra publicada.

### 1.2.5 Feature customer/passenger/publish

La feature Publish permite publicar un pasajero, que se encuentra actualmente en estado Borrador, una vez éste cumple con todos los criterios de restricciones y validaciones especificadas en los requisitos. Una vez publicado un pasajero no se podrá modificar.

Los casos de prueba implementados han sido:

* Caso positivo:
  + Publicar un pasajero, por parte de su propietario, que se encuentra en estado borrador y con todos sus campos válidos.
* Casos negativos:
  + Publicar un pasajero con todos los campos del formulario vacíos.
  + Publicar un pasajero, por parte de su propietario, que se encuentra en estado borrador y con alguno de sus campos inválidos.
* Casos de hacking:
  + Intento de publicar un pasajero por parte de un realm distinto de Customer. O un Customer distinto de Customer1, propietario del pasajero. Como acción no autorizada, intento de publicación, por parte del propietario, de un pasajero que ya se encuentra publicado.

## 1.3 BookingRecords

La entidad BookingRecord representa la entidad intermedia que permite implementar la asociación de Reservas y Pasajeros, pudiendo asociar o enlazar pasajeros a reservas a través de ella. A continuación, se relacionan los Casos de Prueba generados y ejecutados para la entidad BookingRecords de Student#2.

### 1.3.1 Feature customer/booking-record/list

* Casos positivos:
  + Visualización del listado de pasajeros asociados a la reserva desde la que se pulsa el botón “Passengers” correspondiente al Customer logado.
* Casos negativos:
* Casos de hacking:
  + Varios intentos de visualización del listado de pasajeros de una reserva desde un rol distinto a Customer, por ejemplo, Manager o Anonymous. Se debe desautorizar el acceso a la visualización. En el caso de “Right Realm/Wrong User”, otro Customer, también se desautoriza.

### 1.3.2 Feature customer/booking-record/show

* Caso positivo:
  + Acceso al formulario de detalles de un Pasajero por parte del principal propietario, desde la lista de pasajeros asociados una reserva perteneciente a su vez al principal logado.
* Caso negativo:
* Casos de hacking:
  + Varios intentos de visualización del formulario de un Pasajero asociado a una reserva de Customer1 desde distintos Realms ( Manager, Anonymous ), mismo Real pero distinto User, por ejemplo, Customer2 o Customer3.

### 1.3.3 Feature customer/booking-record/create

* Casos positivos:
  + Asociación o enlazado de un Pasajero a la reserva desde la que se llama a la lista de pasajeros asociados.
* Casos negativos:
* Casos de hacking:
  + Intento de asociación o enlazado de un Pasajero desde un Realm incorrecto ( Manager, Anonymous , ...) o desde un realm Customer pero con otro usuario distinto al propietario de la reserva a la que se quiere asociar al pasajero, o modificación del valor del pasajero en la lista de selección para tratar de asociar a la reserva un pasajero que no pertenece al Customer1 logado.

# Pruebas de rendimiento (Performance testing)

## 2.1 Método aplicado al cálculo de rendimiento

* Se han generado los Casos de Prueba para las features utilizando el launcher testing-tester#record para grabar las trazas de ejecución y posteriormente testing-tester#replay para reproducir esos casos de pruebas y obtener el archivo tester.trace con la información de las pruebas reproducidas y las métricas de tiempo.
* Se han extraído las métricas correspondientes a la primera ejecución de esas pruebas.
* Se ha calculado el intervalo de confianza al 95% sobre esa muestra de tiempos obtenida.
* Como alternativa de optimización del rendimiento se han añadido índices a las entidades Booking y Passenger para intentar mejorar los tiempos de consulta a la base de datos, y por tanto los tiempos de respuesta de las features de la aplicación.
* Se ha vuelto a ejecutar el testing-tester#replay para obtener los nuevos tiempos de ejecución, una vez implementados los índices en las tablas de la base de datos.
* Y finalmente se ha vuelto a calcular de nuevo el intervalo de confianza al 95% para el tiempo total, y el indicador P-Value a través del Z-Test con la nueva muestra de tiempos obtenidos tras las mejoras de optimización, para determinar si dichas mejoras han sido significativas o no.

## 2.2 Resultados obtenidos

### 2.2.1 Cobertura de líneas de código y ramas. Analyser

La cobertura de líneas y ramas de código conseguida, tras la ejecución de la Test Suite generada por Student#2 para sus features, ha sido la siguiente.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

En varias clases con líneas de código con cobertura en color amarillo, sin haber pasado por todas las ramas de esa línea, es debido bien a que se trata de operadores ternarios A ? B : C para comprobar que un objeto no es nulo y poder acceder a alguno de sus atributos, o bien se trata de líneas con operadores lógicos && en la que no se ha entrado por alguna de las alternativas lógicas al no ser nunca un objeto nulo. Se muestra un ejemplo de cada uno de estos casos en la siguiente captura de CustomerPassengerShowService.

En la siguiente captura, en el momento de la toma de la captura no se había pasado, en la línea 46 por ejemplo, por la alternativa “passenger == null” , y por eso aparece en amarillo, o “customer != null” en la línea 48.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Captura del Analyser tras una de las primeras ejecuciones de la Test Suite completa

Interfaz de usuario gráfica, Aplicación, Tabla

El contenido generado por IA puede ser incorrecto.

### 2.2.2 Gráfica de tiempos de respuesta sin índices

Captura de pantalla de la gráfica obtenida sobre el tiempo medio de las features antes de la implementación de los índices, con las celdas de promedios de tiempos por features obtenidas:

Gráfico, Gráfico en cascada

El contenido generado por IA puede ser incorrecto.

Captura de pantalla sólo de la gráfica, sin las celdas promedio de Excel, para su mejor visualización. Se observa que la “Solicitud Más Ineficiente” (MIR) es customer/booking/create por lo que se revisa su código para tratar de encontrar un posible motivo.

Gráfico, Gráfico de barras, Gráfico en cascada

El contenido generado por IA puede ser incorrecto.

### 2.2.3 Creación de índices

Captura de índices creados para las entidades Booking y Passenger, tras consultar sus repositorios y analizar los atributos empleados en las cláusulas Where para las distintas consultas de datos @Query(“”) en sus respectivas tablas.

Dos índices creados para Booking:

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Y otros dos índices para Passenger, uno de ellos compuesto:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

### 2.2.4 Gráfica de tiempos de respuesta con índices

Aunque se observa que la “Solicitud Más Ineficiente” (MIR) sigue siendo customer/booking/create se aprecia que el promedio general obtenido ha bajado desde los 17,62 ms a 13,34 ms, probablemente debido al uso de los índices en las consultas a base de datos de Bookings y Passenger que los utilizan.

Gráfico, Gráfico en cascada

El contenido generado por IA puede ser incorrecto.

### 2.2.3 Comparativa entre ejecuciones de la Test Suite : Z-Test

Las Estadísticas Descriptivas obtenidas para cada ejecución de la Test Suite, antes y después de la optimización obtenida con la incorporación de los índices a las tablas, junto con el cálculo de sus Intervalos de Confianza en milisegundos y segundos, ha sido la siguiente:

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

El contenido generado por IA puede ser incorrecto.

La diferencia del número de peticiones, o features, realizadas en cada muestra, 887 para la primera y 1253 para la segunda, es debido a que, tras el último Follow-Up del viernes 23 de mayo, pude comprobar con el profesor Rafael Corchuelo, a través de la consulta al Analyser, que para muchas features grabadas en los Casos de Prueba no había aplicado todos los límites de rango de valores recomendados para varios atributos de Booking y Passenger, tanto en Create, Update y Publish, y por tanto volví a crear las pruebas para esas features aplicando todos los límites de rango y valores propuestos en el documento Sample-Data.xls

El cálculo del P-Value, a través de la Prueba Z-Test, para ambas muestras ha sido el siguiente:

Captura con los comentarios realizados en el propio libro de Excel referentes a la posible diferencia de descripciones de variables debido a la versión de Excel.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Como indico a la derecha de la variable en el propio libro, en la versión de Excel de la que dispongo, la variable con la descripción “dos colas” a la que hace referencia la diapositiva de Teoría se llama en este caso “Valor crítico de z(dos colas)”, y creo que es el valor a referir del que habla la Teoría, ya que la variable que se llama en cambio parecido “P(Z<=z) una cola)” es sólo para una cola, y en el nombre de la variable en inglés de teoría sí se especifica “two tails”.

Captura de la diapositiva 75 del tema “S02-Performance Testing” donde se indica qué variable de la prueba Z-test se debe consultar.

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

El contenido generado por IA puede ser incorrecto.

Captura de pantalla del cuadro de cálculo de P-Value (o ”Valor crítico de z(dos colas)” en mi versión de Excel) sin comentarios para mejor visualización. En caso de que en mi versión el valor fuera ”P(Z<=z) una cola”, pero sería el valor a “una cola” no como índica la diapositiva que debería ser “dos colas”, ese valor es incluso menor que “Valor crítico de z (dos colas)”, con lo cual seguiría siendo válida la conclusión final.

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

El contenido generado por IA puede ser incorrecto.

## 2.3 Contraste de hipótesis

Una vez realizada la prueba “Z-test para media de dos muestras” con los valores medidos para las dos muestras de ejecución de la Test Suite, antes y después de la optimización con índices, podemos pasar a analizar y valorar los resultados obtenidos y en concreto el denominado P-Value sobre el Intervalo de Confianza establecido a un nivel de confianza del 95%, lo que implica un valor de α=0,05, y cuyo valor (de P-Value) nos indicará si podemos comparar ambas medias.

**Hipótesis de contraste:**

* Hipótesis nula (H₀): μ₁ = μ₂

Los tiempos medios de ejecución de la Test Suite, antes y después de la creación de los índices en las entidades Booking y Passenger, son iguales.

* Hipótesis alternativa (H₁): μ₁ > μ₂

El tiempo medio de ejecución de la Test Suite, antes de la creación de los índices en las entidades de Student#2 Booking y Passenger, es mayor que el tiempo obtenido posteriormente con la creación de los índices (lo que supone una optimización del rendimiento de la aplicación).

Comprobados los resultados de la prueba Z-Test , si el valor a analizar según la diapositiva 75 es el “Valor crítico de z (dos colas)” cuyo valor es 0,000034 remarcado antes en verde (equivalente al P-Value del que se habla en la diapositiva 76 aunque con ese otro nombre en mi versión de Excel), al estar en el intervalo [0,00, α=0,05) “*entonces podemos comparar las medias de nuestros tiempos: si la media de después de los cambios es más pequeña, entonces has conseguido mejorar el rendimiento: de otra forma, no*”

Texto

El contenido generado por IA puede ser incorrecto.

**Conclusión final del Z-Test:**

Al estar nuestro valor “Valor crítico de z (dos colas)” = 0,000034 dentro de ese intervalo [0,00, α=0,05) podemos comparar las medias, y como en nuestro caso la media de **después** de los cambios introducidos a través de la creación de índices **es más pequeña** que la media de **antes** de los cambios, entonces podemos rechazar la hipótesis nula y concluir, a un 95% de confianza, que hemos mejorado el rendimiento tras la creación de índices.

Media obtenida para la muestra **antes** de la creación de índices: 17,62 ms

Media obtenida para la muestra **después** de la creación de índices: 13,34 ms