

TESTING REPORT



REPOSITORIO: <https://github.com/marrivbec/gii-is-DP2-C1.033.git>

GRUPO: C1.033

Jesús Villalba Fernández (jesvilfer@alum.us.es)

Domingo 25 de Mayo de 2025

Tabla de contenido

RESUMEN EJECUTIVO 3

TABLA DE REVISIÓN 3

INTRODUCCIÓN 3

CAPÍTULO SOBRE PRUEBAS FUNCIONALES 4

BOOKINGS..... 4

BOOKING RECORD 10

PASSENGERS 14

CAPÍTULO SOBRE PRUEBAS DE RENDIMIENTO 20

GRÁFICAS DE RENDIMIENTO 20

SIN ÍNDICES 20

CON ÍNDICES..... 21

DESDE OTRO ORDENADOR 21

INTERVALO DE CONFIANZA DEL 95%..... 22

SIN ÍNDICES 22

CON ÍNDICES..... 22

DESDE OTRO ORDENADOR 23

COMPARATIVA Z-TEST 23

SIN ÍNDICES VS CON ÍNDICES..... 23

MI ORDENADOR (CON ÍNDICES) VS OTRO ORDENADOR (CON ÍNDICES) 24

CONCLUSIÓN 24

BIBLIOGRAFÍA 24

Resumen ejecutivo

Este informe presenta las pruebas funcionales y de rendimiento realizadas sobre una aplicación de gestión de reservas. Se probaron los procesos de listado, creación, actualización, publicación y eliminación de reservas, registros y pasajeros, incluyendo tanto casos normales como pruebas de seguridad para detectar accesos no autorizados. Además, se evaluó el rendimiento del sistema con y sin índices, y en diferentes equipos. Los resultados confirman la correcta funcionalidad y robustez del sistema, identificando áreas de mayor carga para optimización antes de su despliegue.

Tabla de revisión

Versión	Fecha	Descripción
1.0.0	25/5/2025	Versión con el testing funcional y sin las gráficas de rendimiento.
1.1.0	26/5/2025	Versión con el testing funcional y con las gráficas de rendimiento.

Introducción

El objetivo de este documento es detallar las pruebas realizadas para validar la funcionalidad, seguridad y rendimiento de la aplicación de reservas. Se incluyen pruebas positivas, negativas y de hacking para garantizar que el sistema maneja adecuadamente diferentes escenarios y protege los datos frente a accesos indebidos. También se analizan los tiempos de respuesta bajo distintas condiciones. El informe está organizado para mostrar la metodología, los resultados de las pruebas y las conclusiones obtenidas.

Capítulo sobre pruebas funcionales

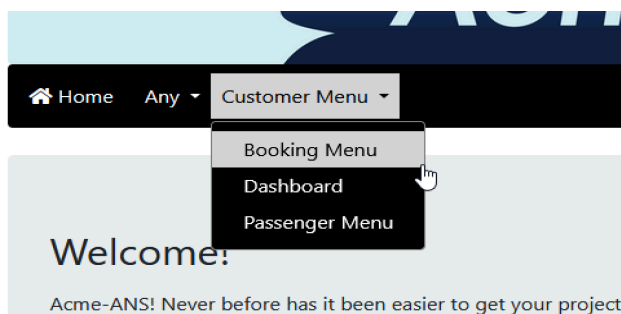
Para cada prueba, antes de todo me he logueado como customer, cuyo usuario es “customer1” y contraseña “customer1”. Estos casos de prueba se han llevado a cabo siguiendo el documento “Sample-Data.xlsx” y se han realizado tanto casos de pruebas positivos y negativos, como de hacking. En customer podemos observar que tenemos acceso al menú de Bookings y Passengers”, y a la entidad intermedia Booking Record que podemos acceder mediante una booking . A continuación explicaremos paso a paso cómo se han llevado sus respectivas pruebas.

Primero de todo, tenemos que diferenciar entre el testing.safe y el testing.hack. En el testing.safe lo que se ha llevado a cabo los casos de prueba positivos y negativos, y en el testing.hack se han llevado pruebas de hacking.

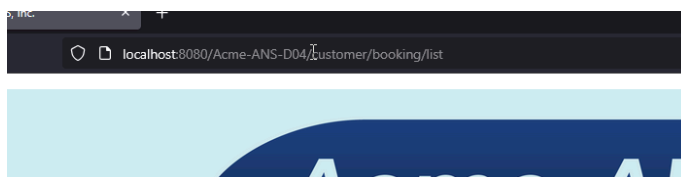
Bookings

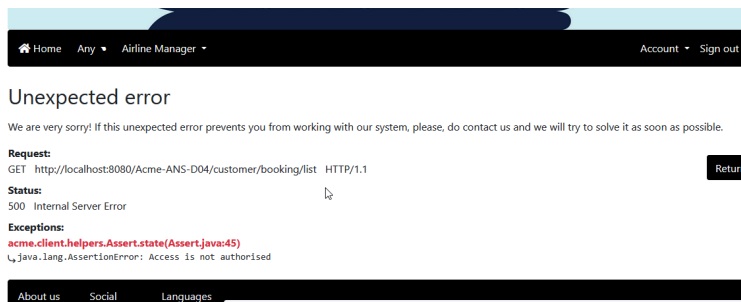
Listar

Para llevar a cabo el **list.safe** lo que hemos hecho es listar las bookings, accediendo a este listado mediante el botón Booking Menu.



Para el **list.hack** me he metido con otro usuario (ej:manager1) y he intentado acceder a la URL del listado del booking.





Mostrar

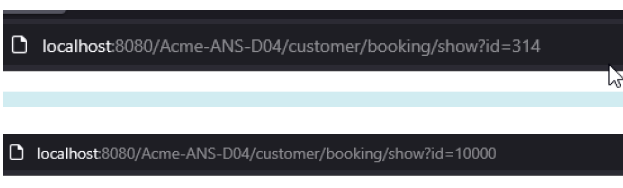
Para el **show.safe** lo que hemos hecho es listar las bookings, accediendo a este listado mediante el botón Booking Menu, y pinchar en una de las bookings de esa lista.

Locator Code	Purchase Moment	Travel Class	Price	Last Nibble	DraftMode
123456	2012/01/01 00:00	ECONOMY	EUR 500,000.00	1234	true
12345678	2012/01/01 00:00	ECONOMY	EUR 500,000.00	1234	true
1234ABCD	2012/01/01 00:00	ECONOMY	EUR 500,000.00	1234	true
123ABC	2012/01/01 00:00	ECONOMY	EUR 500,000.00	1234	true
AAAA123	2012/01/02 00:00	ECONOMY	EUR 500,000.00	1234	true

Para el **show.hack** he probado a cambiar la URL de la booking para que en vez de aparecerme esa booking me apareciera una que no está creada y otra booking de otro customer. En ambos casos me ha salido error 500.

```
@Override
public void authorise() {
    boolean status;
    int masterId;
    int customerId;
    Booking booking;
    Customer customer;
    Customer customerLogged;
    customerId = super.getRequest().getPrincipal().getActiveRealm().getId();
    customerLogged = this.repository.findCustomerLogged(customerId);
    masterId = super.getRequest().getData("id", int.class);
    booking = this.repository.findBookingById(masterId);
    customer = booking == null ? null : booking.getCustomer();
    status = booking != null && booking.getCustomer().equals(customerLogged);

    super.getResponse().setAuthorised(status);
}
```



Unexpected error

We are very sorry! If this unexpected error prevents you from working with our system, please, do contact us and we will try to solve it as soon as possible.

Request:

GET http://localhost:8080/Acme-ANS-D04/customer/booking/show?id=314 HTTP/1.1

Status:

500 Internal Server Error

Exceptions:

acme.client.helpers.Assert.state(Assert.java:45)

↳ java.lang.AssertionError: Access is not authorised

Retu

Crear

Para el **create.safe** hemos entrado en el formulario de crear una booking mediante el botón de create que aparece en la lista de bookings. Primero de todo, hacíamos una prueba dejando el formulario en blanco, todos los atributos sin valor.

Booking Details

Locator Code

↳ Lorem ipsum

Invalid locator code. It must be between 6 and 8 characters long, using only uppercase letters and numbers. May not be null.

Purchase Moment

↳ yyyy/mm/dd hh:mm

May not be null.

Travel Class

↳ ----

May not be null.

Flight

↳ ----

May not be null.

Last Nibble

↳ Lorem ipsum

The last nibble must be exactly 4 digits.

☐ Confirm

Must confirm this.

Create

Después de esto, se prueba cada atributo individualmente con los valores positivos y negativos del Excel “Sample-Data”, dándole a “create” tras probar cada atributo y esperando los correspondientes mensajes. Por último, se crea una booking con todos los valores positivos.

Booking Details

Locator Code

Purchase Moment

May not be null.

Travel Class

May not be null.

Flight

May not be null.

Last Nibble

The last nibble must be exactly 4 digits.

☐ **Confirm**

Must confirm this.

Para el **create.hack** con la herramienta para desarrolladores de Firefox se prueba el enumerado “Flight” cambiando la propiedad “value” por un valor de un flight que no existiera y un flight que no estuviera publicado, esperando un error 500. Aquí las demás propiedades las tengo que tener con casos de prueba positivos.

```
>  
<option value="0" selected="">----</option>  
<option value="310">Lorem ipsum dolor sit ame</option>  
<option value="112">Lorem ipsum dolor sit ame</option>  
<option value="113">Lorem ipsum dolor sit ame</option>  
</select>
```

```
<option value="0" selected="">----</option>  
<option value="10000">Lorem ipsum dolor sit ame</option>  
<option value="112">Lorem ipsum dolor sit ame</option>  
<option value="113">Lorem ipsum dolor sit ame</option>  
</select>
```

Unexpected error

We are very sorry! If this unexpected error prevents you from working with our system, please, do contact us and we will try to solve it as soon as possible.

Request:

POST http://localhost:8080/Acme-ANS-D04/customer/booking/create HTTP/1.1

Status:

500 Internal Server Error

Exceptions:

acme.client.helpers.Assert.state(Assert.java:45)

↳ java.lang.AssertionError: Access is not authorised

```
@Override  
public void authorise() {  
    boolean validFlight = true;  
    boolean validId = true;  
    if (super.getRequest().getMethod().equals("POST")) {  
        int flightId = super.getRequest().getData("flight", int.class);  
        if (flightId != 0) {  
            Flight flight = this.repository.findFlightById(flightId);  
            validFlight = flight != null && !flight.isDraftMode();  
        }  
        int id = super.getRequest().getData("id", int.class, 0);  
        validId = id == 0;  
    }  
    super.getResponse().setAuthorised(validFlight && validId);  
}
```

Actualizar, publicar

Las pruebas para actualizar y publicar son idénticas por lo tanto las explico conjuntamente.

Para el **update.safe** y **publish.safe** nos metemos en una booking ya creada y hacemos lo mismo que hemos hecho en el create, primero dejamos todo el formulario en blanco y después se prueba cada atributo individualmente con los valores positivos y negativos del Excel “Sample-Data”. Por último lo actualizamos y publicamos con nuevos valores positivos.

Locator Code

Lorem ipsum

Invalid locator code. It must be between 6 and 8 characters long, using only uppercase letters and numbers. May not be null.

Purchase Moment

yyyy/mm/dd hh:mm

May not be null.

Travel Class

May not be null.

Flight

May not be null.

Price

Last Nibble

Lorem ipsum

The last nibble must be exactly 4 digits.

☐ Confirm

Must confirm this.

Passengers

Update

Add Passengers

Publish

Delete

Booking Details

Locator Code

ABDD1234

Purchase Moment

yyyy/mm/dd hh:mm

May not be null.

Travel Class

May not be null.

Flight

May not be null.

Price

Last Nibble

Lorem ipsum

The last nibble must be exactly 4 digits.

☐ Confirm

Must confirm this.

Para el **update.hack** y **publish.hack** con la herramienta para desarrolladores de Firefox se prueba que la booking que se actualiza y publica no existe, que es de otro customer, que está ya publicada y también se hacen las pruebas de hacking que se han llevado a cabo en el **create.hack** cambiando las propiedades “value” correspondiente en cada caso, esperando un error 500. Aquí las demás propiedades las debo tener con casos de prueba positivos.

```
public void authorize() {
    int bookingId = super.getRequest().getData("id", int.class);
    int customerId = super.getRequest().getPrincipal().getActiveRealm().getId();
    Customer customer = this.repository.findCustomerLogged(customerId);
    Booking booking = this.repository.findBookingById(bookingId);
    boolean validFlight = true;
    if (super.getRequest().getMethod().equals("POST")) {
        int flightId = super.getRequest().getData("flight", int.class);
        if (flightId != 0) {
            Flight flight = this.repository.findFlightById(flightId);
            validFlight = flight != null && !flight.isDraftMode();
        }
    }
    super.getResponse().setAuthorised(booking != null && customer.equals(booking.getCustomer()) && booking.isDraftMode() && validFlight);
}
```

```
<form id="form"> (event)
  <input id="id" name="id" value="10000" type="hidden"> (event)
  <input id="customer" name="customer" value="1" type="hidden"> (event)
  <input id="flight" name="flight" value="314" type="hidden"> (event)
  <input id="price" name="price" value="309" type="hidden"> (event)
  <input id="nibble" name="nibble" value="4" type="hidden"> (event)
  <input type="checkbox" value="confirm" /> Confirm
  <input type="button" value="Passengers" />
  <input type="button" value="Update" />
  <input type="button" value="Add Passengers" />
  <input type="button" value="Publish" />
  <input type="button" value="Delete" />
```


Unexpected error

We are very sorry! If this unexpected error prevents you from working with our system, please, do contact us and we will try to solve it as soon as possible.

Request:
POST http://localhost:8080/Acme-ANS-D04/customer/booking/update HTTP/1.1

[Return](#)

Status:
500 Internal Server Error

Exceptions:
`acme.client.helpers.Assert.state(Assert.java:45)`
`java.lang.AssertionError: Access is not authorised`

Borrar

Para el **delete.safe** me voy a una booking no publicada, ya que en las publicadas no me sale botón de borrar, y pruebo que borre.

Booking Details

Locator Code

123456

Purchase Moment

2012/01/01 00:00

Travel Class

ECONOMY

Flight

Lorem ipsum dolor sit ame

Price

EUR 500,000.00

Last Nibble

1234

☐ Confirm

Passengers

Update

Add Passengers

Publish

Delete

Para el **delete.hack** se ha cambiado en la URL “show” por “delete”, saliendo el correspondiente error 500 “Access is not authorised”, habiendo mostrado previamente una booking publicada. Por último, se mostraba una booking, después se copiaba esa URL, tras ello el deslogueo y se pegaba esa URL sustituyendo “show” por “delete”, saliendo el correspondiente error 500 “Access is not authorised”.

XXXX123	2012/01/01 00:00	ECONOMY	EUR 500,000.00	1234	false
---------	------------------	---------	----------------	------	-------

localhost:8080/Acme-ANS-D04/customer/booking/delete?id=309

Unexpected error

We are very sorry! If this unexpected error prevents you from working with our system, please, do contact us and we will try to solve it as soon as possible.

Request:

GET http://localhost:8080/Acme-ANS-D04/customer/booking/delete?id=309 HTTP/1.1

[Return](#)

Status:

500 Internal Server Error

Exceptions:

acme.client.helpers.Assert.state(Assert.java:45)

↳ java.lang.AssertionError: Access is not authorised

Booking Record

Hay que tener en cuenta que en Booking Record no hay pruebas de actualizar ni publicar.

Listar

Para llevar a cabo el **list.safe** lo que hemos hecho es listar las bookings, meternos en una booking y pulsar el botón passengers accediendo a este listado.

Booking Details

Locator Code

123456

Purchase Moment

2012/01/01 00:00

Travel Class

ECONOMY

Flight

Lorem ipsum dolor sit ame

Price

EUR 500,000.00

Last Nibble

1234

☐ Confirm

Passengers

Update

Ad

Para el **list.hack** me he metido con otro usuario (ej:manager1) y he intentado acceder a la URL del listado del bookingRecord.

localhost:8080/Acme-ANS-D04/customer/booking-record/list

Unexpected error

We are very sorry! If this unexpected error prevents you from working with our system, please, do contact us and we will try to solve it as soon as possible.

Request:

GET http://localhost:8080/Acme-ANS-D04/customer/booking-record/list HTTP/1.1

Return

Status:

500 Internal Server Error

Exceptions:

acme.client.helpers.Assert.state(Assert.java:45)

↳ java.lang.AssertionError: Access is not authorised

Mostrar

Para el **show.safe** lo que hemos hecho es listar las Bookings Record, accediendo a este listado mediante se ha explicado en el list, y pinchar en una de los pasajeros de esa lista. El botón de delete aparece si no está publicado.

Booking Records

Search:

Passport Number	Date of birth	Special needs	DraftMode
B1000001	2012/01/01 00:00	None	true

Show 1 items

Passenger

Full Name

L

Email

user1@example.com

Passport number

B1000001

Special needs

None

Date of birth

2012/01/01 00:00

Delete Passenger

Para el **show.hack** he probado a cambiar la URL de la bookingRecord para que en vez de aparecerme esa bookingRecord me apareciera una que no está creada y otra bookingRecord de otro customer. En ambos casos me ha salido error 500.

```
@Override
public void authorise() {
    int customerId = this.getRequest().getPrincipal().getActiveRealm().getId();
    Customer customer = this.repository.findCustomerById(customerId);
    int bookingRecordId = super.getRequest().getData("id", int.class);
    BookingRecord bookingRecord = this.repository.findBookingRecordById(bookingRecordId);
    super.getResponse().setAuthorised(bookingRecord != null && customer.equals(bookingRecord.getBooking().getCustomer()));
}
```

localhost:8080/Acme-ANS-D04/customer/booking-record/show?id=10000

localhost:8080/Acme-ANS-D04/customer/booking-record/show?id=388

Unexpected error

We are very sorry! If this unexpected error prevents you from working with our system, please, do contact us and we will try to solve it as soon as possible.

Request:

GET http://localhost:8080/Acme-ANS-D04/customer/booking-record/show?id=388 HTTP/1.1

Return

Status:

500 Internal Server Error

Exceptions:

acme.client.helpers.Assert.state(Assert.java:45)

↳ java.lang.AssertionError: Access is not authorised

Crear

Para el **create.safe** hemos entrado en el formulario de crear un passenger en una booking mediante el botón de addPassengers que aparece en la lista de booking Record. Primero de todo, hacíamos una prueba sin elegir pasajero y después elegimos .

Passenger

Choose Passenger

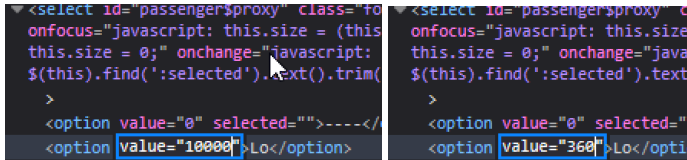
May not be null.

Add Passengers

Add Passenger

Para el **create.hack** con la herramienta para desarrolladores de Firefox se prueba cambiando la propiedad “value” por un valor de un pasajero que no existiera y uno de otro customer en el choosePassenger. Además, también se prueba que la booking sea de ese customer y que exista. Todo se prueba esperando un error 500.

```
@Override
public void authorise() {
    int customerId = this.getRequest().getPrincipal().getActiveRealm().getId();
    int bookingId = super.getRequest().getData("bookingId", int.class);
    Customer customer = this.repository.findCustomerById(customerId);
    Booking booking = this.repository.findBookingById(bookingId);
    boolean validPassenger = true;
    boolean validId = true;
    if (super.getRequest().getMethod().equals("POST")) {
        int passengerId = super.getRequest().getData("passenger", int.class);
        if (passengerId != 0) {
            Passenger passenger = this.repository.findPassengerById(passengerId);
            validPassenger = passenger != null && passenger.getCustomer().equals(customer) && this.repository.findBookingById(
            )
            int id = super.getRequest().getData("id", int.class, 0);
            validId = id == 0;
        }
        super.getResponse().setAuthorised(booking != null && booking.getCustomer().equals(customer) && validPassenger && validId);
    }
}
```



Unexpected error

We are very sorry! If this unexpected error prevents you from working with our system, please, do contact us and we will try to solve it as soon as possible.

Request:

POST http://localhost:8080/Acme-ANS-D04/customer/booking-record/create?bookingId=288 HTTP/1.1

Return

Status:

500 Internal Server Error

Exceptions:

acme.client.helpers.Assert.state(Assert.java:45)

↳ java.lang.AssertionError: Access is not authorised

Borrar

Para el **delete.safe** me voy a un pasajero de una booking mediante el botón passengers y pincho en uno, que tiene que estar en no publicada la booking ya que en las publicadas no me sale botón de borrar, y pruebo que borre.

Passenger

Full Name

L

Email

user1@example.com

Passport number

B1000001

Special needs

None

Date of birth

2012/01/01 00:00

Delete Passenger

Para el **delete.hack** se ha cambiado en la URL “show” por “delete”, saliendo el correspondiente error 500 “Access is not authorised”, habiendo mostrado previamente una booking publicada. Por último, se mostraba una booking record, después se copiaba esa URL, tras ello el deslogueo y se pegaba esa URL sustituyendo “show” por “delete”, saliendo el correspondiente error 500 “Access is not authorised”.

XXXX123	2012/01/01 00:00	ECONOMY	EUR 500,000.00	1234	false
---------	------------------	---------	----------------	------	-------

localhost:8080/Acme-ANS-D04/customer/booking-record/delete?id=383

Unexpected error

We are very sorry! If this unexpected error prevents you from working with our system, please, do contact us and we will try to solve it as soon as possible.

Request:

GET http://localhost:8080/Acme-ANS-D04/customer/booking-record/delete?id=383 HTTP/1.1

Return

Status:

500 Internal Server Error

Exceptions:

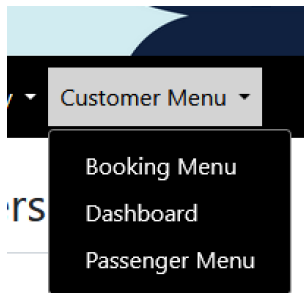
acme.client.helpers.Assert.state(Assert.java:45)

↳ java.lang.AssertionError: Access is not authorised

Passengers

Listar

Para llevar a cabo el **list.safe** lo que hemos hecho es listar los passengers, accediendo a este listado mediante el botón Passenger Menu.



Para el **list.hack** me he metido con otro usuario (ej:manager1) y he intentado acceder a la URL del listado del Passenger.

localhost:8080/Acme-ANS-D04/customer/passenger/list

Unexpected error

We are very sorry! If this unexpected error prevents you from working with our system, please, do contact us and we will try to solve it as soon as possible.

Request:

GET http://localhost:8080/Acme-ANS-D04/customer/passenger/list HTTP/1.1

Return

Status:

500 Internal Server Error

Exceptions:

acme.client.helpers.Assert.state(Assert.java:45)

↳ java.lang.AssertionError: Access is not authorised

Mostrar

Para el **show.safe** lo que hemos hecho es listar los passengers, accediendo a este listado mediante se ha explicado en el list, y pinchar en una de los pasajeros de esa lista. El botón de delete aparece si no está publicado.

Passenger Details

Full Name	<input type="text" value=" <marquee>Hacked</marquee>"/>
Email	<input type="text" value="user1@example.com"/>
Passport number	<input type="text" value="B1000001"/>
Special needs	<input type="text" value="None"/>
Date of birth	<input type="text" value="2012/01/01 00:00"/>
<input type="checkbox"/> Confirm	
<div><div>Update</div><div>Publish</div><div>Delete</div></div>	

Para el **show.hack** he probado a cambiar la URL del Passenger para que en vez de aparecerme ese passenger me apareciera uno que no exista y otro de otro customer. En ambos casos me ha salido error 500.

```
localhost:8080/Acme-ANS-D04/customer/passenger/show?id=10000
```

```
localhost:8080/Acme-ANS-D04/customer/passenger/show?id=360
```

Unexpected error

We are very sorry! If this unexpected error prevents you from working with our system, please, do contact us and we will try to solve it as soon as possible.

Request:
GET http://localhost:8080/Acme-ANS-D04/customer/passenger/show?id=360 HTTP/1.1

Status:
500 Internal Server Error

Exceptions:
acme.client.helpers.Assert.state(Assert.java:45)
↳ java.lang.AssertionError: Access is not authorised

[Return](#)

Crear

Para el **create.safe** hemos entrado en el formulario de crear un passenger mediante el botón de create que aparece en la lista de passengers. Primero de todo, hacíamos una prueba dejando el formulario en blanco, todos los atributos sin valor.

Passenger Details

Full Name

May not be null.

Email

May not be null.

Passport number

Invalid passport number. It must be 6 to 9 characters long, using only uppercase letters and numbers. May not be null.

Special needs

Must have from 1 to 50 characters.

Date of birth

May not be null.

☐ **Confirm**

Must confirm this.

Create

Después de esto, se prueba cada atributo individualmente con los valores positivos y negativos del Excel “Sample-Data”, dándole a “create” tras probar cada atributo y esperando los correspondientes mensajes. Por último, se crea una booking con todos los valores positivos.

Passenger Details

Full Name

Email

May not be null.

Passport number

Invalid passport number. It must be 6 to 9 characters long, using only uppercase letters and numbers. May not be null.

Special needs

Must have from 1 to 50 characters.

Date of birth

May not be null.

☐ **Confirm**

Must confirm this.

Create

Para el **create.hack** con la herramienta para desarrolladores de Firefox se prueba un valor de un passenger que no existiera y otro que no fuera de ese customer, esperando un error 500. Aquí las demás propiedades las tengo que tener con casos de prueba positivos.


```
@Override
public void authorise() {
    boolean validId = true;
    if (super.getRequest().getMethod().equals("POST")) {
        int id = super.getRequest().getData("id", int.class, 0);
        validId = id == 0;
    }
    super.getResponse().setAuthorised(validId);
}
```

```
<h1>Passenger Details</h1>
<form id="form" class="dirty">
  <input id="id" name="id" value="360">
  <input id="version" name="version" value="1">
```

```
<h1>Passenger Details</h1>
<form id="form" class="dirty">
  <input id="id" name="id" value="10000">
  <input id="version" name="version" value="1">
```

Unexpected error

We are very sorry! If this unexpected error prevents you from working with our system, please, do contact us and we will try to solve it as soon as possible.

Request:

POST http://localhost:8080/Acme-ANS-D04/customer/passenger/create HTTP/1.1

Return

Status:

500 Internal Server Error

Exceptions:

acme.client.helpers.Assert.state(Assert.java:45)

↳ java.lang.AssertionError: Access is not authorised

Actualizar, publicar

Las pruebas para actualizar y publicar son idénticas por lo tanto las explico conjuntamente.

Para el **update.safe** y **publish.safe** nos metemos en un passenger ya creado y hacemos lo mismo que hemos hecho en el create, primero dejamos todo el formulario en blanco y después se prueba cada atributo individualmente con los valores positivos y negativos del Excel “Sample-Data”. Por último lo actualizamos y publicamos con nuevos valores positivos.

Passenger Details	Passenger Details
Full Name Lorem ipsum	Full Name <marquee>Hacked</marquee>
Email Lorem ipsum	Email Lorem ipsum
Passport number Lorem ipsum	Passport number Lorem ipsum
Special needs Lorem ipsum	Special needs Lorem ipsum
Date of birth yyyy/mm/dd hh:mm	Date of birth yyyy/mm/dd hh:mm
<input type="checkbox"/> Confirm Must confirm this.	<input type="checkbox"/> Confirm Must confirm this.
Update Publish Delete	Update Publish Delete

Para el **update.hack** y **publish.hack** con la herramienta para desarrolladores de Firefox se prueba que el passenger que se actualiza y publica no existe, que es de otro customer, que está ya publicado y también se hacen las pruebas de hacking que se han llevado a cabo en el **create.hack** cambiando las propiedades “value”

correspondiente en cada caso, esperando un error 500. Aquí las demás propiedades las debo tener con casos de prueba positivos.

```
@Override
public void authorise() {
    int customerId = this.getRequest().getPrincipal().getActiveRealm().getId();
    Customer customer = this.repository.findCustomerById(customerId);
    int passengerId = super.getRequest().getData("id", int.class);
    Passenger passenger = this.repository.findPassengerById(passengerId);
    super.getResponse().setAuthorised(passenger != null && passenger.isDraftMode() && passenger.getCustomer().equals(customer));
}
```

```
<h1>Passenger Details</h1>
<form id="form">
  <input id="id" name="id" value="10000" type="text">
  <input id="version" name="version" value="0" type="text">
</form>
```

```
<h1>Passenger Details</h1>
<form id="form">
  <input id="id" name="id" value="354" type="hidden">
  <input id="version" name="version" value="0" type="text">
</form>
```

```
<h1>Passenger Details</h1>
<form id="form">
  <input id="id" name="id" value="360" type="text">
  <input id="version" name="version" value="0" type="text">
</form>
```

Unexpected error

We are very sorry! If this unexpected error prevents you from working with our system, please, do contact us and we will try to solve it as soon as possible.

Request:
POST http://localhost:8080/Acme-ANS-D04/customer/passenger/publish HTTP/1.1

Status:
500 Internal Server Error

Exceptions:
acme.client.helpers.Assert.state(Assert.java:45)
↳ java.lang.AssertionError: Access is not authorised

Return

Borrar

Para el **delete.safe** me voy a un passenger no publicado, ya que en los publicados no me sale botón de borrar, y pruebo que borre.

Passenger Details

Full Name

<marquee>Hacked</marquee>

Email

user1@example.com

Passport number

B1000001

Special needs

None

Date of birth

2012/01/01 00:00


☐ Confirm

Update

Publish

Delete

Para el **delete.hack** se ha cambiado en la URL “show” por “delete”, saliendo el correspondiente error 500 “Access is not authorised”, habiendo mostrado previamente un passenger publicado. Por último, se mostraba un passenger, después se copiaba esa URL, tras ello el deslogueo y se pegaba esa URL sustituyendo “show” por “delete”, saliendo el correspondiente error 500 “Access is not authorised”.

 Peter Parker	B1000001	2012/01/01 00:00	None	false
--	----------	---------------------	------	-------

localhost8080/Acme-ANS-D04/customer/passenger/delete?id=354

Unexpected error

We are very sorry! If this unexpected error prevents you from working with our system, please, do contact us and we will try to solve it as soon as possible.

Request:

GET http://localhost:8080/Acme-ANS-D04/customer/passenger/delete?id=354 HTTP/1.1

Return

Status:

500 Internal Server Error

Exceptions:

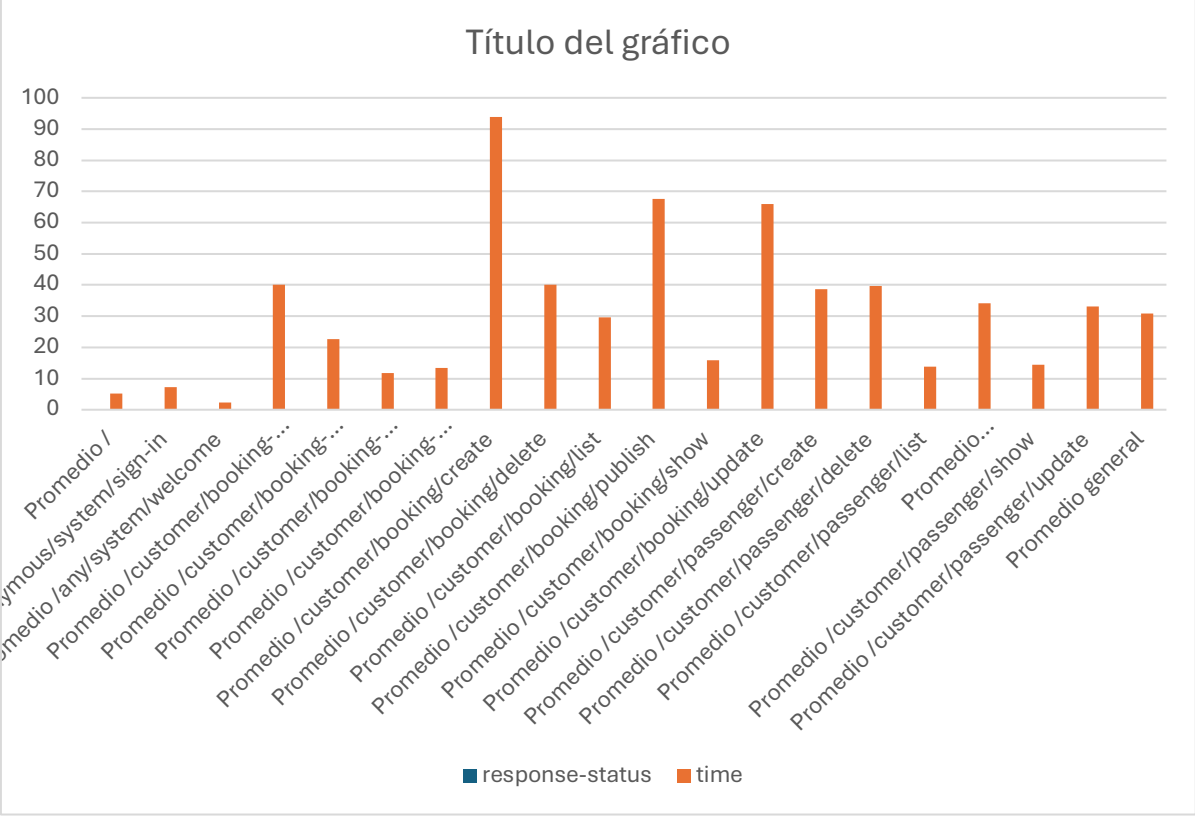
acme.client.helpers.Assert.state(Assert.java:45)
↳ java.lang.AssertionError: Access is not authorised



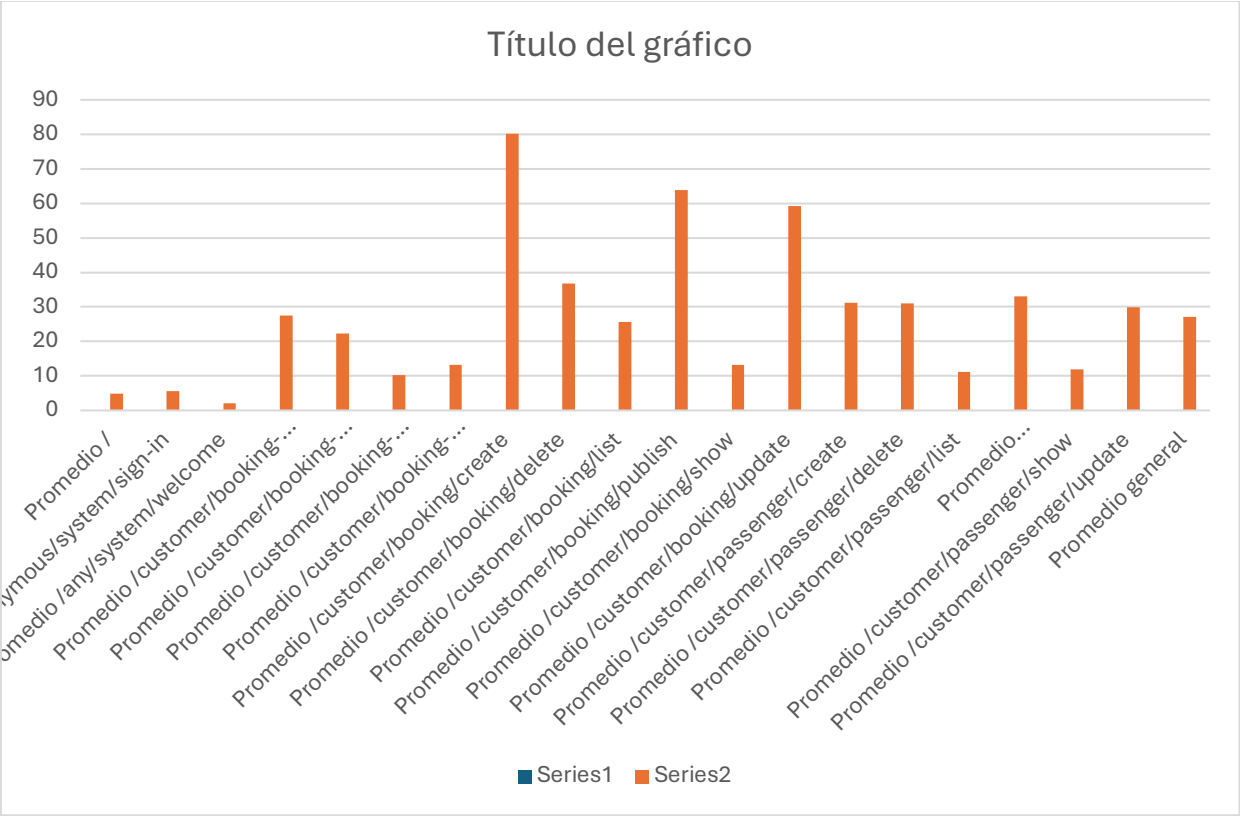
Capítulo sobre pruebas de rendimiento

Gráficas de rendimiento

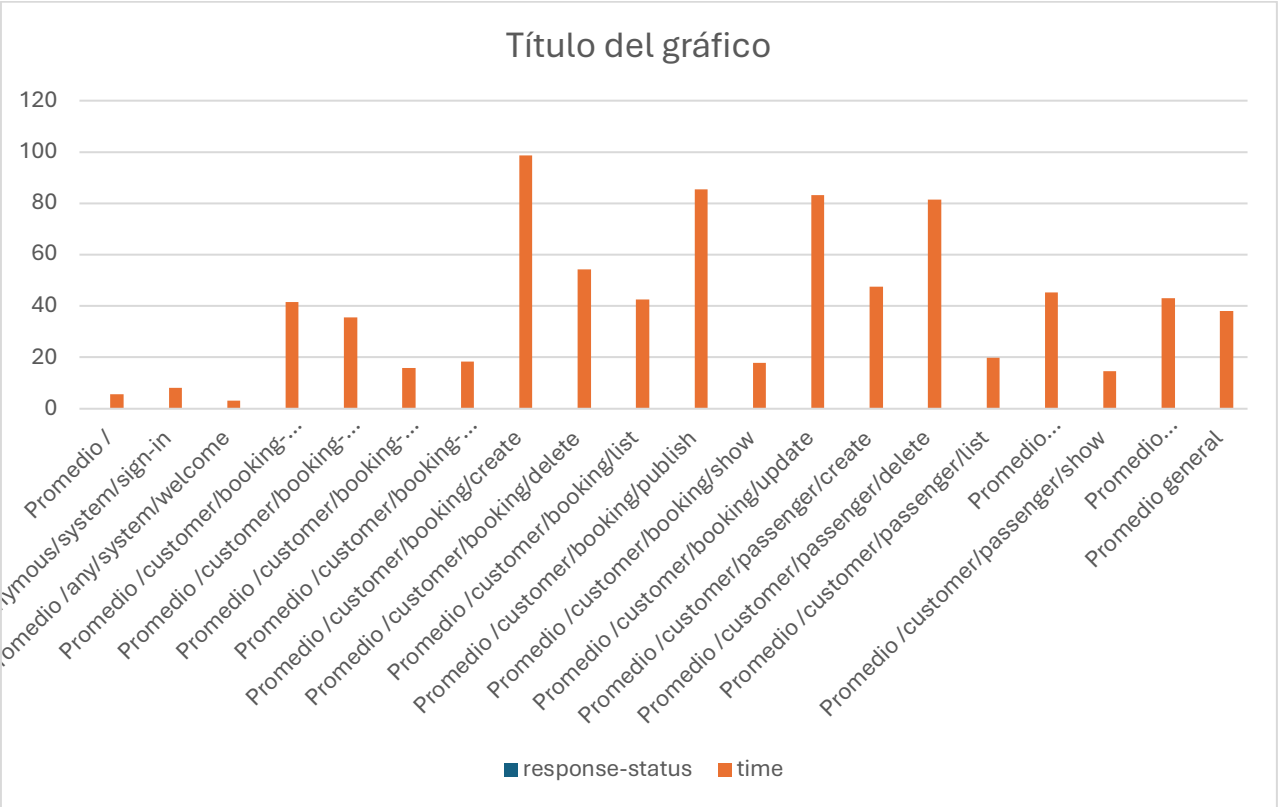
Sin índices



Con índices



Desde otro ordenador



Como podemos ver, todas las gráficas concuerdan en que las peticiones que más tarda son las de crear, actualizar y publicar booking ya que son las que más restricciones tienen que comprobar y las que más datos van a procesar.

Intervalo de confianza del 95%

Sin índices

Columna1					
Media	30,89548228	Interval(ms)	28,4619435	33,3290211	
Error típico	1,239455729	Interval(s)	0,02846194	0,03332902	
Mediana	24,87995				
Moda	#N/D				
Desviación estándar	32,65207268				
Varianza de la muestra	1066,157851				
Curtosis	7,394945773				
Coefficiente de asimetría	2,11655822				
Rango	262,0249				
Mínimo	1,2358				
Máximo	263,2607				
Suma	21441,4647				
Cuenta	694				
Nivel de confianza(95,0%)	2,433538778				

Los resultados estadísticos obtenidos en el análisis sin índice nos muestra que el intervalo de confianza se sitúa entre 28,46 ms y 33,3 ms lo que nos hace indicar que es bastante lento.

Con índices

	A	B	C	D	E	F
1	Columna1					
2				Interval(ms)	25,1061869	29,2842759
3	Media	27,19523141		Interval(s)	0,02510619	0,02928428
4	Error típico	1,06399707				
5	Mediana	22,21475				
6	Moda	1,2677				
7	Desviación estándar	28,02981085				
8	Varianza de la muestra	785,6702963				
9	Curtosis	6,906296114				
10	Coefficiente de asimetría	1,949378924				
11	Rango	247,8903				
12	Mínimo	1,1282				
13	Máximo	249,0185				
14	Suma	18873,4906				
15	Cuenta	694				
16	Nivel de confianza(95,0%)	2,089044463				
17						

Los resultados estadísticos obtenidos en el análisis con índice nos muestra que el intervalo de confianza se sitúa entre 25,10 ms y 29,28 ms lo que nos hace indicar que es bastante lento, aunque un poco menos que sin índice.

Desde otro ordenador

A	B	C	D	E	F
158,2426					
Media	37,86081962		interval(ms)	35,0872257	40,6344135
Error típico	1,41264983		interval(s)	0,03508723	0,04063441
Mediana	33,841				
Moda	2,0197				
Desviación estándar	37,18785584				
Varianza de la muestra	1382,936622				
Curtosis	2,235561953				
Coefficiente de asimetría	1,391044658				
Rango	239,0834				
Mínimo	1,6524				
Máximo	240,7358				
Suma	26237,548				
Cuenta	693				
Nivel de confianza(95,0%)	2,773593884				

Los resultados estadísticos obtenidos en el análisis en otro ordenador nos muestra que el intervalo de confianza se sitúa entre 35,08 ms y 40,63 ms lo que nos hace indicar que es bastante más lento que mi máquina.

Comparativa z-test

Sin índices vs con índices

A	B	C
Prueba z para medias de dos muestras		
	114,3946	113,033
Media	30,77499293	27,0713674
Varianza (conocida)	1066,15785	785,670296
Observaciones	693	693
Diferencia hipotética de las medias	0	
z	2,265652749	
P(Z<=z) una cola	0,011736326	
Valor crítico de z (una cola)	1,644853627	
Valor crítico de z (dos colas)	0,023472653	
Valor crítico de z (dos colas)	1,959963985	

El valor de p-value obtenido es de 0,011 por lo que se pueden comparar las medias, pero como la media de mi ordenador sin índice es 30,77 ms y con índice es de 27,07 ms, podemos afirmar que no hay apenas diferencia porque son muy cercanas las medias.

Mi ordenador (con índices) vs otro ordenador (con índices)

A	B	C
Prueba z para medias de dos muestras		
	113,033	158,2426
Media	27,07136739	37,86081962
Varianza (conocida)	785,670296	1066,15785
Observaciones	693	693
Diferencia hipotética de las medias	0	
z	-6,600330366	
P(Z<=z) una cola	2,05121E-11	
Valor crítico de z (una cola)	1,644853627	
Valor crítico de z (dos colas)	4,10243E-11	
Valor crítico de z (dos colas)	1,959963985	

Como el p-value obtenido es de 2.05121E-11, que es un valor muy cercano a 0, confirmamos que se pueden comparar las medias obtenidas. Como en mi ordenador la media es de 27,07ms y en el otro ordenador es de 37,86ms, llegamos a la conclusión de que mi ordenador es bastante mejor.

Conclusión

Las pruebas realizadas han permitido verificar que la aplicación cumple con los requisitos funcionales esperados y responde de forma segura ante intentos de acceso no autorizado. Las funcionalidades clave como la gestión de reservas, registros y pasajeros han sido validadas mediante casos positivos, negativos y de hacking, demostrando un comportamiento robusto y fiable.

Por otro lado, las pruebas de rendimiento han evidenciado que las operaciones de mayor complejidad, como crear o actualizar reservas, requieren más tiempo de procesamiento, especialmente sin optimizaciones. La inclusión de índices y la comparación entre diferentes equipos permiten identificar oportunidades claras de mejora en la eficiencia del sistema.

Bibliografía

Intencionalmente en blanco