ACME AirNav Solutions

TESTING REPORT – Student 2

Grupo: C1.050
Miembros: Cristina Fernández Chica (criferchi@alum.us.es), Ángel Amo Sánchez (angamosan@alum.us.es), Candela Jazmín Gutiérrez González (cangutgon@alum.us.es), Marta Aguilar Morcillo (maragumor@alum.us.es) y Luis Emmanuel Chávez Malavé (luichamal@alum.us.es)
Repositorio: https://github.com/Cristinafernandezchica/Acme-ANS
Planning dashboard: https://github.com/users/Cristinafernandezchica/projects/1/views/1

Sevilla 25 Mayo, 2025

TABLA DE CONTENIDOS

Resumen Ejecutivo

Este documento ofrece una descripción detallada de los diferentes procedimientos aplicados en el contexto del testing formal del proyecto Acme-ANS. Abarca desde la creación de una cantidad adecuada de datos de prueba hasta la verificación minuciosa del cumplimiento de los requisitos funcionales. Además, se explican las herramientas estadísticas empleadas para analizar los tiempos de respuesta entre distintas peticiones, evaluar el rendimiento comparativo entre dos equipos y observar las mejoras en un mismo equipo tras la incorporación de índices en la base de datos.

En definitiva, se ha adoptado una metodología rigurosa para identificar y corregir los errores detectados durante las pruebas, con el objetivo de garantizar un producto final de alta calidad que cumpla con las expectativas del cliente.

Tabla de Revisiones

Número de revisión	Fecha	Descripción de revisión	Autor
1.0	25/05/2025	Se han realizado todos los apartados del reporte.	Ángel Amo Sánchez

Introducción

Este informe tiene como propósito detallar el proceso seguido para llevar a cabo el testing funcional y el análisis de rendimiento en el marco del proyecto. En particular, se centra en la verificación de los requisitos funcionales 8 y 9 correspondientes al Student Nº2, siendo estos las operaciones sobre registros passengers, bookings y su clase intermedia booking-record.

La estructura del documento responde a los lineamientos establecidos en el anexo compartido a través de Enseñanza Virtual. Comienza con una portada que incluye la información del autor, seguida de una tabla de control de versiones donde se documentan las actualizaciones del informe, indicando el número de versión y la fecha de cada modificación. A continuación, se presenta un resumen ejecutivo que proporciona una visión general del contenido, acompañado de una introducción en la que se anticipan los temas tratados y se explica cómo se ha organizado el texto.

El cuerpo principal del informe se divide en dos secciones fundamentales, cada una dedicada a una fase específica del testing. La primera sección aborda las pruebas funcionales, mostrando los distintos casos de prueba agrupados por funcionalidad. De cada uno se ofrece una breve explicación junto con una evaluación de su capacidad para detectar fallos. La segunda sección está dedicada a las pruebas de rendimiento, donde se analizan los tiempos de respuesta mediante representaciones gráficas y se calcula un intervalo de confianza del 95% para dichos tiempos.

Esta parte también incluye una comparación entre dos escenarios —antes y después de aplicar índices a la base de datos— mediante un contraste de hipótesis (Z-test) que permite determinar si la mejora observada es estadísticamente significativa. Además del análisis de las pruebas en dos ordenadores diferentes y ver la diferencia de rendimientos.

Contenido

Pruebas funcionales

Para cada servicio se han realizado tanto pruebas positivas (.safe) como de hacking (.hack) de la siguiente forma:

/customer/booking/…

Tipo	Acción	Descripción	Bugs Detectados
List-mine.safe	list	Se listaron todas las reservas de los clientes individualmente. Se probaron listados con 0 y varias reservas (con y sin vuelos).	No se detectaron bugs.
Show-mine.safe	show	Se mostraron las reservas de cada cliente individualmente tanto publicadas como no publicadas, con vuelos (validos o caducados) como sin vuelos y las propiedades Transient cambiando.	No se detectaron bugs.
Create-mine.safe	create	Se probó cada variación posible de los campos del formulario de creación de reservas (Create-mine02.safe), formularios vacíos y validaciones.	No se detectaron bugs.
Update-mine.safe	update	Se verificaron variaciones del formulario de actualización de las reservas, el formulario vacío y comprobaciones como que las propiedades Transient van cambiando.	No se detectaron bugs.
Publish-mine.safe	publish	Se verificaron variaciones del formulario de publicación de las reservas, el formulario vacío y validaciones (vuelo válido asignado, tener pasajeros y publicados y el cardNibble tiene que estar presente) y como que las propiedades Transient van cambiando.	No se detectaron bugs.
List-mine.hack	list	Se intentó acceder al listado de reservas sin rol o con un rol incorrecto.	No se detectaron bugs.
Show-mine.hack	show	Se intentó ver reservas con un rol incorrecto o sin rol, ver reservas de otros clientes o ver reservas con un id incorrecto (-1 y 1)	No se detectaron bugs.
Create-mine.hack	create	Se intentó acceder al formulario de creación de reservas desde roles incorrectos o sin roles; o crear una reserva con un tipo de vuelo erróneo, un vuelo que no existe (flightId incorrecto: -1 y 1), vuelo en borrador o un vuelo pasado; o crear usando un parámetro id e intentar modificar los campos de solo lectura	He detectado y solucionado el no poder acceder a la URL de create con id, actuando como update.
Update-mine.hack	update	Se intentó acceder al formulario de actualización de reservas publicadas (del propio cliente como otro cliente), con id (incorrecto o publicadas) o sin id sin estar en una reserva, desde roles incorrectos o sin roles, actualizar campos de solo lectura o actualizar una reserva con un tipo de vuelo erróneo, un vuelo que no existe (flightId incorrecto: -1 y 1), vuelo en borrador o un vuelo pasado	He detectado y solucionado el no poder acceder a la URL de update con id.

Tipo	Acción	Descripción	Bugs Detectados
Publish-mine.hack	publish	Se intentó acceder al formulario de publicación de reservas publicadas (del propio cliente como otro cliente), con id o sin id sin estar en una reserva, desde roles incorrectos o sin roles, modificar campos de solo lectura o publicar una reserva con un tipo de vuelo erróneo, un vuelo que no existe (flightId incorrecto: -1 y 1), vuelo en borrador o un vuelo pasado.	He detectado y solucionado el no poder acceder a la URL de publish con id.

/customer/passenger/…

Tipo	Acción	Descripción	Bugs Detectados
List-mine.safe	list	Se listaron los pasajeros de los clientes (List-mine02.safe). Se probaron listados con 0 y varios pasajeros, tanto de los clientes individualmente como los pasajeros asignados a una reserva.	No se detectaron bugs.
Show-mine.safe	show	Se mostraron los pasajeros de cada cliente individualmente tanto publicados como no publicados (Show-mine03.safe).	No se detectaron bugs.
Create-mine.safe	create	Se probó cada variación posible de los campos del formulario de creación de pasajeros (Create-mine02.safe) y la creación de un pasajero desde una reserva, los formularios vacíos y validaciones.	No se detectaron bugs.
Update-mine.safe	update	Se verificaron variaciones del formulario de actualización de los pasajeros, el formulario vacío y validaciones como el mismo pasaporte en un mismo cliente o fecha de nacimiento pasada.	No se detectaron bugs.
Publish-mine.safe	publish	Se probó las mismas variaciones de los campos del formulario de publicación, el formulario vacío y validaciones anteriores.	No se detectaron bugs.
List-mine.hack	list	Se intentó acceder al listado de pasajeros sin rol o con un rol incorrecto (tanto de clientes como los pasajeros asignados a una reserva) o ver pasajeros de una reserva con id incorrecto (-1 y 1).	No se detectaron bugs.
Show-mine.hack	show	Se intentó ver pasajeros con un rol incorrecto o sin rol, ver pasajeros de otros clientes o ver pasajeros con un id incorrecto (-1 y 1)	No se detectaron bugs.
Create-mine.hack	create	Se intentó acceder al formulario de creación de pasajeros desde roles incorrectos o sin roles; o crear un pasajero en una reserva que no existe (bookingId incorrecto: -1 y 1) o que ya esté publicado; o crear usando un parámetro id	He detectado y solucionado el no poder acceder a la URL de create con id, actuando como update.
Update-mine.hack	update	Se intentó acceder al formulario de actualización de pasajeros publicados (del propio cliente como otro cliente), con id (incorrecto o publicado) o sin id sin entrar en una reserva, desde roles incorrectos o sin roles	He detectado y solucionado el no poder acceder a la URL de update con id.

Tipo	Acción	Descripción	Bugs Detectados
Publish-mine.hack	publish	Se intentó acceder al formulario de publicación de pasajeros publicados (del propio cliente como de otro cliente), con id (incorrecto o publicado) o sin id sin entrar en una reserva, desde roles incorrectos o sin roles	He detectado y solucionado el no poder acceder a la URL de publish con id.

/customer/booking-record/…

Tipo	Acción	Descripción	Bugs Detectados
Create-mine.safe	create	Se probó a vincular los pasajeros disponibles en el desplegable, además de probar sin tarea "---". - **Create-mine02.safe**: Desplegables de distintas reservas para mayor cantidad de variedad de datos.	No se encontraron bugs.
Create-mine.hack	create	**Sin rol o con rol incorrecto:** Intentos de crear siendo administrator o sin rol. **Id incorrecto**: Intentos de acceso a la pantalla de vinculación con id 1 y -1 y a reservas existentes ya publicadas. **Con rol customer:** Acceso a formularios de creación pertenecientes a otro customer. **Pasajeros incorrectos:** Intentos de creación con pasajeros invalidos de id= -1, 1, "", pasajeros de otros customer o pasajeros ya asignados en la reserva y no aparecen en el seleccionador.	He detectado y solucionado el no poder acceder a la URL de un booking-record en borrador por id, actuando como update

Recubrimiento del código

Las pruebas son bastante fiables al contar con un porcentaje de cobertura en torno al 99%, pues tenemos que tener en cuenta las instrucciones que no se realizarán o únicamente en casos extremos (como en el BookingCreate, la búsqueda de códigos identificadores libres tras el caso de la asignación completa de la secuencia).

Element	Coverage	Covered Instru...	Missed Instruct...	Total Instructio...
acme.features.customer.booking	99,0 %	1.894	19	1.913
CustomerBookingCreateService.java	97,5 %	508	13	521
CustomerBookingUpdateService.java	98,9 %	463	5	468
CustomerBookingShowService.java	99,7 %	343	1	344
CustomerBookingController.java	100,0 %	30	0	30
CustomerBookingListService.java	100,0 %	67	0	67
CustomerBookingPublishService.java	100,0 %	483	0	483
acme.features.customer.passenger	99,4 %	986	6	992
CustomerPassengerPublishService.java	98,6 %	212	3	215
CustomerPassengerUpdateService.java	98,6 %	209	3	212
CustomerPassengerController.java	100,0 %	30	0	30
CustomerPassengerCreateService.java	100,0 %	304	0	304
CustomerPassengerListService.java	100,0 %	131	0	131
CustomerPassengerShowService.java	100,0 %	100	0	100
acme.features.customer.bookingRecord	98,9 %	278	3	281
CustomerBookingRecordCreateService.java	98,9 %	269	3	272
CustomerBookingRecordController.java	100,0 %	9	0	9
acme.features.customer.recommendation	89,5 %	342	40	382
RecommendationApiService.java	89,5 %	342	40	382
acme.entities.claims	61,8 %	47	29	76
acme.features.administrator.recommendation	91,5 %	300	28	328
AdministratorRecommendationPopulateServic	91,2 %	290	28	318
AdministratorRecommendationController.java	100,0 %	10	0	10

Análisis de desempeño

En primer lugar, analizamos las estadísticas sobre el rendimiento de la aplicación antes y después de los índices, así como sus gráficas de tiempo. Para poder comparar estos tiempos, se ha realizado el Z-test y lo compararemos con las estadísticas descriptivas para poder llegar a una conclusión y comprobar si los cambios tienen significancia estadística.

Prueba z para medias de dos muestras

	Before	*After*
Media	26,9693858	30,1841121
Varianza (conocida)	9338,13112	5588,01545
Observaciones	826	786
Diferencia hipotética de las medias	0	
z	-0,74913818	
P(Z<=z) una cola	0,22688696	
Valor crítico de z (una cola)	1,64485363	
Valor crítico de z (dos colas)	0,45377392	
Valor crítico de z (dos colas)	1,95996398	

Como podemos observar, el p-valor obtenido en la prueba Z es de 0,45377392. Considerando un nivel de confianza del 95% y un nivel de significancia de α = 0.05. Dado que el p-valor es bastante mayor que α, indica que los cambios realizados no tienen significancia estadística. Es cierto que, tras los índices fue necesario modificar los tests, ya que realizaba la búsqueda ordenando los valores de los vuelos de una forma diferente proporcionando unos valores diferentes como resultado; esto llevó a reemplazar los tests conflictivos. Además, se realizaron refactorizaciones y se tomaron menos casos de datos (ya que observe que en ciertos tests se repetían mucho varios datos).

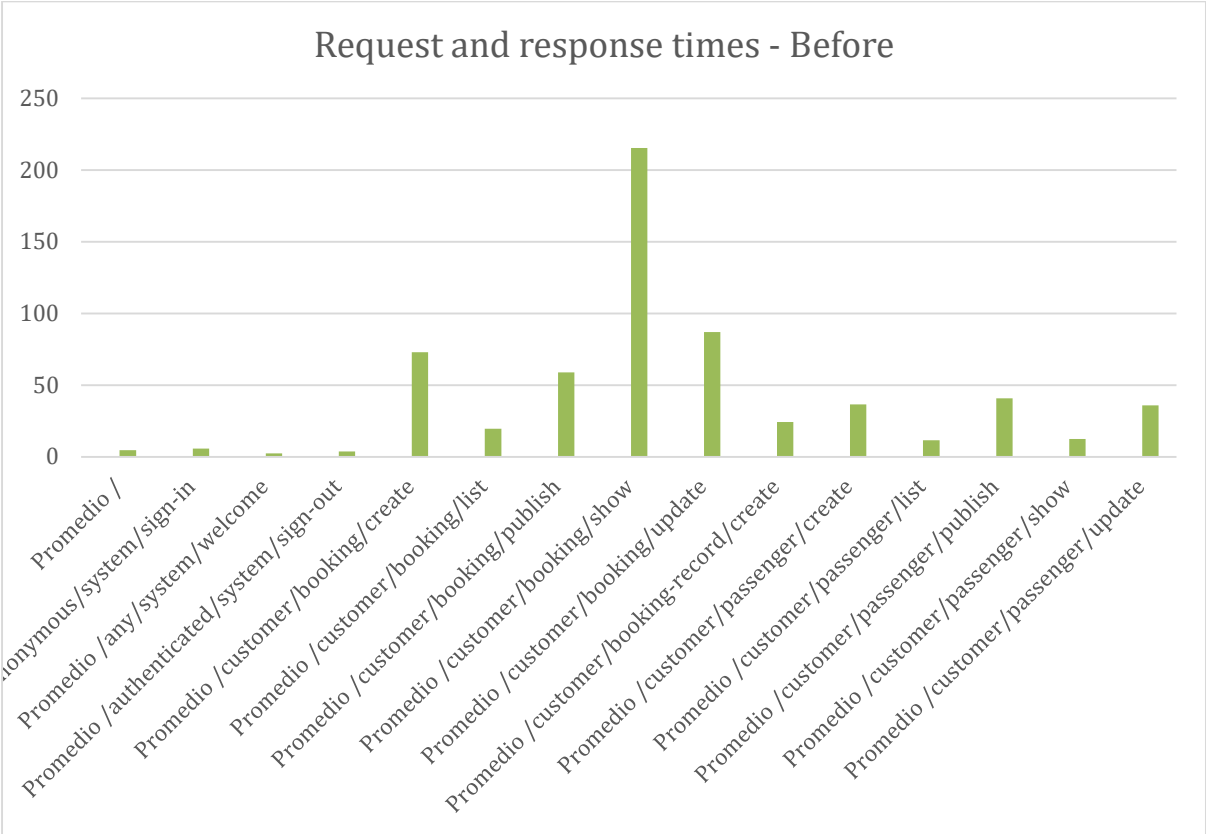
Por lo tanto, es cierto que con los datos proporcionados por la prueba Z se puede decir que no ha sido significativo. Sin embargo, la varianza es muchísimo menor para tan poca diferencia que hay entre números de observaciones y los valores. Por lo general, es cierto que los datos han disminuido, esto lo podemos ver en la siguiente comparación entre tiempos:

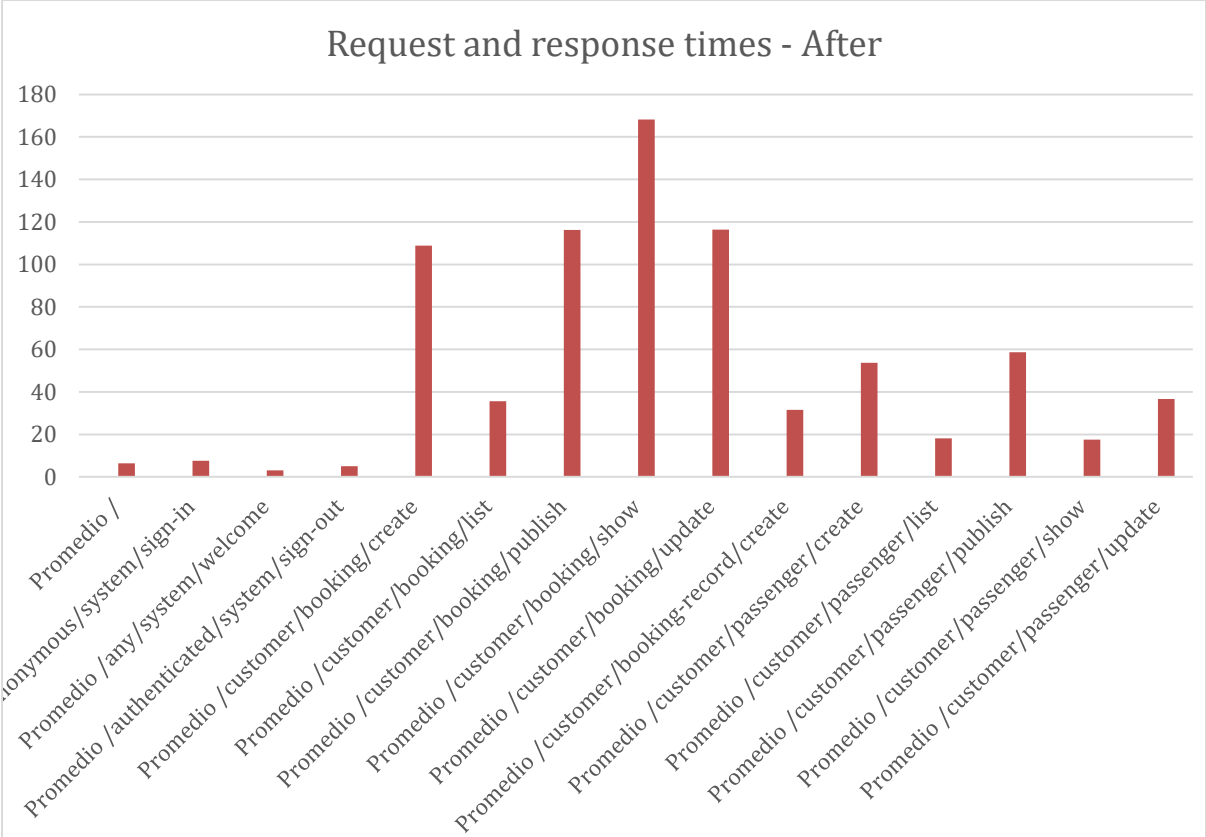
Before		*After*	
Media	26,9693858	Media	30,1841121
Error típico	3,36232709	Error típico	2,66635227
Mediana	6,01115	Mediana	5,90275
Moda	2,1698	Moda	2,417
Desviación estándar	96,634006	Desviación estándar	74,7530297
Varianza de la muestra	9338,13112	Varianza de la muestra	5588,01545
Curtosis	136,249577	Curtosis	97,4354914
Coeficiente de asimetría	10,8107778	Coeficiente de asimetría	8,05546062
Rango	1560,3675	Rango	1215,6935
Mínimo	1,3394	Mínimo	1,3703
Máximo	1561,7069	Máximo	1217,0638
Suma	22276,7127	Suma	23724,7121
Cuenta	826	Cuenta	786
Nivel de confianza(95,0%)	6,59972227	Nivel de confianza(95,0%)	5,23402436
Interval (ms)	20,3696636 33,5691081	Interval (ms)	24,9500877 35,4181364
Interval (s)	0,02036966 0,03356911	Interval (s)	0,02495009 0,03541814

Como se puede apreciar, pese a la diferencia de cantidad de observaciones por lo general la comparación es notoria pues podemos resolver peticiones en un tiempo menor o igual a 33,5691081 ms y este valor frente al optimizado ha aumentado 1,84902834 ms, por lo que, en este aspecto estaríamos hablando de un empeoramiento del rendimiento. Sin embargo, no será percibida por la diferencia de tests y las varianzas, por lo cual, teniendo un valor de p= 0,45377392 el resultado no es significativo como para ser visible y los tiempos no son significativamente diferentes entre ellos.

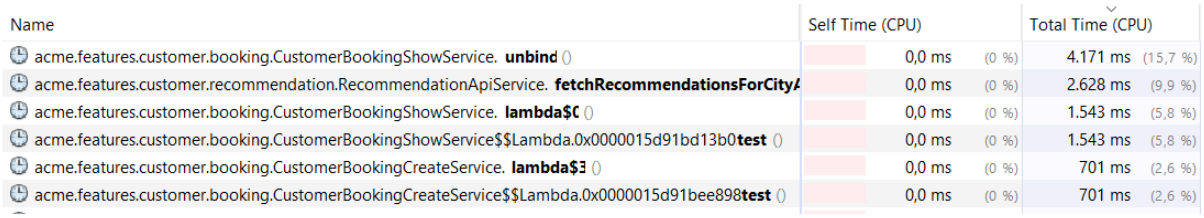
Gráficas

Tras el análisis podremos observar cómo han quedado finalmente las graficas junto con un análisis de los MIR que no han sido posible de reducir:





Tras mostrar las gráficas como podemos ver en VisualVM:



Name	Self Time (CPU)	Total Time (CPU)
acme.features.customer.booking.CustomerBookingShowService. **unbind** ()	0,0 ms (0 %)	4.171 ms (15,7 %)
acme.features.customer.recommendation.RecommendationApiService. **fetchRecommendationsForCity**	0,0 ms (0 %)	2.628 ms (9,9 %)
acme.features.customer.booking.CustomerBookingShowService. **lambda$** ()	0,0 ms (0 %)	1.543 ms (5,8 %)
acme.features.customer.booking.CustomerBookingShowService$$Lambda.0x0000015d91bd13b0**test** ()	0,0 ms (0 %)	1.543 ms (5,8 %)
acme.features.customer.booking.CustomerBookingCreateService. **lambda$** ()	0,0 ms (0 %)	701 ms (2,6 %)
acme.features.customer.booking.CustomerBookingCreateService$$Lambda.0x0000015d91bee898**test** ()	0,0 ms (0 %)	701 ms (2,6 %)

El principal problema ha sido la incorporación del Suplementary II, el cual se muestra en el bookingShow y con la API implementada hace muchas llamadas y toma muchos recursos como se puede apreciar, ya que luego estos datos tienen que ser procesados en el unbind(). La columna de los Recomendations fue excluida de los gráficos al no formar parte de los Mandatory, sin embargo, no es posible eliminarlo de Show al consistir en una serie de llamadas externas (y estas aparecen cada vez que se hace un show de una reserva publicada, teniendo en cuenta que se ha hecho Show de todas las reservas del sistema…), todo esto es comprobable con el Self Time (CPU) que es igual a 0,0 ms midiendo que el rendimiento excesivo no es por el método en sí, sino por las llamadas de su interior.

Lo que sí quiero que se tenga en cuenta son todas las demás barras que tenemos y se puede observar como, por lo general, los valores que eran mayores mejoran (especialmente el show). Por lo cual, puedo concluir que la implementación de índices en este sentido ha sido beneficiosa para los valores más altos, beneficiosa para los tests no modificados en el “After” y para las clases que fueron modificadas tras los índices (por el problema explicado anteriormente) los resultados han sido diversos entre empeoramientos (en su mayoría) y beneficios. Sin embargo, esto no es algo que pueda ser refactorizado, pues, al ser una API externa no es posible simplificar más las llamadas de esta. Todas las demás entidades fueron refactorizadas mediante la eliminación de ciertos “if” omisibles, redundantes o innecesarios; o la eliminación de código, como pueden ser ciertos hackeos que anteriormente se controlaban mediante validadores o excepciones ahora se controlan mediante el authorise y el código no es necesario.

Comparación de rendimiento con otro ordenador

Finalmente realizaremos una comparativa entre dos ordenadores distintos, pudiendo demostrar como el hardware también influye a la hora de responder peticiones. La comparativa es únicamente tras la optimización de los índices.
El PC_1 es el mismo utilizado para el reporte, un portátil LG GRAM con un Intel Core i7 sin tarjeta gráfica dedicada, y para el PC_2 se ha utilizado un portátil HP Victus 16-d1001ns con procesador Intel Core i7 12700H y tarjeta gráfica dedicada (GeForce RTX 3050 Ti) de una de mis compañeras del grupo.

PC_1	
Media	30,1841121
Error típico	2,66635227
Mediana	5,90275
Moda	2,417
Desviación estándar	74,7530297
Varianza de la muestra	5588,01545
Curtosis	97,4354914
Coeficiente de asimetría	8,05546062
Rango	1215,6935
Mínimo	1,3703
Máximo	1217,0638
Suma	23724,7121
Cuenta	786
Nivel de confianza(95,0%)	5,23402436

Interval (ms)	24,9500877	35,4181364
Interval (s)	0,02495009	0,03541814

PC_2	
Media	22,2485926
Error típico	2,8110071
Mediana	5,028
Moda	2,6518
Desviación estándar	78,8085279
Varianza de la muestra	6210,78408
Curtosis	225,007858
Coeficiente de asimetría	13,0348915
Rango	1600,4854
Mínimo	1,1494
Máximo	1601,6348
Suma	17487,3938
Cuenta	786
Nivel de confianza(95,0%)	5,51798043

Interval (ms)	16,7306122	27,7665731
Interval (s)	0,01673061	0,02776657

Podemos observar como el PC_2, el intervalo de confianza con una cota superior de 27,7665731 ms, siendo esta un 78,4% de la cota superior del PC_1. Ante esto podemos concluir con que el PC_2 tiene un mejor desempeño con respecto a tiempos de respuesta, siendo notable que es menor que la del PC_1.

Conclusión

En este informe se analizamos el proceso de testing formal tras completar el proyecto y con el objetivo de encontrar todos los errores posibles y así corregirlos. Las pruebas realizadas indican que se han probado múltiples escenarios posibles y nos permiten documentar ciertos bugs y fallos del sistema para los requisitos 8 y 9 del Student 2, pudiendo reducir la posibilidad de fallo del sistema al mínimo y a la misma vez mejorar el rendimiento del mismo.

Por otra parte, el análisis de rendimiento ha descubierto, mediante un Z-test con significancia estadística del 95%, que las optimizaciones aplicadas para una mayor rapidez en la búsqueda en base de datos, no han sido suficientes para mejorar el tiempo de respuesta del sistema, lo que nos ha llevado a identificar ciertos problemas con algunos métodos, que a pesar de ser revisados exhaustivamente e incluso refactorizados, no se ha tenido éxito en la mejora del rendimiento.

En conjunto, los resultados en base a los test realizados demuestran que es un sistema con una calidad funcional bastante robusta y nos asegura el cumplimiento de los estándares de calidad del proyecto, pero con un rendimiento que puede llegar a dar ciertos problemas en algunos dispositivos con sistemas menos avanzados como puede ser este mismo.

De esto sacamos que debemos revisar la eficiencia desde un inicio a la hora de implementar un proyecto, pues a la hora de querer refactorizar puede dar bastantes problemas y resultar en un sistema demasiado lento. Debemos seguir intentando de todas las formas el limitar llamadas o la reducción de código demasiado largo o inservible.

Bibliografía

Enseñanza virtual: https://ev.us.es