

Universidad de Sevilla

Escuela Técnica Superior de Ingeniería Informática

TestingReport



Acme ANS

OUR FIRST PROJECT IN D&T

Grado en Ingeniería Informática – Ingeniería del Software


Diseño y Pruebas 2

Curso 2024 – 2025

Grupo de prácticas: C1-055


Autores por orden alfabético

Eduardo Bustamante Lucena

	Diseño y Pruebas II Acme-Software-Factory
	Testing Report


Índice de contenido

1. Resumen ejecutivo	3
2. Tabla de revisiones	4
3. Introducción	5
4. Contenido	6
Pruebas funcionales	6
Pruebas de rendimiento	13
Análisis de los resultados obtenidos.....	17
5. Conclusiones	19
6. Bibliografía.....	20

	Diseño y Pruebas II Acme-Software-Factory
	Testing Report


1. Resumen ejecutivo

En este informe se presenta un análisis detallado de las pruebas y los resultados obtenidos para las funcionalidades correspondientes al módulo Student 2.

	Diseño y Pruebas II Acme-Software-Factory
	Testing Report

2. Tabla de revisiones

Fecha	Versión	Descripción
20/05/2025	1.0	Primera versión del documento
24/05/2025	1.1	Revisión del análisis de rendimiento


	Diseño y Pruebas II Acme-Software-Factory
	Testing Report

3. Introducción

El documento se divide en dos partes principales: pruebas funcionales y pruebas de rendimiento. El objetivo de este informe es proporcionar un análisis detallado de los casos de prueba y del rendimiento de los mismo.

La primera parte del informe abarca las pruebas funcionales, las cuales se centran en evaluar las funcionalidades de dos entidades específicas: booking y passenger y una última resultante del estudio de los requisitos bookingRecord. Se presentan los casos de prueba realizados, organizados por entidad, con descripciones claras y precisas para cada uno. Esto permite una evaluación concisa y efectiva de las funcionalidades asociadas a dichas entidades.

En la segunda parte, el documento se centrará en proporcionar gráficos detallados que muestran los tiempos de respuesta del sistema durante las pruebas funcionales mencionadas anteriormente. Se comparará el desempeño del proyecto en dos entornos distintos, proporcionando un análisis más exhaustivo del rendimiento. Esta comparación permitirá identificar las diferencias en eficiencia y capacidad de manejo de carga entre los dos sistemas evaluados.
















	Diseño y Pruebas II Acme-Software-Factory
	Testing Report


4. Contenido

Pruebas funcionales

Requirement 8

Para este requisito se han implementado las siguientes pruebas:

- ▼  booking
 -  last-nibble.safe
 -  navigation-atribute.hack
 -  navigation-atribute2.hack
 -  negative-create.safe
 -  negative-publish.safe
 -  negative-update.safe
 -  positive-create.safe
 -  positive-delete.safe
 -  positive-publish.safe
 -  positive-update.safe
 -  wrong-action.hack
 -  wrong-action2.hack
 -  wrong-realm.hack
 -  wrong-user.hack

	Diseño y Pruebas II Acme-Software-Factory
	Testing Report

Last-nibble.safe: Con este test se valida que el campo lastNibble del formulario de reservas solo acepte caracteres numéricos, impidiendo la entrada de letras u otros símbolos.

navigation-attribute.hack: Se comprueba que el sistema reacciona correctamente ante accesos no autorizados. En concreto, se verifica que el formulario de actualización y publicación de reservas no permita seleccionar vuelos que no estén publicados, que ya hayan ocurrido o que no existan.

navigation-attribute2.hack: Similar a la prueba anterior, pero aplicada al formulario de creación de reservas. Se asegura que no sea posible seleccionar vuelos inválidos (no publicados, pasados o inexistentes).

negative-create.safe: Se han probado todos los casos negativos posibles en el formulario de creación de reservas, respetando las restricciones definidas para cada campo, según la metodología establecida.

negative-publish.safe: Se han evaluado todos los escenarios incorrectos al intentar publicar una reserva, verificando que se cumplan las reglas y restricciones del formulario.

negative-update.safe: Se prueban escenarios correctos en el formulario de actualización de reservas, asegurando que todas las restricciones se respeten adecuadamente.

positive-create.safe: Se validan todos los casos de creación de reservas en los que la operación debería ser exitosa, confirmando que se respeten las restricciones del sistema y la lógica de negocio.

positive-delete.safe: Se verifica que es posible eliminar reservas que estén en estado de "borrador".

positive-update.safe: Se prueban escenarios válidos en el formulario de actualización de reservas, asegurando que se cumplan las condiciones establecidas.

wrong-action.hack: Como usuario "customer1", se verifica que no es posible borrar, actualizar ni volver a publicar una reserva que ya ha sido publicada.






wrong-action2.hack: También como "customer1", se comprueba que no se puede publicar ni modificar una reserva si el vuelo asociado tiene una fecha posterior a la actual.

wrong-realm.hack: Se asegura que los usuarios anónimos no puedan realizar ninguna acción sobre las reservas (crear, publicar, actualizar o eliminar), sin importar si están publicadas o no.


wrong-user.hack: Registrado como "customer2", se valida que no es posible manipular reservas que pertenezcan a "customer1".

Además, se ha comprobado de manera manual que los atributos **readOnly** no se pueden modificar, lo que hace imposible realizar un intento de manipulación sobre ellos.

Tras ejecutar el replayer obtenemos la siguiente cobertura.

acme.features.customer.booking		96,3 %
> CustomerBookingDeleteService.java		89,3 %
> CustomerBookingPublishService.java		96,9 %
> CustomerBookingUpdateService.java		96,6 %
> CustomerBookingCreateService.java		97,2 %
> CustomerBookingListService.java		93,3 %
> CustomerBookingShowService.java		98,1 %
> CustomerBookingController.java		100,0 %

Como vemos, la mayoría de la funcionalidad está cubierta, aunque estas excepciones:

	Diseño y Pruebas II Acme-Software-Factory
	Testing Report











- En el unbind del servicio CustomerBookingDeleteService, puesto que el objetivo es eliminar el objeto passenger, no mostrarlo ni preparar datos para una vista.
- En el método onSuccess del servicio CustomerBookingUpdateService, solo se llama a onSuccess() después de un POST en el flujo de update, así que la rama donde no es "POST" nunca ocurre.
- En el unbind del servicio CustomerBookingListService la línea 57 no ha sido probada puesto que los test se han realizado con el sistema configurado en inglés y por lo tanto no ha sido posible realizar un test para cubrir esta línea.

Errores localizados tras la realización de los test:

- Se han corregido los validadores en los métodos create, update y publish, ya que anteriormente no estaban vinculados correctamente a los campos correspondientes del formulario.
- Se ha ajustado el comportamiento de los menús desplegables en el selector de vuelos, con el objetivo de ofrecer una experiencia de usuario más clara e intuitiva.
- Han permitido detectar varios intentos de uso indebido del sistema, como por ejemplo:
 - Selección de vuelos con fecha pasada, inexistente o no publicados desde el selector de vuelos.
 - Envío de datos inválidos en el campo correspondiente al tipo de reserva
- Ha permitido corregir la manipulación de URLs para acceder a identificadores de reservas inexistentes, lo cual anteriormente provocaba errores críticos (tipo PANIC) durante la ejecución del código.


Requirement 9

Para este requisito se han implementado las siguientes pruebas:

 negative-create.safe
 negative-publish.safe
 negative-update.safe
 positive-create.safe
 positive-delete.safe
 positive-publish.safe
 positive-update.safe
 wrong-action.hack
 wrong-realm.hack
 wrong-user.hack

Wrong-Realm.hack: Se comprueba que un usuario anónimo no puede realizar ninguna operación relacionada con los passengers (crear, publicar, actualizar o eliminar), tanto si están publicados como si no lo están.

Wrong-User.hack: Se realiza un registro como "customer2" y se verifica que no es posible operar sobre los passengers pertenecientes a "customer1".

	Diseño y Pruebas II Acme-Software-Factory
	Testing Report

Wrong-Action.hack: Registrado como “customer1”, se comprueba que no es posible borrar, actualizar o volver a publicar un passenger ya publicado.

Negative-Create.safe: Se han evaluado todos los casos negativos posibles en el formulario de creación de un passenger, siguiendo la metodología establecida y considerando las restricciones de cada campo.

Negative-Publish.safe: Se han analizado todos los escenarios negativos posibles al publicar un passenger, de acuerdo con las restricciones de los campos del formulario.

Negative-Update.safe: Se evalúan todos los casos negativos posibles al actualizar un passenger, teniendo en cuenta las restricciones definidas.

Positive-Create.safe: Se comprueban todas las posibilidades positivas en el formulario de creación de un passenger, garantizando que se respete la lógica de negocio y las restricciones de los campos.

















Positive-Delete.safe: Se verifica que es posible eliminar passengers que se encuentren en estado de “borrador”.

Positive-Publish.safe: Se prueban todos los casos positivos para la publicación de un passenger, asegurando que se cumplan las restricciones y reglas establecidas.

Positive-Update.safe: Se evalúan los escenarios positivos en el formulario de actualización de un passenger, respetando las restricciones correspondientes.

En este caso no ha sido necesario realizar test relacionados a read-only-attributes y a navigation-attributes puesto que el formulario de creación, actualización y publicación de pasajeros no cuenta con estos elementos.


Tras ejecutar el replayer obtenemos la siguiente cobertura.

▼  acme.features.customer.passenger	 94,8 %
>  CustomerPassengerDeleteService.java	 88,9 %
>  CustomerPassengerUpdateService.java	 94,1 %
>  CustomerPassengerPublishService.java	 95,8 %
>  CustomerPassengerListService.java	 93,5 %
>  CustomerPassengerCreateService.java	 97,4 %
>  CustomerPassengerShowService.java	 95,8 %
>  CustomerPassengerController.java	 100,0 %


Como vemos, la mayoría de la funcionalidad está cubierta, aunque encontramos tres excepciones:

- En el unbind del servicio CustomerPassengerListService la línea 56 no ha sido probada puesto que los test se han realizado con el sistema configurado en inglés y por lo tanto no ha sido posible realizar un test para cubrir esta línea.
De forma manual se ha comprobado que la traducción se realiza de forma correcta.
- En el unbind del servicio CustomerPassengerDeleteService, puesto que el objetivo es eliminar el objeto passenger, no mostrarlo ni preparar datos para una vista.
- En el método onSuccess del servicio CustomerPassengerUpdateService, solo se llama a onSuccess() después de un POST en el flujo de update, así que la rama donde no es "POST" nunca ocurre.

Errores localizados tras realizar los test:

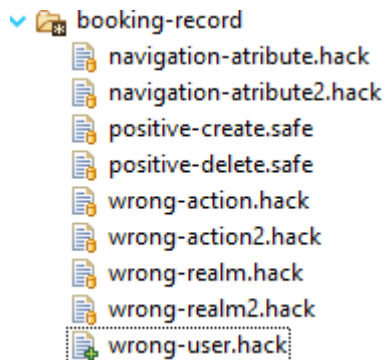
	Diseño y Pruebas II Acme-Software-Factory
	Testing Report

- Se han corregido los validadores en los métodos create, update y publish, ya que anteriormente no estaban vinculados correctamente a los campos correspondientes del formulario.
- Ha permitido corregir la manipulación de URLs para acceder a identificadores de reservas inexistentes, lo cual anteriormente provocaba errores críticos (tipo PANIC) durante la ejecución del código.

	Diseño y Pruebas II Acme-Software-Factory
	Testing Report

De manera adicional tras el análisis de los requisitos #8 y #9 fue necesario la creación de una entidad intermedia booking-record la cual también ha sido probada.

Para esta entidad intermedia se han implementado las siguientes pruebas:



Navigation-atribute.hack: Se verifica que el sistema responde correctamente ante un intento de añadir un pasajero duplicado a una reserva, impidiendo esta acción no válida.

Navigation-atribute2.hack: Se prueba que el sistema bloquea correctamente los intentos de añadir a una reserva un pasajero no publicado o un pasajero que no pertenece al cliente actualmente autenticado.

Positive-create.safe: Se valida que es posible agregar un pasajero a una reserva de manera correcta y conforme a las reglas del sistema.

Positive-delete.safe: Se confirma que el sistema permite eliminar un pasajero de una reserva correctamente.

Wrong-action.hack: Iniciado sesión como customer2, se comprueba que no es posible modificar los pasajeros de una reserva que ya ha sido publicada.













Wrong-action2.hack: Con sesión iniciada como customer2, se verifica que no es posible acceder ni operar sobre una reserva si la fecha del vuelo ya ha pasado.

Wrong-realm.hack: Con sesión iniciada como customer2, se verifica que no es posible acceder ni operar sobre una reserva si la fecha del vuelo ya ha pasado.


Wrong-realm2.hack: Con sesión iniciada como customer2, se verifica que no es posible acceder ni operar sobre una reserva si la fecha del vuelo ya ha pasado.

Wrong-user.hack: Iniciado sesión como customer1, se intenta acceder a la lista de pasajeros de una reserva perteneciente a customer2, comprobando que esta acción no está permitida.

Obteniendo la siguiente cobertura:

✓  acme.features.customer.bookingRecord	 97,9 %
>  CustomerBookingRecordDeleteService.j	 87,4 %
>  CustomerBookingRecordController.java	 100,0 %
>  CustomerBookingRecordCreateService.j	 100,0 %
>  CustomerBookingRecordListService.java	 100,0 %
>  CustomerBookingRecordShowService.je	 100,0 %

Como vemos, la mayoría de la funcionalidad está cubierta, aunque encontramos una excepción:

	Diseño y Pruebas II Acme-Software-Factory
	Testing Report

- En el unbind del servicio CustomerBookinRecordDeleteService, puesto que el objetivo es eliminar el objeto passenger, no mostrarlo ni preparar datos para una vista.

Errores localizados tras realizar los test:

- Se ha corregido que desde el formulario se puedan añadir pasajeros que no pertenecen al cliente que se encuentra registrado.
- Se ha corregido la vulnerabilidad que permitía la manipulación de URLs para acceder a identificadores de reservas inexistentes. Anteriormente, este comportamiento provocaba errores críticos en tiempo de ejecución (tipo PANIC), los cuales han sido eliminados con la validación adecuada de los identificadores.



Pruebas de rendimiento

Las pruebas de rendimiento se realizaron con la herramienta "tester-replayer", que reproduce los casos de prueba grabados para medir el tiempo que se tarda en procesar las peticiones. Las pruebas de rendimiento se ejecutaron dos veces:

1. En mi PC personal.
2. En el PC de mi compañero.

Métricas para el caso 1

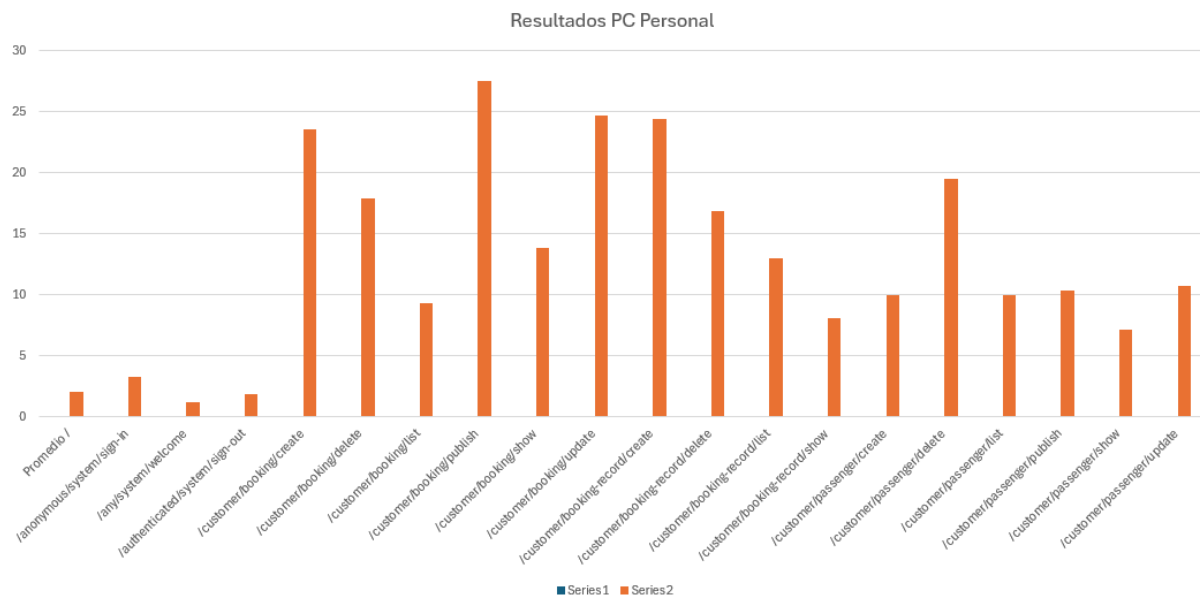
Estos han sido los resultados obtenidos:

Promedio /	2,02627629
/anonymous/system/sign-in	3,27529219
/any/system/welcome	1,21374922
/authenticated/system/sign-out	1,85817419
/customer/booking/create	23,5814262
/customer/booking/delete	17,89547
/customer/booking/list	9,31945116
/customer/booking/publish	27,5445042
/customer/booking/show	13,8500718
/customer/booking/update	24,7369986
/customer/booking-record/create	24,383637
/customer/booking-record/delete	16,8388571
/customer/booking-record/list	13,0250556
/customer/booking-record/show	8,10056667
/customer/passenger/create	9,97748831
/customer/passenger/delete	19,52519
/customer/passenger/list	9,96105
/customer/passenger/publish	10,3657741
/customer/passenger/show	7,10421333
/customer/passenger/update	10,7643325
Promedio general	10,9185453



Diseño y Pruebas II Acme-Software-Factory

Testing Report



También se ha calculado el intervalo de confianza del 95 % para el tiempo que tarda en atender las solicitudes anteriores mi ordenador personal. Obteniendo como resultado:

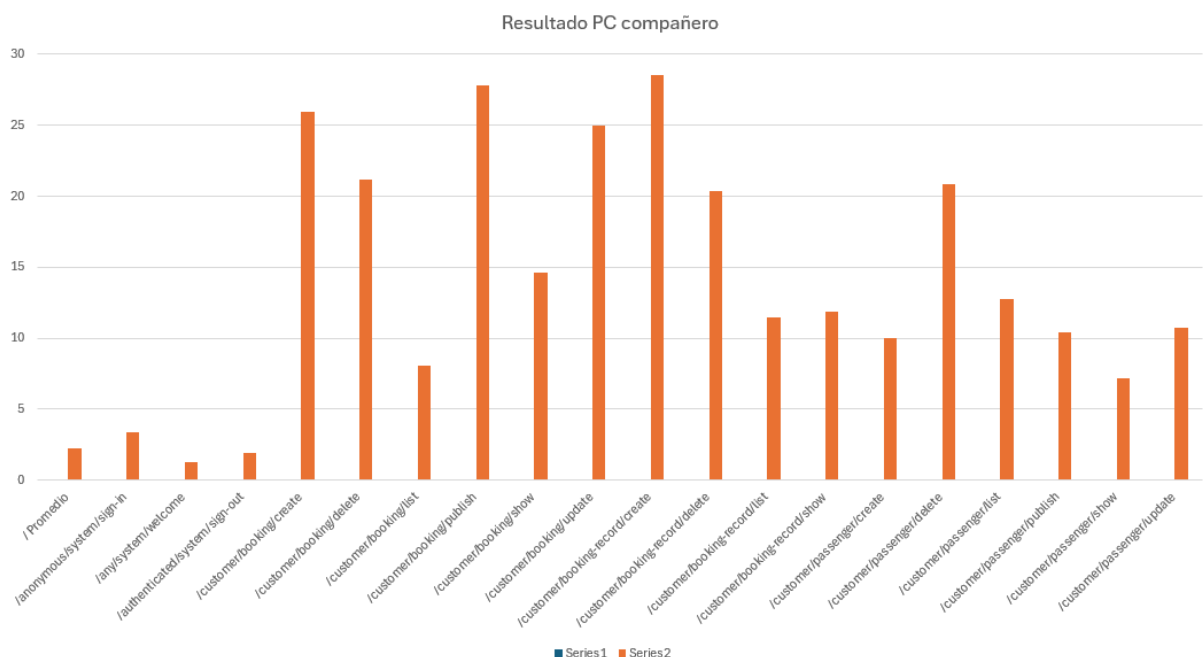
Interval(ms)	10,0890385	11,748052
Interval(s)	0,01008904	0,01174805




Métricas para el caso 2

Estos han sido los resultados obtenidos:

/ Promedio	2,26380309
/anonymous/system/sign-in	3,38305625
/any/system/welcome	1,22915313
/authenticated/system/sign-out	1,93464839
/customer/booking/create	25,9234541
/customer/booking/delete	21,1631
/customer/booking/list	8,09816512
/customer/booking/publish	27,8405141
/customer/booking/show	14,6294436
/customer/booking/update	24,9689086
/customer/booking-record/create	28,5322963
/customer/booking-record/delete	20,3323286
/customer/booking-record/list	11,4927111
/customer/booking-record/show	11,8466
/customer/passenger/create	10,0383597
/customer/passenger/delete	20,85546
/customer/passenger/list	12,7706938
/customer/passenger/publish	10,4419469
/customer/passenger/show	7,14374
/customer/passenger/update	10,7143373
Promedio general	11,3226108



	Diseño y Pruebas II Acme-Software-Factory
	Testing Report

También se ha calculado el intervalo de confianza del 95 % para el tiempo que tarda en atender las solicitudes anteriores el ordenador de mi compañero. Obteniendo como resultado:

Interval(ms)	10,4388061	12,2064155
Interval(s)	0,01043881	0,00122064



Análisis de los resultados obtenidos

Análisis estadístico:

Para validar estadísticamente las mejoras observadas, se calculó un intervalo de confianza del 95% para los tiempos medios de respuesta para los resultados de ambos PC. Además, se realizó un contraste de hipótesis.

MI PC		PC COMPAÑERO	
Media	10,9185453	Media	11,3226108
Error típico	0,42272623	Error típico	0,45039712
Mediana	5,6519	Mediana	5,7004
Moda	1,1917	Moda	1,2502
Desviación estándar	13,5470402	Desviación estándar	14,4338048
Varianza de la muestra	183,522298	Varianza de la muestra	208,334722
Curtosis	13,7680771	Curtosis	15,6047388
Coefficiente de asimetría	3,06482577	Coefficiente de asimetría	3,26988364
Rango	114,0045	Rango	119,0649
Mínimo	0,7281	Mínimo	0,7368
Máximo	114,7326	Máximo	119,8017
Suma	11213,346	Suma	11628,3213
Cuenta	1027	Cuenta	1027
Nivel de confianza(95,0%)	0,82950673	Nivel de confianza(95,0%)	0,88380473



Diseño y Pruebas II
Acme-Software-Factory

Testing Report

Contraste de hipótesis al 95% de confianza:

	<i>MI PC</i>	<i>PC COMPAÑERO</i>
Media	10,91854528	11,32261081
Varianza (conocida)	183,5222978	208,3347217
Observaciones	1027	1027
Diferencia hipotética de las medias	0	
z	-0,654143771	
P(Z<=z) una cola	0,256509592	
Valor crítico de z (una cola)	1,644853627	
Valor crítico de z (dos colas)	0,513019184	
Valor crítico de z (dos colas)	1,959963985	

Ambos PCs ofrecen un rendimiento estadísticamente equivalente en términos de tiempo de ejecución. Aunque mi PC muestra valores ligeramente mejores en promedio y en consistencia, la diferencia no es significativa desde un punto de vista estadístico con un 95% de confianza.




5. Conclusiones

Tras la ejecución de las pruebas funcionales y de rendimiento, se puede concluir que el sistema cumple de manera satisfactoria con los requisitos funcionales especificados para las entidades *booking*, *passenger* y *bookingRecord*. Las pruebas han permitido validar tanto los escenarios positivos como negativos, asegurando que se respetan las restricciones establecidas y que el comportamiento del sistema es coherente con la lógica de negocio esperada.

Además, se identificaron y corrigieron múltiples errores y vulnerabilidades críticas, como validadores mal implementados, problemas en la manipulación de URLs y validaciones insuficientes en los formularios. Estas correcciones mejoraron significativamente la estabilidad y seguridad de la aplicación.

En cuanto al análisis de rendimiento, los resultados obtenidos muestran que no existen diferencias estadísticamente significativas entre los dos entornos de ejecución utilizados. Esto indica que el sistema mantiene un rendimiento consistente, independientemente del entorno, lo cual es un indicador positivo de su robustez y escalabilidad.

En resumen, el proceso de pruebas ha sido efectivo para validar la funcionalidad, detectar errores, y comprobar el rendimiento del sistema, permitiendo así entregar un producto más fiable y preparado para su despliegue.

	Diseño y Pruebas II Acme-Software-Factory
	Testing Report

6. Bibliografía

<https://ev.us.es/>