Universidad de Sevilla  
Escuela Técnica Superior de Ingeniería Informática



Grado en Ingeniería Informática – Ingeniería del Software  
Diseño y Pruebas II

Curso 2023 – 2024

|  |  |
| --- | --- |
| Fecha | Versión |
| 13/02/24 | 1.0 |

|  |  |
| --- | --- |
| Grupo de Prácticas: C1.012 | |
| **Repositorio:** [https://github.com/DP2-C1-012/Acme-SF](mailto:https://github.com/DP2-C1-012/Acme-SF) | |
| Autores por orden alfabético | Correo |
| Bernal Caunedo, Álvaro Jesús | [alvbercau@alum.us.es](mailto:alvbercau@alum.us.es) |
| Caballero Hernández, Jaime | [jaicabher1@alum.us.es](mailto:jaicabher1@alum.us.es) |
| Casamitjana Benítez, Juan José | [juacasben@alum.us.es](mailto:juacasben@alum.us.es) |
| Herrera Lobo, Nicolás | [nicherlob@alum.us.es](mailto:nicherlob@alum.us.es) |
| Montoya Albitres, Ronald Alexander | [ronmonalb@alum.us.es](mailto:ronmonalb@alum.us.es) |

Índice de contenido

[1. Resumen Ejecutivo 3](#_Toc167553628)

[2. Tabla de revisiones 3](#_Toc167553629)

[3. Introducción 3](#_Toc167553630)

[4. Contenido 3](#_Toc167553631)

[4.1. Casos de prueba 3](#_Toc167553632)

[4.1.1. Auditaciones de código 3](#_Toc167553633)

[4.1.2. Registros de auditación 5](#_Toc167553634)

[4.2. Recubrimiento de las pruebas 6](#_Toc167553635)

[4.3. Análisis del rendimiento 6](#_Toc167553636)

[4.3.1. Primera ejecución 6](#_Toc167553637)

[4.3.2. Segunda ejecución 7](#_Toc167553638)

[4.3.3. Análisis 7](#_Toc167553639)

[5. Conclusiones 8](#_Toc167553640)

[6. Bibliografía 8](#_Toc167553641)

# Resumen Ejecutivo

En este informe vamos a explorar el desarrollo de los casos de test, qué casos de prueba se han implementado, el porcentaje de código que prueban y cuál podría ser su efectividad a la hora de detectar errores.

# Tabla de revisiones

|  |  |  |
| --- | --- | --- |
| Fecha | Revisión | Descripción |
| 25/05/2024 | 1.0 | Inicio |
|  |  |  |

# Introducción

A continuación, se va a dividir el documento en dos secciones, en la primera hablaré de los casos de prueba que se han implementado en cada caso, diferenciando entre funcionalidades y caso específico. Y luego hablaré sobre el análisis de rendimiento de la aplicación.

# Contenido

## Casos de prueba

### Auditaciones de código

A la hora de crear auditaciones de código se han buscado primero que se cumplan los casos positivos, en su mayor parte con valores en los límites superior e inferior de cada campo, si los hubiera, y además buscando alcanzar un recubrimiento lo más grande posible en la ejecución del código. Con todo esto iremos caso por caso analizando como de bien se ha conseguido hacer.

**Crear**

En cuanto a los casos de creación se ha considerado comprobar todos los casos definidos en los requisitos, pero especialmente aquellos designados que no se corresponden con una anotación y que, por consiguiente, deben de validarse en el servicio. Es decir, se ha comprobado de que un formulario es incorrecto cuando:

* Se elige un código ya existente o que está mal formado.
* Una fecha de ejecución está en el futuro a la fecha configurada en el entorno de desarrollo como fecha actual.
* Las acciones correctivas están en blanco o contienen más de 100 caracteres.
* El enlace opcional no es una url o contiene más de 255 caracteres.

En cuanto a los casos positivos se ha comprobado que:

* Un código no existe en base de datos y bien formado.
* Una fecha de ejecución correcta en el pasado.
* Para cada valor de los enumerados que se pueden modificar, Type.
* Las acciones correctivas con exactamente 1 caracter y 100 caracteres.
* El enlace vacío o con una url válida.

**Borrar**

En cuanto a las opciones de borrado debido a que no hay ninguna comprobación especial salvo cuando está publicado no se ha hecho ninguna comprobación a parte de casos positivos, eliminando auditaciones de código, con uno, muchos y ningún registro. En cuanto a los tests de hacking se ha intentado hacer una petición get al endpoint de borrado de algunas auditaciones y comprobando que no aparece el botón correspondiente. También se ha intentado mostrar las auditaciones no publicadas y publicadas de otros auditores.

**Actualizar**

Se ha seguido el mismo procedimiento que con los casos de creación ya que no varían en este sentido, salvo en el detalle de que hay que validar que la entidad recupera de base de datos y la que se está modificando son la misma a la hora de comprobar la disponibilidad del código de la entidad. Es decir, hay que guardarla si modificaciones. Se han hecho los mismos casos de hacking que en el delete.

**Publicar**

Durante la publicación se validan todos los campos de la misma forma que en actualización con la añadidura de que hay que comprobar casos positivos para cuando la nota no sea F o F- y negativos para cuando sí. Se han hecho los mismos casos de hacking que en los casos anteriores.

**Mostrar**

En este caso solo se han validad los casos de hacking porque son los únicos relevantes, se ha seguido el mismo método que en casos anteriores.

### Registros de auditación

**Crear**

Se han diseñado casos de prueba que manejen todas las variantes posibles de los campos que lo permiten. En cuanto a los casos negativos:

* Un código ya ocupado o mal formado.
* Un comienzo de período que sea antes de la fecha de ejecución de la auditación de código asignada, incluso si no tiene asignada una auditación de código.
* Un fin de período antes que el comienzo de período.
* Una auditación nula, sin asignar.
* Un link opcional que no sea una url o tenga más de 255 caracteres.

Y en cuanto a los casos positivos:

* Un código válido bien formado y que no está en uso.
* Un comienzo de período posterior a la fecha de ejecución.
* Se prueban todas las variantes posibles de los enumerados.
* Un fin de período posterior a la fecha de comienzo de período.
* Un link vacío o bien formado.

**Borrar**

De la misma forma que las auditaciones, aquí no se valida ningún dato, ya que la entidad se va a borrar y esto no se puede hacer en el caso de que esté publicada. En el caso de que alguien intente hacer una petición get o post esta será denegada en el caso de que la entidad esté publicada, ya que su cambio no se autorizará.

**Actualizar**

Se comprueban los mismo casos de los atributos que en la creación. También se comprueba que se deniegue la autorización si ya está publicado.

**Listar desde auditación**

Se comprueba que se deniegue la autorización si se intenta listar los registros de una auditación que no pertenece al usuario logueado.

**Mostrar**

Se comprueba que se deniegue la autorización si se intenta ver un registro que no pertenece al usuario logueado.

## Recubrimiento de las pruebas

## Análisis del rendimiento

Se ha hecho una ejecución de los tests en la que no se modificó el código y luego otra después de realizar unos cambios mínimos, obteniendo los siguientes resultados:

### Primera ejecución

|  |  |
| --- | --- |
| Media | 130,925022 |
| Error típico | 13,8318537 |
| Mediana | 53,1653 |
| Desviación estándar | 207,477806 |
| Varianza de la muestra | 43047,0399 |
| Curtosis | 14,2274988 |
| Coeficiente de asimetría | 3,37248108 |
| Rango | 1494,1838 |
| Mínimo | 2,4715 |
| Máximo | 1496,6553 |
| Suma | 29458,13 |
| Cuenta | 225 |
| Nivel de confianza(95,0%) | 27,2572026 |

|  |  |  |
| --- | --- | --- |
|  | Inferior | Superior |
| Nivel de confianza | 103,66782 | 158,182225 |
| En segundos | 0,10366782 | 0,15818222 |

### Segunda ejecución

|  |  |
| --- | --- |
| Media | 120,01848 |
| Error típico | 26,1895858 |
| Mediana | 33,5972 |
| Desviación estándar | 392,843788 |
| Varianza de la muestra | 154326,241 |
| Curtosis | 119,804978 |
| Coeficiente de asimetría | 10,2804056 |
| Rango | 5054,3677 |
| Mínimo | 1,8936 |
| Máximo | 5056,2613 |
| Suma | 27004,1579 |
| Cuenta | 225 |
| Nivel de confianza(95,0%) | 51,6094849 |

|  |  |  |
| --- | --- | --- |
|  | Inferior | Superior |
| Nivel de confianza | 68,40899465 | 171,627964 |
| En segundos | 0,068408995 | 0,17162796 |

### Análisis

Estos serían los resultados de los z-test:

|  |  |  |
| --- | --- | --- |
|  | *182,1022* | *218,7401* |
| Media | 118,641917 | 108,136253 |
| Varianza (conocida) | 43047,0399 | 154326,241 |
| Observaciones | 251 | 251 |
| Diferencia hipotética de las medias | 0 |  |
| z | 0,37464175 |  |
| P(Z<=z) una cola | 0,35396346 |  |
| Valor crítico de z (una cola) | 1,64485363 |  |
| Valor crítico de z (dos colas) | 0,70792692 |  |
| Valor crítico de z (dos colas) | 1,95996398 |  |

En cuenta a los resultados podemos comprobar que de primeras no son comparables, porque el p valor no entra dentro del rango de confianza, y además la variabilidad de la segunda prueba es mucho mayor que la de la primera. Esto se puede deber a que el código desarrollado puede tener comportamientos inesperados o que sencillamente se dieran algunos casos de valores extremadamente atípicos.

# Conclusiones

Los casos de prueba son exhaustivos y combinan correctamente todas las posibilidades, de esta forma es más fácil encontrar errores o riesgos potenciales. Además, los resultados del análisis a pesar de no ser concluyentes, pueden indicar algún tipo de inestabilidad en alguna de las partes del sistema, ya sea el código o el hardware, que habría que investigar más a fondo.

# Bibliografía

Intencionalmente en blanco.