INFORME DE PRUEBAS

GRUPO C1.02.03

Autores:

- **Juan Jesús Campos Garrido – juacamgar2@alum.us.es**

Repositorio: https://github.com/Alejandrocg024/Acme-L3-D03

Fecha del documento: 27/04/2023

Tabla de contenidos.

Resumen general.

A lo largo de este documento se han registrado todas las pruebas de nuestra aplicación, pruebas que hemos agrupado en primer lugar sobre la entidad a la que corresponden, también han sido agrupadas según la funcionalidad que comprueban y por si son positivos(comportamiento natural de cualquier usuario con datos validos), negativos(comportamiento natural de cualquier usuario con datos no validos) y hacking(usuarios que intentan hacer acciones ilegales). En total la aplicación tiene 56 pruebas.

Considerando el rendimiento de nuestra aplicación, hemos llegado a la conclusión de que nuestra aplicación de media puede servir cualquier petición en un intervalo de entre 0,00505872 y 0,0064086 segundos.

Tabla de revisión.

Fecha	Versión	Descripción
27/04/2023	0.1	Creación del documento
10/05/2023	0.7	Test suite registrada
15/05/2023	1.0	Documento terminado

Introducción.

Tras haber desarrollado todos los requisitos de nuestra aplicación llega el momento de asegurarnos de que nuestra implementación es correcta y cumple con el nivel de calidad esperado por nuestro cliente, para esto será necesario desarrollar un conjunto de test que comprueben que no hay fallos en la funcionalidad de nuestra aplicación, la idea de estos test es comprobar en primer lugar que la aplicación funciona de la forma esperada con los datos que según la definición de nuestro cliente son considerado correcto, en segundo lugar comprobar que el sistema no tiene problemas cuando un usuario introduce datos que según el cliente son incorrectos, además de comprobar que el sistema sabe diferenciar cuales son los datos correctos y cuales no. En último lugar, comprobamos que haciendo operaciones ilegales interactuando con la url de nuestra aplicación, un cliente no puede realizar acciones para las cuales no está autorizado. Este documento registrará todas las pruebas implementadas para esta labor.

Por otro lado, a parte de que nuestra aplicación funcione correctamente, el cliente espera que este funcionamiento sea con el mínimo coste de tiempo posible, por lo que en este documento también me encargaré de analizar la capacidad de respuesta del sistema, para lo que definiremos un rango de tiempo que nos permitirá asegurar al cliente que de media todas las peticiones que se le hagan al sistema, serán respondidas en un tiempo del intervalo.

Contenido.

Conjunto de pruebas realizadas:

Para los cursos:

List:

- Test100Positive()
  - Descripción: Iniciamos sesión como “lecturer1” y accedemos al listado de sus cursos, una vez en el listado, ordenamos y comprobamos que los cursos que aparecen tienen los datos que esperábamos.
  - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test200Negative()
  - No hay pruebas negativas para los listados.
- Test300Hacking()
  - Descripción: Iniciamos sesión como “administrador” e intentamos acceder a nuestro listado de cursos, como no tenemos el rol de profesor el sistema no debe permitir que accedamos a este listado. Repetimos el proceso como “auditor1”, como “student1”, como “assistant1” y como “company1”, como ninguno de los usuarios tiene el rol de profesor ninguno debe poder acceder al listado.
  - Efectividad: No se detectaron fallos gracias a esta prueba.

Show:

- Test100Positive()
  - Descripción: Iniciamos sesión como “lecturer1” y accedemos al listado de sus cursos, una vez en el listado, ordenamos y pulsamos sobre el primero de todos para poder ver sus datos, una vez en el formulario con los datos del curso, comprobamos que los datos de este son los que esperábamos. Repetimos este con varios cursos.
  - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test200Negative()
  - No hay pruebas negativas a la hora de mostrar un curso.
- Test300Hacking()
  - Descripción: Iniciamos sesión como “administrador” e intentamos mostrar los detalles de los cursos del “lecturer2”, como no tenemos el rol de profesor el sistema no debe permitir que accedamos a este formulario. Repetimos el proceso como “auditor1”, como “student1”, como “assistant1” y como “company1”, como ninguno de los usuarios tiene el rol de profesor ninguno debe poder acceder al formulario.
  - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test301Hacking()

  - Descripción: Iniciamos sesión como ”lecturer1” e intentamos acceder a los detalles de los cursos del “lecturer2” y como no somos los creadores de dichos cursos el sistema no debe permitirnos ver sus detalles.
  - Efectividad: No se detectaron fallos gracias a esta prueba.

Publish:

- Test100Positive()
  - Descripción: Iniciamos sesión como “lecturer1” y accedemos al listado de sus cursos pulsamos sobre un curso que se pueda publicar para acceder a los detalles de este curso, comprobamos que cumple los requisitos para ser publicado, estos requisitos son, que tenga al menos una lección, al menos una lección práctica y que todas las lecciones de este curso estén publicadas. Volvemos a ver los detalles del curso, pulsamos el botón de publicar. Acto seguido volvemos a acceder al formulario con los detalles del curso y comprobamos que ya no está el botón de publicar, por lo que se ha publicado.
  - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test200Negative()
  - Descripción: Iniciamos sesión como “lecturer1” accedemos a un curso sin lecciones, intentamos publicarlo siguiendo el procedimiento descrito en la prueba test100Positive() y comprobamos que efectivamente hay un error y no se puede publicar.
  - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test201Negative()
  - Descripción: Iniciamos sesión como “lecturer1” accedemos a un curso con lecciones, pero cuyas lecciones solo sean solo del tipo “THEORETICAL”, intentamos publicarlo siguiendo el procedimiento descrito en la prueba test100Positive() y comprobamos que efectivamente hay un error y no se puede publicar.
  - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test202Negative()
  - Descripción: Iniciamos sesión como “lecturer1” accedemos a un curso con lecciones, pero cuyas lecciones no estén publicadas, intentamos publicarlo siguiendo el procedimiento descrito en la prueba test100Positive() y comprobamos que efectivamente hay un error y no se puede publicar.
  - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test300Hacking()
  - Descripción: Iniciamos sesión como “administrador” e intentamos publicar los cursos del profesor “lecturer1”, como no tenemos el rol de profesor, no debemos poder publicar un curso. Repetimos el proceso como “auditor1”, como “student1”, como “assistant1” y como “company1”, como ninguno de estos usuarios tiene el rol de profesor ninguno debe poder publicar un curso.

- Test301Hacking()
    - Descripción: Iniciamos sesión como ”lecturer2” e intentamos publicar los cursos de “lecturer1” como no somos los dueños del curso, el sistema no nos debe permitir publicarlo.
    - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test302Hacking ()
    - Descripción: Iniciamos sesión como “lecturer1” accedemos a un curso ya publicado, accedemos a la url para publicarlo y comprobamos que no se puede volver a publicar.
    - Efectividad: No se detectaron fallos gracias a esta prueba.

Delete:

- Test100Positive()
    - Descripción: Iniciamos sesión como “lecturer1” y uno a uno accedemos a los detalles de los cursos de este profesor que no han sido publicados, dentro del formulario de los detalles de los cursos, pulsamos el botón de borrado y comprobamos que no ha habido ningún problema.
    - Efectividad: No se detectaron fallos gracias a esta prueba.
- No hay caso de prueba negativo a la hora de borrar un curso, la única restricción a la hora de borrar un curso es que este no haya sido publicado, pero para intentar borrar un curso ya publicado, hay que hacerlo a partir de la URL intentado hacer hacking.
- Test300Hacking()
    - Descripción: Iniciamos sesión como “administrador” e intentamos borrar los cursos del profesor “lecturer2”, como no tenemos el rol de profesor, no debemos poder borrar cursos. Repetimos el proceso como “auditor1”, como “student1”, como “assistant1” y como “company1”, como ninguno tiene el rol de profesor ninguno debe poder borrar un curso.
    - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test301Hacking()
    - Descripción: Iniciamos sesión como ”lecturer1” e intentamos borrar los cursos del “lecturer2” como no somos los dueños de los cursos, el sistema no nos debe permitir borrarlo.
    - Efectividad: No se detectaron fallos gracias a esta prueba
- Test302Negative()
    - Descripción: Iniciamos sesión como “lecturer1” y uno a uno accedemos a los detalles de los cursos de este profesor que han sido publicados, dentro del formulario con los datos de los cursos, comprobamos que no hay botón para borrarlos, aun así, intentamos borrarlos accediendo a la url de borrado y comprobamos que el sistema no nos permite borrar cursos ya publicados.
    - Efectividad: No se detectaron fallos gracias a esta prueba.

Update:

- Test100Positive()
    - Descripción: Iniciamos sesión como “lecturer1”, accedemos al listado de sus cursos y pulsamos sobre el primero para acceder a sus detalles, una vez en el formulario cambiamos los valores de los datos de este curso por valores que sean válidos. Vamos realizando iteraciones actualizando constantemente el mismo curso y cambiando solo un campo para probar varios datos validos en cada campo. Por ejemplo, en los campos cuya entrada es texto, probamos palabras de longitud 1, de longitud 2, en lenguajes extranjeros, HTML, Javascript, el límite superior de caracteres de ese campo, el límite superior menos uno. Para las monedas probamos los limites inferiores y superiores sin sobrepasarlos y varios tipos de moneda aceptadas por el sistema. Para los enlaces probamos varios tipos de enlace, para el código probamos con dos letras tres números y con una letra y tres números. Todos valores aceptados por el sistema y tras cada modificación comprobamos que los cambios se han guardado adecuadamente.
    - Efectividad: Se detectó un fallo por el cual cuando se creaba un curso sin introducir un código, la aplicación fallaba y saltaba una pantalla de pánico.
- Test200Negative ()
    - Descripción: Iniciamos sesión como “lecturer1, accedemos al listado de sus cursos y pulsamos sobre el primero para acceder a sus detalles, una vez en el formulario cambiamos los valores de los datos de este curso por valores que no sean válidos. Vamos realizando iteraciones actualizando constantemente el mismo curso y cambiando solo un campo para probar que el sistema no deja actualizar un curso si sus datos no cumplen con las restricciones. Algunos de estos datos podrían ser campos vacíos, frases de tamaño superior al límite de caracteres, precios negativos, precios superiores al precio máximo, introducir palabras donde se deben introducir enlaces, tipos de moneda no aceptadas por el sistema. Tras cada intento de modificación comprobamos que el sistema no nos permite guardar un curso con datos que no cumplen las restricciones.
    - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test300Hacking()
    - Descripción: Iniciamos sesión como “administrador” e intentamos actualizar los cursos del profesor “lecturer2”, como no tenemos el rol de profesor, no debemos poder actualizar cursos. Repetimos el proceso como “auditor1”, como “student1”, como “assistant1” y como “company1”, como ninguno tiene el rol de profesor ninguno debe poder actualizar un curso.
    - Efectividad: No se detectaron fallos gracias a esta prueba.

- Test301Hacking()
    - Descripción: Iniciamos sesión como ”lecturer1” e intentamos actualizar los cursos del “lecturer2” como no somos los dueños de los cursos, el sistema no nos debe permitir actualizarlos.
    - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test302Hacking()
    - Descripción: Iniciamos sesión como ”lecturer1” e intentamos actualizar los cursos del “lecturer1” que ya han sido publicados, como ya están publicados, el sistema no nos debe permitir actualizarlos.
    - Efectividad: No se detectaron fallos gracias a esta prueba.

Create:

- Test100Positive()
    - Descripción: Iniciamos sesión como “lecturer1”, accedemos al listado de sus cursos y pulsamos sobre el botón de crear un curso. Vamos creando cursos siempre con los mismos datos excepto uno el cual vamos cambiando para probar varios datos validos en cada campo. Por ejemplo, en los campos cuya entrada es texto, probamos palabras de longitud 1, de longitud 2, en lenguajes extranjeros, HTML, Javascript, el límite superior de caracteres de ese campo, el límite superior menos uno. Para las monedas probamos los limites inferiores y superiores sin sobrepasarlos y varios tipos de moneda aceptadas por el sistema. Para los enlaces probamos varios tipos de enlace, para el código probamos con dos letras tres números y con una letra y tres números. Todos valores aceptados por el sistema y tras crear cada curso accedemos a los detalles de este para comprobar que se ha creado con los datos esperados.
- Test200Negative ()
    - Descripción: Iniciamos sesión como “lecturer1”, accedemos al listado de sus cursos y pulsamos sobre el botón de crear un curso, vamos intentando crear cursos con datos que no sean válidos. Vamos realizando iteraciones siempre con los mismos datos y cambiando solo un campo para probar que el sistema no deja crear un curso si sus datos no cumplen con las restricciones. Algunos de estos datos podrían ser campos vacíos, frases de tamaño superior al límite de caracteres, precios negativos, precios superiores al precio máximo, palabras donde se deben introducir enlaces, monedas no aceptadas por el sistema. Tras cada intento de modificación comprobamos que el sistema no nos permite guardar un curso con datos que no cumple las restricciones.
- Test300Hacking()
    - Descripción: Iniciamos sesión como “administrador” e intentamos acceder al formulario para crear un curso, como no tenemos el rol de profesor, no debemos poder crear cursos. Repetimos el proceso como “auditor1”, como “student1”, como “assistant1” y como “company1”,

como ninguno tiene el rol de profesor ninguno debe poder crear un curso.
    - Efectividad: No se detectaron fallos gracias a esta prueba.

Para las lecciones:

List-all:

- Test100Positive()
    - Descripción: Iniciamos sesión como “lecturer1” y accedemos al listado de todas sus lecciones, una vez en el listado, ordenamos y comprobamos que las lecciones que aparecen tienen los datos que esperábamos.
    - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test200Negative()
    - No hay pruebas negativas para los listados.
- Test300Hacking()
    - Descripción: Iniciamos sesión como “administrador” e intentamos acceder a nuestro listado de lecciones, como no tenemos el rol de profesor el sistema no debe permitir que accedamos a este listado. Repetimos el proceso como “auditor1”, como “student1”, como “assistant1” y como “company1”, como ninguno de los usuarios tiene el rol de profesor ninguno debe poder acceder al listado.
    - Efectividad: No se detectaron fallos gracias a esta prueba.

List:

- Test100Positive()
    - Descripción: Iniciamos sesión como “lecturer1” y accedemos a los detalles de un curso que tenga lecciones asociadas, una vez en el formulario con los datos del curso pulsamos sobre el botón para ver su listado de lecciones y una vez en el listado de lecciones del curso comprobamos que las lecciones que nos aparecen tienen los datos esperados.
    - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test200Negative()
    - No hay pruebas negativas para los listados.
- Test300Hacking()
    - Descripción: Iniciamos sesión como “administrador” e intentamos acceder a nuestro listado de lecciones de un curso del profesor “lecturer1” que tenga lecciones, como no tenemos el rol de profesor el sistema no debe permitir que accedamos a este listado. Repetimos el proceso como “auditor1”, como “student1”, como “assistant1” y como “company1”, como ninguno de los usuarios tiene el rol de profesor ninguno debe poder acceder al listado.
    - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test301Hacking()

  - Descripción: Iniciamos sesión como “lecturer2” e intentamos acceder al listado de lecciones de un curso del profesor “lecturer1”, como no somos el propietario de este curso, el sistema no debe permitirnos acceder al listado.
  - Efectividad: No se detectaron fallos gracias a esta prueba.

Show:

- Test100Positive()
  - Descripción: Iniciamos sesión como “lecturer1” y accedemos al listado de sus lecciones, una vez en el listado, ordenamos y pulsamos sobre una lección para poder ver sus datos, una vez en el formulario con los datos de la lección, comprobamos que los datos de esta son los que esperábamos. Repetimos el proceso para varias lecciones.
  - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test200Negative()
  - No hay pruebas negativas a la hora de mostrar un curso.
- Test300Hacking()
  - Descripción: Iniciamos sesión como “administrador” e intentamos mostrar los detalles de las lecciones del “lecturer2”, como no tenemos el rol de profesor el sistema no debe permitir que accedamos a este formulario. Repetimos el proceso como “auditor1”, como “student1”, como “assistant1” y como “company1”, como ninguno de los usuarios tiene el rol de profesor ninguno debe poder acceder al formulario.
  - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test301Hacking()
  - Descripción: Iniciamos sesión como ”lecturer1” e intentamos acceder a los detalles de las lecciones del profesor “lecturer2” y como no somos los creadores de dichas lecciones no debe permitirnos ver sus detalles.
  - Efectividad: Tras realizar esta prueba se detectó un fallo en la autorización a la hora de ver una lección de la cual no somos los propietarios. Lo esperado es que al acceder a los detalles de una lección de otro profesor el sistema no debería dejarnos, pero con la implementación que teníamos, si nos permitía ver los detalles. Por lo que el fallo fue detectado y corregido.

Publish:

- Test100Positive()
  - Descripción: Iniciamos sesión como “lecturer1” una por una accedemos a los detalles de las lecciones no publicadas, pulsamos en el botón de publicar y comprobamos que han sido publicadas.
  - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test200Negative()
  - No hay pruebas negativas ya que no hay restricciones a la hora de publicar una lección.

- Test300Hacking()
    - Descripción: Iniciamos sesión como “administrador” e intentamos publicar una lección no publicada del profesor “lecturer2”, como no tenemos el rol de profesor, no debemos poder publicar una lección. Repetimos el proceso como “auditor1”, como “student1”, como “assistant1” y como “company1”, como ninguno de estos usuarios tiene el rol de profesor ninguno debe poder publicar una lección.
    - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test301Hacking()
    - Descripción: Iniciamos sesión como “lecturer1” e intentamos volver a publicar todas las lecciones que ya han sido publicadas, el sistema no debe permitirnos.
    - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test302Hacking()
    - Descripción: Iniciamos sesión como ”lecturer1” e intentamos publicar una lección del profesor “lecturer2” como no somos los dueños de la lección, el sistema no nos debe permitir publicarla.
    - Efectividad: No se detectaron fallos gracias a esta prueba.

Delete:

- Test100Positive()
    - Descripción: Iniciamos sesión como “lecturer1” y uno a uno accedemos a los detalles de las lecciones de este profesor que no han sido publicadas, dentro del formulario de los detalles de las lecciones, pulsamos el botón de borrado y comprobamos que no ha habido ningún problema.
    - Efectividad: No se detectaron fallos gracias a esta prueba.
- No hay pruebas negativas ya que no tenemos restricciones que nos impidan borrar lecciones, solo, no se pueden borrar lecciones ya publicadas y eso se comprobará en el hacking.
- Test300Hacking()
    - Descripción: Iniciamos sesión como “administrador” e intentamos borrar una lección del profesor “lecturer2”, como no tenemos el rol de profesor, no debemos poder borrar una lección. Repetimos el proceso como “auditor1”, como “student1”, como “assistant1” y como “company1”, como ninguno tiene el rol de profesor ninguno debe poder borrar una lección.
    - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test301Hacking()
    - Descripción: Iniciamos sesión como “lecturer1” e intentamos borrar todas las lecciones que ya han sido publicadas, y comprobamos que el sistema no nos lo permite.
    - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test302Hacking()

  - Descripción: Iniciamos sesión como ”lecturer1” e intentamos borrar una lección de “lecturer2”, como no somos los dueños de la lección, el sistema no nos debe permitir borrarla.
  - Efectividad: No se detectaron fallos gracias a esta prueba.

Update:

- Test100Positive()
  - Descripción: Iniciamos sesión como “lecturer1, accedemos al listado de sus lecciones y pulsamos sobre la primera para acceder a sus detalles, una vez en el formulario cambiamos los valores de los datos de esta lección por valores que sean válidos. Vamos realizando iteraciones actualizando constantemente la misma lección y cambiando solo un campo para probar varios datos validos en cada campo. Por ejemplo, en los campos cuya entrada es texto, probamos palabras de longitud 1, de longitud 2, en lenguajes extranjeros, HTML, Javascript, el límite superior de caracteres de ese campo, el límite superior menos uno. Para el tiempo estimado ponemos números positivos, inferiores al máximo. Para los enlaces probamos varios tipos de enlace. Todos valores aceptados por el sistema y tras cada modificación comprobamos que los cambios se han guardado adecuadamente.
- Test200Negative ()
  - Descripción: Iniciamos sesión como “lecturer1, accedemos al listado de sus lecciones y pulsamos sobre la primera para acceder a sus detalles, una vez en el formulario cambiamos los valores de los datos de esta lección por valores que no sean válidos. Vamos realizando iteraciones actualizando constantemente la misma lección y cambiando solo un campo para probar que el sistema no deja actualizar una lección si sus datos no cumplen con las restricciones. Algunos de estos datos podrían ser campos vacíos, frases de tamaño superior al límite de caracteres, números negativos como tiempo estimado, tiempos de aprendizaje estimados superiores al máximo, palabras donde se deben introducir enlaces. Tras cada intento de modificación comprobamos que el sistema no nos permite guardar una lección con datos que no cumplen las restricciones.
- Test300Hacking()
  - Descripción: Iniciamos sesión como “administrador” e intentamos actualizar las lecciones del profesor “lecturer2”, como no tenemos el rol de profesor, no debemos poder actualizar lecciones. Repetimos el proceso como “auditor1”, como “student1”, como “assistant1” y como “company1”, como ninguno tiene el rol de profesor ninguno debe poder actualizar una lección.
  - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test301Hacking()

- Descripción: Iniciamos sesión como ”lecturer1” e intentamos actualizar las lecciones del “lecturer2” como no somos los dueños de las lecciones, el sistema no nos debe permitir actualizarlas.
- Efectividad: No se detectaron fallos gracias a esta prueba.

Create:

- Test100Positive()
  - Descripción: Iniciamos sesión como “lecturer1, accedemos al listado de sus lecciones y pulsamos sobre el botón de crear una lección. Vamos creando lecciones siempre con los mismos datos excepto uno el cual vamos cambiando para probar varios datos validos en cada campo, Por ejemplo, en los campos cuya entrada es texto, probamos palabras de longitud 1, de longitud 2, en lenguajes extranjeros, HTML, Javascript, el límite superior de caracteres de ese campo, el límite superior menos uno. Para el tiempo estimado ponemos números positivos, inferiores al máximo. Para los enlaces probamos varios tipos de enlace. Probamos los tres tipos de naturaleza. Todos valores aceptados por el sistema y tras cada modificación comprobamos que las lecciones se han creado con los datos correctos.
  - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test200Negative ()
  - Descripción: Iniciamos sesión como “lecturer1, accedemos al listado de sus lecciones y pulsamos sobre el botón de crear una lección, vamos intentando crear lecciones con datos que no sean válidos. Vamos realizando iteraciones siempre con los mismos datos y cambiando solo un campo para probar que el sistema no deja crear una lección si sus datos no cumplen con las restricciones Algunos de estos datos podrían ser campos vacíos, frases de tamaño superior al límite de caracteres, números negativos como tiempo estimado, tiempos de aprendizaje estimados superiores al máximo, palabras donde se deben introducir enlaces. Tras cada intento de modificación comprobamos que el sistema no nos permite crear una lección con datos que no cumplen las restricciones.
  - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test300Hacking()
  - Descripción: Iniciamos sesión como “administrador” e intentamos acceder al formulario para crear una lección, como no tenemos el rol de profesor, no debemos poder actualizar lecciones. Repetimos el proceso como “auditor1”, como “student1”, como “assistant1” y como “company1”, como ninguno tiene el rol de profesor ninguno debe poder crear una lección.
  - Efectividad: No se detectaron fallos gracias a esta prueba.

Para las añadir o quitar una lección de un curso (CourseLecture):

Añadir(create):

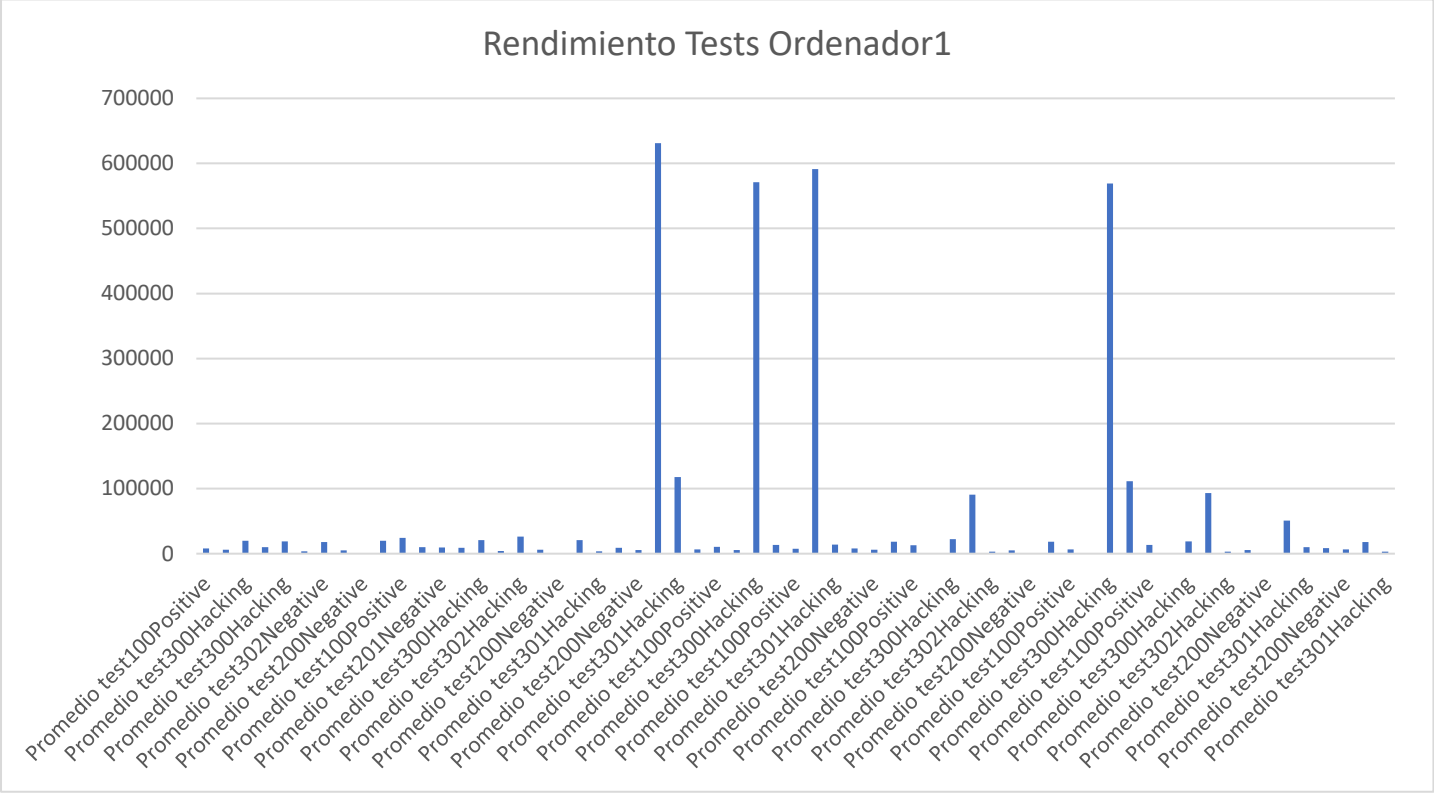
- Test100Positive()
    - Descripción: Iniciamos sesión como “lecturer1” accedemos al listado de sus lecciones y una a una vamos viendo los detalles de las veinte primeras, y en el formulario con los detalles de cada lección, pulsamos sobre el botón añadir la lección a un curso(no publicado), rellenamos el formulario, siempre con la misma información para añadir todas las lecciones al mismo curso, cada vez que añadimos una lección al curso, accedemos al listado de lecciones del curso y comprobamos que la lección se ha añadido correctamente al curso.
    - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test200Negative()
    - Descripción: Tras la prueba anterior, iniciamos sesión como “lecturer1” y repetimos el proceso para intentar volver a añadir las primeras veinte lecciones al mismo curso, pero esta vez el sistema no debe permitirnos añadir dos veces la misma lección al mismo curso.
    - Efectividad: No se detectaron fallos gracias a esta prueba.
- Adicionalmente se intentó desarrollar un test para intentar añadir una lección de profesor “lecturer1” a un curso que no exista o que esté publicado, pero al utilizar un selector para escoger el curso al que se quiere añadir la lección y solo pasarle como opciones al selector los cursos existentes no publicados del profesor logueado, no nos permite seleccionar ni un curso publicado ni un curso que no exista.
- Test300Hacking()
    - Descripción: Sin iniciar sesión intentamos acceder al formulario para añadir las lecciones del profesor “lecturer1” a un curso, al no tener el rol de profesor, el sistema no debe permitirnos acceder al formulario, repetimos el proceso iniciando sesión como “administrator”, como “assistant1”, como “audirtor1”, como “student1” y como “company1” y en ningún caso debe permitirnos.
    - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test301Hacking()
    - Descripción: Iniciamos sesión como “lecturer2” e intentamos acceder al formulario de para añadir una lección del profesor “lecturer1” a un curso, como el profesor “lecturer2” no es propietario de ninguna lección del profesor “lecturer1” no le permite acceder a dicho formulario.
    - Efectividad: No se detectaron fallos gracias a esta prueba.
- Adicionalmente se intentó desarrollar un test para intentar añadir una lección de profesor “lecturer2” a un curso de otro profesor, pero al utilizar un selector para escoger el curso al que se quiere añadir la lección y solo pasarle como opciones al selector los cursos existentes no publicados del profesor logueado, no nos permite seleccionar ni un curso de otro profesor.

Quitar(delete):

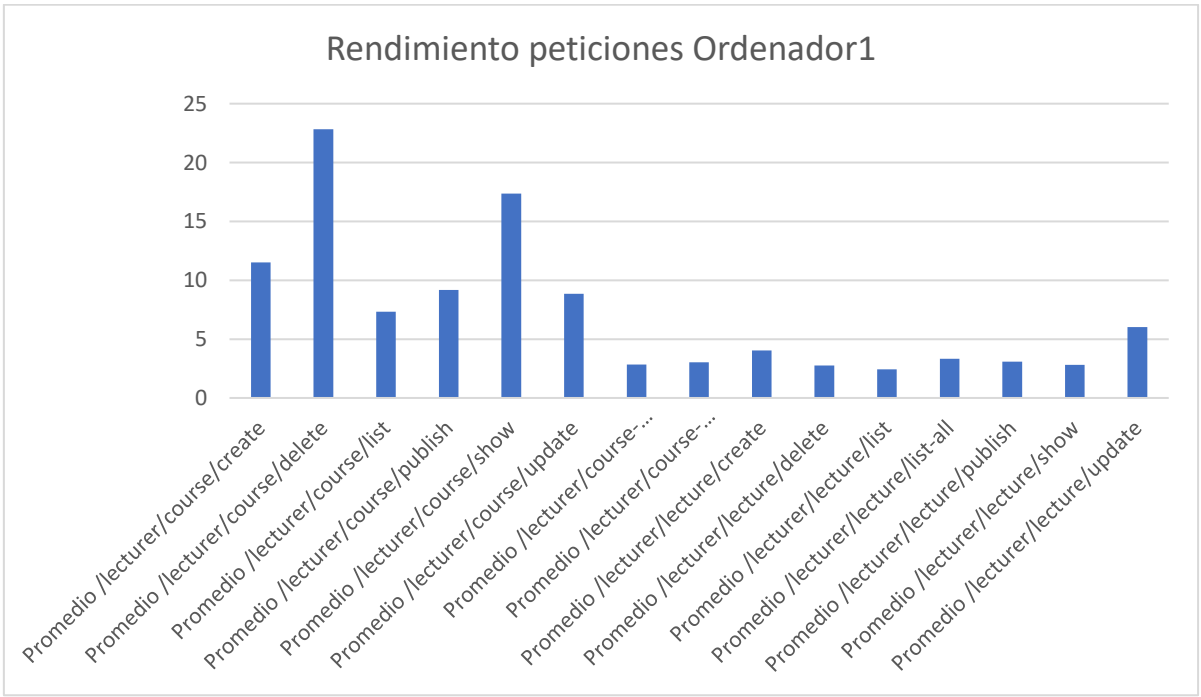
- Test100Positive()
    - Descripción: Iniciamos sesión como “lecturer1” accedemos al listado de sus lecciones y una a una vamos viendo los detalles de las veinte primeras, y en el formulario con los detalles de cada lección, pulsamos sobre el botón quitar la lección de un curso(no publicado), todas estas lecciones estarán incluidas en el mismo curso, así que rellenamos el formulario para quitarlas de este, una vez hallamos quitado todas las lecciones del curso, accedemos a los detalles de este, vemos el listado de sus lecciones y comprobamos que está vacío.
    - Efectividad: No se detectaron fallos gracias a esta prueba.
- Adicionalmente se intentó desarrollar un test negativo para intentar añadir una lección de profesor “lecturer1” de un curso en el cual no se encuentra la lección, pero al utilizar un selector para seleccionar el curso del que se quiere eliminar la lección y solo pasar como opciones al selector los cursos no publicados a los que pertenece esa lección, es imposible que un usuario normal consiga responder al formulario intentando borrar una lección de un curso en el que no estaba ya que no tendrá esa opción, además, si la lección no estaba en ningún curso, el formulario no tendrá botón de “submit” y será solo un formulario de lectura.
- Test300Hacking()
    - Descripción: Sin iniciar sesión intentamos acceder al formulario para quitar las lecciones del profesor “lecturer1” de un curso, al no tener el rol de profesor, el sistema no debe permitirnos acceder al formulario, repetimos el proceso iniciando sesión como “administrator”, como “assistant1”, como “audirtor1”, como “student1” y como “company1” y en ningún caso debe permitirnos.
    - Efectividad: No se detectaron fallos gracias a esta prueba.
- Test301Hacking()
    - Descripción: Iniciamos sesión como “lecturer2” e intentamos acceder al formulario de para quitar una lección del profesor “lecturer1” de un curso, como el profesor “lecturer2” no es propietario de ninguna lección del profesor “lecturer1” no le permite acceder a dicho formulario.
    - Efectividad: No se detectaron fallos gracias a esta prueba.

Una vez comentadas las distintas pruebas del sistema, es el momento de ver cual es el rendimiento de este mismo y de sus pruebas, para esta labor analizaremos distintos datos sobre las pruebas y sobre las peticiones realizadas al sistema mientras que estas se ejecutan. Analizaremos el rendimiento del sistema en dos sistemas con grandes diferencias hardware y compararemos los resultados:

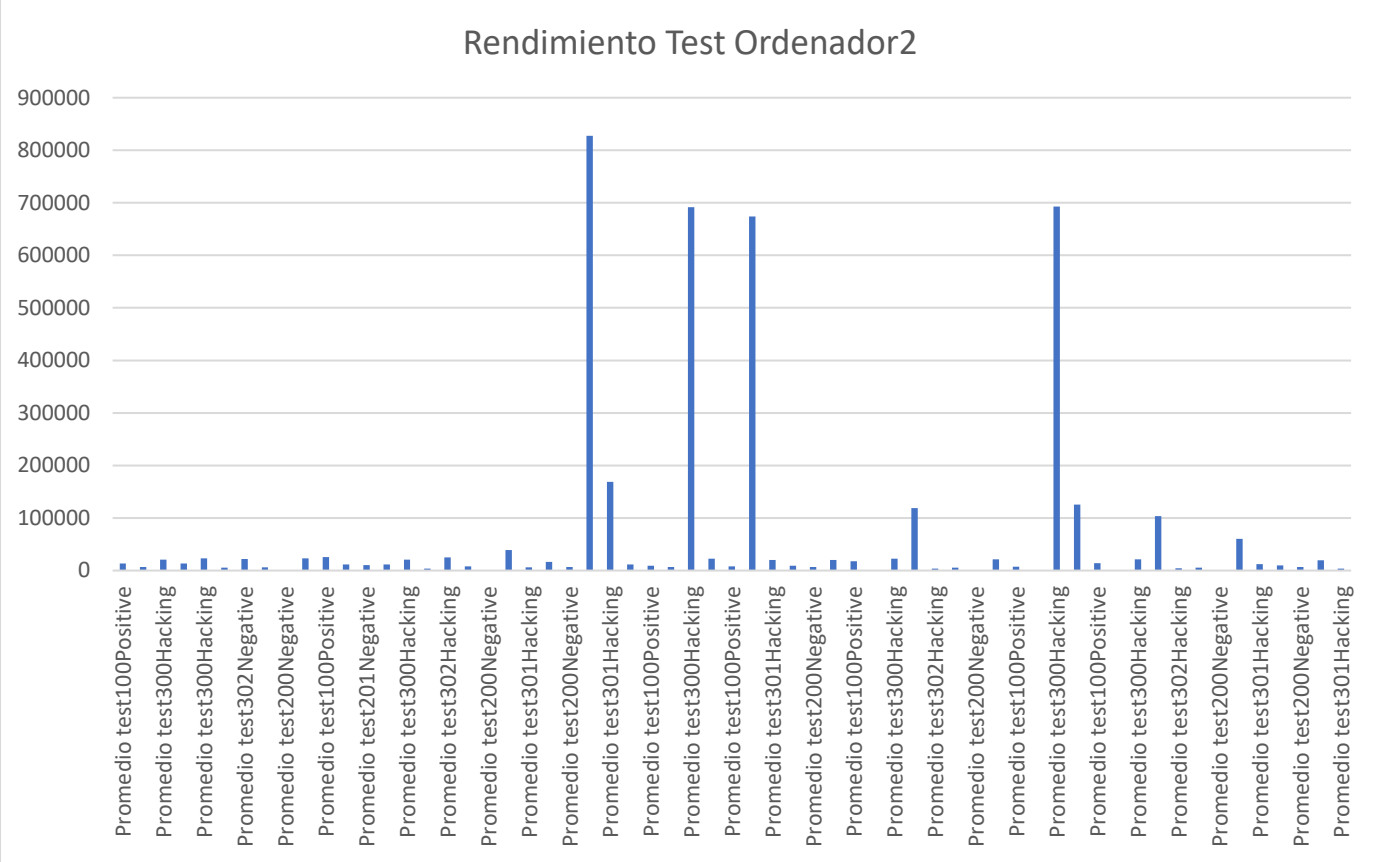
En el primer ordenador cuyos componentes principales son, un procesador AMD Ryzen 7 5800H de 3.2GHz y 16 Gigas de RAM, el tiempo que ha tardado en ejecutarse cada prueba puede verse en la siguiente gráfica:



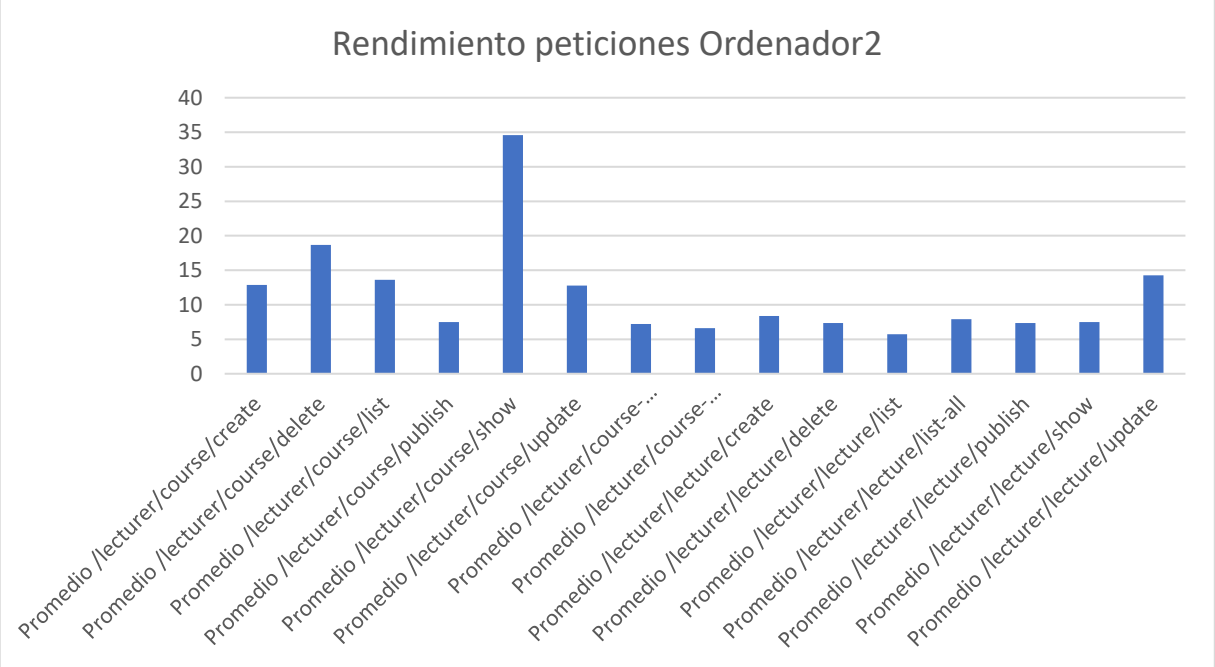
En cuanto a la velocidad con la que se han servido las distintas peticiones en este ordenador, queda la siguiente gráfica:



El segundo ordenador tiene como componentes principales, un procesador Intel(R) Core(TM) i5-10210U CPU 2.10 GHz y 8 Gigas de RAM, el tiempo que ha tardado en ejecutarse cada prueba puede verse en la siguiente gráfica:



En cuanto a la velocidad con la que se han servido las distintas peticiones en este ordenador, queda la siguiente gráfica:



Una vez comentado esto deberíamos hacer los cálculos para poder definir un rango de confianza en el cual podamos asegurar a nuestro cliente que de media nuestra aplicación servirá todas las peticiones en un tiempo dentro de ese rango, para calcular este rango hemos utilizado la siguiente tabla de Excel:

Columna1	
Media	5,73366075
Error típico	0,34420754
Mediana	3
Moda	2
Desviación e	17,7091351
Varianza de l	313,613466
Curtosis	281,569414
Coeficiente d	15,237546
Rango	445
Mínimo	0
Máximo	445
Suma	15177
Cuenta	2647
Nivel de conf	0,67494312

Intervalo(ms)	6,40860387	5,05871763
Intervalo(s)	0,0064086	0,00505872

Dicho esto, podemos asegurar a nuestro cliente que de media todas las peticiones realizadas al sistema se servirán en un intervalo de entre 0,00505872 y 0,0064086 segundos.

Conclusiones.

Llegados a este punto tenemos que analizar cual a sido la efectividad final que han tenido las pruebas de nuestra aplicación, gracias a las pruebas hemos podido detectar dos fallos que pueden ser considerados cruciales en la aplicación.

En primer lugar, cuando un profesor accedía a un curso del cual no era propietario, un fallo en la autorización le permitía acceder a los detalles del curso sin ningún problema.

En segundo lugar, cuando un profesor intentaba actualizar un curso no publicado, introduciendo un código vacío, la aplicación fallaba dando lugar a una pantalla de pánico.

Finalmente, ambos errores han sido corregidos.

Además, como hemos demostrado a lo largo de este documento, podemos asegurar a nuestros clientes que absolutamente todas las peticiones que hagan a nuestra aplicación, será resueltas en un tiempo inferior a un segundo.

Bibliografía.

Intencionalmente vacío.