Universidad de Castilla-La Mancha

Plan de Pruebas: LosKelvinSoftware

Versión 1.4

Histórico de revisiones

Fecha	Versión	Descripción	Autor
13/12/2025	1.1	Desarrollo de los puntos 1 al 3	Juan Pedro Serrano Garvi
13/12/2025	1.2	Desarrollo de los puntos 4 al 6	Telmo García Cho
14/12/2025	1.3	Desarrollo de los puntos 7 al 9	Javier Mellado Montejano
15/12/2025	1.4	Desarrollo de los puntos 10 al 12	Javier Saelices López-Gasco

Índice

Plan de Pruebas

1. Introducción

En este documento se describirá el plan de pruebas del proyecto de la asignatura Ingeniería del Software II del grupo LosKelvinSoftware. Su objetivo será definir todos los aspectos relacionados con todas las pruebas, desde las funcionales hasta las no funcionales.

1.1 Objetivos

Los objetivos principales de este plan de pruebas son los siguientes:

- Verificar que las funcionalidades implementadas cumplan con los requisitos que se nos piden.
- Detectar defectos tanto del backend (API) como del frontend (interfaz web).
- Asegurar que el software sea estable, seguro y que tiene un rendimiento adecuado.

1.2 Ámbito

Como hemos comentado anteriormente, en este plan de pruebas se cubren las pruebas sobre la API y sobre la interfaz web, yendo así desde las pruebas de unidad, específicas de cada funcionalidad implementada, hasta las pruebas funcionales, que cubren las acciones que el usuario realizará en el sistema, siendo así estas mucho más generales.

1.3 Referencias

Para el desarrollo del presente documento nos hemos basado en los siguientes recursos y documentos:

- IEEE 829 Format, Test Plan Outline
- Repositorio del proyecto de ISO II del grupo LosKelvinSoftware

2. Elementos de prueba

Con respecto a los componentes software, vamos a especificar todos aquellos que requieren ser probados, ya sea porque son el resultado de la implementación de una función principal del sistema, o porque conforman en mayor o menor medida una función mayor.

- Por un lado, tenemos la API REST, desarrollada en .NET, en la cual se encapsula la lógica de negocio, yendo desde los controladores hasta los DTOs pasando por la definición de la base de datos.
- Una interfaz Web, a través de la cual el usuario podrá interactuar con el sistema.

3. Características que se deben probar

Las características del proyecto que debemos probar, en función del documento de casos de uso sobre el que hemos basado el desarrollo del proyecto, son las siguientes:

- Compra de herramientas, desde la búsqueda de estas, la gestión del carrito, la intatos y la confirmación, siguiendo además todos sus flujos alternativos (CU 1).
- Reparación de herramientas, desde la selección, siguiendo con la introducción de datos, validandose las fechas, hasta su confirmación, todos ellos con sus correspondientes flujos alternativos (CU 2).
- Creación de ofertas por parte del administrador, partiendo desde la selección hasta la finalización con sus flujos alternativos (CU 3).
- Alquiler de herramientas, desde la selección, pasando por la comprobación de disponibilidad e introducción de datos y terminando en la confirmación, pasando por sus flujos alternativos correspondientes (CU 4).
- Como aspectos generales de todos los casos de uso, podemos destacar la validación de datos, así como otros aspectos como la introducción de fechas válidas o la existencia de stock. También se debe probar la gestión de errores, comprobandose

así que estos surgen cuando deben surgir y que comunican al usuario de manera correcta la información.

4. Características que no se van a probar

Desde el punto de vista de los casos de uso, no se probarán las siguientes funcionalidades:

- La gestión avanzada de usuarios. Es decir, no se comprobará que el inicio de sesión de un usuario corresponda a un rol concreto, como la precondición de que el usuario se loguee como administrador en el CU-Crear Ofertas. Esta funcionalidad de inicio de sesión con un rol no está incluida en el software actual por falta de tiempo y recursos, además de no ser una condición prioritaria de momento.
- Tampoco se probará que el sistema guarde el progreso de un usuario en cada caso de uso en caso del que el sistema caiga, al ser una funcionalidad difícil de implementar y por falta de tiempo y recursos.

5. Aproximación

Para probar el software, se usará Selenium para las pruebas funcionales y Jmeter para las pruebas de rendimiento y carga.

5.1 Pruebas funcionales

Cada miembro del equipo de desarrollo implementará sus pruebas funcionales para su Caso de Uso concreto. Para ello, utilizaremos la herramienta Selenium, un software automatizado de pruebas que rellenará los campos de los formularios de forma automática y comparará los resultados obtenidos con los esperados.

5.2 Pruebas de carga

Cada miembro del equipo implementará dos pruebas de carga para su caso de uso, una sin filtros y otra con filtros. En ambas, se someterá al sistema a peticiones de múltiples hilos para asegurar que el sistema se comporta como es debido con este estrés.

5.3 Pruebas de rendimiento

Para el mismo fin, se usará también Jmeter con el propósito de comprobar que el sistema proporciona las respuestas en el límite establecido de tiempo.

5.4 Pruebas de seguridad

Como este software no tiene exactamente un propósito comercial, no se someterá a pruebas de seguridad. Al no tener un dominio en la red ni ser accesible por cualquier equipo de Internet, no es necesario comprobar sus vulnerabilidades ni prepararlo frente a ataques.

6. Criterios para decidir si pasa/falla un elemento

Para decidir si pasa o falla un elemento del software, se valorará que se comporte de acuerdo con los flujos básicos y alternativos de los casos de uso. Esto significa que debe cumplir su función para que se cumplan todos los flujos, o al menos el que está relacionado con el mismo elemento. Con lo cual, si responde con unos resultados diferentes a los especificados en un caso de uso, ha fallado.

7. Criterios de suspensión y reanudación

7.1 Suspensión

Por ejemplo, las pruebas se detendrán si la API no se levanta o si la base de datos no es accesible o está vacía, impidiendo cualquier prueba funcional.
Este error de base de datos también se puede dar si existe algún fallo en el script sql que genera los datos

7.2 Reanudación

Se reanudarán una vez el entorno de desarrollo (Visual Studio/SQL Server) esté estable y la API responda a swagger/index.html y podamos acceder a la página de inicio de la WEB.

8. Pruebas que se deben entregar

- Plan de Pruebas (este documento).
- Especificación de Casos de Prueba (documentos individuales por CU).
- Código de pruebas automatizadas (.UT y .UIT).

9. Tareas de prueba

- Configuración del Entorno: Despliegue de la base de datos y ejecución de migraciones.
- Inserción de datos en el sistema (Ejecutar script sql que rellena la base de datos).
- Desarollo de los Test, tanto de API como de UI (Pruebas unitarias y de Selenium)
- Ejecución: Puede realizarse una ejecución manual (accediendo a la UI/API directamente) o automática (ejecutando pruebas)

10. Necesidades ambientales

10.1 Hardware

- Equipo estándar (Procesador i5, 16GB RAM)

10.2 Software

- Visual Studio 2022
- Framework: .Net 8.0.
- Navegador: Google Chrome (Version 143.0.7499.41)

10.3 Seguridad

Ningún tipo de seguridad, se encuentra público el repositorio.

10.4 Herramientas

- XUnit: Framework de pruebas unitarias

- Selenium WebDriver: Automatización de navegador
- Swagger para pruebas manuales de la API

10.5 Publicaciones

- Los resultados se publicarán en la rama main del repositorio del proyecto

11. Responsabilidades.

- Desarrolladores (Todos): Responsables de crear y mantener los test unitarios (.UT) y corregir los bugs detectados sobre los controllers y los DTOs.
- Testers/QA (Todos): Responsables de diseñar los casos de prueba y ejecutar los tests de interfaz (.UIT) y mantener este documento.
- ScrumMaster (Javier Mellado): siempre al día con las pull y informando con lo demás temas.

12. Aprobación

En cada Sprint, para la revisión de cambios se ha requerido una segunda persona, Scrum Master, y en caso de que fuese él quien solicitara los cambios, una persona aleatoria. Con respecto a la implementación de los cambios, se ha seguido el mismo procedimiento. Sólo el Scrum Master aceptaría las Pull-Requests, mientras que, si fuesen suyas, las aceptaría una segunda persona.