**Plan de Pruebas para Inventory Motor**

**1. Introducción**

**1.1 Propósito**

Este documento describe la estrategia de pruebas para el sistema de gestión de inventario para la empresa Moto Repuestos Jors, el documento incluye las pruebas que validarán la funcionalidad y estabilidad del sistema.

**1.2 Alcance**

Este plan de pruebas abarca la validación de todas las funcionalidades relacionadas con la gestión de inventario, la gestión de pedidos, la gestión de ventas, la gestión de clientes y la gestión de proveedores. Dentro de estos módulos es incluida la validación para la creación, actualización y eliminación.

También se cubrirán la autenticación y validación de usuarios con un control de acceso basado en roles. Finalmente se validará la interacción entre el frontend, el backend y Las pruebas de integración con la base de datos MySQL.

**1.3 Objetivos de Pruebas**

* Verificar la funcionalidad del software.
* Validar que el software cumple con los requisitos especificados.
* Identificar y corregir defectos antes del lanzamiento.

**2. Estrategia de Pruebas**

**2.1 Enfoque de Pruebas**

* Enfoque Unitario: Validar que cada función y componente individual, tanto en el backend como en el frontend funcione correctamente.
* Enfoque de Integración: Garantizar que los diferentes módulos (frontend, backend y base de datos MySQL) interactúan correctamente. También hay que garantizar que las capas internas del backend puedan interactúen correctamente.
* Enfoque Funcional: Comprobar que las funcionalidades desarrolladas en cada historia de usuario cumplen con los requisitos definidos.

**2.2 Recursos de Pruebas**

* Equipo de Pruebas: En lugar de un equipo de QA separado, los desarrolladores también se encargarán de las pruebas.
* Desarrollador backend: responsable del backend (Spring Boot). Este desarrollador se encargará de las pruebas unitarias del backend y las pruebas de integración internas de las capas del backend además de con la base de datos MySQL.
* Desarrollador frontend: responsable del frontend (React). Este desarrollador realizará pruebas unitarias en el frontend.
* Ambiente de pruebas: Todo el desarrollo y las pruebas se ejecutarán en máquinas locales. Cada miembro del equipo debe tener su entorno de desarrollo configurado con todas las herramientas necesarias (IDE, servidor de base de datos, etc.).
  + **Backend**: Spring Boot ejecutándose localmente, utilizando un servidor como Tomcat embebido que viene con Spring Boot.
  + **Base de datos**: **MySQL** instalado localmente, donde todos los datos de prueba se gestionarán de forma local.
  + **Frontend**: Aplicación de **React** ejecutándose en el servidor local (puerto de desarrollo predeterminado: 3000).
  + **Git**: se usa como sistema de control de versiones. Para trabajar de forma coordinada y compartir el progreso sin problemas. También se utilizar un repositorio remoto en GitHub para mantener el código centralizado y versionado.
* Herramientas de pruebas
* JUnit: Framework para escribir y ejecutar pruebas unitarias del backend, sirve para escribir pruebas unitarias para los controladores, servicios y repositorios.
* Mockito: Herramienta para simular dependencias en las pruebas unitarias, permitiendo verificar que los servicios se comportan como se espera sin necesidad de usar una base de datos real.
* Jest: Framework de pruebas para JavaScript, ideal para React. Jest permite ejecutar pruebas unitarias en los componentes de React y verificar su comportamiento.
* React Testing Library: Extensión de Jest, diseñada para probar los componentes de React de una manera que simule la interacción del usuario con la UI.
* Postman: Herramienta para probar la API RESTful. Postman se puede usar para realizar solicitudes HTTP y verificar que los endpoints del backend están respondiendo correctamente (GET, POST, PUT, DELETE).
* Selenium: para realizar pruebas automatizadas del frontend, Selenium es una opción gratuita que permite simular interacciones de usuarios (clicks, formularios, navegación) y verificar que la aplicación React responde correctamente.

**2.3 Planificación de Pruebas**

Las pruebas se realizan dentro de cada sprint de 2 semanas.

* Cada sprint incluye pruebas para las funcionalidades desarrolladas para asegurar que los módulos previos siguen funcionando correctamente.

**2.4 Criterios de Aceptación**

* Las historias de usuario no se consideran "hechas" hasta que pasan las pruebas unitarias, de integración y funcionales.
* Pruebas unitarias: Al menos el 80% del código de cada clase (servicios, controladores, repositorios) debe estar cubierto por pruebas unitarias. Cada función importante debe tener pruebas que validen tanto los casos de éxito como los casos de error.
* Pruebas de integración: La interacción entre diferentes componentes (servicios, controladores, repositorios y la base de datos) debe estar cubierta por pruebas de integración. Se deben probar los casos críticos de uso, como el proceso completo de creación, consulta y eliminación de datos.
* Pruebas funcionales: Las funcionalidades clave (gestión de productos, pedidos, ventas, autenticación, etc.) deben probarse manualmente o con herramientas automatizadas para verificar que cumplen con los requisitos de negocio.

**3. Casos de Prueba**

**3.1 Identificación de Casos de Prueba**

Los casos de prueba estarán alineados con las historias de usuario planificadas para cada sprint. Por ejemplo:

* Gestión de productos (añadir, eliminar, actualizar).
* Gestión de pedidos (añadir, eliminar, actualizar).
* Gestión de ventas (añadir, eliminar, actualizar).
* Gestión de proveedores (añadir, eliminar, actualizar).
* Gestión de clientes (añadir, eliminar, actualizar).
* Autenticación de usuario.
* Permisos de usuario.
* Pruebas de integración (Frontend y Backend).

**3.2 Descripción de Casos de Prueba**

|  |  |
| --- | --- |
| Identificador del caso de prueba | CT-001 |
| Descripción del caso de prueba | Validar que el sistema permita crear un nuevo producto en la tienda con los datos validos |
| Precondiciones | * El servicio de productos está activo y la base de datos está disponible. * El usuario tiene permisos de administrador. |
| Datos de entrada | * Nombre del producto: Llanta de motocicleta * Descripción: Llanta 90/90-18 para moto * Categoría: Accesorios * Precio: 150000 * Stock: 4 |
| Pasos para la ejecución | 1. realizar una solicitud POST al endpoint /products/save con los datos de entrada en el cuerpo de la solicitud 2. Verificar la respuesta del servidor. 3. Comprobar que el producto ha sido almacenado correctamente en la base de datos. 4. Verificar que el sistema devuelva el producto creado con un ID generado automáticamente. |
| Resultados esperados | * La respuesta del servidor debe ser 201 Created. * El producto debe aparecer en la base de datos con un ID único. * El sistema debe devolver los detalles del producto creado, incluyendo su ID. |
| Resultados reales | * Código de estado HTTP: 201 Created * Producto creado con éxito en la base de datos. * ID del producto: 1. |
| Estado de la prueba | Pasado |

|  |  |
| --- | --- |
| Identificador del caso de prueba | CT-002 |
| Descripción del caso de prueba | Validar que el sistema permita eliminar un producto existente. |
| Precondiciones | * El producto ya existe en la base de datos. * El usuario tiene permisos de administrador. |
| Datos de entrada | * ID del producto: 1 |
| Pasos para la ejecución | 1. Realizar una solicitud DELETE al endpoint /products/delete/1. 2. Verificar que la respuesta del servidor sea exitosa. |
| Resultados esperados | * La respuesta del servidor debe ser 200 OK. * El producto debe eliminarse correctamente de la base de datos. |
| Resultados reales | * Código de estado HTTP: 200 OK. * Producto eliminado de la base de datos. |
| Estado de la prueba | Pasado |

|  |  |
| --- | --- |
| Identificador del caso de prueba | CT-003 |
| Descripción del caso de prueba | Validar que el sistema permita actualizar un producto existente con datos válidos. |
| Precondiciones | * El producto ya existe en la base de datos. * El usuario tiene permisos de administrador. |
| Datos de entrada | * ID del producto: 1 * Nombre: Filtro de aceite actualizado * Precio: 60.000 |
| Pasos para la ejecución | 1. Realizar una solicitud PUT al endpoint /products/update con los datos actualizados. 2. Verificar la respuesta del servidor. 3. Comprobar que los datos del producto han sido actualizados correctamente en la base de datos. |
| Resultados esperados | * La respuesta del servidor debe ser 200 OK. * El producto debe actualizarse correctamente en la base de datos. |
| Resultados reales | * Código de estado HTTP: 200 OK. * Producto actualizado en la base de datos. |
| Estado de la prueba | Pasado |

|  |  |
| --- | --- |
| Identificador del caso de prueba | CT-004 |
| Descripción del caso de prueba | Validar que el sistema permita crear un nuevo pedido con los datos válidos. |
| Precondiciones | * El servicio de pedidos está activo y la base de datos disponible. * El usuario tiene permisos para gestionar pedidos. |
| Datos de entrada | * Cliente ID: 5 * Lista de productos: [ID: 1, cantidad: 2] * Total: 120.000 |
| Pasos para la ejecución | 1. Realizar una solicitud POST al endpoint /orders/save con los datos del pedido. 2. Verificar la respuesta del servidor. 3. Comprobar que el pedido ha sido almacenado en la base de datos. |
| Resultados esperados | * La respuesta del servidor debe ser 201 Created. * El pedido debe almacenarse correctamente en la base de datos. |
| Resultados reales | * Código de estado HTTP: 201 Created. * Pedido almacenado con éxito en la base de datos. |
| Estado de la prueba | Pasado |

|  |  |
| --- | --- |
| Identificador del caso de prueba | CT-005 |
| Descripción del caso de prueba | Validar que el sistema permita eliminar un pedido existente. |
| Precondiciones | * El pedido ya existe en la base de datos. * El usuario tiene permisos para gestionar pedidos. |
| Datos de entrada | * ID del pedido: 2 |
| Pasos para la ejecución | 1. Realizar una solicitud DELETE al endpoint /orders/delete/2. 2. Verificar la respuesta del servidor. |
| Resultados esperados | * La respuesta del servidor debe ser 200 OK. * El pedido debe eliminarse correctamente de la base de datos. |
| Resultados reales | * Código de estado HTTP: 200 OK. * Pedido eliminado de la base de datos. |
| Estado de la prueba | No pasada |

|  |  |
| --- | --- |
| Identificador del caso de prueba | CT-006 |
| Descripción del caso de prueba | Validar que el sistema permita actualizar un pedido existente. |
| Precondiciones | * El pedido ya existe en la base de datos. * El usuario tiene permisos para gestionar pedidos. |
| Datos de entrada | * Cliente ID: 5 * Lista de productos: [ID: 1, cantidad: 2] * Total: 120.000 |
| Pasos para la ejecución | 1. Realizar una solicitud PUT al endpoint /orders/update/2. 2. Verificar la respuesta del servidor. |
| Resultados esperados | * La respuesta del servidor debe ser 200 OK. * El pedido debe eliminarse correctamente de la base de datos. |
| Resultados reales | * Código de estado HTTP: 200 OK. * Pedido no actualizado de la base de datos. |
| Estado de la prueba | No Pasado |

|  |  |
| --- | --- |
| Identificador del caso de prueba | CT-007 |
| Descripción del caso de prueba | Validar que el usuario pueda autenticarse correctamente con credenciales válidas y reciba un token JWT. |
| Precondiciones | * El servicio de autenticación está activo y funcionando correctamente. * El usuario existe en el sistema con las credenciales válidas. |
| Datos de entrada | * Nombre de usuario: Jorge * Contraseña: admin |
| Pasos para la ejecución | 1. Realizar una solicitud POST al endpoint /auth/login con las credenciales de inicio de sesión. 2. Verificar la respuesta del servidor. 3. Comprobar que se devuelve un token JWT en el encabezado Authorization. |
| Resultados esperados | * La respuesta del servidor debe ser 200 OK. * El token JWT debe aparecer en el encabezado Authorization. |
| Resultados reales | * Código de estado HTTP: 200 OK. * Token JWT generado con éxito. |
| Estado de la prueba | Pasado |

|  |  |
| --- | --- |
| Identificador del caso de prueba | CT-008 |
| Descripción del caso de prueba | Validar que el administrador pueda actualizar ventas y pedidos existentes con datos válidos. |
| Precondiciones | * El servicio de autenticación está activo y funcionando correctamente. * El usuario tiene permisos de administrador. * El pedido o la venta a actualizar existe en la base de datos. |
| Datos de entrada | * Nombre de usuario: Jorge * Contraseña: admin * Datos del pedido: * ID del pedido: 2 * Cliente ID: 5 * Lista de productos: [ID: 1, cantidad: 3] * Total: 180,000 |
| Pasos para la ejecución | * Realizar una solicitud **POST** al endpoint /auth/login con las credenciales de inicio de sesión para obtener un token JWT. * Verificar la respuesta del servidor y extraer el token JWT del encabezado Authorization. * Realizar una solicitud **PUT** al endpoint /orders/update/2 con los datos de entrada en el cuerpo de la solicitud y el token JWT en el encabezado. * Verificar la respuesta del servidor. * Comprobar que los datos del pedido han sido actualizados correctamente en la base de datos. |
| Resultados esperados | * La respuesta del servidor para el login debe ser 200 OK, y el token JWT debe generarse correctamente. * La respuesta del servidor para la actualización del pedido debe ser 200 OK. * Los datos actualizados del pedido deben reflejarse correctamente en la base de datos. |
| Resultados reales | * Código de estado HTTP (login): 200 OK. * Token JWT generado con éxito. * Código de estado HTTP (actualización): 404 OK. * Pedido no actualizado en la base de datos. |
| Estado de la prueba | No Pasado |

**4. Ejecución de Pruebas**

**4.1 Procedimientos de Ejecución**

* Las pruebas unitarias se ejecutan continuamente como parte del flujo de trabajo de los desarrolladores.
* Las pruebas de integración y funcionales se ejecutarán durante cada sprint para validar las historias de usuario desarrolladas.

**4.2 Registro de Defectos**

Cualquier defecto encontrado se registra en el **backlog** utilizando herramientas como **Jira**, y se prioriza para su corrección en el sprint actual o los futuros.

**5. Riesgos y Contingencias**

**5.1 Riesgos**

* Posibles fallos en la integración entre frontend y backend.
* Introducción de errores en funcionalidades ya existentes debido a cambios en nuevos sprints.
* Dependencia del tiempo limitado para realizar pruebas exhaustivas dentro de cada sprint.

**5.2 Estrategias de Mitigación**

* Uso de pruebas automatizadas para garantizar la cobertura de pruebas regresivas.
* Comunicación constante entre desarrolladores durante los **Daily Scrums** para identificar y abordar los riesgos de forma proactiva.

**6. Entregables**

**6.1 Informes de Pruebas**

Especifica los informes que se generarán, incluyendo informes de resumen, informes detallados y métricas de pruebas.

**Informes de sprint**: Al final de cada sprint, se generará un reporte de pruebas que detallará los casos de prueba ejecutados, los defectos encontrados y los resultados obtenidos.

**6.2 Documentación Actualizada**

Indica cómo se mantendrá actualizada la documentación de pruebas a lo largo del proyecto.

**7. Aprobación**

**7.1 Aprobación del Cliente**

Después de completados los primeros 3 sprints y una que la interfaz gráfica esté implementada y comunicándose con el backend, se podrá hacer una demostración visualmente atractiva para el cliente, donde se mostrara:

* Cómo se pueden gestionar los productos desde la interfaz gráfica.
* Actualización de stock, viendo cómo las solicitudes llegan al backend.

De esta manera se busca mantener al cliente involucrado desde las primeras etapas, además de darle una visión más completa y final del producto después del sprint 3, para que este genere una retroalimentación.

**8. Historial de Revisiones**

|  |  |  |  |
| --- | --- | --- | --- |
| **Versión** | **Descripción** | **Autor** | **Fecha** |
| **1.0** | **Creación del documento** | **Omar Castro** | **15/10/2024** |
| **2.0** | **Pruebas de Estrés y Rendimiento** | **Omar Castro** | **30/10/2024** |
| **3.0** | **Pruebas de Regresión** | **Jorge Ibarra** | **20/11/2024** |
| **4.0** | **Pruebas de Compatibilidad y Usabilidad** | **Junior Ravelo** | **25/11/2024** |