**Informe Técnico Inventory Motor**

**1. Resumen Ejecutivo:**

* Objetivo principal de la aplicación:

Mejorar la eficiencia y efectividad en la gestión de los recursos de inventario. permitir el seguimiento y el control de los niveles de inventario, mejorar la gestión de compras y ventas, el rastreo de información acerca de clientes y proveedores, así como para facilitar la identificación de productos en exceso de inventario.

* Funcionalidades clave: Agregar productos, Actualizar productos, Eliminar productos, Crear venta , actualizar venta, agregar clientes, realizar seguimiento de stock, generar reportes, etc.
* Beneficios para el usuario: Mayor eficiencia, reducción de errores, toma de decisiones basada en datos.
* Resultados destacados: Aumento en la precisión del inventario, reducción de costos, etc.

**2. Introducción:**

* Contexto del problema:
* Moto repuestos Jors enfrenta varios desafíos en relación a su gestión de inventario. La empresa no cuenta con un sistema automatizado de gestión de inventario, lo que dificulta la tarea de llevar un registro preciso y actualizado de las existencias de repuestos para motocicletas. Esto ha llevado a situaciones en las que Moto repuestos Jors se queda sin stock de un determinado repuesto, lo que puede generar retrasos y pérdida de ventas. Además, la gestión manual del inventario puede ser costosa en términos de tiempo y esfuerzo, limitando la capacidad de la empresa para enfocarse en otras áreas importantes de su negocio, como la atención al cliente o la expansión de su oferta de productos y servicios. Por último, la falta de un control adecuado del inventario puede llevar a Moto repuestos Jors a incurrir en costos adicionales por la compra excesiva de repuestos que no se necesitan, o por la acumulación de inventario obsoleto o inservible.
* Objetivos específicos del proyecto:
  + Identificar los procesos actuales de gestión de inventario en Moto repuestos Jors.
  + Diseñar un software de sistema de gestión de inventario para Moto repuestos Jors.
  + Implementar el sistema de gestión de inventario en Moto repuestos Jors.

**3. Requerimientos del Sistema:**

* **Funcionales:**
* Gestión de inventario: Debe poder agregar, eliminar y actualizar los productos fácilmente. Además, necesita tener un seguimiento en tiempo real del stock disponible de cada repuesto.
* Búsqueda avanzada: Implementa una función de búsqueda que permita a los usuarios buscar productos por nombre, número de parte, marca, modelo de moto, etc.
* Información detallada del producto: Proporciona descripciones detalladas de cada repuesto, incluyendo imágenes, especificaciones técnicas, compatibilidad con modelos de motos específicos, etc.
* Registro de compras: Debe haber un módulo dedicado para registrar las compras realizadas a los proveedores. Esto incluye ingresar información como proveedor, productos comprados, cantidad, fecha de compra, precio unitario, etc.
* Registro de ventas: Debe haber una función para registrar las ventas realizadas a los clientes. Esto implica ingresar detalles como cliente (si es necesario), productos vendidos, cantidad, fecha de venta, precio unitario, etc.
* Generación de facturas: El software debe permitir generar facturas para las ventas realizadas. Las facturas deben contener toda la información relevante de la venta, como productos vendidos, precios, impuestos, etc.
* Seguimiento de stock: El software debe realizar un seguimiento en tiempo real del stock disponible para cada producto, actualizándose automáticamente con cada compra o venta registrada.
* Reportes: Debería haber herramientas de generación de informes que permitan a los administradores analizar las transacciones, el estado del inventario, las compras a proveedores, las ventas realizadas, etc.
* Getión de usuarios y permisos: El software debe permitir la creación de cuentas de usuario para los empleados y administradores, con diferentes niveles de acceso y permisos según sus roles en la empresa.
* **No funcionales:** 
  + Seguridad: El sistema debe cumplir con los estándares de seguridad de la industria para proteger la información confidencial de los clientes, proveedores y transacciones. Esto incluye el cifrado de datos, la autenticación de usuarios, el control de acceso basado en roles y la protección contra ataques cibernéticos.
  + Rendimiento: El software debe ser capaz de manejar cargas de trabajo significativas sin experimentar degradación del rendimiento. Debe tener tiempos de respuesta rápidos tanto para los usuarios que ingresan datos como para aquellos que consultan información en el sistema.
  + Disponibilidad: El sistema debe estar disponible en todo momento durante el horario de operación del negocio. Debe tener una alta disponibilidad para minimizar el tiempo de inactividad y garantizar que los usuarios puedan acceder al sistema cuando lo necesiten.
  + Usabilidad: El software debe ser intuitivo y fácil de usar para los empleados que lo utilizarán en su día a día. Debe tener una interfaz de usuario limpia y bien diseñada que facilite la navegación y la entrada de datos.
  + Compatibilidad: El sistema debe ser compatible con una variedad de dispositivos y navegadores web para garantizar que los usuarios puedan acceder a él desde diferentes plataformas y sistemas operativos.
  + Mantenibilidad: El código del software debe estar bien estructurado y documentado para facilitar su mantenimiento y futuras actualizaciones. Debe ser fácil para los desarrolladores realizar cambios y mejoras en el sistema sin afectar su estabilidad.
* **4. Diseño:**
* Arquitectura de la aplicación:
  + Frontend: Framework web moderno como React .
  + Backend: Spring boot.
  + Base de datos: MySQL para la gestión de datos.
* Diseño de la interfaz de usuario (UI): Diseño intuitivo.
* Base de datos: MySQL para la gestión de datos.

**5. Desarrollo:**

* **Tecnologías**: Spring boot, React, MySQL.
* **Metodología**: Scrum, con sprints de 1 a 2 semanas.
* **Desafíos**: Integrar múltiples categorías y manejar altos volúmenes de datos.

**6. Pruebas:**

* Tipos de pruebas realizadas:



* Herramientas de prueba utilizadas.
* 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.

**Anexos:**

* Diagramas de flujo, diagramas de clases. 

