Pontificia Universidad Católica Madre y Maestra (PUCMM)

Facultad de Ciencias e Ingeniería

Escuela de Ingeniería en Computación y Telecomunicaciones

Estudiante | Matrícula:

Steven Manuel Mateo Ramos | 1014-7069

Romario Isaias Abreu Santos | 1014-5837

Profesor:

Freddy A. Peña P.

Materia:

Aseguramiento de la calidad de software

Asunto:

Avance acerca Sistema de Gestión de Inventarios con QAS

Santiago de los Caballeros

Miércoles, 9 de julio de 2025

Alcance y Objetivos

¿Qué vamos a construir?

Quantum Stock será un sistema de gestión de inventarios pensado para PyMEs que necesitan tener control sobre sus productos sin complicarse la vida. La idea surgió después de ver cómo muchas empresas pequeñas todavía manejan sus inventarios en Excel o, peor aún, en papel.

Funcionalidades principales

Gestión de productos

Lo básico pero bien hecho: poder agregar productos nuevos, editarlos cuando cambien los precios, y eliminarlos cuando ya no los vendamos. Cada producto tendrá su nombre, descripción, categoría, precio y la cantidad que tenemos en stock. También queremos incluir un "stock mínimo" para que el sistema nos avise cuando se esté acabando algo.

La búsqueda tiene que ser rápida - si tienes 500 productos, no puedes estar scrolleando toda la lista para encontrar uno.

Control de stock

Aquí está la parte importante: registrar cuando entra y sale mercancía. Queremos que sea fácil de usar, porque probablemente lo va a usar el empleado del almacén que no necesariamente es experto en sistemas.

Las alertas de stock mínimo van a ser clave. Nada de que se te acabe un producto estrella porque nadie se dio cuenta.

Dashboard e interfaz

Un tablero principal donde veas de un vistazo cómo está tu inventario. Productos que se están acabando, movimientos recientes, tal vez algunas estadísticas básicas. Sin sobrecargar con información innecesaria.

La interfaz tiene que ser intuitiva. Mi referencia es: si mi tía puede usarla sin manual, entonces está bien diseñada.

Roles de usuario

- Administrador: Puede hacer todo.
- Empleado: Puede gestionar productos y stock, pero no puede eliminar productos por seguridad
- Invitado: Sólo puede ver productos y reportes básicos (útil para mostrar catálogos)

API para integraciones

Vamos a crear una API REST decente para que otros sistemas se puedan conectar. Pensamos en puntos de venta, software de contabilidad, incluso apps móviles en el futuro.

Pruebas y calidad

Aquí no vamos a escatimar:

- Pruebas automatizadas con JUnit y Selenium
- Pruebas de seguridad (porque los datos de inventario son sensibles)
- Pruebas de usabilidad con usuarios reales
- Pipeline de CI/CD con GitHub Actions

Para el monitoreo usaremos Prometheus y Grafana. Queremos saber si algo se rompe antes de que el cliente se queje.

Nuestros objetivos

Lo que queremos lograr:

1. Que funcione bien: Suena obvio, pero hemos visto muchos sistemas que no hacen lo básico correctamente.
2. Que sea fácil de usar: Si necesitas un manual de 50 páginas para usar el sistema, algo está mal.
3. Que sea seguro: Los datos de inventario son críticos para cualquier negocio. Nada de vulnerabilidades tontas.
4. Que crezca con el negocio: El sistema debe funcionar igual de bien con 100 productos que con 10,000.
5. Que se integre fácilmente: La API tiene que ser clara y bien documentada.

Gestión de Riesgos

Tipo de Riesgo	Descripción
Riesgos Técnicos (RT)	Riesgos relacionados con la tecnología, arquitectura, desarrollo e implementación del sistema.
Riesgos del Proyecto (RP)	Riesgos relacionados con la gestión, organización, recursos y ejecución del proyecto.
Riesgos Externos (RE)	Riesgos fuera del control directo del equipo que pueden afectar el proyecto.

Código	Riesgo	Descripción	Probabilidad	Impacto	Nivel de Riesgo
RT001	Incompatibilidad entre Tecnologías	Las tecnologías seleccionadas (Frontend, Backend, Base de Datos) presentan incompatibilidades o problemas de integración.	3	4	12 (Alto)
RT002	Problemas de Rendimiento y Escalabilidad	El sistema no cumple con los requisitos de rendimiento bajo carga o no escala adecuadamente	3	4	12 (Alto)
RT003	Vulnerabilidades de Seguridad	El sistema presenta vulnerabilidades que comprometen la seguridad y confidencialidad de los datos	4	5	20 (Crítico)
RT004	Corrupción o Pérdida de Datos	Pérdida, corrupción o inconsistencia de datos críticos del inventario	2	5	10 (Medio)
RT005	Fallos en Pipeline CI/CD y Despliegue	Problemas en la configuración, ejecución o confiabilidad del pipeline de integración y despliegue continuo	3	3	9 (Medio)

RT006	Problemas de Compatibilidad Multi-Browser/ Dispositivo	La aplicación no funciona correctamente en diferentes navegadores o dispositivos	3	3	9 (Medio)
RT007	Problemas con Contenedorización	Issues con Docker, configuración de contenedores	2	3	6 (Medio)
RP001	Retrasos en el Cronograma del Proyecto	El proyecto no se completa dentro de los tiempos establecidos por limitaciones académicas	4	4	16 (Crítico)
RP002	Subestimación de la Complejidad de QAS	La complejidad de implementar todas las etapas de QAS es mayor a la estimada inicialmente	4	3	12 (Alto)
RP003	Falta de Experiencia en Tecnologías Específicas	El equipo no tiene suficiente experiencia en algunas tecnologías requeridas del stack	3	3	9 (Medio)
RE001	Indisponibilidad de Servicios Cloud o Herramientas	Servicios críticos como GitHub, servicios de hosting, o herramientas de desarrollo presentan downtime	2	3	6 (Medio)
RE002	Cambios en Políticas de Herramientas o Licencias	Cambios inesperados en términos de servicio, precios o disponibilidad de herramientas gratuitas	2	2	4 (Bajo)
RE003	Problemas de Conectividad a Internet	Problemas de conectividad que afecten el desarrollo colaborativo o acceso a recursos	2	3	6 (Medio)
RE004	Actualizaciones Críticas de Seguridad en Dependencias	Vulnerabilidades críticas descubiertas en librerías o frameworks que requieren actualizaciones urgentes	3	4	12 (Alto)

Código	Riesgo	Planes de Contingencia
RT001	Incompatibilidad entre Tecnologías	● Consultar con expertos en las tecnologías seleccionadas. ● Preparar stack tecnológico alternativo como plan B.
RT002	Problemas de Rendimiento y Escalabilidad	● Implementar pruebas de estrés con JMeter desde etapas tempranas del desarrollo. ● Diseñar arquitectura escalable y modular desde el inicio. ● Optimizar consultas de base de datos. ● Establecer métricas de rendimiento y monitoreo continuo con Prometheus/Grafana.
RT003	Vulnerabilidades de Seguridad	● Implementar autenticación y autorización robusta (KeyCloak) desde el inicio. ● Mantener dependencias actualizadas y sin vulnerabilidades conocidas. ● Realizar auditorías de seguridad y code reviews enfocados en seguridad
RT004	Corrupción o Pérdida de Datos	● Usar Hibernate Envers para auditoría completa de cambios. ● Implementar validaciones de integridad de datos a nivel de aplicación. ● Realizar pruebas de recuperación de datos mensualmente
RT005	Fallos en Pipeline CI/CD y Despliegue	● Configurar pipeline incrementalmente con validación en cada etapa ● Implementar rollback automático y blue-green deployments
RT006	Problemas de Compatibilidad Multi-Browser/Dispositivo	● Implementar pruebas automatizadas con Playwright en múltiples navegadores ● Realizar pruebas manuales regulares en dispositivos físicos diversos ● Usar CSS Grid/Flexbox y tecnologías estándar para responsividad
RT007	Problemas con Contenedorización	● Usar imágenes base oficiales y estables ● Documentar completamente Dockerfiles y docker-compose configurations
RP001	Retrasos en el Cronograma del Proyecto	● Implementar metodología ágil con sprints de 1-2 semanas ● Realizar daily stand-ups y revisiones semanales de progreso

		● Usar herramientas de gestión de proyecto (Jira)
RP002	Subestimación de la Complejidad de QAS	● Dividir requisitos de QAS en fases implementables incrementalmente ● Buscar asesoría del docente y recursos adicionales tempranamente ● Usar herramientas y frameworks que simplifiquen la implementación
RP003	Falta de Experiencia en Tecnologías Específicas	● Asignar tiempo específico para investigación de nuevas tecnologías ● Comenzar con implementaciones simples antes de funcionalidades complejas
RE001	Indisponibilidad de Servicios Cloud o Herramientas	● Mantener repositorios de código en múltiples ubicaciones ● Usar múltiples proveedores de hosting/cloud como alternativas
RE002	Cambios en Políticas de Herramientas o Licencias	● Priorizar herramientas open source y con licencias estables ● Mantener alternativas identificadas para cada herramienta crítica
RE003	Problemas de Conectividad a Internet	● Configurar entornos de desarrollo que funcionen offline
RE004	Actualizaciones Críticas de Seguridad en Dependencias	● Configurar alertas automáticas de seguridad en GitHub/repositorios ● Mantener dependencias actualizadas regularmente (no esperar a emergencias)

Documentación de Requisitos

Requisitos Funcionales

Gestión de Productos

- **RF-1.1**: El sistema permitirá agregar un nuevo producto con los siguientes atributos: nombre, descripción, categoría, precio, cantidad inicial y stock mínimo.
- **RF-1.2**: El sistema permitirá editar la información de un producto existente, actualizando cualquiera de los atributos mencionados en RF-1.1.
- **RF-1.3**: El sistema permitirá eliminar un producto del inventario (sólo para administradores).
- **RF-1.4**: El sistema mostrará una lista de todos los productos con opciones de búsqueda por nombre o categoría y filtrado por atributos como precio o stock.
- **RF-1.5**: Los usuarios invitados solo podrán visualizar la lista de productos y sus detalles básicos, sin opciones de edición o eliminación.

Control de Stock

- **RF-2.1**: El sistema permitirá actualizar la cantidad de productos en el inventario para registrar entradas y salidas.
- **RF-2.2**: El sistema generará alertas automáticas cuando un producto alcance o esté por debajo del stock mínimo establecido.
- **RF-2.3**: El sistema mantendrá un historial de movimientos de stock, registrando: fecha, tipo de movimiento, cantidad y usuario responsable.
- **RF-2.4**: Los usuarios invitados no tendrán acceso al control de stock ni al historial de movimientos.

Integración con Otros Sistemas

- **RF-3.1**: El sistema proporcionará una API RESTful para integrarse con otras aplicaciones (ej. sistemas de contabilidad o puntos de venta).
- **RF-3.2**: La API utilizará autenticación basada en JWT para garantizar la seguridad.
- **RF-3.3**: La API permitirá realizar operaciones CRUD sobre productos y consultar el historial de movimientos (solo para administradores).

Interfaz de Usuario

- **RF-4.1**: El sistema incluirá un dashboard que muestre:
    - Resumen del estado del inventario (ej. total de productos, productos con stock bajo).
    - Estadísticas clave (ej. movimientos recientes, productos más vendidos).
- **RF-4.2**: La interfaz será intuitiva, con navegación clara y accesible para todos los roles de usuario.
- **RF-4.3**: El sistema soportará visualización en diferentes dispositivos (responsivo).

Roles y Permisos

- **RF-5.1**: El sistema soportará tres roles de usuario:
    - **Administrador**:
        - Acceso completo a todas las funcionalidades (CRUD en productos y stock, acceso a API).
    - **Empleado**:
        - Puede agregar, editar y visualizar productos (sin eliminación).
        - Puede actualizar stock y visualizar historial de movimientos.
    - **Usuario Invitado/Cliente**:
        - Solo puede visualizar productos y reportes públicos básicos.
- **RF-5.2**: El sistema implementará autenticación y autorización basadas en OAuth2 o JWT para controlar el acceso según el rol.

Reportes

- **RF-6.1**: El sistema generará reportes básicos de inventario (ej. lista de productos, stock actual) accesibles para todos los roles.
- **RF-6.2**: Los administradores y empleados podrán acceder a reportes detallados, como historial de movimientos.

Requisitos No Funcionales

Rendimiento

- **RNF-1.1**: El sistema manejará hasta grandes cantidades de transacciones simultáneas sin degradación significativa del rendimiento.
- **RNF-1.2**: El tiempo de respuesta para operaciones CRUD no excederá los 2 segundos bajo carga normal.
- **RNF-1.3**: El sistema soportará hasta un alto volúmen de productos registrados en la base de datos.

3.2 Seguridad

- **RNF-2.1**: El sistema implementará medidas contra ataques comunes (ej. inyección SQL, XSS).
- **RNF-2.2**: Las sesiones de usuario tendrán un tiempo de expiración configurable.
- **RNF-2.3**: La API estará protegida contra accesos no autorizados mediante JWT.

3.3 Usabilidad

- **RNF-3.1**: El sistema tendrá una interfaz intuitiva, evaluada mediante pruebas de usabilidad con usuarios finales.

3.4 Compatibilidad

- **RNF-4.1**: El sistema será compatible con navegadores modernos (Chrome, Firefox, Edge, Safari) en sus últimas versiones.
- **RNF-4.2**: El sistema funcionará en dispositivos móviles y de escritorio.

3.5 Escalabilidad

- **RNF-5.1**: El sistema permitirá escalar horizontalmente mediante contenedores.
- **RNF-5.2**: La base de datos soportará migraciones automáticas usando herramientas como Flyway o Liquibase.

3.6 Mantenibilidad

- **RNF-6.1**: El sistema incluirá documentación técnica completa (arquitectura, instalación, mantenimiento).

- **RNF-6.2**: Las revisiones de código serán obligatorias para cada pull request.

3.7 Monitoreo y Observabilidad

- **RNF-7.1**: El sistema implementará monitoreo en producción usando Prometheus y Grafana.
- **RNF-7.2**: Los errores críticos generarán alertas automáticas al equipo de soporte.

3.8 Calidad

- **RNF-8.1**: Se definirán métricas de calidad (cobertura de pruebas, tiempo de respuesta) y se monitorearán regularmente.

Mantenimiento y Actualizaciones

Monitoreo Diario

- Monitoreo del Sistema: Verificación automática de la salud de los servicios
- Backup de Base de Datos: Respaldo automático de MySQL con retención de 7 días
- Revisión de Logs: Análisis automatizado de errores críticos
- Verificación de Certificados SSL: Monitoreo de expiración de certificados

Mantenimiento Semanal

- Análisis de Performance: Revisión de métricas de rendimiento
- Limpieza de Logs: Rotación y archivado de logs antiguos
- Verificación de Backups: Validación de integridad de respaldos
- Actualización de Dependencias Menores: Parches de seguridad automáticos

Mantenimiento Mensual

- Revisión de Seguridad: Análisis de vulnerabilidades conocidas
- Optimización de Base de Datos: Limpieza y optimización de índices MySQL
- Actualización de Documentación: Revisión y actualización de documentación técnica
- Pruebas de Recuperación: Validación de procedimientos de disaster recovery