Plan de Proyecto: Sistema de Gestión de Inventarios con QAS

1. INFORMACIÓN GENERAL DEL PROYECTO

1.1 Título del Proyecto

Sistema de Gestión de Inventarios con Aseguramiento de Calidad del Software (QAS)

1.2 Descripción

Desarrollo de un sistema web para la gestión de inventarios dirigido a pequeñas y medianas empresas, implementando todas las etapas del ciclo de vida del aseguramiento de la calidad del software mediante metodologías ágiles y herramientas modernas.

1.3 Objetivos del Proyecto

Objetivo General

Desarrollar un sistema de gestión de inventarios robusto, seguro y escalable que permita a las PyMEs (Pequeñas y Medianas Empresas) gestionar eficientemente sus productos y stock, aplicando estándares profesionales de calidad del software y todas las etapas del ciclo QAS.

Objetivos Específicos

- Implementar funcionalidades completas de gestión de productos (CRUD) con categorización
- Desarrollar un sistema de control de stock con etiquetas automatizadas
- Crear una API REST segura con autenticación JWT para integración con sistemas externos
- Diseñar una interfaz de usuario intuitiva y responsive usando Thymeleaf
- Establecer un sistema de roles y permisos de acceso (Administrador, Empleado, Invitado)
- Aplicar rigurosamente todas las etapas del ciclo QAS con pruebas automatizadas
- Implementar CI/CD completo con GitHub Actions
- Configurar monitoreo y observabilidad del sistema con Prometheus y Grafana

1.4 Tecnologías del Proyecto

- **Backend**: Spring Boot
- **Frontend**: Thymeleaf
- **Base de Datos**: MySQL

- **Auditoría**: Hibernate Envers
- **Autenticación**: OAuth2
- **Pruebas**: JUnit/TestNG, Playwright
- **CI/CD**: GitHub Actions
- **Contenedorización**: Docker
- **Migración de BD**: Flyway
- **Monitoreo**: Prometheus, Grafana

2. ALCANCE DEL PROYECTO

2.1 Incluido en el Alcance

Funcionalidades

- **Gestión de Productos**: CRUD completo con nombre, descripción, categoría, precio, cantidad y stock mínimo
- **Control de Stock**: Actualización de inventario, etiquetas automáticas por stock mínimo.
- **Sistema de Usuarios**: 3 niveles de acceso con autenticación OAuth2
- **API RESTful**: Endpoints documentados para integraciones externas
- **Dashboard**: Panel de control con métricas y estadísticas clave
- **Auditoría**: Trazabilidad completa de cambios con Hibernate Envers

Aspectos Técnicos

- Arquitectura web cliente-servidor
- Diseño responsive y accesible
- Pipeline CI/CD automatizado
- Contenedorización completa
- Monitoreo en tiempo real
- Documentación técnica y de usuario

2.2 Excluido del Alcance

- Aplicación móvil nativa
- Módulo de facturación avanzada
- Integración directa con sistemas contables específicos
- Reportes financieros complejos
- Soporte para múltiples idiomas
- Gestión de múltiples sucursales

3. ENTREGABLES DEL PROYECTO

3.1 Entregables de Desarrollo

1. **Código Fuente Completo e Implementación desplegada en local**
    - Backend Spring Boot con arquitectura en capas
    - Frontend Thymeleaf con componentes reutilizables
    - Scripts de base de datos MySQL con Flyway
    - Configuraciones Docker y Docker Compose
2. **Sistema Funcional Desplegado**
    - Ambiente de desarrollo local
    - Ambiente de staging para pruebas (No en etapa inicial)
    - Configuración para producción (No en etapa inicial)

3.2 Entregables de Documentación

1. **Documentación de Requisitos**
    - Especificación de requisitos funcionales y no funcionales
    - Casos de uso detallados con diagramas
    - Historias de usuario priorizadas
2. **Documentación Técnica**
    - Arquitectura del sistema y patrones de diseño
    - Modelo de datos y diagramas Entidad-Relación
    - Documentación completa de API REST
    - Guía de instalación y configuración
    - Manual de despliegue y mantenimiento (No en etapa inicial)
3. **Documentación de Pruebas**
    - Plan maestro de pruebas
    - Casos de prueba automatizados
    - Reportes de ejecución y cobertura
    - Análisis de rendimiento y seguridad
4. **Documentación de Usuario**
    - Manual de usuario por rol (Admin, Empleado, Invitado)
    - Guías de uso rápido con capturas (No en etapa inicial)

3.3 Entregables de Calidad

1. **Suite de Pruebas Automatizadas**
    - Pruebas unitarias (JUnit/TestNG)
    - Pruebas de integración
    - Pruebas E2E (Playwright)
    - Pruebas de seguridad y rendimiento
2. **Infraestructura QAS**
    - Pipeline CI/CD configurado y funcional (No en etapa inicial)
    - Monitoreo con Prometheus y Grafana (No en etapa inicial)

- Métricas de calidad establecidas
- Dashboards de observabilidad (No en etapa inicial)

4. CRONOGRAMA DETALLADO DEL PROYECTO

Semana 1:

Actividades:

- Análisis detallado de requisitos funcionales y no funcionales
- Diseño de arquitectura del sistema y patrones
- Modelado de base de datos y relaciones
- Creación de wireframes y mockups de UI
- Selección final de tecnologías y herramientas
- Configuración del repositorio GitHub con estructura de branches
- Setup inicial de Spring Boot con dependencias
- Configuración de MySQL y Flyway
- Configuración inicial de Docker
- Setup de herramientas de desarrollo (IDEs, plugins)
- Creación de estructura base del proyecto

Entregables:

- Documento de requisitos completo
- Diagramas de arquitectura
- Modelo de datos MySQL
- Mockups de interfaz
- Repositorio GitHub configurado
- Proyecto Spring Boot inicializado
- Base de datos MySQL configurada
- Ambiente de desarrollo funcional

Semana 2:

Actividades:

- Implementación de entidades JPA (Producto, Usuario, Movimiento, Categoría)
- Configuración de Hibernate Envers para auditoría
- Creación de repositorios JPA con consultas customizadas
- Configuración inicial de OAuth2
- Implementación de servicios de negocio (ProductoService, StockService, UsuarioService)
- Desarrollo de controladores REST con validaciones
- Implementación de sistema de roles y permisos
- Manejo de excepciones global

Entregables:

- Modelo de datos implementado
- Repositorios funcionales
- Configuración OAuth2 básica
- API REST funcional para productos

- Sistema de autenticación OAuth2
- Controladores con manejo de errores

Semana 3:

Actividades:

- Implementación de alertas de stock mínimo
- API de integración para sistemas externos
- Pruebas unitarias con JUnit
- Configuración de Thymeleaf
- Implementación de sistema de autenticación frontend
- Desarrollo de componentes base y fragmentos reutilizables
- Implementación de manejo de sesiones

Entregables:

- Backend completo con todas las funcionalidades
- Suite de pruebas unitarias
- Documentación de API
- Estructura base de templates
- Sistema de login funcional
- Componentes reutilizables

Semana 4:

Actividades:

- Desarrollo del dashboard principal con estadísticas
- Implementación de CRUD de productos con validaciones frontend
- Páginas de gestión de stock
- Sistema de alertas visuales
- Implementación de búsqueda y filtros
- Integración completa frontend-backend
- Implementación de manejo de estados y errores
- Optimización de rendimiento y carga
- Implementación de feedback visual para usuarios
- Refinamiento de UX/UI basado en testing

Entregables:

- Dashboard completo con métricas
- Páginas de gestión funcionales
- Sistema de alertas implementado
- Sistema integrado completamente funcional
- Interfaz optimizada y responsive
- Manejo robusto de errores

Semana 5:

Actividades:

- Implementación de pruebas de integración
- Desarrollo de pruebas E2E con Playwright
- Pruebas de usabilidad y accesibilidad
- Pruebas de compatibilidad cross-browser
- Análisis de cobertura de código
- Auditoría de seguridad
- Pruebas de rendimiento y estrés con JMeter
- Optimización de consultas de base de datos
- Implementación de rate limiting y validaciones de seguridad
- Corrección de vulnerabilidades identificadas

Entregables:

- Suite completa de pruebas automatizadas
- Reportes de cobertura de código
- Resultados de pruebas de usabilidad
- Reporte de auditoría de seguridad
- Métricas de rendimiento
- Sistema optimizado y seguro

Semana 6:

Actividades:

- Configuración completa de GitHub Actions
- Configuración de contenedores Docker optimizados
- Configuración de ambientes de staging
- Configuración de Prometheus para métricas
- Setup de Grafana con dashboards customizados
- Implementación de alertas y notificaciones
- Configuración de logs centralizados
- Pruebas de monitoreo en diferentes escenarios

Entregables:

- Pipeline CI/CD completamente automatizado
- Contenedores Docker optimizados
- Ambiente de staging funcional
- Sistema de monitoreo completo
- Dashboards de Grafana configurados
- Alertas automatizadas funcionando

Semana 7:

Actividades:

- Finalización de documentación técnica
- Creación de manuales de usuario detallados
- Revisión y actualización de toda la documentación
- Pruebas finales del sistema completo
- Revisión final de código y documentación
- Presentación en clase con demostración funcional
- Revisión post-mortem del proyecto

Entregables:

- Documentación técnica completa
- Manuales de usuario finalizados
- Guías de instalación y mantenimiento
- Presentación final del proyecto
- Sistema completamente funcional
- Entrega final con todos los componentes

5. GESTIÓN DE RIESGOS

5.1 Matrix de Riesgos Identificados

Riesgo	Probabilidad	Impacto	Nivel	Estrategia de Mitigación
Retrasos en cronograma	Alta	Alto	Crítico	• Buffer de 10% en cada fase • Reuniones de seguimiento bi-semanales • Priorización clara de funcionalidades MVP
Complejidad OAuth2	Media	Alto	Alto	• Investigación temprana y prototipos • Documentación detallada • Plan B con autenticación básica JWT
Problemas de integración	Media	Alto	Alto	• Definición clara de contratos API • Pruebas de integración desde etapa temprana • Desarrollo incremental con validaciones
Curva de aprendizaje tecnológico	Alta	Medio	Medio	• Capacitación previa en tecnologías clave • Documentación de aprendizajes • Pair programming para transferir conocimiento
Fallos en CI/CD	Media	Medio	Medio	• Configuración incremental y validada • Backup de configuraciones • Testing local antes de push
Problemas de rendimiento	Baja	Alto	Medio	• Pruebas de carga desde etapas tempranas • Optimización de consultas DB • Monitoreo continuo con métricas
Indisponibilidad del equipo	Baja	Alto	Medio	• Documentación compartida y actualizada • Distribución equitativa del conocimiento • Backup plans para tareas críticas

5.2 Plan de Contingencia

- **Escalación**: Problemas críticos se escalan inmediatamente al instructor

- **Reuniones de Crisis**: Sesiones extraordinarias para resolver blockers
- **Repriorización**: Flexibilidad para ajustar alcance manteniendo funcionalidades core
- **Recursos Adicionales**: Identificación de fuentes de ayuda externa (documentación, comunidad)

6. GESTIÓN DE TAREAS Y SEGUIMIENTO

6.1 Herramientas de Gestión

- **GitHub Projects**: Kanban board principal para tracking
- **GitHub Issues**: Gestión detallada de tareas, bugs y features
- **GitHub Milestones**: Seguimiento de hitos principales
- **Pull Requests**: Revisión de código obligatoria
- **GitHub Actions**: Automatización y CI/CD

6.2 Metodología de Trabajo

Flujo de Trabajo

1. **Creación de Issues**: Cada tarea como issue con labels y assignee
2. **Feature Branches**: Una rama por funcionalidad
3. **Pull Requests**: Review obligatorio antes de merge
4. **Testing**: CI/CD ejecuta tests automáticamente
5. **Deployment**: Automático a staging, manual a producción

6.3 Definición de "Terminado" (DoD)

Una tarea se considera completada cuando:

- Funcionalidad implementada según acceptance criteria
- Pruebas unitarias escritas y pasando (>80% cobertura)
- Pruebas de integración pasando
- Code review aprobado por otro miembro del equipo
- Documentación actualizada (código, API, usuario)
- Sin vulnerabilidades de seguridad críticas o altas

7. COMUNICACIÓN Y SEGUIMIENTO

7.1 Reuniones Programadas

- **Daily Standups**: Lunes, Miércoles, Viernes (15 minutos)

    - ¿Qué hice desde la última reunión?
    - ¿Qué haré hasta la próxima?
    - ¿Qué impedimentos tengo?
- **Sprint Reviews**: Cada 2 semanas (1 hora)
    - Demo de funcionalidades completadas
    - Revisión de métricas y progreso
    - Planificación del siguiente sprint
- **Retrospectivas**: Cada 2 semanas (45 minutos)
    - ¿Qué funcionó bien?
    - ¿Qué se puede mejorar?
    - Acciones de mejora para el siguiente sprint

7.2 Canales de Comunicación

- **GitHub**: Canal principal para todo lo técnico
- **WhatsApp/Discord**: Comunicación rápida y coordinación
- **Teams**: Reuniones virtuales y pair programming
- **Outlook/Teams**: Comunicación formal con instructor

7.3 Reportes de Progreso

- **Reporte de Hitos**: Al completar cada fase principal
- **Reporte Final**: Análisis completo y lecciones aprendidas