Plan de Pruebas Ágil

Fire Prevention

Integrantes: Vicente Gutierrez

Martín Vega

Monserrat Vilches

Nicolás Yañez

| Proyecto | Fire Prevention - Sistema de Gestión de Inventario |
| --- | --- |
| Equipo | Vicente Gutierrez  Martín Vega  Monserrat Vilches  Nicolas Yañez |
| Fecha | 2 de Octubre de 2025 |

## Objetivo de la prueba

El objetivo de este documento es establecer cómo planificaremos, diseñaremos, ejecutaremos y reportaremos las pruebas del sistema de manera iterativa, asegurando la integridad de los datos y trazabilidad por lote/bodega, control de accesos por rol, junto a la correcta ejecución del flujo de negocio; petición → reserva → preparación → despacho → recepción → devolución, además de reportes y alertas de stock. El plan se alinea con las Historias de Usuario + Criterios de Aceptación y con los objetivos del proyecto.

## Alcance

Dentro de los módulos y flujos que se probarán incluyen:

* Autenticación y autorización: Login correcto/incorrecto, rutas y botones según rol; acceso a “Administración” solo a quien corresponda.
* Productos: Crear/editar productos con SKU único, filtros combinados, búsqueda por SKU/nombre.
* Bodegas y ubicaciones: Creación y validación de códigos/formato.
* Lotes y movimientos: Registro de lote al ingresar stock, devolución trazada al lote original, control de PPP (Precio Promedio Ponderado) y logs de movimiento.
* Peticiones / reservas / preparación: Crear, aprobar/rechazar con motivo.
* Despacho y recepción (total/parcial): Salida con comprobante/guía, recepción total o parcial con kardex y redondeos controlados.
* Reportes / costos: Secuencia y saldos correctos por producto/lote/bodega; valoración por PPP donde aplique.
* Alertas de stock: Umbral configurable por producto/bodega y notificación visible.
* API: Validación de payload (errores 400 con mensajes claros) y estructura de respuesta.
* Rendimiento base: Respuesta aceptable en productos con datos semilla

## Tipos de pruebas

Pruebas Funcionales:

* Unitarias: Validación lógica en aislamiento (servicios/validadores/utils y lógica de componentes)
* Integración (API/DB): Rutas express + servicios + Prisma + DB; HTTP y efectos en BD.
* E2E: Flujos críticos punta a punta (UI + API + datos); pocos y estables.

Pruebas de regresión:

* Suite mínima por sprint para asegurar que lo entregado sigue funcionando tras cambios.

Aceptación:

* Validación con PO por criterios de cada HU.

Usabilidad ligera:

* Chequeos rápidos: Foco en inputs, estado loading/empty/error, búsqueda por SKU/nombre.

Seguridad básica:

* Autenticación/roles, rate-limit de login, mensajes de error sin filtrar info sensible.

Datos (Consistencia e integridad):

* Kardex/saldos: Suma de todas las entradas - suma de todas las salidas = stock; devoluciones al lote original; PPP correcto donde aplique.

## Criterios de aceptación

Transversales:

* No debe quedar pendiente ningún bug crítico, ni grave que afecte la HU o módulo.
* Errores de validación en 4xx con mensajes claros.
* Permisos correctos y UI sin acciones ocultas para roles no autorizados.
* Consistencia de datos: El stock nunca debe ser menor a 0, la suma de entradas - suma de salidas = stock por producto-bodega-lote, devoluciones al lote original.
* UI debe mostrar de forma clara cuando se están cargando los datos, cuando no hay resultados y cuando no hubo un problema.
* El tiempo de respuesta del 95% de las peticiones son más rápidas que 250 ms.

## Estrategia de pruebas

* Diseñar y automatizar pruebas
* Cada HU sale con pruebas y evidencias.
* Unitarias: Reglas de negocio, validaciones. Cobertura de más o menos el 80% del código.
* Integración: Endpoints, servicios, prisma, DB (Docker).
* E2E: Sólo flujos críticos y regresión mínima.
* Datos de prueba y ambientes:Seeds versionados, DB efímera en CI, migraciones/rollback, test aislados.

## Responsables

* Product Owner: Prioriza HU, define criterios y aprueba las pruebas de aceptación de usuario.
* Scrum Master: Facilita ceremonias, remueve impedimentos y verifica las condiciones de calidad obligatorias.
* Equipo Dev/QA: Diseña casos, automatiza, ejecuta y adjunta evidencias.

## Ambientes

* Local: En donde se puede programar y probar rápido. Se realizan pruebas unitarias y se prueban con datos ficticios.
* Staging/Demo: Entorno casi real para validar el sprint. Aquí corremos smoke + regresión mínima y hacemos la demo.
* Producción: Esto ya es el despliegue final, en donde tras publicar, corremos un post-deploy (login, listar productos, buscar ´por SKU/nombre, flujo preparar)

## Datos de prueba (Sin datos todavía)

* Estado: Este plan solo fija mínimos y reglas para cuando generemos el seed.
* Mínimos por ambiente:
* Local: 10 productos, 3 bodegas, 2 lotes (en 3-5 productos), 10-20 movimientos, 5 órdenes.
* Staging/demo: 200-1000 productos, mismas bodegas, >50 movimientos, 15-25 órdenes.
* Reglas obligatorias:
* Stock > 0 siempre; suma de entradas - suma de salidas = stock por producto-bodega-lote.
* Las devoluciones vuelven al lote original.
* Roles: Admin / Operador / Bodega (permisos).
* Casos que el seed debe incluir cuando existan datos:
* PPP: Un producto con dos entradas a costos distintos.
* Picking: Producto con múltiples ubicaciones.
* Concurrencia: 2 peticiones sobre el mismo SKU/lote.
* API: Ejemplos de payload inválido y de paginación/orden