|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
| <aaaa-mm-dd> | <1.1.0> | Documento inicial | <Nombre> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

| **Nº Riesgo** | **Descripción** | **Gravedad** | **Acción** |
| --- | --- | --- | --- |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |
| 6 |  |  |  |

| **Nro** | **Caso de Uso** | **Input** | **Output** |
| --- | --- | --- | --- |
| 1 |  |  |  |

| **Nro** | **Caso de Uso** | **Input** | **Output** |
| --- | --- | --- | --- |
|  |  |  |  |

|  |  |
| --- | --- |
| Objetivo de la prueba: | Asegurar la funcionalidad del conjunto de casos, incluyendo la navegación en la aplicación, el ingreso de datos, el proceso y la recuperación (resultados).  Que la navegación a través de los casos de prueba refleje apropiadamente las reglas del negocio y los requerimientos, incluyendo ventana a ventana, campo a campo y usando los métodos de acceso correctamente (tecla *tab*, movimiento del *mouse*, etc.)  Que los objetos de las ventanas y sus características, tales como menús, tamaño, posición, estados y foco, estén de acuerdo a los estándares. |
| Técnica a utilizar: | Ejecutar cada caso de uso, su flujo y funcionalidad usando tanto datos válidos como inválidos para comprobar lo siguiente:   * Que los resultados esperados ocurren cuando los datos válidos son utilizados. * Que el mensaje de error es apropiado cuando se utilizan datos inválidos. * Que cada regla de negocio se utiliza apropiadamente. * Crear y modificar los procedimientos de prueba para cada ventana, para comprobar los estados de los objetos y de la aplicación. |
| Criterio de validación: | * Todas las pruebas planificadas se ejecutaron correctamente. * Todos los defectos identificados han sido asignados. * Cada ventana debe ser verificada para mantener la consistencia con la versión maestra y comprobar que esté dentro de los estándares aceptables. |
| Consideraciones especiales: | [Puede que no todas las propiedades sean verificadas, considerar las más importantes y/o definidas y no incluir todos los objetos de terceras partes.] |
| Observaciones: |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Herramienta** | **Vendedor/Tienda** | **Versión** |
| Guía de Referencia | Rational Unified Process | Rational Software Corp. | 5.5 |
| Administración de Testing | Rational TestManager incluida en el Rational TeamTest | Rational Software Corp. | 7.5 |
| Seguimiento de Defectos | Rational ClearQuest TT Edition incluida en el Rational TeamTest | Rational Software Corp. | 2.0 |
| Testing Funcional | Rational Robot incluido en el Rational TeamTest | Rational Software Corp. | 7.5 |
| ASQ Tool (performance testing) | No definido |  |  |
| Test Coverage Monitor or Profiler | No definido |  |  |
| Herramienta de DBMS | Microsoft Access | Microsoft Software Corp. | 97 |
| SQL Server | Microsoft Software Corp. | 7.0 |

| **Recursos Humanos** | | |
| --- | --- | --- |
| **Rol** | **Recursos mínimos recomendados**  **(Número de personas full-time)** | **Responsabilidades específicas / Comentarios** |
| Diseñador de casos de prueba |  | Responsabilidades   * Identificar, priorizar e implementar los casos de prueba. * Evaluar de forma el esfuerzo de *testing*. |
| Testeador |  | Responsabilidades:   * Ejecutar los casos de prueba. * Guardar estado de los resultados. * Recuperación de errores. * Generar peticiones de cambios en la documentación. |
| Administrador de sistema del pruebas |  | Responsabilidades   * Administrar el sistema de control de pruebas. * Instalar / administrar el acceso al sistema de pruebas. |
| Administrador de la base de datos / Encargado de la base de datos |  | Responsabilidades:   * Administra los datos del prueba (Base de Datos) * Asegurar que el entorno de datos de prueba (base de datos) y los valores que contiene son controlados y mantenidos. |
| Diseñador |  | Responsabilidades:   * Identificar y definir las operaciones, atributos y asociaciones de las clases de prueba. * Identificar y definir los paquetes de prueba. |
| Implementador |  | Responsabilidades:   * Implementar las clases de prueba y los paquetes de prueba. |

| **ITEM** | **DESCRIPCIÓN** | **OBSERVACIONES** |
| --- | --- | --- |
| **HARDWARE** |  |  |
| **Estación de Pruebas (Cliente)** |  |  |
| Procesador | *Pentium II 300 MHz o superior* |  |
| Memoria RAM | *64 MB (128 MB ideal)* |  |
| Espacio en Disco | *600 MB* |  |
| Tipo Monitor y Resolución | *SVGA .28 I 800x600* |  |
| Unidad de Disquete | *1.44 MB* |  |
| Tarjeta de red |  |  |
| Modem |  |  |
| Mouse |  |  |
| Tipo Enlace |  |  |
| **Servidores** |  |  |
| ***Base de Datos (pruebas)*** |  |  |
| Procesador |  |  |
| Memoria RAM |  |  |
| Espacio en Disco |  |  |
| Tipo Monitor y Resolución |  |  |
| Unidad de Disquete |  |  |
| Tarjeta de red |  |  |
| Modem |  |  |
| Mouse |  |  |
| Tipo Enlace |  |  |
| ***Aplicación (pruebas)*** |  |  |
| Procesador |  |  |
| Memoria RAM |  |  |
| Espacio en Disco |  |  |
| Tipo Monitor y Resolución |  |  |
| Unidad de Disquete |  |  |
| Tarjeta de red |  |  |
| Módem |  |  |
| Mouse |  |  |
| Tipo Enlace |  |  |
| **Impresoras** |  |  |
| Marca y Modelo |  |  |
| Tipo |  |  |
| Resolución |  |  |
| Rendimiento |  |  |
| Dedicación |  |  |
| **RED** |  |  |
| Topología |  |  |
| Medio |  |  |
| Velocidad |  |  |
| Protocolo |  |  |
| Módems |  |  |
| Conexión Internet |  |  |
| **Sistema de Respaldo / Restauración** |  |  |
| Unidad (Modelo y Marca) |  |  |
| Capacidad |  |  |
| Ubicación |  |  |
|  |  |  |
| **SOFTWARE** |  |  |
| **Estación de Pruebas (Cliente)** |  |  |
| Sistema Operativo |  |  |
| Herramienta de *testing* | *Rational TeamTest compuesta por Robot, ClearQuest TT, Test Manager* |  |
| Herramienta de Modelamiento | *Rational Rose 2000* |  |
| BDMA | *Access* |  |
| Browser | *Microsoft Explorer* |  |
| Software de Escritorio | *Microsoft Office* |  |
| **Base de Datos (pruebas)** |  |  |
| Sistema Operativo |  |  |
| Software de Red |  |  |
| Dominio/Cuenta |  |  |
|  |  |  |
| **Aplicación (pruebas)** |  |  |
| Sistema Operativo |  |  |
| Software de Red |  |  |
| Dominio/Cuenta |  |  |
|  |  |  |
| **Repositorios** |  |  |
| Servidor |  |  |
| Dominio / Cuenta |  |  |
|  |  |  |
| **Seguridad** |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Actividad** | **Responsable** | **Fecha Estimada** | **Fecha Real** | **Observaciones** |
|  |  |  |  |  |

|  | **Tarea** | **Responsable** | **Esfuerzo** | **Fecha Inicio** | **Fecha Término** |
| --- | --- | --- | --- | --- | --- |
|  | **Plan de Pruebas** |  |  |  |  |
|  | Identificar el Proyecto |  |  |  |  |
|  | Definir Estrategia |  |  |  |  |
|  | Estimar Actividades |  |  |  |  |
|  | Identificar Recursos |  |  |  |  |
|  | Documentar el “Plan de Pruebas” |  |  |  |  |
|  | Agendar de Actividades |  |  |  |  |
|  | Revisar el “Plan de Pruebas” |  |  |  |  |
|  | **Diseño de las Pruebas** |  |  |  |  |
|  | Analizar Requerimientos |  |  |  |  |
|  | Especificar Procedimientos de prueba |  |  |  |  |
|  | Especificar casos de prueba |  |  |  |  |
|  | Revisar Cobertura de los requerimientos de prueba |  |  |  |  |
|  | **Implementación de las Pruebas** |  |  |  |  |
|  | Establecer Ambiente de Implementación |  |  |  |  |
|  | Desarrollar los Procedimientos de Prueba |  |  |  |  |
|  | Probar y depurar los procedimientos de prueba |  |  |  |  |
|  | Modificar los procedimientos de prueba |  |  |  |  |
|  |  |  |  |  |  |
|  | **Ejecución de las pruebas** |  |  |  |  |
|  | Ejecutar pruebas |  |  |  |  |
|  | Comprobar resultados esperados |  |  |  |  |
|  | Investigar resultados inesperados |  |  |  |  |
|  | Registrar defectos (log) |  |  |  |  |
|  | Re-Ejecutar las pruebas |  |  |  |  |
|  | **Evaluación de las pruebas** |  |  |  |  |
|  | Revisar el Log de pruebas |  |  |  |  |
|  | Evaluar cobertura de los casos de prueba |  |  |  |  |
|  | Evaluar defectos |  |  |  |  |
|  | Reportar defectos |  |  |  |  |

|  |  |
| --- | --- |
| *✓* | “Plan de pruebas” (preliminar, al inicio del proyecto) |
|  | Identificar requerimientos para el testing |
|  | Identificar los riesgos, cuantificar impacto |
|  | Desarrollar la estrategia de pruebas |
|  | Identificar los recursos para las pruebas |
|  | Generar “Plan de Pruebas” detallado |
|  | Diseño general de las pruebas |
|  | Análisis de carga |
|  | Identificar y describir los casos de prueba |
|  | Identificar y estructurar los procedimientos de prueba |
|  | Revisar y accesar la cobertura de las pruebas |
|  | Implementar las pruebas |
|  | Grabar o programar los *scripts* de las pruebas, si aplica |
|  | Identificar las funcionalidades a probar, específicos en el modelo de diseño e implementación |
|  | Establecer el conjunto de datos externos |
|  | Ejecutar las pruebas |
|  | Ejecutar los procedimientos de prueba |
|  | Evaluar la ejecución de las pruebas |
|  | Comprobar los resultados |
|  | Investigar los resultados inesperados |
|  | Registro de defectos, Informe de Resultados |
|  | Evaluar las pruebas |
|  | Evaluar la cobertura de los casos de prueba |
|  | Evaluar la cobertura del código |
|  | Analizar defectos |

|  |  |
| --- | --- |
| Objetivo de la prueba: | Comprobar la conducta de rendimiento para las transacciones seleccionadas o funcionalidades bajo las siguientes condiciones:  - Una carga de trabajo normal.  - Una sobrecarga de trabajo. |
| Técnica a usar: | Usar los procedimientos de pruebas desarrollados para el testing funcional.  Modificar los archivos de datos para aumentar las transacciones o los *script* de robotización para incrementar el número de iteraciones de cada transacción.  Los *script* deberán correr en una máquina (la mejor referencia es un solo usuario y una única transacción) y repetirla con múltiples clientes (virtuales o reales). |
| Criterio de validación: | Una Transacción / Un Usuario: La finalización exitosa de los *scripts* de prueba sin ninguna falla dentro del tiempo esperado (por transacción en forma independiente).  Múltiples Transacciones / Múltiples Usuarios: La finalización exitosa de los *scripts* de prueba sin ninguna falla dentro tiempo estimado. |
| Consideraciones especiales: | La extensión del testing de rendimiento requiere tener en *background* la carga de trabajo en el servidor.  Existen varios métodos que se pueden usar para realizar esto como por ejemplo:  Gatillar transacciones directamente al servidor, normalmente en forma de llamadas de SQL.  Crear una carga de usuarios virtuales para simular (normalmente varios cientos) los clientes. Para esto se utilizan herramientas de emulación de terminales remotas para lograr esta carga. Esta técnica también puede usarse para someter a la red a un alto tráfico.  Usar múltiples clientes físicos, cada uno corriendo los Test scripts para agregar una carga al sistema.  El testing de rendimiento debería realizarse en una máquina dedicada o en un tiempo dedicado. Esto permite un control total y una exacta medición.  Las bases de datos utilizadas para realizar el testing de rendimiento deberán ser del tamaño equivalente a las de producción o a escala similar. |
| Observaciones: |  |

|  |  |
| --- | --- |
| Objetivo de la prueba: | Seguridad a Nivel de Aplicación: comprobar que un usuario puede acceder sólo a las funcionalidades y datos para las cuales ese tipo de usuario tiene permiso.  Seguridad a Nivel de Sistema: comprobar que sólo esos usuarios con acceso al sistema y aplicación tienen permitido el acceso. |
| Técnica a usar: | Nivel de Aplicación: Identifique y liste cada tipo de usuario y las funcionalidades y datos de cada tipo para las cuales tiene permiso.  Cree pruebas para cada tipo de usuario y verifique cada permiso creando transacciones específicas para cada usuario.  Modifique los tipos de usuarios y vuelva a ejecutar los casos de prueba para los mismos usuarios. En cada caso verifique si las funcionalidades y los datos están correctamente disponibles o denegados.  Acceso a Nivel de Sistema: vea las consideraciones especiales más abajo. |
| Criterio de validación: | Para cada tipo de usuario conocido, las funcionalidades y los datos correctos debieran estar disponibles y todas las transacciones ejecutadas debieran ejecutarse de acuerdo a lo esperado. |
| Consideraciones especiales: | El acceso al sistema debería ser comprobado con el administrador de la red o del sistema.  Este *testing* quizás pueda requerir de la participación del administrador de la red o del sistema. |
| Observaciones: |  |

PLAN DE PRUEBAS - PROYECTO FERREMAS

===================================

1. INFORMACIÓN GENERAL

======================

1.1 Identificación del Proyecto

- Nombre del Proyecto: FERREMAS - Sistema de Gestión de Ferretería

- Versión: 1.0.0

- Fecha de Creación: 15 de Diciembre de 2024

- Responsable de Pruebas: Equipo de Desarrollo

- Fecha de Última Actualización: 19 de Diciembre de 2024

1.2 Objetivo del Plan

Este plan de pruebas tiene como objetivo verificar que el sistema FERREMAS cumple con todos los requisitos funcionales y no funcionales especificados en la rúbrica del proyecto, incluyendo la implementación de API REST, gRPC, Server-Sent Events (SSE), y todas las funcionalidades del frontend.

1.3 Alcance de las Pruebas

- API REST para gestión de productos y sucursales

- Servidor gRPC para operaciones de productos

- Server-Sent Events para notificaciones en tiempo real

- Frontend React con validaciones y funcionalidades completas

- Integración entre todos los componentes del sistema

- Pruebas de rendimiento y seguridad

2. ESTRATEGIA DE PRUEBAS

=========================

2.1 Tipos de Pruebas

a) Pruebas Unitarias

- Pruebas de funciones individuales

- Pruebas de métodos de clases

- Pruebas de validaciones

b) Pruebas de Integración

- Pruebas de comunicación entre componentes

- Pruebas de API REST

- Pruebas de gRPC

- Pruebas de SSE

c) Pruebas de Sistema

- Pruebas de flujos completos

- Pruebas de interfaz de usuario

- Pruebas de base de datos

d) Pruebas de Rendimiento

- Pruebas de carga

- Pruebas de concurrencia

- Pruebas de tiempo de respuesta

e) Pruebas de Seguridad

- Pruebas de autenticación

- Pruebas de autorización

- Pruebas de validación de entrada

2.2 Criterios de Aceptación

- Todas las funcionalidades deben funcionar correctamente

- El sistema debe manejar errores de manera apropiada

- Los tiempos de respuesta deben ser aceptables (< 2 segundos)

- La seguridad debe estar implementada correctamente

- La documentación debe estar completa

3. AMBIENTE DE PRUEBAS

=======================

3.1 Configuración del Ambiente

- Sistema Operativo: Windows 10/11, Linux, macOS

- Base de Datos: PostgreSQL 13+

- Backend: Python 3.11+, Flask 2.3+

- Frontend: Node.js 16+, React 18+

- Redis: 6.0+ (para SSE)

- gRPC: Protocol Buffers 3.0+

3.2 Herramientas de Pruebas

- Python unittest para pruebas automatizadas

- Postman/Insomnia para pruebas de API

- Jest para pruebas de frontend

- Redis CLI para verificación de SSE

- gRPC CLI para pruebas de gRPC

3.3 Datos de Pruebas

- Usuarios de prueba con diferentes roles

- Productos de prueba con diferentes categorías

- Sucursales de prueba

- Datos de stock para validaciones

4. CASOS DE PRUEBA DETALLADOS

==============================

4.1 Pruebas de API REST (10 casos)

a) Caso 1: Respuesta 200 OK con datos válidos

- Objetivo: Verificar que los endpoints GET devuelven datos correctos

- Precondiciones: Backend corriendo en puerto 5000

- Pasos: Hacer GET request a endpoints de productos

- Resultado Esperado: Status 200 con datos válidos

b) Caso 2: Creación de recurso (POST)

- Objetivo: Verificar creación de productos

- Precondiciones: Usuario autenticado

- Pasos: POST request con datos válidos

- Resultado Esperado: Status 201, producto creado

c) Caso 3: Actualización de recurso (PUT/PATCH)

- Objetivo: Verificar actualización de productos

- Precondiciones: Producto existente

- Pasos: PUT request con datos actualizados

- Resultado Esperado: Status 200, datos actualizados

d) Caso 4: Eliminación de recurso (DELETE)

- Objetivo: Verificar eliminación de productos

- Precondiciones: Producto existente

- Pasos: DELETE request

- Resultado Esperado: Status 204, producto eliminado

e) Caso 5: Manejo de parámetros inválidos

- Objetivo: Verificar validación de datos

- Precondiciones: Backend corriendo

- Pasos: Enviar datos inválidos

- Resultado Esperado: Status 400, mensaje de error

f) Caso 6: Autenticación y autorización

- Objetivo: Verificar seguridad de endpoints

- Precondiciones: Endpoints protegidos

- Pasos: Acceso sin token y con token inválido

- Resultado Esperado: Status 401

g) Caso 7: Paginación y filtrado

- Objetivo: Verificar funcionalidades avanzadas

- Precondiciones: Productos en base de datos

- Pasos: GET con parámetros de paginación y filtros

- Resultado Esperado: Status 200, datos paginados

h) Caso 8: Prueba de carga

- Objetivo: Verificar rendimiento

- Precondiciones: Sistema funcionando

- Pasos: Múltiples requests simultáneos

- Resultado Esperado: Respuestas rápidas y consistentes

i) Caso 9: Prueba de CORS

- Objetivo: Verificar configuración CORS

- Precondiciones: Frontend y backend corriendo

- Pasos: Requests desde frontend

- Resultado Esperado: Sin errores CORS

j) Caso 10: Manejo de errores del servidor

- Objetivo: Verificar robustez del sistema

- Precondiciones: Sistema funcionando

- Pasos: Forzar errores internos

- Resultado Esperado: Errores manejados apropiadamente

4.2 Pruebas de gRPC (10 casos)

a) Caso 1: Conexión exitosa

- Objetivo: Verificar conectividad gRPC

- Precondiciones: Servidor gRPC corriendo

- Pasos: Crear canal y stub

- Resultado Esperado: Conexión establecida

b) Caso 2: Llamada a método válido

- Objetivo: Verificar funcionalidad básica

- Precondiciones: Método implementado

- Pasos: Llamada con parámetros válidos

- Resultado Esperado: Respuesta correcta

c) Caso 3: Llamada con parámetros inválidos

- Objetivo: Verificar validación

- Precondiciones: Cliente conectado

- Pasos: Enviar datos inválidos

- Resultado Esperado: Error apropiado

d) Caso 4: Timeout de llamada

- Objetivo: Verificar manejo de timeouts

- Precondiciones: Timeout configurado

- Pasos: Llamada con timeout corto

- Resultado Esperado: Error DEADLINE\_EXCEEDED

e) Caso 5: Prueba de carga

- Objetivo: Verificar rendimiento

- Precondiciones: Sistema funcionando

- Pasos: Múltiples llamadas simultáneas

- Resultado Esperado: Todas las llamadas completadas

f) Caso 6: Cancelación de llamada

- Objetivo: Verificar cancelación

- Precondiciones: Llamada en progreso

- Pasos: Cancelar llamada

- Resultado Esperado: Llamada cancelada

g) Caso 7: Autenticación y autorización

- Objetivo: Verificar seguridad

- Precondiciones: Métodos protegidos

- Pasos: Acceso sin credenciales

- Resultado Esperado: Error de autenticación

h) Caso 8: Manejo de errores del servidor

- Objetivo: Verificar robustez

- Precondiciones: Servidor funcionando

- Pasos: Forzar errores internos

- Resultado Esperado: Códigos de error correctos

i) Caso 9: Streaming bidireccional

- Objetivo: Verificar comunicación streaming

- Precondiciones: Método streaming implementado

- Pasos: Establecer streaming

- Resultado Esperado: Comunicación bidireccional

j) Caso 10: Interoperabilidad

- Objetivo: Verificar compatibilidad

- Precondiciones: Cliente en Python

- Pasos: Comunicación con servidor

- Resultado Esperado: Comunicación exitosa

4.3 Pruebas de Server-Sent Events (10 casos)

a) Caso 1: Conexión establecida correctamente

- Objetivo: Verificar conectividad SSE

- Precondiciones: Backend y Redis corriendo

- Pasos: Conectar cliente SSE

- Resultado Esperado: Status 200, headers correctos

b) Caso 2: Recepción de mensajes

- Objetivo: Verificar comunicación

- Precondiciones: SSE conectado

- Pasos: Enviar evento desde servidor

- Resultado Esperado: Evento recibido

c) Caso 3: Reconexión automática

- Objetivo: Verificar robustez

- Precondiciones: Conexión establecida

- Pasos: Interrumpir conexión

- Resultado Esperado: Reconexión automática

d) Caso 4: Orden de mensajes

- Objetivo: Verificar integridad

- Precondiciones: Múltiples eventos

- Pasos: Enviar eventos secuenciales

- Resultado Esperado: Orden mantenido

e) Caso 5: Manejo de datos JSON

- Objetivo: Verificar formato de datos

- Precondiciones: Eventos con datos JSON

- Pasos: Enviar eventos JSON

- Resultado Esperado: JSON parseado correctamente

f) Caso 6: Evento de error

- Objetivo: Verificar manejo de errores

- Precondiciones: Sistema funcionando

- Pasos: Forzar error en conexión

- Resultado Esperado: Error manejado apropiadamente

g) Caso 7: Prueba de rendimiento

- Objetivo: Verificar capacidad

- Precondiciones: Sistema funcionando

- Pasos: Enviar gran volumen de eventos

- Resultado Esperado: Procesamiento eficiente

h) Caso 8: Cerrar conexión desde servidor

- Objetivo: Verificar detección de desconexión

- Precondiciones: Conexión activa

- Pasos: Cerrar conexión desde servidor

- Resultado Esperado: Cliente detecta desconexión

i) Caso 9: Manejo de múltiples conexiones

- Objetivo: Verificar escalabilidad

- Precondiciones: Sistema funcionando

- Pasos: Conectar múltiples clientes

- Resultado Esperado: Todas las conexiones funcionando

j) Caso 10: Tipos de eventos específicos

- Objetivo: Verificar especificidad

- Precondiciones: Diferentes tipos de eventos

- Pasos: Enviar eventos específicos

- Resultado Esperado: Eventos distinguidos correctamente

4.4 Pruebas de Frontend (5 casos)

a) Caso 1: Validación de stock en compra

- Objetivo: Verificar validación de stock

- Precondiciones: Productos con stock limitado

- Pasos: Intentar agregar más del stock disponible

- Resultado Esperado: Mensaje de error, no se agrega

b) Caso 2: Validación de valor mayor que cero

- Objetivo: Verificar validación de cantidades

- Precondiciones: Formulario de compra

- Pasos: Intentar agregar cantidad 0 o negativa

- Resultado Esperado: Mensaje de error

c) Caso 3: Pantalla de creación de producto

- Objetivo: Verificar CRUD de productos

- Precondiciones: Usuario autenticado

- Pasos: Crear producto con datos válidos

- Resultado Esperado: Producto creado, redirección

d) Caso 4: Select con valores de sucursal

- Objetivo: Verificar carga de sucursales

- Precondiciones: Sucursales en base de datos

- Pasos: Cargar página con select de sucursales

- Resultado Esperado: Lista de sucursales cargada

e) Caso 5: Notificaciones SSE en frontend

- Objetivo: Verificar notificaciones en tiempo real

- Precondiciones: SSE conectado

- Pasos: Recibir notificación de stock bajo

- Resultado Esperado: Notificación visible en UI

4.5 Pruebas de Integración (5 casos)

a) Caso 1: Flujo completo de compra

- Objetivo: Verificar integración completa

- Precondiciones: Todos los servicios corriendo

- Pasos: Completar flujo de compra

- Resultado Esperado: Proceso exitoso end-to-end

b) Caso 2: Creación de producto integrada

- Objetivo: Verificar integración de CRUD

- Precondiciones: Sistema completo funcionando

- Pasos: Crear producto desde frontend

- Resultado Esperado: Producto creado en todas las capas

c) Caso 3: Alertas de stock integradas

- Objetivo: Verificar sistema de alertas

- Precondiciones: Múltiples clientes conectados

- Pasos: Reducir stock a nivel bajo

- Resultado Esperado: Alertas enviadas a todos los clientes

d) Caso 4: Conversión de moneda integrada

- Objetivo: Verificar servicio de moneda

- Precondiciones: Servicio de moneda configurado

- Pasos: Realizar conversión en proceso de pago

- Resultado Esperado: Conversión exitosa

e) Caso 5: Autenticación integrada

- Objetivo: Verificar seguridad completa

- Precondiciones: Sistema con autenticación

- Pasos: Acceder a recursos protegidos

- Resultado Esperado: Acceso controlado apropiadamente

5. CRONOGRAMA DE PRUEBAS

=========================

5.1 Fase 1: Preparación (Días 1-2)

- Configuración del ambiente de pruebas

- Instalación de herramientas

- Preparación de datos de prueba

- Revisión de documentación

5.2 Fase 2: Pruebas Unitarias (Días 3-5)

- Ejecución de pruebas unitarias

- Corrección de defectos críticos

- Documentación de resultados

5.3 Fase 3: Pruebas de Integración (Días 6-10)

- Pruebas de API REST

- Pruebas de gRPC

- Pruebas de SSE

- Corrección de defectos

5.4 Fase 4: Pruebas de Sistema (Días 11-13)

- Pruebas de frontend

- Pruebas de integración completa

- Pruebas de flujos de usuario

5.5 Fase 5: Pruebas de Rendimiento y Seguridad (Días 14-15)

- Pruebas de carga

- Pruebas de seguridad

- Optimización de rendimiento

5.6 Fase 6: Documentación y Cierre (Día 16)

- Documentación de resultados

- Generación de reportes

- Cierre del plan de pruebas

6. CRITERIOS DE EXITOSIDAD

===========================

6.1 Criterios de Aceptación

- 100% de los casos de prueba deben pasar

- Tiempo de respuesta promedio < 2 segundos

- 0 defectos críticos o de alta severidad

- Cobertura de código > 90%

- Documentación completa

6.2 Criterios de Suspensión

- Más de 5 defectos críticos

- Tiempo de respuesta promedio > 5 segundos

- Fallas en funcionalidades core

- Problemas de seguridad críticos

6.3 Criterios de Reanudación

- Corrección de defectos críticos

- Verificación de funcionalidades core

- Aprobación del equipo de desarrollo

7. RIESGOS Y CONTINGENCIAS

===========================

7.1 Riesgos Identificados

- Servidores no disponibles

- Problemas de conectividad de red

- Dependencias externas no disponibles

- Limitaciones de tiempo

- Falta de datos de prueba

7.2 Planes de Contingencia

- Ambiente de respaldo configurado

- Datos de prueba alternativos

- Herramientas de prueba alternativas

- Extensión de cronograma si es necesario

- Escalamiento de recursos

8. RECURSOS REQUERIDOS

=======================

8.1 Recursos Humanos

- 1 Líder de Pruebas

- 2 Testers Senior

- 1 Desarrollador de Soporte

- 1 DBA de Soporte

8.2 Recursos Técnicos

- Servidores de prueba

- Herramientas de automatización

- Licencias de software

- Acceso a bases de datos

- Herramientas de monitoreo

8.3 Recursos de Infraestructura

- Ambiente de desarrollo

- Ambiente de pruebas

- Ambiente de staging

- Herramientas de CI/CD

9. ENTREGABLES

===============

9.1 Documentos de Pruebas

- Plan de pruebas (este documento)

- Casos de prueba detallados

- Scripts de automatización

- Datos de prueba

9.2 Reportes de Pruebas

- Reporte de ejecución diario

- Reporte de defectos

- Reporte de métricas

- Reporte final de pruebas

9.3 Artefactos de Pruebas

- Scripts de configuración

- Datos de prueba

- Herramientas de automatización

- Documentación de configuración

10. APROBACIONES

=================

10.1 Aprobaciones Requeridas

- Aprobación del Líder de Proyecto

- Aprobación del Arquitecto de Software

- Aprobación del Líder de Desarrollo

- Aprobación del Líder de QA

10.2 Firma de Aprobación

- Nombre: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

- Cargo: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

- Fecha: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

- Firma: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

RESUMEN EJECUTIVO

==================

Este plan de pruebas cubre completamente todos los requisitos del proyecto FERREMAS, incluyendo:

✅ API REST para obtener productos x sucursal (5 puntos)

✅ Select con valores de sucursal (2 puntos)

✅ API para conversión de dólar (5 puntos)

✅ Validación de stock en compra (2 puntos)

✅ Validación valor mayor que cero (2 puntos)

✅ Server-Sent Events de stock bajo (15 puntos)

✅ Pantalla de creación de producto (5 puntos)

✅ Revisión de gRPC Bridge (10 puntos)

✅ Creación servidor gRPC (20 puntos)

Total de casos de prueba: 40

Cobertura estimada: 100%

Duración estimada: 16 días

Recursos estimados: 5 personas

El proyecto está listo para la ejecución de pruebas con una estrategia completa y detallada que garantiza la calidad del software entregado.