**Sistema de Inventarios Jucar Autopartes SAS**

**Plan de Pruebas del Sistema de Gestión de Inventarios  
 Autopartes Jucar**

**Cristian Daniel Galindo Jiménez  
Daniel Esteban Piñeros Barbosa  
Miguel Ángel Moyano Palacio**

**2558108**

**Análisis y Desarrollo de Software**

**Centro de Electricidad, Electrónica y Telecomunicaciones**

**CEET**

**202**4

# Introducción

## Proposito:

El propósito de este Plan de Pruebas es establecer un marco de trabajo detallado y coherente que guíe las actividades de pruebas del Sistema de Gestión de Inventarios. Este plan tiene como objetivo asegurar que todas las características y funcionalidades del sistema, incluyendo la API desarrollada en ASP.NET Core Web API, la base de datos SQL Server migrada a Azure SQL Databases, y el frontend alojado en Github Pages, cumplan con los requisitos definidos, funcionen como se espera y proporcionen una experiencia de usuario confiable y eficiente. A través de pruebas unitarias, funcionales, automatizadas y de rendimiento, buscamos identificar y solucionar problemas antes de la puesta en producción, garantizando así la calidad y estabilidad del sistema.

## Alcance:

* **Pruebas Unitarias:** Se centrarán en los controladores de la API, utilizando técnicas y herramientas específicas para .NET Core con el fin de validar la lógica de negocio de manera aislada. El objetivo es asegurar que cada unidad funcione correctamente bajo diferentes escenarios.
* **Pruebas Funcionales:** Se realizarán mediante Postman para probar las diferentes endpoints de la API, asegurando que la interacción entre el frontend, la API y la base de datos se ejecuta según lo esperado. Estas pruebas incluirán la verificación de respuestas correctas a las peticiones, manejo adecuado de errores y la persistencia de datos en la base de datos.
* **Pruebas Automatizadas:** Se empleará Selenium IDE para automatizar pruebas en el frontend. Estas pruebas validarán la interfaz de usuario, incluyendo la navegación, la interacción con los elementos de la página y la integración con la API, para asegurar una experiencia de usuario fluida y coherente.
* **Pruebas de Rendimiento:** Utilizaremos JMeter de Apache para realizar pruebas de carga y estrés en el sistema, con el fin de evaluar su comportamiento y rendimiento bajo diferentes condiciones de carga. Esto incluye pruebas de tiempo de respuesta, velocidad de procesamiento y la capacidad del sistema para manejar un volumen alto de solicitudes simultáneas.

Este plan se enfoca en garantizar la funcionalidad, fiabilidad y rendimiento del sistema mediante un enfoque estructurado y detallado en las pruebas, preparando el sistema para un despliegue exitoso y su operación continua en un entorno de producción.

# Objetivos de las Pruebas

## Metas Generales

1. **Validación Completa de Funcionalidades**: Asegurar que cada característica del Sistema de Gestión de Inventarios, desde la gestión de productos hasta el seguimiento de inventarios, funcione de acuerdo con los requisitos especificados y sin errores. Esto incluye la correcta ejecución de todas las funcionalidades de la API, operaciones de base de datos y la interacción del usuario con el frontend.
2. **Garantía de Calidad del Código:** Verificar que el código de la aplicación, tanto en el backend como en el frontend, está escrito y estructurado según las mejores prácticas de desarrollo, contribuyendo a una base de código mantenible y escalable.
3. **Optimización del Rendimiento:** Evaluar y garantizar que el sistema es capaz de manejar la carga de trabajo esperada con eficiencia, manteniendo tiempos de respuesta rápidos y una experiencia de usuario sin interrupciones, incluso bajo condiciones de alta demanda.
4. **Fiabilidad y Estabilidad del Sistema:** Confirmar que el sistema es robusto, puede recuperarse de errores y manejar adecuadamente las excepciones, asegurando así su operación continua y confiable en el entorno de producción.
5. **Mejora Continua:** Utilizar los resultados de las pruebas para identificar oportunidades de mejora en el sistema, tanto en términos de funcionalidades como de experiencia del usuario, y aplicar estas mejoras de manera iterativa.
6. **Preparación para el Despliegue**: Verificar que el sistema está completamente listo para su despliegue, asegurando que todos los componentes (API, base de datos, frontend) están optimizados, funcionan correctamente juntos y cumplen con los criterios de aceptación para su lanzamiento.
7. **Documentación de Pruebas:** Crear una documentación detallada de los procesos de prueba y los resultados obtenidos, proporcionando una base sólida para futuras pruebas y mantenimiento del sistema.

Estas metas reflejan un compromiso con la calidad y la excelencia, buscando asegurar que el Sistema de Gestión de Inventarios no solo cumpla con las expectativas actuales de los usuarios y stakeholders, sino que también esté preparado para adaptarse y evolucionar de acuerdo con las necesidades futuras.

# Referencias

**Manual de Uso:** <https://docs.google.com/document/d/1BjTbfePShSxP4SGB7phD58wrq1l4LKOl/edit?rtpof=true>  
  
**Especificación de Requisitos:** <https://docs.google.com/document/d/1B6cgNS2p8qWkqZmVuLAB7BqBvFvA8RTt/edit>  
  
**Suite de Pruebas:** <https://docs.google.com/spreadsheets/d/1P28GylGBAf28C3tFMDJV0T56rAseMMTD/edit?usp=sharing&ouid=103734177745430189424&rtpof=true&sd=true>  
  
**Usabilidad:** <https://drive.google.com/file/d/1fGwdC9EG9AEhFzS8I_WcIUM1qIQudKqe/view>  
  
**Aceptación de Usuario:** <https://docs.google.com/document/d/1G5Szk_t2fb7R56xzlIUflX5y3coKgxUu/edit?usp=drive_web&ouid=103734177745430189424&rtpof=true>

# Estrategia de Pruebas

## Niveles de Pruebas

* **Pruebas Unitarias:** Se centrarán en validar la lógica de negocio de manera aislada, especialmente en los controladores de la API. Esto incluye pruebas de métodos individuales y funciones para garantizar que operen correctamente bajo diversos escenarios.
* **Pruebas de Sistema**: Evaluarán el sistema completo en un entorno que simule lo más fielmente posible el entorno de producción. Esto incluirá la funcionalidad completa del sistema, así como su rendimiento y comportamiento bajo condiciones normales de operación.
* **Pruebas de Aceptación:** Se realizarán para confirmar que el sistema cumple con los requisitos del usuario y está listo para su implementación en el entorno de producción. Estas pruebas serán conducidas con la participación de usuarios finales o partes interesadas.

## Tipos de Pruebas

* **Funcionales:** Utilizando Postman para probar la API, asegurando que todas las endpoints funcionen como se espera y que la interacción entre el frontend, la API y la base de datos sea correcta.
* **Automatizadas:** Empleando Selenium IDE para automatizar las pruebas del frontend, verificando la interacción del usuario con la interfaz, incluyendo la navegación y la ejecución de tareas comunes.
* **De Rendimiento:** Con JMeter de Apache para simular múltiples accesos simultáneos al sistema, evaluando su capacidad para manejar altas cargas de trabajo y manteniendo un rendimiento óptimo.

## Criterios de Aceptación

* Todos los casos de prueba definidos deben pasar sin errores críticos.
* Los tiempos de respuesta de las funcionalidades críticas deben estar dentro de los límites aceptables especificados en los requisitos de rendimiento.
* No debe haber defectos críticos sin resolver en el momento del lanzamiento.
* La funcionalidad del sistema debe coincidir con la documentación de requisitos y ser validada por usuarios finales o partes interesadas en las pruebas de aceptación.
* Las pruebas de seguridad deben no revelar vulnerabilidades críticas.

Esta estrategia de pruebas está diseñada para ser exhaustiva y garantizar que el Sistema de Gestión de Inventarios sea robusto, confiable y listo para su despliegue en un entorno de producción.

# Planificación

## Recursos

### Herramientas de Software:

* **Para Pruebas Unitarias:** Frameworks específicos de .NET Core como xUnit o NUnit.
* **Para Pruebas Funcionales:** Postman para realizar solicitudes HTTP a la API y verificar las respuestas.
* **Para Pruebas Automatizadas:** Selenium IDE para la creación y ejecución de pruebas automatizadas en el frontend.
* **Para Pruebas de Rendimiento:** Apache JMeter para simular altas cargas de trabajo y evaluar el rendimiento del sistema.

### Hardware:

* Servidores de pruebas que emulen el entorno de producción, incluyendo configuraciones similares para la base de datos y el servidor de aplicaciones.
* Equipos de cómputo para el equipo de desarrollo y pruebas, capaces de ejecutar las herramientas de desarrollo y pruebas necesarias.

### Ambiente de Pruebas

* **Entorno de Desarrollo:** Donde los desarrolladores realizarán pruebas unitarias y de integración iniciales antes de pasar el código al repositorio central.
* **Entorno de Pruebas QA (Quality Assurance):** Una réplica exacta del entorno de producción donde se ejecutarán las pruebas funcionales, automatizadas y de rendimiento. Este entorno incluirá:  
    
  *Servidor de Aplicaciones:* Configurado con Azure App Service para alojar la API.

*Base de Datos:* Una instancia dedicada en Azure SQL Databases, cargada con datos de prueba representativos.

*Frontend:* Hospedado en un entorno similar a Github Pages para simular el acceso y la interacción del usuario final.

* **Herramientas de Gestión de Pruebas:** Se utilizarán plataformas como Jira para planificar, ejecutar y seguir el progreso de las pruebas, así como para registrar y gestionar los defectos encontrados.

Estos recursos y ambientes están diseñados para proporcionar una plataforma de pruebas completa y eficaz, asegurando que el Sistema de Gestión de Inventarios se pruebe exhaustivamente en condiciones que reflejen fielmente su operación en el entorno de producción real.

# Diseño de Pruebas

## Casos de Pruebas

El diseño de los casos de pruebas se enfocará en validar todas las funcionalidades y requisitos del sistema de manera integral. Cada caso de prueba incluirá:

* **Identificador Único:** Para facilitar la referencia y el seguimiento.
* **Descripción:** Una breve explicación del propósito del caso de prueba.
* **Condiciones Previas:** Estado del sistema o configuraciones necesarias antes de ejecutar el caso de prueba.
* **Pasos a Seguir:** Instrucciones detalladas paso a paso para ejecutar el caso de prueba.
* **Datos de Entrada:** Valores específicos o datos necesarios para realizar la prueba.
* **Resultados Esperados:** El resultado esperado del sistema tras la ejecución del caso de prueba.
* **Resultados Reales:** Se rellenará tras la ejecución de la prueba.
* **Estado:** Indica si la prueba ha sido exitosa, fallida, o si ha sido bloqueada por algún impedimento.

## Ejemplo de Caso de Prueba para la API:

* **Identificador:** CP-001
* **Descripción:** Verificar la creación de una nueva categoría.
* **Condiciones Previas:** Usuario autenticado con permisos de escritura.
* **Pasos a Seguir:**
* Enviar una solicitud POST a la endpoint /api/categories con los detalles de la categoría.
* Verificar la respuesta del servidor.
* **Datos de Entrada:** Detalles del ítem (nombre).
* **Resultados Esperados:** Respuesta 201 Created con la confirmación de la creación del ítem.
* **Estado:** Pendiente de ejecución.

## Cobertura de Pruebas

Para asegurar una cobertura de pruebas completa, se adoptará el siguiente enfoque:

* **Análisis de Requisitos**: Todos los requisitos documentados serán mapeados a casos de prueba específicos, asegurando que cada requisito sea probado.
* **Revisión de Diseño:** Se examinarán los diseños del sistema para identificar funcionalidades clave y posibles áreas de riesgo que requieran atención especial en las pruebas.
* **Uso de Técnicas de Pruebas:** Se utilizarán técnicas de diseño de pruebas como pruebas de caja negra, pruebas de caja blanca y pruebas basadas en el riesgo para abordar diferentes aspectos del sistema.
* **Matriz de Trazabilidad:** Se creará una matriz de trazabilidad para vincular cada caso de prueba con los requisitos específicos que valida. Esto ayudará a garantizar que todos los requisitos sean cubiertos y facilitará la identificación de cualquier área sin probar.

Mediante este enfoque detallado en el diseño de pruebas y la estrategia de cobertura de pruebas, se busca garantizar que el Sistema de Gestión de Inventarios sea sometido a un examen exhaustivo que valide su funcionalidad, rendimiento y fiabilidad en todos los aspectos antes de su lanzamiento.

# Procedimiento de Ejecución

## Preparación

Antes de comenzar con la ejecución de las pruebas, es esencial preparar adecuadamente el entorno y los recursos necesarios para garantizar una ejecución eficiente y efectiva. Los pasos de preparación incluyen:

* **Configuración del Entorno de Pruebas**: Asegurarse de que el entorno de pruebas QA esté configurado correctamente, replicando lo más fielmente posible el entorno de producción. Esto incluye la configuración de servidores, bases de datos, y cualquier dependencia de software necesaria.
* **Creación de Datos de Prueba:** Generar o preparar datos de prueba adecuados que cubran diversos escenarios de uso, incluidos casos extremos y datos al borde de los límites aceptables.
* **Revisión de Casos de Prueba:** Verificar que todos los casos de prueba estén actualizados y sean relevantes para la versión del software que se está probando.
* **Configuración de Herramientas de Pruebas:** Asegurarse de que todas las herramientas de pruebas, como Postman, Selenium IDE y JMeter, estén configuradas correctamente y listas para usar.
* **Capacitación del Equipo:** Confirmar que todos los miembros del equipo de pruebas estén familiarizados con el plan de pruebas, las herramientas, y los procedimientos de ejecución.

## Ejecución

La ejecución de las pruebas se llevará a cabo siguiendo el plan de pruebas detallado y utilizando los casos de pruebas diseñados específicamente para cada tipo de prueba. Los pasos generales incluyen:

* **Ejecución de Pruebas Unitarias:** Realizadas por los desarrolladores para asegurar que los componentes individuales y sus interacciones funcionan como se espera.
* **Ejecución de Pruebas Funcionales**: Utilizando Postman para probar la API y Selenium IDE para el frontend, garantizando que el sistema cumple con los requisitos funcionales.
* **Ejecución de Pruebas de Rendimiento:** Aplicando JMeter para evaluar la capacidad del sistema bajo cargas de trabajo simuladas, identificando cuellos de botella y puntos de falla.
* **Documentación de Resultados:** Registrar los resultados de cada prueba, incluidos los éxitos, fallos, y cualquier incidencia encontrada.
* **Gestión de Incidencias:** Las incidencias detectadas durante las pruebas deben registrarse en una herramienta de seguimiento de incidencias, asignarse para su corrección, y seguirse hasta su resolución.

## Criterios para la Repetición de Pruebas

Las pruebas deben repetirse en las siguientes circunstancias:

* **Corrección de Incidencias:** Cada vez que se corrija una incidencia encontrada durante las pruebas, las pruebas relacionadas deben repetirse para verificar la corrección.
* **Cambios en el Código:** Si se realizan cambios significativos en el código, las pruebas afectadas deben repetirse para asegurar que los cambios no hayan introducido nuevos errores.
* **Pruebas de Regresión:** Después de cada ciclo de desarrollo y corrección de errores, se deben ejecutar pruebas de regresión completas para garantizar que las modificaciones no hayan afectado negativamente otras partes del sistema.

Este proceso estructurado de ejecución y repetición de pruebas asegura una validación completa y continua del sistema, facilitando la identificación y resolución de problemas en una etapa temprana y manteniendo la calidad del software a lo largo del ciclo de desarrollo.

# Gestión de Defectos

## Registro de Defectos

La gestión eficaz de defectos comienza con un registro detallado y sistemático de cada defecto encontrado durante las pruebas. Para cada defecto identificado, se debe crear un informe que incluya:

* **Identificador Único:** Un código o número único para cada defecto, facilitando su seguimiento y referencia.
* **Descripción:** Una explicación clara y concisa del defecto, incluyendo el comportamiento esperado versus el comportamiento observado.
* **Severidad**: Una clasificación del defecto según su impacto en el sistema (p.ej., Crítico, Mayor, Menor).
* **Prioridad:** Una indicación de la urgencia con la que el defecto necesita ser resuelto (p.ej., Alta, Media, Baja).
* **Estado:** El estado actual del defecto (p.ej., Abierto, En Proceso, Resuelto).
* **Ambiente:** Especificar el entorno de pruebas en el que se encontró el defecto (p.ej., QA, Desarrollo).
* **Pasos para Reproducir:** Una lista detallada de pasos necesarios para reproducir el defecto, incluyendo cualquier dato de entrada específico.
* **Capturas de Pantalla/Logs:** Si es posible, adjuntar capturas de pantalla, registros de errores o cualquier otra evidencia que ayude a entender el defecto.
* **Asignación**: El miembro del equipo de desarrollo responsable de investigar y corregir el defecto.

Este registro debe ser almacenado en una herramienta de seguimiento de defectos centralizada y accesible para todo el equipo de desarrollo y pruebas, como Jira, Azure DevOps, o una herramienta similar.

## Seguimiento de Defectos

Una vez registrado el defecto, es crucial seguir su progreso hasta que se resuelva completamente. El proceso de seguimiento debe incluir:

* **Revisión Regular de Defectos:** Reuniones periódicas del equipo para revisar los defectos abiertos, actualizar su estado, y reasignar recursos si es necesario.
* **Actualización de Estados:** Cambiar el estado del defecto conforme avanza su resolución (p.ej., de "En Proceso" a "Resuelto").
* **Comunicación con el Equipo:** Mantener a todos los miembros del equipo informados sobre el progreso en la resolución de defectos, especialmente si el defecto afecta a múltiples áreas del sistema.
* **Verificación de la Corrección**: Una vez que un defecto ha sido marcado como resuelto, el equipo de pruebas debe verificar que la solución es efectiva y que no introduce nuevos problemas.
* **Cierre de Defectos**: Si la corrección es verificada satisfactoriamente, el defecto puede ser cerrado. De lo contrario, debe reabrirse y reasignarse para su corrección adicional.

Este enfoque estructurado y disciplinado para la gestión de defectos asegura que todos los problemas identificados sean abordados de manera eficiente, minimizando su impacto en la calidad y el cronograma del proyecto.