**PLAN DE PRUEBAS DETALLADO**

**Proyecto: Sistema Web SPP**

**Versión del Documento: 1.1**

**Fecha de Creación: 18 de febrero de 2025**

**Responsable: María Alejandra Escobar – Sebastian Garzón – Samuel Diaz**

**1️. Introducción**

**1.1 Objetivo**

El objetivo de este plan de pruebas es garantizar la calidad y estabilidad del Sistema Web (backend) mediante la ejecución de pruebas exhaustivas en cada fase del desarrollo y en un entorno DevOps, asegurando calidad con pruebas integradas en la pipeline CI/CD.

**1.2 Alcance**

El objetivo principal de este plan de pruebas es garantizar la calidad del sistema web mediante la detección temprana de errores, asegurando su funcionalidad, seguridad, rendimiento y compatibilidad en diferentes entornos.

**1.3 Funcionalidades Críticas a Evaluar**

✔ **Gestión de Productos**: Crear, editar y eliminar productos, ver detalles y aplicar filtros de búsqueda.  
✔ **Gestión de Pedidos**: Generar, modificar, confirmar y cancelar pedidos, además de rastrear su estado.  
✔ **Gestión de Usuarios**: Registrar, iniciar sesión, recuperar contraseña y administrar datos personales y roles.  
✔ **Gestión de Proveedores**: Registrar, actualizar y vincular proveedores con productos, además de consultar su historial.  
✔ **Lista de Favoritos**: Agregar, eliminar y gestionar productos marcados como favoritos

.**2️. Estrategia de Pruebas**

**2.1 Niveles de Prueba**

| **Nivel** | **Descripción** |
| --- | --- |
| **Pruebas Unitarias** | Validan métodos y funciones individuales. |
| **Pruebas de Integración** | Verificar la comunicación entre módulos como API y base de datos. |
| **Pruebas de Sistema** | Evalúan el sistema completo con datos reales. |
| **Pruebas de Aceptación** | Probar el sistema con usuarios antes del despliegue. |

**2.2 Tipos de Pruebas**

| **Tipo** | **Descripción** |
| --- | --- |
| **Funcionales** | Validar que la gestión de productos, pedidos, usuarios, proveedores y favoritos funcione correctamente. |
| **No funcionales** | Analizan la velocidad, estabilidad y facilidad de uso en la gestión de productos, pedidos, usuarios, proveedores y favoritos, garantizando una experiencia eficiente y segura. |
| **Regresión** | Asegurar que las modificaciones en productos, pedidos o usuarios no afecten otras funciones. |
| **Carga y Estrés** | Simulan usuarios concurrentes para medir el rendimiento. |
| **Seguridad** | Detectan vulnerabilidades en autenticación y datos sensibles. |

**Pruebas de Humo** Verifican que las funcionalidades básicas del sistema estén operativas después de cada despliegue.

**Pruebas de Contenido** Evalúan que los textos, etiquetas y datos visibles sean correctos, completos y estén alineados con los requisitos del usuario.

**Pruebas de Base de Datos** Validan la consistencia, integridad y rendimiento de las operaciones directamente en la base de datos.

**3️. Recursos y Herramientas**

**3.1 Equipo de Pruebas**

| **Rol** | **Responsabilidades** |
| --- | --- |
| **QA Analyst** | Diseña y ejecuta pruebas para validar la gestión de productos, pedidos, usuarios, proveedores y lista de favoritos. |
| **QA Automation** | Implementa pruebas automatizadas para agilizar la validación de funcionalidades críticas. |
| **Desarrollador** | Corrige errores detectados en pruebas y optimiza el código según los resultados. |

**3.2 Herramientas de Pruebas**

| **Tipo** | **Herramienta** |
| --- | --- |
| Unitarias | Jest, JUnit, Mocha |
| API | Postman, Newman |
| UI/E2E | Cypress, Selenium |
| Seguridad | OWASP ZAP, SonarQube |
| Carga | JMeter, k6 |

**4️. Criterios de Aceptación y Salida**

**4.1 Criterios de Aceptación**

✅ Todos los endpoints deben responder correctamente en Postman.  
✅ Las consultas en SQL Server deben ejecutarse sin errores ni demoras excesivas.  
✅ No se permiten errores críticos en la gestión de productos, pedidos, usuarios y proveedores.

**4.2 Criterios de Salida**

✅ Se han corregido todos los errores críticos detectados en Postman y SQL Server.  
✅ Las pruebas de integración y funcionales han sido validadas con éxito.  
✅ El sistema cumple con los tiempos de respuesta esperados en consultas y transacciones.

**5️. Plan de Ejecución**

**5.1 Cronograma de Pruebas**

| **Fase** | **Duración** | **Responsable** |
| --- | --- | --- |
| Diseño de casos de prueba | 1 semana | QA Analyst |
| Pruebas en Postman (API y flujo) | 1 semana | QA Team |
| Pruebas en SQL Server (Consultas) | 1 semana | Desarrolladores |
| Pruebas unitarias | Continuo en CI/CD | Desarrolladores |
| Pruebas de integración | 1 semana | QA Team |
| Pruebas de sistema | 1 semana | QA Team |
| Pruebas de aceptación | 1 semana | Usuarios finales |

**6️. Casos de Prueba Detallados**

**Funcionalidad: Casos de Prueba Detallados (Base de Datos - Usuarios)**

| **ID** | **Escenario** | **Precondiciones** | **Entrada** | **Resultado Esperado** | **Prioridad** |
| --- | --- | --- | --- | --- | --- |
| TC001 | Crear Usuario | Base de datos activa | Datos validos | Usuario registrado | Alta |
| TC002 | Leer Usuario | Usuario existente | ID válido | Datos del usuario | Alta |
| TC003 | Actualizar Usuario | Usuario existente | ID y nuevos datos | Datos actualizados | Media |
| TC004 | Eliminar Usuario | Usuario existente | ID válido | |  | | --- | | Usuario eliminado |  |  | | --- | |  | | Alta |

**Proveedores**

| **ID** | **Escenario** | **Precondiciones** | **Entrada** | **Resultado Esperado** | **Prioridad** |
| --- | --- | --- | --- | --- | --- |
| TC001 | Crear proveedor | Base de datos activa | Datos válidos | Proveedor registrado | Alta |
| TC002 | Leer proveedor | Proveedor existente | ID válido | Datos del proveedor | Alta |
| TC003 | Actualizar proveedor | Proveedor existente | ID y nuevos datos | Datos actualizados | Alta |
| TC004 | Eliminar proveedor | Proveedor existente | ID válido | Proveedor eliminado | Alta |

**Favoritos**

| **ID** | **Escenario** | **Precondiciones** | **Entrada** | **Resultado Esperado** | **Prioridad** |
| --- | --- | --- | --- | --- | --- |
| TC005 | Agregar a favoritos | Usuario y producto existentes | ID de producto | Producto agregado a favoritos | Alta |
| TC006 | Ver favoritos | Usuario con productos en favoritos | ID de usuario | Lista de favoritos mostrada | Alta |
| TC007 | Eliminar de favoritos | Producto en favoritos | ID de producto | Producto eliminado de favoritos | Alta |

**Productos**

| **ID** | **Escenario** | **Precondiciones** | **Entrada** | **Resultado Esperado** | **Prioridad** |
| --- | --- | --- | --- | --- | --- |
| TC008 | Crear producto | Base de datos activa | Datos válidos | Producto registrado | Alta |
| TC009 | Leer producto | Producto existente | ID válido | Datos del producto | Alta |
| TC010 | Actualizar producto | Producto existente | ID y nuevos datos | Datos actualizados | Alta |
| TC011 | Eliminar producto | Producto existente | ID válido | Producto eliminado | Alta |

**Funcionalidad: Pruebas de Humo**

| **ID** | **Escenario** | **Precondiciones** | **Entrada** | **Resultado Esperado** | **Prioridad** |
| --- | --- | --- | --- | --- | --- |
| TC100 | Verificar carga inicial del sistema | Despliegue exitoso | Acceso a sistema | Página principal cargada sin errores críticos | Alta |

**Funcionalidad: Pruebas de Contenido**

| **ID** | **Escenario** | **Precondiciones** | **Entrada** | **Resultado Esperado** | **Prioridad** |
| --- | --- | --- | --- | --- | --- |
| TC110 | Validar textos de botones | Sistema cargado | Navegación UI | Todos los botones muestran etiquetas correctas y claras | Media |

**7️. Gestión de Defectos**

Los errores detectados serán reportados y gestionados manualmente en un documento de seguimiento, con las siguientes prioridades:

| **Severidad** | **Impacto** |
| --- | --- |
| **Crítica** | Impide completamente el acceso o modificación de datos en la base de datos. Debe corregirse antes del despliegue. |
| **Alta** | Afecta operaciones principales como creación, lectura, actualización o eliminación de proveedores, productos o favoritos. Requiere corrección inmediata. |
| **Media** | Impacta funcionalidades secundarias, como consultas específicas o validaciones de datos. Se corrige en próximas versiones. |
| **Baja** | Errores menores en formatos o mensajes, sin afectar la integridad de los datos. Se corrige según prioridad del equipo. |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**PLAN DE PRUEBAS ENFOCADO EN DEVOPS Y CI/CD**

**1. Introducción**

Este documento define el Plan de Pruebas para una **aplicación web** en un entorno DevOps, asegurando calidad en cada fase del desarrollo mediante **pruebas automatizadas y continuas**.

* **Objetivo:** Garantizar la calidad del software con pruebas integradas en la pipeline CI/CD.
* **Alcance:** Pruebas desde la fase de desarrollo hasta la producción, asegurando funcionalidad, seguridad y rendimiento.

**2. Estrategia de Pruebas**

La estrategia está basada en el modelo **Shift Left Testing**, aplicando pruebas desde las primeras etapas del desarrollo.

| Fase | Tipo de Prueba | Objetivo | Estrategia |
| --- | --- | --- | --- |
| Desarrollo | Pruebas Unitarias | Validar operaciones CRUD en BD | Scripts SQL ejecutados en cada commit |
| Integración Continua | Pruebas de Integración | Verificar la conexión entre tablas y consultas | Postman y SQL Server en cada push |
| Preproducción | Pruebas de Sistema | Evaluar consistencia y rendimiento de BD | Pruebas con datos simulados en entorno de staging |
| Preproducción | Pruebas de Seguridad | Identificar vulnerabilidades en datos | Validación de accesos y cifrado de datos |
| Producción | Monitoreo y Observabilidad | Detectar errores y optimizar consultas | Logs de consultas y alertas en SQL Server |
| Fase | Tipo de Prueba | Objetivo | Estrategia |

**3. Herramientas de Pruebas**

|  |  |
| --- | --- |
| **Tipo de Prueba** | **Herramienta Sugerida** |
| Pruebas Unitarias | Scripts SQL en SQL Server |
| Pruebas de Integración | Postman para pruebas API |
| Pruebas de Seguridad | Auditoría de permisos y cifrado en BD |
| Pruebas de Rendimiento | Optimización de consultas y análisis de índices en SQL Server |
| Monitoreo y Logging | Registros y alertas en SQL Server Profiler |

**4. Casos de Prueba Ejemplo**

**4.1 Prueba Unitaria (Gestión de Usuarios en Base de Datos)**

* **Descripción:** Validar que un usuario pueda ser registrado correctamente en la base de datos.
* **Entradas:** Datos completos del usuario (nombre, correo, contraseña).
* **Resultado Esperado:** El usuario se almacena correctamente en la base de datos.
* **Automatización:** Scripts SQL ejecutados en SQL Server.

**4.2 Prueba de Integración (Operaciones CRUD en la Base de Datos)**

* **Descripción:** Validar que las operaciones de creación, lectura, actualización y eliminación (CRUD) funcionen correctamente en la base de datos.
* **Entradas:** Sentencias SQL para insertar, actualizar y eliminar registros.
* **Resultado Esperado:** Las operaciones reflejan correctamente los cambios en la base de datos.
* **Automatización:** Pruebas ejecutadas en Postman para verificar la API conectada a SQL Server.

**4.3 Prueba de Seguridad (Protección contra Inyección SQL)**

* **Descripción:** Intentar ejecutar consultas maliciosas en la base de datos para verificar su seguridad.
* **Entradas:** Valores como "' OR 1=1 --" en los campos de entrada de usuario.
* **Resultado Esperado:** La base de datos rechaza la consulta y no compromete los datos almacenados.
* **Automatización:** Análisis manual y validación de seguridad en SQL Server.

**4.4 Prueba de Rendimiento (Optimización de Consultas en SQL Server)**

* **Descripción:** Evaluar el tiempo de respuesta de las consultas con grandes volúmenes de datos.
* **Entradas:** Consulta SQL sobre una tabla con más de 100,000 registros.
* **Resultado Esperado:** La consulta se ejecuta en un tiempo óptimo sin afectar el rendimiento del servidor.
* **Automatización:** Uso de SQL Server Profiler para monitorear tiempos de ejecución.

**5. Flujo de Pruebas en CI/CD**

**Pipeline de CI/CD en GitHub Actions/Jenkins/GitLab**

1️.**Commit & Push** ➝ El desarrollador sube cambios en la estructura de la base de datos (scripts SQL).  
2️.**Ejecución de Pruebas Unitarias** Verificación de integridad de datos y validaciones en stored procedures.  
3️. **Ejecución de Pruebas de Integración** Validación de operaciones CRUD con Postman y scripts automatizados.  
4️. **Análisis de Código y Seguridad** Revisión de consultas para evitar inyección SQL (SonarQube, OWASP ZAP).  
5️. **Despliegue a Staging** Implementación de cambios en un entorno de prueba con datos de prueba.  
6️. **Ejecución de Pruebas de Sistema & Carga** Evaluación de consultas con grandes volúmenes de datos (SQL Server Profiler, k6).  
7️. **Despliegue Canary en Producción** Aplicación progresiva de cambios en la base de datos mediante Feature Flags.  
8️. **Monitoreo en Producción** Supervisión en tiempo real del rendimiento y errores con herramientas como ELK Stack y Datadog.

**6. Métricas y KPIs**

| **Métrica** | **Objetivo** |
| --- | --- |
| Cobertura de Código | > 80% en pruebas unitarias |
| Tiempo de Respuesta API | < 300 ms |
| Errores Críticos en Producción | 0 bloqueantes |
| Disponibilidad del Sistema | 99.9% |

**TIPOS DE PRUEBAS DE SOFTWARE**

**1. Según el Enfoque**

| **Tipo de Prueba** | **Descripción** |
| --- | --- |
| **Pruebas Manuales** | Validación de consultas, stored procedures y triggers de forma manual mediante ejecución directa en SQL Server. |
| **Pruebas Automatizadas** | Uso de scripts en SQL Server (T-SQL) para validar integridad de datos y ejecución de procedimientos almacenados. |
| **Pruebas Basadas en Riesgo** | Priorización de pruebas en tablas críticas como transacciones de ventas, stock y facturación |
| **Pruebas Exploratorias** | Ejecución de consultas sin casos de prueba predefinidos para detectar inconsistencias en datos o estructura. |
| **Pruebas Basadas en Especificaciones** | Validación de integridad referencial, normalización y restricciones según el modelo de datos. |
| **Pruebas Basadas en Estructura** | Evaluación del rendimiento y optimización de índices, vistas y particionamiento de tablas. |

**2. Según la Necesidad de Ejecución**

| **Tipo de Prueba** | **Descripción** |
| --- | --- |
| **Pruebas Estáticas** | Se analizan estructuras de tablas, relaciones, integridad referencial, índices y stored procedures sin ejecutarlos. Incluye revisión de modelos de datos y documentación técnica. |
| **Pruebas Dinámicas** | Se ejecutan consultas SQL, triggers y stored procedures para validar el correcto procesamiento de datos, detección de errores y rendimiento de la base de datos. |

**3. Según el Tipo de Ejecución**

| **Tipo de Prueba** | **Descripción** |
| --- | --- |
| **Pruebas Funcionales** | Validan que las consultas, inserciones, actualizaciones y eliminaciones de datos cumplan con los requerimientos establecidos en la aplicación. |
| **Pruebas No Funcionales** | Evalúan el rendimiento de la base de datos, la seguridad de los datos almacenados y la eficiencia en la recuperación de información. |
| **Pruebas de Caja Negra** | Se prueban las entradas y salidas de la base de datos sin conocer su estructura interna. Por ejemplo, verificar que una consulta SQL devuelva los datos esperados. |
| **Pruebas de Caja Blanca** | Se examinan triggers, stored procedures y la lógica interna de la base de datos para asegurar su correcto funcionamiento. |
| **Pruebas de Caja Gris** | Se combinan pruebas funcionales y estructurales, analizando parcialmente la base de datos mientras se prueban sus interacciones con la aplicación. |

**4. Según el Ámbito de Pruebas**

| **Tipo de Prueba** | **Descripción** |
| --- | --- |
| **Pruebas Unitarias** | Se prueban consultas SQL, procedimientos almacenados y funciones individuales para validar su correcto funcionamiento en la base de datos. |
| **Pruebas de Integración** | Se verifica la interacción entre la base de datos y otros módulos del sistema, asegurando que las operaciones CRUD funcionen correctamente con la aplicación. |
| **Pruebas de Sistema** | Evalúan la base de datos en un entorno de prueba completo, verificando su rendimiento, seguridad y consistencia de datos en escenarios reales. |
| **Pruebas de Aceptación** | Se realizan con usuarios finales o testers para validar si la base de datos soporta correctamente las necesidades del negocio y cumple con los requisitos funcionales. |

**5. Pruebas Funcionales**

| **Tipo de Prueba** | **Descripción** |
| --- | --- |
| **Pruebas de Requisitos** | Verifican que el software cumpla con los requerimientos definidos. |
| **Pruebas de Regresión** | Se ejecutan tras cambios en la base de datos para asegurar que nuevas modificaciones no afecten la integridad ni la funcionalidad del sistema. |
| **Pruebas de Interfaz de Usuario (UI)** | Evalúan la correcta visualización y funcionamiento de la interfaz gráfica. |
| **Pruebas de Base de Datos** | Verifican la integridad y seguridad de la base de datos. |
| **Pruebas de API** | Se validan los endpoints de comunicación entre sistemas. |

**6. Pruebas No Funcionales**

| **Tipo de Prueba** | **Descripción** |
| --- | --- |
| **Pruebas de Rendimiento** | Evalúan la velocidad, estabilidad y capacidad del sistema. |
| **Pruebas de Carga** | Analizan el comportamiento del software con múltiples usuarios concurrentes. |
| **Pruebas de Estrés** | Evalúan el software bajo condiciones extremas para medir su resistencia. |
| **Pruebas de Seguridad** | Detectan vulnerabilidades en la autenticación, encriptación y control de accesos. |
| **Pruebas de Usabilidad** | Analizan la facilidad de uso e intuitividad del software. |
| **Pruebas de Compatibilidad** | Verifican el funcionamiento en diferentes dispositivos y navegadores. |
| **Pruebas de Configuración** | Evalúan el impacto de cambios en la configuración del software. |

**7. Pruebas de Mantenimiento**

| **Tipo de Prueba** | **Descripción** |
| --- | --- |
| **Pruebas de Reprueba (Re-testing)** | Se ejecutan para verificar que un error reportado ha sido corregido. |
| **Pruebas de Migración** | Evaluan el software al ser trasladado a otro entorno o tecnología. |
| **Pruebas de Recuperación** | Verifican la capacidad del sistema para recuperarse ante fallos. |

**8. Pruebas en DevOps y CI/CD**

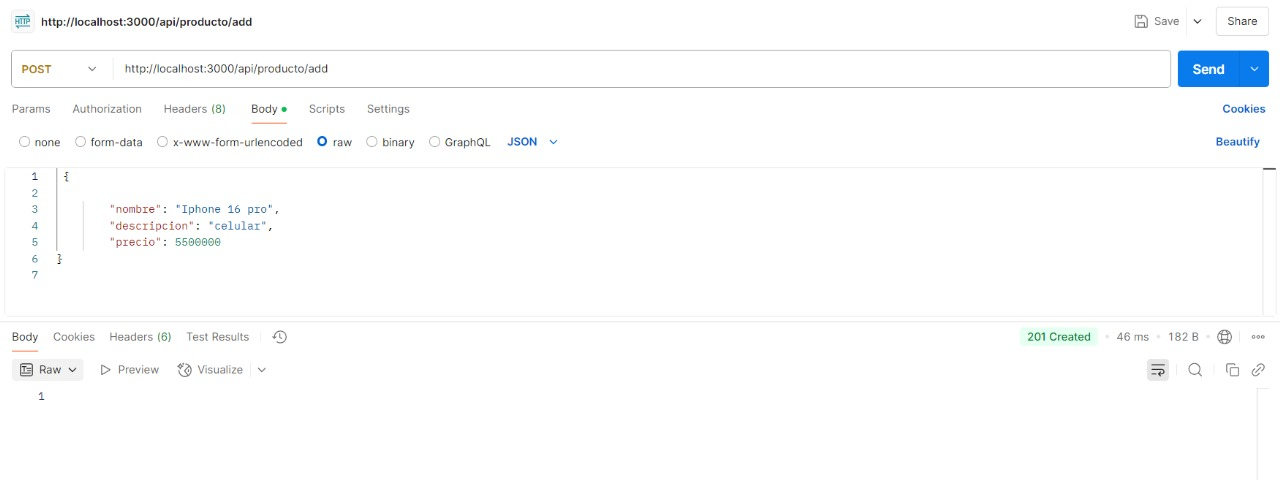
| **Tipo de Prueba** | **Descripción** |
| --- | --- |
| **Pruebas en CI/CD** | Automatización de pruebas en la integración y entrega continua. |
| **Pruebas Canary** | Despliegue de una nueva versión del software a un grupo reducido de usuarios antes de lanzarlo completamente. |
| **Pruebas de Feature Flags** | Se activan/desactivan funcionalidades en producción sin necesidad de nuevos despliegues. |

**8️. Evidencia de Pruebas (Postman)**

**Pruebas de navegación/base de datos:**

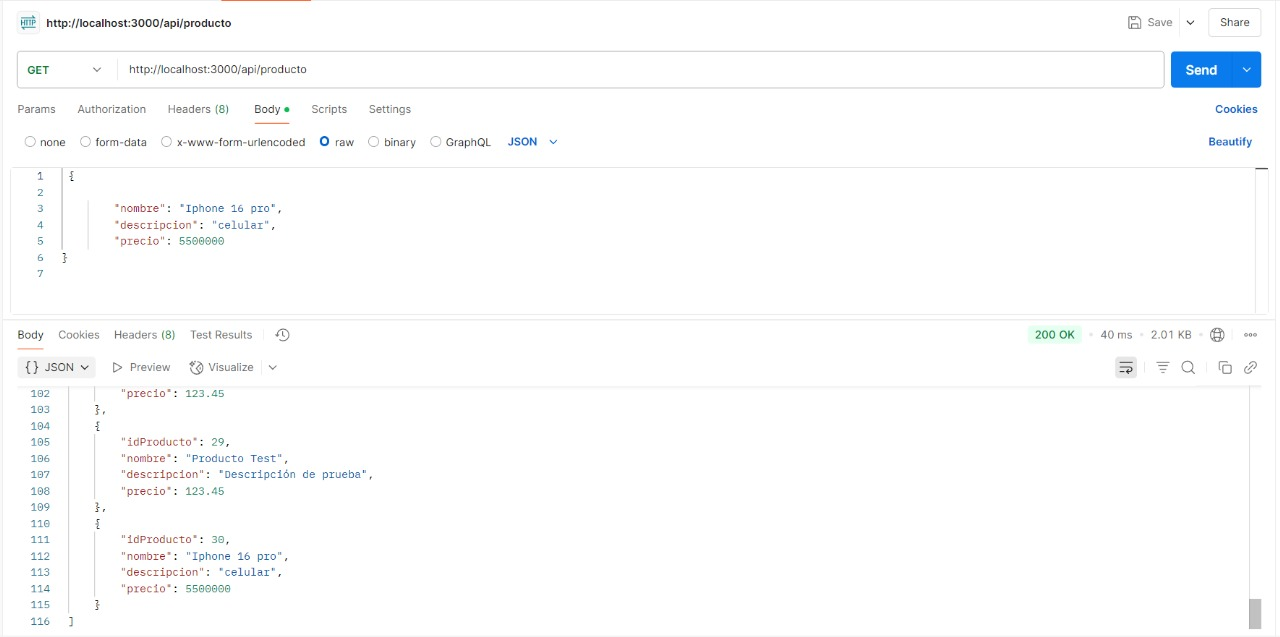
**Gestión de Producto (CRUD)**

TC001 – Crear Producto (POST)



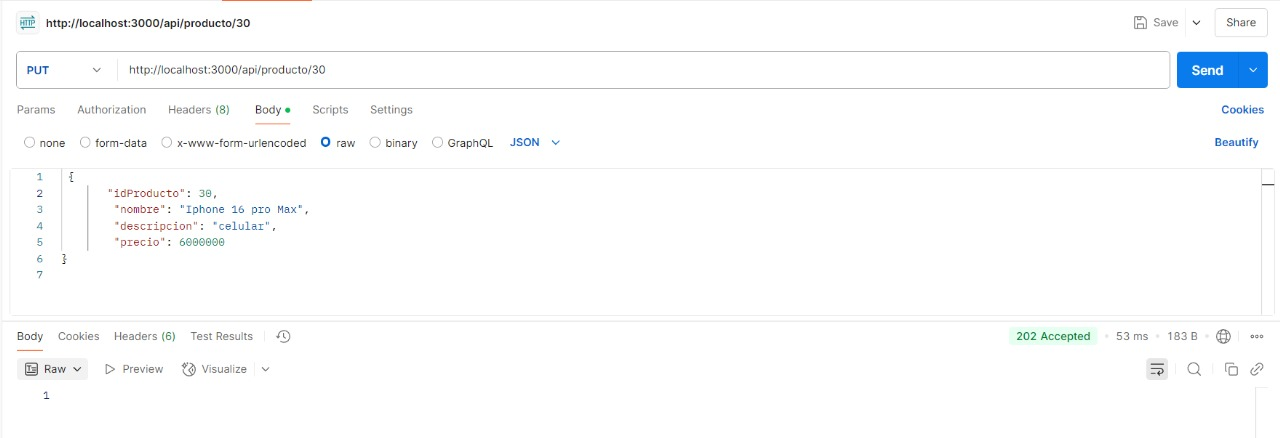
Resultado esperado: Código 201 – Producto creado correctamente.

TC002 – Leer Producto (GET)



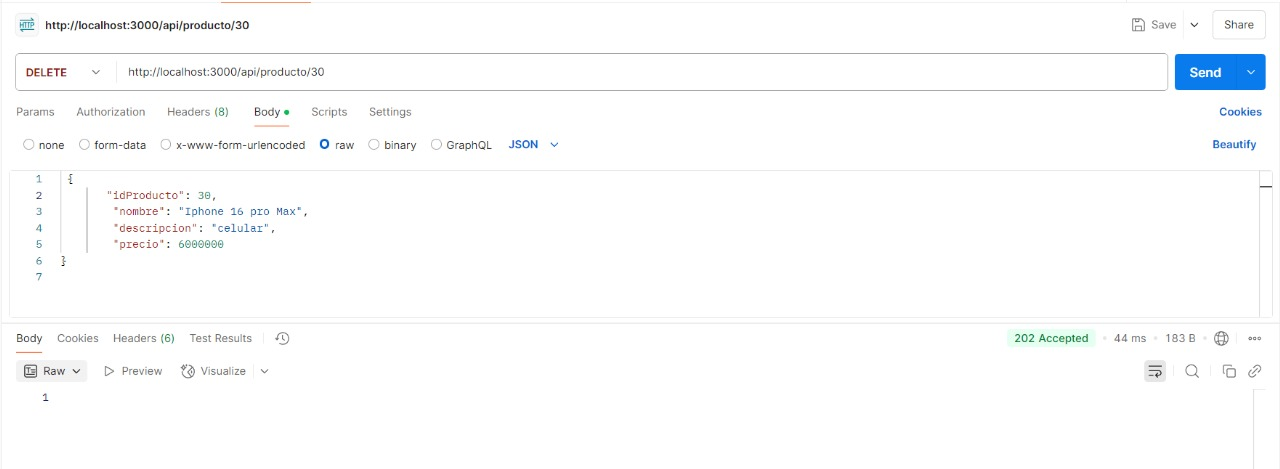
Resultado esperado: Código 200 – Datos del producto recuperados.

TC003 – Actualizar Producto (PUT)



Resultado esperado: Código 200 – Producto actualizado.

TC004 – Eliminar Producto (DELETE)



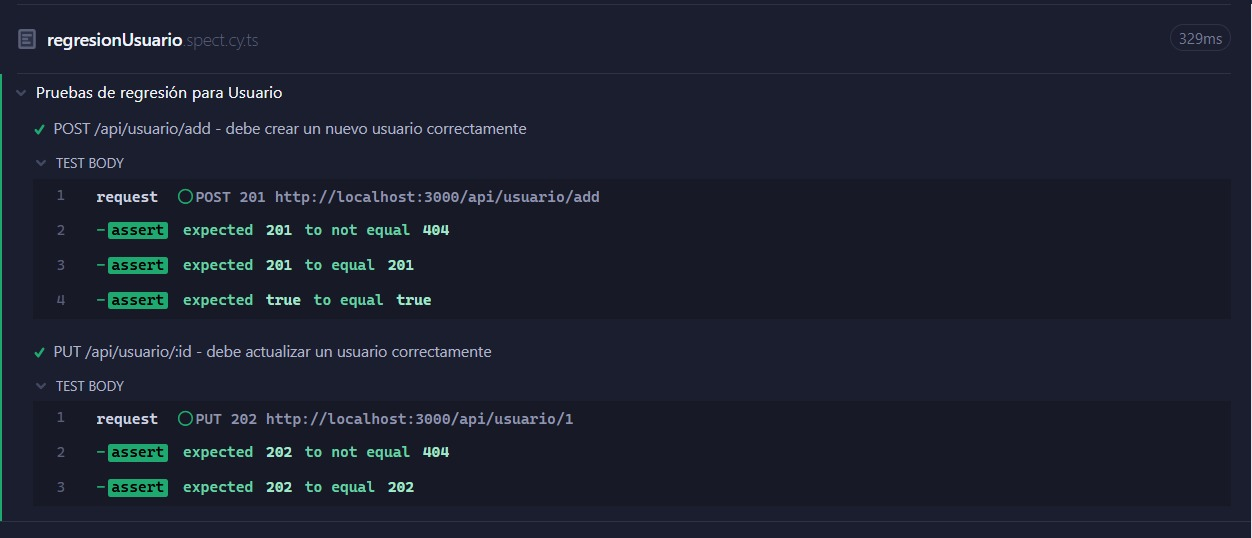
Resultado esperado: Código 200 – Producto eliminado.

**Pruebas de regresión:**

TC005 – Crear Usuario (POST)

Resultado esperado:

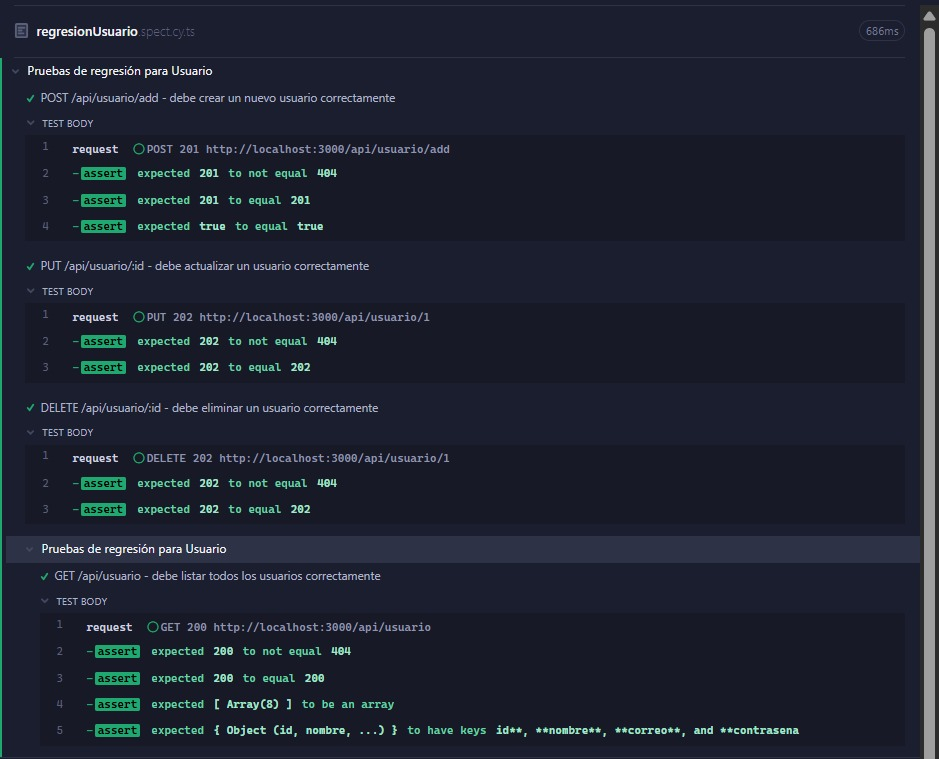
Código 201 – Usuario creado correctamente.



TC006 – Leer Usuario (GET)

Resultado esperado:

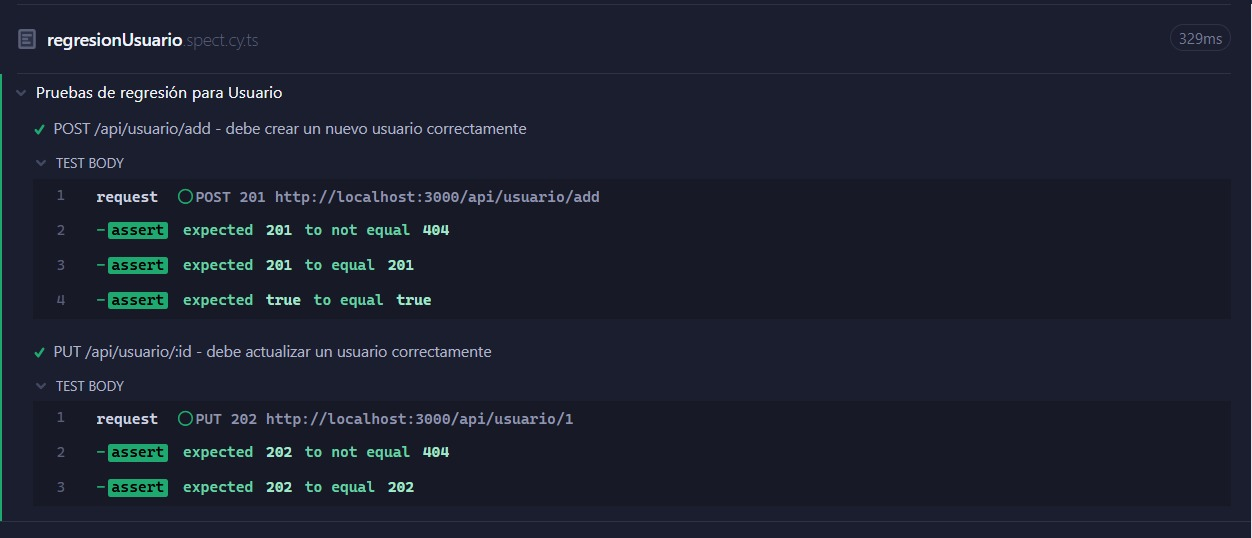
Código 200 – Datos del usuario recuperados.



TC007 – Actualizar Usuario (PUT)

Resultado esperado:

Código 202 – Usuario actualizado correctamente.



TC008 – Eliminar Usuario (DELETE)

Resultado esperado:

Código 202 – Usuario eliminado correctamente.

