PLAN DE PRUEBAS DE SOFTWARE

PROYECTO: SISTEMA DE PEDIDOS EN LÍNEA «RESTAURANTE BAMBÚ»

AUTOR: David Manuel Mamani Huanca

FECHA: Noviembre de 2025

El Alto - Bolivia

ÍNDICE DE CONTENIDO

PLAN DE PRUEBAS

El presente documento detalla la estrategia integral de aseguramiento de la calidad (QA) para el «Sistema de Pedidos en Línea del Restaurante Bambú». Este sistema, desarrollado como una aplicación web progresiva utilizando el framework Next.js, tiene como misión crítica la digitalización del proceso de ventas, desde la selección de productos hasta el procesamiento de pagos en tiempo real.

Dada la naturaleza transaccional del sistema y la sensibilidad de los datos manejados (información personal, transacciones financieras), este plan adopta un enfoque riguroso basado en estándares internacionales como ISO/IEC 25010 [1] para la calidad del producto software y metodologías ágiles adaptadas como DSDM [2].

Objetivos del Plan de Pruebas

El objetivo primordial es certificar que el Producto Mínimo Viable (MVP) cumple con los requisitos funcionales y no funcionales especificados, garantizando una experiencia de usuario fluida, segura y libre de defectos críticos antes de su despliegue en producción.

Los objetivos específicos incluyen:

- **Validación Funcional:** Asegurar que los 24 Requisitos Funcionales (RF) operan exactamente como se describe en la documentación técnica, cubriendo el ciclo completo de vida del pedido.
- **Integridad Transaccional:** Verificar la fiabilidad de las integraciones con pasarelas de pago (Stripe) y servicios de terceros, garantizando la consistencia de los datos financieros.
- **Robustez y Seguridad:** Identificar y mitigar vulnerabilidades de seguridad web comunes (OWASP Top 10) y asegurar la resiliencia del sistema bajo cargas de tráfico esperadas.
- **Usabilidad:** Garantizar que la interfaz sea intuitiva y accesible (WCAG 2.1 AA) para una base de usuarios diversa en múltiples dispositivos.

Alcance y Elementos de Prueba

El alcance de este plan abarca la verificación de todos los módulos desarrollados en el sprint de 13 días (30 Oct - 12 Nov 2024).

Módulos Incluidos (In Scope):

- **Módulo de Cliente:** Autenticación (NextAuth), Catálogo de Productos, Carrito de Compras (Context API), Checkout (Stripe Integration), Gestión de Perfil y Historial de Pedidos.

- **Módulo Administrativo:** Panel de Control, CRUD de Productos y Categorías, Gestión de Usuarios y Supervisión de Pedidos.
- **Infraestructura:** API Routes de Next.js, Base de Datos MongoDB y Almacenamiento de Archivos en AWS S3.

Elementos Excluidos (Out of Scope):

- Pruebas de penetración profunda (Pentesting) por terceros externos (se realizarán escaneos automatizados internos).
- Pruebas de carga distribuida a escala global (se simulará carga localizada).

Enfoque de Pruebas por Capa y Tecnología

La estrategia de pruebas sigue el modelo de la «Pirámide de Pruebas», distribuyendo el esfuerzo para maximizar el retorno de inversión (ROI) y la velocidad de detección de errores.

- **Capa de Presentación (Frontend):** Se validará la interfaz de usuario construida con React, asegurando la correcta renderización de componentes, la gestión del estado global (Carrito) y la responsividad.
- **Capa de Aplicación (Backend/API):** Se probarán los endpoints de la API RESTful de Next.js, validando la lógica de negocio, la autorización de rutas y el manejo de errores HTTP.
- **Capa de Datos:** Se verificará la integridad referencial y la persistencia de datos en MongoDB, asegurando que las operaciones CRUD no corrompan la estructura de las colecciones.

TIPOS DE PRUEBAS

Pruebas Funcionales: Asegurando la Lógica de Negocio

Las pruebas funcionales verifican que el software realiza las operaciones esperadas bajo condiciones normales y de borde.

Pruebas Unitarias en Frontend (React/Next.js)

El frontend, desarrollado con React y Tailwind CSS, requiere una validación granular de sus componentes. Utilizando Jest [4] y React Testing Library [13], se aislarán componentes críticos para verificar su comportamiento.

Objetivo: Validar que los componentes visuales reaccionan correctamente a las interacciones del usuario (clics, entradas de texto) y que el estado se actualiza según lo esperado.

Alcance:

- **Componentes de Forms (Login, Registro, Dirección):** Validación de campos obligatorios, formatos de email y manejo de errores visuales.
- **Componente Cart:** Verificación de la lógica de adición, eliminación y cálculo de subtotales en el carrito de compras.
- **Componente ProductCard:** Renderizado correcto de precios, imágenes y estados de disponibilidad.

Pruebas Unitarias en Backend (Node.js/API Routes)

La lógica del servidor, implementada mediante API Routes en Next.js, será probada utilizando Jest y Supertest [6]. Estas pruebas se centran en la lógica pura sin depender de la interfaz gráfica.

Objetivo: Asegurar que las reglas de negocio (ej. cálculo de impuestos, validación de stock, creación de usuarios) se ejecuten correctamente.

Alcance:

- **Validación de Payloads:** Verificar que la API rechaza datos mal formados con los códigos de estado HTTP correctos (400 Bad Request).
- **Lógica de Autenticación:** Verificar que las rutas protegidas retornan 401 Unauthorized sin un token válido.

Pruebas de Integración de Servicios

El sistema depende de servicios externos críticos. Las pruebas de integración aseguran que la comunicación con estos sistemas es fluida y maneja las excepciones correctamente.

- **Stripe (Pagos):** Validación de la creación de PaymentIntents, manejo de webhooks para confirmación de pagos y escenarios de tarjetas rechazadas [10].
- **AWS S3 (Imágenes):** Verificación de la subida de imágenes de productos, validación de tipos MIME permitidos y generación correcta de URLs públicas [11].
- **NextAuth (Identidad):** Pruebas del flujo de OAuth con Google y credenciales propias, verificando la persistencia de la sesión [12].

Pruebas End-to-End (E2E) y Validación de Flujos

Estas pruebas simulan el comportamiento de un usuario real navegando por la aplicación, validando que todos los componentes integrados funcionen en armonía. Se utilizará Playwright [5] por su capacidad de ejecutar pruebas en múltiples motores de navegador.

Validación de Flujos Críticos de Usuario

- **Flujo de Compra Completa:** Navegación Home → Selección de Categoría → Agregar al Carrito → Modificar Cantidad → Checkout → Pago Exitoso → Pantalla de Confirmación.
- **Flujo de Registro y Perfil:** Registro de nuevo usuario → Login → Actualización de dirección de envío → Logout.

Validación de Flujos Administrativos

- **Gestión de Catálogo:** Un administrador inicia sesión → Crea un nuevo producto con imagen → Verifica que aparece en la tienda → Edita el precio → Elimina el producto.

Pruebas No Funcionales (Calidad y Robustez)

Pruebas de Seguridad (OWASP)

Siguiendo la metodología de OWASP Testing Guide [3], se realizarán análisis estáticos y dinámicos para detectar vulnerabilidades.

- **Inyección NoSQL:** Verificación de sanitización de entradas en consultas a MongoDB.
- **Cross-Site Scripting (XSS):** Intento de inyección de scripts maliciosos en campos de comentarios o descripciones de productos.
- **Control de Acceso:** Verificación de que usuarios normales no puedan acceder a rutas `/admin`.

Pruebas de Rendimiento y Carga

Utilizando Artillery [7], se someterá al sistema a cargas simuladas para verificar su estabilidad bajo presión.

- **Prueba de Carga:** 50 usuarios concurrentes navegando y añadiendo productos.
- **Prueba de Estrés:** 200 usuarios concurrentes intentando realizar checkout simultáneamente (evaluación de concurrencia en base de datos y API).
- **Métricas de Éxito:** Tiempo de respuesta < 2000 ms para el 95% de las peticiones (p95).

Pruebas de Usabilidad y Accesibilidad

- **Accesibilidad (WCAG):** Auditoría automatizada con axe DevTools y Lighthouse para asegurar cumplimiento de nivel AA (contraste de colores, etiquetas ARIA, navegación por teclado) [14].
- **Compatibilidad:** Verificación visual y funcional en Chrome, Firefox, Safari (macOS/iOS) y Edge mediante BrowserStack.

HERRAMIENTAS PARA PRUEBAS

La selección de herramientas se ha realizado priorizando la compatibilidad con el ecosistema JavaScript/React y su adopción en la industria.

Herramienta	Tipo	Descripción y Justificación
Jest	Framework de Pruebas	Motor principal para pruebas unitarias y de integración. Rápido, soporta mocking y cobertura de código.
React Testing Library	Pruebas de Componentes	Enfocado en probar componentes desde la perspectiva del usuario (DOM), evitando probar detalles de implementación.
Playwright	Automatización E2E	Permite pruebas cross-browser (Chromium, WebKit, Firefox) fiables y rápidas, con soporte nativo para esperas asíncronas.
Supertest	Pruebas de API	Librería para realizar peticiones HTTP a la API de Node.js de forma simplificada en los tests de integración.
Artillery	Pruebas de Carga	Herramienta moderna para SRE y DevOps, permite definir escenarios de carga en YAML y genera reportes detallados.
OWASP ZAP	Seguridad	Escáner de seguridad de aplicaciones web para detectar vulnerabilidades comunes automáticamente.
Lighthouse	Auditoría Web	Herramienta de Google para auditar rendimiento, accesibilidad y SEO.

CASOS DE PRUEBA DETALLADOS

A continuación, se presenta una muestra representativa de los casos de prueba diseñados. La suite completa consta de 32 casos.

ID	Título	Precondición	Pasos de Ejecución	Resultado Esperado	Prioridad
CP-001	Registro de Usuario Exitoso	Usuario no registrado, en página `/register`.	1. Ingresar nombre válido. 2. Ingresar email único. 3. Ingresar contraseña (>6 caracteres). 4. Click en «Registrarse».	Sistema crea usuario en BD, redirige a `/login` o hace auto-login, muestra mensaje de éxito.	Alta
CP-004	Agregar Producto al Carrito	Usuario en página de detalle de producto.	1. Seleccionar cantidad (ej. 2). 2. Click en «Agregar al Carrito».	Contador del icono de carrito se incrementa. Notificación «Producto agregado». Estado global actualizado.	Alta
CP-008	Procesamiento de Pago (Stripe)	Carrito con items, usuario en `/checkout`.	1. Ingresar dirección de envío. 2. Ingresar datos de tarjeta de prueba (Visa 4242…). 3. Click en «Pagar».	Stripe procesa pago. Redirección a `/success`. Orden creada en BD con estado «Pagado». Carrito se vacía.	Crítica

CP-012	Validación de Seguridad NoSQL	API endpoint `/api/auth/login`.	1. Enviar payload malicioso en password: `{"$ne": null}`.	API retorna error 400 o 401. No permite acceso (bypass) sin contraseña correcta.	Alta
CP-024	Carga de Imagen de Producto	Admin logueado en `/admin/products/new`.	1. Seleccionar archivo .jpg válido (2MB). 2. Completar datos. 3. Guardar.	Imagen se sube a AWS S3. URL se guarda en MongoDB. Producto se visualiza con imagen correcta.	Media

MATRIZ DE TRAZABILIDAD DE REQUISITOS

Esta matriz asegura que cada requisito funcional definido en la fase de análisis (info.pdf) está cubierto por al menos un caso de prueba.

ID Req.	Descripción Requisito	ID Caso de Prueba	Cobertura
RF-01	El sistema permitirá el registro de usuarios con email/password.	CP-001, CP-003	100%
RF-05	El sistema debe permitir agregar productos al carrito.	CP-004, CP-025	100%
RF-10	Integración con pasarela de pagos Stripe.	CP-007, CP-008, CP-013	100%
RF-15	Administrador puede crear productos con imágenes.	CP-010, CP-024	100%
RF-20	Sistema debe ser responsivo (Móvil/ Desktop).	CP-014, CP-015	100%

ESTRATEGIA DE AUTOMATIZACIÓN Y CRONOGRAMA

La ejecución se realizará en un periodo de 14 días, siguiendo un flujo iterativo.

Cronograma de Actividades

- **Fase 1: Preparación (Días 1-2):** Configuración de entornos de prueba (Testing, Staging), generación de datos de prueba (seeds) en MongoDB, configuración de pipelines de CI/CD.
- **Fase 2: Pruebas Unitarias e Integración (Días 3-8):** Desarrollo y ejecución de scripts en Jest. Foco en la lógica de backend y componentes críticos de frontend.
- **Fase 3: Pruebas E2E y Sistema (Días 9-11):** Ejecución de suites de Playwright para flujos completos. Verificación visual.
- **Fase 4: Pruebas No Funcionales (Días 12-13):** Ejecución de scripts de Artillery (carga) y escaneos OWASP ZAP (seguridad).
- **Fase 5: Reporte y Cierre (Día 14):** Análisis de resultados, re-testing de defectos corregidos y generación del informe final de aceptación.

Entregables

- Código fuente de los scripts de prueba automatizados.
- Reporte de ejecución de pruebas (HTML/PDF) con gráficos de pasados/fallados.
- Lista de defectos encontrados (Bug Report) en Jira/Trello.
- Certificado de Calidad del Software (si se cumplen los criterios de éxito).

REFERENCIAS BIBLIOGRÁFICAS

International Organization for Standardization (ISO) & International Electrotechnical Commission (IEC). (2023). *ISO/IEC 25010:2023 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*.

DSDM Consortium. (n.d.). *Dynamic Systems Development Method (DSDM)*. Recuperado de DSDM Handbook.

Open Web Application Security Project (OWASP). (2023). *OWASP Testing Guide v4.2*. OWASP Foundation.

Facebook Open Source. (2024). *Jest: Delightful JavaScript Testing*. Recuperado de https://jestjs.io/

Microsoft. (2024). *Playwright: Fast and reliable end-to-end testing*. Recuperado de https://playwright.dev/

Visionmedia. (n.d.). *Supertest: HTTP assertions made easy*. Recuperado de https://www.npmjs.com/package/supertest

Artillery.io. (2024). *Artillery: Cloud-scale load testing*. Recuperado de https://www.artillery.io/

Vercel. (2024). *Next.js Documentation: The React Framework for the Web*. Recuperado de https://nextjs.org/docs

MongoDB Inc. (2024). *Mongoose ODM v8.0.0 Documentation*. Recuperado de https://mongoosejs.com/

Stripe Inc. (2024). *Stripe API Reference*. Recuperado de https://stripe.com/docs/api

Amazon Web Services. (2024). *Amazon S3 Developer Guide*. Recuperado de https://docs.aws.amazon.com/s3/

Balázs, I. (2024). *NextAuth.js: Authentication for Next.js*. Recuperado de https://next--auth.js.org/

Dodds, K. C. (2024). *React Testing Library: Simple and complete testing utilities*. Recuperado de https://testing-library.com/

World Wide Web Consortium (W3C). (2018). *Web Content Accessibility Guidelines (WCAG) 2.1*. Recuperado de https://www.w3.org/TR/WCAG21/

Brooke, J. (1996). SUS: A quick and dirty usability scale. En P. W. Jordan, B. Thomas, B. A. Weerdmeester, & I. L. McClelland (Eds.), *Usability Evaluation in Industry*. Taylor & Francis.