**Plan de Pruebas**

Autores:

Moisés Pérez

Miller Hernández

Jose Jimenez

Julio Peñaloza

Heidy Serrano

Universidad de Pamplona

Facultad de Ingenierías y Arquitectura

Ingeniería de Sistemas

Villa del Rosario – Norte de Santander

2025

**1.Investigacion inicial**

**¿Qué es un plan de pruebas de software?**  
Un plan de pruebas de software es un documento formal que describe el enfoque, los objetivos, los recursos y el cronograma para las actividades de prueba de un proyecto de software. Funciona como una guía detallada que asegura que las pruebas se realicen de manera sistemática y completa, cubriendo aspectos como funcionalidad, rendimiento, seguridad y usabilidad del software.

**. ¿Cuál es su importancia en el ciclo de vida del desarrollo de software?**  
El plan de pruebas es crucial porque:

* **Garantiza la calidad**: Ayuda a identificar y corregir errores antes de que el software llegue al usuario final.
* **Reduce costos**: Detectar fallos en etapas tempranas evita gastos altos en correcciones posteriores.
* **Organiza el proceso**: Define roles, responsabilidades y recursos necesarios, evitando improvisaciones.
* **Valida requisitos**: Verifica que el software cumpla con los requerimientos establecidos inicialmente.
* **Facilita la comunicación**: Sirve como referencia para el equipo de desarrollo, testers y stakeholders.

**. ¿Qué elementos debe contener un plan de pruebas típico?**  
Un plan de pruebas completo generalmente incluye:

* **Objetivos**: Propósito y alcance de las pruebas.
* **Estrategia**: Tipos de pruebas a realizar (unitarias, integración, sistema, aceptación).
* **Criterios de entrada/salida**: Condiciones para comenzar y finalizar las pruebas.
* **Recursos**: Herramientas (como Selenium, JIRA), equipos y personal asignado.
* **Cronograma**: Fechas y tiempos estimados para cada fase de prueba.
* **Ambientes de prueba**: Configuraciones de hardware/software necesarias.
* **Casos de prueba**: Descripción de escenarios específicos a evaluar.
* **Riesgos y mitigaciones**: Posibles problemas y planes para abordarlos.
* **Informes**: Formatos para documentar resultados y hallazgos.

**Diseño del plan de pruebas**

El desarrollo de software es un proceso complejo que requiere una planificación meticulosa para garantizar que el producto final cumpla con los estándares de calidad, funcionalidad y seguridad esperados por los stakeholders. En el contexto del sitio web para la peluquería Paris Estilos, la implementación de un Plan de Pruebas de Software es fundamental para validar que el sistema cumpla con los requisitos técnicos, operativos y legales definidos en el análisis de viabilidad del proyecto (Pérez et al., 2024). Este plan se alinea con las mejores prácticas de la ingeniería de software y sigue los lineamientos de normas internacionales como el IEEE 829 para la documentación de pruebas y el ISTQB (International Software Testing Qualifications Board, 2018) para la estandarización de procesos.

El sitio web de Paris Estilos busca optimizar procesos clave como la gestión de citas en línea, el control de inventario y la facturación electrónica, lo que demanda un enfoque riguroso en la detección temprana de errores y la validación de la experiencia del usuario. Según Pressman (2010), las pruebas de software no solo deben verificar la funcionalidad del sistema, sino también su capacidad para operar bajo condiciones reales, incluyendo cargas de trabajo elevadas y posibles amenazas de seguridad. Este plan adopta una estrategia integral que abarca pruebas unitarias, de integración, de sistema y de aceptación, asegurando que cada módulo cumpla con los criterios de calidad antes de su despliegue.

Además, el proyecto debe cumplir con regulaciones locales como la Ley 1581 de 2012 (protección de datos personales) y la Resolución 000042 de 2020 (facturación electrónica), lo que exige pruebas específicas para garantizar el manejo adecuado de información sensible y la generación de documentos legales válidos (Ministerio de Tecnologías de la Información y Comunicaciones, 2020). La combinación de metodologías ágiles y un enfoque basado en riesgos permitirá priorizar las pruebas en áreas críticas, como el módulo de reservas y la integración con pasarelas de pago, mitigando así posibles fallos que podrían impactar la operación del negocio.

**Objetivos de las Pruebas**

El Plan de Pruebas de Software para el sitio web de *Paris Estilos* tiene los siguientes objetivos, alineados con los estándares de calidad y los requisitos del proyecto:

* **Validar la funcionalidad del sistema**
  + Verificar que los módulos de **reservas en línea, gestión de inventario y facturación electrónica** operen conforme a los requisitos funcionales (RF1-RF4) definidos en el documento de análisis (International Software Testing Qualifications Board [ISTQB], 2018).
  + Asegurar que las integraciones entre sistemas (ej: base de datos, API de pagos) funcionen sin errores.
* **Garantizar la usabilidad y accesibilidad**
  + Evaluar que la interfaz de usuario (UI) cumpla con los principios de **diseño intuitivo y responsividad** en dispositivos móviles y de escritorio (Nielsen, 2012).
  + Realizar pruebas con usuarios finales (empleados y clientes) para identificar posibles barreras en la navegación.
* **Comprobar el rendimiento y escalabilidad**
  + Testear la capacidad del sistema para manejar **50 usuarios concurrentes** (RNF-Rendimiento) sin degradación en la velocidad de respuesta (Pressman, 2010).
  + Simular picos de tráfico para evaluar la estabilidad del servidor.
* **Asegurar la seguridad de los datos**
  + Validar el cumplimiento de la **Ley 1581 de 2012** (protección de datos personales) mediante pruebas de encriptación y controles de acceso (Ministerio de Tecnologías de la Información y Comunicaciones, 2020).
  + Ejecutar pruebas de penetración (*pentesting*) con herramientas como **OWASP ZAP** para detectar vulnerabilidades (OWASP Foundation, 2021).
* **Verificar el cumplimiento legal**
  + Confirmar que las **facturas electrónicas** generadas cumplan con la **Resolución 000042 de 2020**.
  + Asegurar que las políticas de privacidad y términos de uso sean claros y accesibles (Ley 527 de 1999).

**Estrategia General de Pruebas**

La estrategia de pruebas se basa en metodologías ágiles y estándares internacionales (ISTQB, 2018), estructurada en las siguientes etapas:

1. **Tipos de Pruebas**

* **Pruebas unitarias** (*Unit Testing*):
  + Validar funciones individuales (ej: cálculo de horarios disponibles) con **Jest (JavaScript)** o **PHPUnit (PHP)**.
  + Cobertura mínima del **95% del código** (Sommerville, 2011).
* **Pruebas de integración** (*Integration Testing*):
  + Verificar la comunicación entre módulos (ej: reservas → base de datos) mediante **Postman** y pruebas E2E.
* **Pruebas de sistema** (*System Testing*):
  + Evaluar el flujo completo (agendamiento → confirmación → facturación) en un entorno controlado.
* **Pruebas de aceptación** (*UAT - User Acceptance Testing*):
  + Involucrar a los stakeholders (dueños, empleados) para validar que el sistema cumpla con sus necesidades.

1. **Metodología**

* **Enfoque ágil**: Sprints de pruebas en paralelo al desarrollo, con retroalimentación continua.
* **Pruebas basadas en riesgos**: Priorizar módulos críticos (reservas, facturación) y requisitos legales.

1. Herramientas y Entornos

| **Area** | **Herramientas** | **Propósito** |
| --- | --- | --- |
| Automatización UI | Selenium, Cypress | Validar navegación y usabilidad. |
| APIs | Postman, Swagger | Probar endpoints y respuestas HTTP. |
| Seguridad | OWASP ZAP, Burp Suite | Detectar vulnerabilidades (ej: SQL injection). |
| Rendimiento | JMeter, LoadRunner | Simular carga de usuarios concurrentes. |

1. **Criterios de Aprobación**

* **0 errores críticos** (ej: pérdida de datos, fallas en reservas).
* **100% de cumplimiento** de normativas legales.
* **Tiempo de respuesta** ≤ 2 segundos en operaciones clave.

1. **Gestión de Defectos**

* Uso de **Jira** para registrar y priorizar errores por severidad (Crítico, Alto, Medio, Bajo).
* Protocolo de corrección:
  + Errores **críticos**: Resolución en ≤ 24 horas.
  + Errores **menores**: Incluidos en el siguiente sprint.

**Tipo de Pruebas**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipo de Prueba** | **Descripción** | **Herramientas** | **Ejemplo** |
| **Pruebas de regresión** | Verifican que las nuevas funciones no dañen funcionalidades previas. | Playwright, Testim, GitHub Actions | Tras añadir un botón de “cancelar cita”, se revisa que la reserva siga funcionando. |
| **Pruebas de compatibilidad** | Aseguran que el sistema funcione en diferentes dispositivos, navegadores y sistemas operativos. | BrowserStack, LambdaTest | Verificar el módulo de reservas en Chrome, Safari y móvil Android. |
| **Pruebas de accesibilidad** | Evalúan si el sitio puede ser usado por personas con discapacidad. | axe DevTools, Lighthouse | Comprobar que el contraste de color y el uso de teclado estén correctos. |
| **Seguridad** | Detectan vulnerabilidades. | OWASP ZAP, Burp Suite |  |
| **Integración** | Verifican interacción entre módulos (ej. frontend ↔ backend ↔ base de datos). | Postman, Supertest | Al reservar una cita en el frontend, esta se almacena correctamente en la base de datos y se refleja en el módulo de administración. |

**Recursos Necesarios**

* Recursos Humanos
  + Tester QA funcional: para pruebas manuales.
  + Tester QA técnico: para pruebas automatizadas y de seguridad.
  + Desarrollador de soporte: para corregir errores detectados.
  + Usuario final (empleado y cliente): para pruebas de aceptación
* Herramientas
  + Automatización: Playwright, Cypress, Selenium
  + Rendimiento: K6, Gatling
  + Seguridad: OWASP ZAP, Burp Suite, Snyk
  + Gestión de pruebas: TestRail, Zephyr, Jira
  + Control de versiones: GitHub, GitLab
* Entornos
  + Ambiente de staging: réplica del entorno real.
  + Ambiente de desarrollo local: para pruebas de unidad y regresión.
* Datos de prueba
  + Generados artificialmente (mock data) o con anonimización si son reales.
* Documentación
  + Casos de prueba, matriz de trazabilidad, guías de instalación y manuales.

**Cronograma de Pruebas**

Semana Actividad principal

1-2 Pruebas unitarias e integración en módulos iniciales

Preparación del entorno, diseño de casos de prueba.

3-4 Ejecución de pruebas unitarias y de integración.

Pruebas de sistema, accesibilidad y regresión

5-6 Pruebas de rendimiento, sistema y seguridad

7-8 Pruebas de aceptación (UAT) con stakeholders

Pruebas de aceptación con usuarios finales, cierre.

9 Revisión de hallazgos, corrección y pruebas finales

**Criterios de Aceptación y Rechazo**

|  |  |  |
| --- | --- | --- |
| **Criterio** | **Aceptación** | **Rechazo** |
| **Errores críticos** | 0 errores críticos en módulos esenciales. | ≥ 1 error crítico que afecte flujo clave. |
| **Normativa legal** | Cumplimiento total (ej. marco legal). | Inconsistencias legales o no conformidad. |
| **Usabilidad** | Navegación clara, responsiva, sin obstáculos. | Interfaces confusas o inaccesibles. |
| **Rendimiento** | Respuesta ≤ 2 segundos para 95% de casos. | Latencia alta o caídas bajo carga. |
| **Seguridad** | Sin vulnerabilidades severas (OWASP Top 10). | Presencia de riesgos no mitigados. |
| **Cobertura de pruebas unitarias** | ≥ 90% del código cubierto. | < 90% de cobertura sin justificación. |

Si el sistema tarda 5 segundos en agendar una cita y debe hacerlo en menos de 2, se considera rechazado.

**Referencias**

ISTQB. (2018). Standard Glossary of Terms Used in Software Testing. International Software Testing Qualifications Board.

https://www.istqb.org/downloads/send/2-foundation-level-documents/17-glossary-of-terms.html.

Pressman, R. S., & Maxim, B. R. (2020). Ingeniería del Software: Un enfoque práctico (8ª ed.). McGraw-Hill.

Sommerville, I. (2016). Software Engineering (10th ed.). Pearson.

OWASP Foundation. (2021). OWASP Testing Guide v4. Open Web Application Security Project.

https://owasp.org/www-project-web-security-testing-guide/

Google Developers. (2023). Web Performance Optimization. Google.

https://web.dev/performance-scoring/

Apache Software Foundation. (2022). JMeter User Manual.

https://jmeter.apache.org/usermanual/index.html

Nielsen Norman Group. (2022). Usability Heuristics for User Interface Design.

https://www.nngroup.com/articles/ten-usability-heuristics/

Ley 1581 de 2012 (Colombia). Protección de datos personales.

https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=49981

Resolución 000042 de 2020 (DIAN, Colombia). Régimen de facturación electrónica.

https://www.dian.gov.co/

International Software Testing Qualifications Board. (2018). *Standard Glossary of Terms Used in Software Testing*. [https://www.istqb.org](https://www.istqb.org/)

Ministerio de Tecnologías de la Información y Comunicaciones. (2020). *Resolución 000042 de 2020: Facturación Electrónica*. Colombia.

Nielsen, J. (2012). *Usability 101: Introduction to Usability*. Nielsen Norman Group. <https://www.nngroup.com/articles/usability-101-introduction-to-usability/>

OWASP Foundation. (2021). *OWASP Testing Guide v4*. <https://owasp.org/www-project-web-security-testing-guide/>

Pérez, M., Hernández, M., Jiménez, J., Peñaloza, J., & Serrano, H. (2024). *Análisis y Desarrollo del Sitio Web de Paris Estilos*. Universidad de Pamplona.

Pressman, R. S. (2010). *Ingeniería del Software: Un Enfoque Práctico* (7ª ed.). McGraw-Hill.

Sommerville, I. (2011). *Software Engineering* (9ª ed.). Pearson Education.