**Documentación Detallada del Proceso de Pruebas QA para Sistema de Login**

**1. Contexto y Propósito del Proyecto**

El equipo de Quality Assurance se enfrentó a un problema recurrente en el sistema de autenticación de una aplicación web alojada localmente. El síntoma principal consistía en que, tras introducir credenciales válidas, el sistema en ocasiones no completaba la transición a la siguiente pantalla, mostrando comportamientos inconsistentes como:

* Recarga de la página de login sin mensaje de error
* Pantallas en blanco después de enviar credenciales
* Errores de tiempo de espera sin causa aparente

El objetivo del proceso de QA fue triple:

1. **Reproducir sistemáticamente** el problema para entender su naturaleza
2. **Identificar patrones** en los fallos para determinar causas raíz
3. **Generar evidencia técnica** para soportar las correcciones necesarias

**2. Configuración del Ambiente de Pruebas**

**2.1. Infraestructura Local**

Se implementó un entorno controlado utilizando XAMPP por sus ventajas clave:

* **Paquete integrado** (Apache + MySQL + PHP) que replica entornos productivos
* **Configuración aislada** para pruebas sin afectar otros sistemas
* **Facilidad de monitoreo** directo de logs y rendimiento

La configuración específica incluía:

* Puerto 80 para el servicio Apache
* Tiempos de espera estándar (30 segundos)
* Configuración básica de MySQL con usuario root sin contraseña para pruebas

**2.2. Caracterización del Problema**

El análisis inicial reveló que el fallo presentaba estas particularidades:

* **Inconsistencia temporal**: Ocurría en aproximadamente el 30% de los intentos
* **Agrupación temporal**: Los fallos tendían a ocurrir en ráfagas
* **Dependencia de uso**: La frecuencia de errores aumentaba tras horas de operación continua

**3. Estrategia de Automatización con Selenium**

**3.1. Enfoque Metodológico**

Se diseñó un proceso de pruebas automatizado que permitiera:

* **Ejecuciones masivas** (100 iteraciones) para obtener significancia estadística
* **Registro detallado** de cada intento con timestamp y tipo de fallo
* **Variación controlada** de parámetros (tiempos de espera, datos de entrada)

**3.2. Implementación Práctica**

El flujo de trabajo automatizado seguía esta secuencia:

1. **Inicialización del navegador** con opciones específicas
2. **Navegación a la página** de login con verificación de carga completa
3. **Introducción de credenciales** con validación de campos
4. **Envío del formulario** con manejo explícito de esperas
5. **Verificación de éxito** mediante múltiples criterios:
   * Cambio de URL esperado
   * Presencia de elementos exclusivos del dashboard
   * Ausencia de mensajes de error
6. **Cierre de sesión** limpio para próxima iteración

**3.3. Hallazgos Relevantes**

El análisis de los 100 intentos reveló:

* **Tasa de éxito general**: 72%
* **Distribución temporal**: 80% de fallos ocurrieron en el último tercio de las pruebas
* **Tipología de errores**:
  + 60% Timeout esperando redirección
  + 30% Reaparición del formulario de login
  + 10% Errores HTTP 500

**4. Pruebas de Carga con JMeter**

**4.1. Diseño del Plan de Pruebas**

Se configuró un escenario que simulaba:

* **10 usuarios concurrentes** ejecutando 10 iteraciones cada uno
* **Peticiones HTTP directas** omitiendo el renderizado gráfico
* **Variación de parámetros** mediante CSV Data Set Config

Los componentes principales incluían:

* **Preprocesadores** para establecer headers HTTP
* **Extractores** para capturar tokens de sesión
* **Assertions** para validar respuestas

**4.2. Métricas Clave Analizadas**

* **Throughput**: Capacidad del sistema para procesar 15 peticiones/segundo
* **Latencia**: Tiempo promedio de respuesta de 3.2 segundos
* **Curva de degradación**: Aumento exponencial de errores tras 70 peticiones
* **Consumo de recursos**: Uso de CPU en servidor alcanzando 85%

**5. Análisis Integrado de Resultados**

**5.1. Correlación de Hallazgos**

Ambas metodologías confirmaron:

* **Problemas de gestión de sesiones**: Cookies no persistentes
* **Cuellos de botella en base de datos**: Tiempos de consulta variables
* **Falta de escalabilidad**: Degradación progresiva del rendimiento

**5.2. Hipótesis de Causa Raíz**

La evidencia sugiere tres factores principales:

1. **Configuración subóptima de Apache**:
   * KeepAlive desactivado
   * Timeout demasiado corto (30 segundos)
2. **Problemas en la capa de datos**:
   * Consultas sin índices adecuados
   * Pool de conexiones insuficiente
3. **Manejo inadecuado de errores**:
   * Excepciones no capturadas en backend
   * Feedback insuficiente al frontend

**6. Recomendaciones Técnicas**

**6.1. Mejoras Inmediatas**

1. **Ajustes de configuración**:
   * Aumentar Timeout a 120 segundos
   * Activar KeepAlive en Apache
   * Optimizar parámetros de conexión MySQL
2. **Modificaciones de código**:
   * Implementar reintentos automáticos para consultas fallidas
   * Agregar logs detallados por capa (frontend, backend, BD)

**6.2. Mejoras a Mediano Plazo**

1. **Arquitectura**:
   * Implementar caché para consultas frecuentes
   * Considerar balanceador de carga para peticiones de login
2. **Monitorización**:
   * Dashboard en tiempo real para métricas clave
   * Alertas tempranas para degradación de rendimiento

**7. Lecciones Aprendidas**

1. **Importancia de pruebas combinadas**:
   * Las pruebas GUI (Selenium) y de protocolo (JMeter) se complementan
   * Cada enfoque revela diferentes tipos de problemas
2. **Valor de los datos cuantitativos**:
   * 100 iteraciones proporcionaron patrones estadísticamente significativos
   * Los logs detallados permitieron análisis retrospectivos
3. **Necesidad de ambiente controlado**:
   * XAMPP permitió aislar variables y reproducir consistentemente

**8. Proceso de Documentación**

La estrategia de documentación incluyó:

1. **Repositorio estructurado** con:
   * Directorio para resultados de pruebas
   * Carpeta de configuración de entorno
   * Sección de análisis y conclusiones
2. **Versionado cuidadoso**:
   * Tags para cada ronda de pruebas
   * Commits descriptivos por cada modificación
3. **Artefactos generados**:
   * Reportes ejecutivos resumidos
   * Datos crudos para análisis posterior
   * Capturas de pantalla de errores críticos

**9. Siguientes Pasos**

El roadmap propuesto incluye:

1. **Validación de correcciones**:
   * Re-ejecución de pruebas automatizadas
   * Pruebas de regresión para nuevas versiones
2. **Expansión de cobertura**:
   * Pruebas en múltiples navegadores
   * Escenarios de prueba adicionales
3. **Automatización avanzada**:
   * Integración con pipelines CI/CD
   * Reportes automáticos de calidad

Esta documentación exhaustiva no solo registra los hallazgos técnicos, sino que establece un marco metodológico para futuras iniciativas de garantía de calidad en el sistema.