FLASHBOX

Ingeniería del software II

Plan de Mantenimiento

Gestión del proyecto

Introducción

En este documento se define el plan de mantenimiento del sistema FlashBox, una aplicación desarrollada con Java, Spring Boot, Thymeleaf y base de datos Derby embebida. Este plan tiene como finalidad garantizar la correcta operación, estabilidad, seguridad y evolución del sistema, el cual da servicio a usuarios de tipo cliente, repartidor y restaurante

Dado que el sistema se encuentra en una fase funcional avanzada y ha sido sometido a auditorías con herramientas como SonarCloud, Junit, se establecen acciones de mantenimiento clasificadas en las categorías de correctivo, preventivo, adaptativo y perfectivo

Plan de Mantenimiento

Objetivos

Los objetivos del mantenimiento del sistema FlashBox son:

- Detectar y corregir errores en producción en el menor tiempo posible
- Prevenir futuros problemas mediante monitoreo, backups y actualizaciones
- Adaptar el sistema a nuevas tecnologías, requerimientos o entornos motivados
- Mejorar continuamente el rendimiento, experiencia de usuario y usabilidad

Plan de Mantenimiento

Mantenimiento Correctivo	Diario	Semanal	Mensual	Trimestral	Anual
Monitorear logs y errores del sistema	X				
Atender errores funcionales reportados por usuarios		X			
Revisar incidencias repetidas y aplicar soluciones estructurales			X		
Auditar incidencias para evaluar impacto y calidad de resolución				X	
Revisar código completo para eliminar deuda técnica					X

Mantenimiento Preventivo	Diario	Semanal	Mensual	Trimestral	Anual
Verificar disponibilidad del sistema y su rendimiento	X				
Realizar copias de seguridad incrementales		X			
Actualizar librerías, dependencias y plugins Maven			X		
Ejecutar pruebas de seguridad (OWASP, dependabot, etc.)				X	
Evaluar infraestructura (Derby, despliegue, posibles migraciones)					X

Mantenimiento Adaptativo	Diario	Semanal	Mensual	Trimestral	Anual
Aplicar cambios rápidos por APIs externas o normativa	X				
Adaptar compatibilidad por navegador o sistema operativo		X			
Verificar estabilidad tras adaptaciones			X		
Evaluar cumplimiento de normativas (ej. RGPD)				X	
Analizar nuevas integraciones (pago, geolocalización, etc.)					X

Mantenimiento Perfectivo	Diario	Semanal	Mensual	Trimestral	Anual
Recoger sugerencias de usuarios (clientes, restaurantes, etc.	X				
Optimizar consultas SQL, rendimiento y limpieza de código		X			
Mejorar experiencia de usuario (diseño, formularios, navegación)			X		
Analizar feedback general y planificar nuevas funcionalidades				X	
Añadir mejoras mayores (dashboard, notificaciones, etc.)					X

Plan de pruebas

Objetivos

- Verificar que todas las funcionalidades cumplan con los requisitos definidos
- Asegurar la robustez y confiabilidad del sistema en diferentes escenarios
- Detectar y corregir errores funcionales y de integración
- Validar la cobertura de pruebas automatizadas mediante Junit y SonarCloud

Tipos de Prueba

Pruebas unitarias → validación de métodos individuales en los servicios y controladores

Pruebas de integración→ verificación de la comunicación entre componentes (DAOS, Servicios, controladores)

Pruebas funcionales→ Simulación de casos de uso desde la perspectiva del usuario final

Pruebas de regresión→ repetición de pruebas tras cambios para evitar efectos colaterales

Pruebas de interfaz→ validación de formularios HTML, plantillas Thymeleaf y navegación

Pruebas de cobertura→ Análisis con JaCoCo y SonarCloud para asegurar calidad y profundidad

Herramientas utilizadas ´

- Junit 5 para pruebas unitarias e integración
- Maven Surefire Plugin para ejecución automática
- JaCoCo para métricas de coberturas de código
- SonarCloud para análisis estático y calidad del código
- Postman para pruebas manuales de endpoints
- Navegador (Chrome) para pruebas de interfaz y formularios

Casos de prueba representativos

Registro de Usuario

ID	Caso de prueba	Entrada	Resultado Esperado
T1	Registro de cliente	Formulario válido	Cliente creado y redirigido
T2	Registro con campos vacíos	Campos vacíos	Mensaje de error mostrado
T3	Registro de restaurante	Datos válidos + tipo cocina	Restaurante creado correctamente

Inicio de Sesión

ID	Caso de prueba	Entrada	Resultado Esperado
T4	Login con datos válidos	Usuario y contraseña correctos	Acceso al panel correspondiente
T5	Login con datos inválidos	Contraseña incorrecta	Mensaje de error mostrado
T6	Acceso sin login	URL directa	Redirección a login

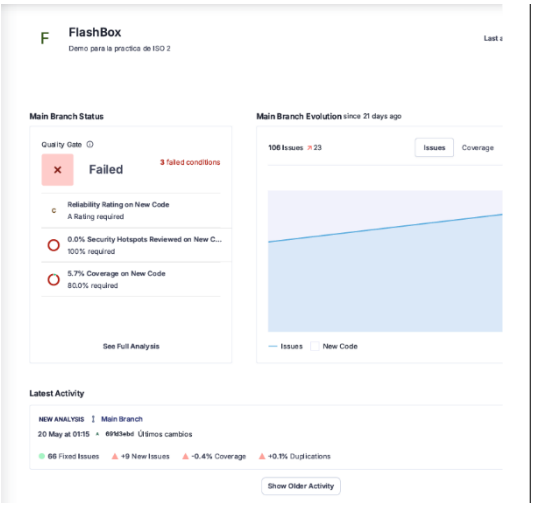
Gestión del menú (Restaurante)

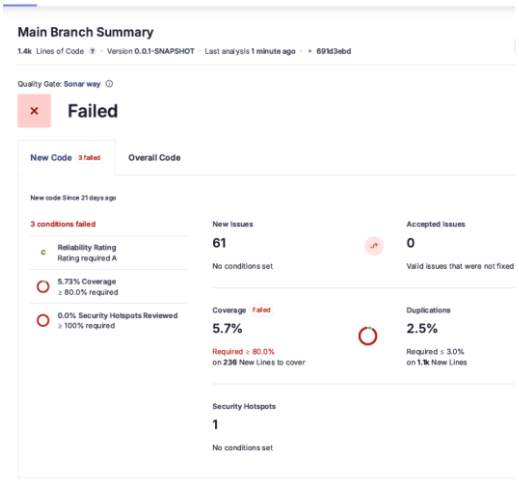
ID	Caso de prueba	Entrada	Resultado Esperado
T7	Crear ítem de menú	Nombre, precio, tipo	Ítem añadido correctamente
T8	Editar ítem	Nuevos datos válidos	Cambios guardados
T9	Eliminar ítem	ID válido	Ítem eliminado del menú

Pedido y Entrega

ID	Caso de prueba	Entrada	Resultado Esperado
T10	Realizar pedido completo	Carrito + dirección	Pedido creado (pendiente de pago)
T11	Asignar repartidor	Pedido válido	Pedido pasa a “En reparto”
T12	Marcar pedido como entregado	Pedido en reparto	Estado: “Entregado”

Resultados esperados de calidad





Métrica	Resultado obtenido	Umbral requerido	Estado
Cobertura de código	5.73%	≥ 80%	Fallido
Duplicación de código	2.5%	≤ 3.0%	Aprobado
Reliability Rating	A (0 bugs críticos)	A	Aprobado
Security Hotspots	0% revisado	100%	Fallido
Quality Gate (Sonar way)	3 condiciones fallidas	-	Fallido

- Aunque no se ha detectado bugs críticos ni problemas de fiabilidad, la cobertura de pruebas unitarias sigue siendo baja
- La duplicación de código está dentro del límite aceptable (25%)
- Es recomendable aumentar la cobertura mediante Junit e inspeccionar posibles puntos de mejora en seguridad y refactorización