**UNIVERSIDAD DE GUAYAQUIL**  
**FACULTAD DE CIENCIAS MATEMÁTICAS Y FÍSICAS**  
**CARRERA DE INGENIERÍA EN SOFTWARE**

**POLÍTICA DE MANTENIMIENTO DEL SISTEMA**

**"GESTIÓN DE REPARACIONES"**

**Asignatura:** Construcción de Software  
**Docente:** [Nombre del docente]

**Estudiante(s):** Joffre Yagual

**Política de Mantenimiento del Sistema: Gestión de Reparaciones**

**1. Introducción**

El mantenimiento de software es un proceso fundamental que garantiza la continuidad operativa, la mejora continua y la adaptación del sistema a nuevas necesidades o contextos tecnológicos. Esta política establece las directrices, tipos de mantenimiento, roles, tiempos de respuesta y procedimientos aplicables al sistema “Gestión de Reparaciones” desarrollado en el marco de la asignatura Construcción de Software.

**2. Objetivos de la política**

* Asegurar la disponibilidad y el correcto funcionamiento del sistema.
* Establecer responsabilidades y tiempos de atención para cada tipo de incidencia.
* Garantizar la trazabilidad de los cambios realizados en el software.
* Mantener la integridad de los datos ante posibles fallos o modificaciones.
* Facilitar la evolución y mejora del sistema con el tiempo.

**3. Alcance**

Esta política aplica a todos los módulos del sistema “Gestión de Reparaciones”, incluyendo:

* Módulo de autenticación (login).
* Registro y gestión de clientes.
* Registro y seguimiento de reparaciones.
* Módulo de visualización y filtrado.
* Funciones de validación, mensajes y control de errores.

Aplica también a todos los archivos de código fuente, base de datos, archivos de configuración y documentación asociada.

**4. Tipos de mantenimiento**

**4.1 Correctivo**

Se refiere a las acciones necesarias para corregir errores que impidan el funcionamiento del sistema. Ejemplos:

* Fallos en el login.
* Errores en consultas a la base de datos.
* Vistas o formularios que no se cargan correctamente.

**Tiempo de respuesta estimado:** 24 a 48 horas, según la criticidad.

**4.2 Adaptativo**

Consiste en realizar modificaciones al sistema para que funcione en nuevos entornos o tecnologías. Ejemplos:

* Migración de base de datos a otro servidor.
* Adaptación a nuevas versiones de PHP o PostgreSQL.

**Tiempo estimado:** Hasta 5 días hábiles, dependiendo de la complejidad.

**4.3 Perfectivo**

Implica mejoras en la interfaz, usabilidad o eficiencia del sistema sin que exista un error como tal. Ejemplos:

* Optimización de tiempo de carga.
* Reorganización visual de los formularios.
* Mejora en mensajes de confirmación o error.

**Tiempo estimado:** Planificado de común acuerdo con el usuario o equipo.

**4.4 Preventivo**

Se realiza con el objetivo de evitar futuros fallos. Incluye:

* Revisión del código para detectar posibles vulnerabilidades.
* Refactorización de funciones innecesariamente complejas.
* Limpieza de datos duplicados en la base.

**Frecuencia recomendada:** 1 vez por trimestre.

**5. Tiempos de respuesta y clasificación de incidencias**

| **Nivel** | **Descripción** | **Tiempo máximo de atención** |
| --- | --- | --- |
| Crítico | El sistema no funciona o compromete la integridad de los datos | 24 horas |
| Alto | Función importante no disponible | 48 horas |
| Medio | Error menor que no impide continuar | 3 días hábiles |
| Bajo | Sugerencia o mejora no urgente | Planificación flexible |

**6. Roles y responsabilidades**

* **Equipo de desarrollo:**  
  Responsable de atender reportes, analizar causas y aplicar soluciones.
* **Usuarios finales (técnicos, administrativos):**  
  Deben reportar errores, describir las condiciones del fallo y evitar manipular los archivos del sistema.
* **Administrador de la base de datos (si aplica):**  
  Encargado de mantener respaldos actualizados y asistir en migraciones o recuperación de datos.

**7. Procedimiento de mantenimiento**

1. El usuario reporta la falla al equipo de desarrollo (vía correo o WhatsApp).
2. El equipo registra la incidencia con fecha y descripción.
3. Se clasifica la severidad y se define el tipo de mantenimiento necesario.
4. Se realiza el análisis técnico y se aplica la solución.
5. Se actualiza la bitácora del proyecto con el cambio realizado.
6. En caso de modificación crítica, se informa al resto del equipo y se realiza respaldo completo antes de aplicar.

**8. Control de versiones y registro de cambios**

Todo cambio en el sistema debe estar registrado en el repositorio Git (en GitHub), acompañado de un mensaje claro sobre qué se cambió y por qué. Se recomienda utilizar etiquetas o ramas para diferenciar versiones estables de versiones en desarrollo.

**9. Backup y recuperación**

* La base de datos será respaldada manualmente antes de cualquier mantenimiento adaptativo o correctivo.
* Se recomienda mantener al menos una copia de respaldo semanal.
* Los archivos .sql de respaldo deben almacenarse en una carpeta segura y con nombre que indique la fecha del respaldo.

**10. Conclusión**

La presente política establece un marco claro para el mantenimiento del sistema, garantizando que el mismo pueda evolucionar sin comprometer su estabilidad. Su cumplimiento facilitará el trabajo colaborativo, la trazabilidad de los cambios y la satisfacción del usuario final.