**DOCUMENTO DE TOLERANCIA A FALLOS**

**Registro de asistencia**

Curso de construcción de Software

(Crespo Arias Denisse Karolina

Esteves Valero Arianna Lisbeth

León Salazar Xavier Alejandro

Mehler Castro Natasha Victoria

Monar Zambrano Nohelya Carolina)

**Tabla de contenido**

[1. **Introducción** 4](#_Toc197510933)

[2. **Fallos Identificados** 4](#_Toc197510934)

[3. **Mecanismos de Tolerancia a Fallos Implementados** 4](#_Toc197510935)

[4. **Prevención de Defectos** 5](#_Toc197510936)

[4.1 Validaciones de entrada 5](#_Toc197510937)

[4.2 Modularidad 5](#_Toc197510938)

[4.3 Controles visuales 5](#_Toc197510939)

[4.4 Prototipado y pruebas exploratorias 5](#_Toc197510940)

[5. **Uso de Redundancia y Diversidad** 5](#_Toc197510941)

[5.1 Redundancia de almacenamiento 5](#_Toc197510942)

[5.2 Diversidad en validaciones 5](#_Toc197510943)

[5.3 Redundancia funcional futura 5](#_Toc197510944)

[6. **Características del Proceso de Desarrollo Confiable** 5](#_Toc197510945)

[6.1 Desarrollo incremental 5](#_Toc197510946)

[6.2 Control de versiones 5](#_Toc197510947)

[6.3 Pruebas frecuentes 5](#_Toc197510948)

[6.4 Diseño centrado en el usuario 5](#_Toc197510949)

[7. **Escenarios de Tolerancia a Fallos** 6](#_Toc197510950)

[7.1 Escenario 1 6](#_Toc197510951)

[7.2 Escenario 2 6](#_Toc197510952)

[7.3 Escenario 3 6](#_Toc197510953)

[7.4 Escenario 4 6](#_Toc197510954)

[7.5 Escenario 5 6](#_Toc197510955)

[8. **Limitaciones y Mejoras Futuras** 7](#_Toc197510956)

# Introducción

La tolerancia a fallos es la capacidad de un sistema para continuar funcionando de manera aceptable ante errores, caídas o condiciones imprevistas. En la aplicación Web PWA de Registro de Asistencia, esta característica es esencial debido a su enfoque multiusuario, la gestión en línea de datos y la necesidad de asegurar una experiencia robusta para el usuario, incluso en caso de fallos de red, errores de validación o problemas en el servidor. Este documento describe los mecanismos implementados para la tolerancia a fallos, así como las prácticas de prevención de defectos y elementos relacionados con el desarrollo de software confiable, según el enfoque planteado por Sommerville.

# Fallos Identificados

|  |  |
| --- | --- |
| **Tipo de Fallo** | **Descripción** |
| **Fallo de conexión a Internet** | El usuario intenta registrar asistencia sin acceso a red. |
| **Errores en la base de datos** | Fallos al guardar, consultar o sincronizar información con MySQL. |
| **Fallos de AJAX** | Errores en la comunicación entre frontend y backend (peticiones AJAX). |
| **Ingreso incompleto de materias** | El usuario intenta registrar asistencia sin haber definido materias. |
| **Registro fuera de horario** | El usuario intenta registrar asistencia en un horario incorrecto. |
| **Recarga inesperada** | Se pierde la edición de una materia si la página se recarga. |
| **Errores de validación** | El formulario acepta datos vacíos o mal estructurados. |
| **Sesión expirada** | El usuario intenta interactuar sin sesión activa. |

# Mecanismos de Tolerancia a Fallos Implementados

|  |  |
| --- | --- |
| **Fallo Previsto** | **Mecanismo de tolerancia implementado** |
| **Sin conexión a Internet** | Validaciones en JavaScript: Se bloquea el acceso a funciones críticas si no hay conexión. |
| **Errores en base de datos** | Captura de excepciones y mensajes de error claros en PHP; manejo de errores PDO. |
| **Fallos de AJAX** | Manejo de respuestas de error (status HTTP y mensajes JSON); mensajes visuales al usuario. |
| **Datos incompletos o inválidos** | Validaciones en frontend y backend: No se permite guardar materias/asistencias con campos vacíos o incorrectos. |
| **Registro sin materias** | El módulo de asistencia muestra advertencia si no hay materias registradas y deshabilita el registro. |
| **Registro fuera de horario** | Lógica que valida el día y la hora actual para permitir o bloquear el registro. |
| **Recarga inesperada** | El formulario de edición se limpia y muestra advertencia para evitar confusión. |
| **Sesión expirada** | Antes de cada acción, se verifica la sesión en PHP y JavaScript; se redirige a login si no está activa. |
| **Interface adaptable** | Diseño responsive que previene errores visuales en dispositivos móviles. |

# Prevención de Defectos

Para aumentar la confiabilidad, se han tomado medidas de prevención de errores en fases tempranas:

* 1. Validaciones de entrada**:** Se bloquea el registro de datos incompletos o erróneos tanto en el frontend (JavaScript) como en el backend (PHP).
  2. Modularidad**:** La aplicación está estructurada por módulos funcionales (materias, asistencias, usuarios), lo que reduce el riesgo de errores cruzados.
  3. Controles visuales**:** Se utiliza retroalimentación visual (alertas, mensajes, colores) para evitar acciones erróneas.
  4. Prototipado y pruebas exploratorias**:** Se realizaron pruebas con usuarios simulados antes de avanzar al backend.

# Uso de Redundancia y Diversidad

Con la integración de backend y base de datos, la aplicación incorpora mecanismos de redundancia y diversidad:

* 1. Redundancia de almacenamiento**:** La información clave se almacena persistentemente en MySQL; los formularios no se envían sin conexión.
  2. Diversidad en validaciones**:** Se implementan validaciones tanto en frontend como en backend para evitar que un error en una sola capa cause corrupción de datos.
  3. Redundancia funcional**:** El uso de sesiones permite mantener la autenticación y prevenir pérdidas de contexto.
  4. **Manejo de errores en AJAX:** Las respuestas del backend ante errores (por ejemplo, usuario no autenticado, error en base de datos) son capturadas y mostradas al usuario.

# Características del Proceso de Desarrollo Confiable

De acuerdo con Sommerville, el desarrollo de software confiable incluye procesos y prácticas que reducen el riesgo de fallos:

* 1. Desarrollo incremental**:** La app se construyó por etapas (materias, horarios, asistencia) permitiendo ajustes tempranos.
  2. Control de versiones**:** El código se gestiona con Git y GitHub para permitir recuperaciones rápidas y trazabilidad de cambios.
  3. Pruebas frecuentes**:** Cada módulo se valida manualmente en distintos dispositivos y navegadores.
  4. Diseño centrado en el usuario**:** a interfaz fue probada con estudiantes para asegurar facilidad de uso y minimizar errores.

# Escenarios de Tolerancia a Fallos

7.1 Escenario 1**: Usuario intenta registrar asistencia sin conexión**

* **Situación:** El estudiante entra a la app sin conexión a Internet.
* **Resultado esperado:** El sistema notifica la falta de conexión y bloquea la acción..
* **Manejo de fallo:** Se muestra un mensaje y se impide el registro hasta recuperar la conexión.

7.2 Escenario 2**: No hay materias registradas**

* **Situación:** El usuario quiere marcar asistencia pero no ha ingresado materias.
* **Resultado esperado:** Se muestra un mensaje informativo: “No se han registrado materias, por favor agregue una primero.”
* **Manejo de fallo:** El botón de asistencia está deshabilitado hasta que se registren materias.

7.3 Escenario 3**: El usuario intenta registrar asistencia fuera del horario de clases**

* **Situación:** Intenta registrar una clase en un horario no programado.
* **Resultado esperado:** El sistema impide el registro.
* **Manejo de fallo:** Se muestra una alerta indicando que no hay clases en ese horario.

7.4 Escenario 4**: El usuario recarga la página durante la edición de una materia**

* **Situación:** Mientras edita una materia, accidentalmente actualiza el navegador.
* **Resultado esperado:** Los datos no guardados se pierden.
* **Manejo de fallo:** Se limpia el formulario y se muestra un mensaje para evitar confusión.

7.5 Escenario 5**: Campos vacíos o inválidos en el formulario**

* **Situación:** El usuario intenta guardar una materia sin completar todos los campos.
* **Resultado esperado:** Se bloquea el envío del formulario.
* **Manejo de fallo:** Aparecen mensajes de error indicando qué campos deben completarse correctamente.

7.6 Escenario 6**: Error en la base de datos o backend**

* **Situación:** El servidor no puede guardar o consultar datos por un error inesperado.
* **Resultado esperado:** El usuario recibe un mensaje claro sobre el fallo.
* **Manejo de fallo:** El backend responde con mensajes JSON de error y el frontend los muestra en pantalla.

# Limitaciones y Mejoras Futuras

Actualmente, aunque el sistema incluye validación en varias capas y manejo de sesiones, aún pueden implementarse mecanismos adicionales como:

* Reintentos automáticos de sincronización ante fallos temporales.
* Backups programados de la base de datos.
* Logs de errores del servidor y monitoreo.
* Alertas de error con detalles técnicos para administradores.
* Implementación de pruebas automatizadas y cobertura de tests de integración.