Universidad Tecnológica de Tehuacán

# Desarrollo Web Profesional


## José Miguel Carrera Pacheco

## Emmanuel Castro Salvador


## 8.-A

# Documentación Técnica: Sistema de Gestión de Incidencias (SGIM)

## 20/01/2026

# Bloque 1: Fundamentos del Proyecto e Investigación

## 1.1 Justificación Técnica: Aplicación Web vs. Página Web

Para el SGIM, se ha determinado que el desarrollo debe ser una Aplicación Web profesional. A diferencia de una página web (cuyo propósito es principalmente informativo y estático), nuestra solución requiere:

- Gestión de Estados Dinámicos: El sistema debe rastrear el ciclo de vida de una incidencia (Pendiente -> En Proceso -> Resuelto).
- Interactividad Compleja: Los usuarios no solo consumen información, sino que generan datos (reportes, carga de imágenes) que activan procesos en el servidor.
- Autenticación y Autorización: Implementación de niveles de acceso estrictos donde un alumno solo puede reportar y ver sus reportes, mientras que un administrador tiene permisos de edición y asignación.

## 1.2 Arquitectura General del Sistema

El equipo opta por una arquitectura Monolítica Modular que asegura la escalabilidad desde el día uno:

- Frontend: Una interfaz reactiva diseñada para la eficiencia operativa, priorizando la velocidad de carga en dispositivos móviles y de escritorio.
- Backend: Una API robusta que centraliza la lógica de negocio, validaciones de seguridad y la comunicación con la base de datos.
- Infraestructura: Uso de contenedores para garantizar que el entorno de desarrollo sea idéntico al de producción, eliminando fallos por diferencias de sistema operativo.

## 1.3 Análisis Comparativo de Soluciones Reales

Como parte de mi proceso de diseño técnico, analicé personalmente dos plataformas líderes que resuelven problemas similares para extraer mejores prácticas aplicables al SGIM:

1. Jira Service Management: Investigué su flujo de tickets. Aunque es extremadamente robusto, noté que para una institución educativa es demasiado complejo. De aquí extraje la necesidad de tener un historial de reportes claro, pero decidí que nuestra interfaz debe ser mucho más directa para reducir el tiempo de reporte del alumno.
2. FixMyStreet: Analicé cómo esta plataforma ciudadana gestiona reportes de baches o luces fundidas. Me inspiré en su flujo de "evidencia visual" y transparencia. Gracias a este análisis, determinamos que el SGIM debe permitir subir una fotografía obligatoria para evitar reportes falsos o poco claros, optimizando así el tiempo del técnico de mantenimiento.

# Bloque 2: Arquitectura de Información y Accesibilidad

## 2.1 Definición y Justificación del Problema Real

En instituciones educativas como la nuestra, los daños en infraestructura (luces, sanitarios, puertas) son constantes. El problema real que identificamos es la informalidad del reporte. Actualmente, se depende de mensajes de WhatsApp o avisos verbales que se olvidan o no tienen seguimiento. El SGIM resuelve esto mediante una plataforma donde el reporte es oficial, tiene un folio y un responsable asignado, eliminando la pérdida de tiempo y la desorganización del área de mantenimiento.

### 2.2 Jerarquías y Navegación Accesible

Para que el sistema sea profesional, estamos aplicando estándares de Arquitectura de Información:

- Jerarquía de Contenido: Priorizamos el estado del reporte por encima de cualquier otro dato, permitiendo que el encargado identifique urgencias de un vistazo.
- Accesibilidad (A11y): El diseño contempla la Navegación por Teclado total. Esto significa que el orden de tabulación es lógico, permitiendo que personas que no pueden usar un mouse puedan navegar y reportar incidencias sin problemas.

### 2.3 Definición de Flujos y Rutas del Sistema

El sistema se ha estructurado en flujos lógicos para evitar pantallas aisladas:

- Rutas Públicas: Acceso limitado al Login y Landing Page informativa.
- Rutas Privadas (Alumno/Personal): Acceso al formulario de reporte y a su historial personal de incidencias.
- Rutas de Administración: Panel exclusivo para el encargado donde puede realizar el CRUD de reportes, cambiar estatus y asignar técnicos.
-

**Estructura del Sitio (Sitemap)**

- Nivel 1: Acceso Público
  - **Landing Page:** Presentación del sistema y beneficios.
  - **Login:** Formulario de acceso con validación de credenciales.
- Nivel 2: Módulo de Usuario (Privado)
  - **Dashboard Usuario:** Vista rápida de reportes activos.
  - **Formulario de Reporte:** Registro de incidencia (Título, Ubicación, Foto).
  - **Historial:** Lista detallada de reportes realizados por el usuario.
- Nivel 3: Módulo Administrativo (Restringido)
  - **Panel de Gestión:** Control total de incidencias institucionales.
  - **Detalle de Incidencia:** Cambio de estados y asignación de técnicos.

**Flujo Crítico: Reporte de Daño**

1. El usuario inicia sesión --> es redirigido al Dashboard.
2. Selecciona "Nuevo Reporte" --> el sistema abre el formulario.
3. El usuario completa los datos y sube foto --> el backend valida la información.
4. El sistema genera un folio y regresa al usuario al historial con el estado "Pendiente".

## Bloque 3: Base Técnica e Infraestructura

Como equipo, hemos establecido los siguientes estándares de trabajo:

- Repositorio Git Profesional: Uso de ramas (feature branching) para el desarrollo de cada módulo.
- Contenerización con Docker: Archivo docker-compose.yml integrado que permite levantar el servidor y la base de datos en un solo comando, asegurando la portabilidad.
- CI/CD (Integración Continua): Configuración de un pipeline que valida automáticamente la sintaxis del código (Linting) antes de permitir la unión de cambios a la rama principal.