UNIVERSIDAD TECNOLÓGICA DE TEHUACÁN

Nombre del alumno:

Alan Baltazar Figueroa

Docente:

José Miguel Carrera Pacheco

Asignatura:

Desarrollo Web Profesional

Rol: DevOps / CI-CD

Proyecto: REVIDA

Equipo: 2

TEHUACÁN, PUEBLA, ENERO-ABRIL 2026

Fundamentos del Proyecto

1. Diferencia entre Página Web y Aplicación Web

Basado en el análisis técnico de nuestro proyecto, esta distinción es crítica para la infraestructura que estoy implementando:

- Página Web (Informativa): Se limita a mostrar información estática sin requerir procesamiento complejo. En nuestro caso, una página web tradicional no sería viable porque no permitiría cubrir necesidades funcionales como la "interacción constante entre usuarios", el "manejo de datos dinámicos" o la "autenticación de usuarios".
- Aplicación Web (REVIDA): Es un sistema complejo que requiere lógica de negocio y gestión de estados. REVIDA se clasifica como aplicación web porque gestiona procesos activos como la "actualización del estado de las donaciones" y el "seguimiento de solicitudes", lo cual requiere un Backend activo y una base de datos.
- Implicación DevOps: Al ser una aplicación, mi rol exige configurar contenedores (Docker) y un entorno de ejecución (Runtime) para soportar la lógica del servidor, no solo un servidor de archivos estáticos.

2. Arquitectura General del Sistema

La arquitectura de REVIDA se divide en tres capas principales que debo orquestar mediante CI/CD:

- Frontend (Interfaz): Desarrollado para presentar una estructura clara y usable. Se encarga de la accesibilidad y la experiencia de usuario.
- Backend (Lógica y Datos): Responsable de la lógica de negocio, gestión de usuarios y control de los estados del sistema (Disponible, Solicitado, Entregado).
- Infraestructura (Mi Responsabilidad):
    - Docker: Utilizamos contenedores para estandarizar el entorno de desarrollo y producción.
    - CI/CD: Implementé un pipeline de integración continua para automatizar la instalación de dependencias y pruebas, asegurando "buenas prácticas desde el inicio del proyecto".

3. Análisis de Plataformas Similares

En nuestra investigación comparativa analizamos dos plataformas existentes para justificar la necesidad técnica de REVIDA:

- Plataforma 1: Facebook Marketplace (Sección Gratis)

    - Limitación detectada: Aunque tiene gran alcance, "no existe un historial organizado de donaciones" ni permite "dar seguimiento al estado de las solicitudes".
    - Solución REVIDA: Nuestra infraestructura soporta un control estricto de estados y un historial estructurado que Marketplace no ofrece.

- Plataforma 2: Freecycle
    - Limitación detectada: Presenta una "accesibilidad deficiente" y "no existe un flujo claro de estados de donación".
    - Solución REVIDA: Desde el inicio integramos accesibilidad y flujos de navegación definidos como requerimientos técnicos.

Arquitectura de Información y Accesibilidad

1. Rutas y Seguridad

Como encargado de la infraestructura, debo asegurar que las rutas definidas en el Sitemap estén correctamente protegidas en el servidor:

- Rutas Públicas: Accesibles para visitantes, como la Landing Page (/), el Catálogo de Donaciones (/donaciones) y el Centro de Soporte (/soporte).
- Rutas Privadas: Requieren validación de sesión (Middleware). Incluyen el Dashboard (/dashboard), el Gestor de Publicaciones (/dashboard/publicar) y el Centro de Mensajería (/mensajes).

2. Accesibilidad desde la Infraestructura

El proyecto considera la accesibilidad como un "eje fundamental" desde el arranque. Mi contribución técnica para asegurar esto incluye:

- Validación de Estándares: La infraestructura debe soportar navegación semántica que permita el "uso de etiquetas y descripciones comprensibles" y un "orden lógico de tabulación" para usuarios que navegan sin mouse.
- Automatización: El repositorio está configurado para permitir revisiones de código que validen estos estándares antes de fusionar cambios a la rama principal.

Evidencia de Contribución Individual

Como parte de mi rol de DevOps en esta etapa inicial:

1. Pipeline CI/CD (Backend): Configuré exitosamente el flujo de trabajo en GitHub Actions para el Backend, solucionando errores de rutas y activando el caché de dependencias (npm ci) para optimizar tiempos de construcción.
2. Estado del Proyecto: Implementé el Status Badge en el README.md principal, permitiendo al equipo visualizar en tiempo real si el build pasa exitosamente (Passing) o falla.

3. Configuración del Repositorio: Estructuré el archivo ci.yml para trabajar en un entorno de monorepo, separando los flujos de trabajo según la carpeta (/backend vs /frontend).