UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO
Programa Académico de Ingeniería Informática y de Sistemas

ESTRATEGIA DE IMPLEMENTACIÓN

Documento Técnico Profesional

PLATAFORMA WEB PARA LA DENUNCIA CIUDADANA DE PROBLEMAS URBANOS

Stack Tecnológico: XAMPP + React.js
Asignatura: Desarrollo de Software I

Docente: Gabriela Zuñiga Rojas

Cusco, Perú - 2025

ÍNDICE DE CONTENIDOS

1. RESUMEN EJECUTIVO

Este documento presenta la estrategia completa de implementación para la Plataforma Web de Denuncia Ciudadana de Problemas Urbanos, alineada con los Objetivos de Desarrollo Sostenible (ODS 11 y ODS 16). La solución utiliza XAMPP como entorno de desarrollo local con MySQL/MariaDB como base de datos, PHP para el backend API, y React.js para el frontend.

1.1 Objetivos del Documento

- Definir la arquitectura técnica completa del sistema
- Establecer el stack tecnológico y justificar las decisiones técnicas
- Proporcionar guías paso a paso para la configuración del entorno
- Detallar el cronograma de desarrollo por fases
- Especificar estándares de código y mejores prácticas

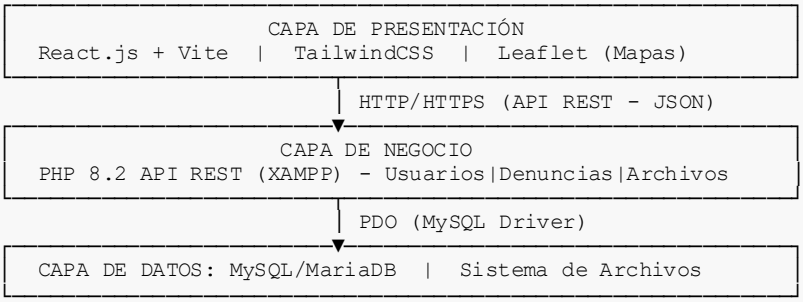
1.2 Alcance Técnico

Componente	Tecnología
Frontend	React.js 18 + Vite + TailwindCSS
Backend	PHP 8.2 (API REST)
Base de Datos	MySQL 8.0 / MariaDB 10.4
Servidor Local	XAMPP (Apache + MySQL + PHP)
Mapas	Leaflet + OpenStreetMap
Control de Versiones	Git + GitHub

2. ARQUITECTURA DEL SISTEMA

2.1 Diagrama de Arquitectura General

La arquitectura sigue el patrón Cliente-Servidor con separación clara entre frontend y backend, comunicándose mediante API REST.



2.2 Flujo de Datos

- **Frontend** → El usuario interactúa con la interfaz React
- **API REST** → Peticiones HTTP (GET, POST, PUT, DELETE) al backend PHP
- **Backend** → Procesamiento de lógica de negocio y validaciones
- **Base de Datos** → Persistencia de información con MySQL/MariaDB
- **Respuesta** → JSON estructurado retornado al frontend

3. STACK TECNOLÓGICO DETALLADO

3.1 Frontend - React.js 18

Librería	Propósito
Vite	Build tool ultrarrápido, HMR instantáneo, configuración mínima
React Router v6	Navegación SPA, rutas protegidas, lazy loading
TailwindCSS	Framework CSS utility-first, diseño responsivo, color primario #9C221C
Axios	Cliente HTTP para consumir API REST, interceptores, manejo de errores
React Leaflet	Componentes React para mapas interactivos con OpenStreetMap
React Hook Form	Gestión de formularios performante, validaciones integradas
Zustand	Estado global minimalista, alternativa ligera a Redux
Chart.js	Gráficos y visualizaciones para dashboard de estadísticas

3.2 Backend - PHP 8.2 con XAMPP

Se implementará una API REST estructurada siguiendo el patrón MVC simplificado:

- **PDO (PHP Data Objects):** Conexión segura a MySQL con prepared statements
- **JWT (JSON Web Tokens):** Autenticación stateless con firebase/php-jwt
- **PHPMailer:** Envío de correos para notificaciones y verificación
- **Composer:** Gestión de dependencias PHP

3.3 Paleta de Colores del Sistema

Variable	Código HEX	Uso
primary	**#9C221C**	Botones principales, enlaces, iconos activos
primary-dark	**#7A1A16**	Estados hover, sombras, énfasis
primary-light	**#FDF6F5**	Fondos de cards, alertas suaves
success	**#22C55E**	Estado resuelto, confirmaciones
warning	**#F59E0B**	Estado en proceso, advertencias
danger	**#EF4444**	Errores, alertas críticas, eliminación

4. ESTRUCTURA DEL PROYECTO

4.1 Estructura de Directorios

```
denuncia-ciudadana/
├── backend/                    # API PHP
│   ├── api/auth/               # login.php, register.php, verify.php
│   ├── api/denuncias/          # create.php, read.php, update.php, delete.php
│   ├── api/archivos/           # Gestión de evidencias
│   ├── api/estadisticas/       # Reportes y métricas
│   ├── config/                 # database.php, cors.php
│   ├── models/                 # Clases de entidades
│   ├── middleware/             # JWT, validaciones
│   └── uploads/                # Archivos subidos
├── frontend/                   # Aplicación React
│   ├── src/components/         # Componentes reutilizables
│   ├── src/pages/              # Vistas principales
│   ├── src/services/           # Llamadas API (Axios)
│   ├── src/store/              # Estado global (Zustand)
│   └── tailwind.config.js      # Config con color #9C221C
├── database/schema.sql         # Estructura de tablas
└── README.md
```

5. CONFIGURACIÓN DEL ENTORNO DE DESARROLLO

5.1 Requisitos Previos

- **XAMPP 8.2+** (incluye Apache, MySQL, PHP)
- **Node.js 18 LTS+** (para React y npm)
- **Git** (control de versiones)
- **VS Code** (IDE recomendado)
- **Composer** (gestor de dependencias PHP)

5.2 Instalación Paso a Paso

Paso 1: Configurar XAMPP

```
# Iniciar servicios de XAMPP
# Windows: Abrir XAMPP Control Panel > Iniciar Apache y MySQL
# Verificar en: http://localhost/phpmyadmin
```

Paso 2: Clonar Repositorio

```
cd C:/xampp/htdocs
git clone https://github.com/[usuario]/denuncia-ciudadana.git
```

Paso 3: Configurar Backend

```
cd denuncia-ciudadana/backend && composer install
```

Paso 4: Configurar Frontend

```
cd ../frontend && npm install && npm run dev
```

5.3 Configuración TailwindCSS

```
// tailwind.config.js
export default {
  theme: { extend: { colors: {
    primary: { DEFAULT:'#9C221C', dark:'#7A1A16', light:'#FDF6F5' }
  }}}
}
```

6. DISEÑO DE BASE DE DATOS

6.1 Tablas Principales

Tabla	Descripción
usuarios	Ciudadanos y autoridades municipales registradas
denuncias	Registro principal de denuncias urbanas con geolocalización
categorias	Tipos de problemas urbanos (baches, alumbrado, basura, etc.)
evidencias	Archivos multimedia (fotos, videos) adjuntos a denuncias
seguimientos	Historial de cambios de estado de cada denuncia
areas_municipales	Dependencias municipales responsables de resolver denuncias
notificaciones	Registro de notificaciones enviadas por email

6.2 Script SQL - Tabla Usuarios

```
-- database/schema.sql
CREATE DATABASE IF NOT EXISTS denuncia ciudadana
CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
USE denuncia ciudadana;

CREATE TABLE usuarios (
  id INT PRIMARY KEY AUTO INCREMENT,
  dni VARCHAR(8) UNIQUE NOT NULL,
  nombres VARCHAR(100) NOT NULL,
  apellidos VARCHAR(100) NOT NULL,
  email VARCHAR(150) UNIQUE NOT NULL,
  password_hash VARCHAR(255) NOT NULL,
  rol ENUM('ciudadano','operador','supervisor','admin') DEFAULT 'ciudadano',
  verificado BOOLEAN DEFAULT FALSE,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

7. GUÍA DE IMPLEMENTACIÓN POR FASES

FASE 1: Fundación (Semanas 1-2)

Objetivo: Establecer la base técnica del proyecto

1. **Configuración del entorno:** Instalar XAMPP, Node.js, Git. Crear repositorio GitHub.
2. **Diseño de base de datos:** Crear schema.sql con todas las tablas, índices y relaciones.
3. **Estructura backend:** Configurar API REST con PHP, establecer conexión PDO a MySQL.
4. **Estructura frontend:** Inicializar proyecto React con Vite, configurar TailwindCSS.
5. **Autenticación básica:** Implementar registro/login con JWT, verificación por email.

Entregables: Prototipo funcional de login/registro, diagrama E-R, repositorio.

FASE 2: Core Features (Semanas 3-4)

Objetivo: Implementar funcionalidades principales de denuncias

1. **Sistema de geolocalización:** Integrar Leaflet + OpenStreetMap, captura de coordenadas GPS.
2. **CRUD de denuncias:** Formulario completo con categorías, descripción, ubicación.
3. **Sistema de archivos:** Upload de fotos/videos con validación, compresión.
4. **Generación de códigos:** Crear código único DU-YYYY-NNNNNN con QR para seguimiento.

Entregables: Módulo completo de denuncias, mapa interactivo funcional.

FASE 3: Gestión Municipal (Semanas 5-6)

Objetivo: Desarrollar herramientas para autoridades

1. **Dashboard administrativo:** Panel con métricas en tiempo real, gráficos con Chart.js.
2. **Sistema de asignaciones:** Derivar denuncias a áreas municipales, notificaciones.
3. **Gestión de respuestas:** Documentar acciones, subir evidencias de resolución.
4. **Portal de consulta pública:** Búsqueda por código sin registro.

Entregables: Dashboard funcional, sistema de asignaciones operativo.

FASE 4: Optimización y Deploy (Semanas 7-9)

Objetivo: Finalizar el sistema y desplegarlo en la nube

1. **Reportes estadísticos:** Generación de informes por categoría, área, período.
2. **Mapa de calor:** Visualización de zonas con mayor incidencia de problemas.
3. **Testing y QA:** Pruebas unitarias, de integración y de usuario.
4. **Despliegue:** Subir aplicación a servidor en la nube (Railway, Vercel, etc.).

Entregables: Sistema completo desplegado, documentación final, exposición.

8. SISTEMA DE AUTENTICACIÓN Y SEGURIDAD

8.1 Flujo de Autenticación JWT

El sistema utiliza JSON Web Tokens (JWT) para autenticación stateless, permitiendo escalabilidad y seguridad.

- **Registro:** Usuario envía datos → PHP valida → Guarda hash bcrypt → Envía email de verificación
- **Login:** Credenciales → Verificación bcrypt → Genera JWT (access + refresh) → Retorna tokens
- **Peticiones:** Frontend envía JWT en header Authorization → Middleware valida → Procesa request
- **Refresh:** Access token expira → Usa refresh token → Obtiene nuevo access token

8.2 Medidas de Seguridad

Medida	Implementación
Cifrado de contraseñas	bcrypt con cost factor 12 (password_hash)
Prepared Statements	PDO con parámetros enlazados para prevenir SQL Injection
Validación de entrada	Sanitización server-side + validación client-side
CORS configurado	Headers restrictivos, solo dominios autorizados
Rate Limiting	Límite de intentos de login (5 intentos/15 min)
XSS Prevention	htmlspecialchars() + Content Security Policy headers

9. INTEGRACIÓN DE MAPAS Y GEOLOCALIZACIÓN

9.1 Componente de Mapa con React-Leaflet

```
// frontend/src/components/MapSelector.jsx
import { MapContainer, TileLayer, Marker, useMapEvents } from 'react-leaflet';
import { useState } from 'react';

const CUSCO_CENTER = [-13.5319, -71.9675]; // Centro: Cusco

export default function MapSelector({ onLocationSelect }) {
  const [position, setPosition] = useState(null);
  return (
    <MapContainer center={CUSCO CENTER} zoom={14}>
      <TileLayer url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png" />
      {position && <Marker position={position} />}
    </MapContainer>
  );
}
```

10. CRONOGRAMA DETALLADO

10.1 Vista General por Semanas

Semana	Actividades Principales
Semana 1	Configuración entorno, diseño BD, estructura proyecto
Semana 2	Sistema de autenticación JWT, registro/login
Semana 3	Integración mapas Leaflet, geolocalización
Semana 4	CRUD denuncias, upload archivos, estados
Semana 5	Dashboard administrativo, gráficos Chart.js
Semana 6	Sistema notificaciones, asignaciones, portal público
Semana 7	Reportes estadísticos, mapa de calor
Semana 8	Optimización, testing, documentación
Semana 9	Despliegue en la nube, pruebas finales, exposición

11. MÉTRICAS DE CALIDAD Y TESTING

11.1 Estrategia de Pruebas

Tipo	Herramienta	Cobertura
Unitarias	PHPUnit, Jest	Funciones críticas, validaciones
Integración	Postman Collections	API endpoints, flujos completos
E2E	Cypress	Flujos de usuario principales
Performance	Lighthouse	Tiempos de carga, Core Web Vitals

11.2 Criterios de Aceptación

- Tiempo de carga inicial < 3 segundos
- API response time < 500ms para operaciones CRUD
- Cobertura de código mínima: 70%
- Zero vulnerabilidades críticas en análisis de seguridad
- Compatibilidad con Chrome, Firefox, Safari, Edge
- Diseño responsivo funcional en dispositivos >= 320px

12. CONCLUSIONES Y RECOMENDACIONES

12.1 Factores Críticos de Éxito

- **Comunicación del equipo:** Reuniones semanales de sincronización, uso activo de GitHub Issues
- **Control de versiones:** Commits descriptivos, branches por feature, code reviews obligatorios
- **Documentación continua:** Actualizar README, documentar APIs, mantener changelog
- **Testing temprano:** Escribir tests junto con el código, no al final del proyecto

12.2 Riesgos y Mitigación

Riesgo	Probabilidad	Mitigación
Retrasos en cronograma	Media	Buffer de 1 semana, priorizar MVP
Problemas de integración	Media	CI/CD, tests automatizados
Curva de aprendizaje	Baja	Documentación, pair programming

12.3 Trabajo Futuro

- Aplicación móvil nativa (React Native)
- Integración con RENIEC para validación automática de DNI
- Sistema de IA para clasificación automática de denuncias
- Chatbot para atención ciudadana 24/7
- Dashboard público con datos abiertos (Open Data)

— Fin del Documento —

Versión 1.0 | Diciembre 2025