Documentación Técnica - Sistema de Gestión de Mantenimientos (PWA-GLPI)

1. Visión General del Sistema

El sistema es una Aplicación Web Progresiva (PWA) diseñada para la gestión de mantenimientos preventivos y correctivos de equipos de cómputo. Su característica principal es la capacidad de operar bajo una arquitectura **Offline-First**, permitiendo a los técnicos registrar actividades, firmar digitalmente y capturar evidencias sin necesidad de conexión a internet permanente.

Arquitectura de Alto Nivel

- **Frontend (Cliente):** SPA desarrollada en React con Vite. Utiliza IndexedDB (vía Dexie.js) para almacenamiento local robusto.
- **Backend (Servidor):** API RESTful en Node.js con Express. Gestiona la lógica de negocio, autenticación, generación de reportes PDF y sincronización con GLPI.
- **Base de Datos:** MongoDB para el almacenamiento centralizado en el servidor.
- **Integración:** Sincronización bidireccional con el sistema GLPI (Gestión Libre de Parque Informático).

2. Stack Tecnológico

Frontend

- **Framework:** React 18
- **Build Tool:** Vite
- **Lenguaje:** JavaScript (ESModules)

- **Estado/Almacenamiento Local:** Dexie.js (Wrapper para IndexedDB)
- **UI/Estilos:** Tailwind CSS, Lucide React (Iconos)
- **Funcionalidades PWA:** vite-plugin-pwa (Service Workers, Manifiesto)

Backend

- **Runtime:** Node.js
- **Framework Web:** Express.js
- **Base de Datos:** Mongoose (ODM para MongoDB)
- **Generación de PDF:** Puppeteer
- **Seguridad:** Helmet, CORS, JWT (JSON Web Tokens)
- **Utilidades:** dotenv, multer (subida de archivos), axios

3. Módulos Principales del Cliente

3.1. Gestión de Estado y Almacenamiento (`store/db.js`)

El núcleo de la persistencia local. Define el esquema de la base de datos IndexedDB para almacenar:

- `users` : Credenciales y perfiles cacheados.
- `tasks` : Tareas asignadas y creadas localmente.
- `acts` : Actas de mantenimiento (borradores y finalizadas).
- `sync_queue` : Cola de elementos pendientes de sincronización.

3.2. Servicio de Sincronización (`services/SyncService.js`)

Maneja la lógica de intercambio de datos entre Cliente y Servidor.

- `syncPendingActs()` : Envía actas finalizadas cuando hay conexión.
- `syncPendingTasks()` : Sincroniza cambios en tareas (estados, nuevas tareas).
- `pullRemoteChanges()` : Descarga nuevas asignaciones o actualizaciones desde el servidor.
- **Estrategia:** "Last Write Wins" con prioridad a la consistencia del servidor para IDs, pero preservando datos locales no sincronizados.

3.3. Componentes Clave

- `App.jsx`: Controlador principal, manejo de rutas (vistas), control de tema (Dark/Light) y detección de estado Online/Offline.
- `MaintenanceForm.jsx`: Formulario dinámico para actas Preventivas/Correctivas. Incluye validaciones y captura de firmas.
- `TaskBoard.jsx`: Vista Kanban para gestión de tareas.
- `SignaturePad.jsx`: Componente para captura de firmas digitales en canvas.
- `PhotoCapture.jsx`: Interfaz para uso de cámara o selección de archivos.

4. API del Servidor (`server/src/routes`)

4.1. Autenticación (`/api/auth`)

- **POST** `/login`: Valida credenciales contra la base de datos (o GLPI) y emite un JWT.

4.2. Tareas (`/api/tasks`)

- **GET** `/all`: Obtiene tareas visibles para el usuario (filtrado por rol/permisos).
- **POST** `/create`: Crea una nueva tarea.
- **PATCH** `/:id`: Actualiza estado o detalles de una tarea.
- **POST** `/sync`: Endpoint para sincronización masiva de tareas desde el cliente.

4.3. Sincronización y Actas (`/api/sync`)

- **POST** `/maintenance`: Recibe un acta completa (JSON), procesa las firmas y fotos, y guarda en MongoDB. Puede desencadenar la creación de tickets en GLPI.

4.4. Reportes (`/api/reports`)

- Generación de documentos PDF basados en las actas sincronizadas, utilizando plantillas HTML renderizadas con Puppeteer.

5. Modelo de Datos (Esquema Conceptual)

Tarea (`Task`)

- `title` : String
- `description` : String
- `status` : Enum (PROGRAMADA, ASIGNADA, EN_EJECUCION, COMPLETADA, CANCELADA)
- `priority` : Enum (BAJA, MEDIA, ALTA)
- `assigned_technicians` : Array[String]
- `scheduled_at` : Date
- `isPrivate` : Boolean

Acta (`Act / Maintenance`)

- `type` : Enum (PREVENTIVO, CORRECTIVO)
- `glpi_ticket_id` : String
- `client_name` : String
- `equipment_details` : Object (serial, modelo, hostname)
- `checklist` : Object (booleans o strings según tipo)
- `signatures` : Object (técnico, cliente - base64)
- `photos` : Array (rutas o base64)

6. Configuración y Despliegue

Requisitos Previos

- Node.js v18+
- MongoDB (Local o Atlas)

Variables de Entorno (.env)

Server:

```
 PORT=5000
MONGO_URI=mongodb://localhost:27017/ticketsign
JWT_SECRET=tu_secreto_seguro
GLPI_URL=https://tu-glpi.com/apirest.php
GLPI_APP_TOKEN=token_app
GLPI_USER_TOKEN=token_user
```

Client:

```
VITE_API_URL=http://localhost:5000/api
```

Ejecución en Desarrollo

1. **Backend:** `npm run dev` (puerto 5000)
2. **Frontend:** `npm run dev` (puerto variable, por defecto 5173)

Construcción para Producción

1. **Frontend:** `npm run build` -> Genera carpeta `dist`.
2. **Backend:** Servir los archivos estáticos de `dist` o usar un servidor web (Nginx/Apache) como proxy reverso hacia el puerto de la API.