Documentación Técnica - DroneAgri Panel

Sistema de Gestión de Servicios de Drones Agrícolas

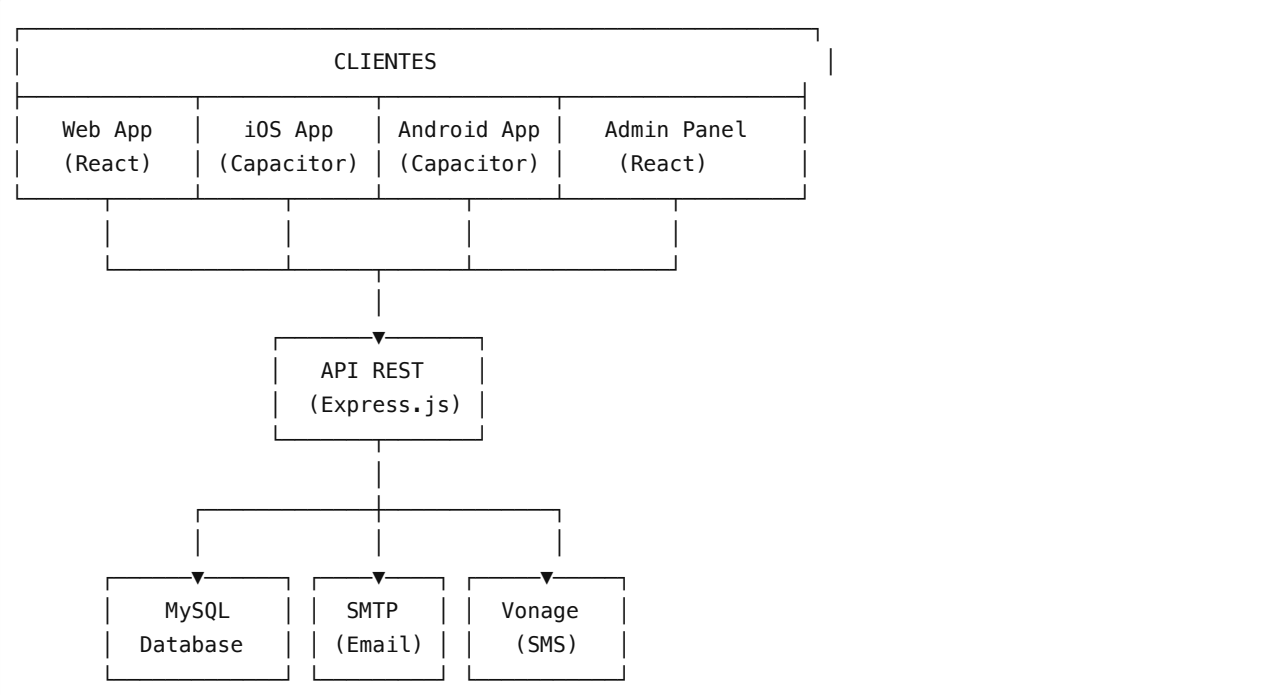
Versión: 1.0 **Fecha:** Enero 2025 **Repositorio:** https://github.com/yasnieldiaz/panel-agricultura-drones

---

Tabla de Contenidos

---

1. Arquitectura del Sistema



---

2. Stack Tecnológico

Frontend

Tecnología	Versión	Uso
React	19.2.0	Framework UI
TypeScript	5.9.3	Tipado estático

Vite	7.3.0	Build tool
Tailwind CSS	3.4.19	Estilos
Framer Motion	12.x	Animaciones
React Router	7.11.0	Enrutamiento
Lucide React	0.562.0	Iconos

Backend

Tecnología	Versión	Uso
Node.js	18+	Runtime
Express.js	4.x	Framework API
MySQL	8.x	Base de datos
JWT	-	Autenticación
bcryptjs	-	Hash passwords
Nodemailer	-	Emails
Vonage SDK	-	SMS

Mobile

Tecnología	Versión	Uso
Capacitor	8.0.0	Bridge nativo
iOS	8.0.0	Plugin iOS
Android	8.0.0	Plugin Android

---

3. Estructura del Proyecto

```
panel-agricultura-drones/
├── src/                    # Código fuente frontend
│   ├── components/         # Componentes React
│   │   ├── AdminRoute.tsx
│   │   ├── CookieConsent.tsx
│   │   ├── LanguageSelector.tsx
│   │   ├── Logo.tsx
│   │   ├── OfflineIndicator.tsx
│   │   ├── ProtectedRoute.tsx
│   │   ├── ServiceRequestModal.tsx
│   │   └── WeatherForecast.tsx
│   ├── contexts/           # Contextos React
│   │   └── LanguageContext.tsx
│   ├── hooks/              # Custom hooks
│   │   ├── useAuth.tsx
│   │   └── useNetworkStatus.tsx
│   ├── lib/                # Utilidades
│   │   ├── api.ts          # Cliente API
```

```
│   │   └── cache.ts         # Sistema cache offline
│   ├── pages/               # Páginas/Vistas
│   │   ├── Landing.tsx
│   │   ├── Auth.tsx
│   │   ├── Dashboard.tsx
│   │   ├── AdminDashboard.tsx
│   │   ├── AdminClients.tsx
│   │   ├── AdminSettings.tsx
│   │   ├── Privacy.tsx
│   │   └── ResetPassword.tsx
│   ├── App.tsx              # Componente raíz
│   ├── main.tsx             # Entry point
│   └── index.css            # Estilos globales
├── api/                     # Backend API
│   └── index.js             # Servidor Express
├── ios/                     # Proyecto iOS (Capacitor)
├── android/                 # Proyecto Android (Capacitor)
├── dist/                    # Build de producción
├── docs/                    # Documentación
├── capacitor.config.ts      # Config Capacitor
├── vite.config.ts           # Config Vite
├── tailwind.config.js       # Config Tailwind
├── tsconfig.json            # Config TypeScript
├── package.json             # Dependencias
└── deploy.sh                # Script de despliegue
```

---

4. Frontend

Rutas de la Aplicación

Ruta	Componente	Acceso	Descripción
`/`	Landing	Público	Página principal
`/auth`	Auth	Público	Login/Registro
`/reset-password`	ResetPassword	Público	Restablecer contraseña
`/privacy`	Privacy	Público	Política de privacidad
`/dashboard`	Dashboard	Usuario	Panel de cliente
`/admin`	AdminDashboard	Admin	Panel administrador
`/admin/settings`	AdminSettings	Admin	Configuración
`/admin/clients`	AdminClients	Admin	Gestión usuarios

Componentes Principales

ProtectedRoute

```
// Protege rutas que requieren autenticación
<ProtectedRoute>
  <Dashboard />
</ProtectedRoute>
```

AdminRoute

```
// Protege rutas que requieren rol admin
<AdminRoute>
  <AdminDashboard />
</AdminRoute>
```

ServiceRequestModal

Modal de 3 pasos para solicitar servicios:

1. Selección de servicio
2. Fecha y hora
3. Datos de contacto

Custom Hooks

useAuth

```
const { user, token, loading, signIn, signOut } = useAuth()
```

- Gestiona estado de autenticación
- Persiste token en localStorage
- Auto-login al cargar la app

useNetworkStatus

```
const { isOnline, wasOffline, resetWasOffline } = useNetworkStatus()
```

- Detecta estado de conexión
- Eventos online/offline
- Flag para reconexión

---

5. Backend API

Base URL

- **Desarrollo:** `http://localhost:3001/api`
- **Producción:** `https://cieniowanie.droneagri.pl/api`

Endpoints de Autenticación

Método	Endpoint	Descripción
POST	`/auth/register`	Registrar usuario
POST	`/auth/login`	Iniciar sesión
POST	`/auth/logout`	Cerrar sesión
GET	`/auth/me`	Usuario actual
POST	`/auth/forgot-password`	Solicitar reset
POST	`/auth/reset-password`	Restablecer password
POST	`/auth/change-password`	Cambiar password

Endpoints de Servicios

Método	Endpoint	Descripción
POST	`/service-requests`	Crear solicitud
GET	`/service-requests`	Mis solicitudes

Endpoints de Admin

Método	Endpoint	Descripción
GET	`/admin/users`	Listar usuarios
POST	`/admin/users`	Crear usuario
DELETE	`/admin/users/:id`	Eliminar usuario
PUT	`/admin/users/:id/password`	Cambiar password
POST	`/admin/users/:id/send-reset`	Enviar reset email
GET	`/admin/service-requests`	Todas las solicitudes
PUT	`/admin/service-requests/:id/status`	Cambiar estado
GET	`/admin/config`	Obtener config
PUT	`/admin/config`	Guardar config
POST	`/admin/config/test-sms`	Test SMS
POST	`/admin/config/test-email`	Test email

Formato de Respuesta

```
// Éxito
{
  "success": true,
  "data": { ... },
  "message": "Operación exitosa"
}

// Error
{
  "error": "Mensaje de error"
}
```

Autenticación API

```
Authorization: Bearer <jwt_token>
```

---

6. Base de Datos

Esquema MySQL

Tabla: users

```sql
CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  email VARCHAR(255) UNIQUE NOT NULL,
  password VARCHAR(255) NOT NULL,
  name VARCHAR(255),
  role ENUM('user', 'admin') DEFAULT 'user',
  language VARCHAR(5) DEFAULT 'es',
  reset_token VARCHAR(255),
  reset_token_expiry DATETIME,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Tabla: service_requests

```sql
CREATE TABLE service_requests (
  id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT,
  service_type VARCHAR(50) NOT NULL,
  scheduled_date DATE NOT NULL,
  scheduled_end_date DATE,
  scheduled_time TIME NOT NULL,
  name VARCHAR(255) NOT NULL,
  email VARCHAR(255) NOT NULL,
  phone VARCHAR(50) NOT NULL,
  location VARCHAR(500) NOT NULL,
  area DECIMAL(10,2),
  notes TEXT,
  status ENUM('pending','confirmed','in_progress','completed','cancelled') DEFAULT 'pending',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id)
);
```

Tipos de Servicio

- `fumigation` - Fumigación
- `painting` - Pintura invernaderos
- `mapping` - Mapeo aéreo
- `elevation` - Modelos de elevación
- `rental` - Alquiler de drones
- `repair` - Reparación y mantenimiento

Estados de Solicitud

- `pending` - Pendiente
- `confirmed` - Confirmado
- `in_progress` - En progreso
- `completed` - Completado
- `cancelled` - Cancelado

---

7. Autenticación

Flujo JWT

```
1. Usuario envía credenciales
   POST /auth/login { email, password }

2. Servidor valida y genera JWT
   { user, token }

3. Cliente almacena token
   localStorage.setItem('auth_token', token)

4. Requests autenticados
   Authorization: Bearer <token>

5. Servidor valida token en cada request
```

Estructura del Token

```json
{
  "userId": 1,
  "email": "user@example.com",
  "role": "user",
  "iat": 1234567890,
  "exp": 1234657890
}
```

Expiración

- Token válido por 24 horas
- Reset token válido por 1 hora

---

8. Modo Offline

Sistema de Cache

```javascript
// src/lib/cache.ts
const CACHE_PREFIX = 'app_cache_'
const CACHE_EXPIRY = 24 * 60 * 60 * 1000 // 24 horas

// Claves de cache
CACHE_KEYS = {
  USER: 'current_user',
  SERVICE_REQUESTS: 'service_requests',
  ALL_SERVICE_REQUESTS: 'all_service_requests',
  USERS: 'admin_users'
}
```

Flujo Offline

```
1. Request al servidor
   ↓
2. ¿Éxito?
   ├─ SÍ → Guardar en cache + Retornar datos
   └─ NO → ¿Offline?
```

```
        ├─ SÍ → Retornar datos de cache
        └─ NO → Lanzar error
```

Componente OfflineIndicator

```
// Detecta estado de red
const { isOnline, wasOffline } = useNetworkStatus()

// Muestra banners
- Offline: Banner amarillo "Sin conexión"
- Reconexión: Banner verde "¡Conexión restaurada!"
```

---

9. Internacionalización

Idiomas Soportados

Código	Idioma	Locale
`es`	Español	es-ES
`en`	English	en-US
`pl`	Polski	pl-PL
`cs`	Čeština	cs-CZ
`sk`	Slovenčina	sk-SK
`it`	Italiano	it-IT

Uso

```
// Contexto
const { t, language, setLanguage } = useLanguage()

// Traducir
t('dashboard.welcome') // "Bienvenido"

// Cambiar idioma
setLanguage('en')
```

Estructura de Traducciones

```
// src/contexts/LanguageContext.tsx
const translations = {
  es: {
    'nav.services': 'Servicios',
    'dashboard.welcome': 'Bienvenido',
    // ...
  },
  en: {
    'nav.services': 'Services',
    'dashboard.welcome': 'Welcome',
    // ...
```

```
  },
  // ...
}
```

---

10. Despliegue

Script de Despliegue

```
./deploy.sh
```

Ejecuta:

1. `npm run build` - Compila frontend
2. Copia `dist/` al servidor via SCP
3. Copia `api/` al servidor
4. `npm install` en servidor
5. `pm2 restart` para reiniciar API

Servidor de Producción

- **URL:** https://cieniowanie.droneagri.pl
- **Frontend:** Servido por Nginx
- **API:** PM2 + Node.js
- **DB:** MySQL 8.x

Comandos PM2

```
# Ver estado
pm2 status

# Reiniciar
pm2 restart panel-drones-api

# Logs
pm2 logs panel-drones-api

# Monitoreo
pm2 monit
```

---

11. Apps Móviles

Capacitor

```
# Sincronizar cambios
npx cap sync

# Abrir en Xcode
npx cap open ios

# Abrir en Android Studio
npx cap open android
```

Configuración

```
// capacitor.config.ts
const config: CapacitorConfig = {
  appId: 'com.droneagri.panel',
  appName: 'DroneAgri',
  webDir: 'dist',
  server: {
    androidScheme: 'https'
  }
}
```

Build iOS

1. `npm run build`
2. `npx cap sync ios`
3. `npx cap open ios`
4. En Xcode: Product → Archive
5. Subir a App Store Connect

Build Android

1. `npm run build`
2. `npx cap sync android`
3. `npx cap open android`
4. En Android Studio: Build → Generate Signed Bundle

---

12. Variables de Entorno

Frontend (.env)

```
VITE_API_URL=https://cieniowanie.droneagri.pl/api
```

Backend (.env)

```
# Base de datos
DB_HOST=localhost
DB_USER=root
DB_PASSWORD=password
DB_NAME=drone_panel

# JWT
JWT_SECRET=your-secret-key-minimum-32-characters

# Email (SMTP)
SMTP_HOST=smtp.gmail.com
SMTP_PORT=587
SMTP_USER=your-email@gmail.com
SMTP_PASS=your-app-password
FROM_EMAIL=noreply@droneagri.pl

# SMS (Vonage)
VONAGE_API_KEY=your-api-key
```

```
VONAGE_API_SECRET=your-api-secret
VONAGE_FROM_NUMBER=DroneAgri

# Server
PORT=3001
```

Comandos de Desarrollo

```
# Instalar dependencias
npm install

# Desarrollo
npm run dev

# Build producción
npm run build

# Preview build
npm run preview

# Lint
npm run lint

# Sincronizar móvil
npx cap sync

# Desplegar
./deploy.sh
```

Seguridad

Medidas Implementadas

1. **Helmet.js** - Headers de seguridad HTTP
2. **Rate Limiting** - Prevención de brute force
3. **bcrypt** - Hash de contraseñas (salt rounds: 10)
4. **JWT** - Tokens firmados y con expiración
5. **CORS** - Configurado para dominios permitidos
6. **Input Validation** - Validación en frontend y backend
7. **SQL Injection** - Prepared statements (mysql2)

Recomendaciones

- Mantener JWT_SECRET seguro y rotarlo periódicamente
- Usar HTTPS en producción
- Configurar backups de base de datos
- Monitorear logs de errores
- Actualizar dependencias regularmente

Documentación generada para DroneAgri Panel v1.0