

Technical Documentation - DroneAgri Panel

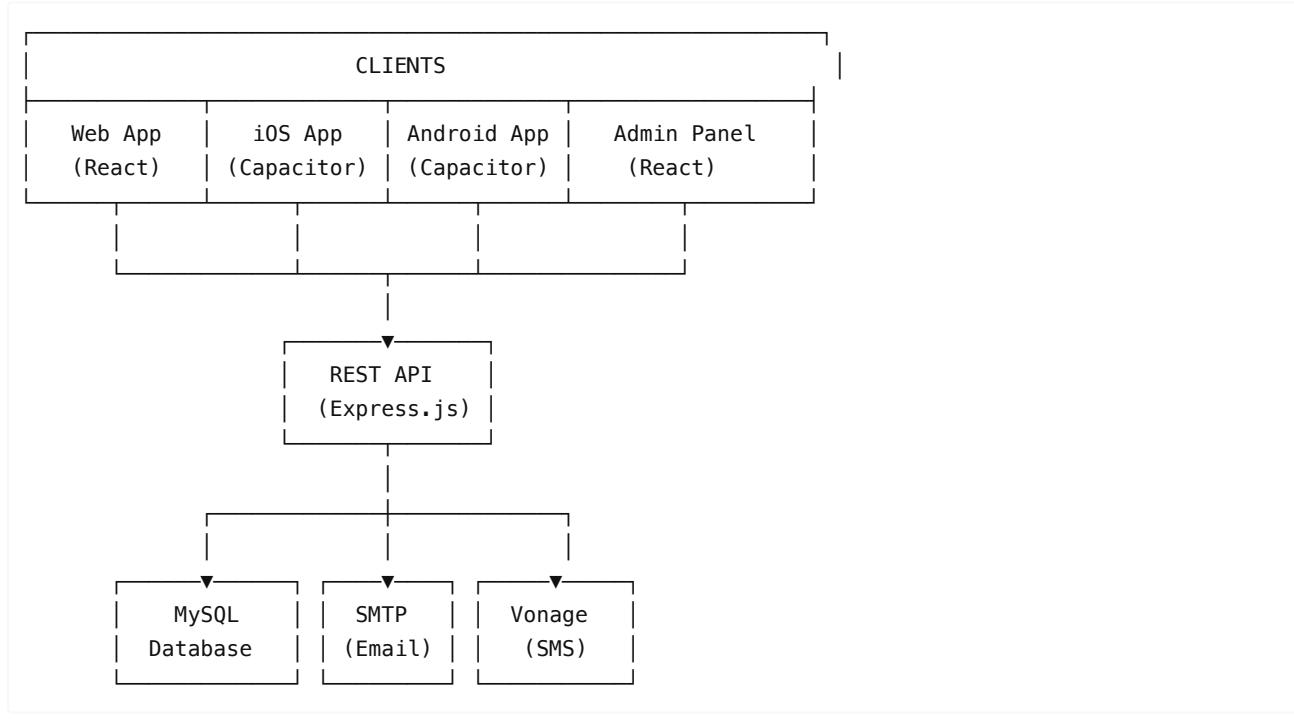
Agricultural Drone Services Management System

Version: 1.0 Date: January 2025 Repository: <https://github.com/yasnieldiaz/panel-agricultura-drones>

Table of Contents

1. [System Architecture](#)
2. [Technology Stack](#)
3. [Project Structure](#)
4. [Frontend](#)
5. [Backend API](#)
6. [Database](#)
7. [Authentication](#)
8. [Offline Mode](#)
9. [Internationalization](#)
10. [Deployment](#)
11. [Mobile Apps](#)
12. [Environment Variables](#)

1. System Architecture



2. Technology Stack

Frontend

Technology	Version	Use
React	19.2.0	UI Framework
TypeScript	5.9.3	Static typing

Vite	7.3.0	Build tool
Tailwind CSS	3.4.19	Styling
Framer Motion	12.x	Animations
React Router	7.11.0	Routing
Lucide React	0.562.0	Icons

Backend

Technology	Version	Use
Node.js	18+	Runtime
Express.js	4.x	API Framework
MySQL	8.x	Database
JWT	-	Authentication
bcryptjs	-	Password hashing
Nodemailer	-	Emails
Vonage SDK	-	SMS

Mobile

Technology	Version	Use
Capacitor	8.0.0	Native bridge
iOS	8.0.0	iOS Plugin
Android	8.0.0	Android Plugin

3. Project Structure

```

panel-agricultura-drones/
├── src/                      # Frontend source code
│   ├── components/            # React components
│   │   ├── AdminRoute.tsx
│   │   ├── CookieConsent.tsx
│   │   ├── LanguageSelector.tsx
│   │   ├── Logo.tsx
│   │   ├── OfflineIndicator.tsx
│   │   ├── ProtectedRoute.tsx
│   │   ├── ServiceRequestModal.tsx
│   │   └── WeatherForecast.tsx
│   ├── contexts/              # React contexts
│   │   └── LanguageContext.tsx
│   ├── hooks/                 # Custom hooks
│   │   ├── useAuth.tsx
│   │   └── useNetworkStatus.tsx
│   ├── lib/                   # Utilities
│   |   └── api.ts             # API client

```

```

|   |   └── cache.ts      # Offline cache system
|   └── pages/           # Pages/Views
|       ├── Landing.tsx
|       ├── Auth.tsx
|       ├── Dashboard.tsx
|       ├── AdminDashboard.tsx
|       ├── AdminClients.tsx
|       ├── AdminSettings.tsx
|       ├── Privacy.tsx
|       └── ResetPassword.tsx
|   └── App.tsx          # Root component
|   └── main.tsx         # Entry point
|   └── index.css        # Global styles
└── api/
    └── index.js        # Backend API
└── ios/               # iOS project (Capacitor)
└── android/           # Android project (Capacitor)
└── dist/              # Production build
└── docs/              # Documentation
└── capacitor.config.ts # Capacitor config
└── vite.config.ts     # Vite config
└── tailwind.config.js # Tailwind config
└── tsconfig.json      # TypeScript config
└── package.json        # Dependencies
└── deploy.sh          # Deployment script

```

4. Frontend

Application Routes

Route	Component	Access	Description
/	Landing	Public	Main page
/auth	Auth	Public	Login/Register
/reset-password	ResetPassword	Public	Reset password
/privacy	Privacy	Public	Privacy policy
/dashboard	Dashboard	User	Client panel
/admin	AdminDashboard	Admin	Admin panel
/admin/settings	AdminSettings	Admin	Settings
/admin/clients	AdminClients	Admin	User management

Main Components

ProtectedRoute

```
// Protects routes that require authentication
<ProtectedRoute>
  <Dashboard />
</ProtectedRoute>
```

AdminRoute

```
// Protects routes that require admin role
<AdminRoute>
  <AdminDashboard />
</AdminRoute>
```

ServiceRequestModal

3-step modal for requesting services:

1. Service selection
2. Date and time
3. Contact details

Custom Hooks

useAuth

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

- Manages authentication state
- Persists token in localStorage
- Auto-login on app load

useNetworkStatus

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

- Detects connection state
- Online/offline events
- Reconnection flag

5. Backend API

Base URL

- Development: <http://localhost:3001/api>
- Production: <https://cieniowanie.droneagri.pl/api>

Authentication Endpoints

Method	Endpoint	Description
POST	/auth/register	Register user
POST	/auth/login	Log in
POST	/auth/logout	Log out
GET	/auth/me	Current user
POST	/auth/forgot-password	Request reset
POST	/auth/reset-password	Reset password
POST	/auth/change-password	Change password

Service Endpoints

Method	Endpoint	Description
POST	/service-requests	Create request
GET	/service-requests	My requests

Admin Endpoints

Method	Endpoint	Description
GET	/admin/users	List users
POST	/admin/users	Create user
DELETE	/admin/users/:id	Delete user
PUT	/admin/users/:id/password	Change password
POST	/admin/users/:id/send-reset	Send reset email
GET	/admin/service-requests	All requests
PUT	/admin/service-requests/:id/status	Change status
GET	/admin/config	Get config
PUT	/admin/config	Save config
POST	/admin/config/test-sms	Test SMS
POST	/admin/config/test-email	Test email

Response Format

```
// Success
{
  "success": true,
  "data": { ... },
  "message": "Operation successful"
}

// Error
{
  "error": "Error message"
}
```

API Authentication

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

6. Database

MySQL Schema

Table: users

```

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
);

```

Table: service_requests

```

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)
);

```

Service Types

- `fumigation` - Fumigation
- `painting` - Greenhouse painting
- `mapping` - Aerial mapping
- `elevation` - Elevation models
- `rental` - Drone rental
- `repair` - Repair and maintenance

Request Status

- `pending` - Pending
- `confirmed` - Confirmed
- `in_progress` - In progress
- `completed` - Completed
- `cancelled` - Cancelled

7. Authentication

JWT Flow

1. User sends credentials
POST /auth/login { email, password }
2. Server validates and generates JWT
{ user, token }
3. Client stores token
localStorage.setItem('auth_token', token)
4. Authenticated requests
Authorization: Bearer <token>
5. Server validates token on each request

Token Structure

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

Expiration

- Token valid for 24 hours
- Reset token valid for 1 hour

8. Offline Mode

Cache System

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

// Cache keys
CACHE_KEYS = {
  USER: 'current_user',
  SERVICE_REQUESTS: 'service_requests',
  ALL_SERVICE_REQUESTS: 'all_service_requests',
  USERS: 'admin_users'
}
```

Offline Flow

1. Request to server
↓
2. Success?
└ YES → Save to cache + Return data
└ NO → Offline?

```
|— YES → Return cached data  
└— NO → Throw error
```

OfflineIndicator Component

```
// Detects network state  
const { isOnline, wasOffline } = useNetworkStatus()  
  
// Shows banners  
- Offline: Yellow banner "No connection"  
- Reconnection: Green banner "Connection restored!"
```

9. Internationalization

Supported Languages

Code	Language	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

Usage

```
// Context  
const { t, language, setLanguage } = useLanguage()  
  
// Translate  
t('dashboard.welcome') // "Welcome"  
  
// Change language  
setLanguage('en')
```

Translation Structure

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

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

10. Deployment

Deployment Script

```
./deploy.sh
```

Executes:

1. `npm run build` - Compiles frontend
2. Copies `dist/` to server via SCP
3. Copies `api/` to server
4. `npm install` on server
5. `pm2 restart` to restart API

Production Server

- **URL:** <https://cieniowanie.droneagri.pl>
- **Frontend:** Served by Nginx
- **API:** PM2 + Node.js
- **DB:** MySQL 8.x

PM2 Commands

```
# View status  
pm2 status  
  
# Restart  
pm2 restart panel-drones-api  
  
# Logs  
pm2 logs panel-drones-api  
  
# Monitoring  
pm2 monit
```

11. Mobile Apps

Capacitor

```
# Sync changes  
npx cap sync  
  
# Open in Xcode  
npx cap open ios  
  
# Open in Android Studio  
npx cap open android
```

Configuration

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

iOS Build

1. npm run build
2. npx cap sync ios
3. npx cap open ios
4. In Xcode: Product → Archive
5. Upload to App Store Connect

Android Build

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

12. Environment Variables

Frontend (.env)

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

Backend (.env)

```
# Database
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
```

Development Commands

```
# Install dependencies  
npm install  
  
# Development  
npm run dev  
  
# Production build  
npm run build  
  
# Preview build  
npm run preview  
  
# Lint  
npm run lint  
  
# Sync mobile  
npx cap sync  
  
# Deploy  
./deploy.sh
```

Security

Implemented Measures

1. **Helmet.js** - HTTP security headers
2. **Rate Limiting** - Brute force prevention
3. **bcrypt** - Password hashing (salt rounds: 10)
4. **JWT** - Signed tokens with expiration
5. **CORS** - Configured for allowed domains
6. **Input Validation** - Frontend and backend validation
7. **SQL Injection** - Prepared statements (mysql2)

Recommendations

- Keep JWT_SECRET secure and rotate it periodically
- Use HTTPS in production
- Configure database backups
- Monitor error logs
- Update dependencies regularly