

## **Documentación - Aplicación de Notificaciones de Invernadero**

La aplicación AlertApp representa una solución integral para el monitoreo de invernaderos inteligentes, desarrollada específicamente para la plataforma Android utilizando React Native y Android Studio como entorno de desarrollo principal.

El proyecto combina tecnologías IoT modernas con una interfaz móvil nativa, donde la arquitectura basada en MQTT garantiza comunicación eficiente y en tiempo real. El desarrollo en Android Studio proporciona herramientas robustas de debugging, testing y optimizaciones específicas para dispositivos Android.

La implementación de sistemas de predicción y detección de objetos añade valor agregado al sistema básico de monitoreo, convirtiendo la aplicación en una herramienta completa para la gestión moderna de invernaderos con enfoque en la plataforma móvil más utilizada globalmente. **Documentación - Aplicación de Notificaciones de Invernadero**

### **1. Descripción General**

La aplicación AlertApp es una solución móvil desarrollada en React Native para el monitoreo en tiempo real de un sistema de invernadero inteligente. La aplicación recibe datos de sensores IoT a través del protocolo MQTT y presenta alertas y mediciones en una interfaz intuitiva.

#### **Características Principales**

- Monitoreo en tiempo real de sensores IoT
- Comunicación MQTT para recepción de datos
- Interfaz adaptativa (modo claro/oscuro)
- Historial de datos con límite de 50 registros
- Detección de objetos mediante visión artificial

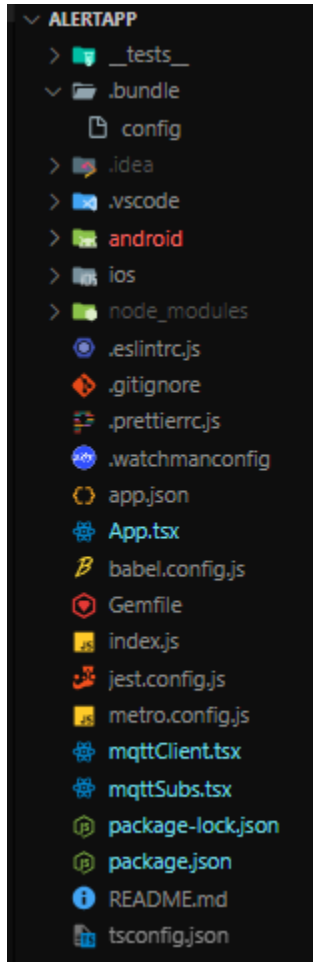
### **2. Arquitectura del Sistema**

#### **2.1. Tecnologías Utilizadas**

- **React Native (v0.79.2)** - Framework de desarrollo móvil
- **Android Studio** - Entorno de desarrollo integrado (IDE)
- **TypeScript** - Tipado estático
- **MQTT (v5.13.0)** - Protocolo de comunicación IoT

- **React Native AsyncStorage** - Almacenamiento local
- **React Native Sound** - Notificaciones sonoras

## 2.2. Estructura del Proyecto



## 3. Funcionalidades del Sistema

### 3.1. Monitoreo de Variables Ambientales

#### **Sensores de Suelo** (``iot/suelo``)

- Humedad del suelo: Medición de la humedad presente en el sustrato
- Valores de referencia: Permite determinar necesidades de riego

#### **Sensores Ambientales** (``iot/ambiente``)

- Temperatura ambiente: Monitoreo de temperatura del invernadero
- Humedad relativa: Control de humedad del aire

### **Nivel de Agua** (``iot/nivel``)

- Medición horizontal: Nivel de agua en tanques de almacenamiento
- Medición vertical: Altura del nivel de agua

## **3.2. Sistemas de Predicción**

### **Predicción de Temperatura** (``iot/temperatura/prediccion``)

- Algoritmos de predicción para anticipar cambios térmicos
- Ayuda en la toma de decisiones preventivas

### **Predicción de Humedad** (``iot/humedad/predicción``)

- Pronóstico de niveles de humedad
- Optimización del riego automatizado

## **3.3. Sistema de Detección de Objetos**

### **Detección de Personas** (``deteccion/personas``)

- **Objetos detectables:** `person`, `vase`, `suitcase`
- **Visión artificial:** Procesamiento de imágenes en tiempo real
- **Seguridad:** Monitoreo de acceso al invernadero
- **Filtrado de datos:** Exclusión de ``file_id`` y ``fecha_deteccion`` en la visualización

## **4. Implementación Técnica**

### **4.1. Configuración MQTT**

```
1  ``typescript
2  // Configuración del broker MQTT
3  const host = '192.168.204.153';
4  const port = 9001;
5  const options = {
6    reconnectPeriod: 1000,
7    connectTimeout: 30 * 1000,
8  };
9  ``
```

## 4.2. Topics MQTT Suscritos

```
1 "iot/suelo" // Datos de sensores de suelo
2 "iot/ambiente" // Datos ambientales
3 "iot/nivel" // Nivel de agua
4 "deteccion/personas" // Detección de objetos
5 "iot/temperatura/prediccion" // Predicción térmica
6 "iot/humedad/prediccion" // Predicción de humedad
```

## 4.3. Estructura de Datos

```
1  ``typescript
2  // Formato de Mensaje
3
4  type SensorData = {
5    topic: string;
6    data: any;
7    time: string;
8  };
9  ``
```

### ***Procesamiento de Datos***

- **Parsing JSON:** Conversión automática de mensajes MQTT
- **Timestamp:** Registro de tiempo de recepción
- **Límite de registros:** Máximo 50 elementos en memoria
- **Filtrado específico:** Procesamiento especial para detección de personas

## 5. Interfaz de Usuario

### 5.1. Características de la UI

- **Diseño responsivo:** Adaptación a diferentes tamaños de pantalla
- **Tema adaptativo:** Soporte para modo claro y oscuro automático
- **Lista scrolleable:** Visualización de historial de datos
- **Cards informativas:** Presentación organizada de cada sensor

### 5.2. Estados de la Aplicación

- **Estado de espera:** *"Esperando datos de sensores..."*
- **Visualización activa:** Lista en tiempo real de mediciones
- **Actualización automática:** Refresh instantáneo de datos

## 6. Configuración y Despliegue

### 6.1. Requisitos del Sistema


#### ***Entorno de Desarrollo***

- **Node.js:**  $\geq 18.0.0$
- **Android Studio:** IDE principal para desarrollo Android
- **Android SDK:** API level mínimo requerido
- **Java Development Kit (JDK):** Para compilación Android
- **React Native CLI:** Herramientas de línea de comandos

#### ***Configuración de Android Studio***

- **Emulador Android:** Para pruebas en dispositivos virtuales
- **USB Debugging:** Para pruebas en dispositivos físicos
- **Network Security Config:** Para conexiones MQTT no seguras


## 6.2. Instalación



```
1  ``bash
2  # Instalación de dependencias
3  npm install
4
5  # Para iOS (adicional)
6  bundle install
7  bundle exec pod install
8  ``
```

## 6.3. Ejecución

Desarrollo en Android Studio



```
1  ``bash
2  # Iniciar Metro bundler
3  npm start
4
5  # Compilar para Android
6  npm run android
7
8  # Compilar para iOS
9  npm run ios
10
11 # O abrir directamente en Android Studio
12 # File > Open > [directorio del proyecto]
13 # Run > Run 'app'
14  ``
```

### ***Configuración del Emulador***

- Abrir Android Studio
- Tools > AVD Manager
- Create Virtual Device
- Seleccionar dispositivo y API level
- Ejecutar el emulador antes de ``npm run android``

### ***Debugging***

- **React Native Debugger:** Para debugging de JavaScript
- **Android Studio Logcat:** Para logs nativos de Android
- **Metro Bundler:** Hot reload automático durante desarrollo

## **7. Casos de Uso**

### **7.1. Monitoreo Continuo**

- Supervisión 24/7 de condiciones del invernadero
- Alertas automáticas ante cambios críticos
- Historial de tendencias de sensores

### **7.2. Gestión Preventiva**

- Predicciones para anticipar problemas
- Optimización de recursos (agua, energía)
- Mantenimiento predictivo de equipos

### **7.3. Seguridad y Control de Acceso**

- Detección automática de personas
- Registro de eventos de seguridad
- Monitoreo de objetos no autorizados

## **8. Beneficios del Sistema**

### **8.1. Eficiencia Operativa**

- Automatización del monitoreo
- Reducción de intervención manual
- Optimización de recursos

### **8.2. Toma de Decisiones**

- Datos en tiempo real para decisiones inmediatas

- Análisis predictivo para planificación
- Historial para análisis de tendencias

### **8.3. Escalabilidad**

- Fácil adición de nuevos sensores
- Arquitectura modular
- Integración con sistemas existentes

## **9. Consideraciones Técnicas**

### **9.1. Consideraciones de Desarrollo Android**

#### ***Configuración de Red***

- **Network Security Config:** Necesario para conexiones MQTT WebSocket
- **Permisos de Internet:** Declarados en AndroidManifest.xml
- **Conexiones no HTTPS:** Configuración especial para broker MQTT local

#### ***Rendimiento Android***

- **Memoria:** Optimización para dispositivos Android con recursos limitados
- **Batería:** Manejo eficiente de conexiones MQTT en background
- **Conectividad:** Reconexión automática ante cambios de red

#### ***Debugging y Testing***

- **Android Studio Profiler:** Monitoreo de performance
- **Logcat:** Logs detallados de la aplicación
- **Device Testing:** Pruebas en múltiples versiones de Android

### **9.2. Conectividad y Red**

- Dependencia de red WiFi estable
- Reconexión automática MQTT
- Manejo de errores de conexión
- Configuración de network security para Android

### **9.3. Rendimiento y Optimización**

- Límite de 50 registros para optimizar memoria
- Procesamiento eficiente de JSON
- Actualización de UI optimizada
- Optimizaciones específicas para Android



#### **9.4. Mantenimiento y Desarrollo**

- Logs de errores para debugging en Android Studio
- Estructura de código modular
- Documentación de APIs
- Proceso de build y distribución APK

### **10. Conclusiones**

La aplicación AlertApp proporciona una plataforma robusta y eficiente para el monitoreo de invernaderos inteligentes, integrando tecnologías IoT con visión artificial y sistemas predictivos. Su diseño modular y enfoque en la experiencia móvil facilitan la adopción y escalabilidad, garantizando un monitoreo en tiempo real confiable y optimizado para dispositivos Android.