ESCUELA POLITÉCNICA SUPERIOR DE CÓRDOBA
Universidad de Córdoba

EPSC

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Sistema para la gestión y control de sensor ITH en explotaciones ganaderas
System for the management and control of THI sensors in livestock farms

Manual de Código (Tipología B)

Autor: Javier Romero Ramos
Directores: Ezequiel Herruzo Gómez
Miguel Ángel Montijano Vizcaíno

ROMERO RAMOS JAVIER - 46267837 W

Firmado digitalmente por ROMERO RAMOS JAVIER - 46267837W
Fecha: 2025.09.10 13:28:36 +02'00'

UNIVERSIDAD DE CÓRDOBA

Índice general

1 Introducción **4**

2 Requisitos técnicos del sistema **5**
2.1 Sistema de adquisición de datos 5
2.2 Sistema de recolección y envío de datos 5
2.3 Sistema de gestión de datos 5
2.4 Sistema de almacenamiento de datos 6

3 Descripción del código **7**
3.1 Sistema de adquisición de datos 7
3.2 Sistema de recolección y envió de datos 14
3.3 Sistema de gestión de datos 14
3.3.1 Backend (Node.js + Express) 15
3.3.1.1 Estructura del proyecto 15
3.3.1.2 Inicialización, CORS y conexión a MySQL 15
3.3.1.3 Recepción de medidas desde TTN (webhook) 16
3.3.1.4 Utilidades de dispositivo (DEV_EUI) 17
3.3.1.5 Gestión de tablas en base de datos (granjas, usuarios, sensores) 18
3.3.1.6 Consulta para la app (ultima medición) 19
3.3.1.7 Envio de downlink a TTN 20
3.3.1.8 Actualizacion de metadatos del sensor 21
3.3.1.9 Arranque del servidor en Railway 22
3.3.2 Aplicación móvil (Medidas_widget.dart) 22
3.3.2.1 Dependencias e *imports* 23
3.3.2.2 Constantes de configuración 23
3.3.2.3 Ciclo de vida y consulta periódica de datos 23
3.3.2.4 Acceso al servidor: temperatura, humedad e ITH . . . 24
3.3.2.5 Notificaciones locales por ITH alto 25
3.3.2.6 Envío de *downlinks* a TTN 26
3.3.2.7 Generación de informe PDF desde la app 26
3.3.2.8 Visualización reactiva en la app móvil 27
3.3.2.9 Estado del sistema: “última actualización” 29
3.3.2.10 Acciones del usuario 29
3.3.2.11 Configuración de alertas (UI) 30
3.3.2.12 Menú lateral (*drawer*) 30
3.3.2.13 Modelo asociado (`MedidasModel`) 31
3.4 Aplicación móvil (Dispositivos_widget.dart) 31

3.4.1 Funciones principales 31
3.4.2 Codigo (Dart) 32
3.4.3 Aplicación móvil: pantalla de autenticación (`LoginWidget`) . . . 42
3.4.3.1 Imports principales 42
3.4.3.2 Estructura del widget y estado 43
3.4.3.3 Interfaz: formulario de acceso 43
3.4.3.4 Acciones: recuperación, inicio de sesión y registro . . . 44
3.4.3.5 Composición visual 45
3.4.3.6 Modelo asociado (`LoginModel`) 46
3.4.4 Aplicación móvil: pantalla de registro (`RegistroWidget`) 47
3.4.4.1 Imports principales 47
3.4.4.2 Estado y controladores de formulario 47
3.4.4.3 Estado y controladores de formulario 48
3.4.4.4 Ayudas de acceso a datos (Railway) 49
3.4.4.5 Campos del formulario 53
3.4.4.6 Fecha de nacimiento 54
3.4.4.7 Datos de granja y sensor 54
3.4.4.8 Acción principal: guardar registro 55
3.4.4.9 Modelo asociado (`RegistroModel`) 60
3.4.5 Aplicación móvil: recuperación de contraseña (`ForgotpasswordWidget`) 62
3.4.5.1 Imports principales 63
3.4.5.2 Estructura y estado 63
3.4.5.3 Interfaz: campo de email 63
3.4.5.4 Acción: enviar correo de restablecimiento 64
3.4.5.5 Composición visual 64
3.4.5.6 Modelo asociado (`ForgotpasswordModel`) 65
3.4.6 Aplicación móvil: administración de perfil (`AdministrarPerfilWidget`) 66
3.4.6.1 Imports principales 66
3.4.6.2 Estructura y estado 66
3.4.6.3 Carga y persistencia de la foto 67
3.4.6.4 Seleccion de avatar y UI 68
3.4.7 Aplicación móvil: Archivo principal (`main.dart`) 70
3.4.7.1 Notificaciones locales 70
3.4.7.2 Funcion `main`: orden de arranque 70
3.4.7.3 Raiz de la app: `MyApp` 71
3.5 Sistema de almacenamiento de datos 79
3.5.1 Backend (Node.js/Express) conectado a MySQL en Railway . . 80
3.5.2 Esquema de base de datos 84

CAPÍTULO 1

Introducción

En este documento se describe el código fuente desarrollado para el sistema de monitorización y gestión de datos del sensor ITH en explotaciones ganaderas.

El sistema integra un dispositivo Arduino MKR WAN 1310 con un sensor DHT11 para la lectura de temperatura y humedad, empleando la red LoRaWAN y la plataforma The Things Network para la transmisión de datos. Posteriormente, un backend desarrollado en Node.js recibe la información, la procesa y la almacena en una base de datos MySQL. Finalmente, una aplicación móvil desarrollada en Flutter permite al usuario consultar las mediciones, recibir alertas, generar informes y gestionar la configuración del sistema.

Este manual detalla el código y la lógica implementada en cada uno de estos componentes, facilitando así la comprensión, el mantenimiento y la posible ampliación futura del sistema.

CAPÍTULO 2

Requisitos técnicos del sistema

En este capitulo se detallan los requisitos técnicos que el sistema debe tener para poder ser instalado y ejecutado de forma correcta.

2.1. Sistema de adquisición de datos

- Módulo DHT11 para medir temperatura y humedad mediante salida digital calibrada.
- Arduino MKR WAN 1310 con radio LoRa/LoRaWAN para leer el DHT11 y enviar datos por LoRaWAN.
- Arduino IDE versión 2.3.6.
- Librería DHT (2.1.0) y librería MKRWAN (1.1.1).
- The Things Indoor Gateway conectado a Internet para entregar los paquetes a TTN.

2.2. Sistema de recolección y envío de datos

- Cuenta y aplicación en The Things Network (TTN) con dispositivo dado de alta por OTAA.
- Configuración del *Payload Formatter* para obtener temperatura, humedad e ITH.
- Gateway LoRaWAN con salida a Internet (Ethernet/Wi-Fi/4G) y consola TTN accesible.
- Webhook de TTN apuntando al dominio público del backend en Railway.

2.3. Sistema de gestión de datos

- Backend en Node.js 18+ con Express y cliente MySQL `mysql2`.

- Despliegue en Railway con dominio público y variables de entorno configuradas (`MYSQLHOST`, `MYSQLUSER`, `MYSQLPASSWORD`, `MYSQLDATABASE`, `MYSQLPORT`, `PORT`).
- Endpoint de verificación `/health` y consulta de logs desde el panel de Railway.
- Visual Studio Code versión 1.92.1 para edición y depuración del código.

2.4. Sistema de almacenamiento de datos

- MySQL gestionada en Railway para persistencia de mediciones, sensores, granjas y usuarios.

CAPÍTULO 3

Descripción del código

En este capitulo, se redacta el código usado para todo el sistema completo, el contenido y ubicación de los ficheros usados para el funcionamiento del sistema y una breve explicación de lo que hace cada fichero de código.

Todos los archivos del código para el funcionamiento y configuración del proyecto, se pueden encontrar dentro del archivo llamado "ithapp' por parte del código fuente de la aplicación móvil, para la parte del servidor o backend en "Proyecto-tfg-ithapp" y el archivo de Arduino llamado "Codigo-tfg.ino" en el siguiente enlace de repositorio:

3.1. Sistema de adquisición de datos

El fichero `codigo-tfg.ino` implementa el nodo IoT basado en **Arduino MKR WAN 1310** y el sensor **DHT11**. Su función es: leer temperatura y humedad, calcular el índice **ITH**, enviar las mediciones por **LoRaWAN (OTAA)** a **TTN** en intervalos regulares, y atender *downlinks* de texto (ASCII) para operar en **modo Automático/-Manual** con umbral y para **activar/desactivar** el actuador (ventilador) conectado al pin `D6`.

Resumen de diseño

- **Región y unión**: EU868, unión *OTAA* con `AppEUI` y `AppKey`.
- **Medida y cálculo**: lectura periódica del DHT11 (˜1–2 s) y cálculo de ITH con la fórmula usada en la memoria.
- **Envío**: *uplink* no confirmado cada `SEND_EVERY_MS` (por defecto 30 s, ajustable por normativa/duty cycle).
- **Formato de payload**: 6 bytes: `T/H/ITH` empaquetados en **x100** como *entero* (byte 0,2,4) + *centésimas* (byte 1,3,5).
- **Downlink ASCII** (FPort 1): comandos como `MODE=AUTO;TH=75`, `MODE=MANUAL`, `ACTIVATE`, `DEACTIVATE`, `ON`, `OFF`, `1`, `0`. Se normaliza a mayúsculas y sin espacios.
- **Control del ventilador**:
  - *Manual*: el estado se fuerza con `ACTIVATE`/`DEACTIVATE`.

- *Auto*: se enciende si `ITH` $\geq$ `ithOn` y se apaga si `ITH` $\leq$ `ithOff` (**histéresis** de 2 puntos por defecto, configurable con el umbral `TH`).

Código del nodo (Arduino)

El siguiente código está documentado paso a paso para facilitar su lectura y mantenimiento.

Listing 3.1: Nodo MKR WAN 1310 + DHT11 con control AUTO/MANUAL por downlink

```
#include <MKRWAN.h>
#include "DHT.h"

//


// Hardware y librerias
//


#define DHTPIN   7                  // Pin del DHT11
#define DHTTYPE  DHT11
const int PIN_FAN = 6;              // Transistor/rel  del
   ventilador (HIGH=ON)

DHT dht(DHTPIN, DHTTYPE);
LoRaModem modem;

//


// Credenciales LoRaWAN (OTAA) - EU868
//


String appEui = "XXXXXXXXXXXXXXXX"; // JoinEUI / AppEUI
String appKey = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"; // AppKey
   (32 hex)
const uint8_t UPLINK_FPORT = 1;

//


// Parametros de operacion y temporizado
//


const bool   DEFAULT_AUTO_MODE = false;   // Arranque en MANUAL
   por defecto
```

```
25 bool          autoMode = DEFAULT_AUTO_MODE;
26
27 float ithOn  = 75.0;                        // Umbral ENCENDER (
      AUTO)
28 float ithOff = 73.0;                        // Umbral APAGAR (AUTO
      ) = ithOn - 2
29 bool  fanOn  = false;                       // Estado actual del
      ventilador
30
31 const unsigned long SENSOR_EVERY_MS = 2000UL;  // Periodo de
      lectura DHT11
32 const unsigned long SEND_EVERY_MS   = 30000UL; // Periodo de
      uplink (ajustar a duty-cycle)
33
34 unsigned long lastSensorRead = 0;
35 unsigned long lastSend       = 0;
36
37 //  ltima  medida  v lida
38 bool  haveValid = false;
39 float lastT = NAN, lastH = NAN, lastITH = NAN;
40
41
42 // Actualiza el actuador y el estado interno
43 void setFan(bool on) {
44   fanOn = on;
45   digitalWrite(PIN_FAN, on ? HIGH : LOW);
46   if (Serial) Serial.println(on ? "Ventilador: ON" : "Ventilador
       : OFF");
47 }
48
49 // F rmula ITH usada en la memoria del TFG
50 float computeITH(float tC, float rh) {
51   return (1.8 * tC + 32) - (0.55 - 0.55 * rh / 100.0) * (1.8 *
       tC - 26);
52 }
53
54 // Empaqueta un float con 2 decimales (x100) en 2 bytes: entero
      y centesimas
55 void toBytes100(float v, byte outAB[2]) {
56   if (v < 0) v = 0;
57   if (v > 255.99) v = 255.99;
58   int ent  = (int)v;
59   int cent = (int)round((v - ent) * 100.0);
60   if (cent == 100) { ent += 1; cent = 0; }
61   outAB[0] = (byte)ent;
62   outAB[1] = (byte)cent;
63 }
64
65 // Lee posibles downlinks durante RX1/RX2 (ventana indicativa)
66 String readDownlink(uint16_t windowMs = 6000) {
```

```
67    String cmd = "";
68    unsigned long t0 = millis();
69    while (millis() - t0 < windowMs) {
70      if (modem.available() > 0) {
71        while (modem.available() > 0) {
72          cmd += (char)modem.read();
73        }
74        break;
75      }
76      delay(40);
77    }
78    cmd.trim();
79    return cmd;
80  }
81
82  // Normaliza texto: quita espacios/saltos y pasa a MAYUSCULAS
83  String sanitize(String s) {
84    s.trim();
85    String r = "";
86    for (size_t i = 0; i < s.length(); ++i) {
87      char c = s.charAt(i);
88      if (c == ' ' || c == '\r' || c == '\n' || c == '\t')
          continue;
89      r += (char)toupper(c);
90    }
91    return r;
92  }
93
94  // Interpreta y ejecuta el comando recibido por downlink
95  void handleCommand(String raw) {
96    if (raw.length() == 0) return;
97
98    String s = sanitize(raw);
99    if (Serial) { Serial.print("Downlink recibido: "); Serial.
        println(s); }
100
101   // Atajos ON/OFF (en MANUAL)
102   if (s == "ON" || s == "ACTIVATE" || s == "1") {
103     autoMode = false;
104     setFan(true);
105     return;
106   }
107   if (s == "OFF" || s == "DEACTIVATE" || s == "0") {
108     autoMode = false;
109     setFan(false);
110     return;
111   }
112
113   // Soporta "MODE=AUTO;TH=75"
114   int start = 0;
```

```
115   while (start < (int)s.length()) {
116     int sep = s.indexOf(';', start);
117     String tok = (sep == -1) ? s.substring(start) : s.substring(
          start, sep);
118     start = (sep == -1) ? s.length() : sep + 1;
119
120     if (tok.startsWith("MODE=")) {
121       if (tok.endsWith("AUTO")) {
122         autoMode = true;
123         if (Serial) Serial.println("Modo: AUTO");
124       } else if (tok.endsWith("MANUAL")) {
125         autoMode = false;
126         if (Serial) Serial.println("Modo: MANUAL");
127       }
128     } else if (tok.startsWith("TH=") || tok.startsWith("UMBRAL="
          )) {
129       float th = tok.substring(tok.indexOf('=') + 1).toFloat();
130       if (th > 0.0) {
131         ithOn  = th;
132         ithOff = th - 2.0; // hist resis simple
133         if (Serial) {
134           Serial.print("Umbral AUTO actualizado: ON=");
135           Serial.print(ithOn, 1);
136           Serial.print("  OFF=");
137           Serial.println(ithOff, 1);
138         }
139       }
140     }
141   }
142 }
143
144 // Aplica control automatico del ventilador.
145 void applyAutoControl(float ith) {
146   if (!autoMode) return;
147   if (!fanOn && ith >= ithOn) {
148     setFan(true);
149     if (Serial) Serial.println("[AUTO] ITH >= ON -> Ventilador
          ON");
150   } else if (fanOn && ith <= ithOff) {
151     setFan(false);
152     if (Serial) Serial.println("[AUTO] ITH <= OFF -> Ventilador
          OFF");
153   }
154 }
155
156 // Setup y bucle principal
157
158 void setup() {
159   Serial.begin(115200);
160   unsigned long t0 = millis();
```

```
161   while (!Serial && (millis() - t0) < 2000) { /* espera m x 2s */ }
162
163   pinMode(PIN_FAN, OUTPUT);
164   setFan(false);
165
166   dht.begin();
167
168   if (!modem.begin(EU868)) {
169     if (Serial) Serial.println("ERROR: modem.begin(EU868)");
170     delay(2000);
171   }
172   modem.setADR(true);
173   modem.setPort(UPLINK_FPORT);
174
175   if (Serial) Serial.println("Uniendo a TTN (OTAA)...");
176   bool joined = false;
177   for (int i = 0; i < 5 && !joined; i++) {
178     if (modem.joinOTAA(appEui, appKey)) {
179       joined = true;
180     } else {
181       if (Serial) Serial.println("Join fallido. Reintento en 10 s...");
182       delay(10000);
183     }
184   }
185   if (Serial) Serial.println(joined ? "Conectado a TTN" : "No se pudo unir (seguir  intentando enviar)");
186 }
187
188 void loop() {
189   unsigned long now = millis();
190
191   // 1) Lectura periodica del sensor (respetando l mites del DHT11)
192   if (now - lastSensorRead >= SENSOR_EVERY_MS) {
193     lastSensorRead = now;
194
195     float t = dht.readTemperature();  // [ C ]
196     float h = dht.readHumidity();     // [%]
197     if (!isnan(t) && !isnan(h)) {
198       haveValid = true;
199       lastT  = t;
200       lastH  = h;
201       lastITH = computeITH(t, h);
202
203       // Control automatico (solo con medida valida)
204       applyAutoControl(lastITH);
205
206       if (Serial) {
```

```
207         Serial.print("Medida: T="); Serial.print(lastT, 1);
208         Serial.print("  H="); Serial.print(lastH, 0); Serial.
               print("%");
209         Serial.print("  ITH="); Serial.println(lastITH, 1);
210       }
211     } else {
212       if (Serial) Serial.println("Lectura DHT11 invalida.
             Mantengo ultima medida/estado.");
213     }
214   }
215
216   // 2) Envio periodico de uplink con ultima medida conocida
217   if (now - lastSend >= SEND_EVERY_MS) {
218     lastSend = now;
219
220     byte payload[6];
221     float tEnc = haveValid ? lastT  : 0.0;
222     float hEnc = haveValid ? lastH  : 0.0;
223     float iEnc = haveValid ? lastITH: 0.0;
224
225     toBytes100(tEnc, &payload[0]); // T entero y centesimas
226     toBytes100(hEnc, &payload[2]); // H entero y centesimas
227     toBytes100(iEnc, &payload[4]); // ITH entero y centesimas
228
229     modem.beginPacket();
230     modem.write(payload, sizeof(payload));
231     int sent = modem.endPacket(false); // no confirmado
232     if (Serial) {
233       Serial.print("Uplink ");
234       Serial.println(sent > 0 ? "OK" : "FALLO");
235       Serial.print("T=");   Serial.print(tEnc, 1);
236       Serial.print("  H="); Serial.print(hEnc, 0); Serial.print(
             "%");
237       Serial.print("  ITH="); Serial.print(iEnc, 1);
238       Serial.print("  MODO="); Serial.print(autoMode ? "AUTO" :
             "MANUAL");
239       Serial.print("  FAN=");  Serial.println(fanOn ? "ON" : "
             OFF");
240     }
241
242     // 3) Lectura de downlink en ventanas RX1/RX2
243     String cmd = readDownlink(6000);
244     if (cmd.length() > 0) handleCommand(cmd);
245   }
246
247   delay(10000); // Respiro de CPU
248 }
```

3.2. Sistema de recolección y envió de datos

En este subsistema sólo interviene el payload formatter de The Things Network. Se encarga de transformar el payload en bruto (seis bytes enviados por el Arduino: dos para temperatura, dos para humedad y dos para el ITH) en un objeto `JSON` fácilmente entendible por el servidor.

La idea es muy simple: cada par de bytes se interpreta como *parte entera* y *centésimas*, y el resultado se expone con las claves `temperatura`, `humedad` e `ith`.

A continuación se muestra el decodificador usado en TTN. Los comentarios explican cada paso de forma directa.

Listing 3.2: Decodificador de uplink en TTN

```
function decodeUplink(input) {
  // Bytes esperados: [T_int, T_cent, H_int, H_cent, ITH_int,
     ITH_cent]
  // Combinar entero + centesimas => valor con dos decimales
  let t   = input.bytes[0] + input.bytes[1] / 100;
  let h   = input.bytes[2] + input.bytes[3] / 100;
  let ith = input.bytes[4] + input.bytes[5] / 100;

  // Devolver JSON con nombres claros de campo
  return {
    data: {
      temperatura: t,
      humedad: h,
      ith: ith
    }
  };
}
```

Con este formateador, cada *uplink* que llega a TTN se publica ya como `JSON` estructurado, listo para que el sistema de backend lo reciba por webhook sin tener que interpretar bytes en la aplicación servidor.

3.3. Sistema de gestión de datos

Este sistema recibe los datos desde The Things Network (TTN), los valida y los almacena en la base de datos para que la aplicación móvil pueda consultarlos. El backend está implementado en **Node.js/Express** y se despliega en **Railway**, con una API[1] y una base de datos **MySQL** gestionadas mediante variables de entorno.

[1] Application Programming Interface

3.3.1. Backend (Node.js + Express)

3.3.1.1. Estructura del proyecto

La estructura del servidor es:

- `package.json` y `package-lock.json`: dependencias y scripts.
- `server.js`: punto de entrada del servidor (endpoints REST, lógica y acceso a datos).
- `node_modules/`: dependencias instaladas.

3.3.1.2. Inicialización, CORS y conexión a MySQL

Se utilizan `express`, `cors` y `mysql2/promise`. Railway inyecta `PORT` y las credenciales de MySQL como variables de entorno. Se emplea un *pool* de conexiones para eficiencia y concurrencia, y un endpoint `/health` para verificar liveness.

Listing 3.3: Inicializacion del servidor, CORS y conexion a MySQL

```
const express = require("express");
const mysql = require("mysql2/promise");
const cors = require("cors");

const app = express();

// 1) Middleware
app.use(express.json());
app.use(cors({ origin: "*" }));

// 2) Puerto (Railway coloca PORT)
const port = process.env.PORT || 3000;

// 3) Config MySQL (Railway inyecta estas vars al crear la DB)
const pool = mysql.createPool({
  host: process.env.MYSQLHOST || "localhost",
  port: process.env.MYSQLPORT ? Number(process.env.MYSQLPORT) :
     3306,
  user: process.env.MYSQLUSER || "root",
  password: process.env.MYSQLPASSWORD || "",
  database: process.env.MYSQLDATABASE || "ganaderapp",
  waitForConnections: true,
  connectionLimit: 10,
});

// Cache en memoria del ultimo DEV_EUI conocido
let lastDevEui = null;
```

```
27
28 // 4) Salud (verifica liveness y conectividad a MySQL)
29 app.get("/health", async (_req, res) => {
30   try {
31     const conn = await pool.getConnection();
32     await conn.ping();
33     conn.release();
34     res.json({ ok: true });
35   } catch (e) {
36     console.error(e);
37     res.status(500).json({ ok: false, error: "DB ping failed" })
         ;
38   }
39 });
```

El endpoint `/health` permite a Railway (o a cualquier monitor) comprobar que la API está viva y que existe conectividad hacia la base de datos.

3.3.1.3. Recepción de medidas desde TTN (webhook)

El endpoint `/webhook` recibe el *uplink* de TTN, cachea el `dev_eui` para usos posteriores y valida el `decoded_payload`. Si todo es correcto, inserta la medición en `mediciones`.

Listing 3.4: Webhook de TTN: validacion, cache de DEV_EUI e insercion

```
1  app.post("/webhook", async (req, res) => {
2    try {
3      // Cachea dev_eui en memoria (se pierde si el proceso se
         reinicia)
4      const devEui = req.body?.end_device_ids?.dev_eui
5                  || req.body?.end_device_ids?.device_id
6                  || null;
7      if (devEui) lastDevEui = devEui;
8
9      const uplink = req.body.uplink_message;
10     if (!uplink?.decoded_payload) {
11       return res.status(400).send("Payload invalido");
12     }
13
14     const { temperatura, humedad, ith } = uplink.decoded_payload
         ;
15     await pool.query(
16       "INSERT INTO mediciones (temperatura, humedad, ith) VALUES
           (?, ?, ?)",
17       [temperatura, humedad, ith]
18     );
19
```

```
    console.log("Datos recibidos:", { temperatura, humedad, ith,
        devEui });
    res.send("OK");
  } catch (err) {
    console.error("Error insertando mediciones:", err);
    res.status(500).send("Error en base de datos");
  }
});
```

3.3.1.4. Utilidades de dispositivo (DEV_EUI)

Se exponen dos rutas para recuperar un identificador de dispositivo válido. La primera devuelve el último `DEV_EUI` cacheado por el webhook; la segunda consulta la tabla `sensores` con dos estrategias: por `updated_at` y por orden de `id_sensor`.

Listing 3.5: DEV_EUI: cache en memoria y ultima referencia en DB

```
app.get("/dev-eui-ultimo", (_req, res) => {
  if (!lastDevEui) return res.status(404).json({ error: "
     sin_dev_eui" });
  res.json({ dev_eui: lastDevEui });
});

app.get('/api/dev-eui-ultimo', async (_req, res) => {
  const q1 = `
    SELECT dev_eui
    FROM sensores
    WHERE dev_eui IS NOT NULL AND dev_eui <> ''
    ORDER BY updated_at DESC, id_sensor DESC
    LIMIT 1`;
  const q2 = `
    SELECT dev_eui
    FROM sensores
    WHERE dev_eui IS NOT NULL AND dev_eui <> ''
    ORDER BY id_sensor DESC
    LIMIT 1`;
  try {
    const [rows] = await pool.query(q1);
    if (!rows.length) return res.status(404).json({ error: '
       sin_dev_eui' });
    res.json({ dev_eui: rows[0].dev_eui });
  } catch (e) {
    // Si updated_at no existe, probamos sin ella
    if (e.code === 'ER_BAD_FIELD_ERROR') {
      const [rows] = await pool.query(q2);
      if (!rows.length) return res.status(404).json({ error: '
         sin_dev_eui' });
      return res.json({ dev_eui: rows[0].dev_eui });
    }
```

```
    console.error(e);
    res.status(500).json({ error: 'db_error' });
  }
});
```

3.3.1.5. Gestión de tablas en base de datos (granjas, usuarios, sensores)

Se implementan endpoints de alta con validación mínima y consultas parametrizadas para evitar inyecciones SQL.

Listing 3.6: Alta de granja

```
app.post("/granjas", async (req, res) => {
  try {
    const { nombre_granja, direccion } = req.body;
    if (!nombre_granja || !direccion) {
      return res.status(400).send("Faltan datos de la granja");
    }

    const [result] = await pool.query(
      "INSERT INTO granjas (nombre_granja, direccion) VALUES (?, ?)",
      [nombre_granja, direccion]
    );

    res.status(201).json({ id_granja: result.insertId, mensaje: "Granja guardada" });
  } catch (err) {
    console.error("Error insertando granja:", err);
    res.status(500).send("Error en base de datos");
  }
});
```

Listing 3.7: Alta de usuario

```
app.post("/usuarios", async (req, res) => {
  try {
    const { nombre, apellidos, email, fecha_nacimiento, password, id_granja } = req.body;
    if (!nombre || !email || !fecha_nacimiento || !password || !id_granja) {
      return res.status(400).send("Faltan datos del usuario");
    }

    const [result] = await pool.query(
      `INSERT INTO usuarios (nombre, apellidos, email, fecha_nacimiento, password, id_granja)
       VALUES (?, ?, ?, ?, ?, ?)`,
      [nombre, apellidos || "", email, fecha_nacimiento, password, id_granja]
```

```
    );

    res.status(201).json({ id_usuario: result.insertId, mensaje:
        "Usuario guardado" });
  } catch (err) {
    console.error("Error insertando usuario:", err);
    res.status(500).send("Error en base de datos");
  }
});
```

Listing 3.8: Alta de sensor

```
app.post("/sensores", async (req, res) => {
  try {
    const { nombre_sensor, id_granja } = req.body;
    if (!nombre_sensor || !id_granja) {
      return res.status(400).send("Faltan datos del sensor");
    }

    const [result] = await pool.query(
      "INSERT INTO sensores (nombre_sensor, id_granja) VALUES
         (?, ?)",
      [nombre_sensor, id_granja]
    );

    res.status(201).json({ id_sensor: result.insertId, mensaje:
       "Sensor guardado" });
  } catch (err) {
    console.error("Error insertando sensor:", err);
    res.status(500).send("Error en base de datos");
  }
});
```

3.3.1.6. Consulta para la app (ultima medición)

Se sirve la ultima medición registrada, pensada para la pantalla principal de la app Flutter.

Listing 3.9: Ultima medicion

```
app.get("/mediciones", async (_req, res) => {
  try {
    const [rows] = await pool.query("SELECT * FROM mediciones
       ORDER BY id DESC LIMIT 1");
    if (rows.length > 0) return res.json(rows[0]);
    res.status(404).send("No hay datos disponibles");
  } catch (err) {
    console.error("Error al consultar mediciones:", err);
    res.status(500).send("Error en base de datos");
  }
```

```
});
```

3.3.1.7. Envio de downlink a TTN

Para controlar el nodo desde la app se implementa un manejador que: (a) valida `dev_eui` y busca `app_id`/`device_id` en la tabla `sensores`, (b) codifica el comando ASCII a base64 y (c) llama al endpoint `down/push` de TTN con `f_port=1`.

Listing 3.10: Handler de downlink a TTN y registro de endpoints

```
const idRe = /^[a-z0-9](?:-?[a-z0-9]){2,}$/;

async function pushDownlinkHandler(req, res) {
  try {
    let { dev_eui, cmd } = req.body || {};
    if (!dev_eui || !cmd) return res.status(400).json({ error: '
       faltan_campos' });

    dev_eui = String(dev_eui).trim().toUpperCase();

    // Busca app_id y device_id asociados a dev_eui
    const [rows] = await pool.query(
      `SELECT app_id, device_id
         FROM sensores
        WHERE UPPER(TRIM(dev_eui)) = ?
        LIMIT 1`, [dev_eui]
    );
    if (!rows.length) return res.status(404).json({ error: '
       sensor_no_encontrado' });

    const { app_id, device_id } = rows[0];
    if (!app_id || !device_id) {
      return res.status(409).json({ error: '
         falta_app_o_device_id' });
    }
    if (!idRe.test(app_id))    return res.status(400).json({
       error: 'app_id_invalido',    valor: app_id });
    if (!idRe.test(device_id)) return res.status(400).json({
       error: 'device_id_invalido', valor: device_id });

    const TTN_API_KEY = process.env.TTN_API_KEY;
    if (!TTN_API_KEY) return res.status(500).json({ error: '
       ttn_api_key_no_configurada' });

    console.log('[Downlink] app_id=%s device_id=%s dev_eui=%s
       cmd=%s',
      app_id, device_id, dev_eui, cmd);

```

```
32      const url = `https://eu1.cloud.thethings.network/api/v3/as/
           applications/${app_id}/devices/${device_id}/down/push`;
33      const frmPayloadB64 = Buffer.from(String(cmd), 'utf8').
           toString('base64');
34
35      const body = { downlinks: [{ f_port: 1, frm_payload:
           frmPayloadB64, priority: 'NORMAL' }] };
36
37      const r = await fetch(url, {
38        method: 'POST',
39        headers: {
40          'Authorization': `Bearer ${TTN_API_KEY}`,
41          'Content-Type': 'application/json'
42        },
43        body: JSON.stringify(body)
44      });
45
46      const text = await r.text();
47      if (!r.ok) {
48        console.error('TTN downlink error', r.status, text);
49        return res.status(502).json({ error: 'ttn_error', status:
             r.status, detail: text });
50      }
51      res.json({ ok: true });
52    } catch (e) {
53      console.error(e);
54      res.status(500).json({ error: 'downlink_error' });
55    }
56  }
57
58  app.post('/api/ttn/downlink', pushDownlinkHandler);
59  app.post('/api/downlink',     pushDownlinkHandler);
```

3.3.1.8. Actualizacion de metadatos del sensor

Endpoint para actualizar metadatos del sensor (modo, umbral ITH, ubicacion, etc.) localizado por `dev_eui`. Solo se actualizan los campos provistos; el resto se mantiene sin cambios.

Listing 3.11: PUT /api/sensores/actualizar

```
1  app.put('/api/sensores/actualizar', async (req, res) => {
2    try {
3      let { dev_eui, modelo, area, zona, sala, modo, umbral_ith }
          = req.body;
4      if (!dev_eui) return res.status(400).json({error:'
          dev_eui_requerido'});
5      dev_eui = String(dev_eui).trim().toUpperCase();
6
```

```
    // Localiza por dev_eui (normalizando)
    const [rows] = await pool.query(
      `SELECT id_sensor, UPPER(TRIM(dev_eui)) AS eui FROM sensores WHERE dev_eui IS NOT NULL`
    );
    const row = rows.find(r => r.eui === dev_eui);
    if (!row) return res.status(404).json({error:'sensor_no_encontrado'});

    // Actualiza solo lo que venga con valor
    await pool.query(
      `UPDATE sensores SET
          modelo     = COALESCE(NULLIF(?, ''), modelo),
          area       = COALESCE(NULLIF(?, ''), area),
          zona       = COALESCE(?, zona),
          sala       = COALESCE(NULLIF(?, ''), sala),
          modo       = COALESCE(NULLIF(?, ''), modo),
          umbral_ith = COALESCE(?, umbral_ith),
          updated_at = NOW()
        WHERE id_sensor = ?`,
      [modelo, area, zona, sala, modo, umbral_ith, row.id_sensor]
    );

    res.json({ ok:true });
  } catch (e) {
    console.error(e);
    res.status(500).json({error:'db_error'});
  }
});
```

3.3.1.9. Arranque del servidor en Railway

El servidor escucha en el puerto indicado por Railway (o 3000 en local). El mensaje de arranque evita tildes para compatibilidad con listados.

Listing 3.12: Arranque del servidor

```
app.listen(port, () => {
  console.log(`API escuchando en puerto ${port}`);
});
```

3.3.2. Aplicación móvil (Medidas_widget.dart)

Esta sección describe cómo la app Flutter se conecta con el backend desplegado en Railway, muestra las medidas casi en tiempo real, lanza alertas locales si el ITH supera un umbral y permite enviar *downlinks* a TTN.

El código se centra en el widget `MedidasWidget` y funciones asociadas, adaptadas al dominio público de Railway.

3.3.2.1. Dependencias e *imports*

La app usa `http` para llamadas REST, `flutter_local_notifications` para alertas locales y `printing/pdf` para generar informes.

```
import 'package:flutter_local_notifications/
   flutter_local_notifications.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';
import 'package:intl/intl.dart';
import 'package:pdf/widgets.dart' as pw;
import 'package:pdf/pdf.dart';
import 'package:printing/printing.dart';
import 'package:path_provider/path_provider.dart';
import 'package:open_filex/open_filex.dart';
```

3.3.2.2. Constantes de configuración

Se apunta al dominio público del backend en Railway y se definen los datos de TTN.

```
static const String kApiBase =
  'https://tfg-proyecto-ithapp-production.up.railway.app';

static const String TTN_APPLICATION_ID = 'tfg-ganadera';
static const String TTN_DEVICE_ID      = 'a8610a3436375f17';
static const String TTN_API_KEY        = '<API_KEY>';
static const String TTN_BASE_URL       = 'https://eu1.cloud.
   thethings.network';
```

3.3.2.3. Ciclo de vida y consulta periódica de datos

Cada 10 segundos se consulta `/mediciones`. Se conserva una marca temporal para la etiqueta "última actualización".

```
late final Stream<double?> temperatura$;
late final Stream<int?>     humedad$;
late final Stream<double?>  ith$;
late final Stream<DateTime> ultimaFecha$;
late final Stream<int>      _ticker$;

DateTime? ultimaFechaMedicion;

@override
```

```
10 void initState() {
11   super.initState();
12
13   _ticker$ = Stream.periodic(const Duration(seconds: 1), (i) =>
       i);
14
15   temperatura$ = Stream.periodic(
16     const Duration(seconds: 10), (_) => fetchTemperatura(),
17   ).asyncMap((fut) => fut);
18
19   humedad$ = Stream.periodic(
20     const Duration(seconds: 10), (_) => fetchHumedad(),
21   ).asyncMap((fut) => fut);
22
23   ith$ = Stream.periodic(
24     const Duration(seconds: 10), (_) => fetchITH(),
25   ).asyncMap((fut) => fut);
26
27   ultimaFecha$ = Stream.periodic(
28     const Duration(seconds: 10), (_) => DateTime.now(),
29   );
30 }
```

3.3.2.4. Acceso al servidor: temperatura, humedad e ITH

Las funciones consultan `/mediciones` y devuelven valores ya decodificados. `fetchITH()` lanza notificación local si el ITH ≥ 75.

```
1  Future<Map<String, dynamic>?> _getUltimaMedicion() async {
2    try {
3      final res = await http.get(Uri.parse('$kApiBase/mediciones')
         ,
4                                  headers: {'Accept': 'application/
                                     json'});
5      if (res.statusCode == 200) {
6        final data = jsonDecode(res.body);
7        if (data is Map<String, dynamic>) {
8          final fechaRaw = data['fecha'];
9          if (fechaRaw is String) {
10           try {
11             final parsed = DateTime.parse(fechaRaw).toLocal();
12             if (mounted) setState(() => ultimaFechaMedicion =
                  parsed);
13           } catch (_) {
14             if (mounted) setState(() => ultimaFechaMedicion =
                  DateTime.now());
15           }
16         }
17         return data;
```

```
      }
    }
  } catch (_) {}
  return null;
}

Future<double?> fetchTemperatura() async {
  final data = await _getUltimaMedicion();
  final v = data?['temperatura'];
  return v is num ? v.toDouble() : null;
}

Future<int?> fetchHumedad() async {
  final data = await _getUltimaMedicion();
  final v = data?['humedad'];
  return v is num ? v.round() : null;
}

Future<double?> fetchITH() async {
  final data = await _getUltimaMedicion();
  final v = data?['ith'];
  if (v is num) {
    final ith = v.toDouble();
    if (ith >= 75) {
      await mostrarNotificacionITHAlto(ith);
    }
    return ith;
  }
  return null;
}
```

3.3.2.5. Notificaciones locales por ITH alto

```
Future<void> mostrarNotificacionITHAlto(double ith) async {
  const android = AndroidNotificationDetails(
    'ith_channel_id', 'ITH Alertas',
    channelDescription: 'Notificaciones por ITH alto',
    importance: Importance.max,
    priority: Priority.high,
    playSound: true,
    enableVibration: true,
  );
  await flutterLocalNotificationsPlugin.show(
    0,
    'Estr s t rmico alto',
    'El ITH es $ith. Toma medidas preventivas.',
    const NotificationDetails(android: android),
  );
}
```

3.3.2.6. Envío de *downlinks* a TTN

La app puede enviar comandos al dispositivo. En producción, se recomienda realizar el *downlink* a través del backend para no exponer claves.

```
Future<bool> enviarDownlinkTTN({
  required String payload,
  int puerto = 80,
  bool confirmado = false,
}) async {
  try {
    final url = Uri.parse(
      '$TTN_BASE_URL/api/v3/as/applications/$TTN_APPLICATION_ID/
         devices/$TTN_DEVICE_ID/down/push'
    );
    final headers = {
      'Authorization': 'Bearer $TTN_API_KEY',
      'Content-Type': 'application/json',
      'User-Agent': 'Flutter App',
    };
    final payloadBase64 = base64Encode(utf8.encode(payload));
    final body = jsonEncode({
      "downlinks": [
        {"f_port": puerto, "frm_payload": payloadBase64,
         "confirmed": confirmado, "priority": "NORMAL"}
      ]
    });
    final resp = await http.post(url, headers: headers, body:
       body);
    return resp.statusCode == 200;
  } catch (_) { return false; }
}
```

3.3.2.7. Generación de informe PDF desde la app

La app puede generar un informe puntual con las últimas medidas recogidas.

```

Future<void> generarPdf({
  required double temperatura,
  required int    humedad,
  required double ith,
  required DateTime fecha,
}) async {
  final pdf = pw.Document();
  pdf.addPage(
    pw.Page(
      pageFormat: PdfPageFormat.a4,
      margin: const pw.EdgeInsets.all(40),
```

```
      build: (ctx) => pw.Column(
        crossAxisAlignment: pw.CrossAxisAlignment.start,
        children: [
          pw.Text('Informe de condiciones ambientales',
                  style: pw.TextStyle(fontSize: 24, fontWeight:
                     pw.FontWeight.bold)),
          pw.SizedBox(height: 12),
          pw.Text('Generado: ${DateFormat('dd/MM/yyyy     HH:mm'
             ).format(fecha)}'),
          pw.Divider(),
          pw.Table(children: [
            _dataRow('Temperatura', '${temperatura.
               toStringAsFixed(1)}  C '),
            _dataRow('Humedad',     '$humedad %'),
            _dataRow(' ndice  TH',   ith.toStringAsFixed(1)),
          ]),
        ],
      ),
    ),
  );
  final bytes = await pdf.save();
  final dir   = await getTemporaryDirectory();
  final file  = File('${dir.path}/informe_ith_${DateTime.now().
     millisecondsSinceEpoch}.pdf');
  await file.writeAsBytes(bytes);
  await OpenFilex.open(file.path);
}

pw.TableRow _dataRow(String k, String v) => pw.TableRow(children
   : [
  pw.Padding(padding: const pw.EdgeInsets.symmetric(vertical: 6)
     ,
             child: pw.Text(k, style: pw.TextStyle(fontWeight:
                pw.FontWeight.bold))),
  pw.Padding(padding: const pw.EdgeInsets.symmetric(vertical: 6)
     ,
             child: pw.Text(v)),
]);
```

3.3.2.8. Visualización reactiva en la app móvil

La UI principal muestra las tres magnitudes. Para ITH se colorea el valor según el rango.

```
Row(
  mainAxisAlignment: MainAxisAlignment.spaceAround,
  children: [
    // Temperatura
    Column(children: [
```

```
6         const Icon(Icons.thermostat, color: Colors.white, size:
              40),
7         StreamBuilder<double?>(
8           stream: temperatura$,
9           builder: (context, s) {
10            if (!s.hasData) return const CircularProgressIndicator
                  (color: Colors.white);
11            if (s.hasError)  return const Text('N/A', style:
                  TextStyle(color: Colors.white, fontSize: 32));
12            return Text('${s.data!.toStringAsFixed(1)} C ',
13              style: Theme.of(context).textTheme.headlineMedium?.
                    copyWith(color: Colors.white, fontWeight:
                    FontWeight.bold));
14          },
15        ),
16        const Text('Temperatura', style: TextStyle(color: Colors.
              white)),
17      ]),
18
19      // Humedad
20      Column(children: [
21        const Icon(Icons.water_drop, color: Colors.white, size:
              40),
22        StreamBuilder<int?>(/* ... patron analogo ... */),
23        const Text('Humedad', style: TextStyle(color: Colors.white
              )),
24      ]),
25
26      // ITH con color de severidad
27      Column(children: [
28        const Icon(Icons.speed, color: Colors.white, size: 40),
29        StreamBuilder<double?>(stream: ith$, builder: (context, s)
              {
30          if (!s.hasData) return const CircularProgressIndicator(
                color: Colors.white);
31          final v = s.data!;
32          final Color ithColor = (v <= 70) ? Colors.green
33                               : (v <= 75) ? Colors.amberAccent
34                                           : Colors.redAccent;
35          return Text(v.toStringAsFixed(1),
36            style: Theme.of(context).textTheme.headlineMedium?.
                  copyWith(color: ithColor, fontWeight: FontWeight.
                  bold));
37        }),
38        const Text('ITH', style: TextStyle(color: Colors.white)),
39      ]),
40    ],
41  )
```

3.3.2.9. Estado del sistema: "última actualización"

Se usa un *ticker* de 1 s para mostrar el tiempo transcurrido desde la última medición recibida.

```
StreamBuilder<int>(
  stream: _ticker$,
  builder: (context, _) {
    if (ultimaFechaMedicion == null) return const Text('    ');
    final diff = DateTime.now().difference(ultimaFechaMedicion!);
    final texto = (diff.inSeconds < 60) ? 'hace ${diff.inSeconds}s'
                : (diff.inMinutes < 60) ? 'hace ${diff.inMinutes} min'
                : (diff.inHours   < 24) ? 'hace ${diff.inHours} h'
                                        : 'hace ${diff.inDays} d';
    return Text(texto);
  },
)
```

3.3.2.10. Acciones del usuario

Incluye tres botones: generar informe y activar/desactivar dispositivo (vía *downlink*).

```
Row(children: [
  Expanded(child: ElevatedButton(
    onPressed: () async {
      final t = await fetchTemperatura() ?? 0;
      final h = await fetchHumedad() ?? 0;
      final i = await fetchITH() ?? 0;
      await generarInformePdf(temperatura: t, humedad: h, ith: i, fecha: DateTime.now());
    },
    child: const Text('Generar Informe'),
  )),
  const SizedBox(width: 12),
  Expanded(child: ElevatedButton(
    onPressed: () async {
      await enviarDownlinkTTN(payload: 'ACTIVATE', puerto: 80, confirmado: false);
      ScaffoldMessenger.of(context).showSnackBar(const SnackBar(content: Text('Dispositivo activado')));
    },
    child: const Text('Activar Dispositivo'),
  )),
```

```
]),
const SizedBox(height: 12),
Center(child: SizedBox(
  width: MediaQuery.of(context).size.width * 0.6,
  child: ElevatedButton(
    onPressed: () async {
      await enviarDownlinkTTN(payload: 'DEACTIVATE', puerto: 80,
          confirmado: false);
      ScaffoldMessenger.of(context).showSnackBar(const SnackBar(
         content: Text('Dispositivo desactivado')));
    },
    child: const Text('Desactivar Dispositivo'),
  ),
)),
```

3.3.2.11. Configuración de alertas (UI)

Tres interruptores (*Switch*) controlan si las alertas de temperatura, humedad e ITH están activas, y se muestran los umbrales configurados en UI.

```
Switch(
  value: _model.switchValue3!, // Alertas de ITH
  onChanged: (v) => safeSetState(() => _model.switchValue3 = v),
),
Text('Umbral de ITH: 75')
```

3.3.2.12. Menú lateral (*drawer*)

Permite navegar a administración y modificación de perfil, y cerrar sesión.

```
Widget _buildMenuLateral(BuildContext context) {
  final nombre = currentUserDisplayName;
  final correo = currentUserEmail ?? '';
  return Drawer(
    child: SafeArea(child: ListView(children: [
      DrawerHeader(/* ... avatar y datos ... */),
      ListTile(leading: const Icon(Icons.settings), title: const
          Text('Administrar perfil'),
            onTap: () => context.pushNamed(
               AdministrarPerfilWidget.routeName)),
      ListTile(leading: const Icon(Icons.edit), title: const
         Text('Modificar perfil'),
            onTap: () => context.pushNamed(
               ModificarPerfilWidget.routeName)),
      const Divider(),
      ListTile(leading: const Icon(Icons.logout, color: Colors.
         red),
```

```
                title: const Text('Cerrar sesion', style:
                    TextStyle(color: Colors.red)),
                onTap: () async { await authManager.signOut();
                    context.goNamed(LoginWidget.routeName); }),
    ])),
  );
}
```

3.3.2.13. Modelo asociado (`MedidasModel`)

Este modelo guarda el estado efímero de la pantalla de medidas (conmutadores de alertas) y ofrece los puntos de ciclo de vida para su gestión.

```
import '/flutter_flow/flutter_flow_util.dart';
import 'medidas_widget.dart' show MedidasWidget;
import 'package:flutter/material.dart';

class MedidasModel extends FlutterFlowModel<MedidasWidget> {
  // Switch fields
  bool? switchValue1;
  bool? switchValue2;
  bool? switchValue3;

  @override
  void initState(BuildContext context) {}

  @override
  void dispose() {}
}
```

3.4. Aplicación móvil (Dispositivos_widget.dart)

La pantalla `DispositivosWidget` permite al usuario: recuperar el `DEV_EUI` del ultimo dispositivo conocido, editar metadatos del sensor (modelo, ubicacion, modo), aplicar cambios en el nodo mediante *downlink* ASCII .

La comunicación con el backend se realiza contra la API desplegada en Railway: `https://tfg-proyecto-ithapp-production.up.railway.app`. Las rutas usadas son: `/api/dev-eui-ultimo`, `/dev-eui-ultimo`, `/api/sensores/actualizar` y `/api/downlink`.

3.4.1. Funciones principales

- **Obtencion del DEV_EUI**: primero consulta `/api/dev-eui-ultimo` y, en caso de fallo, `/dev-eui-ultimo`.

- **Actualizacion de sensor**: `PUT /api/sensores/actualizar` con los campos editados.
- **Downlink al nodo**: `POST /api/downlink` enviando un comando ASCII (codificado a base64 en el backend).
- **Modos**:
  - *Automatico*: fija umbral ITH y manda `MODE=AUTO;TH=#`.
  - *Manual*: manda `MODE=MANUAL` y permite `ACTIVATE`/`DEACTIVATE`.
- **UX**: mensajes `SnackBar` para exito/errores y capa de carga mientras se guarda.

Estas son las funciones en el código donde se hace toda la funcionalidad de este archivo de Dart:

- `_tryFetchDevEui(path)`: GET a la ruta indicada; si hay `dev_eui`, lo rellena en el formulario.
- `_obtenerDevEui()`: intenta primero `/api/dev-eui-ultimo`, despues `/dev-eui-ultimo`.
- `_enviarDownlink(cmd)`: POST con `dev_eui` y `cmd`. Muestra resultado por `SnackBar`.
- `_guardarCambios()`: valida, construye `payload`, hace `PUT` y, segun el modo, envia *downlink*.

3.4.2. Codigo (Dart)

A continuacion se incluye el archivo con comentarios y cadenas sin acentos para compatibilidad con `lstlisting`.

Listing 3.13: dispositivos_widget.dart: edicion de sensor y downlink

```
1
2  import 'dart:convert';
3  import 'package:flutter/material.dart';
4  import 'package:google_fonts/google_fonts.dart';
5  import 'package:http/http.dart' as http;
6
7  import '/flutter_flow/flutter_flow_icon_button.dart';
8  import '/flutter_flow/flutter_flow_theme.dart';
9  import '/flutter_flow/flutter_flow_widgets.dart';
10
11 /* API base (Railway) */
12 const String apiBase = 'https://tfg-proyecto-ithapp-production.
      up.railway.app';
13
14 /* Rutas de la API usadas por este widget */
15 const String getLastDevEuiPathApi  = '/api/dev-eui-ultimo';
```

```
16 const String getLastDevEuiPathRoot = '/dev-eui-ultimo';
17 const String updateSensorPath      = '/api/sensores/actualizar';
18 const String downlinkPath          = '/api/downlink';
19
20 class DispositivosWidget extends StatefulWidget {
21   const DispositivosWidget({super.key});
22
23   static const String routeName = 'dispositivos';
24   static const String routePath = '/dispositivos';
25
26   @override
27   State<DispositivosWidget> createState() =>
       _DispositivosWidgetState();
28 }
29
30 class _DispositivosWidgetState extends State<DispositivosWidget>
     {
31   final _scaffoldKey = GlobalKey<ScaffoldState>();
32
33   // Controladores de formulario
34   final _serieCtrl  = TextEditingController();   // dev_eui /
       numero de serie
35   final _modeloCtrl = TextEditingController();
36   final _areaCtrl   = TextEditingController();
37   final _zonaCtrl   = TextEditingController();
38   final _salaCtrl   = TextEditingController();
39
40   // Estado de operacion
41   bool   _saving = false;
42   String _modo   = 'auto';   // 'auto' | 'manual'
43   double _umbral = 75;
44
45   @override
46   void dispose() {
47     // Liberar recursos de los controladores
48     _serieCtrl.dispose();
49     _modeloCtrl.dispose();
50     _areaCtrl.dispose();
51     _zonaCtrl.dispose();
52     _salaCtrl.dispose();
53     super.dispose();
54   }
55
56   // Reglas minimas de validacion: requerimos DEV_EUI y modelo
57   bool _camposValidos() =>
58       _serieCtrl.text.trim().isNotEmpty &&
59       _modeloCtrl.text.trim().isNotEmpty;
60
61   // ------------------------------------------------------------
62   // DEV_EUI: intenta /api/... y luego /...
```

```
63  // ------------------------------------------------------------
64  Future<bool> _tryFetchDevEui(String path) async {
65    final uri  = Uri.parse('$apiBase$path');
66    final resp = await http.get(uri).timeout(const Duration(
        seconds: 12));
67    if (resp.statusCode == 200) {
68      final data = jsonDecode(resp.body) as Map<String, dynamic
          >;
69      final dev  = (data['dev_eui'] ?? '').toString();
70      if (dev.isNotEmpty) {
71        setState(() => _serieCtrl.text = dev);
72        return true;
73      }
74    }
75    return false;
76  }
77
78  Future<void> _obtenerDevEui() async {
79    try {
80      final okApi  = await _tryFetchDevEui(getLastDevEuiPathApi)
          ;
81      final okRoot = okApi ? true : await _tryFetchDevEui(
          getLastDevEuiPathRoot);
82      if (okApi || okRoot) {
83        ScaffoldMessenger.of(context).showSnackBar(
84          const SnackBar(content: Text('DEV_EUI obtenido.')),
85        );
86      } else {
87        throw 'No se pudo obtener el DEV_EUI. Envia un uplink
            desde el nodo.';
88      }
89    } catch (e) {
90      ScaffoldMessenger.of(context).showSnackBar(
91        SnackBar(content: Text('Error: $e')),
92      );
93    }
94  }
95
96  // ------------------------------------------------------------
97  // Downlink generico (comando ASCII que el backend convierte
98  // a base64 y envia a TTN en FPort 1)
99  // ------------------------------------------------------------
100 Future<void> _enviarDownlink(String cmd) async {
101   final dev = _serieCtrl.text.trim();
102   if (dev.isEmpty) {
103     ScaffoldMessenger.of(context).showSnackBar(
104       const SnackBar(content: Text('Indica primero el DEV_EUI.
            ')),
105     );
106     return;
```

```
107     }
108     try {
109       final uri  = Uri.parse('$apiBase$downlinkPath');
110       final resp = await http.post(
111         uri,
112         headers: {'Content-Type': 'application/json'},
113         body: jsonEncode({'dev_eui': dev, 'cmd': cmd}),
114       ).timeout(const Duration(seconds: 12));
115
116       if (resp.statusCode == 200) {
117         ScaffoldMessenger.of(context).showSnackBar(
118           SnackBar(content: Text('Downlink enviado: $cmd')),
119         );
120       } else {
121         throw 'HTTP ${resp.statusCode}';
122       }
123     } catch (e) {
124       ScaffoldMessenger.of(context).showSnackBar(
125         SnackBar(content: Text('Fallo el downlink: $e')),
126       );
127     }
128   }
129
130   // ------------------------------------------------------------
131   // Guardar cambios: PUT del sensor y downlink MODE/TH
132   // ------------------------------------------------------------
133   String? _nullIfEmpty(String s) => s.trim().isEmpty ? null : s.
        trim();
134   int? _intOrNull(String s) => s.trim().isEmpty ? null : int.
        tryParse(s.trim());
135
136   Future<void> _guardarCambios() async {
137     if (!_camposValidos()) {
138       ScaffoldMessenger.of(context).showSnackBar(
139         const SnackBar(content: Text('Completa al menos Nro de
              serie y Modelo.')),
140       );
141       return;
142     }
143
144     setState(() => _saving = true);
145
146     final payload = {
147       'dev_eui'    : _serieCtrl.text.trim().toUpperCase(),
148       'modelo'     : _nullIfEmpty(_modeloCtrl.text),
149       'area'       : _nullIfEmpty(_areaCtrl.text),
150       'zona'       : _intOrNull(_zonaCtrl.text),   // null si
            vacio
151       'sala'       : _nullIfEmpty(_salaCtrl.text),
152       'modo'       : _modo,
```

```
153       'umbral_ith' : _modo == 'auto' ? _umbral.round() : null,
154     };
155
156     try {
157       final uri  = Uri.parse('$apiBase$updateSensorPath');
158       final resp = await http.put(
159         uri,
160         headers: {'Content-Type':'application/json'},
161         body: jsonEncode(payload),
162       ).timeout(const Duration(seconds: 12));
163
164       if (resp.statusCode != 200) throw 'HTTP ${resp.statusCode}
              ';
165
166       // Aplicar downlink segun el modo seleccionado
167       if (_modo == 'auto') {
168         await _enviarDownlink('MODE=AUTO;TH=${_umbral.round()}')
                ;
169       } else {
170         await _enviarDownlink('MODE=MANUAL');
171       }
172
173       if (!mounted) return;
174       ScaffoldMessenger.of(context).showSnackBar(
175         const SnackBar(content: Text('Cambios guardados')),
176       );
177       Navigator.pop(context);
178     } catch (e) {
179       if (mounted) {
180         ScaffoldMessenger.of(context).showSnackBar(
181           SnackBar(content: Text('Error al guardar: $e')),
182         );
183       }
184     } finally {
185       if (mounted) setState(() => _saving = false);
186     }
187   }
188
189   // ------------------------------------------------------------
190   // UI
191   // ------------------------------------------------------------
192   @override
193   Widget build(BuildContext context) {
194     final theme = FlutterFlowTheme.of(context);
195
196     // Decoracion comun de campos de texto
197     InputDecoration deco(String label, {Widget? suffix}) =>
          InputDecoration(
198       labelText: label,
199       filled: true,
```

```
200       fillColor: theme.primaryBackground,
201       enabledBorder: OutlineInputBorder(
202         borderSide: BorderSide(color: theme.tertiary),
203         borderRadius: BorderRadius.circular(8),
204       ),
205       focusedBorder: OutlineInputBorder(
206         borderSide: BorderSide(color: theme.primary),
207         borderRadius: BorderRadius.circular(8),
208       ),
209       suffixIcon: suffix,
210     );
211
212     return GestureDetector(
213       onTap: () => FocusScope.of(context).unfocus(),
214       child: Scaffold(
215         key: _scaffoldKey,
216         backgroundColor: theme.primaryBackground,
217
218         appBar: AppBar(
219           backgroundColor: theme.primary,
220           elevation: 2,
221           leading: FlutterFlowIconButton(
222             borderRadius: 8,
223             buttonSize: 40,
224             icon: const Icon(Icons.arrow_back, color: Colors.
                  white),
225             onPressed: () => Navigator.pop(context),
226           ),
227           title: const Text('Editar sensor',
228               style: TextStyle(color: Colors.white, fontSize:
                     22)),
229         ),
230
231         body: Stack(
232           children: [
233             SingleChildScrollView(
234               padding: const EdgeInsets.fromLTRB(16, 16, 16,
                     120),
235               child: Column(
236                 crossAxisAlignment: CrossAxisAlignment.stretch,
237                 children: [
238
239                   // ---- Informacion del dispositivo
240                   _Panel(
241                     title: 'Informacion del dispositivo',
242                     child: Column(
243                       children: [
244                         TextFormField(
245                           controller: _serieCtrl,
246                           decoration: deco(
```

```
247                         'Numero de serie (DEV_EUI)',
248                         suffix: IconButton(
249                           tooltip: 'Obtener automaticamente',
250                           icon: const Icon(Icons.qr_code_2_outlined),
251                           onPressed: _obtenerDevEui,
252                         ),
253                       ),
254                     ),
255                     const SizedBox(height: 16),
256                     TextFormField(
257                       controller: _modeloCtrl,
258                       decoration: deco('Modelo'),
259                     ),
260                   ],
261                 ),
262               ),
263 
264               const SizedBox(height: 16),
265 
266               // ---- Ubicacion
267               _Panel(
268                 title: 'Ubicacion',
269                 child: Column(
270                   children: [
271                     TextFormField(
272                       controller: _areaCtrl,
273                       decoration: deco('Granja'),
274                     ),
275                     const SizedBox(height: 16),
276                     Row(
277                       children: [
278                         SizedBox(
279                           width: 110,
280                           child: TextFormField(
281                             controller: _zonaCtrl,
282                             keyboardType: TextInputType.text,
283                             decoration: deco('Zona'),
284                           ),
285                         ),
286                         const SizedBox(width: 12),
287                         Expanded(
288                           child: TextFormField(
289                             controller: _salaCtrl,
290                             decoration: deco('Sala / Habitacion'),
291                           ),
292                         ),
```

```
293                     ],
294                   ),
295                 ],
296               ),
297             ),
298
299             const SizedBox(height: 16),
300
301             // ---- Modo de operacion
302             _Panel(
303               title: 'Modo de operacion',
304               child: Column(
305                 crossAxisAlignment: CrossAxisAlignment.
                      start,
306                 children: [
307                   RadioListTile<String>(
308                     value: 'auto',
309                     groupValue: _modo,
310                     title: const Text('Automatico (activa
                        ventilador por ITH)'),
311                     subtitle: const Text('Se encendera
                        cuando el ITH sea >= umbral.'),
312                     onChanged: (v) => setState(() => _modo
                          = v!),
313                   ),
314                   if (_modo == 'auto') ...[
315                     const SizedBox(height: 8),
316                     Text('Umbral ITH: ${_umbral.round()}',
317                         style: GoogleFonts.inter(fontSize:
                              14)),
318                     Slider(
319                       value: _umbral,
320                       onChanged: (v) => setState(() =>
                          _umbral = v),
321                       min: 60, max: 90, divisions: 30,
322                     ),
323                   ],
324                   const Divider(height: 24),
325                   RadioListTile<String>(
326                     value: 'manual',
327                     groupValue: _modo,
328                     title: const Text('Manual'),
329                     subtitle: const Text('Control
                        inmediato desde la app.'),
330                     onChanged: (v) => setState(() => _modo
                          = v!),
331                   ),
332                   if (_modo == 'manual') ...[
333                     const SizedBox(height: 8),
334                     Row(
```

```
335                             children: [
336                               Expanded(
337                                 child: OutlinedButton.icon(
338                                   icon: const Icon(Icons.
                                      power_settings_new),
339                                   label: const Text('Encender'),
340                                   onPressed: () =>
                                      _enviarDownlink('ACTIVATE'),
341                                 ),
342                               ),
343                               const SizedBox(width: 12),
344                               Expanded(
345                                 child: OutlinedButton.icon(
346                                   icon: const Icon(Icons.
                                      power_off),
347                                   label: const Text('Apagar'),
348                                   onPressed: () =>
                                      _enviarDownlink('DEACTIVATE'
                                      ),
349                                 ),
350                               ),
351                             ],
352                           ),
353                         ],
354                       ],
355                     ),
356                   ),
357                 ],
358               ),
359             ),
360
361             if (_saving)
362               Container(
363                 color: Colors.black45,
364                 child: const Center(child:
                    CircularProgressIndicator()),
365               ),
366           ],
367         ),
368
369         bottomNavigationBar: Padding(
370           padding: const EdgeInsets.fromLTRB(24, 0, 24, 24),
371           child: Column(
372             mainAxisSize: MainAxisSize.min,
373             children: [
374               FFButtonWidget(
375                 text: 'Guardar cambios',
376                 onPressed: _saving ? null : _guardarCambios,
377                 options: FFButtonOptions(
378                   width: double.infinity,
```

```
                    height: 50,
                    color: theme.success,
                    textStyle: theme.titleSmall.override(
                      fontFamily: 'InterTight',
                      color: Colors.white,
                    ),
                    borderRadius: BorderRadius.circular(30),
                  ),
                ),
                const SizedBox(height: 12),
                FFButtonWidget(
                  text: 'Cancelar',
                  onPressed: () => Navigator.pop(context),
                  options: FFButtonOptions(
                    width: double.infinity,
                    height: 50,
                    color: theme.alternate,
                    textStyle: theme.titleSmall,
                    borderRadius: BorderRadius.circular(30),
                  ),
                ),
              ],
            ),
          ),
        ),
      );
  }
}

// ------------------------------------------------------------
// Componente contenedor de panel con sombra y titulo
// ------------------------------------------------------------
class _Panel extends StatelessWidget {
  const _Panel({required this.title, required this.child});
  final String title;
  final Widget child;

  @override
  Widget build(BuildContext context) {
    final theme = FlutterFlowTheme.of(context);
    return Container(
      width: double.infinity,
      decoration: BoxDecoration(
        color: theme.secondaryBackground,
        boxShadow: const [
          BoxShadow(blurRadius: 4, color: Color(0x20000000),
             offset: Offset(0, 2))
        ],
        borderRadius: BorderRadius.circular(12),
      ),
```

```
428         padding: const EdgeInsets.all(16),
429         child: Column(
430           crossAxisAlignment: CrossAxisAlignment.start,
431           children: [
432             Text(title,
433                 style: theme.titleMedium.override(
434                   fontFamily: 'InterTight',
435                   fontWeight: FontWeight.w600,
436                 )),
437             const SizedBox(height: 12),
438             child,
439           ],
440         ),
441       );
442   }
443 }
```

3.4.3. Aplicación móvil: pantalla de autenticación (`LoginWidget`)

La pantalla `LoginWidget` implementa el acceso a la aplicación mediante correo y contraseña usando el módulo de autenticación. Sus responsabilidades son:

- Captura y validación básica de email y contraseña.
- Autenticación mediante `authManager.signInWithEmail`.
- Navegación hacia la pantalla principal (`MedidasWidget`) al iniciar sesión.
- Acceso a recuperación de contraseña y registro de nuevos usuarios.

3.4.3.1. Imports principales

Se usan utilidades de FlutterFlow (temas, navegación), el gestor de autenticación y widgets de la propia app.

```
1  import '/auth/firebase_auth/auth_util.dart';
2  import '/flutter_flow/flutter_flow_theme.dart';
3  import '/flutter_flow/flutter_flow_util.dart';
4  import '/flutter_flow/flutter_flow_widgets.dart';
5  import '/index.dart';
6  import 'package:flutter/material.dart';
7  import 'package:google_fonts/google_fonts.dart';
8  import 'package:provider/provider.dart';
9  import 'login_model.dart';
10 export 'login_model.dart';
```

3.4.3.2. Estructura del widget y estado

El estado mantiene controladores y *focus nodes* para los campos de email y contraseña, además de la visibilidad de la contraseña.

```
class LoginWidget extends StatefulWidget {
  const LoginWidget({super.key});
  static String routeName = 'login';
  static String routePath = '/login';
  @override
  State<LoginWidget> createState() => _LoginWidgetState();
}

class _LoginWidgetState extends State<LoginWidget> {
  late LoginModel _model;
  final scaffoldKey = GlobalKey<ScaffoldState>();

  @override
  void initState() {
    super.initState();
    _model = createModel(context, () => LoginModel());
    _model.emailAddressTextController ??= TextEditingController
        ();
    _model.emailAddressFocusNode ??= FocusNode();
    _model.passwordTextController ??= TextEditingController();
    _model.passwordFocusNode ??= FocusNode();
  }

  @override
  void dispose() {
    _model.dispose();
    super.dispose();
  }
}
```

3.4.3.3. Interfaz: formulario de acceso

El formulario incluye campos con estilo de FlutterFlow, con máscara de contraseña y icono para alternar visibilidad.

```
// Campo de email
TextFormField(
  controller: _model.emailAddressTextController,
  focusNode: _model.emailAddressFocusNode,
  decoration: InputDecoration(
    labelText: 'Email',
    hintText: 'Introduzca el email...',
    filled: true,
    fillColor: FlutterFlowTheme.of(context).secondaryBackground,
```

```
    enabledBorder: OutlineInputBorder(
      borderSide: BorderSide(color: FlutterFlowTheme.of(context)
          .primaryBackground, width: 2),
      borderRadius: BorderRadius.circular(40),
    ),
  ),
  validator: _model.emailAddressTextControllerValidator.
      asValidator(context),
),

// Campo de contrasena.
TextFormField(
  controller: _model.passwordTextController,
  focusNode: _model.passwordFocusNode,
  obscureText: !_model.passwordVisibility,
  decoration: InputDecoration(
    labelText: 'Clave',
    hintText: 'Introduzca la contrasena...',
    filled: true,
    fillColor: FlutterFlowTheme.of(context).secondaryBackground,
    suffixIcon: InkWell(
      onTap: () => safeSetState(() => _model.passwordVisibility
          = !_model.passwordVisibility),
      child: Icon(
        _model.passwordVisibility ? Icons.visibility_outlined :
            Icons.visibility_off_outlined,
        color: FlutterFlowTheme.of(context).secondaryText, size:
            22,
      ),
    ),
  ),
  validator: _model.passwordTextControllerValidator.asValidator(
      context),
),
```

3.4.3.4. Acciones: recuperación, inicio de sesión y registro

Se ofrecen rutas para recuperar contraseña, iniciar sesión y registrarse.

```
// Enlace a "  Olvido   la contrasena?"
FFButtonWidget(
  onPressed: () async { context.pushNamed(ForgotpasswordWidget.
      routeName); },
  text: '  Olvido   la contrasena?',
  options: FFButtonOptions(color: const Color(0x00FFFFFF),
      elevation: 0),
),

// Boton "Iniciar Sesion" (flujo de autenticaci n)
```

```
FFButtonWidget(
  onPressed: () async {
    GoRouter.of(context).prepareAuthEvent();
    final user = await authManager.signInWithEmail(
      context,
      _model.emailAddressTextController.text,
      _model.passwordTextController.text,
    );
    if (user == null) return; // Error ya gestionado por
       authManager/FlutterFlow
    context.goNamedAuth(MedidasWidget.routeName, context.mounted
       ); // Navega tras login
  },
  text: 'Iniciar Sesion',
  options: FFButtonOptions(
    color: FlutterFlowTheme.of(context).primary, elevation: 3,
  ),
),

// Enlace a registro
FFButtonWidget(
  onPressed: () async { context.pushNamed(RegistroWidget.
     routeName); },
  text: 'Registrarse',
  options: FFButtonOptions(
    color: FlutterFlowTheme.of(context).primary, elevation: 0,
  ),
),
```

3.4.3.5. Composición visual

La pantalla integra un logotipo, títulos y el formulario dentro de un `SingleChildScrollView` para adaptarse a diferentes tamaños de pantalla y teclado.

```
Scaffold(
  key: scaffoldKey,
  backgroundColor: FlutterFlowTheme.of(context).
     primaryBackground,
  body: SafeArea(
    child: SingleChildScrollView(
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          // Logotipo superior
          Padding(
            padding: const EdgeInsetsDirectional.fromSTEB(70, 0,
                0, 30),
            child: Image.asset('assets/images/ith.PNG', width:
               200, height: 250, fit: BoxFit.cover),
```

```
            ),
            Text('Bienvenido/a', style: FlutterFlowTheme.of(
               context).displaySmall),
            const SizedBox(height: 4),
            Text('Inicia Sesion', style: FlutterFlowTheme.of(
               context).bodySmall),

            // Campos y botones (bloques anteriores)
            // ...
          ],
        ),
      ),
    ),
  ),
)
```

3.4.3.6. Modelo asociado (LoginModel)

Gestiona controladores, focos y la visibilidad de la contraseña para la pantalla de login.

```
import '/flutter_flow/flutter_flow_util.dart';
import 'login_widget.dart' show LoginWidget;
import 'package:flutter/material.dart';

class LoginModel extends FlutterFlowModel<LoginWidget> {
  // Email field
  FocusNode? emailAddressFocusNode;
  TextEditingController? emailAddressTextController;
  String? Function(BuildContext, String?)?
     emailAddressTextControllerValidator;

  // Password field
  FocusNode? passwordFocusNode;
  TextEditingController? passwordTextController;
  late bool passwordVisibility;
  String? Function(BuildContext, String?)?
     passwordTextControllerValidator;

  @override
  void initState(BuildContext context) {
    passwordVisibility = false;
  }

  @override
  void dispose() {
    emailAddressFocusNode?.dispose();
    emailAddressTextController?.dispose();
    passwordFocusNode?.dispose();
    passwordTextController?.dispose();
```

```
28    }
29  }
```

3.4.4. Aplicación móvil: pantalla de registro (`RegistroWidget`)

La pantalla `RegistroWidget` recoge los datos básicos del usuario y de su entorno de trabajo (granja y sensor) y, tras validación, crea las entidades correspondientes a través del backend. Sus responsabilidades son:

- Capturar email, nombre, contraseña, fecha de nacimiento, nombre de la granja, direccion y nombre del sensor.
- Validar campos obligatorios y formatos básicos (email, longitud de la contraseña).
- Llamar a los endpoints REST del backend para crear `/granjas`, `/usuarios` y `/sensores`.
- Mostrar retroalimentación al usuario (cargando, éxito, error) y navegar a `MedidasWidget`.

3.4.4.1. Imports principales

```
1  import '/auth/firebase_auth/auth_util.dart';
2  import '/backend/backend.dart';
3  import '/flutter_flow/flutter_flow_theme.dart';
4  import '/flutter_flow/flutter_flow_util.dart';
5  import '/flutter_flow/flutter_flow_widgets.dart';
6  import '/flutter_flow/random_data_util.dart' as random_data;
7  import '/index.dart';
8  import 'package:flutter/material.dart';
9  import 'package:google_fonts/google_fonts.dart';
10 import 'package:provider/provider.dart';
11 import 'registro_model.dart';
12 export 'registro_model.dart';
13 import 'package:http/http.dart' as http;
14 import 'dart:convert';
15 import 'package:mysql1/mysql1.dart';
```

3.4.4.2. Estado y controladores de formulario

```
1  class RegistroWidget extends StatefulWidget {
2    const RegistroWidget({super.key});
3    static String routeName = 'registro';
4    static String routePath = '/registro';
5    @override
6    State<RegistroWidget> createState() => _RegistroWidgetState();
```

```
}

class _RegistroWidgetState extends State<RegistroWidget> {
  late RegistroModel _model;
  final scaffoldKey = GlobalKey<ScaffoldState>();
  LatLng? currentUserLocationValue;

  @override
  void initState() {
    super.initState();
    _model = createModel(context, () => RegistroModel());
    _model.usuarioTextController ??= TextEditingController();
    _model.usuarioFocusNode ??= FocusNode();
    _model.nombreTextController ??= TextEditingController();
    _model.nombreFocusNode ??= FocusNode();
    _model.claveTextController ??= TextEditingController();
    _model.claveFocusNode ??= FocusNode();
    _model.nGranjaTextController ??= TextEditingController();
    _model.nGranjaFocusNode ??= FocusNode();
    _model.ubicacionTextController ??= TextEditingController();
    _model.ubicacionFocusNode ??= FocusNode();
    _model.nSensorTextController ??= TextEditingController();
    _model.nSensorFocusNode ??= FocusNode();
  }

  @override
  void dispose() {
    _model.dispose();
    super.dispose();
  }
}
```

3.4.4.3. Estado y controladores de formulario

```
class RegistroWidget extends StatefulWidget {
  const RegistroWidget({super.key});
  static String routeName = 'registro';
  static String routePath = '/registro';
  @override
  State<RegistroWidget> createState() => _RegistroWidgetState();
}

class _RegistroWidgetState extends State<RegistroWidget> {
  late RegistroModel _model;
  final scaffoldKey = GlobalKey<ScaffoldState>();
  LatLng? currentUserLocationValue;

  @override
  void initState() {
```

```
16      super.initState();
17      _model = createModel(context, () => RegistroModel());
18      _model.usuarioTextController ??= TextEditingController();
19      _model.usuarioFocusNode ??= FocusNode();
20      _model.nombreTextController ??= TextEditingController();
21      _model.nombreFocusNode ??= FocusNode();
22      _model.claveTextController ??= TextEditingController();
23      _model.claveFocusNode ??= FocusNode();
24      _model.nGranjaTextController ??= TextEditingController();
25      _model.nGranjaFocusNode ??= FocusNode();
26      _model.ubicacionTextController ??= TextEditingController();
27      _model.ubicacionFocusNode ??= FocusNode();
28      _model.nSensorTextController ??= TextEditingController();
29      _model.nSensorFocusNode ??= FocusNode();
30    }
31
32    @override
33    void dispose() {
34      _model.dispose();
35      super.dispose();
36    }
37  }
```

3.4.4.4. Ayudas de acceso a datos (Railway)

En este archivo, todo el registro se realiza contra la API pública desplegada en Railway mediante peticiones `HTTP`.

Base de la API y helper Se define la URL base del backend y un helper para componer rutas.

```
1  const String kApiBase = String.fromEnvironment(
2    'API_BASE_URL',
3    defaultValue: 'https://tfg-proyecto-ithapp-production.up.
       railway.app',
4  );
5
6  Uri api(String path) => Uri.parse('$kApiBase$path');
```

Cliente simple del backend Encapsula las operaciones de alta de granja, usuario y sensor. Devuelve los `id` creados y lanza una excepción si la API no responde 200/201.

```
1  import 'package:http/http.dart' as http;
2  import 'dart:convert';
3  import 'package:intl/intl.dart';
4
5  class Backend {
```

```
6   final http.Client _client;
7   Backend([http.Client? client]) : _client = client ?? http.
        Client();
8
9   Future<int> crearGranja({
10    required String nombre,
11    required String direccion,
12  }) async {
13    final res = await _client.post(
14      api('/granjas'),
15      headers: {'Content-Type': 'application/json'},
16      body: jsonEncode({'nombre_granja': nombre, 'direccion':
            direccion}),
17    );
18    if (res.statusCode != 200 && res.statusCode != 201) {
19      throw Exception('Error creando granja: ${res.body}');
20    }
21    final data = jsonDecode(res.body) as Map<String, dynamic>;
22    return (data['id_granja'] as num).toInt();
23  }
24
25  Future<int> crearUsuario({
26    required String nombre,
27    required String apellidos,
28    required String email,
29    required DateTime fechaNacimiento,
30    required String password,
31    required int idGranja,
32  }) async {
33    final res = await _client.post(
34      api('/usuarios'),
35      headers: {'Content-Type': 'application/json'},
36      body: jsonEncode({
37        'nombre': nombre,
38        'apellidos': apellidos,
39        'email': email,
40        'fecha_nacimiento': DateFormat('yyyy-MM-dd').format(
              fechaNacimiento),
41        'password': password,
42        'id_granja': idGranja,
43      }),
44    );
45    if (res.statusCode != 200 && res.statusCode != 201) {
46      throw Exception('Error creando usuario: ${res.body}');
47    }
48    final data = jsonDecode(res.body) as Map<String, dynamic>;
49    return (data['id_usuario'] as num).toInt();
50  }
51
52  Future<int> crearSensor({
```

```
53      required String nombreSensor,
54      required int idGranja,
55    }) async {
56      final res = await _client.post(
57        api('/sensores'),
58        headers: {'Content-Type': 'application/json'},
59        body: jsonEncode({'nombre_sensor': nombreSensor, '
              id_granja': idGranja}),
60      );
61      if (res.statusCode != 200 && res.statusCode != 201) {
62        throw Exception('Error creando sensor: ${res.body}');
63      }
64      final data = jsonDecode(res.body) as Map<String, dynamic>;
65      return (data['id_sensor'] as num).toInt();
66    }
67
68    /// Flujo completo: granja     usuario     sensor
69    Future<void> registroCompleto({
70      required String nombreGranja,
71      required String direccionGranja,
72      required String nombreUsuario,
73      required String apellidos,
74      required String email,
75      required DateTime fechaNacimiento,
76      required String password,
77      required String nombreSensor,
78    }) async {
79      final idGranja = await crearGranja(
80        nombre: nombreGranja,
81        direccion: direccionGranja,
82      );
83
84      await crearUsuario(
85        nombre: nombreUsuario,
86        apellidos: apellidos,
87        email: email,
88        fechaNacimiento: fechaNacimiento,
89        password: password,
90        idGranja: idGranja,
91      );
92
93      await crearSensor(
94        nombreSensor: nombreSensor,
95        idGranja: idGranja,
96      );
97    }
98  }
```

Acción del botón «Registrar» Ejecuta el flujo completo contra la API de Railway y muestra el resultado.

```
final _api = Backend();
bool _saving = false;

Future<void> _onRegistrar() async {
  try {
    setState(() => _saving = true);

    final nombreGranja    = _model.nGranjaTextController?.text.trim() ?? '';
    final direccionGranja = _model.ubicacionTextController?.text.trim() ?? '';
    final nombreUsuario   = _model.nombreTextController?.text.trim() ?? '';
    final apellidos       = ''; // usa campo real si lo tienes
    final email           = _model.usuarioTextController?.text.trim() ?? '';
    final password        = _model.claveTextController?.text.trim() ?? '';
    final nombreSensor    = _model.nSensorTextController?.text.trim() ?? '';

    final fechaNacimiento = DateTime(2000, 1, 1); // sustituir por selector real

    if ([nombreGranja, direccionGranja, nombreUsuario, email, password, nombreSensor]
        .any((s) => s.isEmpty)) {
      showSnackbar(context, 'Completa todos los campos obligatorios');
      return;
    }

    await _api.registroCompleto(
      nombreGranja: nombreGranja,
      direccionGranja: direccionGranja,
      nombreUsuario: nombreUsuario,
      apellidos: apellidos,
      email: email,
      fechaNacimiento: fechaNacimiento,
      password: password,
      nombreSensor: nombreSensor,
    );

    showSnackbar(context, 'Registro completado con  xito ');
    context.pushNamed(LoginWidget.routeName);
  } catch (e) {
    showSnackbar(context, 'Error registrando: $e');
  } finally {
```

```
    if (mounted) setState(() => _saving = false);
  }
}
```

3.4.4.5. Campos del formulario

```
// Email
TextFormField(
  controller: _model.usuarioTextController,
  focusNode: _model.usuarioFocusNode,
  decoration: InputDecoration(
    labelText: 'Email', hintText: 'Email',
    enabledBorder: OutlineInputBorder(
      borderSide: BorderSide(color: FlutterFlowTheme.of(context)
          .primary, width: 1),
    ),
  ),
  keyboardType: TextInputType.emailAddress,
  validator: _model.usuarioTextControllerValidator.asValidator(
      context),
),

// Nombre y apellidos (texto libre)
TextFormField(
  controller: _model.nombreTextController,
  focusNode: _model.nombreFocusNode,
  decoration: InputDecoration(
    labelText: 'Nombre', hintText: 'Nombre y apellidos',
  ),
  validator: _model.nombreTextControllerValidator.asValidator(
      context),
),

// Contrasena con alternancia de visibilidad
TextFormField(
  controller: _model.claveTextController,
  focusNode: _model.claveFocusNode,
  obscureText: !_model.claveVisibility,
  decoration: InputDecoration(
    labelText: 'Contrasena', hintText: 'Introduzca la contrasena
        ...',
    suffixIcon: InkWell(
      onTap: () => safeSetState(() => _model.claveVisibility = !
          _model.claveVisibility),
      child: Icon(_model.claveVisibility ? Icons.
          visibility_outlined : Icons.visibility_off_outlined),
    ),
  ),
```

```
37    validator: _model.claveTextControllerValidator.asValidator(
          context),
38  ),
```

3.4.4.6. Fecha de nacimiento

```
1   Text(dateTimeFormat("d/M/y", _model.datePicked)),
2   FFButtonWidget(
3     onPressed: () async {
4       final d = await showDatePicker(
5         context: context, initialDate: getCurrentTimestamp,
6         firstDate: DateTime(1900), lastDate: getCurrentTimestamp,
7       );
8       if (d != null) {
9         safeSetState(() => _model.datePicked = DateTime(d.year, d.
              month, d.day));
10      } else if (_model.datePicked != null) {
11        safeSetState(() => _model.datePicked = getCurrentTimestamp
              );
12      }
13      FFAppState().fechanacimiento = _model.datePicked;
14    },
15    text: '', icon: Icon(Icons.date_range_rounded, size: 30),
16  ),
```

3.4.4.7. Datos de granja y sensor

```
1   // Nombre del sensor
2   TextFormField(
3     controller: _model.nSensorTextController,
4     focusNode: _model.nSensorFocusNode,
5     decoration: const InputDecoration(labelText: 'Nombre sensor',
          hintText: 'Nombre sensor'),
6     validator: _model.nSensorTextControllerValidator.asValidator(
          context),
7   ),
8
9   // Nombre de la granja
10  TextFormField(
11    controller: _model.nGranjaTextController,
12    focusNode: _model.nGranjaFocusNode,
13    decoration: const InputDecoration(labelText: 'Nombre granja',
          hintText: 'Nombre granja'),
14    validator: _model.nGranjaTextControllerValidator.asValidator(
          context),
15  ),
16
```

```
// Direccion de la granja (texto libre)
TextFormField(
  controller: _model.ubicacionTextController,
  focusNode: _model.ubicacionFocusNode,
  decoration: const InputDecoration(labelText: 'Direccion granja', hintText: 'Direccion granja'),
  validator: _model.ubicacionTextControllerValidator.asValidator(context),
),
```

3.4.4.8. Acción principal: guardar registro

Botón que valida el formulario, muestra diálogos de éxito o error y ejecuta la secuencia **crear granja** → **crear usuario** → **crear sensor**. Al finalizar, navega a `MedidasWidget`.

```
Padding(
  padding: EdgeInsetsDirectional.fromSTEB(0.0, 30.0, 0.0, 20.0),
  child: FFButtonWidget(
  onPressed: () async {
  GoRouter.of(context).prepareAuthEvent();

  //        Cambia esto si quieres inyectarlo por env var
  const String baseUrl = 'https://tfg-proyecto-ithapp-production.up.railway.app';

  try {
    // 1) Recoger valores del formulario
    final email           = _model.usuarioTextController.text.trim();
    final password        = _model.claveTextController.text.trim();
    final nombre          = _model.nombreTextController.text.trim();
    final nombreGranja    = _model.nGranjaTextController.text.trim();
    final direccionGranja = _model.ubicacionTextController.text.trim();
    final nombreSensor    = _model.nSensorTextController.text.trim();
    final fechaPick       = _model.datePicked;

    // 2) Validaciones
    final List<String> missing = [];
    if (email.isEmpty)            missing.add('Email');
    if (password.isEmpty)         missing.add('Contrase a');
    if (nombre.isEmpty)           missing.add('Nombre');
    if (nombreGranja.isEmpty)     missing.add('Nombre de granja');
```

```
26      if (direccionGranja.isEmpty)  missing.add('Direcci n de
           granja');
27      if (fechaPick == null)        missing.add('Fecha de
           nacimiento');
28      if (nombreSensor.isEmpty)     missing.add('Nombre de sensor'
           );
29
30      if (missing.isNotEmpty) {
31        ScaffoldMessenger.of(context).showSnackBar(
32          SnackBar(
33            content: Text('Campos obligatorios faltantes: ${
                 missing.join(', ')}'),
34            backgroundColor: Colors.red,
35          ),
36        );
37        return;
38      }
39
40      final emailOk = RegExp(r'^[\w\.-]+@([\w-]+\.)+[\w-]{2,4}$').
           hasMatch(email);
41      if (!emailOk) {
42        ScaffoldMessenger.of(context).showSnackBar(
43          const SnackBar(content: Text('Por favor, introduce un
               email v lido'), backgroundColor: Colors.red),
44        );
45        return;
46      }
47
48      if (password.length < 6) {
49        ScaffoldMessenger.of(context).showSnackBar(
50          const SnackBar(content: Text('La contrase a debe tener
               al menos 6 caracteres'), backgroundColor: Colors.red),
51        );
52        return;
53      }
54
55      // 3) Loading
56      showDialog(
57        context: context,
58        barrierDismissible: false,
59        builder: (_) => const AlertDialog(
60          content: Row(
61            children: [
62              CircularProgressIndicator(),
63              SizedBox(width: 16),
64              Text('Registrando usuario...'),
65            ],
66          ),
67        ),
68      );
```

```
69
70      final fechaNacimientoISO = DateFormat('yyyy-MM-dd').format(
            fechaPick!);
71
72      // Logs de depuraci n
73      debugPrint('       $email');
74      debugPrint('       $nombre');
75      debugPrint('       $nombreGranja');
76      debugPrint('       $direccionGranja');
77      debugPrint('       $fechaNacimientoISO');
78      debugPrint('       $nombreSensor');
79
80      // 4) Crear granja
81      final granjaRes = await http.post(
82        Uri.parse('$baseUrl/granjas'),
83        headers: {'Content-Type': 'application/json'},
84        body: jsonEncode({
85          'nombre_granja': nombreGranja,
86          'direccion'    : direccionGranja,
87        }),
88      );
89
90      if (granjaRes.statusCode != 200 && granjaRes.statusCode !=
            201) {
91        Navigator.pop(context);
92        throw Exception('Error creando granja: ${granjaRes.
              statusCode} - ${granjaRes.body}');
93      }
94
95      final granjaJson = jsonDecode(granjaRes.body) as Map<String,
            dynamic>;
96      final int idGranja = (granjaJson['id_granja'] as num?)?.
            toInt() ?? -1;
97      if (idGranja <= 0) {
98        Navigator.pop(context);
99        throw Exception('No se pudo obtener el ID de la granja
              creada');
100     }
101     debugPrint('    Granja ID: $idGranja');
102
103     // 5) Crear usuario (requiere id_granja)
104     final usuarioRes = await http.post(
105       Uri.parse('$baseUrl/usuarios'),
106       headers: {'Content-Type': 'application/json'},
107       body: jsonEncode({
108         'nombre'          : nombre,
109         'apellidos'       : '',                  // ajusta si
              tienes el campo
110         'email'           : email,
111         'fecha_nacimiento': fechaNacimientoISO, // yyyy-MM-dd
```

```
112         'password'         : password,
113         'id_granja'        : idGranja,
114       }),
115     );
116
117     if (usuarioRes.statusCode != 200 && usuarioRes.statusCode !=
            201) {
118       Navigator.pop(context);
119       throw Exception('Error creando usuario: ${usuarioRes.
              statusCode} - ${usuarioRes.body}');
120     }
121     debugPrint('    Usuario creado: ${usuarioRes.body}');
122
123     // 6) Crear sensor
124     final sensorRes = await http.post(
125       Uri.parse('$baseUrl/sensores'),
126       headers: {'Content-Type': 'application/json'},
127       body: jsonEncode({
128         'nombre_sensor': nombreSensor,
129         'id_granja'    : idGranja,
130       }),
131     );
132
133     if (sensorRes.statusCode != 200 && sensorRes.statusCode !=
            201) {
134       Navigator.pop(context);
135       throw Exception('Error creando sensor: ${sensorRes.
              statusCode} - ${sensorRes.body}');
136     }
137     debugPrint('    Sensor creado: ${sensorRes.body}');
138
139     // 7) Guardar en estado global (lo que ya usabas)
140     FFAppState().email           = email;
141     FFAppState().nombre          = nombre;
142     FFAppState().fechanacimiento= fechaPick;
143     FFAppState().nombregranja    = nombreGranja;
144     FFAppState().ubicacion       = direccionGranja;
145     safeSetState(() {});
146
147     // 8) Cerrar loading
148     Navigator.pop(context);
149
150     // 9) Mensaje de  xito
151     await showDialog(
152       context: context,
153       builder: (ctx) => AlertDialog(
154         title: const Row(
155           children: [
156             Icon(Icons.check_circle, color: Colors.green),
157             SizedBox(width: 10),
```

```
158             Text('  Registro   Exitoso!'),
159           ],
160         ),
161         content: Column(
162           mainAxisSize: MainAxisSize.min,
163           crossAxisAlignment: CrossAxisAlignment.start,
164           children: [
165             const Text('Se ha registrado correctamente:'),
166             const SizedBox(height: 10),
167             Text('    Usuario: $nombre'),
168             Text('    Email: $email'),
169             Text('    Granja: $nombreGranja (ID: $idGranja)'),
170             Text('    Sensor: $nombreSensor'),
171           ],
172         ),
173         actions: [
174             TextButton(onPressed: () => Navigator.pop(ctx),
                child: const Text('Continuar')),
175         ],
176       ),
177     );
178
179     // 10) Navegar
180     context.pushNamedAuth(MedidasWidget.routeName, context.
        mounted);
181   } catch (e) {
182     if (Navigator.canPop(context)) Navigator.pop(context);
183     debugPrint('    Error durante el registro: $e');
184
185     await showDialog(
186       context: context,
187       builder: (ctx) => AlertDialog(
188         title: const Row(
189           children: [
190             Icon(Icons.error, color: Colors.red),
191             SizedBox(width: 10),
192             Text('Error de Registro'),
193           ],
194         ),
195         content: Column(
196           mainAxisSize: MainAxisSize.min,
197           crossAxisAlignment: CrossAxisAlignment.start,
198           children: [
199             const Text('Ha ocurrido un error durante el registro
                :'),
200             const SizedBox(height: 10),
201             Text(e.toString(), style: const TextStyle(fontFamily
                : 'monospace', fontSize: 12)),
202             const SizedBox(height: 10),
```

```
203             const Text('Por favor, verifica tu conexi n a
                    internet e int ntalo nuevamente.'),
204           ],
205         ),
206         actions: [
207           TextButton(onPressed: () => Navigator.pop(ctx), child:
                  const Text('Entendido')),
208         ],
209       ),
210     );
211   }
212 },
213   text: 'Guardar',
214   options: FFButtonOptions(
215     width: 130.0,
216     height: 40.0,
217     padding: EdgeInsetsDirectional.fromSTEB(0.0, 0.0, 0.0, 0.0),
218     iconPadding: EdgeInsetsDirectional.fromSTEB(0.0, 0.0, 0.0,
          0.0),
219     color: FlutterFlowTheme.of(context).primary,
220     textStyle: FlutterFlowTheme.of(context).titleSmall.override(
221       font: GoogleFonts.interTight(
222         fontWeight: FlutterFlowTheme.of(context).titleSmall.
              fontWeight,
223         fontStyle: FlutterFlowTheme.of(context).titleSmall.
              fontStyle,
224       ),
225       color: Colors.white,
226       letterSpacing: 0.0,
227       fontWeight: FlutterFlowTheme.of(context).titleSmall.
            fontWeight,
228       fontStyle: FlutterFlowTheme.of(context).titleSmall.
            fontStyle,
229     ),
230     borderSide: BorderSide(
231       color: Colors.transparent,
232       width: 1.0,
233     ),
234     borderRadius: BorderRadius.circular(8.0),
235   ),
236 ),
237 ),
```

3.4.4.9. Modelo asociado (`RegistroModel`)

Centraliza los controladores del formulario de alta (usuario, nombre, contraseña, granja, dirección, sensor), además del valor de fecha y flags de visibilidad.

```
1 import '/flutter_flow/flutter_flow_util.dart';
```

```dart
import 'registro_widget.dart' show RegistroWidget;
import 'package:flutter/material.dart';

class RegistroModel extends FlutterFlowModel<RegistroWidget> {
  // Usuario (email)
  FocusNode? usuarioFocusNode;
  TextEditingController? usuarioTextController;
  String? Function(BuildContext, String?)?
      usuarioTextControllerValidator;

  // Nombre
  FocusNode? nombreFocusNode;
  TextEditingController? nombreTextController;
  String? Function(BuildContext, String?)?
      nombreTextControllerValidator;

  // Clave
  FocusNode? claveFocusNode;
  TextEditingController? claveTextController;
  late bool claveVisibility;
  String? Function(BuildContext, String?)?
      claveTextControllerValidator;

  // Fecha de nacimiento
  DateTime? datePicked;

  // Sexo (si se usa en la UI)
  String? sexoValue;
  FormFieldController<String>? sexoValueController;

  // Codigo postal (si aplica)
  FocusNode? cpFocusNode;
  TextEditingController? cpTextController;
  String? Function(BuildContext, String?)?
      cpTextControllerValidator;

  // Nombre de granja
  FocusNode? nGranjaFocusNode;
  TextEditingController? nGranjaTextController;
  String? Function(BuildContext, String?)?
      nGranjaTextControllerValidator;

  // Nombre de sensor
  FocusNode? nSensorFocusNode;
  TextEditingController? nSensorTextController;
  String? Function(BuildContext, String?)?
      nSensorTextControllerValidator;

  // Direccion de granja
  FocusNode? ubicacionFocusNode;
```

```
46    TextEditingController? ubicacionTextController;
47    String? Function(BuildContext, String?)?
          ubicacionTextControllerValidator;
48
49    @override
50    void initState(BuildContext context) {
51      claveVisibility = false;
52    }
53
54    @override
55    void dispose() {
56      usuarioFocusNode?.dispose();
57      usuarioTextController?.dispose();
58
59      nombreFocusNode?.dispose();
60      nombreTextController?.dispose();
61
62      claveFocusNode?.dispose();
63      claveTextController?.dispose();
64
65      cpFocusNode?.dispose();
66      cpTextController?.dispose();
67
68      nGranjaFocusNode?.dispose();
69      nGranjaTextController?.dispose();
70
71      nSensorFocusNode?.dispose();
72      nSensorTextController?.dispose();
73
74      ubicacionFocusNode?.dispose();
75      ubicacionTextController?.dispose();
76    }
77  }
```

3.4.5. Aplicación móvil: recuperación de contraseña (`ForgotpasswordWidget`

La pantalla `ForgotpasswordWidget` permite al usuario solicitar el envió de un correo de restablecimiento de contraseña a través del servicio de autenticación. Sus responsabilidades son:

- Capturar el email del usuario.
- Invocar `authManager.resetPassword` para enviar el correo de recuperación.
- Informar de errores básicos (campo vació) y volver a `LoginWidget` tras la solicitud.

3.4.5.1. Imports principales

```
import '/auth/firebase_auth/auth_util.dart';
import '/flutter_flow/flutter_flow_icon_button.dart';
import '/flutter_flow/flutter_flow_theme.dart';
import '/flutter_flow/flutter_flow_util.dart';
import '/flutter_flow/flutter_flow_widgets.dart';
import '/index.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:provider/provider.dart';
import 'forgotpassword_model.dart';
export 'forgotpassword_model.dart';
```

3.4.5.2. Estructura y estado

```
class ForgotpasswordWidget extends StatefulWidget {
  const ForgotpasswordWidget({super.key});
  static String routeName = 'forgotpassword';
  static String routePath = '/forgotpassword';
  @override
  State<ForgotpasswordWidget> createState() =>
      _ForgotpasswordWidgetState();
}

class _ForgotpasswordState extends State<ForgotpasswordWidget> {
  late ForgotpasswordModel _model;
  final scaffoldKey = GlobalKey<ScaffoldState>();

  @override
  void initState() {
    super.initState();
    _model = createModel(context, () => ForgotpasswordModel());
    _model.emailTextController ??= TextEditingController();
    _model.textFieldFocusNode ??= FocusNode();
  }

  @override
  void dispose() {
    _model.dispose();
    super.dispose();
  }
}
```

3.4.5.3. Interfaz: campo de email

```
TextFormField(
```

```
  controller: _model.emailTextController,
  focusNode: _model.textFieldFocusNode,
  autofocus: true,
  decoration: InputDecoration(
    labelText: 'Email',
    hintText: 'Introduzca su email',
    enabledBorder: OutlineInputBorder(
      borderSide: BorderSide(color: FlutterFlowTheme.of(context)
          .primaryText, width: 1),
      borderRadius: BorderRadius.only(topLeft: Radius.circular
          (4), topRight: Radius.circular(4)),
    ),
  ),
  validator: _model.emailTextControllerValidator.asValidator(
      context),
)
```

3.4.5.4. Acción: enviar correo de restablecimiento

Valida que el email no este vació; si lo esta, muestra una `SnackBar`. En caso contrario, invoca el flujo de recuperación y navega de vuelta al login.

```
FFButtonWidget(
  onPressed: () async {
    if (_model.emailTextController.text.isEmpty) {
      ScaffoldMessenger.of(context).showSnackBar(const SnackBar(
        content: Text('Email required!'),
      ));
      return;
    }
    await authManager.resetPassword(
      email: _model.emailTextController.text,
      context: context,
    );
    context.pushNamed(LoginWidget.routeName);
  },
  text: 'Mandar correo',
)
```

3.4.5.5. Composición visual

La pantalla mantiene la coherencia visual con el tema de la aplicación y ofrece navegación hacia atrás mediante el icono de la *AppBar*.

```
Scaffold(
  key: scaffoldKey,
  backgroundColor: FlutterFlowTheme.of(context).
      primaryBackground,
```

```
    appBar: AppBar(
      backgroundColor: FlutterFlowTheme.of(context).primary,
      leading: FlutterFlowIconButton(
        icon: const Icon(Icons.arrow_back),
        onPressed: () async { context.pushNamed(LoginWidget.
            routeName); },
      ),
      title: Text('Olvido la contrasena'),
      centerTitle: true,
    ),
    body: SafeArea(
      child: Padding(
        padding: const EdgeInsetsDirectional.fromSTEB(20, 20, 30,
            0),
        child: Column(children: [
          // Campo de email (bloque anterior)
          // Boton "Mandar correo" (bloque anterior)
        ]),
      ),
    ),
)
```

3.4.5.6. Modelo asociado (`ForgotpasswordModel`)

Modelo minimal para la pantalla de recuperación: controlador y foco del campo de email.

```
import '/flutter_flow/flutter_flow_util.dart';
import 'forgotpassword_widget.dart' show ForgotpasswordWidget;
import 'package:flutter/material.dart';

class ForgotpasswordModel extends FlutterFlowModel<
    ForgotpasswordWidget> {
  // Email field
  FocusNode? textFieldFocusNode;
  TextEditingController? emailTextController;
  String? Function(BuildContext, String?)?
      emailTextControllerValidator;

  @override
  void initState(BuildContext context) {}

  @override
  void dispose() {
    textFieldFocusNode?.dispose();
    emailTextController?.dispose();
  }
}
```

3.4.6. Aplicación móvil: administración de perfil (`AdministrarPerfilWidget`)

La pantalla `AdministrarPerfilWidget` permite al usuario ver y actualizar su foto de perfil de manera local, ademas de acceder a acciones habituales de cuenta. Las principales responsabilidades son:

- Cargar y persistir localmente una fotografía de perfil (sin subirla al servidor).
- Mostrar el avatar según prioridad: foto local → `photoURL` de Firebase → imagen *asset*.
- Ofrecer accesos a modificar perfil, preferencias, cambio de contraseña y cierre de sesión.
- Gestionar estados de carga y feedback mediante `SnackBar` y un overlay con `CircularProgressIndicator`.

3.4.6.1. Imports principales

```dart
// administrar_perfil_widget.dart
import 'dart:io';
import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';
import 'package:path_provider/path_provider.dart';
import 'package:path/path.dart' as p;
import 'package:shared_preferences/shared_preferences.dart';
import 'package:firebase_auth/firebase_auth.dart';
```

3.4.6.2. Estructura y estado

El estado mantiene el usuario autenticado, un flag de guardado y la ruta local de la foto persistida en `SharedPreferences`.

```dart
class AdministrarPerfilWidget extends StatefulWidget {
  const AdministrarPerfilWidget({super.key});
  static const String routeName = 'administrarPerfil';
  static const String routePath  = '/administrar-perfil';
  @override
  State<AdministrarPerfilWidget> createState() =>
      _AdministrarPerfilWidgetState();
}

class _AdministrarPerfilWidgetState extends State<
    AdministrarPerfilWidget> {
  final _user   = FirebaseAuth.instance.currentUser!;
  bool  _saving = false;
  String? _localPhotoPath; // ruta local persistida
```

```
13
14    @override
15    void initState() {
16      super.initState();
17      _cargarFotoGuardada();
18    }
19  }
```

3.4.6.3. Carga y persistencia de la foto

Se lee la ruta de la foto desde `SharedPreferences`. Al cambiarla, se copia la imagen al directorio de documentos de la app y se guarda su ruta.

```
1  // Cargar ruta almacenada
2  Future<void> _cargarFotoGuardada() async {
3    final prefs = await SharedPreferences.getInstance();
4    setState(() => _localPhotoPath = prefs.getString('
       perfil_foto_local'));
5  }
6
7  // Cambiar foto (local)
8  Future<void> _cambiarFoto() async {
9    final picker = ImagePicker();
10   final XFile? file = await picker.pickImage(
11     source: ImageSource.gallery,
12     imageQuality: 85,
13   );
14   if (file == null) return;
15
16   setState(() => _saving = true);
17   try {
18     final dir       = await getApplicationDocumentsDirectory();
19     final filename  = 'perfil_${DateTime.now().
         millisecondsSinceEpoch}${p.extension(file.path)}';
20     final savedPath = p.join(dir.path, filename);
21
22     await File(file.path).copy(savedPath);
23
24     final prefs = await SharedPreferences.getInstance();
25     await prefs.setString('perfil_foto_local', savedPath);
26
27     setState(() => _localPhotoPath = savedPath);
28
29     if (mounted) {
30       ScaffoldMessenger.of(context).showSnackBar(
31         const SnackBar(content: Text('Foto actualizada')),
32       );
33     }
34   } catch (e) {
```

```
35      if (mounted) {
36        ScaffoldMessenger.of(context).showSnackBar(
37          SnackBar(content: Text('Error al guardar imagen: $e')),
38        );
39      }
40    } finally {
41      if (mounted) setState(() => _saving = false);
42    }
43  }
```

3.4.6.4. Seleccion de avatar y UI

La imagen de perfil se resuelve en este orden: archivo local → URL remota → *asset* por defecto.

```
1   @override
2   Widget build(BuildContext context) {
3     final ImageProvider avatarProvider;
4     if (_localPhotoPath != null) {
5       avatarProvider = FileImage(File(_localPhotoPath!));
6     } else if (_user.photoURL != null) {
7       avatarProvider = NetworkImage(_user.photoURL!);
8     } else {
9       avatarProvider = const AssetImage('assets/images/favicon.png
          ');
10    }
11
12    return Scaffold(
13      appBar: AppBar(title: Text(_user.displayName ?? 'Mi perfil')
          ),
14      body: Stack(
15        children: [
16          ListView(
17            padding: const EdgeInsets.all(20),
18            children: [
19              Center(
20                child: Stack(
21                  children: [
22                    CircleAvatar(radius: 60, backgroundImage:
                        avatarProvider),
23                    Positioned(
24                      bottom: 0, right: 0,
25                      child: IconButton(
26                        icon: const Icon(Icons.camera_alt, color:
                            Colors.white),
27                        onPressed: _saving ? null : _cambiarFoto,
28                      ),
29                    ),
30                  ],
```

```
31              ),
32            ),
33            const SizedBox(height: 16),
34            Center(
35              child: Text(
36                _user.displayName?.isNotEmpty == true ? _user.
                     displayName! : _user.email ?? '',
37                style: const TextStyle(fontSize: 20, fontWeight:
                     FontWeight.bold),
38              ),
39            ),
40            const SizedBox(height: 32),
41
42            ListTile(
43              leading: const Icon(Icons.edit),
44              title: const Text('Modificar perfil'),
45              onTap: () => Navigator.pushNamed(context, '
                   modificarPerfil'),
46            ),
47            ListTile(
48              leading: const Icon(Icons.settings),
49              title: const Text('Preferencias'),
50              onTap: () => Navigator.pushNamed(context, '
                   preferencias'),
51            ),
52            ListTile(
53              leading: const Icon(Icons.lock_reset),
54              title: const Text('Cambiar contrasena'),
55              onTap: () => Navigator.pushNamed(context, '
                   forgotpassword'),
56            ),
57            ListTile(
58              leading: const Icon(Icons.logout),
59              title: const Text('Cerrar sesion'),
60              onTap: () async {
61                await FirebaseAuth.instance.signOut();
62                if (mounted) {
63                  Navigator.pushNamedAndRemoveUntil(context, '
                       login', (_) => false);
64                }
65              },
66            ),
67          ],
68        ),
69
70        if (_saving)
71          Container(
72            color: Colors.black45,
73            child: const Center(child: CircularProgressIndicator
                 ()),
```

```
            ),
        ],
      ),
    );
}
```

3.4.7. Aplicación móvil: Archivo principal (`main.dart`)

Este fichero inicializa los servicios base (notificaciones locales, permisos en Android 13+, Firebase), carga el tema y el estado persistente de la aplicacion, y arranca el arbol principal con `Provider` y el enrutador (`GoRouter`) de FlutterFlow.

3.4.7.1. Notificaciones locales

Se crea una instancia global del *plugin* y se inicializa con la configuracion de Android. Esto permite que otras pantallas (p.ej., Medidas) disparen avisos locales.

```
import 'package:flutter_local_notifications/
    flutter_local_notifications.dart';

final FlutterLocalNotificationsPlugin
    flutterLocalNotificationsPlugin =
     FlutterLocalNotificationsPlugin();

Future<void> _initLocalNotifications() async {
  const androidInit = AndroidInitializationSettings('@mipmap/
      ic_launcher');
  const initSettings = InitializationSettings(android:
      androidInit);
  await flutterLocalNotificationsPlugin.initialize(initSettings)
      ;
}
```

3.4.7.2. Funcion `main`: orden de arranque

El arranque sigue este orden: vincula Flutter, inicia notificaciones, pide permiso (Android 13+), inicializa Firebase, aplica configuracion de FlutterFlow, hidrata el estado global y finalmente ejecuta la app con `Provider`.

```
Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();

  await _initLocalNotifications();

  // Android 13+ permiso runtime para notificaciones
```

```
  if (Platform.isAndroid) {
    final status = await Permission.notification.status;
    if (!status.isGranted) await Permission.notification.request
       ();
  }

  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );

  await initFirebase();                  // configuracion extra (
     FlutterFlow)
  await FlutterFlowTheme.initialize();  // carga modo claro/
     oscuro guardado

  final appState = FFAppState();
  await appState.initializePersistedState(); // hidrata estado (
     SharedPrefs)

  runApp(ChangeNotifierProvider(
    create: (_) => appState,
    child: const MyApp(),
  ));
}
```

3.4.7.3. Raiz de la app: `MyApp`

`MyApp` mantiene el `ThemeMode`, prepara el enrutador y escucha el flujo de usuario de Firebase para reaccionar a cambios de sesion (mostrar splash, redirigir, etc.).

```
class MyApp extends StatefulWidget {
  const MyApp({super.key});
  static _MyAppState of(BuildContext context) =>
      context.findAncestorStateOfType<_MyAppState>()!;
  @override
  State<MyApp> createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
  ThemeMode _themeMode = FlutterFlowTheme.themeMode;
  late final AppStateNotifier _appStateNotifier;
  late final GoRouter _router;

  // Metodos que usa flutter_flow_util.dart
  String getRoute([RouteMatch? match]) {
    final RouteMatch last =
        match ?? _router.routerDelegate.currentConfiguration.
           last;
    final RouteMatchList list = last is ImperativeRouteMatch
```

```
19          ? last.matches
20          : _router.routerDelegate.currentConfiguration;
21      return list.uri.toString();
22    }
23    List<String> getRouteStack() => _router
24        .routerDelegate.currentConfiguration.matches
25        .map((m) => getRoute(m))
26        .toList();
27
28    @override
29    void initState() {
30      super.initState();
31      _appStateNotifier = AppStateNotifier.instance;
32      _router = createRouter(_appStateNotifier);
33
34      ithappFirebaseUserStream().listen(_appStateNotifier.update);
35      jwtTokenStream.listen((_) {});
36
37      Future.delayed(
38        const Duration(milliseconds: 1000),
39        _appStateNotifier.stopShowingSplashImage,
40      );
41    }
42
43    void setThemeMode(ThemeMode mode) {
44      setState(() => _themeMode = mode);
45      FlutterFlowTheme.saveThemeMode(mode);
46    }
47
48    @override
49    Widget build(BuildContext context) {
50      return MaterialApp.router(
51        debugShowCheckedModeBanner: false,
52        title: 'ithapp',
53        localizationsDelegates: const [
54          GlobalMaterialLocalizations.delegate,
55          GlobalWidgetsLocalizations.delegate,
56          GlobalCupertinoLocalizations.delegate,
57        ],
58        supportedLocales: const [Locale('en', '')],
59        theme: ThemeData(brightness: Brightness.light,
             useMaterial3: false),
60        darkTheme: ThemeData(brightness: Brightness.dark,
             useMaterial3: false),
61        themeMode: _themeMode,
62        routerConfig: _router,
63      );
64    }
65  }
```

El fichero nav.dart define el enrutador central de la app con GoRouter. Mantiene el estado de sesión y splash mediante AppStateNotifier, protege rutas que requieren autenticación (redirigiendo a /login si hace falta), unifica transiciones entre pantallas y ofrece atajos de navegación seguros para usar desde los widgets.

```
// --- Estado global para navegacion y sesion ---
class AppStateNotifier extends ChangeNotifier {
  AppStateNotifier._();
  static AppStateNotifier? _instance;
  static AppStateNotifier get instance => _instance ??= AppStateNotifier._();

  BaseAuthUser? initialUser;
  BaseAuthUser? user;
  bool showSplashImage = true;          // mientras inicia (loader)
  String? _redirectLocation;            // ruta pendiente tras login
  bool notifyOnAuthChange = true;       // evita refrescos durante navegacion

  bool get loading => user == null || showSplashImage;
  bool get loggedIn => user?.loggedIn ?? false;
  bool get shouldRedirect => loggedIn && _redirectLocation != null;

  void setRedirectLocationIfUnset(String loc) => _redirectLocation ??= loc;
  String getRedirectLocation() => _redirectLocation!;
  void clearRedirectLocation() => _redirectLocation = null;
  void updateNotifyOnAuthChange(bool notify) => notifyOnAuthChange = notify;

  // Se invoca cuando cambia el usuario de Firebase
  void update(BaseAuthUser newUser) {
    final changed = user?.uid == null || newUser.uid == null || user?.uid != newUser.uid;
    initialUser ??= newUser;
    user = newUser;
    if (notifyOnAuthChange && changed) notifyListeners();
    updateNotifyOnAuthChange(true);
  }

  void stopShowingSplashImage() { showSplashImage = false; notifyListeners(); }
}

// --- Creacion del router: mapea pantallas y protege rutas ---
GlobalKey<NavigatorState> appNavigatorKey = GlobalKey<NavigatorState>();

```

```
GoRouter createRouter(AppStateNotifier app) => GoRouter(
  initialLocation: '/',
  refreshListenable: app,         // se rehace cuando cambia sesion/splash
  navigatorKey: appNavigatorKey,
  errorBuilder: (context, _) => app.loggedIn ? MedidasWidget() : LoginWidget(),
  routes: [
    // ruta inicial: segun sesion va a Medidas o Login
    FFRoute(name: '_initialize', path: '/', builder: (c, _) =>
      app.loggedIn ? MedidasWidget() : LoginWidget()),

    // pantallas publicas
    FFRoute(name: LoginWidget.routeName,         path: LoginWidget.routePath,
      builder: (c, p) => LoginWidget()),
    FFRoute(name: RegistroWidget.routeName,      path: RegistroWidget.routePath,
      builder: (c, p) => RegistroWidget()),
    FFRoute(name: ForgotpasswordWidget.routeName, path: ForgotpasswordWidget.routePath,
      builder: (c, p) => ForgotpasswordWidget()),

    // pantallas protegidas (requireAuth: true)
    FFRoute(name: MedidasWidget.routeName,       path: MedidasWidget.routePath,
      builder: (c, p) => MedidasWidget(), requireAuth: true),
    FFRoute(name: AdministrarPerfilWidget.routeName, path: AdministrarPerfilWidget.routePath,
      builder: (c, p) => const AdministrarPerfilWidget(),
        requireAuth: true),
    FFRoute(name: ModificarPerfilWidget.routeName,   path: ModificarPerfilWidget.routePath,
      builder: (c, p) => const ModificarPerfilWidget(),
        requireAuth: true),
  ].map((r) => r.toRoute(app)).toList(),
);

// --- Envoltorio de ruta: control de acceso + transiciones + loader ---
class FFRoute {
  const FFRoute({
    required this.name,
    required this.path,
    required this.builder,
    this.requireAuth = false,
    this.asyncParams = const {},
    this.routes = const [],
  });

```

```
76    final String name, path;
77    final bool requireAuth;
78    final Map<String, Future<dynamic> Function(String)>
          asyncParams;
79    final Widget Function(BuildContext, FFParameters) builder;
80    final List<GoRoute> routes;
81
82    GoRoute toRoute(AppStateNotifier app) => GoRoute(
83      name: name,
84      path: path,
85      // Redireccion por autenticacion y "volver a donde iba"
86      redirect: (context, state) {
87        if (app.shouldRedirect) {
88          final loc = app.getRedirectLocation();
89          app.clearRedirectLocation();
90          return loc;
91        }
92        if (requireAuth && !app.loggedIn) {
93          app.setRedirectLocationIfUnset(state.uri.toString());
94          return '/login';
95        }
96        return null;
97      },
98      // PageBuilder: muestra loader durante splash/carga y aplica
            transiciones
99      pageBuilder: (context, state) {
100       final ffParams = FFParameters(state, asyncParams);
101       final page = ffParams.hasFutures
102         ? FutureBuilder(future: ffParams.completeFutures(),
103             builder: (c, _) => builder(c, ffParams))
104         : builder(context, ffParams);
105
106       final child = app.loading
107         ? Center(child: SizedBox(width: 50, height: 50,
108             child: CircularProgressIndicator(
109               valueColor: AlwaysStoppedAnimation<Color>(
110                 FlutterFlowTheme.of(context).primary))))
111         : page;
112
113       final t = state.transitionInfo;
114       return t.hasTransition
115         ? CustomTransitionPage(
116             key: state.pageKey, child: child, transitionDuration
                    : t.duration,
117             transitionsBuilder: (c, a, s, ch) => PageTransition(
118               type: t.transitionType, duration: t.duration,
119               reverseDuration: t.duration, alignment: t.
                      alignment, child: ch,
120             ).buildTransitions(c, a, s, ch),
121           )
```

```
122           : MaterialPage(key: state.pageKey, child: child);
123     },
124     routes: routes,
125   );
126 }
127
128 // --- Atajos de navegacion seguros para usar en widgets ---
129 extension NavigationExtensions on BuildContext {
130   void goNamedAuth(String name, bool mounted, { Map<String,
          String> pathParameters = const {},
131     Map<String,String> queryParameters = const {}, Object? extra
          , bool ignoreRedirect = false }) =>
132     !mounted || GoRouter.of(this).shouldRedirect(ignoreRedirect)
133       ? null
134       : goNamed(name, pathParameters: pathParameters,
135           queryParameters: queryParameters, extra: extra);
136
137   void pushNamedAuth(String name, bool mounted, { ... }) { /*
          idem */ }
138
139   void safePop() { if (canPop()) pop(); else go('/'); }
140 }
```

flutter_flow_util.dart concentra utilidades compartidas por toda la app: formateo de fechas, helpers de geolocalización con permisos, funciones de UI como showSnackbar, extensiones para construir layouts (.divide(...)), un safeSetState para evitar errores de estado, y un fix para el color de la barra de estado en iOS 16. También expone funciones para consultar la ruta actual (usadas por main.dart).

```
1  // Utilidad generica: usa valor por defecto si viene null o
       string vacio
2  T valueOrDefault<T>(T? value, T defaultValue) =>
3    (value is String && value.isEmpty) || value == null ?
         defaultValue : value;
4
5  // Formateo de fechas (incluye modo "relative" tipo "hace 3 min
       ")
6  String dateTimeFormat(String format, DateTime? dateTime, {String
       ? locale}) {
7    if (dateTime == null) return '';
8    if (format == 'relative') {
9      return timeago.format(dateTime, locale: locale, allowFromNow
           : true);
10   }
11   return DateFormat(format, locale).format(dateTime);
12 }
13
14 // Marcas de tiempo comodas
15 DateTime get getCurrentTimestamp => DateTime.now();
16 DateTime dateTimeFromSecondsSinceEpoch(int seconds) =>
```

```
17    DateTime.fromMillisecondsSinceEpoch(seconds * 1000);
18  extension DateTimeConversionExtension on DateTime {
19    int get secondsSinceEpoch => (millisecondsSinceEpoch / 1000).
          round();
20  }
21
22  // ---------------- Geolocalizacion (con permisos)
        ----------------
23  LatLng? cachedUserLocation;
24
25  // Devuelve ubicacion actual (o default) y cachea si se pide
26  Future<LatLng> getCurrentUserLocation(
27    {required LatLng defaultLocation, bool cached = false}) async
          {
28    if (cached && cachedUserLocation != null) return
          cachedUserLocation!;
29    return queryCurrentUserLocation().then((loc) {
30      if (loc != null) cachedUserLocation = loc;
31      return loc ?? defaultLocation;
32    }).onError((error, _) {
33      print("Error querying user location: $error");
34      return defaultLocation;
35    });
36  }
37
38  // Pide permisos y obtiene coordenadas reales del dispositivo
39  Future<LatLng?> queryCurrentUserLocation() async {
40    final serviceEnabled = await Geolocator.
          isLocationServiceEnabled();
41    if (!serviceEnabled) return Future.error('Location services
          are disabled.');
42
43    var permission = await Geolocator.checkPermission();
44    if (permission == LocationPermission.denied) {
45      permission = await Geolocator.requestPermission();
46      if (permission == LocationPermission.denied) {
47        return Future.error('Location permissions are denied');
48      }
49    }
50    if (permission == LocationPermission.deniedForever) {
51      return Future.error(
52        'Location permissions are permanently denied, we cannot
              request permissions.');
53    }
54
55    final position = await Geolocator.getCurrentPosition();
56    return position != null && position.latitude != 0 && position.
          longitude != 0
57      ? LatLng(position.latitude, position.longitude)
58      : null;
```

```
59 }
60
61 // ---------------- Helpers de UI y estado ----------------
62
63 // Mostrar snackbar con opcion de spinner "loading"
64 void showSnackbar(BuildContext context, String message,
65   {bool loading = false, int duration = 4}) {
66   ScaffoldMessenger.of(context).hideCurrentSnackBar();
67   ScaffoldMessenger.of(context).showSnackBar(
68     SnackBar(
69       content: Row(
70         children: [
71           if (loading)
72             Padding(
73               padding: EdgeInsetsDirectional.only(end: 10.0),
74               child: SizedBox(
75                 height: 20, width: 20,
76                 child: const CircularProgressIndicator(color:
                      Colors.white),
77               ),
78             ),
79           Text(message),
80         ],
81       ),
82       duration: Duration(seconds: duration),
83     ),
84   );
85 }
86
87 // setState seguro: solo si el widget sigue montado
88 extension StatefulWidgetExtensions on State<StatefulWidget> {
89   void safeSetState(VoidCallback fn) {
90     if (mounted) setState(fn);
91   }
92 }
93
94 // Extensiones para maquetar listas de widgets:
95 // .divide agrega un separador entre elementos (usado mucho en
      la UI)
96 extension ListDivideExt<T extends Widget> on Iterable<T> {
97   Iterable<MapEntry<int, Widget>> get enumerate => toList().
       asMap().entries;
98
99   List<Widget> divide(Widget t, {bool Function(int)? filterFn})
       => isEmpty
100     ? []
101     : (enumerate
102         .map((e) => [e.value, if (filterFn == null || filterFn(e
              .key)) t])
103         .expand((i) => i).toList()
```

```
104       ..removeLast());
105
106   List<Widget> around(Widget t) => addToStart(t).addToEnd(t);
107   List<Widget> addToStart(Widget t) => enumerate.map((e) => e.
          value).toList()..insert(0, t);
108   List<Widget> addToEnd(Widget t) => enumerate.map((e) => e.
          value).toList()..add(t);
109 }
110
111 // Fix iOS <= 16: alinea el color de la status bar con el tema
       de la app
112 Brightness? _lastBrightness;
113 void fixStatusBarOniOS16AndBelow(BuildContext context) {
114   if (!isiOS) return;
115   final brightness = Theme.of(context).brightness;
116   if (_lastBrightness != brightness) {
117     _lastBrightness = brightness;
118     SystemChrome.setSystemUIOverlayStyle(
119       SystemUiOverlayStyle(
120         statusBarBrightness: brightness,
121         systemStatusBarContrastEnforced: true,
122       ),
123     );
124   }
125 }
126
127 // Color util: aplicar alpha como porcentaje
128 extension ColorOpacityExt on Color {
129   Color applyAlpha(double val) => withValues(alpha: val);
130 }
131
132 // Rutas actuales (usado por main.dart para debugging/telemetria
       )
133 String getCurrentRoute(BuildContext context) =>
134   context.mounted ? MyApp.of(context).getRoute() : '';
135 List<String> getCurrentRouteStack(BuildContext context) =>
136   context.mounted ? MyApp.of(context).getRouteStack() : [];
```

3.5. Sistema de almacenamiento de datos

Este subsistema persiste y sirve las mediciones en una base de datos MySQL gestionada por Railway. El backend, desarrollado con Node.js y Express, utiliza `mysql2/promise` con *pool* de conexiones, expone endpoints REST para altas de entidades (granjas, usuarios, sensores), recibe mediciones desde TTN a través de un webhook y publica un endpoint de consulta de la última medición. A continuación se muestran los fragmentos relevantes y su explicación.

3.5.1. Backend (Node.js/Express) conectado a MySQL en Railway

```
// server.js
const express = require("express");
const mysql = require("mysql2/promise");
const cors = require("cors");

const app = express();

// 1) Middleware
app.use(express.json());
app.use(cors({ origin: "*" }));

// 2) Puerto (Railway coloca PORT)
const port = process.env.PORT || 3000;

// 3) Config MySQL (Railway inyecta estas vars al crear la DB)
const pool = mysql.createPool({
  host: process.env.MYSQLHOST || "localhost",
  port: process.env.MYSQLPORT ? Number(process.env.MYSQLPORT) :
     3306,
  user: process.env.MYSQLUSER || "root",
  password: process.env.MYSQLPASSWORD || "",
  database: process.env.MYSQLDATABASE || "ganaderapp",
  waitForConnections: true,
  connectionLimit: 10,
});

let lastDevEui = null;


// 4) Salud
app.get("/health", async (_req, res) => {
  try {
    const conn = await pool.getConnection();
    await conn.ping();
    conn.release();
    res.json({ ok: true });
  } catch (e) {
    console.error(e);
    res.status(500).json({ ok: false, error: "DB ping failed" })
       ;
  }
});

// 5) Endpoints

// TTN webhook     inserta mediciones y cachea el DEV_EUI
app.post("/webhook", async (req, res) => {
```

```
46   try {
47     // cachear dev_eui en memoria.
48     const devEui = req.body?.end_device_ids?.dev_eui
49                 || req.body?.end_device_ids?.device_id
50                 || null;
51     if (devEui) lastDevEui = devEui;
52
53     const uplink = req.body.uplink_message;
54     if (!uplink?.decoded_payload) {
55       return res.status(400).send("Payload invalido");
56     }
57
58     const { temperatura, humedad, ith } = uplink.decoded_payload
          ;
59     await pool.query(
60       "INSERT INTO mediciones (temperatura, humedad, ith) VALUES
             (?, ?, ?)",
61       [temperatura, humedad, ith]
62     );
63
64     console.log("Datos recibidos:", { temperatura, humedad, ith,
           devEui });
65     res.send("OK");
66   } catch (err) {
67     console.error("Error insertando mediciones:", err);
68     res.status(500).send("Error en base de datos");
69   }
70 });
71
72 // Endpoint simple para la app
73 app.get("/dev-eui-ultimo", (_req, res) => {
74   if (!lastDevEui) return res.status(404).json({ error: "
        sin_dev_eui" });
75   res.json({ dev_eui: lastDevEui });
76 });
77
78 app.get('/api/dev-eui-ultimo', async (_req, res) => {
79   const q1 = `
80     SELECT dev_eui
81     FROM sensores
82     WHERE dev_eui IS NOT NULL AND dev_eui <> ''
83     ORDER BY updated_at DESC, id_sensor DESC
84     LIMIT 1`;
85   const q2 = `
86     SELECT dev_eui
87     FROM sensores
88     WHERE dev_eui IS NOT NULL AND dev_eui <> ''
89     ORDER BY id_sensor DESC
90     LIMIT 1`;
91   try {
```

```
92      const [rows] = await pool.query(q1);
93      if (!rows.length) return res.status(404).json({ error: '
            sin_dev_eui' });
94      res.json({ dev_eui: rows[0].dev_eui });
95    } catch (e) {
96      // Si 'updated_at' no existe, probamos sin ella
97      if (e.code === 'ER_BAD_FIELD_ERROR') {
98        const [rows] = await pool.query(q2);
99        if (!rows.length) return res.status(404).json({ error: '
              sin_dev_eui' });
100       return res.json({ dev_eui: rows[0].dev_eui });
101     }
102     console.error(e);
103     res.status(500).json({ error: 'db_error' });
104   }
105 });
106
107
108 // Granja
109 app.post("/granjas", async (req, res) => {
110   try {
111     const { nombre_granja, direccion } = req.body;
112     if (!nombre_granja || !direccion) {
113       return res.status(400).send("Faltan datos de la granja");
114     }
115
116     const [result] = await pool.query(
117       "INSERT INTO granjas (nombre_granja, direccion) VALUES (?,
              ?)",
118       [nombre_granja, direccion]
119     );
120
121     res.status(201).json({ id_granja: result.insertId, mensaje:
            "Granja guardada" });
122   } catch (err) {
123     console.error(" Error insertando granja:", err);
124     res.status(500).send("Error en base de datos");
125   }
126 });
127
128 // Usuario
129 app.post("/usuarios", async (req, res) => {
130   try {
131     const { nombre, apellidos, email, fecha_nacimiento, password
            , id_granja } = req.body;
132     if (!nombre || !email || !fecha_nacimiento || !password || !
            id_granja) {
133       return res.status(400).send("Faltan datos del usuario");
134     }
135
```

```
136     const [result] = await pool.query(
137       `INSERT INTO usuarios (nombre, apellidos, email,
            fecha_nacimiento, password, id_granja)
138        VALUES (?, ?, ?, ?, ?, ?)`,
139       [nombre, apellidos || "", email, fecha_nacimiento,
            password, id_granja]
140     );
141
142     res.status(201).json({ id_usuario: result.insertId, mensaje:
            "Usuario guardado" });
143   } catch (err) {
144     console.error("Error insertando usuario:", err);
145     res.status(500).send("Error en base de datos");
146   }
147 });
148
149 // Sensor
150 app.post("/sensores", async (req, res) => {
151   try {
152     const { nombre_sensor, id_granja } = req.body;
153     if (!nombre_sensor || !id_granja) {
154       return res.status(400).send("Faltan datos del sensor");
155     }
156
157     const [result] = await pool.query(
158       "INSERT INTO sensores (nombre_sensor, id_granja) VALUES
            (?, ?)",
159       [nombre_sensor, id_granja]
160     );
161
162     res.status(201).json({ id_sensor: result.insertId, mensaje:
           "Sensor guardado" });
163   } catch (err) {
164     console.error("Error insertando sensor:", err);
165     res.status(500).send("Error en base de datos");
166   }
167 });
168
169 // Ultima medicion
170 app.get("/mediciones", async (_req, res) => {
171   try {
172     const [rows] = await pool.query("SELECT * FROM mediciones
           ORDER BY id DESC LIMIT 1");
173     if (rows.length > 0) return res.json(rows[0]);
174     res.status(404).send("No hay datos disponibles");
175   } catch (err) {
176     console.error("Error al consultar mediciones:", err);
177     res.status(500).send("Error en base de datos");
178   }
179 });
```

```
180
181
182
183 // 6) Iniciar
184 app.listen(port, () => {
185   console.log(`API escuchando en puerto ${port}`);
186 });
```

3.5.2. Esquema de base de datos

```
1  // schema.sql (resumen)
2
3  CREATE TABLE granjas (
4    id_granja INT AUTO_INCREMENT PRIMARY KEY,
5    nombre_granja VARCHAR(100) NOT NULL,
6    direccion VARCHAR(255) NOT NULL,
7    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
8  );
9
10 CREATE TABLE usuarios (
11   id_usuario INT AUTO_INCREMENT PRIMARY KEY,
12   nombre VARCHAR(100) NOT NULL,
13   apellidos VARCHAR(100) DEFAULT '',
14   email VARCHAR(150) NOT NULL UNIQUE,
15   fecha_nacimiento DATE NOT NULL,
16   password VARCHAR(255) NOT NULL,
17   id_granja INT NOT NULL,
18   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
19   CONSTRAINT fk_usuario_granja FOREIGN KEY (id_granja)
       REFERENCES granjas(id_granja)
20 );
21
22 -- Tabla SENSORES ACTUALIZADA: solo se a aden nuevas columnas
23 CREATE TABLE sensores (
24   id_sensor INT AUTO_INCREMENT PRIMARY KEY,
25   nombre_sensor VARCHAR(100) NOT NULL,
26   id_granja INT NOT NULL,
27   dev_eui    VARCHAR(32)  DEFAULT NULL,
28   app_id     VARCHAR(100) DEFAULT NULL,
29   device_id  VARCHAR(100) DEFAULT NULL,
30   modelo     VARCHAR(100) DEFAULT NULL,
31   area       VARCHAR(100) DEFAULT NULL,
32   zona       INT          DEFAULT NULL,
33   sala       VARCHAR(100) DEFAULT NULL,
34   modo       ENUM('auto','manual') DEFAULT NULL,
35   umbral_ith INT         DEFAULT NULL,
36
37   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
38    updated_at TIMESTAMP NULL DEFAULT NULL ON UPDATE
          CURRENT_TIMESTAMP,
39
40    CONSTRAINT fk_sensor_granja FOREIGN KEY (id_granja) REFERENCES
          granjas(id_granja)
41  );
42
43  CREATE TABLE mediciones (
44    id INT AUTO_INCREMENT PRIMARY KEY,
45    id_sensor INT NULL,
46    temperatura DOUBLE,
47    humedad INT,
48    ith DOUBLE,
49    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
50    INDEX idx_mediciones_created_at (created_at),
51    INDEX idx_mediciones_sensor (id_sensor)
52    -- opcional: FK si asocias cada medici n a un sensor
53    -- , CONSTRAINT fk_medicion_sensor FOREIGN KEY (id_sensor)
          REFERENCES sensores(id_sensor)
54  );
```