

# **Especificación del Sistema de Monitoreo de Confort Térmico**

**Versión:** 1.0

**Fecha:** 24 de Diciembre de 2025

**Estado:** Implementado

# 1. Introducción

---

## 1.1. Propósito del Sistema

El propósito del sistema es permitir el monitoreo participativo del confort térmico en espacios educativos (aulas). A través de una aplicación web accesible vía código QR, los ocupantes pueden reportar su sensación térmica en tiempo real. Estos datos se agregan para permitir a los administradores visualizar el estado de confort de las instalaciones y tomar decisiones informadas.

## 1.2. Alcance

El sistema abarca:

- **Interfaz de Usuario (Frontend):** Aplicación web progresiva para votación y gestión de perfil de usuario.
- **Backend as a Service (BaaS):** Gestión de base de datos, autenticación y tiempo real mediante Supabase.
- **Panel de Administración:** Dashboard para visualización de datos, gráficas y rankings.
- **Sistema de Gamificación:** Mecanismo de recompensas (puntos) para incentivar la participación.

### 1.3. Definiciones y Acrónimos

- **Supabase:** Plataforma de backend de código abierto (alternativa a Firebase).
- **RLS (Row Level Security):** Políticas de seguridad a nivel de fila en PostgreSQL.
- **QR:** Código de respuesta rápida utilizado para el acceso rápido a aulas específicas.
- **SMTP:** Protocolo para el envío de correos electrónicos (configurado vía Gmail).

## 2. Arquitectura del Sistema

---

### 2.1. Stack Tecnológico

- **Frontend:**
  - **Lenguajes:** HTML5, CSS3, JavaScript (ES6+).
  - **Framework CSS:** Tailwind CSS (vía CDN).
  - **Librerías:** Chart.js (Gráficas), Supabase JS Client.
- **Backend:**
  - **Base de Datos:** PostgreSQL (alojado en Supabase).
  - **Autenticación:** Supabase Auth.
  - **Tiempo Real:** Supabase Realtime (WebSockets).
- **Infraestructura:**
  - **Hosting:** Netlify (Archivos estáticos).
  - **Servidor de Correo:** Gmail SMTP (vía Google App Passwords).

## 2.2. Flujo de Datos

1. El usuario escanea un QR o entra manualmente.
2. La aplicación valida la ubicación (Aula) y la sesión.
3. El voto se envía a Supabase ( `public.feedback` ).
4. Un *Trigger* de base de datos actualiza automáticamente los puntos del usuario ( `public.perfiles` ).
5. El Dashboard de Administración recibe el nuevo dato en milisegundos vía WebSockets y actualiza las gráficas.

## 3. Actores y Roles

---

### 3.1. Usuario Anónimo

- Puede acceder a la aplicación.
- Puede seleccionar un aula.
- Puede emitir votos de confort.
- **Limitación:** No acumula puntos y no tiene historial persistente entre dispositivos.

### 3.2. Usuario Registrado

- Todas las capacidades del usuario anónimo.
- Tiene credenciales de acceso (Email/Contraseña).
- Acumula puntos por cada voto (+10 pts).
- Aparece en el Ranking de usuarios.
- Puede recuperar su contraseña vía email.

### 3.3. Administrador

- Acceso exclusivo al Dashboard ( `admin.html` ).
- Protegido por contraseña de aplicación (Client-side).
- Visualización global de datos, tendencias por aula y leaderboard.

## 4. Especificación de Requisitos Funcionales

---

### 4.1. Módulo de Autenticación y Usuarios

- **Registro/Login:** Se utiliza autenticación por Email y Contraseña. Se ha desactivado la confirmación de email para agilizar el registro.
- **Recuperación de Contraseña:** Implementado mediante flujo SMTP usando Gmail. El usuario recibe un enlace, hace clic y define una nueva clave.
- **Gestión de Perfil:** Los usuarios logueados pueden cambiar su contraseña desde la interfaz principal.



## 4.2. Gestión de Espacios (Aulas)

- **Selección de Aula:**
  - **Automática:** Vía parámetro URL `?aula=ID` (QR).
  - **Manual:** El usuario introduce el ID numérico.
- **Validación Temporal:** La asignación de aula se guarda en `localStorage` con un **Timeout de 90 minutos**. Pasado este tiempo, el sistema obliga al usuario a reconfirmar su ubicación antes de votar.
- **Validación de Existencia:** El sistema verifica que el ID del aula exista en la base de datos antes de permitir el voto.

### 4.3. Sistema de Votación (Feedback)

- **Escala de Voto:** Escala de 5 puntos según norma ISO 7730 (adaptada):
  - -2: Muy Frío (🧊)
  - -1: Fresco (❄️)
  - 0: Bien (😊)
  - +1: Calor (🔥)
  - +2: Muy Calor (😡)
- **Restricciones:** Un usuario no puede votar si no tiene un aula asignada y vigente.

### 4.4. Sistema de Gamificación

- **Puntos:** Cada voto registrado por un usuario autenticado suma **10 puntos**.
- **Automatización:** La suma se realiza mediante un *Trigger* ( `sumar_puntos_feedback` ) en la base de datos, garantizando integridad y seguridad.
- **Ranking:** Visualización de los usuarios con más puntos en el panel de administración.

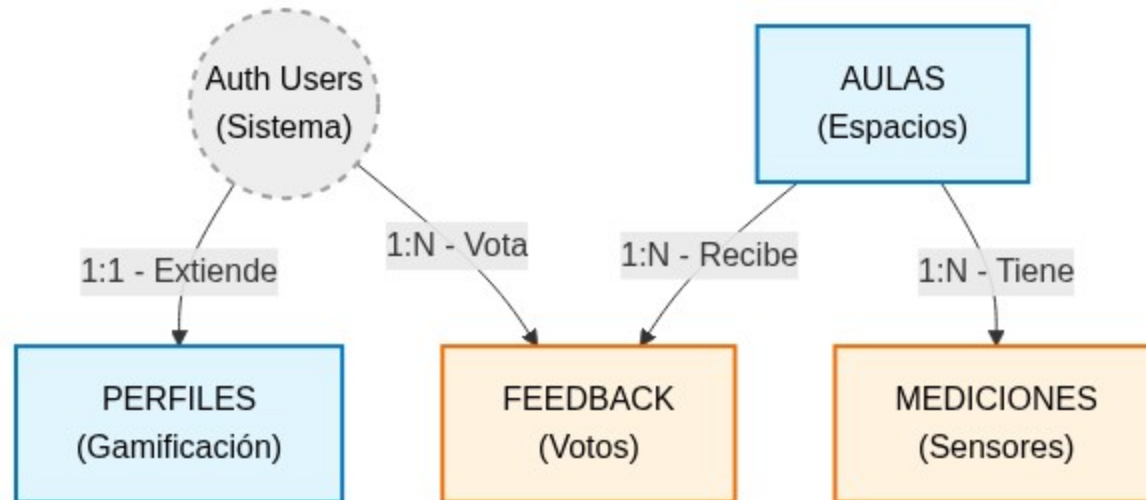
#### 4.5. Panel de Administración (Dashboard)

- **Acceso:** Ruta `/admin.html` , protegida por pantalla de bloqueo con contraseña.
- **Visualización:**
  - **KPIs:** Total votos, Promedio confort.
  - **Distribución:** Gráfico de barras con la cantidad de votos por sensación.
  - **Evolución Temporal:** Gráfico de líneas **multiserie** (una línea por cada aula activa) mostrando la evolución del confort en el tiempo.
  - **Feed en Vivo:** Tabla con los últimos votos recibidos en tiempo real.
  - **Leaderboard:** Tabla de clasificación de usuarios por puntos.

## 5. Modelo de Datos (Base de Datos)

---

### 5.1. Diagrama Entidad-Relación



## 5.2. Tablas Principales

### public.aulas

Catálogo de espacios disponibles.

Campo	Tipo	Descripción
id	bigint (PK)	Identificador único del aula (1-50).
nombre	text	Nombre descriptivo (ej: "Aula 34").
descripcion	text	Detalles adicionales.

### public.perfiles

Información pública de los usuarios (vinculada a auth.users ).

Campo	Tipo	Descripción
id	uuid (PK, FK)	Referencia a auth.users.id .
email	text	Correo electrónico del usuario.
puntos	int	Puntos acumulados (Default: 0).

## **public.feedback**

Registro histórico de votos.

Campo	Tipo	Descripción
id	bigint (PK)	Identificador único del voto.
aula_id	bigint (FK)	Referencia a <code>public.aulas</code> .
user_id	uuid (FK)	Referencia a <code>auth.users</code> (Nullable).
voto	int	Valor del voto (-2 a +2).
session_id	text	ID de sesión para anónimos.
created_at	timestamp	Fecha y hora del voto.

### 5.3. Lógica de Negocio (Base de Datos)

- **Trigger** `on_auth_user_created` : Crea automáticamente una entrada en `public.perfiles` cuando un usuario se registra.
- **Trigger** `on_feedback_insert` : Ejecuta la función `sumar_puntos_feedback()` para añadir 10 puntos al usuario si `user_id` no es nulo.
- **Políticas RLS:**
  - `feedback` : Inserción pública (anon/auth). Lectura restringida (solo admin/dashboard).
  - `perfiles` : Lectura pública habilitada para el Leaderboard. Escritura solo vía Triggers.

## 6. Diseño de Interfaz (UI/UX)

---

### 6.1. Pantalla Principal ( `index.html` )

- **Cabecera:** Estado del usuario (Puntos, Email, Botón Salir/Cambiar Clave) o Botón de Login.
- **Estado Aula:** Muestra el aula actual o alerta "Sin Aula Asignada".
- **Botonera de Voto:** 5 botones grandes con emojis representativos y colores semánticos (Azul a Rojo).
- **Modales:** Ventanas emergentes para Login/Registro, Selección de Aula, Recuperación de Contraseña y Cambio de Contraseña.



## 6.2. Pantalla Admin ( `admin.html` )

- **Bloqueo:** Pantalla inicial solicitando contraseña.
- **Dashboard:** Diseño en rejilla (Grid) responsive.
  - Fila superior: Tarjetas de métricas.
  - Fila central: Gráficos (Chart.js).
  - Fila inferior: Tablas de datos (Feed y Ranking).

## 7. Configuración y Despliegue

---

### 7.1. Configuración SMTP (Gmail)

Para evitar límites de envío, se utiliza un servidor SMTP personalizado:

- **Host:** `smtp.gmail.com`
- **Port:** `587` (SSL OFF / STARTTLS).
- **Auth:** Usuario de Gmail y Contraseña de Aplicación (App Password).

## 7.2. Scripts de Mantenimiento

El sistema cuenta con scripts SQL predefinidos para gestión:

- `reset_full_database.sql` : Reinicio completo de fábrica.
- `add_aulas.sql` : Generación masiva de aulas (1-50).
- `allow_public_profiles.sql` : Configuración de permisos para el ranking.
- `fix_error_voto.sql` : Corrección de permisos para triggers.

## 7.3. Despliegue

- El código fuente se aloja en GitHub.
- Netlify detecta cambios en la rama `main` y despliega automáticamente.
- Archivo `netlify.toml` configura las redirecciones para SPA (Single Page Application).

## 8. Accesos y Credenciales

---

### 8.1. Direcciones Públicas

- Aplicación de Voto (Usuarios): <https://control-termico.netlify.app>
- Panel de Administración (Dashboard): <https://control-termico.netlify.app/admin.html>

### 8.2. Credenciales de Administración

- Contraseña de acceso al Dashboard: `admin1234`
  - Nota: Esta contraseña se valida en el cliente ( `admin.js` ) y sirve para restringir la visualización de datos sensibles.