IES ALBARREGAS

TÉCNICO SUPERIOR EN DESARROLLO DE APLICACIONES WEB

Departamento de Informática

Integración Web para la gestión de máquinas de climatización para AirMagic

MANUAL TÉCNICO

Autor: Santiago Granado Piñero

Curso Académico: DAW 2 A

Índice:

1. Descripción y Alcance

AirMagic Web es la plataforma online que conecta directamente con las instalaciones de climatización de AirMagic. Provee:

- Monitorización y control remoto de máquinas Eco y Ext.
- Registro en tiempo real de temperatura y humedad exterior, y temperatura de expulsión.
- Historial cronológico de alarmas y avisos.
- Gestión de perfiles de usuario (no incluye registro en la web).

Desde la aplicación web puedes dar de alta a las compañías que integrarán esta herramienta. Para poder añadir máquinas y eliminar usuarios se hará desde la aplicación de escritorio para mayor seguridad, además, las máquinas se añaden automáticamente al conectar y añadir las máquinas al ecosistema.

Esta herramienta permitirá a AirMagic ofrecer a sus clientes una solución centralizada donde podrá ver el estado de sus máquinas y gestionarlas desde cualquier parte del mundo.

---

2. Stack Tecnológico

- **Frontend**
  - React 18 + Vite
  - Tailwind CSS para estilos utilitarios
  - React Router para navegación
  - React Icons, React Toastify
- **Backend-as-a-Service**
  - Supabase:
    - Auth (gestión de sesiones y roles)
    - Postgres (base de datos relacional)
    - Storage (avatares y tutoriales)
- **Herramientas de desarrollo**
  - ESLint con configuración custom (eslint.config.js)
  - Prettier para formateo de código
  - Node.js (scripts auxiliares)

---

3. Configuración del Entorno

Variables de entorno (.env / .env.local):

VITE_SUPABASE_URL=your-supabase-url
VITE_SUPABASE_ANON_KEY=your-anon-public-key

- **package.json**:

Dependencias principales:

```
{
  "dependencies": {
    "@supabase/supabase-js": "...",
    "react": "...",
    "react-router-dom": "...",
    "react-icons": "...",
    "react-toastify": "...",
    "tailwindcss": "..."
  },
  "devDependencies": {
    "vite": "...",
    "eslint": "...",
    "prettier": "..."
  },
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview",
    "lint": "eslint . --ext .js,.jsx"
  }
}
```

    -

- **Vite** (vite.config.js): configuración para React y Tailwind.

---

4. Estructura del Proyecto

```
/
├─ public/
│   ├─ img/tutorial.png
│   └─ index.html
├─ src/
│   ├─ components/
│   │   Navbar.jsx
│   │   HistoryModal.jsx
│   │   MachineCard.jsx
│   │   ModalMachine.jsx
│   │   EditProfileModal.jsx
│   │   PasswordChangeModal.jsx
│   │   LoadingSpinner.jsx
│   ├─ context/
│   │   AuthContext.jsx
│   ├─ hooks/
│   │   useAuth.jsx
│   │   useUsers.jsx
│   │   useZones.jsx
│   │   useMachines.jsx
│   │   useProfile.jsx
│   ├─ pages/
│   │   AdminPanel.jsx
│   │   Dashboard.jsx
│   │   Profile.jsx
│   │   ErrorPage.jsx
│   ├─ services/
│   │   authService.js
│   │   userService.js
│   │   zoneService.js
│   │   machineService.js
│   ├─ App.jsx
│   ├─ main.jsx
│   ├─ supabase.js
│   └─ index.css
├─ create-admin.js
├─ eslint.config.js
├─ vite.config.js
├─ package.json
```

```
├─ README.md
└─ .env
```

---

5. Esquema de la Base de Datos

1. Usuarios ← (1:N) Zonas

   - Un usuario puede tener varias zonas.
   - Relación: Zonas.usuario_id → Usuarios.id.

2. Zonas ← (1:N) MaquinasEco

   - Cada zona contiene cero o más máquinas Eco.
   - Relación: MaquinasEco.zona_id → Zonas.id.

3. Zonas ← (1:N) MaquinasExt

   - Cada zona contiene cero o más máquinas Ext.
   - Relación: MaquinasExt.zona_id → Zonas.id.

4. Usuarios ← (1:N) MaquinasEco

   - Un usuario es responsable/propietario de varias máquinas Eco.
   - Relación: MaquinasEco.usuario_id → Usuarios.id.

5. Usuarios ← (1:N) MaquinasExt

   - Similar a Eco: MaquinasExt.usuario_id → Usuarios.id.

6. Usuarios ← (1:N) SondaExterior

   - Un usuario puede tener zero o más sondas exteriores registradas.
   - Relación: SondaExterior.usuario_id → Usuarios.id.

7. Usuarios ← (1:N) HistorialAlarmas

   - Un usuario recibe/genera múltiples alarmas en su historial.
   - Relación: HistorialAlarmas.usuario_id → Usuarios.id.

8. Usuarios ← (1:N) HistorialAvisos

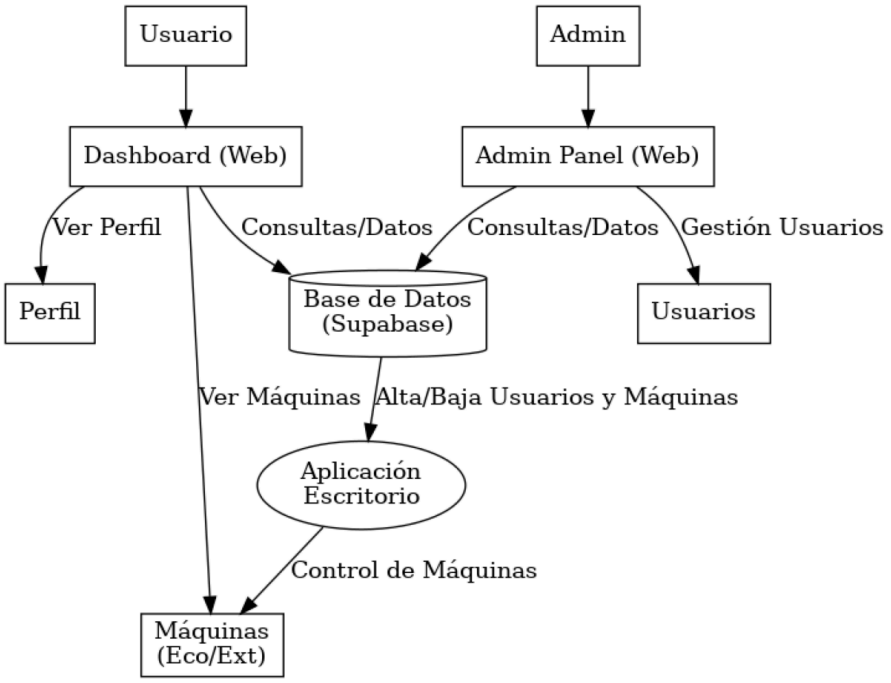
- Un usuario recibe/genera múltiples avisos en su historial.
- Relación: HistorialAvisos.usuario_id → Usuarios.id.

9. MaquinasEco ← (0:N) HistorialAlarmas / HistorialAvisos

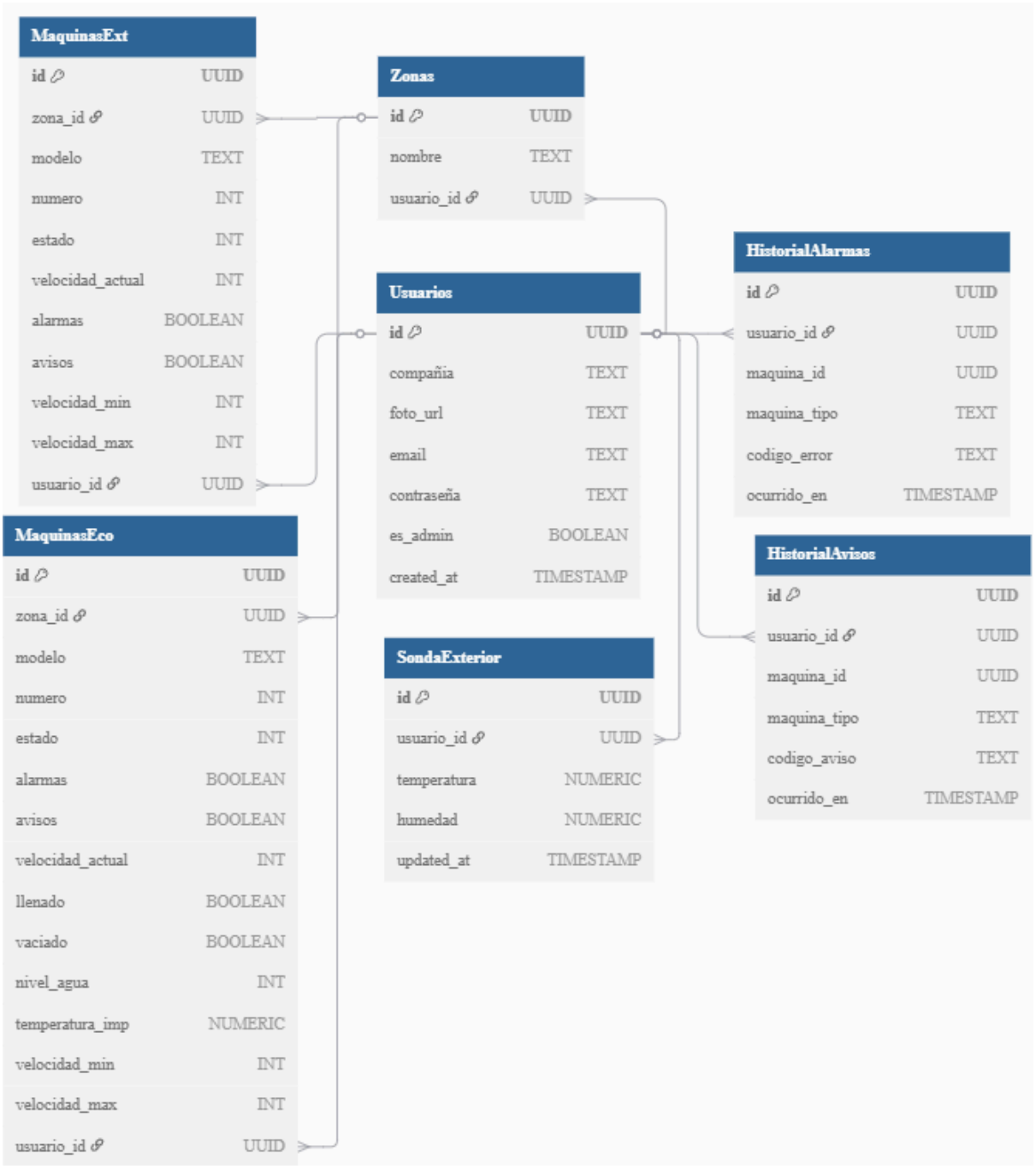
   - Si maquina_tipo = 'eco', entonces HistorialAlarmas.maquina_id → MaquinasEco.id (y lo mismo para avisos).

10. MaquinasExt ← (0:N) HistorialAlarmas / HistorialAvisos

    - Si maquina_tipo = 'ext', entonces HistorialAlarmas.maquina_id → MaquinasExt.id (y lo mismo para avisos).



-


MaquinasExt
id UUID
zona_id UUID
modelo TEXT
numero INT
estado INT
velocidad_actual INT
alarmas BOOLEAN
avisos BOOLEAN
velocidad_min INT
velocidad_max INT
usuario_id UUID
Zonas
id UUID
nombre TEXT
usuario_id UUID
Usuarios
id UUID
compañia TEXT
foto_url TEXT
email TEXT
contraseña TEXT
es_admin BOOLEAN
created_at TIMESTAMP
HistorialAlarmas
id UUID
usuario_id UUID
maquina_id UUID
maquina_tipo TEXT
codigo_error TEXT
ocurrido_en TIMESTAMP
HistorialAvisos
id UUID
usuario_id UUID
maquina_id UUID
maquina_tipo TEXT
codigo_aviso TEXT
ocurrido_en TIMESTAMP
MaquinasEco
id UUID
zona_id UUID
modelo TEXT
numero INT
estado INT
alarmas BOOLEAN
avisos BOOLEAN
velocidad_actual INT
llenado BOOLEAN
vaciado BOOLEAN
nivel_agua INT
temperatura_imp NUMERIC
velocidad_min INT
velocidad_max INT
usuario_id UUID
SondaExterior
id UUID
usuario_id UUID
temperatura NUMERIC
humedad NUMERIC
updated_at TIMESTAMP

---

6. Autenticación y gestión de sesiones

Para garantizar la integridad y confidencialidad de los datos en la plataforma, se han implementado las siguientes medidas:

Autenticación: Se emplea Supabase Auth (JWT) para gestionar sesiones. El frontend (React) suscribe cambios con supabase.auth.onAuthStateChange() y, ante errores 401/403, fuerza logout y redirige a login.

Row-Level Security (RLS): Activo en todas las tablas críticas. Cada política utiliza auth.uid() para que los usuarios solo accedan a sus propios registros, mientras que los administradores tienen permisos globales.

CORS y CSP: En Supabase solo se permiten orígenes https://app.airmagic.com y http://localhost:3000. En producción, se añade una cabecera CSP que restringe scripts, estilos e imágenes a dominios de confianza.

Variables de entorno: Se versiona solo .env.example con VITE_SUPABASE_URL y VITE_SUPABASE_ANON_KEY; las claves privadas (SERVICE_ROLE_KEY) quedan fuera de Git y solo en el backend.

Prevención de ataques comunes:

SQL Injection: Solo consultas parametrizadas via supabase-js.

XSS: React escapa por defecto y, si se inyecta HTML, se sanitiza con DOMPurify.

CSRF: No se usan cookies de sesión; el JWT va en el header Authorization.

HTTPS obligatorio: Todo el tráfico (front↔Supabase) se sirve mediante HTTPS.

---

7. Flujo de Datos y Servicios

- **supabase.js**: inicializa cliente Supabase con URL y ANON key.
- **authService**: métodos signIn(), signOut(), getUser().
- **userService**: fetchUsers(), editUser(), createUser(), deleteUser().
- **zoneService**: fetchZones().
- **machineService**:
    - fetchByZone(zoneId),
    - updateMachine(id, data),
    - fetchAlerts().
- **Historial** dentro de HistoryModal.jsx usa consultas filtradas por usuario_id y ordenadas por ocurrido_en.

---

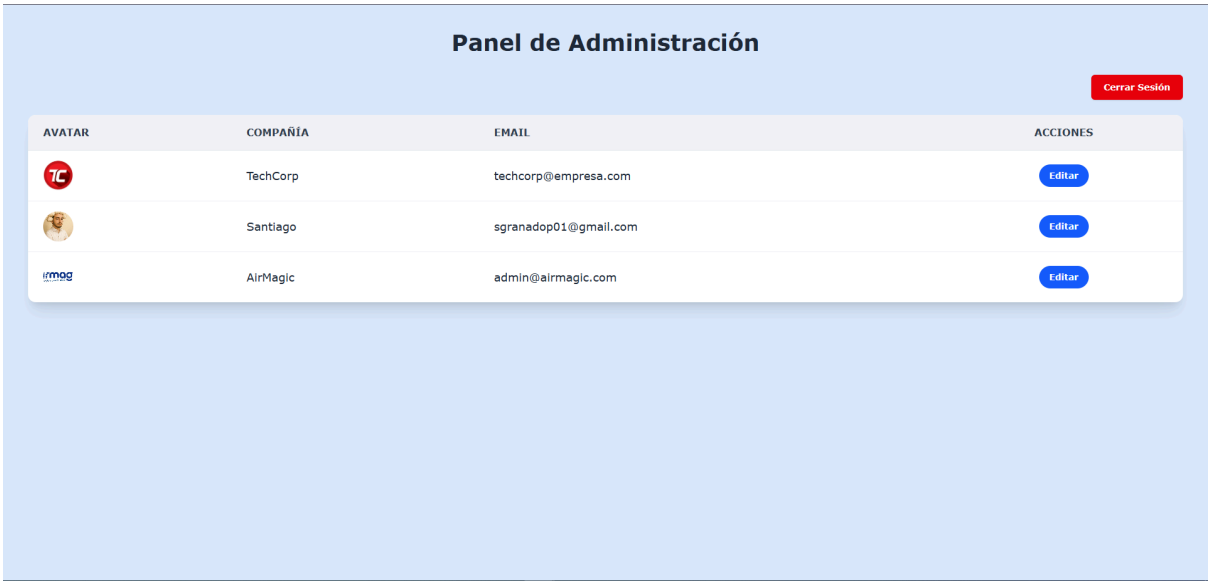
8. Hooks y Contexto

- **AuthContext.jsx**
    - user global, métodos login(), logout().
- **useAuth()**: consume contexto Auth.
- **useUsers()**, **useZones()**, **useMachines(zoneId)**, **useProfile()**:
    - Encapsulan llamadas a servicios + estados loading/error.

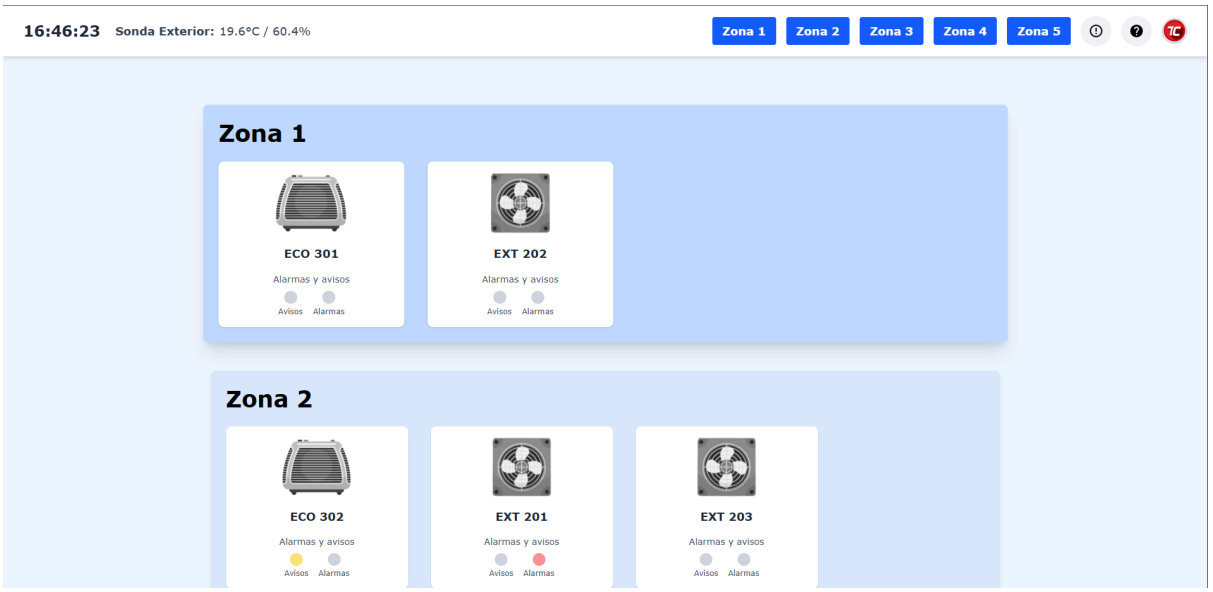
9 Componentes Clave

- **AdminPanel.jsx**: tabla desktop/mobile, modales EditProfileModal, Register.
- **Navbar.jsx**: reloj (useEffect), sonda (consulta), zones (useZones), historial, ayuda, avatar. Las zonas se dibujan dinámicamente.
- **HistoryModal.jsx**: pestañas alarmas/avisos, filtra por user.id, enriquece con nombre de máquina.
- **MachineCard.jsx**: tarjeta resumen con indicadores de estado y botones de acción.
- **ModalMachine.jsx**: detalle extenso y controles de modos.
- **EditProfileModal.jsx**: edita compañía y avatar.
- **PasswordChangeModal.jsx**: formulario de cambio de contraseña.
- **LoadingSpinner.jsx**: indicador de carga.

AdminPanel.jsx



Dashboard.jsx



---

10. Autenticación y Rutas Protegidas

- **ProtectedRoute.jsx**: Redirige a login si no hay sesión.
- **AdminRoute.jsx**: Solo permite acceso si user.es_admin === true.
- **Dashboard.jsx**: Permite acceso a todo usuario que no sea administrador.
- **ErrorPage.jsx**: Redirige si la ruta no pertenece a la aplicación o no puede acceder a ella.

---

11. Despliegue en Producción y CI/CD

Hosting en Vercel (u otro proveedor similar):

- Se publica la carpeta dist/ generada por npm run build en Vercel, Netlify o un servicio equivalente que importe directamente el repositorio.
- En el panel de la plataforma se configuran las variables de entorno (VITE_SUPABASE_URL y VITE_SUPABASE_ANON_KEY) para que el frontend se conecte correctamente a Supabase.

Automatización con GitHub Actions:

- Al hacer push a la rama main, GitHub Actions instala dependencias, ejecuta el linter (npm run lint) y crea el build (npm run build).
- Si no hay errores, se lanza automáticamente el despliegue en Vercel (o Netlify), usando el token y los IDs configurados como secretos en GitHub.

Verificación post-despliegue:

- Comprobar que la URL pública (por ejemplo, https://app.airmagic.com) carga sin errores.
- Asegurarse de que las rutas protegidas redirigen correctamente al login y que las llamadas a Supabase funcionan con las variables de producción.
- El proyecto incluye un fichero llamado vercel.json, para solucionar el problema de las rutas con react y vercel para que redirija a /index.html.

---

12. Propuesta de mejora

Teniendo en cuenta que la herramienta se implantará “en real” en AirMagic (empresa líder en el sector), las principales mejoras se basarán en la experiencia de producción y las necesidades que expresen directamente los clientes. A continuación se describen los tres ámbitos clave que quedan pendientes de optimizar:

1. **Perfeccionar la conexión con las máquinas**

    - Gestión de reconexiones y fiabilidad: implementar lógica de reintento automático cuando una máquina pierda conectividad (por ejemplo, reconexiones exponenciales) para evitar que datos se pierdan en entornos con red intermitente.

    - Monitorización continua: añadir indicadores de “estado de enlace” (online/offline) tanto en el Dashboard como en la App de escritorio, de modo que el técnico sepa en tiempo real si alguna máquina está desconectada.

- Validación de datos entrantes: comprobar en la App de escritorio que los paquetes recibidos desde la máquina cumplen un formato mínimo (timestamp, valores dentro de rangos plausibles) antes de insertarlos en la base de datos.

2. **Mejorar la base de datos para rendimiento y conexión real**

    - Optimización de índices: revisar las consultas más frecuentes (p. ej. lecturas recientes por máquina) y crear índices compuestos (máquina_id + timestamp) que aceleren el acceso a datos de estado actual.

    - Particionado o archiving de históricos: para tablas de alto crecimiento (historial de alarmas/avisos), evaluar particionarlas por rango de fecha (mensual) o mover datos antiguos a un esquema de archiving. Así se reduce el “peso” de las consultas en producción y mejora la velocidad global.

    - Pooling y configuración de conexiones: ajustar parámetros de conexión de Supabase/PostgreSQL (p. ej. max_connections, pool_size) para soportar lecturas en tiempo real de múltiples máquinas sin generar cuellos de botella.

    - Medición en producción y ajustes iterativos: una vez en entorno real, medir latencias (tiempo de respuesta de consultas principales) y, en base a esos datos, refinar tablas, columnas e índices.

3. **Añadir más funcionalidades al Admin Panel**
    - Gestión avanzada de usuarios:

        - Posibilidad de asignar roles intermedios o agrupar usuarios por “zona” para que un técnico solo vea las máquinas de su área.

        - Búsqueda y filtrado dinámico (por email, zona, rol) con paginación, para cuando crezca el número de usuarios.

4. **Control masivo de máquinas:**

    - Funcionalidad de “bulk actions” para activar/desactivar varios equipos a la vez o reasignarlos de zona en lote.

    - Botón de exportar lista de máquinas (CSV/Excel) con columnas clave (estado, zona, última lectura).

5. **Reportes y estadísticas básicas:**

    - Gráfica sencilla de número de alarmas por día/por zona, de forma que el administrador tenga visibilidad rápida de picos de incidencias.

    - Panel de “alertas recientes” con filtros por prioridad (alta, media, baja) y botón para marcar una alarma como “resuelta”.

Nota final:
Debido a que AirMagic ya limita el alcance inicial y muchas de estas mejoras dependen de la retroalimentación de los clientes una vez el sistema esté en producción, el plan de evolución consistirá en:

1. **Recoger métricas y feedback directo** de los usuarios finales para priorizar ajustes de conectividad, nuevos índices o funcionalidades específicas.

2. **Iterar en ciclos cortos**: tras cada sprint, desplegar en staging y, si procede, llevar a producción pequeñas mejoras (p. ej. nuevo índice en la base de datos o un widget extra en el Admin Panel).

3. **Documentar y versionar** cada cambio que mejore la conexión, rendimiento o experiencia de administración, de modo que el historial de necesidades quede registrado y se puedan replicar buenas prácticas para clientes futuros.

Con esta estrategia, la plataforma evolucionará de forma alineada con las necesidades reales de AirMagic y sus clientes, garantizando que los ajustes clave en la conexión con las máquinas, la base de datos y el Admin Panel respondan efectivamente a la demanda en producción.