Documentación Técnica — Sistema de Información Cultural de Tenjo

Introducción

El Sistema Unificado de Información Cultural de Tenjo es una plataforma web desarrollada para el Instituto Municipal de Cultura y Turismo de Tenjo (IMCT), entidad adscrita a la Alcaldía Municipal de Tenjo, Cundinamarca. Su propósito es centralizar y digitalizar los procesos de gestión cultural del municipio, facilitando la interacción entre ciudadanos, artistas, jurados y administradores en un entorno seguro y accesible.

La plataforma integra dos subsistemas complementarios. SICUT (Sistema de Información Cultural de Tenjo) permite el registro y caracterización socioeconómica de los agentes culturales del municipio, así como la construcción de un directorio cultural público. ArTenjo gestiona el ciclo completo de convocatorias artísticas: desde la publicación de convocatorias y la postulación de propuestas por parte de los ciudadanos, hasta la evaluación y calificación por parte de jurados designados.

El sistema contempla tres perfiles de usuario: ciudadano (registro, perfiles y postulación de propuestas), jurado (evaluación y calificación de propuestas artísticas) y superadmin (administración de convocatorias, usuarios, directorio y gestión de jurados). La aplicación está publicada en `cultura.tenjoculturayturismo.gov.co`

---

1. Stack tecnológico

Capa	Tecnología
Frontend	React 18 + TypeScript + Vite 5
Estilos	Tailwind CSS 3 + shadcn/ui (Radix primitives)
Estado / datos	TanStack React Query + Supabase JS SDK

Routing	React Router DOM v6
Backend	Supabase (PostgreSQL 15, Storage, Edge Functions)
Edge Functions	Deno (TypeScript)
Email	SMTP vía denomailer en Edge Function

---

2. Estructura de archivos

```
├── src/
│   ├── pages/                        # 15 vistas
│   │   ├── Login.tsx                  # Inicio de sesión
│   │   ├── Register.tsx               # Registro de usuario
│   │   ├── Home.tsx                   # Selector de servicios
│   │   ├── Index.tsx                  # Redirección inicial
│   │   ├── SicutDashboard.tsx         # Dashboard SICUT
│   │   ├── PerfilSicut.tsx            # Ver perfil SICUT
│   │   ├── PerfilSicutEditar.tsx      # Editar perfil SICUT
│   │   ├── DirectorioSicut.tsx        # Directorio cultural
│   │   ├── ArtenjoDashboard.tsx       # Dashboard ArTenjo
│   │   ├── ArtenjoCrearPropuesta.tsx  # Crear propuesta artística
│   │   ├── PerfilArtenjo.tsx          # Perfil ArTenjo
│   │   ├── AdminSicut.tsx             # Panel admin SICUT
│   │   ├── AdminArtenjo.tsx           # Panel admin ArTenjo
```

```
│   │   ├── JuradoArtenjo.tsx          # Panel de jurado
│   │   └── NotFound.tsx               # 404
│   ├── components/
│   │   ├── ui/                        # 40+ componentes shadcn/ui
│   │   └── NavLink.tsx                # Enlace de navegación
│   ├── assets/
│   │   ├── services.json              # Configuración de módulos
│   │   ├── database_export.sql        # Export SQL de referencia
│   │   └── *.png / *.webp / *.jpg     # Logos e imágenes
│   ├── hooks/
│   │   ├── use-mobile.tsx             # Detección de dispositivo móvil
│   │   └── use-toast.ts              # Hook de notificaciones
│   ├── integrations/supabase/
│   │   ├── client.ts                  # Cliente Supabase (auto-generado)
│   │   └── types.ts                   # Tipos TypeScript (auto-generado)
│   └── lib/
│       ├── utils.ts                   # Utilidades (cn, etc.)
│       └── sendEmail.ts              # Helper para invocar Edge Function de email
├── supabase/
│   ├── functions/
│   │   └── send-email/index.ts        # Edge Function: envío de correos SMTP
│   ├── migrations/                    # 22 migraciones SQL incrementales
│   └── config.toml                    # Configuración del proyecto Supabase
├── TECHNICAL_DOCS.md                  # Este documento
├── index.html                         # Entry point HTML
```

```
├── vite.config.ts                    # Configuración de Vite
├── tailwind.config.ts                # Configuración de Tailwind
└── package.json
```

---

3. Autenticación

El sistema no utiliza Supabase Auth nativo. Implementa un esquema de autenticación personalizado:

- El usuario se registra con nombre, apellidos, tipo/número de documento y teléfono en la tabla `usuarios`.
- El inicio de sesión valida `numero_documento` + últimos 4 dígitos del `telefono`.
- La sesión se almacena en `sessionStorage` con la clave `usuario_id`.

---

4. Roles y permisos

- Enum `app_role`: `ciudadano`, `superadmin`, `jurado_siartenjo`, `profesor`
- Tabla `user_roles`: Relación usuario ↔ rol (many-to-many).
- Función `has_role(_user_id, _role)`: `SECURITY DEFINER`, consulta roles sin recursión RLS.
- Trigger `assign_default_role`: Asigna automáticamente el rol `ciudadano` al crear un usuario.
- Funciones `aprobar_jurado` / `rechazar_jurado`: Gestionan la asignación/revocación del rol `jurado_siartenjo`.

---

5. Base de datos — Esquema (8 tablas)

Tabla	Propósito	Relaciones

`usuarios`	Registro básico (nombre, documento, teléfono)	PK: `id`
`user_roles`	Roles asignados por usuario	FK → `usuarios.id`
`perfil_sicut`	Perfil socioeconómico para SICUT	FK → `usuarios.id` (1:1)
`perfil_artenjo`	Perfil de participante ArTenjo	FK → `usuarios.id` (1:1)
`convocatorias_siartenjo`	Convocatorias artísticas (título, fechas)	—
`propuestas_artenjo`	Propuestas enviadas a convocatorias	FK → `convocatorias_siartenjo.id`
`calificaciones_artenjo`	Calificaciones de jurados a propuestas	FK → `usuarios.id`, `propuestas_artenjo.id`
`directorio_sicut`	Directorio cultural público	FK → `usuarios.id` (1:1)
`perfil_jurados`	Perfiles de aspirantes a jurado	FK → `usuarios.id`

Diagrama de relaciones

```
usuarios (1) ──── (1) perfil_sicut
    │
    ├──── (1) perfil_artenjo
    │
    ├──── (1) directorio_sicut
    │
    ├──── (*) user_roles
    │
    ├──── (*) perfil_jurados
    │
    └──── (*) calificaciones_artenjo ──── (*) propuestas_artenjo ──── (*) convocatorias_siartenjo
```

Soft delete

Todas las tablas principales incluyen la columna `deleted_at` (timestamp nullable). Los registros con `deleted_at IS NOT NULL` se consideran eliminados lógicamente y se filtran en las consultas.

---

6. Funciones de base de datos

Función	Tipo	Descripción
`has_role(_user_id, _role)`	SQL / SECURITY	Verifica si un usuario tiene un rol específico

	DEFINER	
`assign_default_role`	Trigger function	Asigna rol `ciudadano` al insertar en `usuarios`
`soft_delete_propuesta(p_propuesta_id, p_usuario_id)`	RPC	Borrado lógico de propuesta (valida propiedad)
`soft_delete_convocatoria(p_convocatoria_id)`	RPC	Borrado lógico de convocatoria
`aprobar_jurado(p_perfil_jurado_id)`	RPC	Aprueba jurado: actualiza estado + asigna rol
`rechazar_jurado(p_perfil_jurado_id)`	RPC	Rechaza jurado: actualiza estado + revoca rol
`prevent_propuesta_unenviado`	Trigger function	Impide revertir el estado de una propuesta enviada
`update_*_updated_at`	Trigger functions	Actualiza `updated_at` automáticamente en modificaciones

---

7. Storage

- Bucket `directorio-fotos` (público): Almacena fotos de perfil del directorio cultural y assets para templates de email.
- Políticas: lectura pública, escritura restringida a usuarios autenticados.

---

8. Edge Functions

`send-email`

Envía correos electrónicos vía SMTP con template HTML institucional.

Secrets requeridos:

- `SMTP_HOST`
- `SMTP_PORT`
- `SMTP_USER`
- `SMTP_PASS`
- `SMTP_FROM`

Payload:

```
{
  "to": "destinatario@email.com",
  "subject": "Asunto del correo",
  "nombre": "Nombre del destinatario",
  "contenido": "Contenido HTML del cuerpo"
}
```

---

9. Rutas de la aplicación

Ruta	Página	Acceso
`/`	Login	Público
`/registro`	Registro	Público
`/inicio`	Selector de servicios	Autenticado

`/sicut`	Dashboard SICUT	Autenticado
`/perfil-sicut`	Perfil SICUT	Autenticado
`/perfil-sicut-editar`	Editar perfil SICUT	Autenticado
`/directorio-sicut`	Directorio cultural	Autenticado
`/artenjo/inicio`	Dashboard ArTenjo	Autenticado
`/artenjo/propuesta`	Crear propuesta	Autenticado
`/perfil-artenjo`	Perfil ArTenjo	Autenticado
`/admin/sicut`	Admin SICUT	Superadmin
`/admin/artenjo`	Admin ArTenjo	Superadmin
`/jurado/artenjo`	Panel jurado	Jurado

---

10. Configuración de servicios

El archivo `src/assets/services.json` controla la disponibilidad de los módulos:

```
{
  "sicut": { "nombre": "Sistema SICUT", "status": 0 },
  "artenjo": { "nombre": "Sistema ArTenjo", "status": 1 }
}
```

- `status: 0` → Módulo deshabilitado (no visible para usuarios)
- `status: 1` → Módulo habilitado y accesible