Documento de arquitectura y desarrollo del sistema SID

Henry Ricaurte Mora

Abril de 2025

Índice

1. Introducción

En esta sección se brinda una visión introductoria de los conceptos relacionados con la descripción del sistema del SID. Es importante establecer una estructura clara de los componentes que lo conforman.

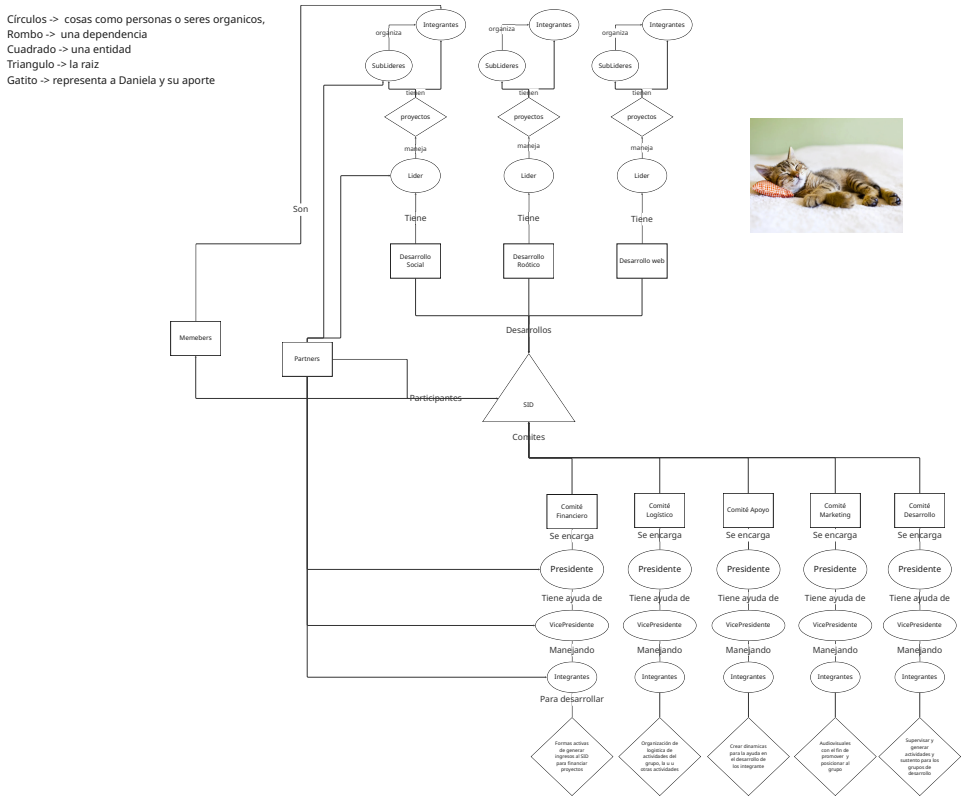

Figura 1: Diagrama de organización del SID

El SID está conformado por tres ramas principales:

- **Desarrollos:** Social, Robótico y Web.
- **Comités:** Financiera, Logística, Apoyo, Marketing y Desarrollo.
- **Participantes:** *Partners* y *Members*.

1.1. Comités

Los comités son responsables del funcionamiento del SID. Cada uno tiene un presidente y un vicepresidente, quienes lideran al equipo para desarrollar estrategias como:

- Generar ingresos para financiar proyectos.
- Organizar la logística de las actividades.
- Establecer dinámicas de formación para los integrantes.
- Crear contenido audiovisual para promoción.
- Supervisar y respaldar a los grupos de desarrollo.

1.2. Desarrollos

Cada desarrollo (Social, Robótico y Web) cuenta con un líder, sublíderes y un equipo. El líder coordina el proyecto general, los sublíderes gestionan apartados específicos y los integrantes trabajan bajo su guía.

Por ejemplo, este documento forma parte de un proyecto en el desarrollo Web.

Los *Partners* incluyen a presidentes, vicepresidentes, líderes y sublíderes. Los *Members* son integrantes que participan activamente en cada uno de los apartados de desarrollo.

2. Descripción General del Sistema

Se desarrollará una plataforma integral para la gestión y difusión de información del grupo SID. Esta incluirá herramientas para la presentación institucional, la oferta de cursos de formación y la divulgación de conocimiento especializado.

2.1. Componentes Principales

- **Portal Web Principal:** Será el sitio informativo del grupo, donde se mostrará quiénes somos y cuáles son nuestras áreas de trabajo. Este portal incluirá secciones dedicadas a los distintos desarrollos del grupo, cada una con su propia página que detallará a sus integrantes, proyectos y líderes. Asimismo, se presentarán los comités con una estructura similar. También se incluirá información general sobre el SID, sus actividades, proyectos, equipos, colaboraciones, cursos, noticias e integrantes, además de un formulario de inscripción para nuevos miembros o interesados.
- **Panel Administrativo:** Permitirá la gestión del contenido visible en el portal principal, así como la administración de los datos de los usuarios provenientes de los formularios de inscripción.
- **Sistema de Gestión Académica:** Este componente permitirá la inscripción y administración de los cursos ofrecidos. Incluirá funcionalidades específicas para tres tipos de usuarios: administrador, profesor y estudiante/usuario. También facilitará la asignación y gestión de profesores por curso.
- **SID Academy:** Proyecto futuro que se plantea como una plataforma educativa de cursos breves, similar a QtAcademy, con el objetivo de fortalecer el aprendizaje especializado dentro de la comunidad.

3. Especificación Microservicios:

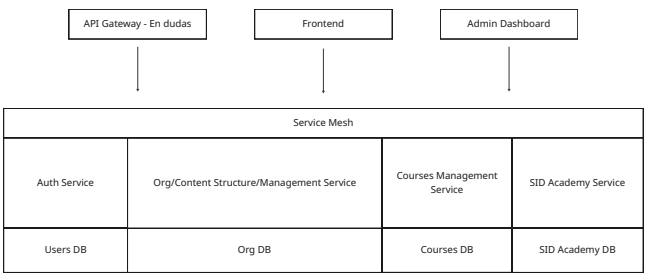

Figura 2: Diagrama de microservicios del proyecto

Microservicios propuestos

3.1. Servicio de Autenticación y Usuarios

- Gestión completa del ciclo de vida de usuarios
- Autenticación y autorización
- Gestión de tokens y sesiones

3.2. Servicio de Estructura Organizacional

- Visualización de desarrollos (Web, Robótica, etc.)
- Visualización de comités (Financiero, Social, etc.)
- Visualización de noticias y actualizaciones
- Visualización de información sobre fundaciones y trabajos realizados
- Visualización de integrantes y Formulario
- Administración y gestión de desarrollos (Web, Robótica, etc.)
- Administración y gestión de comités (Financiero, Social, etc.)
- Administración y gestión de noticias y actualizaciones
- Administración y gestión de información sobre fundaciones y trabajos realizados
- Administración y gestión de integrantes
- Administracion de integrantes mediante Formulario.

3.3. Servicio de Gestión Académica

- Catálogo e inscripción a cursos
- Gestión de profesores y alumnos
- Seguimiento y evaluación

3.4. Servicio QtAcademy (futuro)

- Gestión de contenidos educativos cortos
- Sistema de aprobación y curación
- Análisis de métricas de consumo

Módulo	Función principal	Métodos API	Seguridad
AuthService	Autenticación y autorización Gestión de usuarios y roles Administración de tokens y sesiones	▪ POST ▪ GET ▪ PUT	JWT (Solo login es público)
PortalService y Dashboard control	Visualización pública del SID (desarrollos, comités, noticias) Administración de desarrollos Gestión de comités y fundaciones	▪ GET ▪ POST ▪ PUT ▪ PATCH ▪ DELETE	Mixto: • Público (lectura) • JWT + roles (escritura)
CoursesService	Gestión de cursos e inscripciones Administración de profesores/alumnos Seguimiento y evaluación académica	▪ GET ▪ POST ▪ PUT	JWT + roles (basado en perfiles académicos)
SIDAcademy *(futuro)*	Contenidos educativos cortos Curación de material didáctico Análisis de métricas de aprendizaje	▪ GET ▪ POST ▪ PUT	JWT + roles (según nivel de acceso)

Cuadro 1: Arquitectura de modulos propuesta para el sistema SID

4. Arquitectura de Base de Datos:

Como base de datos se elige una NewSQL como Neon la cual opera con postgresql, elegido por comidad de diseño y ademas brinda lo necesario para el proyecto. La base de datos generada es:

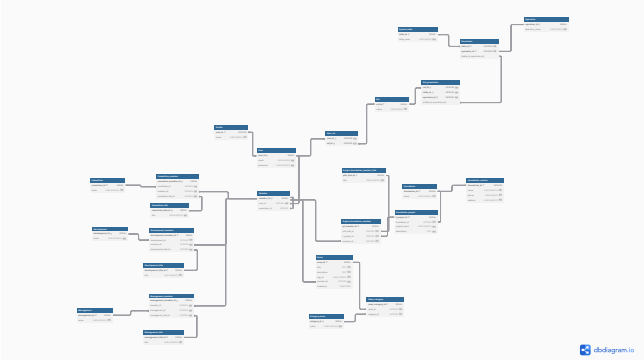

Figura 3: Diagrama de base de datos relacional

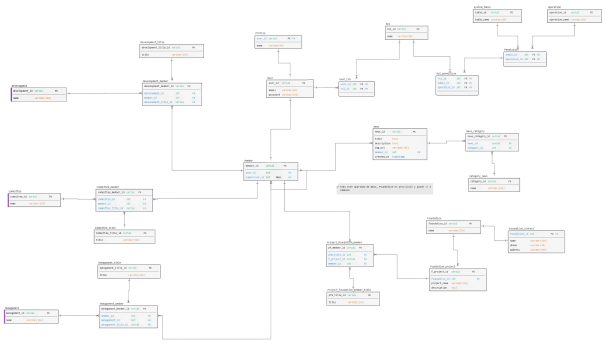

Figura 4: Diagrama de base de datos relacional desde google

5. Arquitectura de modulos y microservicios:

Vamos a abordar todo el tema de los módulos y microservicios ademas de su arquitectura paso por paso. El único microservicio actual es AuthService, luego estan los módulos monoliticos como el **Portal Content Service**

6. Portal Content Service

Este servicio de portal de contenido que se generará es la dicha "Página del SID", desde el lado del backend tenemos que suplir información relevante, esta información está dividida en:

- Actividades (no está en BD)
- Proyectos (no está en BD)
- Cursos (no está en BD) – final
- Equipos (no está en BD)
- Noticias
- Aliados | Fundaciones

6.1. Endpoints REST

Con base en los elementos anteriores, definimos los siguientes endpoints REST con paginación de Spring Boot:

6.1.1. Actividades

6.1.2. Proyectos

6.1.3. Cursos

6.1.4. Equipos

6.1.5. Noticias

GET Todos (Paginado)

GET Noticias (paginado)

```
GET /api/news?page=0&size=10&sort=publishDate,desc
```

Estructura de respuesta

```
{
    "content": [
        {
            "id": integer,                    // ID unico de la noticia
            "title": string,                  // Titulo de la noticia
            "content": string(HTML),          // Contenido completo en HTML
            "publishDate": string(ISO-8601), // Fecha de publicacion
            "author": string,                 // Nombre del autor
            "imageUrl": string(URL),          // Ruta a la imagen destacada
            "tags": string[]                  // Array de etiquetas
        }
        // ... mas elementos
    ],
    "pageable": {
        "sort": object,       // Informacion de ordenamiento
        "pageSize": integer,  // Tamano de pagina solicitado
        "pageNumber": integer // Numero de pagina actual
        // ... mas metadatos de paginacion
    },
    "totalElements": integer, // Total de resultados
    "totalPages": integer,    // Total de paginas disponibles
    "first": boolean,         // Si es la primera pagina
    "last": boolean,          // Si es la ultima pagina
    "size": integer           // Tamano de la pagina actual
}
```

GET Individual

GET /api/news/id - Obtener noticia específica

```
GET /api/news/{id}
```

Estructura de respuesta

```
{
    "id": integer,                    // ID unico de la noticia
    "title": string,                  // Titulo de la noticia
    "content": string(HTML),          // Contenido completo en HTML
    "publishDate": string(ISO-8601), // Fecha de publicacion
    "author": string,                 // Nombre del autor
    "imageUrl": string(URL),          // Ruta a la imagen destacada
    "tags": string[]                  // Array de etiquetas
}
```

POST Creación

POST /api/news - Crear nueva noticia

```
POST /api/news
Content-Type: application/json
Authorization: Bearer <token>

{
    "title": string,       // Requerido: Titulo de la noticia
    "content": string(URL:.MD) // Requerido: Ruta para contenido posiblemente
        HTML o .MD , en su defecto el contenido en un string
    "tags": string[]  //  Array de etiquetas
    "imageURL": String(URL) // Ruta a la Imagen
}
```

Estructura de respuesta

```
{
    "id": integer,                  // ID generado automaticamente
    "title": string,                // Titulo proporcionado
    "status": string(enum),         // Estado inicial: DRAFT
    "createdAt": string(ISO-8601)   // Timestamp de creacion
}
```

PUT (Reemplazo completo)

PUT /api/news/id - Actualizar noticia completa

```
PUT /api/news/{id}
Content-Type: application/json
Authorization: Bearer <token>

{
    "title": string,        // Requerido: Nuevo titulo
    "content": string(HTML) // Requerido: Nuevo contenido
    "imageURL": string(URL) // Ruta de la nueva imagen
    // Todos los campos son requeridos en PUT ,
    // se pueden dejar en blanco si no se quiere actualizar ese apartado
}
```

Estructura de respuesta

```
{
    "id": integer,                  // ID de la noticia actualizada
    "title": string,                // Titulo actualizado
    "lastUpdated": string(ISO-8601)  // Timestamp de actualizacion
}
```

PATCH (Actualización parcial)

PATCH /api/news/id - Actualizar campos específicos

```
PATCH /api/news/{id}
Content-Type: application/json-patch+json
Authorization: Bearer <token>

[
    { "op": string(enum),
      "path": string,
      "value": any
    }, // Operacion JSON Patch

    // Operaciones permitidas: "replace", "add", "remove"
    // Ejemplo: { "op": "replace", "path": "/title", "value": "Nuevo titulo" }
]
```

Estructura de respuesta

```
{
    "id": integer,     // ID de la noticia modificada
    "changes": string[], // Lista de campos modificados
    "version": integer   // Nueva version del documento
}
```

DELETE

DELETE /api/news/id - Eliminar noticia

```
DELETE /api/news/{id}
Authorization: Bearer <token>
```

Respuesta

```
HTTP/1.1 204 No Content
// No se devuelve cuerpo de respuesta
```

6.1.6. Aliados y Fundaciones

GET Todos (Paginado)

GET Aliados (paginado)

```
GET /api/allies?page=0&size=10&type=FOUNDATION
```

Estructura de respuesta

```
{
    "content": [
        {
            "id": integer,                    // ID unico del aliado
            "name": string,                   // Nombre de la organizacion
            "type": string(enum),             // Tipo: FOUNDATION, COMPANY, NGO,
    etc.
            "description": string,            // Descripcion corta
            "logoUrl": string(URL),           // Ruta al logo
            "website": string(URL),           // Sitio web
            "partnerSince": string(ISO-8601) // Fecha de inicio de alianza
        }
        // ... mas elementos
    ],
    "pageable": {
        "sort": object,       // Informacion de ordenamiento
        "pageSize": integer,  // Tamanio de pagina solicitado
        "pageNumber": integer // Numero de pagina actual
        // ... mas metadatos de paginacion
    },
    "totalElements": integer, // Total de resultados
    "totalPages": integer,    // Total de paginas disponibles
    "first": boolean,         // Si es la primera pagina
    "last": boolean           // Si es la ultima pagina
}
```

GET Individual

GET /api/allies/id - Obtener aliado específico

```
GET /api/allies/{id}
```

Estructura de respuesta

```
{
    "id": integer,                    // ID unico del aliado
    "name": string,                   // Nombre de la organizacion
    "type": string(enum),             // Tipo: FOUNDATION, COMPANY, NGO, etc.
    "description": string,            // Descripcion corta
    "logoUrl": string(URL),           // Ruta al logo
    "website": string(URL),           // Sitio web
    "partnerSince": string(ISO-8601), // Fecha de inicio de alianza
    "contactInfo": {                  // Informacion de contacto
        "email": string(email),
        "phone": string
    },
    "projects": [                     // Proyectos asociados
        {
            "id": integer,
            "name": string,
            "status": string(enum)    // ACTIVE, COMPLETED, PLANNED
        }
    ],
    "socialMedia": {                  // Redes sociales
        "facebook": string(URL),
        "twitter": string(URL),
        "linkedin": string(URL)
    }
}
```

POST Creación

POST /api/allies - Crear nuevo aliado

```
POST /api/allies
Content-Type: application/json
Authorization: Bearer <token>

{
    "name": string,                   // Requerido: Nombre de la organizacion
    "type": string(enum),             // Requerido: FOUNDATION, COMPANY, NGO, etc.
    "description": string,            // Requerido: Descripcion
    "website": string(URL),           // Opcional: Sitio web
    "contactInfo": {                  // Opcional: Informacion de contacto
        "email": string(email),
        "phone": string
    },
    "socialMedia": {                  // Opcional: Redes sociales
        "facebook": string(URL),
        "twitter": string(URL),
        "linkedin": string(URL)
    }
}
```

Estructura de respuesta

```
{
    "id": integer,                  // ID generado automaticamente
    "name": string,                 // Nombre proporcionado
    "type": string(enum),           // Tipo proporcionado
    "createdAt": string(ISO-8601),  // Timestamp de creacion
    "status": string(enum)          // Estado inicial: PENDING_REVIEW
}
```

PUT (Reemplazo completo)

PUT /api/allies/id - Actualizar aliado completo

```
PUT /api/allies/{id}
Content-Type: application/json
Authorization: Bearer <token>

{
    "name": string,                 // Requerido: Nombre de la organizacion
    "type": string(enum),           // Requerido: FOUNDATION, COMPANY, NGO, etc.
    "description": string,          // Requerido: Descripcion
    "website": string(URL),         // Opcional: Sitio web
    "contactInfo": {                // Opcional: Informacion de contacto
        "email": string(email),
        "phone": string
    },
    "socialMedia": {                // Opcional: Redes sociales
        "facebook": string(URL),
        "twitter": string(URL),
        "linkedin": string(URL)
    }
}
```

Estructura de respuesta

```
{
    "id": integer,                  // ID del aliado actualizado
    "name": string,                 // Nombre actualizado
    "lastUpdated": string(ISO-8601), // Timestamp de actualizacion
    "status": string(enum)          // Estado: ACTIVE, PENDING_REVIEW, etc.
}
```

PATCH (Actualización parcial)

PATCH /api/allies/id - Actualizar campos específicos

```
PATCH /api/allies/{id}
Content-Type: application/json-patch+json
Authorization: Bearer <token>

[
    { "op": string(enum), "path": string, "value": any }, // Operacion JSON Patch
    // Operaciones permitidas: "replace", "add", "remove"
    // Ejemplo: { "op": "replace", "path": "/description", "value": "Nueva
        descripcion" }
]
```

Estructura de respuesta

```
{
    "id": integer,        // ID del aliado modificado
    "changes": string[],  // Lista de campos modificados
    "version": integer,   // Nueva version del documento
    "modifiedAt": string(ISO-8601) // Timestamp de modificacion
}
```

DELETE

DELETE /api/allies/id - Eliminar aliado

```
DELETE /api/allies/{id}
Authorization: Bearer <token>
```

Respuesta

```
HTTP/1.1 204 No Content
// No se devuelve cuerpo de respuesta
```