

Documento de arquitectura y desarrollo del sistema SID

Henry Ricaurte Mora

Abril de 2025

Índice

1. Introducción	3
1.1. Comités	3
1.2. Desarrollos	4
2. Descripción General del Sistema	4
2.1. Componentes Principales	4
3. Requerimientos del Sistema	4
3.1. Requerimientos Funcionales:	4
3.1.1. Portal Web Principal	4
3.1.2. Panel Administrativo (Dashboard)	5
3.1.3. API de Servicios	5
3.2. Requerimientos No Funcionales	5
3.2.1. Seguridad	5
3.2.2. Rendimiento	6
3.2.3. Escalabilidad	6
3.2.4. Mantenibilidad	6
3.2.5. Compatibilidad	6
3.2.6. Interoperabilidad	6
4. Especificación Módulos:	6
4.1. Servicio de Autenticación y Usuarios	7
4.2. Servicio de Estructura Organizacional	7
4.3. Servicio de Gestión Académica	7
4.4. Servicio QtAcademy (futuro)	7
5. Arquitectura de Base de Datos:	8
6. Arquitectura de modulos y microservicios:	9
7. Portal Content Service	9
7.1. Endpoints REST	9
7.1.1. Actividades	9
7.1.2. Proyectos	9
7.1.3. Cursos	9

7.1.4.	Equipos	9
7.1.5.	Noticias	9
7.1.6.	Aliados y Fundaciones	12

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.

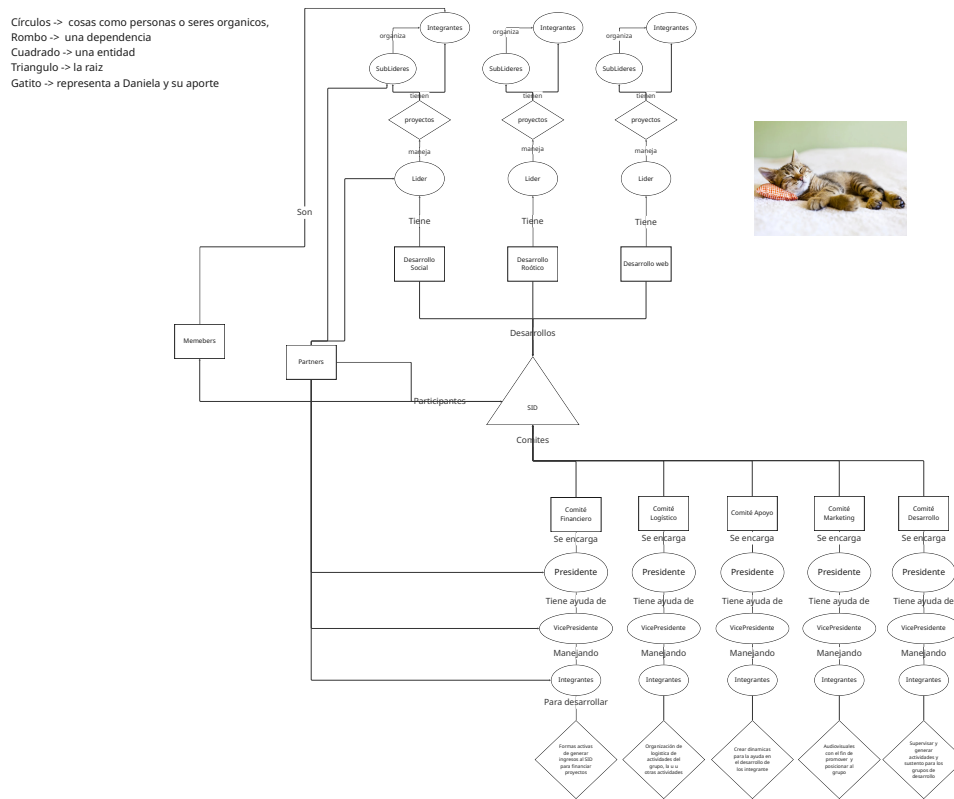


Figura 1: Diagrama de organización del SID

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. Requerimientos del Sistema

3.1. Requerimientos Funcionales:

3.1.1. Portal Web Principal

RF-001 El sistema debe mostrar información institucional del grupo SID.

RF-002 El sistema debe presentar secciones para cada desarrollo del grupo (Web, Robótica, Social) con sus integrantes, proyectos y líderes.

RF-003 El sistema debe mostrar información sobre los comités existentes.

- RF-004** El sistema debe listar actividades realizadas por el grupo.
- RF-005** El sistema debe mostrar proyectos desarrollados por el grupo.
- RF-006** El sistema debe exponer los cursos ofrecidos por el grupo.
- RF-007** El sistema debe presentar los equipos de trabajo.
- RF-008** El sistema debe publicar noticias relacionadas con el grupo.
- RF-009** El sistema debe mostrar información sobre fundaciones y colaboraciones.
- RF-010** El sistema debe incluir un formulario de inscripción para nuevos miembros.

3.1.2. Panel Administrativo (Dashboard)

- RF-011** El sistema debe permitir a los administradores gestionar el contenido del portal web.
- RF-012** El sistema debe proporcionar funcionalidades para administrar los datos de usuarios.
- RF-013** El sistema debe permitir la gestión de actividades (crear, modificar, eliminar).
- RF-014** El sistema debe permitir la gestión de proyectos (crear, modificar, eliminar).
- RF-015** El sistema debe permitir la gestión de cursos (crear, modificar, eliminar).
- RF-016** El sistema debe permitir la gestión de equipos (crear, modificar, eliminar).
- RF-017** El sistema debe permitir la gestión de noticias (crear, modificar, eliminar).
- RF-018** El sistema debe permitir la gestión de fundaciones (crear, modificar, eliminar).
- RF-019** El sistema debe implementar control de acceso basado en roles.
- RF-020** El sistema debe permitir la administración de permisos por rol.

3.1.3. API de Servicios

- RF-021** El sistema debe proporcionar endpoints RESTful para listar contenido (actividades, proyectos, etc.).
- RF-022** El sistema debe implementar paginación en todos los endpoints que devuelven listas.
- RF-023** El sistema debe manejar formatos estándar para solicitudes y respuestas (JSON).
- RF-024** El sistema debe implementar autenticación mediante tokens JWT.
- RF-025** El sistema debe proporcionar endpoints para la gestión de contenidos protegidos por autenticación.

3.2. Requerimientos No Funcionales

3.2.1. Seguridad

- RF-026** El sistema debe implementar autenticación OAuth 2.0 con proveedores de identidad externos (Microsoft Entra ID/Azure AD / Google).
- RF-027** El sistema debe validar y procesar tokens JWT emitidos por el proveedor de identidad.

- RF-028** El sistema debe implementar control de acceso basado en roles mediante la base de datos
- RF-029** El sistema debe proteger los endpoints administrativos con autorización basada en roles
- RF-030** El sistema debe usar HTTPS para todas las comunicaciones (TLS 1.2+). | Toca entender que es a profundidad
- RF-031** El sistema debe implementar protección contra ataques comunes (CSRF, XSS, SQL Injection).

3.2.2. Rendimiento

- RF-032** El sistema debe responder a las solicitudes API en menos de 1 segundo bajo carga normal.
- RF-033** El sistema debe soportar al menos 100 usuarios concurrentes.
- RF-034** Las consultas a la base de datos deben optimizarse para minimizar el tiempo de respuesta.

3.2.3. Escalabilidad

- RF-035** La arquitectura debe permitir escalar componentes individuales según demanda.
- RF-036** El sistema debe ser compatible con despliegue en contenedores (Docker/Kubernetes).

3.2.4. Mantenibilidad

- RF-037** El código debe seguir los principios SOLID y patrones de diseño apropiados.
- RF-038** El sistema debe implementar logging estructurado (JSON) para diagnóstico.
- RF-039** El código debe incluir documentación JavaDoc y OpenAPI para las APIs.

3.2.5. Compatibilidad

- RF-040** Las APIs deben seguir el estándar RESTful nivel 2 de Richardson.
- RF-041** El sistema debe proporcionar documentación interactiva mediante Swagger UI.

3.2.6. Interoperabilidad

- RF-044** El sistema debe exponer APIs con versionado semántico (v1, v2). | Acordar si mandarlo en header o en la api. (obligatorio tag github)
- RF-045** Las integraciones deben usar protocolos estándar (OAuth 2.0, OpenID Connect).

4. Especificación Módulos:

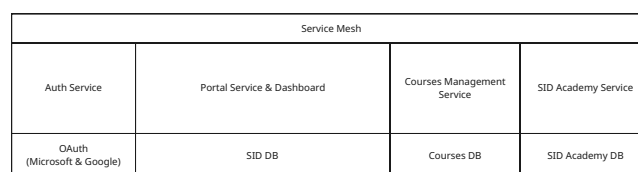


Figura 2: Diagrama de microservicios del proyecto

Microservicios propuestos

4.1. Servicio de Autenticación y Usuarios

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

4.2. 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
- Administración de integrantes mediante Formulario.

4.3. Servicio de Gestión Académica

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

4.4. 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	<ul style="list-style-type: none"> ▪ 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	<ul style="list-style-type: none"> ▪ GET ▪ POST ▪ PUT ▪ PATCH ▪ DELETE 	Mixto: <ul style="list-style-type: none"> • Público (lectura) • JWT + roles (escritura)
CoursesService	Gestión de cursos e inscripciones Administración de profesores/alumnos Seguimiento y evaluación académica	<ul style="list-style-type: none"> ▪ GET ▪ POST ▪ PUT 	JWT + roles (basado en perfiles académicos)
SIDAcademy (futuro)	Contenidos educativos cortos Curación de material didáctico Análisis de métricas de aprendizaje	<ul style="list-style-type: none"> ▪ GET ▪ POST ▪ PUT 	JWT + roles (según nivel de acceso)

Cuadro 1: Arquitectura de modulos propuesta para el sistema SID

5. 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:

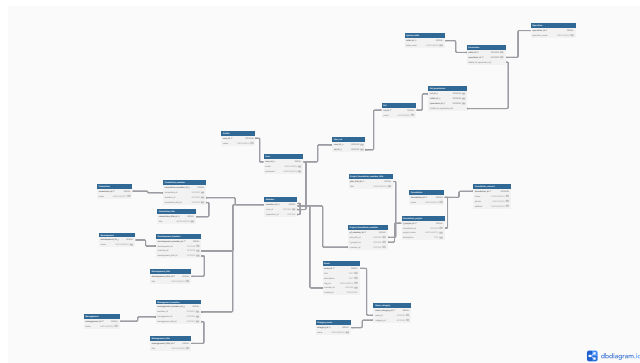


Figura 3: Diagrama de base de datos relacional

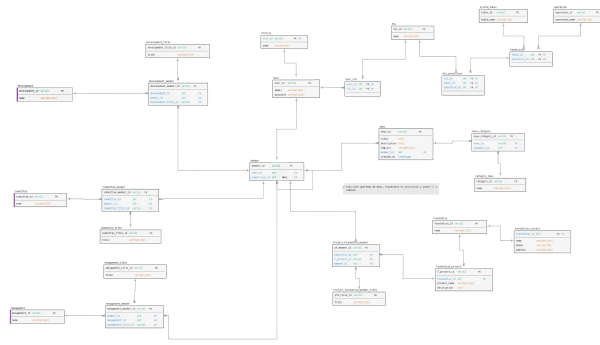


Figura 4: Diagrama de base de datos relacional desde google

6. Arquitectura de módulos y microservicios:

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

7. 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

7.1. Endpoints REST

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

7.1.1. Actividades

7.1.2. Proyectos

7.1.3. Cursos

7.1.4. Equipos

7.1.5. Noticias

GET Todos (Paginado)

GET Noticias (paginado)

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

Estructura de respuesta

```
1 {
2   "content": [
3     {
4       "id": integer,           // ID unico de la noticia
5       "title": string,        // Titulo de la noticia
6       "content": string(HTML), // Contenido completo en HTML
7       "publishDate": string(ISO-8601), // Fecha de publicacion
8       "author": string,       // Nombre del autor
9       "imageUrl": string(URL), // Ruta a la imagen destacada
10      "tags": string[]         // Array de etiquetas
11    }
12    // ... mas elementos
13  ],
14  "pageable": {
15    "sort": object,           // Informacion de ordenamiento
16    "pageSize": integer,     // Tamano de pagina solicitado
17    "pageNumber": integer    // Numero de pagina actual
18    // ... mas metadatos de paginacion
19  },
20  "totalElements": integer, // Total de resultados
21  "totalPages": integer,    // Total de paginas disponibles
22  "first": boolean,         // Si es la primera pagina
23  "last": boolean,          // Si es la ultima pagina
24  "size": integer           // Tamano de la pagina actual
25 }
```

GET Individual

GET /api/news/id - Obtener noticia específica

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

Estructura de respuesta

```
1 {
2   "id": integer,           // ID unico de la noticia
3   "title": string,        // Titulo de la noticia
4   "content": string(HTML), // Contenido completo en HTML
5   "publishDate": string(ISO-8601), // Fecha de publicacion
6   "author": string,       // Nombre del autor
7   "imageUrl": string(URL), // Ruta a la imagen destacada
8   "tags": string[]         // Array de etiquetas
9 }
```

POST Creación

POST /api/news - Crear nueva noticia

```
1 POST /api/news
2 Content-Type: application/json
3 Authorization: Bearer <token>
4
5 {
6     "title": string,           // Requerido: Titulo de la noticia
7     "content": string(URL:.MD) // Requerido: Ruta para contenido posiblemente
                                // HTML o .MD , en su defecto el contenido en un string
8     "tags": string[]          // Array de etiquetas
9     "imageURL": String(URL)    // Ruta a la Imagen
10 }
```

Estructura de respuesta

```
1 {
2     "id": integer,             // ID generado automaticamente
3     "title": string,           // Titulo proporcionado
4     "status": string(enum),     // Estado inicial: DRAFT
5     "createdAt": string(ISO-8601) // Timestamp de creacion
6 }
```

PUT (Reemplazo completo)

PUT /api/news/id - Actualizar noticia completa

```
1 PUT /api/news/{id}
2 Content-Type: application/json
3 Authorization: Bearer <token>
4
5 {
6     "title": string,           // Requerido: Nuevo titulo
7     "content": string(HTML)    // Requerido: Nuevo contenido
8     "imageURL": string(URL)    // Ruta de la nueva imagen
9     // Todos los campos son requeridos en PUT ,
10    // se pueden dejar en blanco si no se quiere actualizar ese apartado
11 }
```

Estructura de respuesta

```
1 {
2     "id": integer,             // ID de la noticia actualizada
3     "title": string,           // Titulo actualizado
4     "lastUpdated": string(ISO-8601) // Timestamp de actualizacion
5 }
```

PATCH (Actualización parcial)

PATCH /api/news/id - Actualizar campos específicos

```
1  PATCH /api/news/{id}
2  Content-Type: application/json-patch+json
3  Authorization: Bearer <token>
4
5  [
6      { "op": string(enum),
7        "path": string,
8        "value": any
9      }, // Operacion JSON Patch
10
11      // Operaciones permitidas: "replace", "add", "remove"
12      // Ejemplo: { "op": "replace", "path": "/title", "value": "Nuevo titulo" }
13  ]
```

Estructura de respuesta

```
1  {
2      "id": integer,      // ID de la noticia modificada
3      "changes": string[], // Lista de campos modificados
4      "version": integer  // Nueva version del documento
5  }
```

DELETE

DELETE /api/news/id - Eliminar noticia

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

Respuesta

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

7.1.6. Aliados y Fundaciones

GET Todos (Paginado)

GET Aliados (paginado)

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

Estructura de respuesta

```
1 {
2   "content": [
3     {
4       "id": integer,           // ID unico del aliado
5       "name": string,         // Nombre de la organizacion
6       "type": string(enum),    // Tipo: FOUNDATION, COMPANY, NGO,
7       etc.
8       "description": string,   // Descripcion corta
9       "logoUrl": string(URL),  // Ruta al logo
10      "website": string(URL),   // Sitio web
11      "partnerSince": string(ISO-8601) // Fecha de inicio de alianza
12    }
13    // ... mas elementos
14  ],
15  "pageable": {
16    "sort": object,           // Informacion de ordenamiento
17    "pageSize": integer,     // Tamano de pagina solicitado
18    "pageNumber": integer    // Numero de pagina actual
19    // ... mas metadatos de paginacion
20  },
21  "totalElements": integer, // Total de resultados
22  "totalPages": integer,    // Total de paginas disponibles
23  "first": boolean,         // Si es la primera pagina
24  "last": boolean           // Si es la ultima pagina
25 }
```

GET Individual

GET /api/allies/id - Obtener aliado especifico

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

Estructura de respuesta

```
1 {
2     "id": integer,           // ID unico del aliado
3     "name": string,         // Nombre de la organizacion
4     "type": string(enum),    // Tipo: FOUNDATION, COMPANY, NGO, etc.
5     "description": string,   // Descripcion corta
6     "logoUrl": string(URL),  // Ruta al logo
7     "website": string(URL),  // Sitio web
8     "partnerSince": string(ISO-8601), // Fecha de inicio de alianza
9     "contactInfo": {        // Informacion de contacto
10         "email": string(email),
11         "phone": string
12     },
13     "projects": [           // Proyectos asociados
14         {
15             "id": integer,
16             "name": string,
17             "status": string(enum) // ACTIVE, COMPLETED, PLANNED
18         }
19     ],
20     "socialMedia": {        // Redes sociales
21         "facebook": string(URL),
22         "twitter": string(URL),
23         "linkedin": string(URL)
24     }
25 }
```

POST Creación

POST /api/allies - Crear nuevo aliado

```
1 POST /api/allies
2 Content-Type: application/json
3 Authorization: Bearer <token>
4
5 {
6     "name": string,           // Requerido: Nombre de la organizacion
7     "type": string(enum),     // Requerido: FOUNDATION, COMPANY, NGO, etc.
8     "description": string,    // Requerido: Descripcion
9     "website": string(URL),   // Opcional: Sitio web
10    "contactInfo": {          // Opcional: Informacion de contacto
11        "email": string(email),
12        "phone": string
13    },
14    "socialMedia": {          // Opcional: Redes sociales
15        "facebook": string(URL),
16        "twitter": string(URL),
17        "linkedin": string(URL)
18    }
19 }
```

Estructura de respuesta

```
1 {
2     "id": integer,           // ID generado automaticamente
3     "name": string,         // Nombre proporcionado
4     "type": string(enum),    // Tipo proporcionado
5     "createdAt": string(ISO-8601), // Timestamp de creacion
6     "status": string(enum)   // Estado inicial: PENDING_REVIEW
7 }
```

PUT (Reemplazo completo)

PUT /api/allies/id - Actualizar aliado completo

```
1 PUT /api/allies/{id}
2 Content-Type: application/json
3 Authorization: Bearer <token>
4
5 {
6     "name": string,           // Requerido: Nombre de la organizacion
7     "type": string(enum),     // Requerido: FOUNDATION, COMPANY, NGO, etc.
8     "description": string,    // Requerido: Descripcion
9     "website": string(URL),   // Opcional: Sitio web
10    "contactInfo": {          // Opcional: Informacion de contacto
11        "email": string(email),
12        "phone": string
13    },
14    "socialMedia": {           // Opcional: Redes sociales
15        "facebook": string(URL),
16        "twitter": string(URL),
17        "linkedin": string(URL)
18    }
19 }
```

Estructura de respuesta

```
1 {
2     "id": integer,           // ID del aliado actualizado
3     "name": string,         // Nombre actualizado
4     "lastUpdated": string(ISO-8601), // Timestamp de actualizacion
5     "status": string(enum)   // Estado: ACTIVE, PENDING_REVIEW, etc.
6 }
```

PATCH (Actualización parcial)

PATCH /api/allies/id - Actualizar campos específicos

```
1  PATCH /api/allies/{id}
2  Content-Type: application/json-patch+json
3  Authorization: Bearer <token>
4
5  [
6      { "op": string(enum), "path": string, "value": any }, // Operacion JSON Patch
7      // Operaciones permitidas: "replace", "add", "remove"
8      // Ejemplo: { "op": "replace", "path": "/description", "value": "Nueva
9                  descripcion" }
```

Estructura de respuesta

```
1  {
2      "id": integer,          // ID del aliado modificado
3      "changes": string[],    // Lista de campos modificados
4      "version": integer,     // Nueva version del documento
5      "modifiedAt": string(ISO-8601) // Timestamp de modificacion
6  }
```

DELETE

DELETE /api/allies/id - Eliminar aliado

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

Respuesta

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