

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
5. Arquitectura de Base de Datos:	7
6. Arquitectura Portal Content Service	8
6.1. Vision General	8
6.2. Capa API (Controladores)	8
6.3. Capa de Servicio	9
6.4. Capa de Repositorio	9
6.5. Capa de Dominio	9
6.6. Capa de Entidad	9
6.7. Flujo de peticiones	10
6.8. Endpoints REST	10
6.8.1. Actividades	10

6.8.2.	Proyectos	15
6.8.3.	Cursos	20
6.8.4.	Equipos	20
6.8.5.	Noticias	23
6.8.6.	Aliados y Fundaciones	26

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.

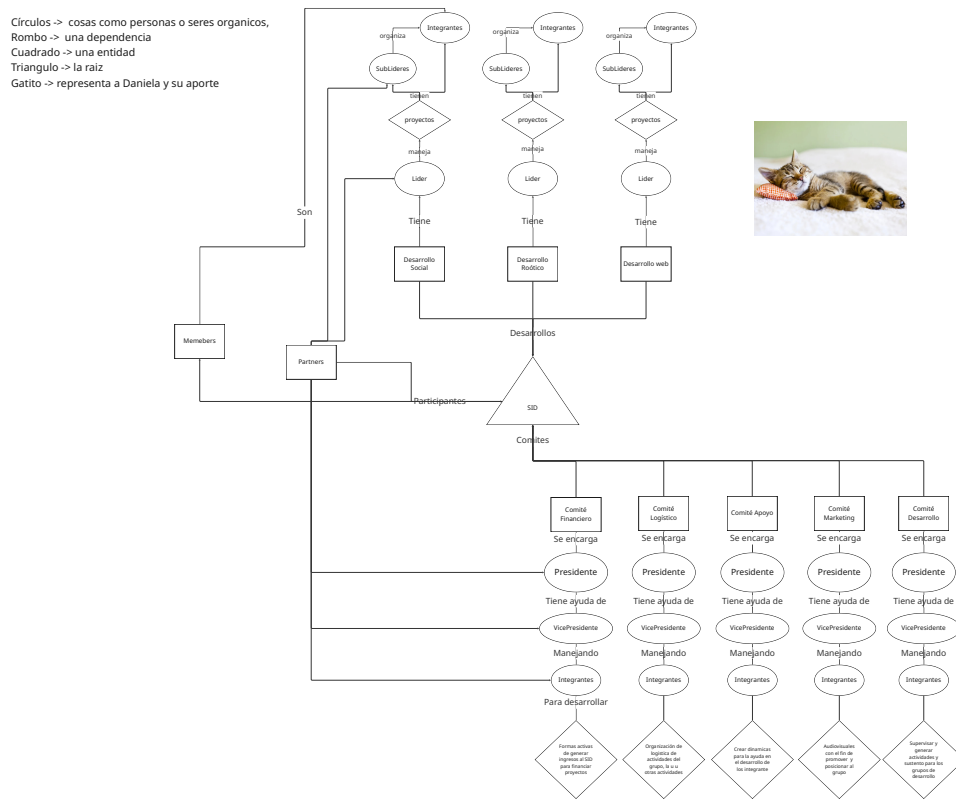


Figura 1: Diagrama de organización del SID

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:

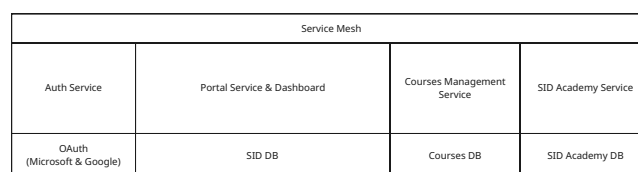


Figura 2: Diagrama de módulos del proyecto

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:

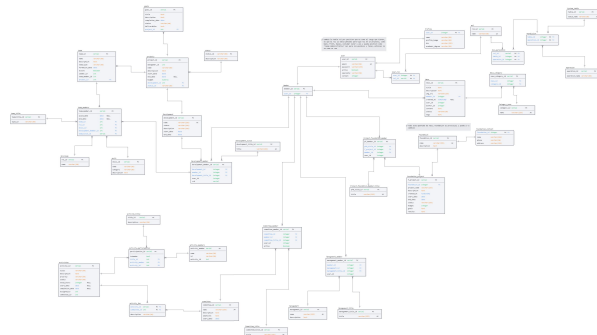


Figura 3: Diagrama de base de datos relacional

6. Arquitectura 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
- Proyectos
- Cursos
- Equipos
- Noticias
- Aliados | Fundaciones

6.1. Vision General

El diseño del sistema sigue un patrón de arquitectura estándar de múltiples capas comúnmente utilizado en aplicaciones de SpringBoot. El sistema está diseñado para mantener el desacoplamiento de funciones para garantizar la mantenibilidad, la comprobabilidad (Testing) y la escalabilidad

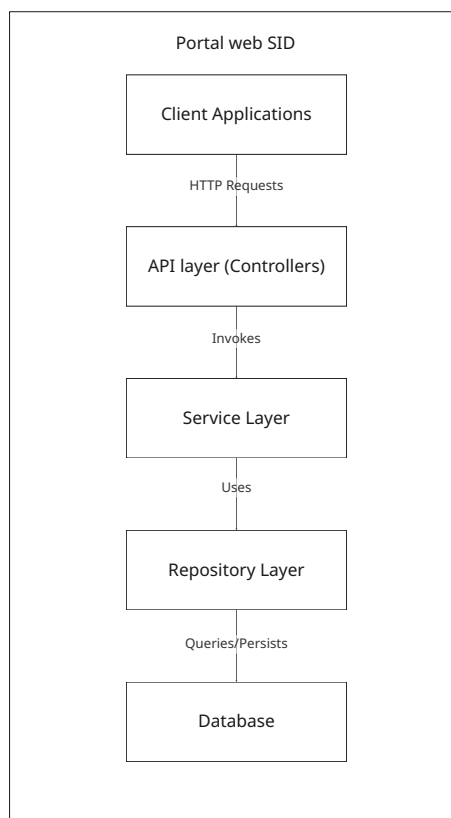


Figura 4: Diagrama de la arquitectura por capas

6.2. Capa API (Controladores)

La capa API es responsable de manejar las solicitudes HTTP entrantes, validar los datos de entrada y devolver respuestas apropiadas. Esta capa está compuesta por clases controladoras que definen los

Capa	Descripción	Responsabilidades principales
Capa API	Punto de entrada para las solicitudes del cliente	Manejo de solicitudes, formateo de respuestas, validación de entradas
Capa de Servicio	Contiene la lógica de la aplicación	Orquesta operaciones, maneja transacciones y logica de la aplicación
Capa de Repositorio	Gestiona el acceso a los datos	Proporciona una abstracción sobre el almacenamiento, maneja operaciones CRUD
Capa de Dominio	Contiene las entidades del dominio	Encapsula los datos y la lógica del negocio

Cuadro 2: Responsabilidades por capa del sistema

endpoints REST expuestos por la aplicación.

Los controladores siguen la convención REST tanto para el nombramiento de endpoints como para el uso de métodos HTTP:

- **GET** para obtener recursos
- **POST** para crear recursos
- **PUT** para actualizar recursos completamente
- **PATCH** para actualizar parcialmente recursos
- **DELETE** para eliminar recursos

6.3. Capa de Servicio

La capa de servicio implementa la lógica central de la aplicación. Se ubica entre los controladores y los repositorios, orquestando operaciones entre múltiples entidades del dominio y aplicando las reglas de negocio.

6.4. Capa de Repositorio

La capa de repositorio proporciona una abstracción sobre el mecanismo de acceso a datos. Utiliza Spring Data JPA para simplificar las operaciones con la base de datos y reducir el código repetitivo. También en los casos necesarios utiliza el patrón repositorio para hacer llamados directos a la base de datos de ser necesario

6.5. Capa de Dominio

La capa de dominio contiene las clases de entidad que representan el dominio del negocio. Estas clases son aquellas que sirven para encapsular la logica del negocio, y se pasa a service para ser manejada.

6.6. Capa de Entidad

La capa de entidad proporciona aquellas clases que representan la abstracción más cercana a la base de datos. Son quienes se usaran mediante los JPA para hacer llamados directos a la base de datos y tambien usada en los repositorios manuales

6.7. Flujo de peticiones

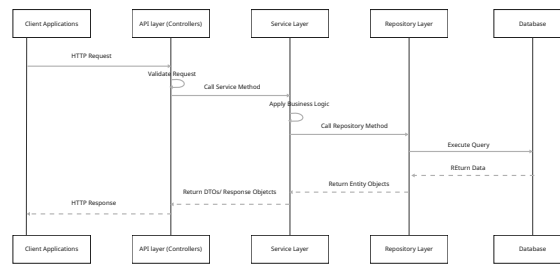


Figura 5: Diagrama de el flujo de las peticiones por capas

6.8. Endpoints REST

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

6.8.1. Actividades

GET Todos (Paginado)

GET Actividades (paginado)

```
1 GET /api/activities?page=0&size=10&sort=startDate,desc
```

Estructura de respuesta

```
1 {
2   "content": [
3     {
4       "id": integer,           // ID unico de la actividad
5       "title": string,        // Titulo de la actividad
6       "description": string,   // Descripcion de la actividad
7       "priority": string(enum), // Prioridad: "ALTA", "MEDIA", "BAJA"
8       "status": string(enum),  // Estado: "ACTIVA", "ESPERA", "
COMPLETADA", "CANCELADA"
9       "total_hours": integer,   // Total de horas estimadas
10      "start_date": string(ISO-8601), // Fecha de inicio
11      "completion_date": string(ISO-8601), // Fecha estimada de
finalizacion
12      "manager": string,        // Nombre del gestor de la actividad
13      "created_at": string(ISO-8601) // Fecha de creacion
14    }
15    // ... mas elementos
16  ],
17  "pageable": {
18    "sort": object,           // Informacion de ordenamiento
19    "pageSize": integer,      // Tamanyoo de pagina solicitado
20    "pageNumber": integer     // Numero de pagina actual
21    // ... mas metadatos de paginacion
22  },
23  "totalElements": integer,   // Total de resultados
24  "totalPages": integer,      // Total de paginas disponibles
25  "first": boolean,          // Si es la primera pagina
26  "last": boolean,           // Si es la ultima pagina
27  "size": integer            // Tamanyio de la pagina actual
28 }
```

GET Individual

GET /api/activities/id - Obtener actividad especifica

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

Estructura de respuesta

```
1 {
2     "id": integer,                // ID unico
3     "title": string,              // Titulo de la actividad
4     "description": string,        // Descripcion detallada
5     "priority": string(enum),     // Prioridad: "ALTA", "MEDIA", "BAJA"
6     "status": string(enum),       // Estado: "ACTIVA", "ESPERA", "COMPLETADA",
    "CANCELADA"
7     "total_hours": integer,       // Total de horas estimadas
8     "start_date": string(ISO-8601), // Fecha de inicio
9     "completation_date": string(ISO-8601), // Fecha estimada de finalizacion
10    "manager": string,             // Nombre del gestor
11    "members": [                  // Miembros asignados
12        {
13            "name": string,        // Nombre del miembro
14            "identify": string,    // Identificador unico
15            "isMember": boolean   // Si es miembro del SID
16        },
17        // ... mas miembros
18    ],
19    "committees": [               // Comites asociados
20        {
21            "name": string,        // Nombre del comite
22            "leader": string,      // Lider del comite
23            "description": string  // Descripcion de funcion
24        },
25        // ... mas comites
26    ]
27 }
```

POST Creación

POST /api/activities - Crear nueva actividad

```
1 POST /api/activities
2 Content-Type: application/json
3 Authorization: Bearer <token>
4
5 {
6     "title": string,           // Requerido: Titulo de la actividad
7     "description": string,     // Requerido: Descripcion
8     "priority": string(enum),  // Requerido: "ALTA", "MEDIA", "BAJA"
9     "status": string(enum),    // Requerido: "ACTIVA", "ESPERA", "
        COMPLETADA", "CANCELADA"
10    "total_hours": integer,     // Requerido: Horas estimadas
11    "start_date": string(ISO-8601), // Requerido: Fecha de inicio
12    "completion_date": string(ISO-8601), // Requerido: Fecha estimada de
        finalizacion
13    "manager": string,         // Requerido: Nombre del gestor
14    "members": [               // Opcional: Miembros asignados
15        {
16            "name": string,     // Nombre del miembro
17            "identify": string, // Identificador unico
18            "isMember": boolean // Si es miembro del SID
19        },
20        // ... mas miembros
21    ],
22    "committees": [           // Opcional: Comites asociados
23        {
24            "committee_id": integer, // ID del comite existente
25            "description": string    // Descripcion de funcion
26        },
27        // ... mas comites
28    ]
29 }
```

Estructura de respuesta

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

PUT (Reemplazo completo)

PUT /api/activities/id - Actualizar actividad completa

```
1  PUT /api/activities/{id}
2  Content-Type: application/json
3  Authorization: Bearer <token>
4
5  {
6      "title": string,                // Requerido: Nuevo titulo
7      "description": string,          // Requerido: Nueva descripcion
8      "priority": string(enum),       // Requerido: Nueva prioridad
9      "status": string(enum),        // Requerido: Nuevo estado
10     "total_hours": integer,          // Requerido: Nuevas horas estimadas
11     "start_date": string(ISO-8601), // Requerido: Nueva fecha de inicio
12     "completion_date": string(ISO-8601), // Requerido: Nueva fecha de
        finalizacion
13     "manager": string,              // Requerido: Nuevo gestor
14     "members": [                   // Requerido: Nuevos miembros
15         {
16             "name": string,
17             "identify": string,
18             "isMember": boolean
19         }
20         // ... todos los miembros
21     ],
22     "committees": [               // Requerido: Nuevos comites
23         {
24             "committee_id": integer,
25             "description": string
26         }
27         // ... todos los comites
28     ]
29 }
```

Estructura de respuesta

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

PATCH (Actualización parcial)

PATCH /api/activities/id - Actualizar campos específicos

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

Estructura de respuesta

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

DELETE

DELETE /api/activities/id - Eliminar actividad

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

Respuesta

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

6.8.2. Proyectos

GET Todos (Paginado)

GET Proyectos (paginado)

```
1  GET /api/projects?page=0&size=10&sort=startDate,desc
```

Estructura de respuesta

```
1 {
2   "content": [
3     {
4       "id": integer,           // ID unico del proyecto
5       "name": string,         // Nombre del proyecto
6       "description": string,   // Descripcion breve
7       "start_date": string(ISO-8601), // Fecha de inicio
8       "end_date": string(ISO-8601), // Fecha de finalizacion planificada
9       "budget": number,       // Presupuesto asignado
10      "development_id": string, // ID de desarrollo (1, 2, 3)
11      "status": string(enum)   // Estado: "ACTIVO", "COMPLETADO", "
EN_PAUSA", "CANCELADO"
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  "size": integer            // Tamano de la pagina actual
26 }
```

GET Individual

GET /api/projects/id - Obtener proyecto especifico

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


Estructura de respuesta

```
1 {
2     "id": integer,           // ID unico
3     "name": string,          // Nombre del proyecto
4     "description": string,    // Descripcion detallada
5     "start_date": string(ISO-8601), // Fecha de inicio
6     "end_date": string(ISO-8601), // Fecha de finalizacion planificada
7     "budget": number,        // Presupuesto asignado
8     "development_id": string, // ID de desarrollo (1, 2, 3)
9     "status": string(enum),   // Estado: "ACTIVO", "COMPLETADO", "EN_PAUSA",
10    "goals": [                // Objetivos del proyecto
11        {
12            "title": string,    // Titulo del objetivo
13            "description": string, // Descripcion del objetivo
14            "completion_date": string(ISO-8601), // Fecha de cumplimiento
15            "status": string(enum) // Estado del objetivo
16        }
17        // ... mas objetivos
18    ],
19    "teams": [                // Equipos asociados
20        {
21            "id": integer,       // ID del equipo
22            "name": string,       // Nombre del equipo
23            "description": string, // Descripcion
24            "formation_date": string(ISO-8601), // Fecha de formacion
25            "active": boolean,    // Si esta activo
26            "leader": string,     // Lider del equipo
27            "members": [         // Miembros del equipo
28                {
29                    "development_member_id": integer, // ID del miembro
30                    "name": string,                 // Nombre del miembro
31                    "join_date": string(ISO-8601), // Fecha de incorporacion
32                    "end_date": string(ISO-8601), // Fecha de salida (o null)
33                    "rol": string,                   // Rol en el equipo
34                    "title": string                  // Titulo o cargo
35                }
36                // ... mas miembros
37            ]
38        }
39        // ... mas equipos
40    ]
41 }
```

POST Creacion

POST /api/projects - Crear nuevo proyecto

```
1 POST /api/projects
2 Content-Type: application/json
3 Authorization: Bearer <token>
4
5 {
6     "name": string,                // Requerido: Nombre del proyecto
7     "description": string,         // Requerido: Descripcion
8     "start_date": string(ISO-8601), // Requerido: Fecha de inicio
9     "end_date": string(ISO-8601),  // Requerido: Fecha de finalizacion
10    "budget": number,              // Requerido: Presupuesto asignado
11    "development_id": string,       // Requerido: ID de desarrollo (1, 2, 3)
12    "status": string(enum),        // Requerido: Estado inicial
13    "goals": [                    // Opcional: Objetivos iniciales
14        {
15            "title": string,        // Titulo del objetivo
16            "description": string,  // Descripcion del objetivo
17            "completion_date": string(ISO-8601), // Fecha de cumplimiento
18            "status": string(enum)  // Estado del objetivo
19        }
20    ],
21    "teams": [                    // Opcional: Equipos iniciales
22        {
23            "name": string,         // Nombre del equipo
24            "description": string,  // Descripcion
25            "formation_date": string(ISO-8601), // Fecha de formacion
26            "active": boolean,     // Si esta activo
27            "leader": string,      // Lider del equipo
28            "members": [          // Miembros del equipo
29                {
30                    "development_member_id": integer, // ID del miembro
31                    "join_date": string(ISO-8601), // Fecha de incorporacion
32                    "end_date": string(ISO-8601), // Fecha de salida (o null)
33                    "rol": string, // Rol en el equipo
34                    "title": string // Titulo o cargo
35                }
36            ]
37        }
38    ]
39 }
```

Estructura de respuesta

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

PUT (Reemplazo completo)

PUT /api/projects/id - Actualizar proyecto completo

```
1 PUT /api/projects/{id}
2 Content-Type: application/json
3 Authorization: Bearer <token>
4
5 {
6     "name": string,
7     "description": string,
8     "start_date": string(ISO-8601),
9     "end_date": string(ISO-8601),
10    "budget": number,
11    "development_id": string,
12    "status": string(enum),
13    "goals": [
14        {
15            "title": string,
16            "description": string,
17            "completion_date": string(ISO-8601),
18            "status": string(enum)
19        }
20    ],
21    "teams": [
22        {
23            "name": string,
24            "description": string,
25            "formation_date": string(ISO-8601),
26            "active": boolean,
27            "leader": string,
28            "members": [
29                {
30                    "development_member_id": integer,
31                    "join_date": string(ISO-8601),
32                    "end_date": string(ISO-8601),
33                    "rol": string,
34                    "title": string
35                }
36            ]
37        }
38    ]
39 }
```

Estructura de respuesta

```
1 {
2     "id": integer,                // ID del proyecto actualizado
3     "name": string,              // Nombre actualizado
4     "last_updated": string(ISO-8601) // Timestamp de actualizacion
5 }
```

PATCH (Actualizacion parcial)

PATCH /api/projects/id - Actualizar campos específicos

```
1  PATCH /api/projects/{id}
2  Content-Type: application/json-patch+json
3  Authorization: Bearer <token>
4
5  [
6      {
7          "op": string(enum),
8          "path": string,
9          "value": any
10     }
11     // Ejemplo: { "op": "replace", "path": "/status", "value": "COMPLETADO" }
12 ]
```

Estructura de respuesta

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

DELETE

DELETE /api/projects/id - Eliminar proyecto

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

Respuesta

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

6.8.3. Cursos

6.8.4. Equipos

GET Todos (Paginado)

GET Equipos (paginado)

```
1  GET /api/teams?page=0&size=10&sort=formation_date,desc
```

Estructura de respuesta

```
1 {
2   "content": [
3     {
4       "id": integer,           // ID unico del equipo
5       "name": string,         // Nombre del equipo
6       "description": string,   // Descripcion del equipo
7       "active": boolean,      // Si el equipo esta activo
8       "leader": string        // Nombre del lider
9     }
10    // ... mas elementos
11  ],
12  "pageable": {
13    "sort": object,
14    "pageSize": integer,
15    "pageNumber": integer
16  },
17  "totalElements": integer,
18  "totalPages": integer,
19  "first": boolean,
20  "last": boolean,
21  "size": integer
22 }
```

GET Individual

GET /api/teams/id - Obtener equipo especifico

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

Estructura de respuesta

```
1 {
2   "id": integer,           // ID unico del equipo
3   "name": string,         // Nombre del equipo
4   "description": string,   // Descripcion
5   "active": boolean,      // Si esta activo
6   "leader": string,       // Nombre del lider
7   "members": [
8     {
9       "name": string,      // Nombre del miembro
10      "rol": string,        // Rol asignado
11      "title": string       // Titulo o cargo
12    }
13    // ... mas miembros
14  ]
15 }
```

POST Creacion

POST /api/teams - Crear nuevo equipo

```
1 POST /api/teams
2 Content-Type: application/json
3 Authorization: Bearer <token>
4
5 {
6   "project_id": integer,      // Requerido: ID del proyecto al que pertenece
7   "name": string,             // Requerido: Nombre del equipo
8   "description": string,      // Requerido: Descripcion
9   "formation_date": string(ISO-8601), // Requerido: Fecha de formacion
10  "active": boolean,           // Requerido: Estado de actividad
11  "leader": string,            // Requerido: Nombre del lider
12  "members": [
13    {
14      "development_member_id": integer, // Requerido
15      "join_date": string(ISO-8601),    // Requerido
16      "end_date": string(ISO-8601),     // Opcional
17      "rol": string,                    // Requerido
18      "title": string                  // Requerido
19    }
20    // ... mas miembros
21  ]
22 }
```

Estructura de respuesta

```
1 {
2   "id": integer,           // ID generado automaticamente
3   "name": string,          // Nombre del equipo creado
4   "created_at": string(ISO-8601) // Timestamp de creacion
5 }
```

PATCH (Actualizacion parcial)

PATCH /api/teams/id - Actualizar campos especificos

```
1 PATCH /api/teams/{id}
2 Content-Type: application/json-patch+json
3 Authorization: Bearer <token>
4
5 [
6   {
7     "op": string(enum),
8     "path": string,
9     "value": any
10  }
11  // Ejemplo: { "op": "replace", "path": "/name", "value": "Equipo Actualizado" }
12 ]
```

Estructura de respuesta

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

DELETE

DELETE /api/teams/id - Eliminar equipo

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

Respuesta

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

6.8.5. 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       "description": string,   // Breve descripcion que se muestra
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 especifica

```
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   "description": string    // Breve descripcion
5   "content": string(HTML o .md), // Contenido completo en HTML o .md
6   "publishDate": string(ISO-8601), // Fecha de publicacion
7   "author": string,       // Nombre del autor
8   "imageUrl": string(URL), // Ruta a la imagen destacada
9   "tags": string[]        // Array de etiquetas
10 }
```

POST Creacion

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     "description": string      // Requerido: Descripcion de la noticia
8     "content": string(URL:.md) // Requerido: Ruta o contenido en HTML o .md
9     "tags": string[],         // Array de etiquetas
10    "imageURL": string(URL)    // Ruta a la imagen
11 }
```

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 (HTML o .md)
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 (Actualizacion parcial)

PATCH /api/news/id - Actualizar campos especificos

```
1  PATCH /api/news/{id}
2  Content-Type: application/json-patch+json
3  Authorization: Bearer <token>
4
5  [
6      {
7          "op": string(enum),
8          "path": string,
9          "value": any
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
```

6.8.6. Aliados y Fundaciones

GET Todos (Paginado)

GET Fundaciones (paginado)

```
1  GET /api/foundations?page=0&size=10&sort=name,asc
```

Estructura de respuesta

```
1 {
2   "content": [
3     {
4       "id": integer,          // ID unico de la fundacion
5       "name": string,         // Nombre
6       "description": string,  // Descripcion breve
7       "logo_url": string(URL) // Logo de la fundacion
8     }
9     // ... mas elementos
10  ],
11  "pageable": {
12    "sort": object,
13    "pageSize": integer,
14    "pageNumber": integer
15  },
16  "totalElements": integer,
17  "totalPages": integer,
18  "first": boolean,
19  "last": boolean,
20  "size": integer
21 }
```

GET Individual

GET /api/foundations/id - Obtener fundacion especifica

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

Estructura de respuesta

```
1 {
2   "id": integer,          // ID unico
3   "name": string,         // Nombre
4   "description": string,  // Descripcion
5   "logo_url": string(URL) // Logo
6 }
```

GET Dashboard (detallado)

GET /api/foundations/id/dashboard - Informacion extendida

```
1 GET /api/foundations/{id}/dashboard
```

Estructura de respuesta

```
1 {
2   "id": integer,
3   "name": string,
4   "description": string,
5   "logo_url": string(URL),
6   "phone": string,
7   "address": string,
8   "location": string,
9   "website": string,
10  "quantity_projects":integer, // aqui podriamos hacer algun apartado visual
    bonito
11  // Puede agregarse un apartado adicional para redes sociales
12 }
```

POST Creacion

POST /api/foundations - Crear fundacion

```
1 POST /api/foundations
2 Content-Type: application/json
3 Authorization: Bearer <token>
4
5 {
6   "name": string,           // Requerido
7   "description": string,    // Requerido
8   "logo_url": string(URL), // Requerido
9   "phone": string,          // Opcional
10  "address": string,         // Opcional
11  "location": string,        // Opcional
12  "website": string          // Opcional
13 }
```

Estructura de respuesta

```
1 {
2   "id": integer,           // ID generado
3   "name": string,          // Nombre
4   "created_at": string(ISO-8601) // Timestamp
5 }
```

PATCH (Actualizacion parcial)

PATCH /api/foundations/id - Actualizar campos especificos

```
1  PATCH /api/foundations/{id}
2  Content-Type: application/json-patch+json
3  Authorization: Bearer <token>
4
5  [
6    {
7      "op": string(enum),
8      "path": string,
9      "value": any
10   }
11   // Ejemplo: { "op": "replace", "path": "/phone", "value": "+51 999999999" }
12 ]
```

Estructura de respuesta

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

DELETE

DELETE /api/foundations/id - Eliminar fundacion

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

Respuesta

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

GET Proyectos por fundacion

GET /api/foundations/projects - Obtener proyectos por fundacion

```
1  GET /api/foundations/projects
```

Estructura de respuesta

```
1  [
2    {
3      "foundation": string,      // Nombre de la fundacion
4      "projects": [
5        {
6          "name": string,          // Nombre del proyecto
7          "description": string,    // Descripcion
8          "created_at": string(ISO-8601),
9          "start_date": string(ISO-8601),
10         "end_date": string(ISO-8601),
11         "status": string(enum),    // Ej: ACTIVA
12         "budget": number,          // Presupuesto
13         "leader": string,          // Nombre del lider
14         "members": [
15           {
16             "name": string,        // Nombre del miembro
17             "title": string        // Cargo
18           }
19         ]
20       }
21     ]
22   }
23 ]
```