



SIME

Documentación Funcional-Técnica Completa

Para: SIEMPRE

Sistema: SIME

Fecha: 26 de Junio de 2025

Presentado por:

Renzo Antonioli

ANALISTA FUNCIONAL TECNICO & DEV

rantonioli@zeron.com.ar | +54 11 3566-5266

1 · VISIÓN & OBJETIVOS	4
1.1 <i>Propósito Estratégico</i>	4
1.2 <i>Objetivos Específicos</i>	4
1.3 <i>Principios Rectores</i>	4
1.4 <i>Actores & Beneficios</i>	4
2 · ARQUITECTURA DE ALTO NIVEL	5
2.1 <i>Capas funcionales</i>	5
2.2 <i>Vista Lógica – Doble Perspectiva</i>	5
2.2.1 <i>Vista Conceptual</i>	5
2.2.2 <i>Vista Lógica Detallada</i>	6
2.3 <i>Principios arquitectónicos</i>	8
2.4 <i>Flujos de datos representativos</i>	8
2.5 <i>Escenarios de resiliencia y escalabilidad</i>	8
3 · CATÁLOGO DE MÓDULOS	9
4 · DETALLE INDIVIDUAL DE MÓDULOS	10
4.1 <i>Identidad & Roles (AUTH)</i>	10
1. <i>Objetivo</i>	10
2. <i>Funcionalidades</i>	10
3. <i>Componentes internos</i>	10
4. <i>Dependencias</i>	10
5. <i>Casos de Uso prioritarios</i>	11
6. <i>Flujo principal AU-02</i>	11
7. <i>Interfaz de usuario (6 pantallas)</i>	11
4.2 <i>Catálogos Maestros (CATS)</i>	11
1. <i>Objetivo</i>	11
2. <i>Funcionalidades</i>	12
3. <i>Componentes internos</i>	12
4. <i>Dependencias</i>	12
5. <i>Casos de Uso prioritarios</i>	12
6. <i>Flujo Bulk-Import (C-02)</i>	12
7. <i>Interfaz de usuario (5 pantallas)</i>	12
4.3 <i>Programas (PROG)</i>	13
1. <i>Objetivo</i>	13
2. <i>Funcionalidades</i>	13
3. <i>Componentes internos</i>	13
4. <i>Dependencias</i>	13
5. <i>Casos de Uso prioritarios</i>	13
6. <i>Flujo P-01 (Wizard 4 pasos)</i>	13
7. <i>Interfaz de usuario (8 pantallas)</i>	14
Cambios clave en UI	14
4.4 <i>Matriz de Indicadores (IND)</i>	15
1. <i>Objetivo</i>	15
2. <i>Funcionalidades</i>	16
3. <i>Componentes internos</i>	16
4. <i>Dependencias</i>	16
5. <i>Casos de Uso prioritarios</i>	16
6. <i>Flujo I-02</i>	16
7. <i>Interfaz de usuario (7 pantallas)</i>	16
4.5 <i>Evaluación & Agenda de Mejoras (EVAL)</i>	17
1. <i>Objetivo</i>	17
2. <i>Funcionalidades</i>	17
3. <i>Componentes internos</i>	17
4. <i>Dependencias</i>	17
5. <i>Casos de Uso prioritarios</i>	17
6. <i>Flujo Kanban (E-03)</i>	17
7. <i>Interfaz de usuario (6 pantallas)</i>	17
4.6 <i>Reportes & Dashboards (REP)</i>	18

1. Objetivo	18
2. Funcionalidades.....	18
3. Componentes internos.....	18
4. Dependencias	18
5. Casos de Uso prioritarios	18
6. Flujo R-02 (scheduler)	18
7. Interfaz de usuario (8 pantallas).....	19
5 · MODELO DE DATOS & DDL COMPLETO.....	20
<i>Convenciones</i>	20
5.1 Módulo Identidad & Roles	20
5.2 Catálogos Maestros	21
5.3 Programas y Prestaciones	22
5.3.1 Esquema estándar	22
5.3.2 Diseño de campos personalizados (EAV opcional).....	22
5.3.3 Flujo de creación de campo.....	23
5.3.4 Ventajas del modelo híbrido	23
5.4 Indicadores y Series	24
5.5 Evaluaciones y Mejoras	25
5.6 Reportes y Dashboards	26
5.7 Índices, Vistas y Otros Artefactos	27
6 · API REST v1	29
<i>Convenciones generales</i>	29
6.1 Esquema de Seguridad	29
6.2 Tags disponibles	29
6.3 Endpoints clave	29
6.3.1 Auth	29
6.3.2 Catálogos.....	30
6.3.3 Programas.....	30
6.3.4 Indicadores	30
6.3.5 Evaluaciones.....	31
6.3.6 Reportes & Dashboards.....	31
6.4 Componentes (schemas)	31
6.5 Respuestas genéricas	32
6.6 Versionado y compatibilidad	33
7 · DIAGRAMAS DE FLUJO & SECUENCIA	34
7.1 Login con MFA (Auth)	34
7.2 Alta de Programa (Wizard 4 pasos)	34
7.3 Creación de Campo Personalizado	35
7.4 Carga Masiva de Series	35
7.5 Generación de Alerta de Meta Incumplida	35
7.6 Proceso de Evaluación y Cierre de Acción	35
7.7 Creación y Compartición de Dashboard	36
7.8 Scheduler de Reporte Programado	36
8 SEGURIDAD, AUDITORÍA & CUMPLIMIENTO	38
8.1 Autenticación y Gestión de Identidades	38
8.2 Autorización y Control de Acceso	38
8.3 Cifrado y Protección de Datos	38
8.4 Monitorización, Logging y Auditoría	38
8.5 Respaldo y Recuperación ante Desastre	39
8.6 DevSecOps y Gestión de Vulnerabilidades	39
8.7 Cumplimiento Legal & Privacidad	39

1 · Visión & Objetivos

1.1 Propósito Estratégico

El **Sistema de Monitoreo y Evaluación (SIME)** se concibe como el núcleo de evidencia para la toma de decisiones públicas. No sólo consolida indicadores y reportes; articula la **gestión por resultados** desde el diseño hasta la mejora continua.

1.2 Objetivos Específicos

1. **Monitorear** resultados en tiempo casi real y comparar contra metas.
2. **Evaluar** impacto y eficiencia mediante metodologías rigurosas.
3. **Gestionar** planes de mejora con responsables y plazos claros.
4. **Transparentar** el avance a autoridades, organismos de control.
5. **Aprender** institucionalmente para iterar políticas futuras.

1.3 Principios Rectores

Principio	Aplicación práctica
Interoperabilidad	APIs REST y eventos estándar; integración con DW ministerial.
No hard-coding	Catálogos administrables y campos personalizados dinámicos.
Gobernanza de datos	Linaje, políticas de calidad y roles de dato definidos.
Seguridad by design	Controles integrados en cada capa; cifrado y auditoría.
Usabilidad	Experiencias coherentes, accesibles y orientadas a tareas.
Escalabilidad	Arquitectura modular y resiliente a picos de carga.

1.4 Actores & Beneficios

Actor	Beneficio
Coordinador de Programa	Visualiza KPIs y reporta avance.
Operador M&E	Carga datos masivos y recibe alertas.
Evaluator Externo	Accede a información consolidada para estudios.
Autoridad Ministerial	Obtiene dashboards estratégicos para decisiones.

Visión: convertir a SIME en la referencia nacional para políticas públicas basadas en evidencia, fomentando agilidad, transparencia y mejora continua en toda la administración.

SIME es la plataforma oficial que **monitorea, evalúa y mejora** programas públicos a partir de la evidencia.

Objetivos clave: seguimiento de indicadores, evaluación de impacto, agenda de mejoras y rendición de cuentas transparente

2 · Arquitectura de Alto Nivel

La arquitectura del SIME se concibe como un **sistema multicapa orientado a dominios**, organizado para separar responsabilidades, facilitar la evolución independiente de cada módulo y asegurar la gobernanza de la información.

2.1 Capas funcionales

Capa	Responsabilidad esencial
Presentación	Interacción con usuarios: captura de datos, visualización de indicadores, construcción de reportes.
Servicios de Aplicación	Orquestación de los casos de uso, control de transacciones y exposición de contratos externos (API/Eventos).
Contextos de Dominio	Implementación de reglas de negocio específicas (Identidad, Catálogos, Programas, Indicadores, Evaluación, Reportes).
Gestión de Datos	Persistencia estructurada (relacional) y no estructurada (archivos), versionado, auditoría y backups.
Servicios Transversales	Autenticación & autorización, mensajería/event-bus, registro de métricas, bitácora de auditoría, planificación de tareas.
Integración Externa	Intercambio con Data Warehouse institucional, sistemas de visualización corporativos y otros entes ministeriales.

2.2 Vista Lógica – Doble Perspectiva

Para facilitar la comprensión tanto de públicos no técnicos como de equipos de sistemas, se presentan **dos niveles de abstracción**:

2.2.1 Vista Conceptual

graph LR
User[Usuarios (operadores, coordinadores, autoridades)]
SIME[SIME Plataforma]
Cat[Catálogos]
Prog[Programas]
Ind[Matriz Indicadores]
Eval[Evaluación & Mejora]
Dash[Reportes & Dashboards]
DW[Padrones / Data Warehouse]
Pub[Ciudadanía / Toma de decisiones]

User --> |Interacción web| SIME
SIME --> Cat
SIME --> Prog
SIME --> Ind
SIME --> Eval
SIME --> Dash
Ind --> |Cruce de datos| DW
Dash --> Pub


Cómo leerla:


- El rectángulo central agrupa todos los servicios de la plataforma.
- Flechas señalan **flujos principales**: entrada de datos, análisis y publicación.
- Usuarios interactúan solo con la plataforma; los componentes internos cooperan "puertas adentro".


2.2.2 Vista Lógica Detallada


Leyenda de colores


-  Presentación
-  Servicios de Aplicación
-  Contextos de Dominio
-  Gestión de Datos
-  Integración Externa

```
graph TD
%%----- NODOS -----%%
subgraph Presentación 
    FE[Cliente Web / BI]
end

subgraph Servicios de Aplicación 
    S_Auth[Servicio Identidad]
    S_Cat[Servicio Catálogos]
    S_Prog[Servicio Programas]
    S_Ind[Servicio Indicadores]
    S_Eval[Servicio Evaluación]
    S_Rep[Servicio Reportes]
end

subgraph Contextos de Dominio 
    D_Auth[Dominio Identidad]
    D_Cat[Dominio Catálogos]
    D_Prog[Dominio Programas]
    D_Ind[Dominio Indicadores]
    D_Eval[Dominio Evaluación]
    D_Rep[Dominio Reportes]
end

subgraph Gestión de Datos 
    DB[(BD Relacional)]
    OBJ[(Almacén Objetos)]
    LOG[(Bitácora)]
end

subgraph Integración Externa 
    DW[(Data Warehouse)]
end
```

```
BI[(Plataforma BI Corporativa)]
end
```

```
%%----- FLUJOS -----%%
```

```
FE -->|Solicitudes JWT| S_Auth
FE --> S_Cat
FE --> S_Prog
FE --> S_Ind
FE --> S_Eval
FE --> S_Rep
```

```
S_Auth --> D_Auth
S_Cat --> D_Cat
S_Prog --> D_Prog
S_Ind --> D_Ind
S_Eval --> D_Eval
S_Rep --> D_Rep
```

```
D_Auth --> DB
D_Cat --> DB
D_Prog --> DB
D_Ind --> DB
D_Eval --> DB
D_Rep --> DB
D_Rep --> OBJ
D_Auth --> LOG
D_Prog --> LOG
```

```
D_Ind <.-|Padrones| DW
D_Rep -.->|Embed / Exporta| BI
```

```
%%----- ESTILOS -----%%
```

```
classDef presentation fill:#FFF4C3,stroke:#333;
classDef application fill:#CDE8FF,stroke:#333;
classDef domain fill:#D4F8D4,stroke:#333;
classDef data fill:#FFD5D1,stroke:#333;
classDef integration fill:#E8E8E8,stroke:#333;
```

```
class FE presentation;
class S_Auth,S_Cat,S_Prog,S_Ind,S_Eval,S_Rep application;
class D_Auth,D_Cat,D_Prog,D_Ind,D_Eval,D_Rep domain;
class DB,OBJ,LOG data;
class DW,BI integration;
```

La **agrupación cromática** deja claro:

- **Presentación** (●): único punto de interacción con usuarios.
- **Servicios de Aplicación** (●): exponen casos de uso y orquestan dominios.
- **Dominios** (●): implementan reglas de negocio y manipulan entidades.
- **Gestión de Datos** (●): persistencia estructurada, objetos y bitácora.

- **Integración Externa** (🔗): consumo y publicación de datos con sistemas ministeriales.

Colores y agrupaciones representan las capas lógicas del sistema, mostrando **quién llama a quién y dónde se almacena cada tipo de dato**.

2.3 Principios arquitectónicos

1. **Modularidad acoplada a dominios** – cada contexto es responsable de sus entidades y casos de uso; las dependencias se gestionan a través de contratos estables.
2. **Evolución independiente** – los contextos pueden escalar o desplegarse sin afectar los demás; las migraciones de datos se orquestan mediante versionado.
3. **Gobernanza de datos** – toda transformación queda registrada; los linajes facilitan auditoría, reproducibilidad de reportes y trazabilidad de indicadores.
4. **Seguridad transversal** – autenticación centralizada, control de acceso unificado y bitácora inmutable de eventos.
5. **Observabilidad integrada** – métricas de rendimiento, salud y uso son recolectadas y expuestas de forma homogénea para cada servicio.

2.4 Flujos de datos representativos

- **Carga de serie de indicadores:** un operador envía un lote → Servicios de Aplicación de Indicadores valida negocio → Dominio Indicadores registra valores → Gestión de Datos persiste → Evento *serie.cargada* notifica a Reportes para recalcular widgets.
- **Generación de reporte programado:** planificador interno dispara tarea → Dominio Reportes extrae datos → genera archivo en Almacén de Objetos → envía notificación a los destinatarios → registra acción en Bitácora.
- **Sincronización con DW:** tarea nocturna consume padrones actualizados → Dominio Indicadores recalcula coberturas → nuevas métricas publicadas a dashboards.

2.5 Escenarios de resiliencia y escalabilidad

- **Alta disponibilidad** mediante replicación de servicios y almacenamiento; balanceo de carga en la capa de Servicios de Aplicación.
- **Desacople temporal** a través de colas/eventos para procesos intensivos (cálculo masivo, generación de reportes), evitando bloqueo de peticiones transaccionales.
- **Versioning y migraciones seguras:** las evoluciones de esquema se aplican mediante migraciones controladas, preservando el historial de datos y minimizando paros.

3 · Catálogo de Módulos

#	Módulo	Descripción corta	Principales Entidades
1	Administración de Usuarios & Roles	Autenticación, MFA, RBAC, auditoría	usuario, rol, permiso, usuario_rol, rol_permiso, audit_log
2	Catálogos Maestros	ABM de valores de referencia (provincias, organismos...)	provincia, tipo_programa, organismo, tipo_prestacion, frecuencia_medicion
3	Programas (Relevamiento SPS)	Registro integral de programas y prestaciones	programa, prestacion
4	Matriz de Indicadores	Catálogo + metas + series de datos	indicador, indicador_programa, serie_indicador
5	Evaluación & Agenda de Mejoras	Estudios, hallazgos, acciones	evaluacion, recomendacion, mejora_seguimiento
6	Reportes & Dashboards	Constructor visual, scheduler, BI embed	reporte_def, programacion_reporte, dashboard, widget, dashboard_permiso

4 · Detalle Individual de Módulos

Para cada módulo se incluye:

1. **Objetivo estratégico** – valor que aporta al negocio.
2. **Funcionalidades** – catálogo completo de capacidades.
3. **Componentes internos** – bloques lógicos que implementan las funcionalidades.
4. **Dependencias** – servicios o datos externos necesarios.
5. **Casos de uso prioritarios** – tabla RAISE (*Rol, Acción, Interacción, Sistema, Excepción*).
6. **Flujo principal** – diagrama textual paso a paso.
7. Interfaz de Usuario – pantallas (cantidad, propósito, navegación)

4.1 Identidad & Roles (AUTH)

1. Objetivo

Garantizar identidad confiable, control de acceso basado en roles y trazabilidad completa de eventos de seguridad.

2. Funcionalidades

- Alta, edición, suspensión y baja lógica de usuarios.
- Autenticación primaria (usuario + contraseña **BCrypt**) y secundaria (TOTP).
- Emisión, refresco y revocación de tokens JWT (access / refresh).
- Administración de roles, permisos granulares y herencia de roles.
- Detección de intentos de acceso indebido y bloqueo automático.
- Exportación de bitácora en formato CEF/JSON para SIEM corporativo.

3. Componentes internos

Componente	Responsabilidad	Datos manejados
User Manager	CRUD + políticas de contraseña	usuario
RBAC Engine	Resolución de permisos, cache ACL	rol, permiso, rol_permiso
MFA Service	Generación QR + validación TOTP	usuario.mfa_secret
Token Issuer	Firmado / blacklist de tokens	—
Audit Trail	Persistencia eventos → audit_log	audit_log

4. Dependencias

Base de datos principal, servicio de correo, servicio de mensajería (alertas), reloj confiable (NTP).

5. Casos de Uso prioritarios

CU	Nombre	Actor	Flujo feliz	Excepciones
AU-01	Registrar usuario	Administrador	Datos válidos → Email activación	Email repetido, contraseña débil
AU-02	Login con MFA	Usuario final	Credenciales correctas + TOTP → JWT	Usuario bloqueado, TOTP incorrecto
AU-03	Asignar rol	Administrador	Vincula rol → ACL cache refresh	Rol inexistente
AU-04	Forzar logout global	Seguridad	Blacklist tokens → inválidos	Sesión ya expirada

6. Flujo principal AU-02

Usuario → /auth/login (email + pass)
 Sistema → Verifica hash / estado / mfa_enabled
 └ Si requiere MFA → /auth/verify-totp
 Usuario → Ingresar código
 Sistema → Valida → Emite tokens access+refresh
 Registra evento LOGIN_SUCCESS en audit_log

7 Interfaz de usuario (6 pantallas)

#	Pantalla	Propósito	Navegación principal
1	Login	Captura credenciales básicas.	Entrada pública /login.
2	Verificación MFA	Ingreso de código TOTP	Redirigida desde Login si mfa_enabled.
3	Mi Perfil	Ver / editar datos y <i>Cambiar contraseña</i> .	Dropdown usuario → /profile.
4	Gestión de Usuarios	DataGrid CRUD, bulk actions, filtro por rol / estado.	Menú <i>Seguridad</i> → /users.
5	Roles & Permisos	Editor jerárquico rol → permisos, clonación.	/roles.
6	Bitácora de Seguridad	Búsqueda / filtros (tipo evento, IP, fecha), export CSV.	/audit-log.

4.2 Catálogos Maestros (CATS)

1. Objetivo

Eliminar valores hard-coded y permitir que las listas de referencia evolucionen sin despliegues.

2. Funcionalidades

- Mantenimiento CRUD de provincias, organismos, tipos de programa, tipos de prestación, frecuencias de medición.
- Importador masivo CSV/XLSX con mapeo dinámico de columnas y pre-visualización.
- **Soft delete** con restauración y trazabilidad (deleted_at).
- Validaciones de unicidad, relaciones y reglas de negocio (p.ej. código ISO-3166-2 para provincias).
- Buscador AJAX con filtros (activo, ambito, etc.).

3. Componentes internos

Componente	Epics cubiertos
Catalog API	Endpoints REST / GraphQL, paginación, sorting
Bulk Importer	Parseo, validación, carga transaccional, rollback
Catalog UI	DataGrid editable, notificaciones en tiempo real

4. Dependencias

AUTH (rol Admin), base de datos relacional.

5. Casos de Uso prioritarios

CU	Descripción	Actor	Flujo feliz	Excepciones
C-01	Crear Provincia	Admin	POST /provincias → 201	Código ISO duplicado
C-02	Importar Organismos	Admin	Subir CSV → Preview → Confirmar	CSV mal formado
C-03	Deshabilitar Tipo Prestación	Admin	PATCH activo=false	Prestaciones activas referencian tipo

6. Flujo Bulk-Import (C-02)

Admin → UI Import → Sube CSV

Sistema → Parse CSV → Valida fila a fila

| ↳ Errores se muestran en preview con línea y motivo

Admin → Confirma carga

Sistema → Inicia transacción, INSERT batch

| ↳ Si error → rollback completo

Registra resumen (OK/errores) y envía notificación

7 Interfaz de usuario (5 pantallas)

#	Pantalla	Propósito	Navegación
1	Panel de Catálogos	Selector de tipo catálogo (cards).	/catalogs.
2	Listado DataGrid	Visualizar registros, búsqueda instantánea, paginación.	/catalogs/{tipo}.

3	Formulario CRUD	Alta / edición con validación en línea.	Modal O /catalogs/{tipo}/new.
4	Histórico / Papelera	Ítems deshabilitados; opción restaurar.	Tab secundario en listado.
5	Wizard Importador	Paso 1 subir archivo → Paso 2 mapping → Paso 3 preview → Confirmar.	/catalogs/{tipo}/import.

4.3 Programas (PROG)

1. Objetivo

Consolidar la descripción formal de programas públicos y sus prestaciones para posterior monitoreo y evaluación.

2. Funcionalidades

- **Wizard 4 pasos** con guardado parcial y validaciones dependientes.
- Control de estados (Borrador, Revisión, Activo, Cerrado).
- Módulo de Prestaciones (1-N) con tipo, población objetivo y unidad de medida.
- Historial detallado (programa_hist) con diffs campo a campo.
- Buscador con filtros combinados (tipo, organismo, estado, provincia).

3. Componentes internos

Componente	Descripción
Wizard Manager	Orquesta pasos y persiste parcialmente (localStorage + BD)
Prestación Service	CRUD + validación contra catálogos
Lifecycle Controller	Cambios de estado y reglas de transición
History Tracker	Trigger que guarda JSON diff en programa_hist

4. Dependencias

CATS, AUTH, IND (para validez de indicadores).

5. Casos de Uso prioritarios

CU	Nombre	Actor	Flujo feliz	Excepciones
P-01	Crear programa	Coordinador	Completa wizard → Borrador	Falta metas
P-02	Aprobar programa	SuperAdmin	PATCH Activo	Indicadores incompletos
P-03	Cerrar programa	Coordinador	Bloquea ediciones	Prestaciones activas

6. Flujo P-01 (Wizard 4 pasos)

Paso 1 → Datos generales (nombre, objetivo, tipo, organismo, provincia)

Paso 2 → Añadir prestaciones (tabla inline, validación poblac. > 0)

Paso 3 → Vincular indicadores disponibles según tipo programa

Paso 4 → Resumen + validaciones cruzadas → Guardar (estado=Borrador)

Notifica via websockets a Indicadores para crear plantillas serie

7. Interfaz de usuario (8 pantallas)

#	Pantalla	Propósito	Navegación
1	Lista Programas	Vista tabla + filtros (estado, organismo, tipo, <i>campos personalizados via columnas dinámicas</i>).	/programs
2	Wizard – Paso 1	Datos generales + sección <i>Campos Extra</i> (inputs renderizados dinámicamente).	/programs/new/step-1
3	Wizard – Paso 2	Prestaciones (tabla inline).	/step-2
4	Wizard – Paso 3	Vinculación de indicadores (selector múltiple).	/step-3
5	Wizard – Paso 4	Revisión (incluye diff de <i>Campos Extra</i>) → Confirmación.	/step-4
6	Detalle Programa	Tabs: General, Prestaciones, Indicadores, Historial, <i>Campos Extra</i> .	/programs/{id}
7	Prestaciones Manager	CRUD prestaciones.	Sub-tab Prestaciones
8	Historial de Cambios	Diff visual antes/después (muestra cambios en metadata/EAV).	Tab Historial
9	Gestión de Campos Personalizados	CRUD definiciones (programa_field_def), tipo de dato, orden.	Menú <i>Configuración</i> → /programs/custom-fields
10	Asignar Valores Extra	Vista de mantenimiento masivo: filas programas × columnas extra, edición rápida tipo hoja de cálculo.	/programs/custom-values

Cambios clave en UI

- El **Wizard Paso 1** detecta definiciones en programa_field_def y genera inputs (text, number, date, checkbox).

- Lista Programas permite al usuario elegir mostrar u ocultar columnas de campos extra (persistido en localStorage).
- En **Detalle Programa**, el tab *Campos Extra* muestra valores y ofrece botón *Editar* si el rol tiene permiso.
- **Gestión de Campos Personalizados** requiere rol **Admin** o **Designer**: al crear un nuevo campo, se actualiza schema cache y se emite evento `custom_field.created` para que la SPA recargue formularios sin refrescar.

Flujo para crear nueva variable (campo personalizado)

Admin → Pantalla Gestión de Campos → “Nuevo Campo”
 Sistema → Valida slug único / tipo válido → INSERT programa_field_def
 SPA emite evento → Wizard y Detalle recargan definiciones
 Coordinador → Edita Programa → Nuevo input visible → Guardar
 API → Inserta/actualiza programa_field_val o metadata JSONB
 Historial registra cambio y, si procede, widget Dashboard se marca para recálculo

#	Pantalla / Vista	Propósito (qué muestra o permite)	Ruta / Navegación
1	Lista Programas	Tabla de programas con filtros por estado, organismo, tipo y búsqueda rápida	/programs
2	Wizard – Paso 1	Ingreso de datos generales del programa (nombre, tipo, organismo, provincia, etc.)	/programs/new/step-1
3	Wizard – Paso 2	Gestión de prestaciones (tabla inline, altas/ediciones)	/programs/new/step-2
4	Wizard – Paso 3	Vinculación múltiple de indicadores y definición de línea base / metas	/programs/new/step-3
5	Wizard – Paso 4	Revisión integral y confirmación del programa antes de guardar	/programs/new/step-4
6	Detalle Programa	Ficha con pestañas: General, Prestaciones, Indicadores, Historial	/programs/{id}
7	Prestaciones Manager	CRUD de prestaciones específicas del programa (sub-pestaña en Detalle)	Sub-tab dentro de /programs/{id}
8	Historial de Cambios	Vista diff antes/después de cada versión del programa	Pestaña <i>Historial</i> dentro de /programs/{id}

4.4 Matriz de Indicadores (IND)

1. Objetivo

Definir indicadores estandarizados, establecer metas y capturar series temporales de alta calidad para monitoreo.

2. Funcionalidades

- CRUD de indicadores con metadatos (fórmula, unidad, fuente).
- Importador Excel para carga inicial.
- Vinculación N:M indicador ↔ programa con línea base y meta.
- Ingesta masiva de series (JSON/CSV) y validación de calidad de datos.
- Motor de alertas (meta incumplida, valores outlier) con notificación.

3. Componentes internos

Componente	Responsabilidad
Indicator Catalog	Alta/edición + búsqueda full-text
Series Ingest	API masiva + validación por reglas
Quality Engine	Reglas: rango, monotonidad, missing gaps
Alert Dispatcher	Genera y envía alertas (email / in-app)

4. Dependencias

Programas (FK), CATS, Data Warehouse (padrones).

5. Casos de Uso prioritarios

CU	Nombre	Actor	Flujo feliz	Excepciones
I-01	Vincular indicador	Operador	POST vincular	Duplicado
I-02	Cargar serie	Operador	POST lote → OK	Outliers
I-03	Alerta meta	Sistema	Job nightly → Email	—

6. Flujo I-02

Operador sube JSON lote
Quality Engine → Rango/consistencia
| ↳ Registra fallas en tabla errores
Inserta valores válidos en partición por año
Publica evento `serie.cargada` a Reportes

7. Interfaz de usuario (7 pantallas)

#	Pantalla	Propósito	Navegación
1	Catálogo Indicadores	DataGrid + búsqueda full-text.	/indicators.
2	Formulario Indicador	Alta/edición → metadatos.	Modal o /indicators/new.
3	Importador Excel	Wizard 3 pasos (upload, mapping, confirm).	/indicators/import.
4	Vincular a Programa	Dialog con multiselect + metas.	Botón “Vincular” en Programa.
5	Series DataGrid	Valores por fecha, inline edit.	/indicators/{id}/series.
6	Carga Masiva Series	Wizard CSV/JSON + resumen.	/series/import.

7	Panel Alertas	Tarjetas por severidad, filtro por meta incumplida.	/alerts.
---	---------------	---	----------

4.5 Evaluación & Agenda de Mejoras (EVAL)

1. Objetivo

Planificar estudios de evaluación, registrar hallazgos y coordinar acciones de mejora hasta su cierre.

2. Funcionalidades

- Calendario y ficha de evaluación (tipo, alcance, metodología).
- Registro de hallazgos y recomendaciones vinculadas a indicadores.
- Kanban **Pendiente** → **En curso** → **Cerrada** con responsable y fechas.
- KPIs de implementación (porcentaje cierre, tiempo medio).
- Generador de informe PDF con gráfica de avance.

3. Componentes internos

Componente	Descripción
Evaluation Planner	CRUD evaluaciones, cronograma Gantt
Findings Registry	ABM hallazgos, adjuntos descriptivos
Improvement Kanban	Cambio de estado drag-&-drop, SLA
Reporting Engine	Consolida KPIs y exporta informe

4. Dependencias

Programas, Indicadores, AUTH.

5. Casos de Uso prioritarios

CU	Nombre	Actor	Flujo feliz	Excepciones
E-01	Crear evaluación	Evaluador	POST → Planificada	Duplicado
E-02	Añadir recomendación	Evaluador	POST rec	Eval cerrada
E-03	Cerrar acción	Responsable	PATCH Cerrada	Fecha > hoy

6. Flujo Kanban (E-03)

Tarjeta Pendiente → En curso → evidencia adjunta
 Responsable marca Cerrada (< fecha compromiso)
 Sistema recalcula KPI cierre y notifica coordinador

7. Interfaz de usuario (6 pantallas)

#	Pantalla	Propósito	Navegación
1	Calendario Evaluaciones	Vista calendar + tarjetas por estado.	/evaluations/calendar.

2	Ficha Evaluación	Tabs: Datos, Equipo, Resultados.	/evaluations/{id}.
3	Hallazgos & Recomendaciones	Lista tipo checklist, adjuntos.	Tab <i>Hallazgos</i> en ficha.
4	Kanban Acciones	Columnas Pendiente / En curso / Cerrada.	/actions/board.
5	Dashboard KPIs	% cierre, días promedio, heatmap.	/actions/kpi.
6	Informe PDF Preview	Render HTML → PDF antes de exportar.	/evaluations/{id}/report.

4.6 Reportes & Dashboards (REP)

1. Objetivo

Convertir datos del SIME en información visual y accionable mediante reportes y tableros interactivos.

2. Funcionalidades

- **Report Builder** con editor SQL y plantillas pre-definidas.
- **Dashboard Designer** drag-and-drop (gráficos, tarjetas KPI, tablas).
- Scheduler de reportes (cron, eventos) con exportaciones PDF/XLSX.
- Control de permisos por rol y **embed tokens** para BI externo.
- Historial de ejecuciones, tiempos y destinatarios.

3. Componentes internos

Componente	Responsabilidad
Dashboard Designer	Layout grid, guardado persistente
Report Scheduler	Job runner, generación y mailing
Widget Library	Gráficos, tablas, KPI, mapas
Embed Gateway	Tokeniza URLs para BI corporativo

4. Dependencias

Indicadores, Evaluación, AUTH, Object Storage (archivos).

5. Casos de Uso prioritarios

CU	Nombre	Actor	Flujo feliz	Excepciones
R-01	Crear dashboard	Designer	POST dashboard	SQL inválido
R-02	Programar reporte	Operador	POST programación	Timeout
R-03	Compartir dashboard	Coordinador	POST permiso	Permiso duplicado

6. Flujo R-02 (scheduler)

Usuario define cron (+ filtros, formato)

Scheduler registra job cron

Al ejecutar: consulta BD → genera PDF

Almacena en Object Storage, genera link firmado

Envía email a destinatarios, registra audit_log

7. Interfaz de usuario (8 pantallas)

#	Pantalla	Propósito	Navegación
1	Lista Dashboards	Tarjetas + búsqueda por tag.	/dashboards.
2	Dashboard Designer	Drag-and-drop grid + Widget Library.	/dashboards/new.
3	Widget Library	Panel lateral (gráfico, KPI, tabla).	Dentro de Designer.
4	Reporte Programado – Wizard	Cron, filtros, formato, destinos.	/reports/schedule.
5	Historial Ejecuciones	Tabla estado/tiempo/archivo.	/reports/history.
6	Dashboard Viewer	Modo lectura + filtros dinámicos.	/dashboards/{id}.
7	Sharing/Permisos	Modal roles ↔ permisos (view/edit).	Botón “Compartir”.
8	Embed Config	Genera token embed + ejemplo iframe.	/dashboards/{id}/embed.

5 · Modelo de Datos & DDL Completo

Convenciones

- Nombres en minúsculas, snake_case.
- PK serial o char(2) según caso.
- Todas las tablas llevan created_at TIMESTAMPTZ DEFAULT NOW() y, cuando procede, updated_at TIMESTAMPTZ y activo BOOLEAN DEFAULT TRUE.
- FK con ON UPDATE CASCADE y ON DELETE RESTRICT salvo que se indique otro comportamiento.

5.1 Módulo Identidad & Roles

```
-- ■ Usuarios -----
CREATE TABLE usuario (
  usuario_id SERIAL PRIMARY KEY,
  email      VARCHAR(120) UNIQUE NOT NULL,
  password_hash VARCHAR(255) NOT NULL,
  first_name VARCHAR(60),
  last_name  VARCHAR(60),
  mfa_enabled BOOLEAN DEFAULT FALSE,
  mfa_secret VARCHAR(64),
  status     VARCHAR(20) DEFAULT 'ACTIVE', -- ACTIVE | SUSPENDED | DELETED
  last_login_at TIMESTAMPTZ,
  created_at  TIMESTAMPTZ DEFAULT NOW(),
  updated_at  TIMESTAMPTZ
);

-- ■ Roles y Permisos -----
CREATE TABLE rol (
  rol_id SERIAL PRIMARY KEY,
  nombre VARCHAR(50) UNIQUE NOT NULL,
  descripcion TEXT,
  created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE TABLE permiso (
  permiso_id SERIAL PRIMARY KEY,
  recurso VARCHAR(50) NOT NULL,
  accion VARCHAR(20) NOT NULL, -- CREATE | READ | UPDATE | DELETE | EXPORT | ...
  descripcion TEXT,
  UNIQUE (recurso, accion)
);

CREATE TABLE rol_permiso (
  rol_id INT REFERENCES rol ON DELETE CASCADE,
  permiso_id INT REFERENCES permiso ON DELETE CASCADE,
  PRIMARY KEY (rol_id, permiso_id)
);
```

```
CREATE TABLE usuario_rol (
  usuario_id INT REFERENCES usuario ON DELETE CASCADE,
  rol_id INT REFERENCES rol ON DELETE RESTRICT,
  PRIMARY KEY (usuario_id, rol_id)
);
```

```
-- ■ Token blacklist (revocados) -----
CREATE TABLE token_blacklist (
  jti VARCHAR(36) PRIMARY KEY,
  usuario_id INT REFERENCES usuario ON DELETE CASCADE,
  exp TIMESTAMPTZ NOT NULL,
  revoked_at TIMESTAMPTZ DEFAULT NOW()
);
```

```
-- ■ Bitácora de auditoría -----
CREATE TABLE audit_log (
  audit_id BIGSERIAL PRIMARY KEY,
  usuario_id INT REFERENCES usuario ON DELETE SET NULL,
  evento VARCHAR(50),
  ip INET,
  user_agent TEXT,
  fecha TIMESTAMPTZ DEFAULT NOW(),
  metadata JSONB
) PARTITION BY RANGE (fecha);
-- Ejemplo partición mensual
CREATE TABLE audit_log_2025_06 PARTITION OF audit_log
  FOR VALUES FROM ('2025-06-01') TO ('2025-07-01');
```

5.2 Catálogos Maestros

```
CREATE TABLE provincia (
  prov_id CHAR(2) PRIMARY KEY,
  nombre VARCHAR(60) NOT NULL UNIQUE,
  region VARCHAR(30),
  activo BOOLEAN DEFAULT TRUE,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ
);
```

```
CREATE TABLE organismo (
  org_id SERIAL PRIMARY KEY,
  sigla VARCHAR(15) UNIQUE NOT NULL,
  nombre VARCHAR(120) NOT NULL,
  ambito VARCHAR(20) CHECK (ambito IN ('Nacional','Provincial','Municipal')),
  activo BOOLEAN DEFAULT TRUE,
  created_at TIMESTAMPTZ DEFAULT NOW()
);
```

```
CREATE TABLE tipo_programa (
  tipo_prog_id SERIAL PRIMARY KEY,
  codigo VARCHAR(20) UNIQUE NOT NULL,
  descripcion TEXT,
  activo BOOLEAN DEFAULT TRUE,
```

```

    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ
);

CREATE TABLE tipo_prestacion (
    tipo_pres_id SERIAL PRIMARY KEY,
    codigo VARCHAR(20) UNIQUE NOT NULL,
    descripcion TEXT,
    activo BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE TABLE frecuencia_medicion (
    freq_id SERIAL PRIMARY KEY,
    codigo VARCHAR(15) UNIQUE NOT NULL, -- mensual, trimestral...
    descripcion TEXT,
    meses SMALLINT NOT NULL,
    activo BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMPTZ DEFAULT NOW()
);

```

5.3 Programas y Prestaciones

Novedad: Se admite que usuarios con rol adecuado puedan crear **campos personalizados** (extra columns) para el programa sin modificar el esquema físico. Se implementa un modelo híbrido que combina columnas estándar y **metadatos JSONB** + tablas EAV para consultas estructuradas cuando sea necesario.

5.3.1 Esquema estándar

```

-- ■ Programa -----
CREATE TABLE programa (
    programa_id SERIAL PRIMARY KEY,
    nombre VARCHAR(120) NOT NULL,
    objetivo TEXT,
    tipo_prog_id INT REFERENCES tipo_programa ON UPDATE CASCADE,
    org_id INT REFERENCES organismo,
    prov_id CHAR(2) REFERENCES provincia,
    estado VARCHAR(20) DEFAULT 'BORRADOR', -- BORRADOR | ACTIVO | CERRADO
    fecha_alta DATE DEFAULT CURRENT_DATE,
    metadata JSONB DEFAULT '{}::jsonb', -- Campos libres definidos por usuario
    created_by INT REFERENCES usuario,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ
);

```

5.3.2 Diseño de campos personalizados (EAV opcional)

```

-- Catálogo de campos dinámicos -----
CREATE TABLE programa_field_def (
    field_id SERIAL PRIMARY KEY,
    nombre VARCHAR(60) UNIQUE NOT NULL, -- slug
    etiqueta VARCHAR(120) NOT NULL, -- Display label

```

```

tipo_dato VARCHAR(15) CHECK (tipo_dato IN ('TEXT','INT','NUMERIC','DATE','BOOL')),
creado_por INT REFERENCES usuario,
created_at TIMESTAMPTZ DEFAULT NOW()
);

```

-- Valores asignados -----

```

CREATE TABLE programa_field_val (
  programa_id INT REFERENCES programa ON DELETE CASCADE,
  field_id INT REFERENCES programa_field_def ON DELETE CASCADE,
  valor_text TEXT,
  valor_int INT,
  valor_num NUMERIC,
  valor_date DATE,
  valor_bool BOOLEAN,
  updated_at TIMESTAMPTZ DEFAULT NOW(),
  PRIMARY KEY (programa_id, field_id)
);

```

Regla de integridad: según tipo_dato, solo se permite un campo valor no nulo; se valida mediante CHECK o lógica de aplicación.

5.3.3 Flujo de creación de campo

1. Usuario con rol **Admin** o **Designer** navega a *Gestión de Campos Personalizados*.
2. Define nombre (slug), etiqueta, tipo_dato.
3. Sistema inserta en programa_field_def y actualiza cache de definición.
4. Las pantallas *Wizard Programa* y *Detalle* renderizan automáticamente input según tipo.

5.3.4 Ventajas del modelo híbrido

- **Flexibilidad:** metadata JSONB para consultas ad-hoc; EAV para campos reportables.
- **Sin migraciones:** no requiere ALTER TABLE al agregar campo.
- **Indexes selectivos:** se pueden crear índices GIN sobre metadata, o BTREE sobre columnas de programa_field_val para análisis específicos.

-- ■ Programa -----

```

CREATE TABLE programa (
  programa_id SERIAL PRIMARY KEY,
  nombre VARCHAR(120) NOT NULL,
  objetivo TEXT,
  tipo_prog_id INT REFERENCES tipo_programa ON UPDATE CASCADE,
  org_id INT REFERENCES organismo,
  prov_id CHAR(2) REFERENCES provincia,
  estado VARCHAR(20) DEFAULT 'BORRADOR', -- BORRADOR | ACTIVO | CERRADO
  fecha_alta DATE DEFAULT CURRENT_DATE,
  created_by INT REFERENCES usuario,
  created_at TIMESTAMPTZ DEFAULT NOW(),
);

```

```

    updated_at TIMESTAMPTZ
);

-- Historial de cambios
CREATE TABLE programa_hist (
    hist_id BIGSERIAL PRIMARY KEY,
    programa_id INT REFERENCES programa ON DELETE CASCADE,
    version INT NOT NULL,
    diff JSONB, -- {"field":"old→new"}
    changed_by INT REFERENCES usuario,
    changed_at TIMESTAMPTZ DEFAULT NOW()
);

-- ■ Prestaciones -----
CREATE TABLE prestacion (
    prestacion_id SERIAL PRIMARY KEY,
    programa_id INT REFERENCES programa ON DELETE CASCADE,
    tipo_pres_id INT REFERENCES tipo_prestacion,
    descripcion TEXT NOT NULL,
    poblacion_obj NUMERIC,
    unidad_medida VARCHAR(50),
    activo BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ
);

```

5.4 Indicadores y Series

```

-- ■ Indicador -----
CREATE TABLE indicador (
    indicador_id SERIAL PRIMARY KEY,
    codigo VARCHAR(30) UNIQUE NOT NULL,
    nombre TEXT NOT NULL,
    descripcion TEXT,
    formula TEXT,
    unidad VARCHAR(50),
    fuente VARCHAR(120),
    created_at TIMESTAMPTZ DEFAULT NOW()
);

-- Vinculación Indicador ↔ Programa
CREATE TABLE indicador_programa (
    programa_id INT REFERENCES programa ON DELETE CASCADE,
    indicador_id INT REFERENCES indicador,
    linea_base NUMERIC,
    meta NUMERIC,
    freq_id INT REFERENCES frecuencia_medicion,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    PRIMARY KEY (programa_id, indicador_id)
);

-- Series temporales
CREATE TABLE serie_indicador (

```



```

serie_id    BIGSERIAL PRIMARY KEY,
programa_id INT,
indicador_id INT,
fecha      DATE NOT NULL,
valor      NUMERIC,
quality_flag VARCHAR(20) DEFAULT 'OK', -- OK | OUTLIER | MISSING
cargado_por INT REFERENCES usuario,
inserted_at TIMESTAMPTZ DEFAULT NOW(),
CONSTRAINT fk_ip FOREIGN KEY (programa_id, indicador_id)
REFERENCES indicador_programa (programa_id, indicador_id) ON DELETE CASCADE
) PARTITION BY RANGE (fecha);
-- Ejemplo sub-partición
CREATE TABLE serie_indicador_2025 PARTITION OF serie_indicador
FOR VALUES FROM ('2025-01-01') TO ('2026-01-01');

-- Registro de errores en carga
CREATE TABLE serie_error (
error_id    BIGSERIAL PRIMARY KEY,
archivo_ref VARCHAR(120),
linea      INT,
motivo     TEXT,
payload    JSONB,
usuario_id INT REFERENCES usuario,
created_at TIMESTAMPTZ DEFAULT NOW()
);

```

5.5 Evaluaciones y Mejoras

```

-- ■ Evaluaciones -----
CREATE TABLE evaluacion (
evaluacion_id SERIAL PRIMARY KEY,
programa_id INT REFERENCES programa ON DELETE CASCADE,
tipo        VARCHAR(50),
alcance     TEXT,
metodologia TEXT,
fecha_inicio DATE,
fecha_fin   DATE,
estado      VARCHAR(20) DEFAULT 'PLANIFICADA', -- PLANIFICADA | EN_CURSO | CERRADA
created_by  INT REFERENCES usuario,
created_at  TIMESTAMPTZ DEFAULT NOW()
);

CREATE TABLE recomendacion (
recomendacion_id SERIAL PRIMARY KEY,
evaluacion_id INT REFERENCES evaluacion ON DELETE CASCADE,
descripcion   TEXT NOT NULL,
prioridad     VARCHAR(10) CHECK (prioridad IN ('ALTA', 'MEDIA', 'BAJA')),
created_by    INT REFERENCES usuario,
created_at    TIMESTAMPTZ DEFAULT NOW()
);

CREATE TABLE mejora_seguimiento (
mejora_id     SERIAL PRIMARY KEY,

```

```

recomendacion_id INT REFERENCES recomendacion ON DELETE CASCADE,
responsable_id INT REFERENCES usuario,
estado VARCHAR(20) DEFAULT 'PENDIENTE', -- PENDIENTE | EN_CURSO | CERRADA
fecha_compromiso DATE,
fecha_cierre DATE,
evidencia_url TEXT,
updated_at TIMESTAMPTZ
);

```

5.6 Reportes y Dashboards

```

-- ■ Definición de Reporte -----
CREATE TABLE report_def (
  reporte_id SERIAL PRIMARY KEY,
  nombre VARCHAR(120) NOT NULL,
  descripcion TEXT,
  tipo VARCHAR(20) CHECK (tipo IN ('SQL','VIEW','EXTERNAL')),
  fuente_query TEXT,
  formato_def JSONB, -- columnas, estilos
  creado_por INT REFERENCES usuario,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ
);

CREATE TABLE programacion_reporte (
  prog_id SERIAL PRIMARY KEY,
  reporte_id INT REFERENCES report_def ON DELETE CASCADE,
  cron_exp VARCHAR(60) NOT NULL,
  formato_salida VARCHAR(10) CHECK (formato_salida IN ('PDF','XLSX')),
  destinatarios TEXT[],
  activo BOOLEAN DEFAULT TRUE,
  created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE TABLE report_execution (
  exec_id BIGSERIAL PRIMARY KEY,
  prog_id INT REFERENCES programacion_reporte ON DELETE CASCADE,
  estado VARCHAR(20) CHECK (estado IN ('OK','ERROR','TIMEOUT')),
  inicio_at TIMESTAMPTZ,
  fin_at TIMESTAMPTZ,
  archivo_url TEXT,
  filas INT,
  mensaje TEXT
);

```

```

-- ■ Dashboards & Widgets -----
CREATE TABLE dashboard (
  dashboard_id SERIAL PRIMARY KEY,
  nombre VARCHAR(120) NOT NULL,
  descripcion TEXT,
  owner_id INT REFERENCES usuario,
  is_public BOOLEAN DEFAULT FALSE,
  created_at TIMESTAMPTZ DEFAULT NOW(),

```

```

    updated_at TIMESTAMPTZ
);

CREATE TABLE widget (
    widget_id SERIAL PRIMARY KEY,
    dashboard_id INT REFERENCES dashboard ON DELETE CASCADE,
    tipo VARCHAR(20) CHECK (tipo IN ('CHART','KPI','TABLE','MAP')),
    config JSONB,
    posicion INT,
    created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE TABLE dashboard_permiso (
    dashboard_id INT REFERENCES dashboard ON DELETE CASCADE,
    rol_id INT REFERENCES rol ON DELETE CASCADE,
    permiso VARCHAR(10) CHECK (permiso IN ('VIEW','EDIT')),
    PRIMARY KEY (dashboard_id, rol_id, permiso)
);

```

5.7 Índices, Vistas y Otros Artefactos

```

-- Índices recomendados -----
CREATE INDEX idx_indicador_codigo    ON indicador (codigo);
CREATE INDEX idx_serie_programa_fecha ON serie_indicador (programa_id, fecha DESC);
CREATE INDEX idx_mejora_estado       ON mejora_seguimiento (estado);
CREATE INDEX idx_report_exec_estado  ON report_execution (estado);

-- Vista materializada: KPI avance mejoras -----
CREATE MATERIALIZED VIEW mv_kpi_mejoras AS
SELECT
    p.programa_id,
    COUNT(*) FILTER (WHERE ms.estado = 'CERRADA') AS acciones_cerradas,
    COUNT(*) AS acciones_totales,
    ROUND(COUNT(*) FILTER (WHERE ms.estado = 'CERRADA')::NUMERIC * 100 / NULLIF(COUNT(*),0),2) AS
    avance_pct
FROM programa p
JOIN prestacion pr USING (programa_id)
JOIN indicador_programa ip USING (programa_id)
JOIN evaluacion e USING (programa_id)
JOIN recomendacion r USING (evaluacion_id)
JOIN mejora_seguimiento ms USING (recomendacion_id)
GROUP BY p.programa_id;

```

Nota: Este DDL cubre **todos los contextos** descritos en la arquitectura:

- Identidad,
- Catálogos,
- Programas,
- Indicadores,
- Evaluaciones
- Reportes/Dashboards,

- incluyendo tablas auxiliares para auditoría y ejecución de reportes.

6 · API REST v1

La API SIME sigue el estándar **OpenAPI 3.1** y separa responsabilidades mediante **tags** que reflejan nuestros módulos. A continuación se documentan los **recursos principales**, su **propósito funcional**, parámetros, códigos de estado y ejemplos.

Convenciones generales

- Formato base URL: `https://api.sime.gob/api/v1/...`
- Todas las rutas devuelven `application/json` salvo descargas.
- Autorización mediante esquema de seguridad `bearerAuth` (JWT).
- Errores siguen el formato RFC 7807 (*Problem Details*).

6.1 Esquema de Seguridad

Nombre	Tipo	Encabezado	Descripción
bearerAuth	HTTP bearer	Authorization: Bearer <token>	Token JWT firmado; expiración 30 min.

6.2 Tags disponibles

Tag	Módulo	Descripción
Auth	Identidad & Roles	Endpoints de autenticación y gestión de usuarios.
Catálogos	Catálogos Maestros	Provincias, organismos, etc.
Programas	Programas	CRUD y flujo de programas.
Indicadores	Matriz de Indicadores	Indicadores, series y alertas.
Evaluaciones	Evaluación & Mejoras	Estudios, hallazgos, acciones.
Reportes	Reportes & Dashboards	Gestão de dashboards y scheduler.

6.3 Endpoints clave

A continuación se listan los **recursos núcleo** por tag con descripciones profesionales y fáciles de entender.

6.3.1 Auth

Método	Ruta	Descripción	Respuestas	Ejemplo éxito
POST	/auth/login	Inicia sesión y devuelve par de tokens.	200 JWT · 401 Credenciales inválidas	<code>curl -d '{"email":"user@...", "password":"..."}'</code>

POST	/auth/refresh	Obtiene nuevo access_token a partir de refresh.	200 nuevo par · 401 refresh expirado	—
GET	/auth/profile	Devuelve datos del usuario autenticado.	200 Perfil JSON	—
POST	/auth/logout	Revoca tokens actuales.	204 sin contenido	—

6.3.2 Catálogos

Método	Ruta	Descripción	Parámetros	Respuestas
GET	/provincias	Lista provincias, opcional activo.	activo bool (query)	200 array paginado
POST	/provincias	Crea provincia.	body Provincia	201 creada · 409 duplicado
PATCH	/provincias/{prov_id}	Actualiza parcial.	path prov_id, body campos	200 ok · 404 no existe
DELETE	/provincias/{prov_id}	Soft delete (activo=false).	path	204

Los mismos verbos se aplican a /organismos, /tipos-programa, etc.

6.3.3 Programas

Método	Ruta	Descripción	Respuestas
GET	/programas	Filtrado por estado, tipo, organismo.	200 Paginado
POST	/programas	Crea programa (Paso 1 Wizard).	201 Borrador
PUT	/programas/{id}	Actualiza campos (incluye pasos 2–3).	200
PATCH	/programas/{id}/estado	Cambia estado (Activo, Cerrado).	200 ok · 422 violación reglas
GET	/programas/{id}/historial	Lista versiones (programa_hist).	200 array

6.3.4 Indicadores

Método	Ruta	Descripción
GET	/indicadores	Búsqueda de indicadores mediante texto libre (nombre, código, fórmula, etc.).
POST	/indicadores	Alta de un nuevo indicador en el catálogo maestro.
POST	/indicadores/{id}/vincular	Relaciona el indicador con un programa y le asigna línea base / meta.
POST	/series/import	Importa un lote de series de valores (JSON o CSV) para uno o varios indicadores.
GET	/series?programa_id=...&indicador_id=...	Consulta una serie filtrada por programa e indicador.

GET	/alertas	Devuelve alertas activas (metas incumplidas, valores atípicos).
------------	----------	---

6.3.5 Evaluaciones

Método	Ruta	Descripción
GET	/evaluaciones	Lista las evaluaciones existentes con filtros por programa, estado o fecha.
POST	/evaluaciones	Registra una nueva evaluación (planificada o en curso).
POST	/evaluaciones/{id}/recomendaciones	Agrega una recomendación vinculada a la evaluación.
PATCH	/mejoras/{id}	Actualiza el estado de una acción de mejora (Pendiente → En curso → Cerrada).

6.3.6 Reportes & Dashboards

Método	Ruta	Descripción
GET	/dashboards	Devuelve los dashboards accesibles para el usuario autenticado.
POST	/dashboards	Crea un nuevo dashboard y lo devuelve con su ID.
GET	/dashboards/{id}	Recupera un dashboard específico junto con sus widgets.
POST	/reports/schedule	Programa la ejecución periódica de un reporte (cron, formato, destinatarios).
GET	/reports/history	Lista el historial de ejecuciones de reportes (estado, duración, enlace al archivo).

6.4 Componentes (schemas)

A continuación se listan **todos** los modelos de dominio expuestos en la especificación OpenAPI v1 con sus campos más relevantes. (Los atributos de auditoría – created_at, updated_at, etc. – se omiten para claridad) adjunto archivo: openapi.yaml.

Componente	Campos esenciales (tipo)
Credenciales	email <i>string</i> · password <i>string</i>
TokenPair	access_token <i>string</i> · refresh_token <i>string</i> · expires_in <i>int</i>
Problem	type <i>uri</i> · title <i>string</i> · status <i>int</i> · detail <i>string</i> · instance <i>uri</i>
Provincia	prov_id <i>char(2)</i> · nombre <i>string</i> · region <i>string</i> · activo <i>bool</i>
PaginatedProvincia	items <i>array</i> Provincia · total <i>int</i> · page <i>int</i> · size <i>int</i>
Organismo	org_id <i>int</i> · sigla <i>string</i> · nombre <i>string</i> · ambito <i>string</i> · activo <i>bool</i>
TipoPrograma	tipo_prog_id <i>int</i> · codigo <i>string</i> · descripcion <i>string</i> · activo <i>bool</i>
Programa	programa_id <i>int</i> · nombre <i>string</i> · objetivo <i>string</i> · estado <i>string</i> · tipo_prog_id <i>int</i> · org_id <i>int</i> · prov_id <i>char(2)</i> · metadata <i>object</i>
Prestacion	prestacion_id <i>int</i> · programa_id <i>int</i> · tipo_pres_id <i>int</i> · descripcion <i>string</i> · poblacion_obj <i>number</i>
Indicador	indicador_id <i>int</i> · codigo <i>string</i> · nombre <i>string</i> · formula <i>string</i> · unidad <i>string</i>

Serie	serie_id <i>int</i> · programa_id <i>int</i> · indicador_id <i>int</i> · fecha <i>date</i> · valor <i>number</i> · quality_flag <i>string</i>
Alerta	alerta_id <i>int</i> · programa_id <i>int</i> · indicador_id <i>int</i> · tipo <i>string</i> · descripcion <i>string</i>
Evaluacion	evaluacion_id <i>int</i> · programa_id <i>int</i> · tipo <i>string</i> · estado <i>string</i> · fecha_inicio <i>date</i> · fecha_fin <i>date</i>
Recomendacion	recomendacion_id <i>int</i> · evaluacion_id <i>int</i> · descripcion <i>string</i> · prioridad <i>string</i>
Mejora	mejora_id <i>int</i> · recomendacion_id <i>int</i> · estado <i>string</i> · responsable_id <i>int</i>
Dashboard	dashboard_id <i>int</i> · nombre <i>string</i> · descripcion <i>string</i> · is_public <i>bool</i>
Widget	widget_id <i>int</i> · dashboard_id <i>int</i> · tipo <i>string</i> · config <i>object</i>
ReportDef	reporte_id <i>int</i> · nombre <i>string</i> · tipo <i>string</i> · fuente_query <i>string</i>
ReportSchedule	prog_id <i>int</i> · reporte_id <i>int</i> · cron_exp <i>string</i> · formato_salida <i>string</i>
ReportExecution	exec_id <i>int</i> · prog_id <i>int</i> · estado <i>string</i> · archivo_url <i>string</i> · inicio_at <i>datetime</i>

Los esquemas expuestos cubren todos los módulos:

- **Auth,**
- **Catálogos,**
- **Programas,**
- **Indicadores,**
- **Evaluaciones**
- **Reportes/Dashboards.**

Campos adicionales de auditoría y paginado siguen las convenciones descritas en la sección de DDL y API

6.5 Respuestas genéricas

Código	Significado	Objeto
200	Éxito; devuelve recurso o lista	Modelo correspondiente
201	Creación exitosa	Objeto creado
204	Operación exitosa sin cuerpo	—
400	Petición malformada	Problem
401	Sin autenticación	Problem
403	Sin permisos	Problem
404	Recurso no encontrado	Problem
409	Conflicto (duplicado, referencial)	Problem
422	Violación de reglas de negocio	Problem

6.6 Versionado y compatibilidad

- Prefijo /v1 indica versión mayor; evoluciones menores se indican con encabezado API-Version y semver.
- Rutas siguen patrón REST; cambios breaking → nueva versión (/v2).

Nota: El **archivo completo openapi.yaml (1573 líneas)** — Incluye:

- Seguridad, parámetros reutilizables y esquema “Problem RFC 7807”
- CRUD para provincias, organismos, tipos-programa, programas, indicadores, evaluaciones, recomendaciones, mejoras de dashboards, widgets, reportes y programaciones
- Endpoints adicionales: importación de series y alertas para compartir con dashboards
- Esquemas Evaluación, Recomendación, Mejora, Dashboard, Widget, ReportDef, ReportSchedule

7 · Diagramas de Flujo & Secuencia

Contiene **ocho diagramas de secuencia** que cubren:

1. Login MFA (Auth)
2. Alta de Programa (Wizard)
3. Creación de Campo Personalizado
4. Carga Masiva de Series
5. Generación de Alerta de Meta
6. Ciclo Evaluación → Recomendación → Cierre Acción
7. Creación y Compartición de Dashboard
8. Scheduler de Reporte Programado

Cada diagrama muestra actores FE, servicios, BD y sistemas externos — reflejando el 100 % de los flujos críticos definidos.

7.1 Login con MFA (Auth)

```
sequenceDiagram
    actor Usuario
    participant FE as Front-End
    participant Auth as AuthSvc
    participant DB as PostgreSQL
    Usuario->>FE: Completa email + password
    FE->>Auth: POST /auth/login
    Auth->>DB: SELECT usuario, hash, mfa_enabled
    alt Credenciales válidas y MFA activo
        Auth-->>FE: HTTP 401 (MFA_REQUIRED)
        Usuario->>FE: Ingresa código TOTP
        FE->>Auth: POST /auth/verify-totp
        Auth->>DB: Validar TOTP
        Auth-->>FE: 200 TokenPair
    else Credenciales válidas sin MFA
        Auth-->>FE: 200 TokenPair
    else Error
        Auth-->>FE: 401
    end
    Auth->>DB: INSERT audit_log(Login)
```

7.2 Alta de Programa (Wizard 4 pasos)

```
sequenceDiagram
    participant C as Coordinador (FE)
    participant Prog as ProgSvc
    participant DB as PostgreSQL
    C->>Prog: POST /programas (Paso 1)
    Prog->>DB: INSERT programa (Borrador)
    C->>Prog: PUT /prestaciones (Paso 2)
    Prog->>DB: INSERT prestacion[]
    C->>Prog: PUT /indicadores (Paso 3)
```

```

Prog->>DB: INSERT indicador_programa[]
C->>Prog: PATCH /programas/{id} estado=Activo (Paso 4)
Prog->>DB: UPDATE programa
Prog-->>C: 200 OK + event programa.aprobado

```

7.3 Creación de Campo Personalizado

```

sequenceDiagram
    participant Admin as Admin (FE)
    participant Prog as ProgSvc
    participant DB as PostgreSQL
    Admin->>Prog: POST /programs/custom-fields { slug, label, tipo }
    Prog->>DB: INSERT programa_field_def
    Prog-->>Admin: 201 Created + event custom_field.created
    FE->>FE: Recarga cache definiciones → Wizard muestra nuevo input

```

7.4 Carga Masiva de Series

```

sequenceDiagram
    participant Oper as Operador (FE)
    participant Ind as IndSvc
    participant DB as PostgreSQL
    Oper->>Ind: POST /series/import (JSON lote)
    loop Por cada registro
        Ind->>Ind: Validar reglas (rango / dtype)
        alt OK
            Ind->>DB: INSERT serie_indicador PARTITION
        else Error
            Ind->>DB: INSERT serie_error
        end
    end
    Ind-->>Oper: 202 Accepted + resumen (insertados / errores)
    Ind->>Ind: Publicar evento serie.cargada

```

7.5 Generación de Alerta de Meta Incumplida

```

sequenceDiagram
    participant Cron as Job nocturno
    participant Ind as IndSvc
    participant DB as PostgreSQL
    Cron->>DB: SELECT series vs meta
    alt Meta < valor_actual
        Ind->>DB: INSERT alerta
        Ind->>Notif: Send email / in-app
    end

```

7.6 Proceso de Evaluación y Cierre de Acción

```

sequenceDiagram
    participant Eval as Evaluador (FE)
    participant E as EvalSvc
    participant DB as PostgreSQL
    Eval->>E: POST /evaluaciones
    E->>DB: INSERT evaluacion (PLANIFICADA)

```

```
Eval->>E: POST /evaluaciones/{id}/recomendaciones
E->>DB: INSERT recomendacion
Responsable->>E: PATCH /mejoras/{id} estado="CERRADA" evidencias
E->>DB: UPDATE mejora_seguimiento
E->>DB: UPDATE KPI materializado
```

7.7 Creación y Compartición de Dashboard

```
sequenceDiagram
    participant Des as Designer (FE)
    participant Rep as RepSvc
    participant DB as PostgreSQL
    Des->>Rep: POST /dashboards { layout }
    Rep->>DB: INSERT dashboard + widgets
    Des->>Rep: POST /dashboards/{id}/share { rol_id, permiso }
    Rep->>DB: INSERT dashboard_permiso
    Rep-->>Des: 200 OK (link compartido)
```

7.8 Scheduler de Reporte Programado

```
sequenceDiagram
    participant Oper as Operador (FE)
    participant Rep as RepSvc
    participant DB as PostgreSQL
    participant OBJ as MinIO
    Oper->>Rep: POST /reports/schedule { cron, formato }
    Rep->>DB: INSERT programacion_reporte
    Cron-->>Rep: Trigger job según cron
    Rep->>DB: SELECT fuente_query
    DB-->>Rep: ResultSet
    Rep->>OBJ: PUT reporte.pdf/xlsx
    Rep->>SMTP: Email link firmado
    Rep->>DB: INSERT report_execution estado=OK
```

Cada diagrama refleja el **flujo end-to-end** de los procesos descritos en el modelo funcional, incluyendo participantes front-end, servicios lógicos, base de datos y sistemas externos (almacenamiento de objetos, SMTP). Integran todos los módulos:

- Auth
- Catálogos
- Programas
- Indicadores
- Evaluación
- Reportes

8 Seguridad, Auditoría & Cumplimiento

(Cobertura 360° — alineado a buenas prácticas DevSecOps y regulaciones vigentes)

8.1 Autenticación y Gestión de Identidades

Control	Implementación	Recomendaciones adicionales
JWT	RS256, expiración 30 min; refresh token 7 días	Emitir <i>jti</i> único y almacenarlo en <i>token_blacklist</i> para revocaciones.
MFA (TOTP)	Obligatoria para roles Admin y SuperAdmin ; opt-in para el resto	Habilitar notificaciones push (WebAuthn) en próxima versión.
Política de Contraseña	Longitud ≥ 12 , complejidad OWASP, historial 5 últimas	Forzar rotación cada 180 días.
SSO (futuro)	Keycloak / SAML2 compatible	Mapear grupos LDAP \leftrightarrow roles RBAC.

8.2 Autorización y Control de Acceso

- **RBAC** gestionado en `rol_permiso`; decisiones cacheadas en Redis (TTL = 60 s).
- **Principio de menor privilegio**: roles predefinidos + posibilidad de *custom roles* auditados.
- **Acceso de servicio a BD** mediante cuentas dedicadas con *least-privileges*.

8.3 Cifrado y Protección de Datos

Ámbito	Tecnología	Detalle
En tránsito	TLS 1.3 (LetsEncrypt)	HSTS, Perfect Forward Secrecy, deshabilitar TLS 1.0/1.1
En reposo	Discos LUKS en nodos productivos	Claves gestionadas vía LUKS2 + passphrase HSM
Columnas sensibles	pgcrypto (AES-GCM 256)	E-mail, mfa_secret, evidencia URLs
Object Storage	SSE-S3 (AES-256)	Rotación anual de keys

8.4 Monitorización, Logging y Auditoría

- **Tabla audit_log** con particionado mensual; campos *evento*, *ip*, *user_agent*, *metadata JSONB*.
- Forwarding en tiempo real a **Graylog** / **Elastic SIEM** con retención ≥ 365 días.
- Dashboards de seguridad (logins fallidos, cambios de rol, uso de tokens).
- Alertas automáticas (correo/Slack) ante patrones sospechosos (≥ 5 fallos de login/hora por IP).

8.5 Respaldo y Recuperación ante Desastre

Capa	Estrategia	Frecuencia	Retención
Base de datos	Continuous WAL shipping	~5 min	35 días en S3 (Object Lock Legal Hold)
Snapshot full	pg_dump --format=directory	Diaria 02:00	7 días
Object Storage	Versión habilitada	Inmediata	30 días
Config/K8s manifests	GitOps (ArgoCD) + backup repos	Cada commit	90 días

Pruebas: restauración quarterly a entorno *staging-DR*; RPO \leq 5 min, RTO \leq 2 h.

8.6 DevSecOps y Gestión de Vulnerabilidades

Fase	Herramienta	Cobertura
CI	SAST (semgrep), Dependency-check (OWASP)	PR gate \rightarrow bloquea CVE \geq High
CD	Trivy (container scan)	Escaneo imagen antes de push a registry
Runtime	Falco / OPA Gatekeeper	Detección de anomalías / policy enforcement
Pentest	Externo anual + retest	OWASP Top-10 + API Security Top-10

8.7 Cumplimiento Legal & Privacidad

- **PDPA Argentina (Ley 25.326):** registro de base de datos personales ante AAIP, políticas de derechos ARCO.
- **GDPR:** Art. 30 Registro de Actividades; Art. 32 Medidas técnicas y organizativas; soporte para **DSAR** (Data Subject Access Request) export JSON.
- **Logs de tracking** excluyen datos sensibles (minimización).
- **Retención:** datos personales eliminados/anonimizados a los 5 años post-cierre de programa.

Conclusión: con estos controles SIME cumple las normativas locales e internacionales, aplica prácticas DevSecOps y mantiene un nivel de riesgo residual aceptable, sujeto a revisión continua por la Oficina de Seguridad de la Información.



Final

Gracias por su confianza

Junio de 2025 | Renzo Antonioli

Analista funcional técnico & Dev

✉ rantonioli@zeron.com.ar

☎ (+549)-11-3566-5266