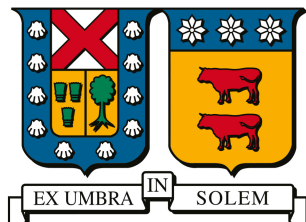


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO – CHILE



“REDISEÑO FUNCIONAL DE SISTEMA DE GESTIÓN PARA
EMERGENCIAS E IMPLEMENTACIÓN DE PMV PARA
VERSIÓN MULTI-MUNICIPAL: CASO APPCOPIO”

MARTINA TEJO ALARCÓN

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERA CIVIL EN INFORMÁTICA

Profesor Guía: Wladimir Ormazábal
Profesor Correferente: Pedro Godoy

Octubre 2026



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título ☐ Tesis de Postgrado

Título del trabajo: Rediseño funcional de sistema de gestión para emergencias e implementación de PMV para versión multi-municipal: caso AppCopio

Nombre del candidato(a): Martina Paz Tejo Alarcón

Carrera / Grado: Ingeniería Civil Informática

Campus: Casa Central Valparaíso **Departamento:** Departamento de Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Wladimir Ormázabal Orellana, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☒ El trabajo **NO contiene** información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un período de confidencialidad (**embargo**) por (**marcar una opción**):

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 19/10/2026

Firma: 

Estudiante o Candidato(a):

Fecha: 19/10/2026

Firma: 

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

Dedicatoria

A la Martina del pasado, que durante mucho tiempo
creyó que no lo lograría, pero aun así siguió adelante.

Y a todos los que alguna vez sintieron lo mismo.

Agradecimientos

A mi mamá, por motivarme y creer en mí incluso cuando yo no creía que lo lograría, y por darme el privilegio de estudiar desde casa. Y a mi hermano, que junto a ella dejó su vida atrás para que yo tuviera la oportunidad de ir a la universidad.

A mi Nonna, que le dio la oportunidad de estudiar a toda su descendencia y que siempre me instó a hacerlo, a pesar de que ella no pudo. Aunque ya no está, sigue presente, porque la gente buena no se entierra, se siembra.

A Vito y Eri, por todas esas horas de estudio y noches de desvelo en que, al girarme, los encontraba acompañándome en silencio. No solo me daban una alegría con su compañía, sino que me enseñaron que también era importante parar y descansar.

A mi primera amiga de la universidad, que si bien solo fuimos cercanas por un año, fue una rama de olivo durante el inicio de esta etapa. Y a Paula, que llegó después, por convertirse en mi dupla para cada trabajo, y por su amistad tanto dentro como fuera de la universidad. No se lo he dicho, pero fue un salvavidas incluso cuando no se daba cuenta.

A Javiera, por su amistad desde mucho antes de entrar a la universidad y por seguir siendo parte de mi vida, aunque nuestras áreas de estudio y caminos muchas veces no se topan.

A Sofía y a Monse, que se emocionaron por mí más de lo que yo hice cuando entré a la universidad, y que aunque no nos veamos seguido, son un refugio cada vez que nos vemos y un apoyo que siempre siento cerca.

Al Tryhard Inf, por el tiempo, las risas y las horas de estudio compartidas, que hicieron mis días y mi paso por la universidad mucho más livianos; por insistir en que me quedara a descansar y compartir más tiempo con ellos, incluso cuando a veces yo aceptaba a regañadientes.

Al profesor Wladimir, por su paciencia durante el desarrollo de esta memoria, y por la confianza que me dio para ser su ayudante durante varios años. Enseñar fue una de las experiencias más desafiantes que tuve en la universidad, y me ayudó a crecer un montón.

Una vez escuché que somos un mosaico de todas las personas que hemos conocido, y estoy convencida de que es verdad. No sería la persona que soy hoy si no fuera por todo lo que he aprendido de cada uno de ustedes. Si llegué hasta aquí, incluso en los días en que dudé de mí, fue porque nunca lo hice sola.

Resumen

Resumen — Las emergencias de gran escala en Chile suelen afectar a varias comunas a la vez, pero las municipalidades, responsables legales de la respuesta local, coordinan sus centros de acopio y albergues con herramientas aisladas entre sí. AppCopio, una plataforma web con capacidad de operación sin conexión desarrollada junto a la Municipalidad de Valparaíso, compartía esa limitación: fue construida para una sola comuna. Este trabajo diseña su evolución hacia una arquitectura multi-municipal (*multi-tenant*) e implementa un Producto Mínimo Viable sobre el código existente. A partir del diagnóstico del sistema se definieron doce requisitos funcionales y ocho no funcionales. Se adoptó un esquema de base de datos compartido con *Row-Level Security* de PostgreSQL, de modo que el aislamiento entre comunas lo aplique el motor y no dependa de cada consulta, y se diseñó un mecanismo de colaboración con consentimiento explícito, organizado en dos niveles de evento, que permite compartir necesidades priorizadas y ofertas de apoyo sin exponer datos de personas. La validación, mediante cinco escenarios críticos y 87 comprobaciones automatizadas reproducibles, acreditó el aislamiento de datos, la colaboración acotada y la continuidad de las funcionalidades heredadas; el rendimiento bajo carga y la validación con usuarios finales quedan como trabajo futuro. El resultado permite incorporar nuevas comunas sin desplegar infraestructura propia y coordinar la respuesta intercomunal sobre una misma plataforma.

Palabras Clave — Arquitectura multi-tenant; Row-Level Security; gestión de emergencias; colaboración intermunicipal; producto mínimo viable

Abstract

Abstract — Large-scale emergencies in Chile often affect several municipalities at once, yet municipalities, which are legally responsible for the local response, coordinate their relief supply centers and shelters with tools that are isolated from one another. AppCopio, a web platform with offline capabilities developed together with the Municipality of Valparaíso, shared this limitation: it was built for a single municipality. This work designs its evolution into a multi-municipal (*multi-tenant*) architecture and implements a Minimum Viable Product on top of the existing codebase. Based on a diagnosis of the system, twelve functional and eight non-functional requirements were defined. A shared database schema with PostgreSQL Row-Level Security was adopted, so that isolation between municipalities is enforced by the database engine instead of depending on each query, and a collaboration mechanism based on explicit consent and organized into two event levels was designed, allowing municipalities to share prioritized needs and support offers without exposing personal data. Validation through five critical scenarios and 87 reproducible automated checks confirmed data isolation, bounded collaboration and the continuity of existing features; performance under load and validation with end users remain as future work. The result allows new municipalities to join without deploying their own infrastructure and enables inter-municipal response coordination on a single platform.

Keywords — Multi-tenant architecture; Row-Level Security; emergency management; inter-municipal collaboration; minimum viable product

Glosario

Siglas y abreviaturas empleadas en el presente documento, en orden alfabético:

API: *Application Programming Interface*. Interfaz de programación de aplicaciones.

C4: Modelo de documentación de arquitectura de software organizado en cuatro niveles de abstracción (contexto, contenedor, componente y código).

COGRID: Comité para la Gestión del Riesgo de Desastres, instancia de coordinación operativa creada por la Ley N° 21.364 en cada nivel territorial.

DIDECO: Dirección de Desarrollo Comunitario, unidad municipal donde se radican los equipos de emergencia social.

FIBE: Ficha Básica de Emergencia, instrumento catastral del Ministerio de Desarrollo Social y Familia.

JWT: *JSON Web Token*. Formato de credencial de sesión firmada criptográficamente.

ONEMI: Oficina Nacional de Emergencia, organismo sustituido por SENAPRED en 2021.

PMV: Producto Mínimo Viable.

PWA: *Progressive Web Application*. Aplicación web con capacidades de operación sin conexión.

RBAC: *Role-Based Access Control*. Control de acceso basado en roles.

RF: Requisito Funcional.

RLS: *Row-Level Security*. Seguridad a nivel de fila; mecanismo de control de acceso por fila nativo de PostgreSQL.

RNF: Requisito No Funcional.

SaaS: *Software as a Service*. Software como servicio.

SENAPRED: Servicio Nacional de Prevención y Respuesta ante Desastres.

SINAPRED: Sistema Nacional de Prevención y Respuesta ante Desastres.

UTFSM: Universidad Técnica Federico Santa María.

Índice de contenidos

Resumen	iii
Abstract	iii
Glosario	iv
Índice de contenidos	v
Índice de figuras	vii
Índice de tablas	vii
Capítulo 1: Introducción	1
1.1 Objetivo general	2
1.2 Objetivos específicos	2
1.3 Metodología y estructura del documento	3
Capítulo 2: Marco teórico	4
2.1 Gestión de emergencias en Chile	4
2.1.1 Ley 21.364	4
2.1.2 Conceptos operativos relevantes	4
2.2 Arquitectura single-tenant	5
2.3 Arquitectura multi-tenant	5
2.3.1 Row-Level Security	6
2.4 Control de acceso y defensa en profundidad	6
2.5 Colaboración inter-organizacional en sistemas de información	6
2.6 Producto Mínimo Viable	7
Capítulo 3: Estado actual de AppCopio	8
3.1 Descripción general del sistema	8
3.1.1 Usuarios y roles del sistema	8
3.1.2 Funcionalidades principales	9
3.1.3 Acceso público y visualización	9
3.2 Arquitectura técnica	10
3.2.1 Stack tecnológico	10
3.2.2 Despliegue de la aplicación	10
3.2.3 Diagrama de arquitectura general	11
3.3 Modelo de datos actual	11
3.3.1 Entidades y relaciones	12
3.4 Limitaciones frente a escenario multi-municipal	12
Capítulo 4: Diseño de la solución	14
4.1 Objetivos y restricciones de diseño	14
4.2 Estrategias multi-tenant	15
4.3 Diseño de la extensión multi-tenant	17
4.3.1 Rediseño del modelo de datos	17
4.3.2 Nuevo rol de administración de plataforma	19
4.3.3 Diseño del aislamiento de datos y políticas de seguridad	20
4.3.4 Diseño del modelo de colaboración inter-municipal	22

4.4	Consideraciones de migración	28
4.5	Alcance y exclusiones de diseño	28
Capítulo 5:	Implementación de la solución	29
5.1	Entorno de desarrollo y configuración del proyecto	29
5.2	Implementación de multi-tenant	29
5.2.1	Resolución del tenant	29
5.2.2	Propagación del discriminador en el esquema	31
5.2.3	Políticas de aislamiento y sus excepciones	33
5.2.4	Contexto por transacción sobre el código existente	34
5.2.5	Catálogo de funciones con privilegios elevados	34
5.3	Implementación de la colaboración inter-municipal	35
5.3.1	Endpoints y lectura cruzada acotada	35
5.3.2	Notificaciones dirigidas	36
5.4	Implementación en el frontend	37
5.4.1	Cómo el cliente conoce su comuna	37
5.4.2	Pantallas nuevas	37
5.4.3	El cliente no es la frontera de seguridad	38
5.5	Desafíos de implementación	39
Capítulo 6:	Validación de la implementación	40
6.1	Escenario 1: Aislamiento estricto de datos	41
6.2	Escenario 2: Colaboración inter-municipal con consentimiento	42
6.3	Escenario 3: Gobernanza de administración	43
6.4	Escenario 4: Continuidad funcional y canal público	45
6.5	Escenario 5: Cierre, retiro y reingreso a la colaboración	46
6.6	Discusión de resultados	48
6.7	Síntesis de resultados	49
Capítulo 7:	Conclusiones	50
7.1	Cumplimiento de los objetivos	50
7.2	Aporte del trabajo	50
7.3	Alcance y limitaciones	51
7.4	Trabajo futuro	52
7.4.1	Lo que este trabajo dejó fuera	52
7.4.2	Evolución natural del producto	52
7.5	Reflexión final	52
Referencias bibliográficas		53
Anexos		55

Índice de figuras

Figura 1	Módulos funcionales de AppCopio. Fuente: Elaboración propia.	9
Figura 2	Diagrama de arquitectura C4, Nivel 2 – <i>Container Diagram</i> . Fuente: Elaboración propia.	11
Figura 3	Modelo de datos simplificado de AppCopio. Fuente: Elaboración propia.	11
Figura 4	Modelo de datos extendido de AppCopio. Fuente: Elaboración propia.	19
Figura 5	Cadena de tres eslabones para aislamiento de datos. Fuente: Elaboración propia.	20
Figura 6	Resolución del contexto al inicio de cada petición. Fuente: Elaboración propia.	20
Figura 7	Los dos niveles de evento del modelo de colaboración. Fuente: Elaboración propia.	22
Figura 8	Modelo de ingreso de municipalidades a SuperEventos. Fuente: Elaboración propia.	24
Figura 9	Ciclo de vida de una oferta de apoyo inter-municipal. Fuente: Elaboración propia.	26
Figura 10	Emisión del token de sesión durante el <i>login</i> . Fuente: Elaboración propia.	30
Figura 11	Ciclo de una petición autenticada, posterior al <i>login</i> de la Figura 10: de la resolución del tenant a la evaluación de la política. Fuente: Elaboración propia.	30
Figura 12	Ruta de doble uso: el mismo endpoint bajo dos contextos excluyentes (Figura 6). Fuente: Elaboración propia.	31
Figura 13	Emergencias de la comuna.	62
Figura 14	Gestión de centros de una emergencia. Fuente: Elaboración propia.	62
Figura 15	SuperEventos de la comuna. Fuente: Elaboración propia.	63
Figura 16	Aceptación de una invitación a un SuperEvento. Fuente: Elaboración propia.	63
Figura 17	Tablero intercomunal, vista de listado.	64
Figura 18	Tablero intercomunal, vista de mapa. Fuente: Elaboración propia.	64
Figura 19	Bandeja de ofertas de apoyo. Fuente: Elaboración propia.	65
Figura 20	Panel de Super Administrador. Fuente: Elaboración propia.	65
Figura 21	Panel de Super Administrador, vista de SuperEventos. Fuente: Elaboración propia.	66
Figura 22	Pantalla «Mis centros», ampliada por la colaboración. Fuente: Elaboración propia.	66
Figura 23	Listado «Mis Centros» en dos sesiones simultáneas. Fuente: Elaboración propia.	69
Figura 24	Intento de acceso por URL a un centro de otra comuna. Fuente: Elaboración propia.	70
Figura 25	Oferta de apoyo sincronizada entre dos comunas. Fuente: Elaboración propia.	70
Figura 26	Mapa público sin sesión iniciada. Fuente: Elaboración propia.	71
Figura 27	Módulo de inventario de un centro. Fuente: Elaboración propia.	71
Figura 28	Efecto del cierre de un SuperEvento en la interfaz. Fuente: Elaboración propia.	72
Figura 29	Salida de <code>scripts/validar_multitenant.sh</code> : 87 de 87 comprobaciones con resultado PASS. Fuente: Elaboración propia.	78

Índice de tablas

Tabla 1	Niveles de eventos en gestión de emergencias. Fuente: Elaboración propia a partir del marco legal.	5
Tabla 2	Roles del sistema AppCopio y sus permisos. Fuente: Elaboración propia.	8
Tabla 3	Requisitos funcionales para el PMV de AppCopio multi-municipal. Fuente: Elaboración propia. .	14
Tabla 4	Requisitos no funcionales para el PMV de AppCopio multi-municipal. Fuente: Elaboración propia.	15
Tabla 5	Análisis comparativo de estrategias multi-tenant en bases de datos relacionales. Fuente: Elaboración propia.	16
Tabla 6	Entidades nuevas del modelo multi-tenant. Fuente: Elaboración propia.	17
Tabla 7	Patrones de propagación del discriminador de tenencia. Fuente: Elaboración propia.	18
Tabla 8	Origen del discriminador de comuna al crear cada entidad. Fuente: Elaboración propia.	22
Tabla 9	Límite de confianza entre municipalidades dentro de un SuperEvento. Fuente: Elaboración propia.	25
Tabla 10	Desglose de las funciones SECURITY DEFINER por motivo. Fuente: Elaboración propia.	34
Tabla 11	Endpoints nuevos de colaboración inter-municipal. Fuente: Elaboración propia.	35
Tabla 12	Pantallas nuevas del frontend multi-tenant. Fuente: Elaboración propia.	37
Tabla 13	Síntesis de la matriz de trazabilidad por modo de acreditación (detalle requisito por requisito en el Anexo F). Fuente: Elaboración propia.	40
Tabla 14	Usuarios de prueba por comuna; además existe superadmin, sin comuna asociada. Fuente: Elaboración propia.	40
Tabla 15	Verificaciones de aislamiento, Escenario 1 (selección representativa, listado completo en el Anexo I). Fuente: Elaboración propia.	41
Tabla 16	Verificaciones de colaboración inter-municipal, Escenario 2 (selección representativa, listado completo en el Anexo I). Fuente: Elaboración propia.	42
Tabla 17	Verificaciones de alcance del Super Administrador y alta de municipalidades, Escenario 3. Fuente: Elaboración propia.	44
Tabla 18	Verificaciones de continuidad funcional y canal público, Escenario 4. Fuente: Elaboración propia.	45
Tabla 19	Verificaciones de cierre, retiro y reingreso a la colaboración, Escenario 5. Fuente: Elaboración propia.	46
Tabla 20	Catálogo de funciones con privilegios elevados. Fuente: Elaboración propia.	58
Tabla 21	Matriz de trazabilidad entre los requisitos del PMV y su forma de acreditación. Fuente: Elaboración propia.	67
Tabla 22	Verificaciones de aislamiento, Escenario 1. Fuente: Elaboración propia.	75
Tabla 23	Verificaciones de colaboración inter-municipal, Escenario 2. Fuente: Elaboración propia.	76

Capítulo 1

Introducción

En las últimas décadas, Chile se ha caracterizado por ser uno de los países con mayor riesgo de desastres a nivel global. El informe de Riesgo Mundial ubica a Chile en el puesto 39 de los países con mayor índice de riesgo de catástrofes naturales del planeta [1], resultado de una combinación particular de exposición sísmica, volcánica y climática que rara vez se repite en otras latitudes. Específicamente, por sus características geográficas y geológicas, el 54% de la población y el 12,9% de la superficie total del país están expuestos a tres o más tipos de amenazas naturales simultáneas, como terremotos, tsunamis, inundaciones, incendios forestales, entre otros [2]. Este riesgo no queda solo en el plano estadístico: según varios informes ([3], [4]), se calcula que el país gasta anualmente entre un 1,2%–1,5% de su Producto Interno Bruto (PIB)¹ únicamente en tareas de respuesta y reconstrucción post-desastre. Es, en otras palabras, un costo recurrente y estructural, no un gasto excepcional.

En este escenario, la ley chilena radica en las municipalidades la responsabilidad de gestionar la emergencia a nivel local [5], por ser la instancia estatal con contacto más directo con el territorio afectado. Sin embargo, en el contexto de disparidad socio-territorial del país, las capacidades técnicas y presupuestarias varían notoriamente de una comuna a otra, y la evidencia académica advierte que las debilidades institucionales a nivel municipal pueden transformarse en un factor subyacente de riesgo que incrementa la vulnerabilidad territorial en contextos de crisis [6]. Este mandato legal tampoco ha sido acompañado de una modernización tecnológica equivalente: en la mayoría de los municipios, la coordinación de albergues y centros de acopio continúa apoyándose en metodologías manuales o herramientas ofimáticas genéricas y diseñadas para tareas cotidianas, no para la gestión de una crisis, como planillas de Google Drive o formularios de Office 365. Estos sistemas carecen de las propiedades que exige un entorno de gestión crítica –interoperabilidad entre distintas administraciones, resiliencia ante fallas de conectividad, trazabilidad de información en tiempo real y capacidad de operación intercomunal–, por lo que no solo limitan la autonomía y agilidad de los equipos en terreno, sino que también generan información fragmentada, duplicidad de esfuerzos y retraso en la toma de decisiones. Asimismo, al carecer de la capacidad de filtrar y traducir datos hacia la ciudadanía, las autoridades necesitan realizar esfuerzo adicional para reportar la situación actual a la comunidad; en otras palabras, deja tanto a los damnificados como a quienes deseen ayudar sin un canal oficial de información actualizado en tiempo real.

Las consecuencias de operar en estas condiciones se hicieron evidentes durante el megaincendio del 2 y 3 de febrero de 2024 en la Región de Valparaíso, uno de los desastres con mayor número de víctimas del país en las últimas tres décadas [7]. El balance –más de 130 personas fallecidas, decenas de miles de viviendas destruidas y sobre 40.000 damnificados, donde un 60% de ellos pertenecía al tramo más vulnerable del Registro Social de Hogares [8]– estuvo atravesado por graves fallas de coordinación operativa y alertas tempranas. La falta de un sistema que mostrara en tiempo real dónde estaba la necesidad real generó un desajuste crítico: mientras algunos puntos de acopio recibían más donaciones de las que podían procesar durante las primeras 48 horas, sectores rurales completos permanecieron sin ayuda básica por más de 72 horas debido a la falta de visibilidad centralizada. Ese mismo diagnóstico, levantado en terreno junto a la sección de Emergencia Social de la Municipalidad de Valparaíso, dio origen a AppCopio, una plataforma creada para otorgar a un municipio visibilidad en tiempo real sobre sus centros de acopio y capacidad de operar sin conexión continua. Su versión actual, sin embargo, fue construida para atender a una sola administración municipal, y no contempla que dos o más comunas coordinen sus recursos sobre una misma instancia de la plataforma.

Esa limitación pesa más de lo que podría parecer a primera vista, considerando que las emergencias que enfrenta Chile tienden a desbordar los límites de una sola comuna. Entre 2017 y 2026 el país ha encadenado episodios que escalaron a catástrofes regionales: los incendios forestales de 2017 y 2023, la megacatástrofe de la Región de Valparaíso de febrero de 2024 –que golpeó simultáneamente a Valparaíso, Viña del Mar, Quilpué y Villa Alemana– y el sistema frontal de julio de 2026, calificado como el más severo en casi 40 años, que afectó

¹ Entre 3,8–4,5 mil millones de dólares.

al menos a 10 regiones, varias de ellas bajo estado de catástrofe. Si esta fragmentación entre comunas no se resuelve, cada evento de gran magnitud repetirá el mismo patrón: municipios vecinos sin visibilidad mutua, ayuda mal distribuida y demoras en la atención de zonas aisladas. A esto se suma un factor de infraestructura crítico: según la Subsecretaría de Telecomunicaciones (SUBTEL), la operatividad de las radiobases en zonas afectadas puede caer hasta un 35% durante las primeras horas de una catástrofe mayor [9], por lo que un sistema estrictamente dependiente de conexión permanente resulta, por definición, insuficiente para responder a esta realidad.

Frente a este desafío existen iniciativas chilenas orientadas a la gestión de riesgos y emergencias municipales, aunque ninguna aborda específicamente la coordinación operativa intercomunal para centros de acopio y albergues. El Sistema de Gestión de Emergencias Regional (SIGER), impulsado por la Asociación Chilena de Municipalidades, entrega herramientas de análisis territorial y datos de vulnerabilidad para apoyar la planificación preventiva [10]; su aporte, sin embargo, se concentra en la etapa de prevención y análisis de datos, no en la gestión logística y operativa en tiempo real durante la emergencia. Algo similar ocurre con la Plataforma de Apoyo a la Gestión de Emergencias desarrollada por CITIAPS de la Universidad de Santiago, que incorpora herramientas de crowdsourcing, coordinación de voluntariado y asistencia ciudadana [11], pero carece de un módulo dedicado a la administración municipal, de interoperabilidad técnica entre distintas municipalidades y requiere de conexión estable para operar. En ambos casos, el problema de fondo persiste: en el ecosistema de herramientas municipales chilenas, hoy no existe una plataforma que permita a múltiples comunas coordinar en tiempo real –y con capacidad de funcionar sin conexión continua– la gestión de sus centros de acopio y albergues durante una emergencia de gran escala.

Así, para responder a la necesidad de una herramienta que permita a las municipalidades chilenas coordinar la gestión de emergencias más allá de sus propios límites administrativos, el presente proyecto no propone el desarrollo de un sistema desde cero, sino la evolución arquitectónica de AppCopio. El valor técnico y aplicado de este trabajo radica en superar sus limitaciones actuales –operar de forma aislada para un único municipio– y transformarlo en una plataforma multi-municipal (multi-tenant). De este modo, se preserva su capacidad de operación en escenarios con baja conectividad, pero dotando al sistema de la escalabilidad necesaria para que comunas vecinas puedan colaborar, visibilizar recursos y gestionar emergencias de escala regional sobre una misma infraestructura.

1.1. Objetivo general

Diseñar una versión multi-municipal (multi-tenant) de AppCopio e implementar un PMV sobre la base actual del sistema que permita operar espacios separados por municipalidad y la colaboración entre comunas mediante un tablero, reglas mínimas de visibilidad y un flujo simple de intercambio de necesidades y ofertas.

1.2. Objetivos específicos

Los objetivos específicos del presente trabajo son:

1. Documentar el estado actual de AppCopio, levantando roles de usuario, modelo de base de datos y funcionalidades existentes, identificando limitaciones relevantes para evolución a la nueva versión.
2. Levantar requerimientos funcionales y no funcionales para definir el alcance del PMV de la extensión multi-municipalidad.
3. Diseñar la arquitectura multi-tenant de AppCopio, garantizando el aislamiento de datos entre municipalidades, y definir los cambios a realizar en arquitectura y modelo de datos necesarios para la nueva versión.
4. Implementar un PMV de separación por municipalidad basado en los requerimientos para la nueva versión.

5. Diseñar e implementar un mecanismo simple de colaboración intermunicipal, acotado al contexto de una emergencia compartida.

1.3. Metodología y estructura del documento

Dado que este trabajo hace evolucionar una plataforma funcional y validada (AppCopio) en lugar de construir un sistema desde cero, la metodología se fundamenta en una estricta trazabilidad: toda limitación diagnosticada genera un requisito, todo requisito guía una decisión de diseño, y toda decisión se contrasta con evidencia. Tras establecer las bases conceptuales sobre gestión de emergencias y arquitecturas multi-tenant (Capítulo 2), el desarrollo inicia levantando el estado actual del sistema –usuarios, roles, funcionalidades, arquitectura y modelo de datos– para diagnosticar las barreras que impiden su operación multi-municipal (Capítulo 3).

Utilizando este diagnóstico como entrada, se definen los requerimientos del Producto Mínimo Viable (PMV) y se elabora su diseño (Capítulo 4). Esta fase evalúa las alternativas arquitectónicas de la literatura, define la estrategia multi-tenant y diseña los modelos de datos, aislamiento y colaboración inter-municipal. Posteriormente, este diseño se materializa de forma aditiva sobre el código fuente existente (Capítulo 5), detallando la construcción de los nuevos esquemas y los desafíos técnicos que la propia implementación reveló.

Finalmente, el resultado se somete a verificación (Capítulo 6). Respondiendo al carácter de PMV, no se emplea una batería exhaustiva de pruebas de software, sino escenarios de uso crítico y comprobaciones automatizadas que acreditan que los mecanismos centrales operan según lo especificado. El documento concluye (Capítulo 7) recogiendo los resultados obtenidos, el aporte del trabajo, las limitaciones de la solución y las líneas de desarrollo futuro.

Capítulo 2

Marco teórico

2.1. Gestión de emergencias en Chile

La gestión del riesgo de desastres en Chile se encuentra regulada por la Ley N° 21.364, promulgada en 2021, que creó el Sistema Nacional de Prevención y Respuesta ante Desastres (SINAPRED) y sustituyó a la antigua Oficina Nacional de Emergencia (ONEMI) por el Servicio Nacional de Prevención y Respuesta ante Desastres (SENAPRED) [5]. La norma establece un modelo de gestión descentralizado y escalonado que opera en los niveles comunal, provincial, regional y nacional, situando como ejes rectores del sistema los principios de coordinación, escalabilidad, apoyo mutuo y transparencia.

2.1.1. Ley 21.364

La Ley N° 21.364 estructura formalmente el ciclo del riesgo de desastres en cuatro fases: mitigación, preparación, respuesta y recuperación [5]. Mientras que las fases de mitigación y preparación se enfocan en la reducción de vulnerabilidades de forma preventiva, las fases de respuesta y recuperación se activan ante la ocurrencia de un evento destructivo.

El principio de escalabilidad, normado en el Artículo 4°, establece que la atención de una emergencia debe ser asumida prioritariamente por el nivel más cercano a la afectación (comunal), movilizandolos recursos de niveles superiores únicamente a medida que las capacidades locales resulten superadas [5]. Para la coordinación operativa, la ley crea los Comités para la Gestión del Riesgo de Desastres (COGRID) en cada nivel territorial.

Asimismo, la ley establece un conjunto de instrumentos formales de gestión, destacando los sistemas de información y comunicaciones, los cuales deben garantizar la trazabilidad y la diseminación de datos entre los actores del SINAPRED [5]. No obstante, existe una brecha crítica entre el mandato legal de coordinación intercomunal e interinstitucional y las herramientas tecnológicas disponibles en la fase de respuesta en el nivel local.

2.1.2. Conceptos operativos relevantes

Para efectos del presente trabajo, resulta necesario precisar la definición operativa de los componentes clave de la respuesta comunal:

- **Centro de acopio:** recinto municipal habilitado para la recepción, clasificación, inventariado y distribución de donaciones e insumos de ayuda humanitaria destinados a la población afectada [12].
- **Albergue:** instalación habilitada para otorgar alojamiento, alimentación y asistencia básica temporal a personas evacuadas o damnificadas² [12].
- **Ficha Básica de Emergencia (FIBE):** instrumento catastral oficial del Ministerio de Desarrollo Social y Familia utilizado para recopilar información sobre los hogares afectados y sus necesidades básicas tras una catástrofe [13].
- **Fase de preparación:** corresponde a las actividades que se desarrollan para prever, responder y recuperarse de forma oportuna y eficaz de los impactos de amenazas inminentes o emergencias [5].
- **Fase de respuesta:** conjunto de acciones operativas propias de atención de una emergencia, desarrolladas de manera inmediata ante la ocurrencia de un evento destructivo, cuyo objetivo es salvar vidas, reducir el sufrimiento y mitigar pérdidas materiales [5].

²En lo sucesivo, y con el fin de evitar reiteraciones, se empleará el término «centro» para hacer referencia a cualquiera de estas dos instalaciones de gestión municipal.

Uno de los elementos centrales del marco legal es la clasificación de los eventos según su magnitud y la capacidad de respuesta requerida:

Nivel de evento	Definición y alcance	Capacidad de respuesta requerida
Emergencia menor	Evento local con afectación acotada a una zona o comuna.	Capacidades y recursos exclusivamente comunales.
Emergencia mayor	Evento que sobrepasa la capacidad de respuesta de la comuna afectada.	Apoyo o intervención del nivel provincial o regional.
Desastre	Afectación severa que sobrepasa la capacidad de respuesta regional.	Mobilización y reasignación de recursos a nivel nacional.
Catástrofe	Afectación de tal magnitud que excede la capacidad de respuesta del país.	Coordinación del Gobierno Central y asistencia internacional.

Tabla 1: Niveles de eventos en gestión de emergencias. Fuente: Elaboración propia a partir del marco legal.

Esta clasificación resulta crítica para la delimitación del alcance de este trabajo: define el espacio del problema que AppCopio en su versión monomunicipal (*single-tenant*) no puede resolver. Por definición, cualquier evento categorizado como emergencia mayor, desastre o catástrofe requiere coordinación y flujo de datos inter-comunal, requiriendo un soporte tecnológico que apoye operativamente dicho nivel de escalabilidad.

En el nivel comunal, las municipalidades son los actores primarios de la respuesta a través de sus equipos de emergencia social radicados en la Dirección de Desarrollo Comunitario (DIDECO) [12]. Actualmente, la gestión operativa de centros y recursos en este nivel se apoya en herramientas ofimáticas genéricas (planillas de cálculo y mensajería instantánea), las cuales carecen de mecanismos de sincronización en tiempo real, trazabilidad de inventarios e interoperabilidad territorial [14].

2.2. Arquitectura single-tenant

La arquitectura single-tenant (o de instancia dedicada) es un patrón de diseño de software en el cual cada cliente (tenant) dispone de una instancia independiente de la aplicación, del entorno de ejecución y de la base de datos [15].

Entre sus principales ventajas destacan el aislamiento completo de los datos a nivel físico o lógico, la facilidad para personalizar componentes por cliente y un impacto acotado ante fallos de infraestructura. Sin embargo, este esquema presenta desventajas operativas y económicas significativas: altos costos de infraestructura acumulados, redundancia de mantenimiento y la necesidad de replicar despliegues e intervenciones de código individualmente por cada cliente [15].

En el estado inicial de este trabajo, AppCopio fue construido bajo una arquitectura single-tenant: cada municipalidad requiere un despliegue aislado de la aplicación y de su base de datos. Si bien este enfoque es suficiente para gestionar emergencias menores a nivel local, constituye un obstáculo estructural frente a emergencias mayores o desastres, imposibilitando la visibilidad compartida de insumos, albergues y necesidades entre municipios colindantes.

2.3. Arquitectura multi-tenant

La arquitectura multi-tenant es un patrón de diseño de software en el que una única instancia de la aplicación sirve simultáneamente a múltiples clientes (tenants), garantizando que sus datos y configuraciones se mantengan lógicamente aislados [16]. Este patrón constituye el fundamento del modelo Software como Servicio (SaaS) moderno, destacando por su eficiencia operativa, reducción de costos de infraestructura y centralización del ciclo de vida del software.

En la literatura de ingeniería de software se identifican distintas estrategias para implementar el aislamiento de datos en bases de datos relacionales –desde una base de datos completamente separada por tenant hasta un esquema compartido con una columna discriminadora (*tenant_id*)–, cada una con distintos costos de aislamiento, seguridad y eficiencia de recursos [16]. La selección e implementación de la estrategia adecuada para AppCopio se aborda en detalle en la Sección 4.2, en el marco del diseño de la solución.

2.3.1. Row-Level Security

Cuando la estrategia escogida es la de esquema compartido, el aislamiento entre tenants deja de ser una propiedad de la infraestructura y pasa a ser una propiedad de las consultas: dos clientes distintos leen las mismas tablas físicas, y lo único que los separa es que cada consulta filtre por el discriminador correcto. Confiar esa separación exclusivamente a la capa de aplicación es frágil, porque basta que un solo endpoint omita el filtro para exponer datos de otro cliente.

Row-Level Security (RLS) es el mecanismo con que PostgreSQL traslada ese filtrado desde la aplicación al motor de base de datos [17]: toda fila leída o escrita debe satisfacer una política declarada en el esquema –*USING* para lectura, *WITH CHECK* para escritura–, de modo que el filtro se aplica aunque la consulta no la incluya. Tres comportamientos del motor son relevantes para su uso posterior: las políticas se combinan por disyunción, de modo que activar RLS sin ninguna política permisiva niega todo acceso en vez de dejarlo abierto; por defecto el dueño de la tabla y los roles con privilegios elevados eluden sus propias políticas, salvo que se declare *FORCE ROW LEVEL SECURITY*; y las excepciones legítimas que necesitan ver más allá del cliente activo se resuelven con funciones *SECURITY DEFINER*, cuyo uso debe acotarse deliberadamente por ser, en sí mismas, una vía que elude el aislamiento [18].

2.4. Control de acceso y defensa en profundidad

El aislamiento entre organizaciones responde a la pregunta de qué datos puede alcanzar un cliente, pero no a la de qué puede hacer cada persona dentro de él. Esa segunda pregunta la resuelve el Control de Acceso Basado en Roles (RBAC), formulado por Sandhu et al. y normalizado como estándar ANSI/INCITS [19], [20]: los permisos no se asignan directamente a las personas, sino a roles, y las personas se asignan a roles, de modo que la administración escala con el número de roles y no con el de usuarios.

En arquitecturas web sin estado, el rol viaja en la credencial de sesión –típicamente un JWT firmado– y se verifica en el servidor antes de resolver cada petición: el cliente puede conocer el rol y usarlo para decidir qué mostrar, pero el control efectivo reside en esa verificación, porque el cliente puede ser modificado o sorteado por completo. De ahí el principio de defensa en profundidad –disponer controles independientes en capas sucesivas, de modo que la falla de una no comprometa al conjunto [21]–, aplicado aquí en tres capas de autoridad creciente: la interfaz, que oculta lo que no corresponde; la aplicación, que valida cada petición contra la credencial verificada; y el motor de base de datos, que aplica el aislamiento con independencia de las capas anteriores. Este principio estructura el diseño que se desarrolla en la Sección 4.3.3.

2.5. Colaboración inter-organizacional en sistemas de información

La colaboración inter-organizacional mediante sistemas de información aborda la capacidad de entidades autónomas –como las municipalidades– para intercambiar datos estructurados y coordinar procesos operativos sobre una plataforma tecnológica compartida [22]. En el sector público, la interoperabilidad representa un desafío complejo debido a barreras organizacionales, diferencias de procesos locales y reticencia al intercambio de información [22].

En la gestión de emergencias, esta dinámica se analiza mediante la transición del paradigma *need-to-know* al paradigma *need-to-share*:

- **Need-to-know:** Modelo restrictivo por defecto, donde la información solo se revela a un actor si este justifica explícitamente su necesidad operativa.
- **Need-to-share:** Modelo colaborativo proactivo, que promueve la visibilidad de la información entre actores interdependientes para optimizar la toma de decisiones oportuna durante una crisis [23].

Este trabajo se sitúa en la tensión entre ambos paradigmas: cómo habilitar un mayor grado de visibilidad inter-comunal sin comprometer datos sensibles de los hogares afectados (como la Ficha FIBE). El diseño concreto de este equilibrio –qué información se comparte, en qué agregación y entre qué actores– se desarrolla en la Sección 4.3.4.

2.6. Producto Mínimo Viable

El concepto de Producto Mínimo Viable (PMV), originado en la metodología Lean Startup, define la versión de un producto que reúne las características suficientes para ser desplegado con usuarios reales, permitiendo validar hipótesis operativas y aprender mediante retroalimentación empírica con el menor esfuerzo de desarrollo [24].

En el marco de esta memoria, la construcción del PMV presenta una restricción técnica particular: no se desarrolla como un sistema desde cero, sino como una refactorización evolutiva sobre una plataforma preexistente y funcional.

La literatura sobre evolución de software advierte que la introducción de nuevas capacidades arquitectónicas sobre software existente exige que las decisiones de diseño mantengan compatibilidad con el sistema base, preserven la estabilidad de los módulos funcionales en producción y dejen la arquitectura en un estado coherente para iteraciones futuras [25]. El alcance del PMV se delimita priorizando los requisitos clave, garantizando que los requerimientos no incluidos en esta etapa queden proyectados como trabajo futuro sobre la nueva estructura multi-tenant.

Capítulo 3

Estado actual de AppCopio

3.1. Descripción general del sistema

AppCopio es una aplicación web de gestión operativa de centros de acopio y albergues durante emergencias. Su propósito central es habilitar el flujo de información en tiempo real entre tres actores: la municipalidad –específicamente, sus equipos de emergencia social–, los encargados de centro desplegados en terreno y la comunidad afectada, permitiendo que los centros sean catastrados y evaluados con anterioridad a la ocurrencia de una emergencia y su activación se reduzca a la disposición de un catálogo ya existente en vez de a un levantamiento de información desde cero. De esta forma, el sistema acompaña tanto la fase de preparación como la fase de respuesta ante una emergencia.

En su estado actual, AppCopio opera bajo una arquitectura single-tenant, desplegada exclusivamente para la Ilustre Municipalidad de Valparaíso: la totalidad de la información gestionada por el sistema –centros, inventarios y personas afectadas– se encuentra acotada al ámbito comunal, sin mecanismos de visibilidad ni coordinación con otras municipalidades.

3.1.1. Usuarios y roles del sistema

Para comprender el diseño de la plataforma, es necesario distinguir entre los actores de la comunidad que interactúan con ella y cómo el sistema traduce su participación en niveles de acceso mediante un Control de Acceso Basado en Roles (RBAC, Sección 2.4).

En el modelo de operación real, se identifican tres grupos de actores:

- **Equipo municipal:** personal operativo de la sección de emergencia social (DIDECO).
- **Contacto ciudadano:** vecinos o dirigentes comunitarios que cumplen un rol preventivo como enlace local en territorio.
- **Usuario público:** cualquier persona de la comunidad (damnificados, donantes, voluntarios).

A nivel de base de datos, el sistema mapea a estos actores en roles específicos que determinan sus permisos:

- Los miembros del **equipo municipal** asumen los roles de **Administrador** (para quien dirige el equipo) o **Trabajador municipal**, dependiendo de si requieren control global o acotado a centros específicos.
- El **contacto ciudadano** opera bajo un rol homónimo de menor privilegio, vinculado únicamente a su centro.
- El **usuario público** no tiene cuenta en el sistema, y, por tanto, no se autentica; su interacción se limita a la vista pública.

Adicionalmente, el sistema contempla un mecanismo de delegación de permisos. Este le permite al Administrador activar a cualquier Trabajador municipal como «Apoyo de administrador» (en adelante, *apoyo administrativo*), otorgándole sus mismos privilegios de gestión sin alterar el rol base del usuario, permitiendo mitigar la carga operativa del Administrador de forma temporal en base a las necesidades del municipio.

El detalle de los permisos y el alcance de cada rol dentro de la plataforma se consolida en la siguiente tabla:

Rol	Descripción y permisos en el sistema
Administrador	Control global. Permite gestionar todos los centros y administrar a los demás usuarios.
Trabajador municipal	Gestión acotada. Operador responsable de la gestión en terreno, inventarios y atención de residentes, con permisos limitados exclusivamente a sus centros asignados.

Contacto ciudadano	Acceso restringido. Perfil preventivo que actúa como enlace local con el equipo municipal durante la fase de preparación, facultado únicamente para enviar actualizaciones asociadas al estado de su centro asignado.
Usuario público	Sin cuenta. Acceso exclusivo de lectura para consultar información del mapa interactivo y avisos en emergencia.

Tabla 2: Roles del sistema AppCopio y sus permisos. Fuente: Elaboración propia.

3.1.2. Funcionalidades principales

En base a los perfiles y roles definidos, las funcionalidades de la aplicación se agrupan en cuatro módulos, que se resumen en la Figura 1:

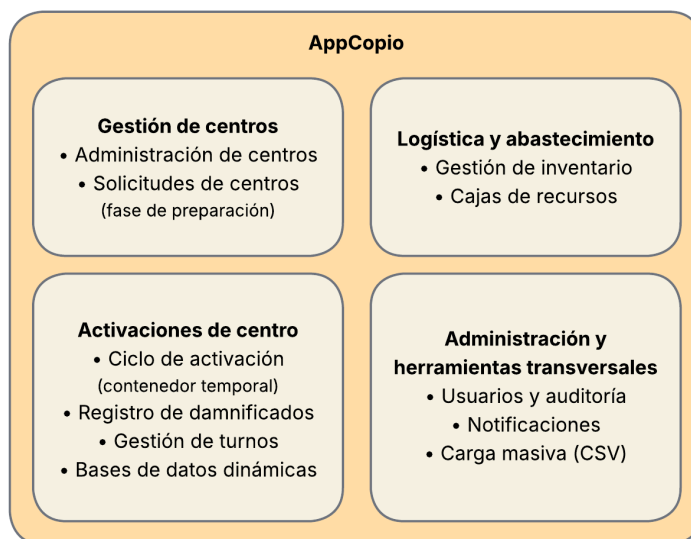


Figura 1: Módulos funcionales de AppCopio. Fuente: Elaboración propia.

El módulo de gestión de centros sostiene la fase de preparación: mantiene la ficha de cada centro y canaliza las solicitudes de actualización que el contacto ciudadano levanta desde terreno. El de activaciones de centro concentra la fase de respuesta en torno a la activación, un contenedor temporal que aísla los registros de damnificados, los turnos del personal y las estructuras de datos ad-hoc de un periodo de emergencia. El de logística y abastecimiento controla el stock por centro y su entrega a familias beneficiarias. Finalmente, las herramientas transversales cubren la administración de usuarios, la auditoría de cambios, las notificaciones y la carga masiva de datos. El detalle de las funcionalidades por módulo se encuentra en el Anexo A.

Cabe destacar que, dado el contexto crítico en que debe operar el sistema, estas funcionalidades no dependen de una red estable: ante una caída de conectividad, el sistema transiciona a un modo de funcionamiento offline que retiene la interfaz gráfica y permite continuar registrando datos localmente. Una vez se restablece la conexión a internet, el sistema sincroniza automáticamente la información con el servidor central.

3.1.3. Acceso público y visualización

A diferencia de los módulos anteriores, orientados a gestión operativa por parte del equipo municipal, el sistema contempla una capa de visualización que atraviesa dicha información y la expone públicamente sin requerir autenticación:

- **Mapa interactivo:** despliegue cartográfico que expone de forma georreferenciada los centros de una comuna. Permite a los usuarios visualizar los centros activos durante una emergencia, localizar los más próximos

a su ubicación actual, identificar su estado operacional mediante un sistema de semáforos y consultar un resumen de sus necesidades priorizadas. El alcance de la información expuesta depende del tipo de usuario: el usuario público solo visualiza los centros activos, con información pública y agregada; el Administrador y los Trabajadores municipales, en cambio, tienen acceso visual a la totalidad del catastro, independientemente de si los centros se encuentran activos o inactivos.

El usuario público no interactúa con los módulos de gestión descritos en la sección anterior: su acceso se limita a esta capa de visualización pública.

3.2. Arquitectura técnica

3.2.1. Stack tecnológico

AppCopio implementa una arquitectura cliente-servidor, compuesta por los siguientes componentes principales:

- **Frontend:** interfaz de usuario desarrollada con React 19 y TypeScript, utilizando Material UI v7 para el diseño responsivo de componentes y Vite como herramienta de empaquetado. La visualización de mapas se integra mediante la API de Google Maps. Además, el sistema cuenta con soporte para funcionamiento sin conexión (*offline*) operado a través de *Service Workers* (*vite-plugin-pwa*), almacenamiento local con IndexedDB (*idb*) y gestión del estado de datos remotos mediante TanStack Query.
- **Backend:** lógica de negocio expuesta mediante una API RESTful construida sobre Node.js y Typescript, con el framework Express 5. La seguridad del sistema implementa autenticación basada en JSON Web Tokens (JWT), utilizando un esquema dual de *access token* y *refresh token*.
- **Base de datos:** administrada por un motor relacional PostgreSQL 15.

3.2.2. Despliegue de la aplicación

Históricamente, el despliegue de validación de AppCopio –durante el año 2025, junto a la Municipalidad de Valparaíso– se realizó de forma desacoplada: el *frontend* en Vercel, y el *backend* junto con la base de datos PostgreSQL en Render. Si bien esta combinación facilitó una puesta en marcha rápida, introdujo discrepancias entre los entornos locales y el ambiente productivo, tales como diferencias en variables de entorno, versiones de Node.js y políticas de red o CORS entre dominios independientes.

Para mitigar estas complicaciones y estandarizar el flujo de trabajo, el desarrollo actual se centralizó mediante contenedores con Docker Compose. Esta configuración orquesta tres servicios locales:

- **Base de datos:** PostgreSQL 15, inicializada automáticamente mediante *scripts* dedicados (*db_init/*).
- **Backend:** Servicio basado en la imagen *node:22-bookworm* con soporte para *hot-reload*.
- **Frontend:** Aplicación servida sobre la imagen *node:22-bookworm*, también con recarga en caliente habilitada.

Esta estrategia permite homogeneizar el entorno de desarrollo y reproducir un comportamiento lo más cercano posible al de producción, eliminando costos adicionales de infraestructura. Por tanto, para efectos de las pruebas y la validación del presente trabajo, AppCopio se evalúa desplegado en este entorno local compilado en modo de producción (*Local Production Build*).

3.2.3. Diagrama de arquitectura general

La Figura 2 presenta el diagrama de arquitectura C4 de AppCopio:

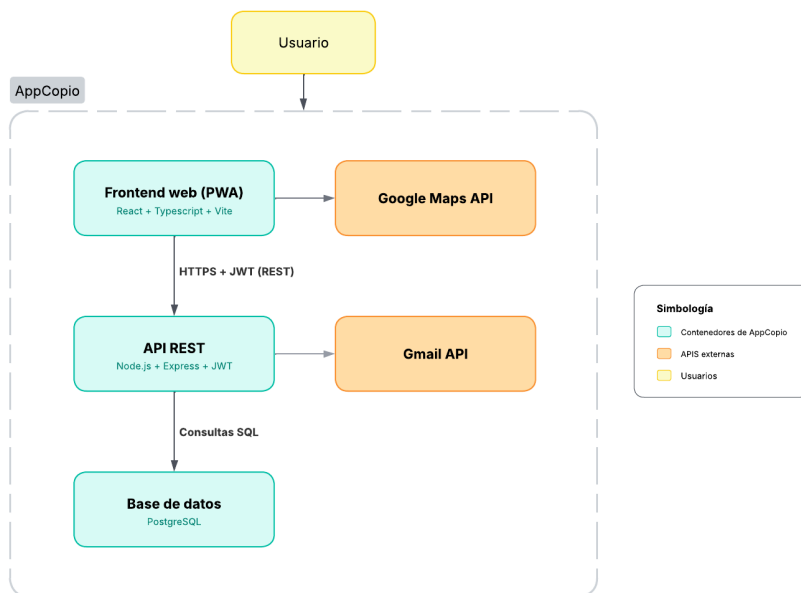


Figura 2: Diagrama de arquitectura C4, Nivel 2 – *Container Diagram*. Fuente: Elaboración propia.

3.3. Modelo de datos actual

El siguiente modelo de datos resume la estructura general de la base de datos de AppCopio y las entidades centrales del modelo relacional.

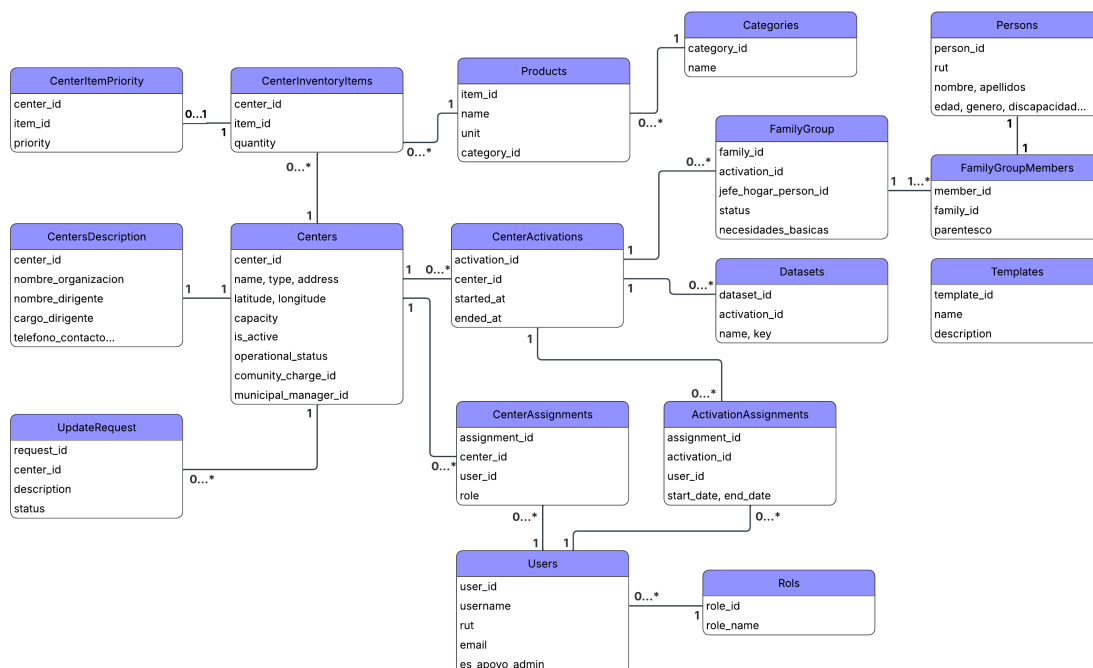


Figura 3: Modelo de datos simplificado de AppCopio. Fuente: Elaboración propia.

3.3.1. Entidades y relaciones

Se describe a nivel general el rol de cada entidad del modelo:

- **Usuarios y permisos**

- **Rols:** catálogo fijo de roles («Administrador», «Trabajador municipal», «Contacto ciudadano»).
- **Users:** usuario del sistema, con rol, credenciales y el flag `es_apoyo_admin` descrito en la Sección 3.1.1. No tiene ninguna referencia a una municipalidad o tenant.

- **Centros y activaciones**

- **Centers:** ficha maestra de cada centro, con su ubicación geográfica, tipo de centro (acopio o albergue), capacidad, estado operacional (disponible, cerrado temporalmente, entre otros) y referencias a su encargado y contacto ciudadano.
- **CentersDescription:** ficha extendida de infraestructura de cada centro (materialidad, servicios básicos, accesibilidad, seguridad, dimensión animal, etc.), en relación 1:1 con Centers.
- **CenterAssignments:** historial de asignación de personas (encargado de centro y contacto ciudadano) a un centro durante la Fase de preparación.
- **CentersActivations:** historial de activaciones de un centro; el registro vigente es aquel con `ended_at` nulo.
- **ActivationAssignments:** asignación del trabajador municipal responsable de una activación de centro específica, con un índice único que garantiza una asignación activa por usuario a la vez.

- **Solicitudes de actualización**

- **UpdateRequests:** solicitudes para actualizar CentersDescriptions reportadas desde un centro, con urgencia, asignación y resolución.

- **Personas y grupos familiares**

- **Persons:** datos de una persona natural (identificación, edad, género, situación laboral/educacional, discapacidad, dependencia); solo se crea en el contexto de un grupo familiar.
- **FamilyGroups:** grupo familiar ingresado a una activación concreta, con jefe de hogar, observaciones, estado y motivo/fecha de salida.
- **FamilyGroupMembers:** tabla puente entre FamilyGroups y Persons, con el parentesco respecto del jefe de hogar.

- **Inventario**

- **Categories y Products:** catálogo de categorías y productos gestionables por el sistema.
- **CenterInventoryItems:** stock actual de cada producto por centro.
- **InventoryLog:** historial de movimientos de inventario (agregar, restar o ajustar).
- **CenterItemPriority:** nivel de prioridad (bajo/medio/alto) de un producto para un centro determinado. Se utiliza para la priorización de necesidades en la vista pública.

- **Bases de datos dinámicas**

- **Datasets:** planillas configurables que permiten definir, dentro de una activación, estructuras de datos a medida sin modificar el esquema SQL. Cada Dataset queda amarrado a una `activation_id`.
- **Templates:** plantillas reutilizables para crear datasets con una estructura predefinida. No están asociadas a un centro específico, ya que son plantillas configurables a nivel de municipalidad.

3.4. Limitaciones frente a escenario multi-municipal

El sistema fue diseñado, desde su modelo de datos hasta su lógica de negocio, para operar como una instancia dedicada a una única municipalidad. Escalar a un escenario donde una misma instancia (o un

conjunto coordinado de instancias) deba atender a varias municipalidades simultáneamente requiere abordar los siguientes puntos:

- **Supuestos single-tenant**

- **Ausencia de discriminador de tenant/municipalidad:** ninguna tabla del esquema de la base de datos ni el payload de autenticación cuentan con un identificador de a qué tenant se pertenece, por lo que todo el modelo asume una única municipalidad implícita. Esto provoca que la información y los catálogos (productos, categorías) sean globales. Compartir una misma base de datos hoy expondría datos cruzados entre comunas.
- **Configuraciones y variables estáticas:** elementos que deberían ser configurables por organización, como el listado de zonas municipales, las coordenadas iniciales del mapa y el branding del frontend, están codificados de forma rígida en el código.
- **Ausencia de jerarquía multi-comunal:** el rol de «Administrador» actual posee privilegios totales sobre la instancia. No existe una capa de administración de plataforma para gestionar distintas municipalidades de forma segura, ni roles regionales que permitan a un usuario operar con distintos permisos en diferentes comunas sin exponer todos los datos.

- **Limitaciones funcionales**

- **Falta de eventos macro:** las emergencias y activaciones viven solo a nivel de centro. No hay una entidad de nivel superior que agrupe activaciones de múltiples centros, menos aún de múltiples municipalidades, bajo un mismo evento de emergencia.
- **Ineficiencia en escalamiento operativo:** a raíz de las limitaciones anteriores, la única vía actual para incorporar nuevas municipalidades es desplegar infraestructuras completamente independientes (bases de datos, backend, frontend y credenciales externas), es decir, un crecimiento por replicación, lo que multiplica linealmente los costos operativos y de mantenimiento.
- **Operatividad aislada:** no existen mecanismos lógicos para consolidar métricas, compartir inventario, trasladar personas o derivar notificaciones y solicitudes entre distintas municipalidades.

En conjunto, estas limitaciones muestran que el problema no es puntual, sino estructural: el modelo single-tenant recorre transversalmente el diseño de AppCopio, desde el esquema de la base de datos hasta la definición misma de los roles y permisos del sistema. Por esta razón, superarlas no es un ajuste incremental sobre la arquitectura existente, sino que exige extender el modelo de datos y la lógica de negocio del sistema hacia una arquitectura multi-tenant, capaz de alojar a múltiples municipalidades sobre una misma instancia sin comprometer el aislamiento de su información. El diseño de dicha extensión, junto con el mecanismo de colaboración inter-municipal que esta arquitectura habilita, se desarrolla en el capítulo siguiente.

Capítulo 4

Diseño de la solución

4.1. Objetivos y restricciones de diseño

El diagnóstico del capítulo anterior mostró que las limitaciones de AppCopio frente a un escenario multi-municipal recorren transversalmente su modelo de datos, sus roles y su lógica de negocio. Esta sección traduce dichas limitaciones en un conjunto acotado de requisitos funcionales y no funcionales que guían el diseño de la extensión multi-tenant y del mecanismo de colaboración inter-municipal.

Los requisitos funcionales se organizan en torno a los dos ejes que define el objetivo general del trabajo: la operación de cada municipalidad como un espacio separado dentro de una misma instancia (RF1-RF6) y la colaboración entre comunas acotada a una emergencia compartida, resguardando en todo momento los datos sensibles de cada centro y activación (RF8-RF12). A ellos se suma un requisito transversal de continuidad funcional (RF7), que se desprende de que el PMV sea una evolución sobre una plataforma ya funcional y validada en terreno, y no una reescritura completa [25].

ID	Requisito funcional
RF1	Múltiples municipalidades deben poder registrarse y operar sobre la misma instancia de la aplicación, sin necesidad de desplegar infraestructura propia.
RF2	Debe existir un rol de administración por sobre las municipalidades («Super Administrador»), capaz de crear y gestionar nuevas comunas, dado que el rol «Administrador» actual no contempla una jerarquía multi-comunal.
RF3	Cada municipalidad mantiene su propio registro de usuarios/trabajadores, administrado de forma independiente del resto.
RF4	Cada municipalidad mantiene su propio registro de centros.
RF5	El administrador de una municipalidad no ve centros de otras municipalidades salvo por colaboración explícita (RF10).
RF6	Los catálogos base del sistema (categorías de inventario, productos y plantillas) pueden seguir siendo compartidos entre todas las municipalidades, pero cada una debe poder extenderlos con ítems propios sin afectar a las demás.
RF7	Las funcionalidades ya existentes en la versión single-tenant (administración de centros, activaciones, registro FIBE, inventarios, mapa público, etc.) deben seguir operando de forma equivalente, ahora acotadas al contexto de cada municipalidad.
RF8	Debe existir una instancia mayor a la activación individual de un centro, que agrupe la participación de centros activos de la misma o de distintas municipalidades bajo un mismo evento de emergencia.
RF9	El ingreso de una municipalidad a la colaboración con otras comunas en torno a un mismo evento de emergencia debe ser explícito (invitación y aceptación), nunca automático, en línea con el principio de escalabilidad del marco legal.
RF10	Cuando dos o más municipalidades colaboran en un mismo evento de emergencia, cada participante debe poder ver un resumen agregado del estado y las necesidades priorizadas de los centros activos que las demás hayan vinculado a ese evento (colaboración inter-municipal acotada).

RF11	Debe existir un flujo simple para que una municipalidad participante declare una necesidad y otra pueda responder con una oferta de apoyo/recursos sobre esa necesidad, en el marco de esa colaboración.
RF12	Los datos sensibles de cada municipalidad –identidad e información de los trabajadores, ficha detallada de infraestructura de centros, datos de una activación tales como identidad y datos de las familias/personas alojadas, fichas FIBE, entre otros– nunca se comparten con otras municipalidades, ni siquiera cuando colaboran en un mismo evento de emergencia.

Tabla 3: Requisitos funcionales para el PMV de AppCopio multi-municipal. Fuente: Elaboración propia.

Además del alcance funcional, la naturaleza misma del PMV impone un conjunto de restricciones no funcionales. La arquitectura multi-tenant descrita en el Capítulo 2 plantea una tensión de fondo entre el nivel de aislamiento de los datos y el costo de infraestructura y mantenimiento asociado a cada estrategia [16]; esa tensión –sumada a que este es un trabajo de tesis con un tiempo de desarrollo acotado y a que AppCopio ya opera sobre un modelo de datos consolidado– es la que estructura los siguientes requisitos no funcionales.

ID	Requisito no funcional
RNF1	Aislamiento de datos: la separación entre municipalidades debe garantizarse a nivel de base de datos y no depender exclusivamente de que cada consulta del backend incluya el filtro correcto (defensa en profundidad ante errores de programación).
RNF2	Compatibilidad con el modelo de datos existente: la extensión debe ser mayormente aditiva sobre el esquema actual, evitando rediseños que obliguen a reescribir la lógica de negocio ya implementada.
RNF3	Rendimiento aceptable: el mecanismo de aislamiento no debe introducir sobrecarga significativa en las tablas de mayor volumen (inventario, historial de movimientos, registros de bases de datos dinámicas).
RNF4	Simplicidad y alcance acotado: dado que este es un trabajo de tesis con tiempo limitado, la solución debe priorizar mecanismos simples, estándar y bien soportados por el motor de base de datos por sobre soluciones a medida más complejas.
RNF5	Escalabilidad y mantenibilidad: el mecanismo de aislamiento no debe exigir duplicar infraestructura (una base de datos o un esquema por municipalidad) para mantener el costo operativo bajo a medida que se suman comunas.
RNF6	No se requiere compatibilidad retroactiva con datos productivos (detalle en la Sección 4.4).
RNF7	Trazabilidad: la información compartida entre municipalidades debe permitir identificar su origen y momento de publicación, en línea con lo que la Ley 21.364 [5] exige a los sistemas de información del SINAPRED.
RNF8	Toda excepción al modelo estándar de aislamiento (funciones con privilegios elevados u otros mecanismos de acceso directo) debe acotarse a un propósito puntual y exponer el mínimo de información necesaria, para no convertirse en una vía alterna de fuga de datos.

Tabla 4: Requisitos no funcionales para el PMV de AppCopio multi-municipal. Fuente: Elaboración propia.

4.2. Estrategias multi-tenant

En arquitecturas de software, existen tres estrategias clásicas para implementar un modelo multi-tenant sobre bases de datos relacionales [26]. Estas se diferencian en el grado de aislamiento físico y lógico de los datos de cada cliente (tenant), lo que determina las concesiones entre el costo de infraestructura, el esfuerzo operativo, la facilidad de mantenimiento y la seguridad del aislamiento [27].

En este contexto, las tres estrategias principales son:

- **Base de datos separada por tenant (database-per-tenant):** cada tenant opera en su propia base de datos física dedicada, con la misma estructura de tablas replicada de forma independiente.
- **Esquema independiente por tenant (schema-per-tenant):** todos los tenants comparten la misma base de datos física –y con ella sus recursos–, pero cada uno dispone de un esquema dedicado: un espacio de nombres lógico separado por permisos de acceso y no por límites físicos.
- **Esquema compartido con columna discriminadora (shared-schema):** todos los tenants comparten la misma instancia de la base de datos, el mismo esquema y las mismas tablas físicas. La segregación de la información se gestiona lógicamente mediante una columna identificadora (por ejemplo, tenant_id) presente en cada registro.

La Tabla 5 sintetiza las ventajas y desventajas de cada estrategia:

Estrategia	Ventajas	Desventajas
Database-per-tenant	• Máximo aislamiento: previene la exposición cruzada de datos por diseño y satisface RNF1 sin mecanismos adicionales. • Respaldos y restauraciones granulares por cliente.	• Mayor costo de infraestructura y mantenimiento, que crece linealmente con cada nueva municipalidad incorporada, incumpliendo RNF5. • Complejidad operativa para sincronizar migraciones (cada cambio de esquema debe aplicarse N veces). • Imposibilita consultas cross-tenant nativas sin diseño de una API o intermediario completamente nuevo, dificultando directamente RF8, RF9, RF10 y RF11, y en tensión con RNF4 al exigir una solución a medida.
Schema-per-tenant	• Consolida recursos físicos en un solo servidor. • Mantiene límite lógico robusto a nivel de base de datos.	• Migraciones estructurales costosas (iteración esquema por esquema), en tensión con RNF2 y RNF5. • Riesgo de desincronización de versiones entre clientes.
Shared-schema	• Máxima eficiencia en el uso de recursos computacionales. • Mantenimiento global simplificado (una sola migración), alineado con RNF5. • Facilita análisis y consultas transversales, habilitando directamente la visibilidad agregada y la entidad de emergencia compartida entre comunas (RF8, RF9, RF10 y RF11).	• Aislamiento puramente lógico: sin mecanismos adicionales depende de que cada endpoint agregue el filtro correcto, y uno solo mal escrito filtra datos de otra comuna. No cumple RNF1 por sí solo. • Posible contención de rendimiento en tablas de alto volumen, en tensión con RNF3.

Tabla 5: Análisis comparativo de estrategias multi-tenant en bases de datos relacionales.

Fuente: Elaboración propia.

Para la evolución de AppCopio a un entorno multi-municipal –donde cada comuna opera como un tenant independiente–, se requiere una arquitectura que escale a múltiples comunas con recursos acotados, y ninguna de las dos primeras estrategias lo permite: database-per-tenant –el más cercano al escenario actual de AppCopio,

que replica no solo la base de datos sino la instancia completa— por su costo creciente y por las restricciones que impone a las consultas transversales, y schema-per-tenant porque traslada ese costo a una carga operativa de migraciones con riesgo de desincronización.

Por ello, se opta por la estrategia de esquema compartido con columna discriminadora (shared-schema), combinada con la funcionalidad *Row-Level Security (RLS)* nativa de PostgreSQL [17]. Esta decisión técnica se fundamenta en tres criterios:

- **Compatibilidad con el modelo de datos existente:** permite evolucionar el Modelo Entidad-Relación preexistente sin requerir una reescritura total del esquema relacional, una migración a otro motor de base de datos o la reescritura de todos los endpoints del backend, aprovechando las capacidades nativas de la arquitectura actual (cumple RNF2 y RNF4, al apoyarse en un mecanismo nativo del motor).
- **Eficiencia de recursos:** elimina la sobrecarga operativa de aprovisionar y administrar bases de datos o esquemas independientes por cada nueva comuna incorporada (cumple RNF5).
- **Seguridad delegada al motor:** RLS traslada el filtrado desde la capa de aplicación al motor de base de datos, de modo que el aislamiento deja de depender de que cada consulta del backend recuerde filtrar (cumple RNF1).

Esta elección no está exenta de tensiones con los propios requisitos no funcionales definidos: concentrar todos los tenants en las mismas tablas físicas arriesga contención de rendimiento en las de mayor volumen (RNF3), y la colaboración inter-municipal (RF8-RF11) exige habilitar de forma controlada consultas que crucen el aislamiento estándar entre municipalidades sin incumplir RNF8. La primera se aborda en la Sección 4.3.1, al decidir dónde propagar el discriminador; la segunda, en la Sección 4.3.3 y la Sección 4.3.4.

4.3. Diseño de la extensión multi-tenant

Elegida la estrategia, el diseño se desarrolla en cuatro frentes: el modelo de datos, el nuevo rol de administración de plataforma, las políticas de aislamiento y el mecanismo de colaboración inter-municipal.

4.3.1. Rediseño del modelo de datos

La versión multi-tenant requiere extender el modelo de datos existente con nuevas entidades y relaciones, y propagar un identificador de tenant a través de las tablas con datos operativos propios de cada municipalidad. Las entidades nuevas son las siguientes:

Entidad nueva	Rol en el dominio
Municipalities	Es la entidad raíz del aislamiento. Todo centro, usuario (a excepción del Super Administrador) y dato operativo del sistema pertenece a una y solo una municipalidad.
Emergencies	Evento local de una sola comuna: agrupa las activaciones de centro de una municipalidad y es su unidad de organización interna. Corresponde al nivel «emergencia menor» de la Tabla 1.
EmergencyActivationInvitations	Registra la respuesta de cada centro a la convocatoria de una emergencia (invitada / aceptada / rechazada). Es una entidad intra-comuna: ningún dato suyo cruza de municipalidad.
SuperEvents	Contenedor de emergencias de varias comunas; único lugar donde vive la colaboración inter-municipal.
SuperEventParticipants	Relaciona municipalidades con SuperEventos mediante una máquina de estados de consentimiento (invitada / participando / rechazada / retirada).

CrossMunicipalSupportOffers	Modela que una comuna ofrezca recursos a un centro de otra comuna dentro de un SuperEvento, con un ciclo de vida propio de oferta.
-----------------------------	--

Tabla 6: Entidades nuevas del modelo multi-tenant. Fuente: Elaboración propia.

EmergencyActivationInvitations es la única de las seis que no se desprende directamente de un requisito, y por eso requiere justificación propia. Sin ella, la respuesta negativa de un centro solo se observa por su efecto (el centro no queda vinculado), indistinguible de no haberlo invitado nunca, y el administrador de la comuna no tendría cómo saber a quién le falta responder ni a quién volver a convocar. Registrar la respuesta, y no solo su efecto, hace de la convocatoria un proceso gestionable en el tiempo y no un acto irrepetible.

Un último cambio afecta al identificador de los centros, que deja de ser un correlativo global (C001, C002, ...) para codificar su pertenencia. Esa secuencia pierde sentido en un escenario multi-municipal: el primer centro de una comuna no debería ser el C047 solo porque otras ya crearon 46 centros antes. Cada municipalidad tiene ahora su propio contador y un nombre corto (SHORTNAME), tal que el identificador queda como SHORTNAME-C00X, lo que además lo hace autoexplicativo en reportes, exportaciones y comunicación entre comunas.

Con la entidad del aislamiento definida, la estrategia elegida exige incorporar una columna que identifique a qué municipalidad pertenece cada registro. En vez de añadirla a todas las entidades, el discriminador de tenencia se propaga sobre un subconjunto del modelo original. Dónde se agrega esa columna, y si es obligatoria u opcional, no es una decisión solo técnica, sino también de negocio, y se organiza en cuatro patrones:

Patrón	Entidades	Criterio	Justificación
Columna obligatoria	Centers, Users	Son las dos entidades raíz del modelo, de las que cuelgan todas las demás; al ser el origen de toda la cadena, su pertenencia no puede derivarse de nadie más.	RF3, RF4
Columna obligatoria, por desnormalización	CentersActivations, FamilyGroups, Persons, CenterInventoryItems, Datasets, InventoryLog, UpdateRequests, CenterAssignments, CenterShifts, ActivationAssignments, DatasetRecords	Entidades de alto volumen cuya pertenencia podría resolverse por navegación desde Centers pero a un costo alto en cada lectura. La comuna se fija al insertar: la escribe la aplicación con la comuna activa o, en las entidades hoja, un disparador la copia de su entidad padre o del contexto de tenant activo.	RNF2, RNF3
Columna opcional, por catálogo compartido	Categories, Products, Templates, ResourceBoxes, municipal_zones	Entidades que admiten registros globales y registros propios de cada comuna. La ausencia de valor en municipality_id significa que es un recurso compartido, visible para todas las comunas.	RF6
Sin columna propia, resuelta contra la entidad padre	Entidades restantes del esquema original, incluidas las que el modelo simplificado (Figura 3) omite.	Entidades satélite de bajo volumen, cuyo padre ya está protegido: basta exigir que ese padre sea visible para quien consulta.	RNF2

Tabla 7: Patrones de propagación del discriminador de tenencia. Fuente: Elaboración propia.

El primer patrón es el que sostiene el modelo; los demás, excepto el tercero, son consecuencia de aplicarlo sin pagar un costo desproporcionado. El segundo responde a RNF3: en entidades de alto volumen, algunas a tres o cuatro saltos de Centers, evaluar cada fila bajo política RLS con EXISTS anidado es exactamente la sobrecarga que el requisito prohíbe; al desnormalizar, el filtro pasa a ser una igualdad plana sobre columna indexada –el patrón que PostgreSQL optimiza mejor– y la pertenencia se resuelve al escribir y no en cada lectura. El cuarto es el caso inverso: donde el volumen no lo justifica, es más simple y barato apoyarse en la protección de la entidad padre. El tercer patrón, en cambio, no responde al costo sino al negocio: dejar `municipality_id` vacío marca un registro como global, y asignarlo marca un registro como propio de una comuna, de modo que los catálogos base se comparten sin duplicarse y cada comuna los extiende con ítems propios.

Quedan fuera de estos cuatro patrones dos entidades del modelo original –Roles, un catálogo fijo común a todas las comunas que no requiere discriminador, y RefreshTokens, cuyo tratamiento se aborda en la Sección 4.3.3 – y las entidades nuevas del modelo, que no necesitan propagar su pertenencia: nacen con su columna de comuna desde el diseño. Las de colaboración tienen además una particularidad: su columna de comuna identifica a una de las partes –quién originó el evento, a quién se invita, quién ofrece–, pero no delimita quién puede ver la fila. Su visibilidad se define por participación y se aborda en la Sección 4.3.4.

La Figura 4 presenta el modelo de datos resultante, con las entidades nuevas y el discriminador ya propagado:

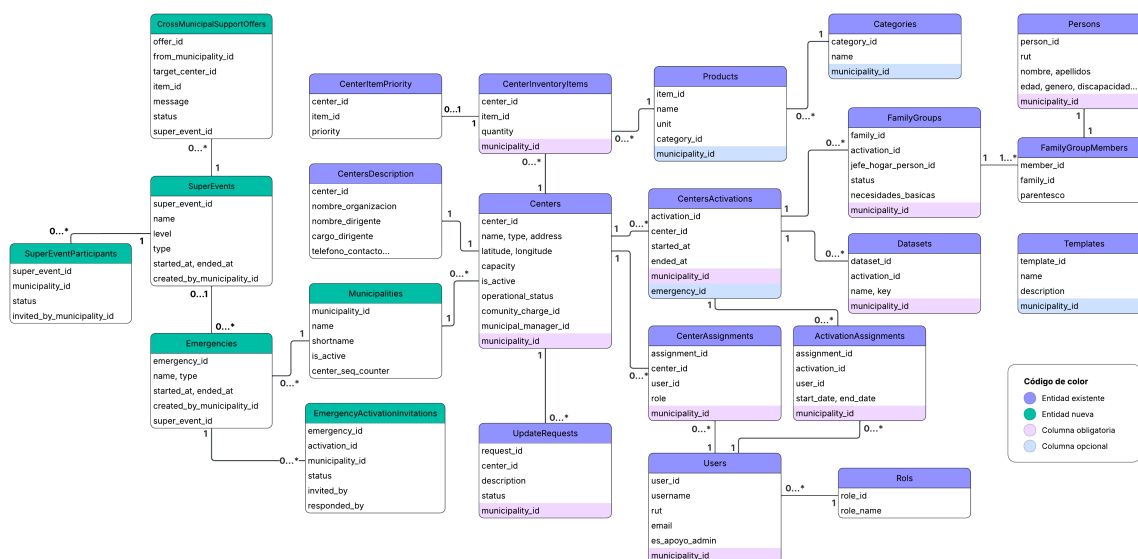


Figura 4: Modelo de datos extendido de AppCopio. Fuente: Elaboración propia.

4.3.2. Nuevo rol de administración de plataforma

Como establece RF2, el modelo multi-tenant requiere un nuevo rol de administración ubicado jerárquicamente por sobre los administradores de cada tenant. Este rol, llamado «Super Administrador», se encarga de crear nuevas municipalidades y asignar sus respectivos administradores en caso de ser necesario. A diferencia del resto de usuarios, es el único perfil que no debe tener una municipalidad asociada. Esta regla se formaliza directamente como una restricción de integridad sobre la entidad Users, y no solo como una validación de la aplicación:

```
-- El rol de Super Administrador (id 4) es la única excepción que
-- no puede ni debe tener municipalidad asociada.
CHECK ((role_id = 4 AND municipality_id IS NULL) OR
       (role_id <> 4 AND municipality_id IS NOT NULL))
```

Implementar la regla en el esquema y no solo en la aplicación es la defensa en profundidad que pide RNF1: si un error de programación asociara al Super Administrador a una municipalidad, perdería su alcance global. La

restricción cubre también el sentido inverso: ningún usuario municipal puede quedar sin comuna y, con ello, fuera de todo tenant.

Su alcance es global sobre la plataforma, pero no sobre los datos de las comunas, y esa diferencia se declara desde el diseño. El Super Administrador no debe acceder a los datos operativos de ninguna comuna –centros, activaciones, personas y grupos familiares, inventario y demás– ni siquiera en lectura. Solo debe poder intervenir en seis entidades, cada una por una razón de su rol: Users (ver, dar de alta y actualizar, nunca borrar, para nombrar y relevar a los administradores), Emergencias (ver y agrupar bajo un SuperEvento, nunca crear ni borrar), SuperEvents y SuperEventParticipants (crearlos, invitar y cerrarlos), Municipalities (darlas de alta) y municipal_zones (el catálogo geográfico de zonas). De Users debe ver la identidad del personal de cada comuna, porque nombrar un administrador exige poder elegirlo; es el único dato de personas que su rol alcanza, y no cruza hacia ninguna comuna. Estas excepciones deben declararse con políticas propias y son, junto con las funciones con privilegios elevados, el otro mecanismo de excepción que RNF8 obliga a mantener acotado, y por eso su lista debe ser explícita y verificable.

4.3.3. Diseño del aislamiento de datos y políticas de seguridad

El aislamiento de datos (RNF1) se diseña en tres eslabones, cada uno dependiente del anterior (Figura 5):

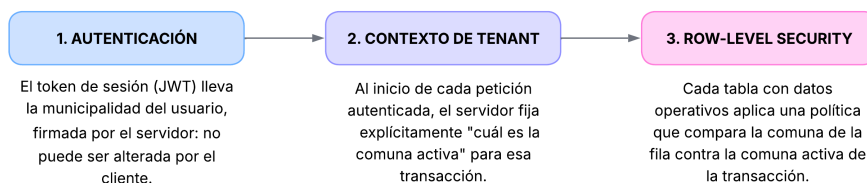


Figura 5: Cadena de tres eslabones para aislamiento de datos. Fuente: Elaboración propia.

El eslabón 3 es el que efectivamente garantiza RNF1 –el motor filtra incluso si una consulta del backend olvida el WHERE–, pero solo puede hacerlo si el eslabón 2 le informa cuál es el contexto de tenant activo, y el eslabón 2 solo puede confiar en un valor que el eslabón 1 ya verificó criptográficamente. Ningún eslabón es suficiente por sí solo.

El esquema anterior describe el caso autenticado, pero AppCopio no opera siempre bajo una municipalidad. El eslabón 2 resuelve, en realidad, uno de tres contextos posibles y mutuamente excluyentes, que la Figura 6 resume:



Figura 6: Resolución del contexto al inicio de cada petición. Fuente: Elaboración propia.

El contexto público es el único de los tres que opera sin comuna fijada, y cubre las tres superficies que AppCopio expone sin sesión: el mapa público, el formulario de voluntarios y el tablero de avisos de servicio. Sus políticas propias acotan la lectura anónima a los centros activos –de todas las comunas–, y la respuesta pública se limita a los mismos campos que el mapa ya exponía en la versión single-tenant.

Que sean excluyentes es una regla de diseño y no un detalle de implementación: en PostgreSQL las políticas permisivas de una misma tabla se combinan con una disyunción, de modo que activar el contexto público dentro de una petición autenticada sumaría la vista pública de todas las demás comunas en vez de acotarla (Anexo D.3). El contexto, por lo tanto, se decide una sola vez al inicio de cada petición y se resuelve por exclusión – hay sesión o no la hay–, nunca acumulando permisos de más de un contexto.

El diseño del canal público, en ese sentido, no es una excepción al aislamiento sino un aislamiento distinto: lo que delimita no es a qué comuna pertenece el dato, sino qué parte de él es información de interés público. Así, la regla general de las políticas se convierte en lo siguiente:

- **Para las tablas con columna de pertenencia obligatoria:** la política de acceso exige igualdad entre la comuna de la fila y la comuna activa de la sesión.
- **Para los catálogos con semántica de «compartido o propio»:** la política se relaja para admitir también las filas con pertenencia nula.

En ambos casos, la política aplica tanto a lectura como a escritura: una comuna nunca puede escribir sobre una fila que no le pertenece.

Las entidades de colaboración, sin embargo, introducen un tercer caso en el que la igualdad con la comuna activa no alcanza.

- En lectura, la visibilidad se define por participación y no por pertenencia de la fila: la oferta de apoyo debe ser visible para la comuna que la emite y la que recibe, y el SuperEvento deben verlo las comunas con una fila de participación en él, incluso si solo fue invitada.
- En escritura se mantiene la regla estricta (cada comuna escribe solo su parte del dato), con tres excepciones acotadas, todas al servicio de una respuesta de la otra comuna: invitar crea la fila de participación de la invitada, únicamente en estado *invitada*, y le deja un aviso; ofrecer apoyo deja un aviso a la destino; y la destino, al responder, modifica el estado de la oferta (creada a nombre de la emisora), así como la invitada el de su participación.

La asimetría es deliberada, pues la lectura se amplía por participación y la escritura solo en estos puntos, nunca sobre datos operativos.

Si bien RLS y las políticas mencionadas resuelven el caso general, existen operaciones legítimas que necesitan ver más allá del tenant activo, como autenticar a un usuario antes de conocer su comuna, o resolver el tablero de un centro compartido que, por definición, pertenece a otra comuna. RNF8 formaliza el principio de diseño para estas excepciones. El mecanismo concreto que materializa este principio –funciones con privilegios elevados y superficie mínima– se describe en el Capítulo 5 (Sección 5.2.1 y Sección 5.3), junto con el catálogo completo de dónde se usa cada una (Sección 5.2.5).

Una tabla del modelo original escapa por completo a este esquema: el almacén de tokens de sesión. Se consulta siempre fuera de todo contexto de tenant –inicio de sesión, refresco y cierre–, de modo que aislarla por comuna rompería el propio mecanismo de autenticación. La decisión de diseño en este caso no es aislar, sino sellar por completo, dejando como única vía de acceso el conjunto acotado de funciones que exige RNF8: una garantía más fuerte que el aislamiento por comuna, porque no depende de la comuna activa ni de si hay sesión.

Las políticas anteriores gobiernan qué puede leer y escribir cada comuna, pero dejan abierta una pregunta previa: de dónde sale la comuna de un dato que recién se crea. Como principio transversal de seguridad, la pertenencia nunca la declara quien hace la petición: se deriva siempre de una fuente que el cliente no controla, según la cadena que resume la Tabla 8.

Fuente (no controlada por el cliente)	Entidad que se crea
Sesión autenticada (JWT verificado) current_tenant()	Centro · Usuario · Persona · Producto · Categoría · Plantilla Emergencia local · Oferta de apoyo (comuna emisora)
↳ Centro al que pertenece	Activación · Inventario · Dataset Oferta de apoyo (comuna destino)
↳ Activación en que se registra	Grupo familiar
Sesión autenticada, más el estado propio de cada comuna	SuperEvento (comuna que lo origina) Participación (solo la comuna invitada cambia su propio estado)
El cuerpo de la petición nunca es fuente: ninguna entidad hereda su comuna de ahí.	

Tabla 8: Origen del discriminador de comuna al crear cada entidad. Fuente: Elaboración propia.

Sin esta regla, bastaría con alterar el cuerpo de una petición para crear datos a nombre de otra comuna; es la aplicación concreta de RNF1 al momento de escribir, no solo de leer.

4.3.4. Diseño del modelo de colaboración inter-municipal

Como se plantea en el marco teórico, este trabajo se sitúa entre los paradigmas *need-to-know* y *need-to-share* para la gestión de emergencias (Sección 2.5). El diseño de la colaboración inter-municipal es la respuesta concreta a esa tensión: define qué información deja de ser estrictamente privada y bajo qué condiciones, sin exponer los datos que RF12 declara como no compartibles. Para definir esto, es necesario considerar seis aspectos: la entidad que sostiene la colaboración, quién decide entrar en ella, qué información cruza de una comuna a otra, qué puede hacerse con esa información, hasta cuándo se mantiene la colaboración, y cómo se avisa a quien debe responder.

4.3.4.1. Dos niveles de evento

En cuanto al primer aspecto -la entidad que sostiene la colaboración-, el modelo se estructura en dos niveles de evento, como se menciona en la Tabla 6. Por un lado, la emergencia es el evento de una comuna: lo declara ella y le pertenece (exigiendo un `created_by_municipality_id` obligatorio), siendo la instancia donde administra sus propios centros activos. Por otro lado, el SuperEvento es el contenedor en que varias comunas colaboran sobre un mismo hecho: registra el nivel del evento (mayor, desastre, catástrofe) y, si lo creó una comuna y no el Super Administrador, identifica la comuna declarante.

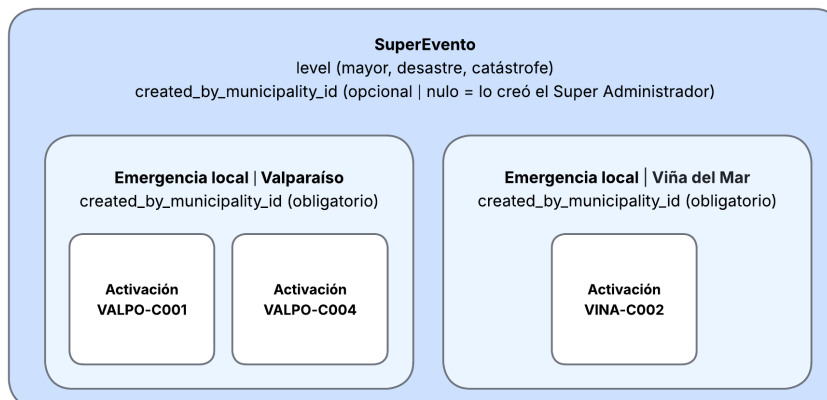


Figura 7: Los dos niveles de evento del modelo de colaboración. Fuente: Elaboración propia.

Ese desdoblamiento no estaba en el primer planteamiento: inicialmente la emergencia era la única entidad de evento y colaborar consistía en que la comuna declarante invitara a otras a su propia emergencia. Ese modelo basta para un evento que nace en una comuna y crece hacia sus vecinas, pero falla en otros tres escenarios igualmente plausibles:

- **Eventos de afectación simultánea:** un evento que afecta a varias comunas desde el inicio –un sismo, un tsunami, un sistema frontal– no tiene una comuna de origen a la cual invitar; modelarlo exigía que el Super Administrador declarara el evento antes que ellas, y su presencia en el momento mismo de la emergencia no puede darse por supuesta.
- **Detección y declaración independiente de un mismo hecho:** cuando dos comunas detectan el mismo evento de forma simultánea –un incendio en un límite comunal– cada una declara su propia emergencia antes de que exista coordinación, y el modelo no ofrecía forma de reconocer que ambas correspondían al mismo hecho: quedaban dos eventos en paralelo, sin uno canónico.
- **Dependencia de la comuna declarante:** la comuna invitada entraba como huésped en un evento ajeno, de modo que sumar una cuarta dependía siempre de la que había declarado primero, un cuello de botella para eventos de mayor magnitud.

Lo que esos tres casos tienen en común es que el modelo suponía un único actor que declara primero y coordina hacia abajo, cuando la operación real exige admitir que varias comunas declaren a la vez y la coordinación se construya después. Mantener una sola entidad habría obligado a que el contenedor existiera antes que las emergencias que agrupa: si varias comunas ya habían declarado la suya para el mismo hecho, agruparlas exigía abandonar una o trasladar su información a mano. Separar los dos niveles permite que el contenedor se cree después, sin tocar lo ya declarado. RF8 exige una instancia capaz de agrupar centros de una o de varias comunas: la emergencia local cubre el primer caso y el SuperEvento el segundo, y separarlos es lo que resuelve los tres escenarios anteriores.

Como resultado directo de esta separación arquitectónica, la progresión descrita en la Tabla 1 queda modelada por completo: la emergencia local es el nivel emergencia menor y los tres superiores son valores de `SuperEvents.level`. Cabe mencionar que el sistema modela la escala, pero no arbitra la escalada, distinción que se detalla en la Sección 4.5.

De la estructura que muestra la Figura 7 se desprenden dos invariantes que conviene fijar desde ya. La primera es que cada comuna aporta una sola emergencia local a un SuperEvento, de modo que el conjunto de centros que expone cada participante queda unívocamente definido (Sección 4.5). La segunda es que entrar en un SuperEvento no cambia de dueño ninguna emergencia: cada comuna sigue siendo dueña de lo que aporta, y el contenedor se limita a agruparlas.

La instanciación de un SuperEvento puede darse a través de dos vías:

1. Creado por el Super Administrador.
2. Originado por una comuna que invita a colaborar a otras municipalidades desde su propia emergencia.

La diferencia entre ambas vías está en quién queda participando en el momento de la creación. Si lo crea el Super Administrador, el SuperEvento nace sin comunas participantes: todas deben ser invitadas y aceptar. Si lo origina una comuna, esa municipalidad queda participando desde el inicio sin mediar invitación, y es el único caso en que esto ocurre; no constituye una excepción al consentimiento estricto (RF9), porque originar el evento es, en sí mismo, su acto de consentimiento. Para toda otra comuna, en ambas vías, el ingreso se rige por el mecanismo de invitación y aceptación que se detalla más adelante en esta sección.

Esta distinción de origen tiene una consecuencia: la gobernanza de ambos niveles se deriva de un mismo atributo, la comuna declarante, sin requerir estructura adicional (RNF2, que pide extender antes que duplicar): la emergencia local siempre tiene una; en un SuperEvento, su ausencia indica que lo creó el Super Administrador. Ese atributo determina solo quién puede cerrar o editar el SuperEvento, no quién puede invitar a él mientras esté vigente.

4.3.4.2. Ingreso a la colaboración

Definidos los dos niveles, el segundo aspecto es cómo se ingresa a cada uno. Son dos decisiones distintas, tomadas por actores distintos y en momentos distintos: que un centro activo entre a la emergencia de su comuna lo decide el encargado de su activación; que la comuna entre a un SuperEvento lo decide su administrador.

El ingreso de un centro a una emergencia local se resuelve con una invitación que el encargado de la activación acepta o rechaza. Crear la emergencia genera automáticamente una invitación a cada activación de la comuna que aún no tenga una, porque una activación pertenece a una sola emergencia a la vez (Sección 4.5); al aceptar, la activación queda vinculada. Esa convocatoria inicial no es la última: el administrador puede invitar más tarde a las activaciones que rechazaron, a las que se abrieron después o a las que están en otra emergencia y deban trasladarse. Conserva además la facultad de vincular o desvincular manualmente para casos especiales, para corregir rechazos erróneos o ante la necesidad operativa de comenzar de inmediato sin esperar respuesta, pero el diseño prioriza la validación de quién administra el recinto en terreno.

La decisión es del encargado porque quien está a cargo de la activación de un centro concreto tiene información operativa sobre la conveniencia de exponer sus necesidades que el administrador comunal no necesariamente tiene. Además, la pregunta se formula al crear la emergencia local por una razón de oportunidad: es el momento en que «¿tu centro se suma?» tiene sentido para él. Realizar la pregunta en ese momento obliga a tres resguardos:

- **La convocatoria inicial solo alcanza a las activaciones que no tienen emergencia:** convocar a las que ya están en otra las trasladaría de lugar sin que nadie lo hubiera pedido; ese traslado existe, pero es un acto explícito y separado.
- **El aviso de una invitación posterior advierte del traslado:** si la activación ya está en otra emergencia, el aviso se lo dice explícitamente al encargado, porque aceptar significa trasladarla.
- **Rechazar no desvincula:** en invitaciones posteriores, el rechazo solo se registra como tal y no altera la emergencia en la que esa activación ya estuviera legítimamente vinculada; de lo contrario, responder que no a una convocatoria nueva tendría el efecto lateral de sacarla de la anterior.

El ingreso de una comuna a un SuperEvento sigue el mismo principio –se es invitado y se decide– pero el consentimiento lo da el administrador comunal. Es el diseño central de RF9, que resume la Figura 8: ninguna comuna es incorporada a un SuperEvento por otra ni por el sistema, salvo que ella misma lo haya originado.

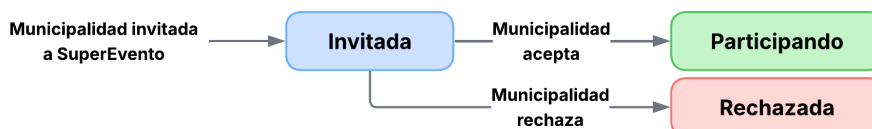


Figura 8: Modelo de ingreso de municipalidades a SuperEventos. Fuente: Elaboración propia.

Durante el ingreso, la participación de una comuna pasa por tres estados (invitada, participando, rechazada), y solo «participando» habilita la visibilidad ampliada que define RF10. La razón de fondo es que compartir información con otra organización es un acto que requiere el consentimiento explícito de quien tiene la autoridad para darlo –la administración de la comuna–, no una decisión que el sistema tome por ella.

La invitación puede emitirse de dos formas. En el caso general se invita a la comuna y su administrador elige qué emergencia aporta al aceptar. Adicionalmente, el Super Administrador puede agrupar emergencias locales que no estén asociadas a otro SuperEvento: la invitación llega entonces con el selector de emergencia precargado con la emergencia agrupada. Agrupar vincula la emergencia, pero no incorpora a su comuna: esta permanece *invitada* y conserva el control total, pudiendo aceptar (confirmando esa emergencia o seleccionando otra) o rechazar. Si rechaza, la emergencia se desvincula automáticamente y vuelve a quedar disponible para agruparse en otro evento.

Sobre esa máquina de estados, el diseño impone tres reglas:

- **Aceptar obliga a aportar una única emergencia, existente o nueva:** una comuna que participa siempre tiene la suya, de modo que el tablero nunca muestra participantes vacíos y siempre se sabe qué conjunto de centros expone cada quien.
- **Invitar puede cualquier comuna participante, no solo su comuna declarante:** es lo que corrige uno de los problemas del modelo de un solo nivel, evitando depender de la que declaró primero. Volver a invitar a una comuna que rechazó, en cambio, queda reservado al Super Administrador, para que ninguna comuna insista sobre otra que ya dijo que no. Cerrar el evento queda reservado al Super Administrador y a la comuna declarante: invitar amplía la colaboración, cerrarla se la corta a todas las demás.
- **Aceptar no vuelve a preguntar a los encargados:** exponer los centros ya vinculados a la emergencia aportada es una decisión del administrador, y la consulta al encargado no se repite en esta etapa. Por eso la interfaz debe mostrar la lista exacta de centros que quedarán compartidos antes de confirmar: es la única oportunidad del administrador de ver qué está exponiendo. Si al ser invitada la comuna todavía no tiene una emergencia local, la crea en ese momento y convoca a sus centros antes de aceptar. La emergencia local, en todo caso, se sigue gestionando después de quedar aportada: un centro puede sumarse a ella con la colaboración ya en curso, y la pregunta al encargado es siempre la misma, se responda antes o después de esa entrada.

La tercera regla abre, sin embargo, una asimetría que el diseño debe cerrar: el encargado consiente que su centro se sume a la emergencia local, y la decisión de colaborar con otras comunas la toma su administrador después. Para que ese consentimiento no quede desactualizado sin que él lo sepa, la ficha de cada uno de sus centros indica de forma permanente a qué emergencia está vinculado y, si un SuperEvento vigente lo está compartiendo, qué se expone de él y qué no. Es un estado consultable en cualquier momento, no un aviso puntual, que desaparece al cerrarse el SuperEvento.

Esa visibilidad ampliada se materializa en un tablero compartido del SuperEvento, visible para toda comuna participante: los centros que cada una aportó, con su estado operacional y sus necesidades priorizadas. Dentro de cada comuna es un tablero para el personal municipal –administrador y trabajadores, por ser quienes conocen la capacidad real de los centros que operan y pueden reconocer qué ofrecer– y no para el contacto ciudadano (Sección 3.1.1), que es un enlace con la comunidad y no personal del municipio.

4.3.4.3. Límite de confianza

El tercer y cuarto aspecto –qué información cruza y qué puede hacerse con ella– se resuelven mediante un límite de confianza, dentro del cuál operan la visibilidad ampliada y las ofertas de apoyo. RF12 exige que los datos sensibles de cada municipalidad nunca se compartan con otras comunas, incluso bajo una emergencia compartida, y la Tabla 9 define qué información puede cruzar ese límite bajo consentimiento mutuo y cuál no:

Nunca cruza de comuna	Puede cruzar con consentimiento mutuo bajo emergencia
Identidad y datos de personas y grupos familiares.	Prioridades de necesidades declaradas por un centro.
Catastro detallado de infraestructura de un centro.	Del centro: identificador, nombre, ubicación, capacidad, nivel de abastecimiento y estado operacional.
Cantidades exactas de inventario de un centro.	Las ofertas de apoyo ya enviadas, y solo entre las dos comunas involucradas en cada una.
Qué centros de una comuna rechazaron sumarse a su emergencia, y su gestión interna de la convocatoria.	Identificador y nombre corto de la comuna a la que pertenece cada centro compartido.
Ofertas de apoyo que siguen en borrador (Figura 9).	Identificador de la activación vigente del centro compartido.

	Identificador y nombre de la emergencia local que aporta cada centro al SuperEvento.
	Nombre, nivel y estado de participación de las comunas del SuperEvento.

Tabla 9: Límite de confianza entre municipalidades dentro de un SuperEvento. Fuente: Elaboración propia.

Las últimas filas de la columna derecha son identificadores y referencias de contexto, sin contenido operativo, sin los cuales el tablero sería una lista de necesidades sin dueño, sin trazabilidad (RNF7) y sin forma de dirigir una oferta a un centro específico.

En otras palabras: una comuna puede saber que el centro vecino necesita agua con prioridad alta y ofrecerle la suya, pero nunca cuántas personas alberga, quiénes son ni cuánta agua tiene exactamente. La visibilidad ampliada es, además, de solo lectura sobre los datos operativos: ninguna comuna puede modificar los datos operativos de otra, ni siquiera dentro de una emergencia compartida. Las únicas escrituras que cruzan son las descritas en la Sección 4.3.3 –invitar, ofrecer apoyo y responder una oferta–, y todas tocan solo la respuesta de la contraparte. El único contenido libre que cruza es el mensaje que la comuna oferente redacta en su propia oferta, y lo que ese mensaje contenga es decisión suya.

Sobre esa información compartida se apoya el segundo mecanismo de la colaboración. Dentro de un SuperEvento, `CrossMunicipalSupportOffers` (RF11) modela que una comuna ofrezca recursos a un centro de otra, con un ciclo de vida propio que muestra la Figura 9:

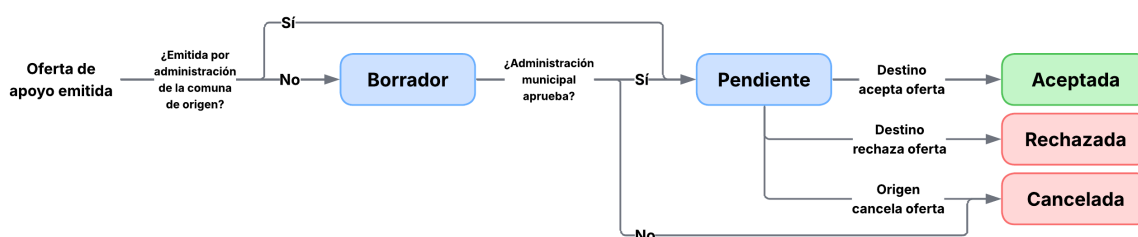


Figura 9: Ciclo de vida de una oferta de apoyo inter-municipal. Fuente: Elaboración propia.

El ciclo distingue dos momentos que conviene no confundir, porque tienen destinatarios distintos. Una oferta se emite cuando alguien del personal municipal la redacta dentro de su comuna, y se envía cuando sale de ella hacia la comuna destino; entre ambos actos puede mediar una aprobación. Si la emite un trabajador municipal, nace como *borrador*: queda a la espera de la aprobación de su propia administración y todavía no ha salido de su comuna. Solo cuando el administrador –o un trabajador con apoyo administrativo (Sección 3.1.1)– la aprueba, esta oferta se envía y pasa a *pendiente*, y recién entonces queda a la espera de la respuesta de la comuna destino. Un administrador –o un trabajador con apoyo administrativo– que redacta una oferta la envía de inmediato, sin pasar por el borrador.

Ese paso intermedio responde a una regla de autoridad y no de aislamiento: quien opera un centro es quien sabe qué puede dar, pero comprometer recursos frente a otra organización es un acto institucional que le corresponde a quien representa a la comuna. El borrador separa ambas cosas sin dejar a nadie fuera del proceso.

Sobre esa base, el resto del ciclo mantiene una asimetría deliberada: solo la comuna que ofrece puede cancelar –sea un borrador o una oferta ya enviada– y solo la que recibe puede aceptar o rechazar, de modo que ninguna decide por la contraparte. La visibilidad es igualmente acotada: una oferta enviada la ven únicamente las dos comunas involucradas, y no el resto de las participantes del evento; un borrador, en cambio, no es visible ni siquiera para la comuna destino, que no tiene por qué enterarse de una deliberación interna ajena.

Una oferta solo es posible si se cumplen tres condiciones a la vez: que el SuperEvento siga vigente, que ambas comunas –la que ofrece y la dueña del centro– estén en estado *participando*, y que el centro destino esté activo y vinculado a una emergencia aportada a ese SuperEvento. Cada una cubre un caso distinto: la primera hace que cerrar el evento tenga consecuencias reales; la segunda impide que una comuna invitada pero indecisa, o que ya se retiró, reciba u ofrezca apoyo; y la tercera evita que la colaboración alcance a centros que su propia comuna no puso en el evento. Las dos últimas condiciones se exigen también para aprobar, aceptar o rechazar una oferta ya emitida: ninguna puede resolverse si una de las dos comunas dejó de participar.

Conviene explicitar el reverso de esas condiciones: no existen ofertas de apoyo entre centros de una misma comuna, y el tablero de una comuna no muestra sus propios centros. Es el límite natural del mecanismo: el ciclo de vida de una oferta existe porque cruza una frontera de autoridad entre dos organizaciones distintas, y dentro de una comuna esa frontera no existe –el administrador ya tiene autoridad sobre todos sus centros y ve las necesidades priorizadas de cada uno–, así que reasignar recursos entre ellos es una instrucción y no una negociación.

4.3.4.4. Cierre de la colaboración

El quinto aspecto, hasta cuándo se mantiene la colaboración, tiene una respuesta directa: es, además de acotada, temporal, pues cerrar el SuperEvento revoca la visibilidad ampliada y corta las ofertas. Toda consulta que arma el conjunto compartido exige que el evento siga vigente, de modo que al cerrarlo el tablero deja de ser consultable y no se pueden emitir ofertas nuevas, sin necesidad de deshacer una por una las vinculaciones de cada centro; un borrador redactado mientras el evento estaba vigente tampoco puede enviarse después, porque la vigencia vuelve a comprobarse al momento del envío. Del mismo modo, una oferta que seguía pendiente queda congelada en ese estado: se conserva como registro, pero ninguna de las dos comunas puede ya aceptarla, rechazarla ni cancelarla. Lo único que sobrevive al cierre es el registro de qué comunas participaron y de las ofertas ya enviadas, que no es visibilidad operativa sino trazabilidad de lo ocurrido (RNF7). Las emergencias locales, en cambio, no se cierran con él: lo que termina es la colaboración, no la operación interna de cada comuna.

La colaboración también puede terminar para una sola comuna sin terminar para las demás. Una comuna participante puede retirarse del SuperEvento, de forma explícita o, automáticamente, al cerrar la emergencia local que aportaba: sin emergencia aportada, no tendría centros que exponer. Su participación pasa entonces a un cuarto estado, *retirada*, distinto de *rechazada* porque la comuna sí participó: deja de compartir sus centros y de ver el tablero, y su emergencia sale del SuperEvento. Sus ofertas aún no resueltas, emitidas o recibidas, se cancelan automáticamente, dejando registrado que la causa fue el retiro y no la decisión de una persona; las ya resueltas se conservan como historial (RNF7). Si quien se retira es la comuna declarante, pierde además la facultad de cerrar el SuperEvento, que queda solo en el Super Administrador. Como la que rechazó, una comuna retirada puede volver a ser invitada, pero solo por el Super Administrador, y debe aceptar nuevamente aportando una emergencia.

4.3.4.5. Avisos dirigidos

Todo el diseño anterior descansa en actos que alguien inicia y otro debe responder: la invitación de una comuna a un SuperEvento (RF9), la convocatoria a un centro activo para que se sume a la emergencia de su comuna, y la oferta de apoyo sobre una necesidad declarada (RF11). El último aspecto, cómo se avisa a quien debe responder, se resuelve con un canal de aviso dirigido. El diseño reutiliza aquí la entidad de notificaciones que el sistema ya tenía –pensada originalmente para avisos dentro de un centro– en lugar de introducir una nueva, y la extiende en tres sentidos:

- Admitir avisos dirigidos a una municipalidad completa, y no solo a un centro.
- Vincular cada aviso al evento que lo origina: la emergencia local cuando el aviso es interno de la comuna, el SuperEvento cuando nace de la colaboración.
- Clasificarlos por tipo.

La clasificación por tipo cumple una función concreta: los tres avisos que introduce la colaboración comparten forma y solo se distinguen por qué campos vienen llenos, de modo que sin un tipo explícito la interfaz no puede decidir a qué pantalla llevar cada uno.

De ahí se desprenden dos consecuencias de diseño. La primera es a quién llega el aviso: no a todo el personal municipal, sino a quienes pueden actuar sobre él –el encargado de la activación, en el caso de una convocatoria; el administrador de la comuna y sus trabajadores con apoyo administrativo, en el de una invitación o una oferta–; y una copia por persona, no un aviso compartido, porque el estado de lectura es por fila y un aviso único haría que el primero en abrirlo lo diera por atendido para todos. La segunda afecta a la propia entidad de notificaciones, que queda sujeta a su política de aislamiento –una comuna solo ve lo dirigido a ella o a sus propios centros–, de modo que avisarle a otra comuna que fue invitada, o que recibió una oferta, es una escritura que cruza el límite entre dos comunas. Es, por tanto, la única de las escrituras cruzadas descritas en la Sección 4.3.3 que la política de la tabla no puede autorizar, y por eso se resuelve con el mecanismo acotado que RNF8 prevé (Sección 4.3.3): funciones de propósito único que no permiten redactar el contenido ni elegir un destinatario fuera de la invitación u oferta que les da origen.

4.4. Consideraciones de migración

Según lo explicado en la Sección 3.2.2, AppCopio no está desplegado en un ambiente productivo y su entorno de desarrollo se recrea por completo en cada iteración con datos semilla, por lo que no existen datos reales que preservar: RNF6 exige explícitamente a este PMV de compatibilidad retroactiva. En una eventual puesta en producción de la versión single-tenant previa a la adopción de la extensión multi-tenant, los datos existentes deberían asociarse a una municipalidad específica; el diseño de ese procedimiento queda como trabajo futuro.

4.5. Alcance y exclusiones de diseño

Consistente con RNF4, el PMV se concentra en aislar la información de cada comuna y en habilitar su intercambio controlado durante una emergencia compartida. Quedan fuera de ese foco, como trabajo futuro, los siguientes aspectos:

- **Arbitraje normativo de la escalada:** la escala de la Tabla 1 se modela (Sección 4.3.4), pero el nivel lo declara quien crea el SuperEvento; AppCopio no aplica las implicancias normativas de cada nivel, no automatiza la escalada ni verifica que la autoridad competente lo haya decretado.
- **Relaciones de pertenencia únicas:** cada usuario pertenece a una sola municipalidad (o a ninguna, el Super Administrador), cada activación a una sola emergencia y cada comuna aporta una sola emergencia por SuperEvento. Quedan fuera, por tanto, los perfiles provinciales o regionales que operen sobre varias comunas y la participación simultánea de un centro en más de una emergencia; a cambio, el conjunto de centros que cada comuna expone en un evento queda unívocamente definido.
- **Traslados automáticos de recursos y personas:** el sistema coordina información, no ejecuta traslados. Aceptar una oferta de apoyo registra el acuerdo entre las dos comunas, pero el envío de los recursos y su descuento del stock ocurren fuera del sistema; lo mismo vale para derivar un grupo familiar a un centro de otra comuna, cuyos datos nunca cruzan de municipalidad (RF12).
- **Configuración por municipalidad:** las coordenadas iniciales del mapa y el branding siguen codificados en la aplicación; parametrizarlos por comuna es trabajo futuro.
- **Traspaso de la titularidad de un SuperEvento:** si la comuna declarante se retira, ninguna otra comuna puede cerrarlo y esa facultad queda solo en el Super Administrador; transferirla a otra comuna participante queda como trabajo futuro.

Con el diseño así delimitado –la estrategia, el modelo de datos, las políticas y el mecanismo de colaboración, junto con lo que queda explícitamente fuera–, el capítulo siguiente describe cómo se materializó sobre el código existente de AppCopio y qué problemas reveló esa implementación que el diseño no había anticipado.

Capítulo 5

Implementación de la solución

Este capítulo describe cómo se implementó sobre el código existente de AppCopio el diseño definido en el Capítulo 4: primero el entorno y los cambios de configuración, luego la extensión multi-tenant en la base de datos y el backend, el mecanismo de colaboración inter-municipal, las pantallas que el frontend requirió, y finalmente los cinco problemas que surgieron durante la implementación y que no habían sido contemplados en el diseño.

5.1. Entorno de desarrollo y configuración del proyecto

El stack tecnológico de AppCopio se mantiene según lo descrito en la Sección 3.2.1. Para esta extensión se agregaron tres elementos:

- Un rol de base de datos de aplicación, `appcopio_app`, sin privilegios de superusuario, que reemplaza al rol `postgres` que usaba el backend hasta ahora (Sección 2.3.1).
- Nuevos archivos de inicialización de la base de datos, junto a modificaciones sobre los existentes, que añaden el esquema multi-tenant y las entidades de colaboración, cierran el aislamiento sobre las tablas restantes y amplían los datos sembrados con varias municipalidades, SuperEventos, emergencias y ofertas de apoyo en distintos estados, suficientes para montar los escenarios de validación del Capítulo 6.
- Un script de verificación, `scripts/validar_multitenant.sh`, que se usa como parte de la validación de la solución del siguiente capítulo.

Como el entorno se recrea desde cero en cada inicialización, no se requirió un procedimiento de migración de datos (RNF6, Sección 4.4).

5.2. Implementación de multi-tenant

La extensión multi-tenant se implementa en cinco frentes sucesivos: cómo cada petición determina su comuna, cómo el discriminador llega a cada tabla del esquema, cómo las políticas del motor lo aplican, cómo todo ello convive con el código que ya existía, y qué excepciones al aislamiento fue necesario abrir.

5.2.1. Resolución del tenant

El sistema determina a qué municipalidad pertenece una petición HTTP a partir del token de sesión (JWT), que incorpora el identificador y el nombre corto de la municipalidad del usuario junto con sus datos habituales. Ese token se verifica criptográficamente en el servidor, por lo que el cliente no puede alterar la comuna que declara.

Sin embargo, como se definió en el Capítulo 4, hay una excepción: durante el *login*, el sistema todavía no sabe a qué comuna pertenece quien se está autenticando —es, precisamente, lo que la autenticación va a determinar—, de modo que la consulta a la tabla de usuarios no puede hacerse con un tenant ya fijado. Como las políticas de RLS comparan cada fila con el tenant de la sesión, sin ese valor ninguna fila coincide: con RLS activo sobre `Users`, un `SELECT` directo devuelve cero filas y el login falla para todo el mundo. Esto se resuelve con `auth_lookup_user`, una función con privilegios elevados y superficie mínima: recibe un nombre de usuario y lo compara por igualdad exacta, sin patrones de búsqueda ni filtros adicionales (definición completa en el Anexo B). La función equivalente, `auth_lookup_user_by_id(INT)`, cubre el mismo caso para `GET /auth/me`, que obtiene al usuario a partir del identificador que viene en el token ya verificado.

La Figura 10 muestra cómo se emite el token en el que descansa todo lo demás:

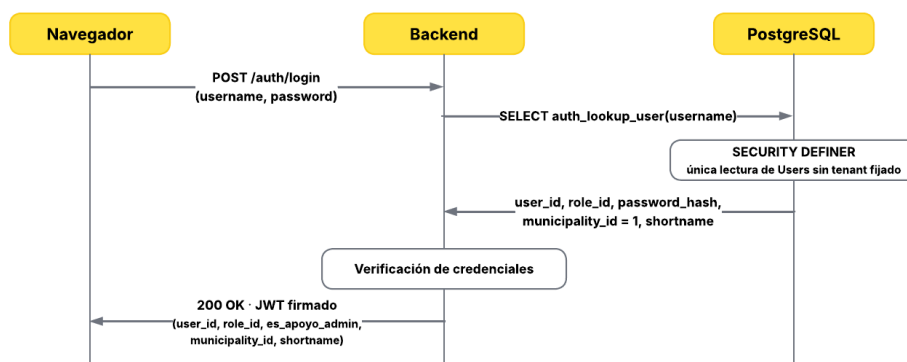


Figura 10: Emisión del token de sesión durante el *login*. Fuente: Elaboración propia.

Con el token ya emitido, cada petición autenticada sigue el recorrido de la Figura 11: un middleware resuelve la comuna, la fija en una transacción y el motor evalúa la política sobre cada fila.

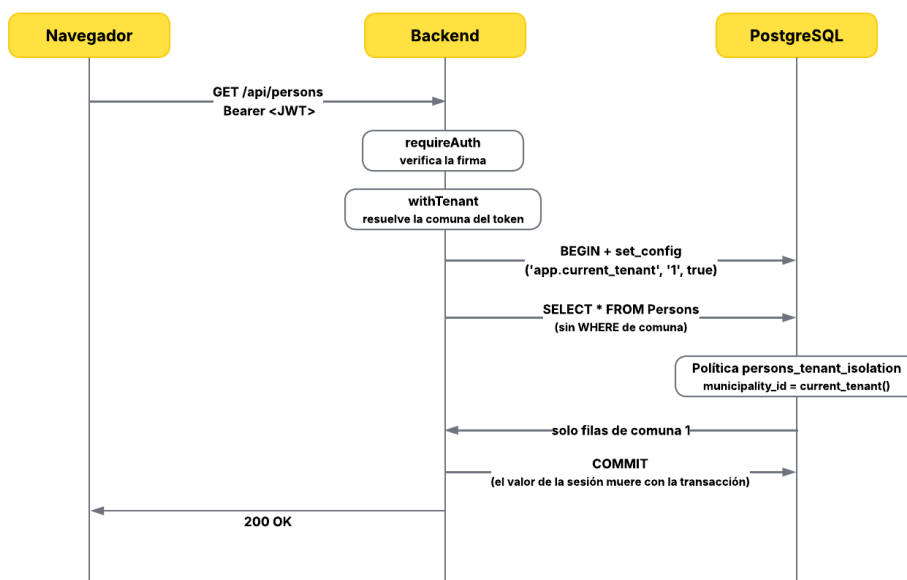


Figura 11: Ciclo de una petición autenticada, posterior al *login* de la Figura 10: de la resolución del tenant a la evaluación de la política. Fuente: Elaboración propia.

El primer paso es el middleware `withTenant`. Una vez conocida la comuna de un usuario autenticado, abre una transacción y fija en ella, mediante `set_config('app.current_tenant', ...)`, el valor de sesión que las políticas de RLS van a leer (versión simplificada del código en el Anexo B).

El detalle de por qué se usa `set_config` y no `SET LOCAL` es trivial pero determinante: el comando `SET` de PostgreSQL solo acepta literales, no parámetros, así que la forma parametrizable –la única segura frente a inyección– es la función. Su tercer argumento en `true` es lo que la hace local a la transacción, de modo que el valor desaparece al cerrarla y una conexión devuelta al *pool* no arrastra la comuna de la petición anterior.

Del otro lado, las políticas leen ese valor a través de tres funciones de contexto –`current_tenant()`, `is_superuser()` e `is_public_context()`–, una por cada uno de los contextos que el Capítulo 4 declara mutuamente excluyentes (Anexo B). El segundo argumento de `current_setting(true, missing_ok)` es lo que permite que una consulta ejecutada fuera de un contexto de tenant –como el propio login– reciba `NULL` en vez

de lanzar una excepción. Ninguna de las tres funciones es SECURITY DEFINER: solo leen una variable de sesión, así que no necesitan privilegios elevados.

A cada contexto le corresponde una variante del middleware: withTenant para las peticiones autenticadas de una comuna –y para el Super Administrador, que en vez de una comuna fija su propia marca, app.is_superadmin–, withPublicContext para las peticiones sin sesión, que fija app.public_access, y withTenantOrPublic para las rutas de doble uso –como el listado de centros, consumido tanto por el mapa público como por el panel municipal–. Esta última variante resuelve el contexto según exista o no una sesión activa, pero nunca activa ambos a la vez: hacerlo no habría acotado la vista, sino ampliado (Figura 12, Anexo D.3).

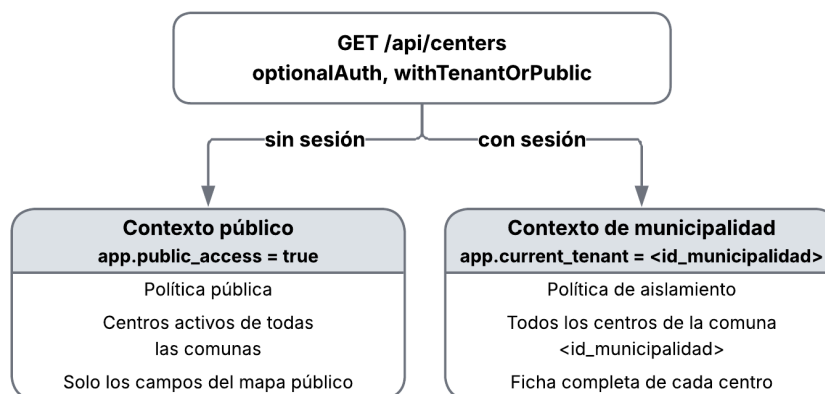


Figura 12: Ruta de doble uso: el mismo endpoint bajo dos contextos excluyentes (Figura 6).

Fuente: Elaboración propia.

El contraste entre ambas ramas es el punto: el manejador de la ruta es el mismo y su consulta también –ninguno de los dos casos agrega un WHERE por comuna–, de modo que lo único que decide qué filas devuelve el endpoint es qué política quedó activa al resolver el contexto; los campos de cada fila los fija la lista de columnas de la consulta, idéntica en ambas ramas. Es la demostración más directa de que el aislamiento no lo pone la aplicación.

5.2.2. Propagación del discriminador en el esquema

Los patrones de propagación que se definen en la Tabla 7 se materializan en el esquema de formas bastante distintas entre sí. El primero es directo –una columna NOT NULL con clave foránea a Municipalities, salvo en Users, donde la obligatoriedad la impone la restricción CHECK de la Sección 4.3.2 – y no merece más comentario. Los otros tres sí, junto con dos cambios estructurales que la extensión obligó a hacer sobre tablas que ya existían.

El primero de esos cambios estructurales afecta al identificador de centro. El Capítulo 4 decide que pase de un correlativo global a codificar su comuna (SHORTNAME-C00X), y el tipo original queda corto: center_id era VARCHAR(10), que alcanza justo para VALP0-C001 pero no deja margen para un correlativo de cuatro dígitos ni para prefijos más largos. Ampliarlo a VARCHAR(16) obligó a modificar once tablas, porque PostgreSQL exige tipos compatibles entre una clave foránea y la columna que referencia: además de Centers, las diez tablas que apuntan a ella. Es un ejemplo del costo real de una extensión aditiva (RNF2): el cambio conceptual es pequeño, pero se propaga mecánicamente por todo el grafo de referencias.

Generar ese identificador exige, además, privilegios elevados. El correlativo por comuna se almacena en la propia Municipalities, en un contador que avanza cada vez que se crea un centro. Un disparador lo resuelve al insertar (función completa en el Anexo B):

```
-- db/002a_multitenant_schema.sql
UPDATE Municipalities
  SET center_seq_counter = center_seq_counter + 1
 WHERE municipality_id = NEW.municipality_id
 RETURNING center_seq_counter, shortname INTO v_seq, v_shortcode;
```

```
NEW.center_id := v_shortcode || '-C' || LPAD(v_seq::text, 3, '0');
```

Aquí SECURITY DEFINER no es una comodidad sino un requisito: Municipalities solo admite escrituras del Super Administrador, de modo que un disparador ejecutado con los privilegios del administrador municipal que está creando el centro afectaría cero filas, el correlativo quedaría nulo y ninguna comuna podría dar de alta un centro. Es el primer caso en que una decisión de aislamiento del diseño –solo la plataforma escribe la tabla de municipalidades– obliga a una excepción acotada para no bloquear una operación legítima.

El tercer patrón –los catálogos compartidos– provocó el segundo cambio estructural. Categories.name y Products.name tenían una restricción UNIQUE global, heredada de un sistema donde solo existía una comuna; con varias, esa restricción impide que dos municipalidades usen el mismo nombre de producto, que es justamente lo que RF6 permite. La solución fue reemplazarla por dos índices únicos parciales por tabla: uno que garantiza unicidad entre los registros globales, y otro que la garantiza dentro de cada comuna.

```
-- db/002a_multitenant_schema.sql
CREATE UNIQUE INDEX categories_name_global_uq ON Categories (name)
  WHERE municipality_id IS NULL;
CREATE UNIQUE INDEX categories_name_tenant_uq ON Categories (name, municipality_id)
  WHERE municipality_id IS NOT NULL;
```

Dos comunas pueden llamar igual a sus productos propios sin colisionar, y ninguna puede duplicar un producto de catálogo base. Es la consecuencia técnica directa de la semántica dual que el capítulo anterior define para estos catálogos.

El segundo patrón se materializa por dos vías. La primera –herencia por disparador– cubre las tablas hoja de mayor volumen, que no tenían columna propia en el modelo original: ahora la reciben al insertarse, copiada de su tabla padre o, si no tiene una de la que heredar, del contexto de tenant activo. Son cuatro funciones genéricas, según de dónde herede cada tabla (de un centro, de una activación, de un *dataset* o del contexto), aplicadas mediante ocho disparadores. Siete recaen sobre tablas con columna obligatoria, incluida EmergencyActivationInvitations, que se comenta más abajo; el octavo, sobre el catálogo de cajas de recursos –del tercer patrón–, que admite registros globales y por eso puede dejarla nula. Una de las cuatro funciones se reproduce en el Anexo B.

Este disparador de herencia cumple dos funciones a la vez. La primera es evidente: propaga la comuna sin que la aplicación tenga que enviarla explícitamente en cada INSERT, que es el principio de que la pertenencia nunca la declara el cliente (Sección 4.3.3). La segunda es menos obvia y depende de un detalle de PostgreSQL: el disparador corre con los privilegios de quien invoca la sentencia, no con privilegios elevados, así que su propia subconsulta al padre pasa por RLS. Si el centro referenciado no es de la comuna activa, la subconsulta no devuelve fila, la columna queda NULL, y la restricción NOT NULL de la tabla rechaza el INSERT completo –el aislamiento de escritura se aplica solo, sin una línea de validación adicional en el backend.

El orden en que se aplican estas piezas importa: los disparadores se crean antes de imponer el NOT NULL, de modo que el sembrado de datos puede insertar sin declarar la columna y dejar que el disparador la complete.

Que estas funciones sean genéricas y no una por tabla tuvo un rédito concreto: EmergencyActivationInvitations, una tabla que no existía cuando se escribieron, se incorporó al mecanismo con una única línea –el disparador que reutiliza heredar_comuna_de_activacion()– sin escribir una sola función nueva.

Queda por precisar la segunda vía del segundo patrón, que es también la de los patrones primero y tercero: la escritura desde la aplicación. Para las entidades que sí llevan columna propia y se crean desde la aplicación, los servicios no la reciben como parámetro ni la leen del cuerpo de la petición: la escriben invocando current_tenant() dentro del propio INSERT.

```
-- backend/src/services/personService.ts
INSERT INTO Persons (rut, nombre, ..., municipality_id)
VALUES ($1, $2, ..., current_tenant())
```

El mismo patrón aparece en los servicios de categorías, productos, inventario, movimientos, plantillas y usuarios. Es una decisión deliberada: al no existir ningún parámetro de comuna en la firma, no hay forma de que un error de programación –o un cuerpo de petición manipulado– introduzca una comuna distinta de la activa. La regla de que la pertenencia nunca la declara quien hace la petición (Sección 4.3.3) queda así garantizada por la forma del código, no por la disciplina de quien lo escribe.

El cuarto patrón no necesita columna en absoluto. Las tablas restantes resuelven su pertenencia contra la tabla padre, que ya está protegida:

```
CREATE POLICY datasetfields_via_padre ON DatasetFields
  USING (dataset_id IN (SELECT d.dataset_id FROM Datasets d));
```

La subconsulta no menciona ninguna comuna, y sin embargo aísla: como `Datasets` tiene su propia política, la subconsulta solo devuelve los *datasets* que quien consulta puede ver. El aislamiento se hereda por composición, sin duplicar la condición ni la columna.

5.2.3. Políticas de aislamiento y sus excepciones

Cada tabla con datos operativos aplica una política de RLS cuya forma general compara la comuna de la fila contra la comuna activa de la transacción:

```
-- Forma general de una política de aislamiento por tenant
ALTER TABLE centers ENABLE ROW LEVEL SECURITY;
ALTER TABLE centers FORCE ROW LEVEL SECURITY;
```

```
CREATE POLICY centers_tenant_isolation ON centers
  USING (municipality_id = current_tenant())
  WITH CHECK (municipality_id = current_tenant());
```

Tres elementos de esa forma merecen comentario. El calificador `FORCE`, descrito en la Sección 2.3.1, es lo que impide que el dueño de la tabla –en PostgreSQL, habitualmente el mismo rol usado para las migraciones– se salte sus propias políticas. Junto con el rol `appcopio_app` sin `SUPERUSER` ni `BYPASSRLS` (Sección 5.1), esto es lo que hace que la política se aplique también al backend real y no solo a usuarios hipotéticos sin privilegios. La cláusula `WITH CHECK` replica la condición de lectura sobre la escritura, y es lo que impide que una comuna inserte o actualice una fila a nombre de otra. Y lo que la forma general no incluye: ninguna cláusula para el Super Administrador. Su contexto no fija comuna alguna, de modo que `current_tenant()` es nulo y la igualdad nunca se cumple: sobre esta tabla no ve nada. Las excepciones que el diseño declara (Sección 4.3.2) se escriben como políticas propias, solo en las seis tablas que su rol administra, y no como una disyunción en la regla general.

```
-- db/002a_multitenant_schema.sql: una excepción declarada, acotada a Users y sin DELETE
CREATE POLICY users_superadmin_update ON Users
  FOR UPDATE USING (is_superadmin()) WITH CHECK (is_superadmin());
```

Una tabla escapa a esta forma general: el almacén de tokens de sesión, que el diseño decide sellar en vez de aislar. La implementación de esa decisión se apoya en un comportamiento concreto de PostgreSQL: activar RLS sobre una tabla sin declarar ninguna política permisiva no la deja abierta, sino que niega todo acceso.

```
-- db/002c_rls_tablas_restantes.sql
-- Sin políticas: RLS activo y ninguna permisiva = denegación total.
ALTER TABLE RefreshTokens ENABLE ROW LEVEL SECURITY;
ALTER TABLE RefreshTokens FORCE ROW LEVEL SECURITY;
```

A partir de ahí, todo acceso al almacén pasa por siete funciones con privilegios elevados, una por operación del ciclo de vida de una sesión: emitir un token, buscarlo, revocarlo por su huella, por su identificador o en bloque

para un usuario, purgar los vencidos y reportar estadísticas de uso. Ninguna consulta SQL de la aplicación puede tocar la tabla, ni por error ni por inyección, con o sin sesión activa.

5.2.4. Contexto por transacción sobre el código existente

El mecanismo descrito hasta aquí tiene una condición incómoda: el valor de sesión que fija el middleware solo vive dentro de la transacción que lo fijó. Cualquier consulta que se ejecute sobre otra conexión –por ejemplo, porque un servicio pidió una del *pool* por su cuenta– saldría de ese contexto y, con RLS activo, no vería nada. El código existente hacía exactamente eso en muchos sitios: quince archivos con transacciones propias y más de ochenta llamadas directas al *pool*, repartidas entre *routers* y servicios.

Convertir cada uno de esos sitios en un punto consciente del tenant habría significado tocar decenas de archivos, justamente el tipo de reescritura que RNF2 buscaba evitar. En vez de eso, se interceptó el objeto *pool* en un único archivo de configuración (`backend/src/config/db.ts`) usando `AsyncLocalStorage` de Node.js: dentro de una petición con contexto de tenant activo, cualquier llamada a `pool.query()` o `pool.connect()` se enruta al mismo cliente que ya tiene el tenant fijado, y los `BEGIN/COMMIT/ROLLBACK` internos de cada servicio se vuelven no-operativos, porque la transacción real la controla el middleware. El resultado es una transacción por petición, con el tenant garantizado, sin modificar el código existente.

Un matiz que esa neutralización obliga a cuidar: un `ROLLBACK` interno que se vuelve no-operativo dejaría de revertir lo que pretendía revertir. Por eso la intercepción no lo descarta en silencio, sino que lo registra, y el middleware cierra la transacción real con `ROLLBACK` si algún servicio lo pidió o si la respuesta terminó en un código de error, con lo que se conserva el comportamiento transaccional previo.

5.2.5. Catálogo de funciones con privilegios elevados

RNF8 exige que toda excepción al aislamiento estándar sea acotada, de propósito único y auditable. Esa exigencia solo es verificable si las excepciones pueden enumerarse, así que el esquema las concentra en diecisiete funciones `SECURITY DEFINER`, agrupadas por motivo en la Tabla 10 (listado completo, con firma y propósito, en la Tabla 20 del Anexo C). Todas comparten tres propiedades: fijan explícitamente su `search_path`, tienen sus permisos de ejecución revocados de `PUBLIC` y otorgados solo al rol de aplicación, y ninguna admite patrones de búsqueda ni permite parametrizar qué comuna consultar más allá de lo que su propio propósito define.

No cruzan el límite entre comunas – 11		Tocan el límite entre comunas – 6	
Sellar el almacén de sesiones • emitir · buscar • revocar (huella / id / usuario) • purgar vencidos · estadísticas	7	Lectura acotada al conjunto compartido • super_event_shared_centers ★ • super_event_shared_center_ids ★ • super_event_participants_of • support_offers_visible	4
Autenticar antes de conocer la comuna • auth_lookup_user • auth_lookup_user_by_id	2	Aviso dirigido a la comuna de enfrente • notify_support_offer • notify_super_event_invitation	2
Operaciones internas • public_center_occupancy • generate_center_id	2		
★ Exigen además que el SuperEvento siga vigente.			
Las diecisiete: search_path fijo · EXECUTE revocado de PUBLIC · sin filtros arbitrarios			

Tabla 10: Desglose de las funciones `SECURITY DEFINER` por motivo. Fuente: Elaboración propia.

La concentración es deliberada: solo seis de las diecisiete tocan el límite entre municipalidades, y de esas, únicamente las dos que resuelven el conjunto de centros compartidos exigen que el SuperEvento siga vigente.

5.3. Implementación de la colaboración inter-municipal

Con el aislamiento ya cerrado, la colaboración inter-municipal se implementa como un conjunto acotado de aperturas deliberadas sobre él: endpoints nuevos, lecturas cruzadas resueltas con privilegios elevados y avisos dirigidos a la comuna de enfrente.

5.3.1. Endpoints y lectura cruzada acotada

La colaboración inter-municipal diseñada en el Capítulo 4 se expone mediante cuatro grupos de endpoints nuevos, veintinueve en total:

Router	Endpoints	Cubre
municipalityRoutes.ts	5	Alta y consulta de municipalidades, y cambio de administrador (Super Administrador).
emergencyRoutes.ts	10	Emergencias locales: declararlas y cerrarlas, y la gestión continua de sus centros (consultar el estado de cada activación abierta, invitar, reinvitar y vincular, individualmente o en bloque).
superEventRoutes.ts	10	SuperEventos: crearlos, agrupar bajo uno emergencias existentes –lo que deja a cada comuna dueña como invitada, no como participante–, autocrearlos desde una emergencia propia, invitar comunas, responder una invitación aportando emergencia, retirarse de uno y cerrarlos.
crossSupportRoutes.ts	4	Tablero intercomunal por SuperEvento y ciclo de vida de las ofertas de apoyo.

Tabla 11: Endpoints nuevos de colaboración inter-municipal. Fuente: Elaboración propia.

Las rutas nuevas reutilizan la regla de delegación que AppCopio ya tenía (Sección 3.1.1): tanto el middleware de autorización como el propio manejador admiten al Administrador de la comuna y a los trabajadores que él haya habilitado como apoyo administrativo. No es lógica nueva, sino la extensión del mecanismo existente a los endpoints de colaboración.

El punto técnicamente más delicado de estos routers es que varios de sus endpoints necesitan leer datos que, por diseño, pertenecen a otra comuna –el tablero intercomunal es, por definición, una lectura cruzada de tenant. Aplicar aquí el mismo mecanismo de RLS estándar los dejaría sin datos: la política de aislamiento no distingue entre «otra comuna que no debería ver esto» y «otra comuna que el SuperEvento autoriza a ver esto». La solución sigue el mismo principio que ya se usó para el login (Sección 5.2.1): funciones con privilegios elevados de propósito único y acotado (catálogo en Anexo C), tal como exige RNF8.

```
-- db/002d_supereventos.sql (forma simplificada)
CREATE OR REPLACE FUNCTION super_event_shared_centers(p_super_event_id INT)
RETURNS TABLE (
  center_id VARCHAR, name TEXT, latitude DECIMAL, longitude DECIMAL,
  capacity INT, fullness_percentage INT, operational_status TEXT,
  municipality_id INT, municipality_shortcode TEXT, activation_id INT,
  emergency_id INT, emergency_name TEXT
)
LANGUAGE sql SECURITY DEFINER SET search_path = public STABLE AS $$
-- Exige 'participando' en AMBOS lados --la comuna dueña del centro y la que lee--
-- y que el SuperEvento siga vigente (ended_at IS NULL).
-- Nunca consulta CentersDescription, FamilyGroups, Persons ni cantidades de inventario.
$$$;
```

Doce columnas, ninguna de ellas sensible: ubicación, capacidad, nivel de abastecimiento y estado operacional del centro, más los identificadores que permiten atribuir el dato a su comuna, dirigir una oferta al centro correcto, identificar su activación vigente y saber a qué emergencia local responde cada centro –necesario desde que un SuperEvento agrupa varias–. El catastro de infraestructura, los grupos familiares, las personas alojadas y las cantidades de inventario no aparecen en la firma, y por lo tanto no pueden filtrarse por un error de la capa de aplicación: no hay ninguna consulta que los pida.

La condición de participación se verifica dentro de la propia función y en ambos sentidos –la comuna dueña del centro y la que consulta deben estar participando–, de modo que una comuna invitada que todavía no ha respondido no ve nada, y tampoco es vista. A esa doble verificación se suma una tercera condición: el SuperEvento debe seguir vigente. Como las dos funciones que resuelven el conjunto de centros compartidos la exigen, cerrar el evento revoca el acceso de inmediato y en un solo punto –el tablero deja de ser consultable y no se pueden emitir ofertas nuevas– sin necesidad de recorrer y desvincular las activaciones de cada comuna.

La asimetría de permisos del ciclo de vida de una oferta de apoyo (Sección 4.3.4) se reparte entre las dos capas según lo que cada una puede garantizar:

- **Servicio:** el estado inicial de la oferta según el rol de quien la emite (pendiente si la emite un administrador o un trabajador con apoyo administrativo, borrador en los demás casos), la validación de las transiciones posteriores y el rechazo de ofertas dirigidas a centros de la propia comuna.
- **Motor (políticas RLS):** la invisibilidad del borrador para la comuna destino (política de lectura) y que ninguna comuna escriba sobre una oferta de la que no es parte (política de escritura).

El criterio es que al motor se delegan las reglas de aislamiento, y al servicio la regla de negocio que expresa para qué existe la entidad.

El retiro de una comuna es la única transición que no ejecuta una persona: sus ofertas no resueltas pasan a canceladas con un motivo (`origen_retirada` o `destino_retirada`) que las distingue de una cancelación manual, cuyo motivo queda nulo. Así el historial conserva por qué terminó cada oferta.

5.3.2. Notificaciones dirigidas

El canal de aviso que el Capítulo 4 describe se implementó extendiendo la entidad de notificaciones existente en vez de crear una nueva. El cambio central fue relajar la exigencia de que todo aviso perteneciera a un centro y admitir, en su lugar, un destino municipal, con una restricción que obliga a que exista al menos uno de los dos. A eso se suman el vínculo con el evento que origina el aviso –la emergencia local cuando el aviso es interno de la comuna, el SuperEvento cuando nace de la colaboración– y una columna de tipo, con cuatro valores posibles: invitación de una comuna a un SuperEvento, invitación a que un centro se sume a la emergencia de su comuna, ofrecimiento de apoyo y, heredado de la versión anterior, contacto de voluntariado. Antes de declarar el tipo explícitamente, la interfaz tenía que inferirlo de los campos presentes: un ofrecimiento de apoyo abría el detalle del centro en vez de la bandeja de ofertas, y el diálogo de invitación confundía una convocatoria de centro con una invitación de comuna.

El aviso a la comuna destino de una oferta es, como anticipa la Sección 4.3.4, una escritura que cruza el límite entre dos comunas, y se resuelve con una función de propósito único que recibe solo el identificador de la oferta:

```
-- db/002d_supereventos.sql --- notify_support_offer(p_offer_id INT), SECURITY DEFINER
-- La lectura de la oferta omite RLS, así que esta verificación es lo único
-- que impide avisar a nombre de otra comuna:
IF v_offer.from_municipality_id IS DISTINCT FROM current_tenant() THEN
  RAISE EXCEPTION 'SOLO_LA_COMUNA_DE_ORIGEN_PUEDE_AVISAR';
END IF;
```

Dos decisiones de esa función merecen destacarse. La primera es que no recibe destinatario ni texto: ambos se derivan de la oferta y se componen dentro, de modo que no hay forma de usarla para enviar un mensaje arbitrario

a una comuna arbitraria. La segunda es que inserta una fila por cada persona que puede actuar sobre el aviso —el administrador de la comuna destino y sus trabajadores con apoyo administrativo, es decir, quienes pueden resolverla—, por la razón expuesta en la Sección 4.3.4. Ese conjunto ya no coincide con el de quienes pueden abrir la bandeja: un trabajador municipal sin apoyo administrativo ve las ofertas de su comuna, pero no puede responderlas, y por eso no recibe el aviso.

La misma forma se repite en `notify_super_event_invitation`, su función hermana, que avisa a una comuna que fue invitada a un SuperEvento. No recibe texto ni destinatario individual, verifica que quien la invoca participe efectivamente del SuperEvento —salvo el Super Administrador, que no necesita participar—, y exige que la fila de invitación ya exista: no es un camino alternativo para invitar, solo para avisar. Que existan dos y no una es consecuencia de haber ampliado quién puede invitar: mientras solo invitaba el Super Administrador, exento del aislamiento por comuna, avisar a la comuna de enfrente nunca chocó con la política de notificaciones; al permitir que invite cualquier comuna participante, ese caso latente desde el primer diseño se activó y hubo que replicar la excepción que ya existía para las ofertas. Conviene revisar, cada vez que se amplía quién puede hacer algo, si ese algo incluye escribir en territorio ajeno.

5.4. Implementación en el frontend

El frontend asume dos responsabilidades en esta extensión: mostrar la comuna del usuario sin convertirla nunca en un dato que él declare, y dar acceso a las pantallas que la colaboración inter-municipal hizo necesarias.

5.4.1. Cómo el cliente conoce su comuna

El frontend recibe el identificador, el nombre corto y el nombre de la municipalidad del usuario junto con el resto de sus datos de sesión, y los usa exclusivamente para mostrar: el distintivo de comuna del menú, el encabezado de las pantallas de colaboración y la etiqueta de cada centro del tablero intercomunal. Lo que no hace es igual de importante: ninguna petición declara a qué comuna pertenece quien la envía; los únicos identificadores de municipalidad que viajan en el cuerpo son los de las comunas *invitadas* a un SuperEvento, que son el objeto de la operación y no la identidad de quien la pide. Es la contraparte cliente del principio del Capítulo 4: el servidor toma la comuna del token verificado y, en las escrituras, del propio motor mediante `current_tenant()` (Sección 5.2.2), de modo que aunque un cliente modificado la enviara, no habría nada del lado del servidor que la leyera.

5.4.2. Pantallas nuevas

La Tabla 12 resume las pantallas que la colaboración inter-municipal y la administración de plataforma requirieron, y que no existían en la versión single-tenant:

Pantalla	Qué resuelve
Emergencias de la comuna	Declarar y cerrar las emergencias locales, y lanzar sobre una de ellas la colaboración con otras comunas.
Gestión de centros de una emergencia (diálogo)	Ver las activaciones abiertas de la comuna con su estado frente a esa emergencia, e invitar, reinvitar o vincular directamente.
SuperEventos de la comuna	Ver los SuperEventos en que participa y con qué emergencia aporta a cada uno, invitar más comunas, retirarse de ellos y cerrar los que originó.
Aceptación de una invitación	Obliga a aportar una emergencia y muestra la vista previa de los centros que quedarán compartidos antes de confirmar.
Tablero intercomunal	Consultar los centros compartidos de un SuperEvento y sus necesidades priorizadas, y ofrecer apoyo sobre uno de ellos.

Bandeja de ofertas de apoyo	Seguir las ofertas emitidas y recibidas, con acciones distintas según el lado y el estado: aprobar y enviar, o cancelar los borradores propios, cancelar las ya enviadas, aceptar o rechazar las ajenas.
Panel de Super Administrador	Alta de municipalidades con su primer administrador y relevo de administradores; creación de SuperEventos, agrupación de emergencias existentes e invitación de comunas.

Tabla 12: Pantallas nuevas del frontend multi-tenant. Fuente: Elaboración propia.

A las pantallas nuevas se suma una heredada que la colaboración obligó a ampliar: la de los centros a cargo de cada encargado. Cada tarjeta indica ahora a qué emergencia local está vinculado el centro y, cuando un SuperEvento vigente lo está compartiendo, enuncia en palabras qué se expone de él –ubicación, capacidad, nivel de abastecimiento y necesidades priorizadas– y qué no: nunca los datos de las personas alojadas. La condición incluye que el SuperEvento siga vigente, de modo que al cerrarse el aviso desaparece; y la carga de ese dato es tolerante a fallos: si la consulta no responde, la tarjeta se muestra igual, porque es información añadida y no un bloqueo. Es la contraparte de interfaz de la transparencia hacia el encargado que exige el diseño (Sección 4.3.4), y evidencia de RF12 desde el lado de quien aporta el dato. Al ser una pantalla heredada y no una nueva, no altera el conteo de la Tabla 12, pero se incluye igualmente en la galería (Figura 22).

Las siete pantallas se reproducen en el Anexo E, en el mismo orden de la tabla. El Capítulo 6 vuelve sobre algunas de ellas, pero con un propósito distinto –acreditar un comportamiento observable, contrastando dos comunas ante la misma acción– y no para describir la interfaz.

El tablero intercomunal es la pantalla más elaborada del conjunto: ofrece dos vistas del mismo conjunto de centros –listado y mapa– ambas deliberadamente presentacionales: consumen el conjunto ya filtrado y ordenado, de modo que cambiar de vista nunca cambia el recorte, solo el eje con que se lee. Los filtros acotan por prioridad mínima, comuna, ítem solicitado y emergencia, y el orden puede ser por urgencia o por cercanía cuando el navegador concede la ubicación. Las dos vistas del tablero se contrastan en la Figura 17 y la Figura 18.

La bandeja de ofertas, por su parte, traduce a la interfaz la asimetría de permisos de la Sección 4.3.4.3, incluidos los dos momentos del ciclo: sobre una oferta propia todavía en borrador ofrece aprobarla y enviarla, o cancelarla; sobre una ya enviada, solo cancelarla; y sobre una recibida, aceptarla o rechazarla. A quien no puede aprobar –un trabajador municipal sin apoyo administrativo– no le muestra botones sobre sus propios borradores, sino la indicación de que están pendientes de aprobación. La comuna nunca ve un botón para una acción que el servidor le va a negar.

5.4.3. El cliente no es la frontera de seguridad

Las rutas de la aplicación están protegidas por guardas que verifican el rol del usuario antes de renderizar: un bloque exclusivo para el Super Administrador, y la distinción entre el personal municipal –administrador y trabajadores, que son quienes acceden al tablero intercomunal– y el resto. Sobre esa base, la marca de apoyo administrativo no amplía qué pantallas se ven, sino qué acciones se ofrecen dentro de ellas: es la que habilita resolver una oferta, no la que da acceso al tablero.

Esas guardas son una ayuda de navegación, no el control de acceso. Su función es que nadie llegue a una pantalla que no le sirve y que la interfaz no ofrezca acciones que van a fallar. La frontera real está en el motor de base de datos, en sus políticas: quien consulte la API directamente, sorteando la interfaz por completo, recibe la misma respuesta vacía o el mismo código de rechazo, porque lo que lo detiene no es el cliente sino RLS.

Esta distinción es la que hace que la solidez del aislamiento no dependa de la calidad del frontend.

5.5. Desafíos de implementación

La implementación reveló cinco problemas particularmente instructivos, porque no eran errores de sintaxis sino consecuencias no evidentes de las propias decisiones de diseño del Capítulo 4: una política de RLS que queda atrapada por su propio aislamiento, tanto al leer otra tabla protegida como al devolver la fila que acaba de escribir; la reutilización de conexiones del pool que dejaba transacciones abiertas indefinidamente; la superposición de dos contextos activos que amplía la vista en lugar de acotarla; las claves foráneas, que omiten RLS y permitían referenciar un evento que la comuna ni siquiera podía ver; y el comportamiento de `ON CONFLICT` bajo RLS. El síntoma, la causa raíz, la solución y la lección de cada uno se detallan en el Anexo D.

Los cinco desafíos comparten una raíz: el aislamiento no se comporta como una condición que se evalúa una vez en el borde del sistema, sino como una propiedad cuyo alcance real no coincide con la intuición de quien escribe la consulta. Se reaplica donde no se espera –dentro de las políticas y en cláusulas que no parecen consultarlo–, se suma donde debía restringir y se omite donde parecía garantizada, como en las comprobaciones de integridad. El capítulo siguiente somete esa propiedad a verificación empírica.

Capítulo 6

Validación de la implementación

El objetivo de este capítulo es demostrar que la arquitectura multi-tenant diseñada en el Capítulo 4 e implementada en el Capítulo 5 funciona según lo especificado y que el PMV cumple los requisitos mínimos definidos. Consistente con su carácter de Producto Mínimo Viable (Capítulo 2), la validación se aborda mediante escenarios de uso críticos –una prueba de concepto de los mecanismos centrales– y no mediante pruebas exhaustivas.

La Tabla 13 resume cómo se acredita cada requisito del Capítulo 4. Los que describen un comportamiento observable se verifican por ejecución en alguno de los cinco escenarios; los que describen una propiedad de la arquitectura –que el esquema no se duplique por comuna, que la extensión sea aditiva– se acreditan por diseño y porque el entorno completo se levanta desde cero en una sola inicialización; y RNF3 no se valida empíricamente –su argumento es de diseño, la desnormalización del discriminador (Sección 5.2.2)–, lo que se declara explícitamente en lugar de darlo por cumplido.

Modo de acreditación	Requisitos	Cantidad
Por ejecución en alguno de los cinco escenarios	RF1-RF12, RNF1, RNF5, RNF7	15
Por diseño o por esquema	RNF2, RNF4, RNF6, RNF8	4
No validado empíricamente (trabajo futuro, Capítulo 7)	RNF3	1

Tabla 13: Síntesis de la matriz de trazabilidad por modo de acreditación (detalle requisito por requisito en el Anexo F). Fuente: Elaboración propia.

Cada escenario se ejecuta contra el entorno local descrito en la Sección 3.2.2, recreado desde cero antes de la validación. Los datos de prueba incluyen las municipalidades y usuarios de la Tabla 14, además de tres SuperEventos –uno de ellos cerrado a propósito–, seis emergencias locales, invitaciones de comuna y de centro en todos sus estados, y cinco ofertas de apoyo que cubren los cinco estados de su ciclo de vida, junto con sus notificaciones. Cada pieza del sembrado existe para sostener una comprobación concreta, que el escenario correspondiente precisa cuando es necesario.

Comuna	Administrador	Trabajador con apoyo	Trabajador sin apoyo	Contacto ciudadano
Valparaíso	admin	martinalina	tito	carla.rojas
Viña del Mar	admin.vina	apoyo.vina	tm.vina	cc.vina
Quilpué	admin.quilpue	—	tm.quilpue	cc.quilpue
Concón	admin.concon	—	tm.concon	cc.concon

Tabla 14: Usuarios de prueba por comuna; además existe superadmin, sin comuna asociada. Fuente: Elaboración propia.

Para dar rigor a la evidencia sin convertir el capítulo en una sucesión de capturas de Postman, se combinan dos tipos de verificación:

- **Evidencia primaria (motor y API):** el script `validar_multitenant.sh` ejecuta contra el sistema real 87 comprobaciones agrupadas en diez secciones y reporta cada resultado. Es reproducible por cualquiera que clone el repositorio, por lo que constituye la evidencia principal de cada escenario. Las comprobaciones sin requisito asociado –disponibilidad del backend e higiene de la sesión– no se tabulan y se comentan en la síntesis de resultados.

- **Evidencia secundaria (interfaz):** una política de base de datos correcta no garantiza que la interfaz respete la misma frontera –podría ocultar datos solo visualmente–, por lo que cada escenario suma hasta dos capturas propias que muestran el mismo resultado desde la aplicación, típicamente contrastando dos comunas ante la misma acción. Se agrupan en el Anexo G, al que remite cada escenario; si la pantalla ya aparece en la galería del Anexo E, se remite a esa.

El script descansa en tres decisiones de construcción que determinan qué puede afirmarse a partir de sus resultados: las comprobaciones sobre datos sensibles se escriben como listas blancas de campos autorizados, y no como listas negras de campos prohibidos; los valores esperados se derivan de la propia base cuando existe una fuente independiente para hacerlo; y las escrituras dirigidas directamente contra la base se ejecutan dentro de una transacción que termina en ROLLBACK, con la única excepción del alta de una comuna nueva (RF1), que crea una comuna real y luego la elimina. Conviene declarar aquí un matiz: las comprobaciones de retiro, reinvitación y cierre ejecutan contra la base las mismas sentencias que el backend, sin invocar sus endpoints, por lo que acreditan lo que las políticas permiten y niegan, no que cada ruta las use.

6.1. Escenario 1: Aislamiento estricto de datos

Objetivo: Demostrar que una municipalidad no tiene acceso –ni de lectura ni de escritura– a los centros, activaciones ni datos de personas de otra municipalidad, y que esta separación se sostiene aunque se intente sortear la interfaz o se consulte la base directamente (RF3, RF4, RF5, RF12 y RNF1).

Procedimiento: Detallado paso a paso en el Anexo H.1.

Evidencia:

Verificación	Resultado esperado	Resultado obtenido	Estado
Base de datos			
Tablas con municipality_id sin RLS habilitado y forzado	0	0	PASS
SELECT COUNT(*) FROM Centers como appcopio_app, sin tenant fijado	0	0	PASS
Filas del almacén de tokens de sesión, consultado con tenant fijado	0	0	PASS
Aislamiento entre comunas			
Conteo de centros por sesión: Valparaíso vs. Viña del Mar	12 / 3	12 / 3	PASS
Lectura de un centro de otra comuna por identificador (código HTTP / contenido de la respuesta)	404 / sin-datos	404 / sin-datos	PASS
Edición de un centro de otra comuna por identificador	bloqueado	bloqueado	PASS
Personas de Valparaíso consultadas con el tenant de Viña del Mar	0	0	PASS
Cuentas de usuario comunes a ambas comunas	ninguno	ninguno	PASS

Tabla 15: Verificaciones de aislamiento, Escenario 1 (selección representativa, listado completo en el Anexo I).

Fuente: Elaboración propia.

La contraparte en la interfaz se recoge en el Anexo G: el listado «Mis Centros» de ambas comunas en sesiones simultáneas (Figura 23) y el intento de alcanzar por URL un centro ajeno desde la sesión de Viña del Mar (Figura 24).

Resultado: El aislamiento se sostiene en el motor, en la API y en la interfaz. A nivel de motor, una consulta sin contexto de tenant no devuelve fila alguna y el rol de aplicación no puede eludir RLS, por carecer de SUPERUSER y BYPASSRLS; la comprobación de que ninguna tabla con columna de pertenencia quedó sin RLS habilitado y forzado acredita cobertura completa, no solo de las tablas que se recordó proteger. A nivel de API, el acceso directo por identificador responde 404 y la edición es rechazada, confirmando aislamiento simétrico entre lectura y escritura. El almacén de tokens de sesión, por su parte, permanece sellado incluso con un tenant activo.

6.2. Escenario 2: Colaboración inter-municipal con consentimiento

Objetivo: Validar que la visibilidad ampliada entre comunas solo se habilita tras una invitación explícitamente aceptada, que lo que cruza se limita exactamente a los límites de confianza inter-municipal (Tabla 9), y que el ciclo de vida de una oferta de apoyo respeta la asimetría de permisos definida en el diseño (Sección 4.3.4; RF8 a RF12 y RNF7).

Procedimiento: Detallado paso a paso en el Anexo H.2.

Evidencia:

Verificación	Resultado esperado	Resultado obtenido	Estado
Roles			
Contacto ciudadano solicita el tablero	403	403	PASS
Trabajador sin apoyo resuelve una oferta de su comuna	403	403	PASS
SuperEventos y colaboración			
Comuna que no participa solicita el tablero	403	403	PASS
Campos del tablero fuera de los autorizados por el diseño	exacto	exacto	PASS
Campos de cada necesidad declarada fuera de ítem y prioridad	exacto	exacto	PASS
Estado de participación de la comuna invitada que aún no responde	invitada	invitada	PASS
Prioridades de esa comuna visibles en el tablero antes de aceptar	0	0	PASS
Comuna no invitada intenta inscribirse escribiendo en la base (debe rechazarlo una política de RLS)	bloqueado	bloqueado	PASS
Una comuna origina su propio SuperEvento e ingresa a él	permitido	permitido	PASS
Ofertas de Valparaíso a Viña visibles para Viña (3 enviadas, una de ellas en borrador)	2	2	PASS
Aceptar un SuperEvento sin aportar una emergencia	400	400	PASS

Aceptar aportando la emergencia de otra comuna	404	404	PASS
Aportar una segunda emergencia propia al mismo SuperEvento (debe rechazarlo el índice único parcial)	bloqueado	bloqueado	PASS
Una comuna participante inscribe a otra como participando (debe rechazarlo una política de RLS)	bloqueado	bloqueado	PASS
Una comuna participante no originaria invita a otra	permitido	permitido	PASS
Comuna ajena intenta resolver una oferta que no le corresponde (por la API)	bloqueado	bloqueado	PASS
Notificaciones			
Tipos de aviso recibidos por el administrador de la comuna destino	super_event_invitation, support_offer	super_event_invitation, support_offer	PASS
Avisos intercomunales recibidos por un trabajador sin apoyo	0	0	PASS

Tabla 16: Verificaciones de colaboración inter-municipal, Escenario 2 (selección representativa, listado completo en el Anexo I). Fuente: Elaboración propia.

En la interfaz, el Anexo G recoge una misma oferta de apoyo vista desde las dos comunas, con el estado sincronizado entre la bandeja de enviadas de Viña del Mar y la de recibidas de Valparaíso (Figura 25). El tablero intercomunal y el diálogo que exige aportar una emergencia al aceptar una invitación ya se reproducen en el Anexo E (Figura 17 y Figura 16).

Resultado: La colaboración queda acotada exactamente a lo definido en el diseño. La verificación campo por campo confirma los límites de confianza inter-municipal en su forma más exigente: no aparece ningún campo fuera del conjunto autorizado –ni los sensibles que uno vigilaría (personas, grupos familiares, inventario) ni otros–, ni en el centro compartido ni en cada necesidad declarada. La visibilidad se activa solo por consentimiento explícito –las prioridades de la comuna invitada que aún no acepta no se comparten con nadie– y este no puede autoconcederse: la inscripción directa contra la base es rechazada por RLS sin intervención de la aplicación.

Las tres reglas del modelo de participación se sostienen cada una en su nivel: aceptar sin aportar una emergencia lo rechaza el servicio (400); aportar la de otra comuna lo impide el aislamiento, pues la emergencia ajena no existe para quien pregunta –de ahí el 404 y no un 403–; y aportar una segunda propia lo bloquea el índice único parcial. La facultad de invitar, ampliada a cualquier comuna participante, y también la restricción correspondiente (solo permite dejar a la otra comuna como invitada, de modo que el consentimiento sigue siendo exclusivo de quien lo da) se verificaron escribiendo directamente contra la base. Del ciclo de vida de las ofertas se comprueba que una comuna ajena no puede resolverlas; la asimetría entre quien ofrece y quien recibe no se ejercitaron de forma directa (Sección 6.6). Por último, la guarda del rol del servidor (el contacto ciudadano no accede al tablero, y el trabajador sin apoyo accede y redacta borradores, pero no declara emergencias ni resuelve ofertas) se verificó llamando a la API directamente (Sección 5.4).

6.3. Escenario 3: Gobernanza de administración

Objetivo: Validar que el Super Administrador –pese a su nivel de privilegio sobre la plataforma– no dispone de ninguna ruta de acceso a los datos operativos de comuna alguna, ni en la aplicación ni en el motor de base de

datos (RF2, Sección 4.3.2), y que una municipalidad nueva puede darse de alta sobre la instancia en marcha sin heredar datos de ninguna otra (RF1, RNF5).

Procedimiento: Detallado paso a paso en el Anexo H.3.

Evidencia:

Verificación	Resultado esperado	Resultado obtenido	Estado
Base de datos			
Esquemas y bases de datos para las cuatro comunas (esquemas / bases)	1 / 1	1 / 1	PASS
Roles			
superadmin solicita el listado de centros	0	0	PASS
superadmin solicita el listado de municipalidades (al menos cuatro sembradas)	varias	varias	PASS
Tablas operativas y de personas, consultadas en la base como Super Administrador (filas)	0	0	PASS
Esas mismas tablas, contadas sin restricción de RLS (¿tienen datos sembrados?)	si	si	PASS
Modificar centros y personas como Super Administrador (filas alcanzadas)	0	0	PASS
Usuarios de las comunas, vistos como Super Administrador (¿tienen datos sembrados?)	si	si	PASS
Tablas cuyas políticas admiten al Super Administrador (contra la lista de seis declaradas)	coincide	coincide	PASS
Funciones que lo contemplan (contra la lista de dos declaradas)	coincide	coincide	PASS
Alta de una municipalidad nueva			
Alta de una comuna nueva sobre la instancia en marcha	creada	creada	PASS
Centros heredados por la comuna recién creada	0	0	PASS
Correlativo de identificadores de la comuna nueva	0	0	PASS
Administrador de la comuna nueva, atado solo a ella	1	1	PASS
Rastro en la base tras eliminar la comuna de prueba	0	0	PASS

Tabla 17: Verificaciones de alcance del Super Administrador y alta de municipalidades, Escenario 3. Fuente: Elaboración propia.

El panel del Super Administrador, sin ninguna opción de navegación hacia centros o datos operativos, ya se reproduce en el Anexo E (Figura 20).

Resultado: El Super Administrador administra la plataforma sin acceso a datos operativos de comuna alguna, y el límite lo impone el motor de base de datos, no la aplicación. Consultadas directamente en la base con su contexto fijado, las tablas operativas y de personas no le devuelven ninguna fila –aunque sí tienen datos sembrados, descartando un cero trivial–, y tampoco puede modificar centros ni personas. Lo que sí ve son los usuarios de las comunas, porque RF2 le exige nombrar y relevar a sus administradores. La lista blanca completa el argumento: solo seis tablas (Users, Emergencies, SuperEvents, SuperEventParticipants, Municipalities y municipal_zones) y dos funciones (la que lista a los participantes de un SuperEvento y la que avisa de una invitación) lo admiten, y si otra política agregara una cláusula para él, la comprobación fallaría. La restricción de la aplicación –el listado de centros le responde vacío– queda como segunda barrera y no como la única. CCabe precisar qué información sí queda accesible: de Users, la identidad del personal de cada comuna; y de Emergencies, sus nombres, para poder agruparlas. Son datos de administración y no los de personas alojadas ni de centros que protege RF12, pero no son agregados; reducirlos a lo estrictamente necesario queda como trabajo futuro.

El alta de una comuna nueva acredita RF1 y RNF5 por ejecución y no solo por diseño: se crea sobre la instancia en marcha, sin heredar centros y con su correlativo en cero. Además, las cuatro comunas sembradas conviven en un único esquema de una única base –el script lo comprueba contando esquemas y bases antes del alta, no lo infiere de la arquitectura–, y la nueva se crea sobre esa misma base sin aprovisionar nada adicional.

6.4. Escenario 4: Continuidad funcional y canal público

Objetivo: Validar los dos extremos que la extensión multi-tenant no debía romper: que las funcionalidades heredadas de la versión single-tenant siguen operando dentro del contexto de una comuna (RF7), que el canal público del mapa –el único que opera sin sesión y, por tanto, sin tenant– sigue mostrando únicamente la información que corresponde (RF12) y que los catálogos base siguen siendo compartidos entre comunas sin impedir que cada una los extienda con ítems propios (RF6).

Procedimiento: Detallado paso a paso en el Anexo H.4.

Evidencia:

Verificación	Resultado esperado	Resultado obtenido	Estado
Catálogos compartidos y propios			
Índices que separan el catálogo global del propio de cada comuna	4	4	PASS
Dos comunas crean una categoría con el mismo nombre	permitido	permitido	PASS
Una comuna duplica un nombre del catálogo base (debe rechazarlo el índice único global)	bloqueado	bloqueado	PASS
Continuidad funcional y canal público			
Inventario de un centro, dentro de la propia comuna	200	200	PASS
Registro de residentes de un centro, dentro de la propia comuna	200	200	PASS
Prioridades de necesidades de un centro, dentro de la propia comuna	200	200	PASS

Centros que devuelve el mapa público sin sesión, frente a los activos que registra la base (mapa / base)	9 / 9	9 / 9	PASS
Comunas representadas en el mapa público, frente a las que tienen centros activos (mapa / base)	4 / 4	4 / 4	PASS
Campos de la respuesta anónima fuera de los autorizados	exacto	exacto	PASS

Tabla 18: Verificaciones de continuidad funcional y canal público, Escenario 4. Fuente: Elaboración propia.

En la interfaz, el Anexo G recoge el mapa público sin sesión, con centros de varias comunas y la ficha reducida de uno de ellos (Figura 26), y el módulo de inventario operando con normalidad dentro de la sesión de una comuna (Figura 27).

Resultado: Los módulos heredados de Valparaíso siguen respondiendo con normalidad, ahora acotados a su municipalidad: la propagación del discriminador y las políticas de RLS no rompieron la lógica de negocio existente (RNF2). El canal público, por su parte, conserva su alcance original –centros activos de todas las comunas, con estado y necesidades priorizadas– sin convertirse en una vía alterna de acceso: expone exactamente los centros activos y, según la lista blanca, ningún campo fuera de los autorizados.

Los catálogos compartidos, por su parte, sostienen la semántica dual que RF6 define: dos comunas pueden nombrar igual a un producto propio sin colisionar, y ninguna puede duplicar un ítem del catálogo base. La comprobación se hace en una sola transacción cambiando de comuna entre ambas inserciones, que es la única forma de que la segunda se enfrente realmente a la primera; hacerlas por separado no probaría nada, porque cada una vería solo su propio catálogo.

6.5. Escenario 5: Cierre, retiro y reingreso a la colaboración

Objetivo: Validar que cerrar un SuperEvento corta efectivamente la visibilidad ampliada entre las comunas que participaban de él, que esa revocación ocurre en el motor de base de datos y no en una guarda de la aplicación, y que el retiro de una sola comuna, su reingreso y la facultad de cerrar el evento respetan las reglas de autoridad definidas en el diseño (Capítulo 4; RF9, RF10, RF11 y RNF7).

Contexto del dato sembrado: El SuperEvento 3 del entorno de prueba («Temporal agosto 2024»), en que participan Viña del Mar y Concón, está cerrado pero sus dos emergencias locales y las activaciones de sus centros siguen abiertas. Esa configuración es deliberada: es la única forma de distinguir si lo que revoca el acceso es el cierre del evento compartido o, simplemente, el término de la operación de cada comuna.

Para el retiro, la reinvitación y el cierre por una comuna, el sembrado aporta además el SuperEvento 1 («Incendio forestal región de Valparaíso 2024»; Valparaíso y Viña participando, con ofertas en los cinco estados) y el SuperEvento 2 («Aluvión Marga Marga»), originado por Viña, en que Concón rechazó y Valparaíso sigue con la invitación pendiente.

Procedimiento: Detallado paso a paso en el Anexo H.5.

Evidencia:

Verificación	Resultado esperado	Resultado obtenido	Estado
SuperEventos y colaboración			
Emergencias locales aún abiertas bajo el SuperEvento cerrado	2	2	PASS

Tablero intercomunal de un SuperEvento cerrado	409	409	PASS
Centros de Concón (SuperEvento 3 cerrado) resueltos en la base, con el tenant de Viña	0	0	PASS
Oferta de apoyo nueva sobre un SuperEvento cerrado	409	409	PASS
Retiro de Valparaíso: motivo de las ofertas sin resolver canceladas	destino_retirada origen_retirada	destino_retirada origen_retirada	PASS
Cancelaciones manuales que llevan motivo de cierre automático	0	0	PASS
Ofertas que cancela el retiro de Viña (las resueltas no se tocan)	1	1	PASS
Una comuna participante reinvita a otra que rechazó (filas afectadas)	0	0	PASS
Super Administrador reinvita a una comuna que rechazó / que se retiró	1 / 1	1 / 1	PASS
Super Administrador reinvita a una que participa / con invitación pendiente	0 / 0	0 / 0	PASS
Cierra su SuperEvento la comuna originaria, participando / ya retirada	1 / 0	1 / 0	PASS
Cierra el SuperEvento una comuna que no lo originó	0	0	PASS
Cierra el SuperEvento el Super Administrador	1	1	PASS

Tabla 19: Verificaciones de cierre, retiro y reingreso a la colaboración, Escenario 5. Fuente: Elaboración propia.

Nota. origen_retirada: la oferta se canceló porque se retiró la comuna que la envió.

destino_retirada: la oferta se canceló porque se retiró la comuna que la recibía.

En la interfaz, el efecto del cierre se recoge en el Anexo G (Figura 28), cerrando a mano el SuperEvento 1 para observar el antes y el después desde la sesión de una comuna participante. La interfaz muestra los centros de Valparaíso mientras el SuperEvento 1 está vigente; esa evidencia es de capa de interfaz y no de motor.

Resultado: El cierre del SuperEvento revoca la colaboración de forma inmediata y completa, aunque la operación local de ambas comunas sigue en marcha: sus emergencias siguen abiertas y las activaciones de sus centros, vigentes. La verificación decisiva es la tercera: la consulta no pasa por la API, sino que interroga directamente a la función que resuelve el conjunto compartido, con el contexto de tenant de una comuna participante del evento cerrado, y aun así no devuelve ni un centro. La revocación no depende, por lo tanto, de que un endpoint recuerde comprobar la vigencia del evento –vive en el motor, junto al resto del aislamiento–, que es lo que distingue cerrar el evento de desvincular una por una las activaciones de cada comuna.

El retiro de una comuna, segundo modo de terminar la colaboración, se verifica en tres frentes: cancela solo lo que está sin resolver, enviado o recibido, y deja intacto el historial; marca esas cancelaciones con un motivo que las distingue de una manual, cuyo motivo queda nulo; y, si quien se retira es la comuna originaria, le quita la facultad de cerrar el evento, que queda solo en el Super Administrador, quien siempre puede. Reinvitar a una comuna que rechazó o se retiró es exclusivo del Super Administrador: una comuna participante no logra

modificar la fila ajena (cero filas afectadas), y el reingreso por el Super Administrador no pisa a quien ya participa ni a quien tiene la invitación pendiente.

6.6. Discusión de resultados

Que las ochenta y siete comprobaciones arrojen PASS acredita algo preciso: sobre un conjunto de datos conocido y en un entorno controlado, los mecanismos de aislamiento y colaboración se comportan exactamente como los especifica el Capítulo 4. No acredita ausencia de defectos, robustez bajo condiciones no ejercitadas ni que el diseño sea el correcto para el problema. Una batería de comprobaciones verifica la conformidad de una implementación con una especificación, y aquí ambas —especificación e implementación— son obra de la misma autora: un requisito mal formulado se verifica con la misma facilidad que uno bien formulado, y el PASS no distingue entre ambos.

De ahí se sigue el inventario de lo que el instrumento estructuralmente no puede detectar. No cubre la concurrencia real: las comprobaciones son secuenciales, de modo que una condición de carrera entre dos peticiones que fijan contextos de tenant distintos sobre conexiones del mismo *pool* quedaría invisible. Es justo la clase de fallo que el mecanismo de la Sección 5.2.4 podría introducir.

La segunda limitación es de alcance: el aislamiento se verifica solo sobre las tablas y endpoints que las comprobaciones tocan, de modo que no se cubren las rutas de la aplicación que el sembrado no ejercita. La comprobación de cobertura de RLS acredita que ninguna tabla con columna de pertenencia quedó sin protección, y esa es una afirmación estructural, no una muestra. Sin embargo, la continuidad de las funcionalidades heredadas (RF7) se acredita por muestra y por revisión manual: el Escenario 4 comprueba tres de sus rutas con el script, y una exploración manual de la aplicación (recorrido de las páginas sin encontrar pantallas caídas, creación de centros y usuarios, activación de un centro con datos de prueba) no detectó fallos. Esa exploración no quedó registrada con criterios reproducibles, de modo que no equivale a una comprobación exhaustiva ni a la cobertura estructural de RLS.

Conviene, con todo, acotar esa objeción. Las reglas de operación que el diseño modela no se formularon en el escritorio: el levantamiento en terreno con la sección de Emergencia Social de la Municipalidad de Valparaíso, del que nació AppCopio, es la fuente de decisiones como el consentimiento explícito de la comuna para entrar en una colaboración o la aprobación administrativa que exige toda oferta dirigida a otra municipalidad. Lo que carece de respaldo externo es la verificación, no el planteamiento: que las reglas sean las que la operación municipal requiere lo sostiene esa contraparte; que el sistema las implemente correctamente lo acredita solo el autor. Esto acota el tipo de error que queda fuera del alcance del instrumento: uno de implementación antes que de especificación.

Esa limitación vuelve relevante la jerarquía entre los tres niveles de evidencia —motor, API e interfaz—, que no son intercambiables. La de motor es la más fuerte por ser la menos dependiente de intermediarios: cuando una consulta ejecutada como el rol de aplicación, con el contexto de tenant fijado y sin pasar por el backend, devuelve cero filas, se observa la política misma, y el resultado no cambiaría aunque toda la capa de servicios se reescribiera. La de API es más débil: una respuesta correcta puede provenir de la política o de un filtro que el servicio aplica por su cuenta, indistinguibles desde afuera; su valor está en acreditar que el sistema integrado se comporta como corresponde. La de interfaz es la más débil, porque una pantalla puede ocultar visualmente datos que el servidor sí entregó; sirve para descartar que el comportamiento correcto se pierda en el último tramo, no para establecerlo. Por eso los resultados más comprometedores del capítulo —la revocación por cierre de la Sección 6.5, la imposibilidad de autoinscribirse en un SuperEvento ajeno, el sello del almacén de sesiones— se verificaron consultando directamente la base y no la API: sobre ellos descansa la afirmación de que el aislamiento vive en el motor y no en una guarda de la aplicación.

Queda la pregunta de qué haría falta observar para afirmar que el sistema sirve en operación real. Quedan al menos cuatro aspectos que este trabajo no abordó y que sería necesario estudiar. Primero, medir el aislamiento

bajo carga concurrente con varias comunas operando a la vez, lo que permitiría completar la validación de RNF3 y pondría a prueba el límite de una conexión por petición. Segundo, hacer que personas de los equipos de emergencia social recorran el flujo completo de colaboración, para distinguir lo que el sistema permite hacer de lo que alguien logra efectivamente hacer con él en una emergencia. Tercero, someter la superficie de funciones con privilegios elevados a revisión independiente, pues es donde el aislamiento se relaja deliberadamente y donde la revisión del propio autor es menos confiable. Y cuarto, operar el sistema con datos reales durante un evento –o un simulacro– con más de una comuna, única forma de saber si el modelo de consentimiento en dos niveles resiste el ritmo de una emergencia o si sus propios usuarios terminan evitando la fricción que introduce. Las cuatro se recogen como trabajo futuro en el Capítulo 7.

6.7. Síntesis de resultados

Los cinco escenarios acreditan por ejecución los requisitos que la Tabla 13 marca como verificables de ese modo: el aislamiento de datos, la colaboración acotada por consentimiento, el cierre y el retiro de la colaboración, el alcance restringido del Super Administrador y la continuidad de las funcionalidades heredadas y del canal público. Los requisitos restantes se acreditan por diseño o por esquema –en RNF6 de forma literal, pues no existen un despliegue productivo ni datos que migrar (Sección 4.4)–, con la excepción explícita de RNF3, que no se verificó empíricamente y se retoma como trabajo futuro en el Capítulo 7.

A lo anterior se suman dos comprobaciones de higiene que no corresponden a un requisito puntual. La primera acredita la corrección del segundo desafío de implementación (Anexo D.2): tras ejecutar la batería completa no queda ninguna transacción abierta sin actividad –la fuga de conexiones que había introducido el modelo de una transacción por petición–. La segunda comprueba que no se registra ningún intento de envío de correo durante la validación.

La evidencia de ejecución del script y sus resultados se encuentran en el Anexo J.

En conjunto, esta evidencia permite afirmar que la arquitectura multi-tenant implementada cumple los requisitos mínimos del PMV (Capítulo 4), en los términos de acreditación que declara la matriz de trazabilidad.

Capítulo 7

Conclusiones

Este trabajo evolucionó AppCopio desde una plataforma single-tenant, acotada a una sola municipalidad, hacia una arquitectura multi-tenant capaz de alojar a varias comunas sobre una misma instancia, con aislamiento estricto de sus datos y un mecanismo acotado de colaboración entre ellas durante una emergencia compartida. El recorrido fue el que anuncia la introducción: diagnosticar el sistema existente, traducir ese diagnóstico en requisitos, resolver el diseño, implementarlo de forma aditiva sobre código en funcionamiento y someter el resultado a verificación. Lo que sigue recoge en qué medida ese recorrido cumplió lo que se propuso, qué aporta, qué limitaciones condicionan sus resultados y qué queda pendiente.

7.1. Cumplimiento de los objetivos

El objetivo general –diseñar una versión multi-municipal de AppCopio e implementar un PMV que permita operar espacios separados por municipalidad y habilitar la colaboración entre comunas– se cumplió, junto con los cinco objetivos específicos que lo componían.

El levantamiento del estado actual (Capítulo 3), correspondiente al objetivo específico 1, mostró que el supuesto *single-tenant* no era una limitación puntual sino una condición transversal: sin discriminador de municipalidad en ninguna tabla y sin una jerarquía por sobre el administrador comunal, la única vía de crecimiento era replicar infraestructura completa por cada comuna. Ese diagnóstico se tradujo en doce requisitos funcionales y ocho no funcionales (Tabla 3, Tabla 4), que delimitaron el alcance del PMV y sirvieron de criterio de aceptación (objetivo específico 2). El diseño (Capítulo 4), objetivo específico 3, adoptó un esquema compartido con *Row-Level Security* nativo de PostgreSQL (Tabla 5), que delega el aislamiento al motor sin duplicar infraestructura. La implementación (Capítulo 5) lo materializó sobre el código existente, dando cumplimiento al objetivo específico 4, y construyó sobre ese aislamiento el mecanismo de colaboración inter-municipal, que cumple el objetivo específico 5: consentimiento en dos niveles, máquinas de estados para la participación y las ofertas de apoyo, veintinueve endpoints nuevos (Tabla 11) y las pantallas asociadas (Tabla 12).

Finalmente, la verificación (Capítulo 6) sometió el resultado a cinco escenarios de uso críticos, respaldados por ochenta y siete comprobaciones automatizadas ejecutadas contra el sistema real y contrastadas con evidencia de interfaz. Los escenarios acreditan el aislamiento estricto entre comunas, la colaboración sin escritura sobre datos operativos ajenos, la revocación inmediata del acceso al cerrar el evento compartido, el alcance acotado del Super Administrador y la continuidad de las funcionalidades heredadas; es decir, respaldan lo construido para los objetivos 4 y 5. La única excepción es RNF3, cuyo supuesto de rendimiento no fue verificado empíricamente y se recoge como trabajo futuro, en lugar de darse por cumplido.

7.2. Aporte del trabajo

El valor de este trabajo se apoya en el contexto que motivó la introducción (Capítulo 1). Chile enfrenta una exposición estructural a desastres cuya escala, con creciente frecuencia, desborda los límites de una sola comuna, mientras la ley radica en el municipio la responsabilidad primaria de la respuesta sin haberla acompañado de un soporte tecnológico equivalente. Las iniciativas municipales existentes cubren la planificación preventiva o la participación ciudadana, pero ninguna resuelve la coordinación operativa intercomunal en tiempo real de centros de acopio y albergues. Esa es la brecha que este trabajo aborda, y lo hace con tres contribuciones principales:

- **Una arquitectura multi-tenant sobre una plataforma existente:** AppCopio evolucionó en vez de reemplazarse: se eliminó la barrera de desplegar infraestructura propia para cada comuna y no se modificó su soporte para operar con conectividad degradada –condición no negociable cuando las radiobases pueden caer durante las primeras horas de una catástrofe–, que existe en la versión *single-tenant* y que esta validación no ejercitó. Técnicamente, la combinación de esquema compartido con RLS, contexto de comuna fijado por

transacción e interceptación de la reserva de conexiones permitió incorporar el aislamiento sin reescribir las llamadas a la base de datos de la capa de servicios existente, que es la dificultad real de llevar RLS a un sistema heredado; por eso es un aporte potencialmente replicable más allá de este caso.

- **Un modelo de colaboración que respeta la autonomía de cada comuna:** nadie queda incorporado a un evento compartido sin haberlo aceptado –salvo la comuna que lo origina, para quien originarlo es el acto de consentimiento–; ningún centro se expone sin que su encargado haya sido consultado o, cuando el administrador lo vincula directamente, sin que lo vea indicado de forma permanente en la ficha de su centro; y la administración revisa la lista exacta de centros antes de aceptar. Es una respuesta concreta a la tensión entre *need-to-know* y *need-to-share* que plantea el marco teórico: una comuna puede saber que el centro vecino necesita agua con prioridad alta y ofrecerle la suya, pero nunca a cuántas personas alberga ni quiénes son.
- **Un registro de las dificultades y una verificación reproducible:** los cinco problemas que reveló la implementación (Anexo D) confirmaron que el alcance real del aislamiento no coincide con la intuición de quien escribe la consulta: la combinación disyuntiva de políticas permisivas, la omisión de RLS en las claves foráneas y las exigencias de ON CONFLICT, entre otros. Registrarlos, más que corregirlos, es parte del aporte, porque son la dificultad que enfrenta cualquiera que lleve aislamiento a nivel de motor a un sistema existente; el script de validación permite además que cualquiera reproduzca las comprobaciones.

7.3. Alcance y limitaciones

Los resultados anteriores deben leerse dentro del alcance que el propio PMV se fijó (RNF4). Algunas limitaciones se asumieron por ese alcance; otras son inseparables del mecanismo elegido y conviene declararlas como tales.

Por alcance:

- **Validación acotada:** se realizó mediante escenarios de uso críticos sobre un entorno local recreado desde cero, y no mediante pruebas exhaustivas: no hubo pruebas de carga ni de concurrencia, no se midió cobertura de código y el sistema no fue ejercitado por personas usuarias finales en un contexto operativo real. Además, especificación, implementación y verificación son obra de una misma autora (Sección 6.6).
- **Sin despliegue productivo:** no existe un despliegue con datos de ningún municipio, razón por la cual este trabajo no implementa un procedimiento de migración de datos heredados (RNF6): no hay un conjunto productivo que migrar.
- **Exclusiones de diseño:** el arbitraje normativo de la escalada, los traslados físicos de recursos y personas, las relaciones de pertenencia únicas, la configuración por municipalidad y el traspaso de la titularidad de un SuperEvento quedaron fuera deliberadamente (Sección 4.5).
- **Deuda técnica heredada:** al construirse de forma aditiva sobre el código existente (RNF2), este trabajo no evaluó ni intervino su calidad interna, cuestión que aborda de forma independiente la evaluación bajo ISO/IEC 25010 de Orellana [28] sobre la versión previa.

Inseparables del mecanismo:

- **Una conexión por petición:** fijar el contexto de comuna en una variable local a la transacción obliga a que cada petición retenga una conexión durante todo su procesamiento, lo que acota la concurrencia máxima al tamaño de la reserva con independencia de la capacidad del servidor; la alternativa –una variable no local– contaminaría la siguiente petición que reutilizara esa conexión.
- **Superficie de funciones con privilegios elevados:** las diecisiete funciones del catálogo (Anexo C) eluden por definición las políticas de aislamiento. Aunque todas están acotadas, su existencia es la contracara honesta de la solución, y mantener esa superficie acotada es una obligación permanente para quien siga desarrollando el sistema (RNF8).

7.4. Trabajo futuro

Las líneas de continuidad son de dos tipos: lo que este trabajo no consideró, y lo que surge naturalmente en la evolución de un producto de software.

7.4.1. Lo que este trabajo dejó fuera

Cada exclusión declarada en la Sección 4.5 define una línea de continuidad:

- **Escalada entre niveles de emergencia:** vincular los niveles del SuperEvento con sus implicancias normativas y con la autoridad que los decreta, idealmente de forma asistida.
- **Traslados de recursos y personas:** extender el ciclo de vida de las ofertas más allá de la aceptación, como paso previo a automatizar el traslado de inventario.
- **Relaciones de pertenencia únicas:** perfiles provinciales o regionales, y más de una emergencia por activación o por comuna dentro de un SuperEvento.
- **Configuración por municipalidad:** parametrizar el mapa y el *branding*, y guiar la incorporación de comunas nuevas.
- **Titularidad de un SuperEvento:** permitir su traspaso a otra comuna participante.

A ellas se suman los pendientes que el trabajo reconoció: medir el rendimiento bajo carga y concurrencia real (RNF3), diseñar la migración de datos heredados en caso de ser necesaria (Sección 4.4), reducir lo que el Super Administrador alcanza de Users y verificar el funcionamiento sin conexión bajo el esquema multi-tenant.

7.4.2. Evolución natural del producto

- **Verificación del sistema con usuarios y en contexto real:** pruebas de usabilidad sobre las tareas críticas, un ejercicio de mesa y un piloto controlado con una o dos comunas, con criterios de éxito definidos de antemano, para comprobar que la implementación se comporta como se especificó cuando la operan sus usuarios finales.
- **Calidad y seguridad:** someter el sistema a pruebas exhaustivas –funcionales, de aislamiento, de carga y de la capa de datos– que vayan más allá de la validación puntual de este trabajo, automatizarlas e integrarlas a una canalización de entrega continua, y realizar evaluaciones de calidad y seguridad del producto, como la ya realizada sobre la versión *single-tenant*, cuyos hallazgos esta extensión no abordó.
- **Marco legal y gobernanza:** AppCopio trata datos sensibles de personas damnificadas, por lo que queda por revisar y adaptar el sistema al marco de protección de datos personales (Ley N° 19.628 y Ley N° 21.719, aún por entrar en vigencia). Esto incluye definir qué rol legal tiene cada parte –cada municipalidad como responsable de los datos de sus centros y personas, y quien opera la plataforma como encargado que los trata por su cuenta–, la retención y eliminación de datos tras la emergencia, los convenios intermunicipales que formalicen lo que se comparte, y un modelo de sostenibilidad para cuando las comunas sean muchas.
- **Evolución funcional:** tablero en tiempo real, otros canales de aviso, informes posteriores al evento e interoperabilidad con los sistemas de información del SINAPRED.

7.5. Reflexión final

En retrospectiva, el aprendizaje central de este trabajo es que extender un sistema en funcionamiento impone una disciplina distinta a la de construir uno desde cero: cada decisión debe convivir con lo que ya existe, y las buenas decisiones son las que se propagan solas. Delegar el aislamiento al motor de base de datos, derivar la pertenencia de un dato de una fuente que el cliente no controla e interceptar el acceso a la base en un único punto son, todas, formas de conseguir que la propiedad deseada se sostenga por la forma del sistema y no por la disciplina de quien lo escribe. Ese es, más allá del PMV entregado, el criterio que este trabajo deja instalado para las iteraciones que vengan.

Referencias bibliográficas

- [1] Bündnis Entwicklung Hilft y IFHV, «WorldRiskReport 2023. Focus: Diversity». Bündnis Entwicklung Hilft, Institute for International Law of Peace and Armed Conflict (IFHV), 2023.
- [2] CNID, «Hacia un Chile resiliente frente a desastres: una oportunidad». Consejo Nacional de Innovación para el Desarrollo (CNID), 2016.
- [3] CIGIDEN, «Informe de Impacto Económico de Desastres en Chile». Centro de Investigación para la Gestión Integrada del Riesgo de Desastres (CIGIDEN), Santiago, Chile, 2023.
- [4] UNDRR y CDRI, «Hoja de Ruta para la Resiliencia de la Infraestructura en la República de Chile». Oficina de las Naciones Unidas para la Reducción del Riesgo de Desastres (UNDRR), Coalición para Infraestructuras Resilientes ante Desastres (CDRI), 2024.
- [5] Ministerio del Interior y Seguridad Pública, «Ley 21.364: Establece el Sistema Nacional de Prevención y Respuesta ante Desastres». Biblioteca del Congreso Nacional de Chile, Santiago, Chile, 2021.
- [6] P. Vega Zurita, «Capacidades institucionales y vulnerabilidad territorial frente a emergencias». Instituto de Estudios Urbanos y Territoriales Pontificia Universidad Católica de Chile, diciembre de 2021.
- [7] ACAPS, «Chile – Wildfires in the central region». Assessment Capacities Project (ACAPS), febrero de 2024.
- [8] C. Martínez *et al.*, «Informe de Daños: Evento Incendios 02 y 03 de febrero de 2024, Viña del Mar (Región de Valparaíso)». Centro de Investigación para la Gestión Integrada del Riesgo de Desastres (CIGIDEN), Santiago, Chile, febrero de 2024.
- [9] Subsecretaría de Telecomunicaciones, «Informe de estado de redes de telecomunicaciones y radiobases durante emergencias mayores», Ministerio de Transportes y Telecomunicaciones, Gobierno de Chile, Santiago, Chile, technical report, 2024.
- [10] Asociación Chilena de Municipalidades y Gobierno Regional Metropolitano, «Sistema de Gestión de Emergencias Regional (SIGER): Plataforma de análisis territorial para municipalidades». Comisión de Gestión de Riesgos y Desastres ACHM, Santiago, Chile, 2023.
- [11] CITIAPS, «Plataforma de Apoyo a la Gestión de Emergencia y Aplicaciones Ciudadanas». Centro de Innovación en Tecnologías de la Información para Aplicaciones Sociales, Universidad de Santiago de Chile (USACH), Santiago, Chile, 2022.
- [12] Ilustre Municipalidad de Valparaíso, «Plan Comunal de Emergencia Valparaíso». Valparaíso, Chile, enero de 2025.
- [13] Ministerio de Desarrollo Social y Familia, «Ficha Básica de Emergencia (FIBE)». Gobierno de Chile, Santiago, Chile, 2024.
- [14] Equipo AppCopio, «AppCopio: levantamiento de requerimientos junto a Municipalidad de Valparaíso». Valparaíso, Chile, 2025.
- [15] Microsoft Azure Architecture Center, «Architectural approaches for multi-tenant solutions». Microsoft Learn, 2022.
- [16] C.-P. Bezemer y A. Zaidman, «Multi-tenant SaaS Applications: Maintenance Dream or Nightmare?». 2010.
- [17] PostgreSQL Global Development Group, «Row Security Policies». PostgreSQL Documentation, 2024.
- [18] PostgreSQL Global Development Group, «PostgreSQL Documentation: CREATE FUNCTION». PostgreSQL Global Development Group, 2026.

- [19] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, y C. E. Youman, «Role-Based Access Control Models», *IEEE Computer*, vol. 29, n.º 2, pp. 38-47, 1996.
- [20] InterNational Committee for Information Technology Standards, «INCITS 359-2012: Information Technology — Role Based Access Control». American National Standards Institute (ANSI), Washington, D.C., Estados Unidos, 2012.
- [21] National Institute of Standards and Technology, «NIST Special Publication 800-53, Revision 5: Security and Privacy Controls for Information Systems and Organizations». U.S. Department of Commerce, Gaithersburg, Maryland, Estados Unidos, 2020.
- [22] J. R. Gil-García, «Enacting Electronic Government Success: An Integrative Architectural Framework». Springer, 2012.
- [23] D4H, «Interagency Collaboration and Information Sharing in Disaster Response». Emergency Management Resources, 2023.
- [24] E. Ries, «The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses». Crown Business, 2011.
- [25] K. H. Bennett y V. T. Rajlich, «Software Maintenance and Evolution: A Roadmap». ACM, 2000.
- [26] F. Chong, G. Carraro, y R. Wolter, «Multi-Tenant Data Architecture». [En línea]. Disponible en: <https://msdn.microsoft.com/en-us/library/aa479086.aspx>
- [27] Microsoft Azure Architecture Center, «Multi-tenant SaaS database tenancy patterns». [En línea]. Disponible en: <https://learn.microsoft.com/en-us/azure/azure-sql/database/saas-tenancy-app-design-patterns>
- [28] T. V. Orellana Barra, «Evaluación de calidad de software según el modelo ISO/IEC 25010: Caso AppCopio», Memoria para optar al título de Ingeniería Civil Informática, Valparaíso, Chile, 2026.

Anexos

Anexo A — Detalle de funcionalidades de AppCopio

Desglose de los módulos funcionales presentados en la Figura 1.

- **Gestión de centros**

- **Administración de centros:** creación y edición de la información detallada de cada centro (estado, materialidad, capacidad, instalaciones, entre otros) y asignación de encargados y contactos ciudadanos, a cargo del Administrador y Trabajadores municipales.
- **Solicitudes de centro:** sistema para notificar cambios de estado en centros, diseñado para operar durante la fase de preparación. Las solicitudes son generadas desde cada recinto por su contacto ciudadano para ser resueltas por el trabajador municipal a cargo del centro. Asegura que la información y estado del centro se encuentren actualizados antes de entrar a la fase de respuesta.

- **Activaciones de centro**

- **Ciclo de activación:** al activar un centro para una emergencia se instancia una «activación». Funciona como un contenedor temporal que aísla y traza los eventos e información asociada a ese período de emergencia (registro de residentes y afectados, turnos de encargados, entre otros).
- **Registro de damnificados:** permite rellenar digitalmente una ficha FIBE por familia damnificada, formalizando su ingreso a un centro, tarea a cargo del Trabajador municipal. Eso permite realizar un seguimiento a los grupos familiares y sus integrantes a lo largo de una activación, así como monitorear la capacidad del centro y las necesidades de inventario.
- **Gestión de turnos:** programación de personal municipal por centro y activación, historial de modificaciones y un portal de «Mis turnos» para los trabajadores.
- **Bases de datos dinámicas:** módulo de estructuras de datos configurables que permite crear tablas personalizadas por activación. Facilita el levantamiento de información ad-hoc en emergencias, dándole libertad al equipo municipal para definir estructuras de información sin alterar el esquema central de la base de datos. También permite crear plantillas reutilizables en otras activaciones.

- **Logística y abastecimiento**

- **Gestión de inventario:** catálogo centralizado de productos y categorías, a cargo de los Trabajadores municipales. Permite controlar el stock por centro, registrando el historial de movimientos y permitiendo vincular las salidas de recursos directamente a familias beneficiarias.
- **Cajas de recursos:** funcionalidad que permite pre-configurar y armar kits estandarizados de ayuda, agilizando la entrega o recepción masiva de productos en un centro.

- **Administración y herramientas transversales**

- **Gestión de usuarios y auditoría:** administración de cuentas y permisos, con un registro de los cambios de estado anterior y posterior en los datos críticos del sistema.
- **Notificaciones:** motor de alertas dirigidas a usuarios y centros específicos mediante diversos canales, registrando su estado de envío y lectura.
- **Carga masiva (CSV):** herramienta de importación de archivos para acelerar el poblamiento inicial de datos (ej. catálogo de centros).

Anexo B — Definiciones SQL de referencia

Función de autenticación de usuario que permite iniciar sesión fuera de contexto multi-tenant:

```
-- db/002a_multitenant_schema.sql
-- Única vía de lectura de Users sin tenant fijado, acotada a un username exacto.
CREATE FUNCTION auth_lookup_user(p_username TEXT)
RETURNS TABLE (user_id INT, username TEXT, password_hash TEXT, role_id INT,
                municipality_id INT, municipality_shortcode TEXT, ...)
LANGUAGE sql SECURITY DEFINER SET search_path = public STABLE AS $$
    SELECT ... FROM Users u
    LEFT JOIN Municipalities m ON m.municipality_id = u.municipality_id
    WHERE u.username = p_username;
$$;
```

Fijación del contexto de tenant en cada petición autenticada (Sección 5.2.1):

```
// backend/src/auth/tenantContext.ts (simplificado)
await client.query("BEGIN");
await client.query(
    "SELECT set_config('app.current_tenant', $1, true)",
    [String(req.user.municipality_id)]
);
// ... el resto del request se ejecuta con este cliente
```

Funciones de contexto que leen las variables de sesión fijadas por el *middleware*, una por cada uno de los tres contextos mutuamente excluyentes descritos en la Sección 5.2.1:

```
-- Contexto autenticado de una comuna
CREATE FUNCTION current_tenant() RETURNS INT AS $$
    SELECT NULLIF(current_setting('app.current_tenant', true), '')::INT;
$$ LANGUAGE sql STABLE;

-- Contexto de Super Administrador
CREATE FUNCTION is_superadmin() RETURNS BOOLEAN AS $$
    SELECT COALESCE(current_setting('app.is_superadmin', true), 'false') = 'true';
$$ LANGUAGE sql STABLE;

-- Contexto público (sin sesión)
CREATE FUNCTION is_public_context() RETURNS BOOLEAN AS $$
    SELECT COALESCE(current_setting('app.public_access', true), 'false') = 'true';
$$ LANGUAGE sql STABLE;
```

Función completa del disparador que genera el identificador de centro por comuna, presentada de forma acotada en la Sección 5.2.2:

```
-- db/002a_multitenant_schema.sql
CREATE FUNCTION generate_center_id() RETURNS TRIGGER
LANGUAGE plpgsql SECURITY DEFINER SET search_path = public AS $$
DECLARE
    v_shortcode TEXT;
    v_seq       INT;
BEGIN
    IF NEW.center_id IS NOT NULL THEN
        RETURN NEW;
    END IF;

    UPDATE Municipalities
        SET center_seq_counter = center_seq_counter + 1
```

```
WHERE municipality_id = NEW.municipality_id
RETURNING center_seq_counter, shortname INTO v_seq, v_shortcode;

IF v_seq IS NULL THEN
    RAISE EXCEPTION 'Municipality % no existe', NEW.municipality_id;
END IF;

NEW.center_id := v_shortcode || '-C' || LPAD(v_seq::text, 3, '0');
RETURN NEW;
END;
$$;

Función de herencia de comuna por disparador, una de las cuatro descritas en la Sección 5.2.2:

-- db/002c_rls_tablas_restantes.sql
CREATE FUNCTION heredar_comuna_de_centro() RETURNS TRIGGER
LANGUAGE plpgsql AS $$
BEGIN
    IF NEW.municipality_id IS NULL THEN
        NEW.municipality_id := (
            SELECT c.municipality_id FROM Centers c WHERE c.center_id = NEW.center_id
        );
    END IF;
    RETURN NEW;    -- si la subconsulta no ve el centro (RLS), queda NULL
END;              -- y el NOT NULL de la columna rechaza el INSERT
$$;
```

Anexo C — Catálogo completo de funciones con privilegios elevados

Detalle completo de las diecisiete funciones SECURITY DEFINER mencionadas en la Sección 5.2.5:

Función	Propósito y qué expone	Quién la invoca
Autenticar antes de conocer la comuna — la comuna se conoce después de leer al usuario		
auth_lookup_user	Resuelve un usuario por nombre de usuario exacto. Expone solo las columnas que el login necesita.	POST /auth/login
auth_lookup_user_by_id	Lo mismo, por identificador ya verificado en el token.	GET /auth/me, refresco de sesión
Sellar el almacén de sesiones — única vía de acceso a una tabla sellada		
refresh_token_issue, _find, _revoke_by_hash, _revoke_by_id, _revoke_all, _purge, _stats	Siete operaciones del ciclo de vida de una sesión: emitir, buscar, revocar de tres formas, purgar vencidos y reportar uso.	Login, refresco, cierre de sesión y tarea de limpieza
Operaciones internas — no cruzan comuna		
public_center_occupancy	Cuenta las personas alojadas en un centro activo. Devuelve únicamente el número: los datos de personas nunca cruzan al contexto público.	Consulta de aforo del mapa público
generate_center_id	Avanza el correlativo de la comuna y compone SHORTNAME-C00X. Escribe una tabla reservada a la plataforma.	Disparador al crear un centro
Lectura acotada al conjunto compartido — cruzan de comuna, solo para leer		
super_event_shared_centers	Tablero intercomunal: los doce campos que los límites de confianza inter-municipal autorizan, y solo si quien pregunta participa de ese SuperEvento y este sigue vigente.	GET /cross-support/board/:id
super_event_shared_center_ids	Devuelve los centros que las comunas participantes de un SuperEvento vigente comparten entre sí. Existe para no quedar atrapada por el propio aislamiento (Anexo D.1).	Política de lectura de prioridades
super_event_participants_of	Participantes de un SuperEvento, su estado y la emergencia con que cada uno aporta. La política de lectura solo deja ver la fila propia, así que una consulta directa nunca mostraría a las demás comunas. También la consulta el Super Administrador, que administra los SuperEventos.	Vista de participantes y conteo de participación
support_offers_visible	Ofertas visibles para quien consulta, con los nombres de centro, comuna y	GET /cross-support/offers

	SuperEvento ya resueltos. Un join normal dejaría sin nombre al centro ajeno.	
Aviso dirigido a la comuna de enfrente — la única escritura cruzada que requiere privilegios elevados		
notify_support_offer	Crea el aviso dirigido a la comuna destino de una oferta. Recibe solo el identificador de la oferta y compone el mensaje internamente.	POST /cross-support/offers
notify_super_event_invitation	Crea el aviso dirigido a la comuna invitada. Recibe solo el SuperEvento y la comuna, y exige que la invitación ya exista: no es una vía alterna para invitar, solo para avisar.	POST /super-events/:id/invite, /group-emergencies y /from-emergency/:id

Tabla 20: Catálogo de funciones con privilegios elevados. Fuente: Elaboración propia.

Anexo D — Desafíos de implementación

Detalle de síntoma, causa raíz, solución y lección de los cinco problemas mencionados en el Capítulo 5.

D.1 — Una política de RLS puede quedar atrapada por su propio aislamiento

Síntoma: una comuna que ya había aceptado participar en un SuperEvento seguía viendo cero prioridades de necesidades de la comuna vecina en el tablero intercomunal, a pesar de que la política de lectura ampliada estaba escrita y activa.

Causa raíz: esa política resolvía los centros compartidos mediante una subconsulta sobre las tablas de centros y activaciones, que —como cualquier otra tabla del esquema— también tienen RLS activo. Al ejecutarse dentro de otra política, la subconsulta seguía sujeta al aislamiento por tenant y solo veía los centros de la propia comuna: la política que debía habilitar la visibilidad cruzada terminaba filtrándose a sí misma. La misma causa raíz reapareció, de forma independiente, en el conteo de comunas participantes y en la resolución del nombre del centro destino de una oferta.

Solución: extraer la lógica de lectura cruzada a una función SECURITY DEFINER acotada (super_event_shared_center_ids para el caso de las prioridades, super_event_shared_centers para el tablero; Sección 5.3), que se ejecuta con privilegios elevados y por lo tanto no queda sujeta a la política que intenta resolver.

Una tercera manifestación, dentro de una escritura: la sentencia que crea un SuperEvento desde la propia emergencia de una comuna devuelve la fila recién insertada, y esa cláusula obliga al motor a evaluar sobre ella la política de lectura. En ese instante la comuna todavía no figura como participante —su fila se inserta un paso después, en la misma transacción—, de modo que la política no reconocía como propia una fila que acababa de crear y la operación fallaba por permisos. No había nada que cruzar de comuna, así que no se usó una función de privilegios elevados: bastó completar la política reconociendo como lectora legítima a la comuna que originó el evento.

Lección: RLS se aplica también dentro de las políticas que se ejecutan sobre otras tablas protegidas, y devolver la fila recién escrita convierte una escritura en una lectura sujeta a política, evaluada antes de que exista el contexto que la autoriza. La respuesta correcta depende de qué falta: si la consulta necesita legítimamente ver más allá del tenant activo, el camino es una función de excepción acotada; si la fila es de la propia comuna y la política aún no la reconoce, corresponde completar la política con el acotamiento más estrecho que resuelva el caso y ni un grado más.

D.2 — La reutilización de conexiones del pool dejaba transacciones abiertas para siempre

Síntoma: bajo uso sostenido, el número de transacciones inactivas en la base de datos crecía sin límite hasta que el proceso del backend caía.

Causa raíz: el mecanismo que evita cerrar dos veces la misma transacción (Sección 5.2.4) identificaba cada conexión por el objeto `PoolClient` que Node.js le entregaba. El *pool* de PostgreSQL, sin embargo, recicla esos objetos: al liberarse, un cliente vuelve a estar disponible y una petición distinta puede recibir el mismo objeto, ya marcado como «cerrado» por la petición anterior, deteniendo la lógica de cierre antes de tiempo y dejando la transacción real abierta indefinidamente. El efecto secundario fue más grave que la fuga misma: PostgreSQL termina matando esas conexiones ociosas por tiempo límite, y el cliente de Node.js emitía un evento de error sin manejador, tratado por el runtime como una excepción no capturada.

Solución: identificar cada transacción por un valor único de la propia petición HTTP en curso, en vez de por el objeto de conexión reciclable, y agregar un manejador explícito para el evento de error de cada cliente.

Lección: interceptar un objeto compartido (como el pool de conexiones) para propagar contexto sin refactorizar el resto del código (Sección 5.2.4) exige entender su ciclo de vida completo; asumir que un objeto de conexión es único por petición fue un supuesto incorrecto y caro.

D.3 — Dos contextos activos amplían la vista en vez de acotarla

Síntoma: al resolver las rutas de doble uso –mapa público y panel municipal– la primera solución activaba siempre el contexto público y sumaba el de comuna cuando había sesión, bajo la intuición de que el contexto público, al ser el más restrictivo, no podía hacer daño. El resultado fue el contrario: un administrador autenticado pasaba a ver centros activos de todas las comunas.

Causa raíz: en PostgreSQL, varias políticas permisivas sobre una misma tabla se combinan con una disyunción, no una conjunción. Tener activos ambos contextos a la vez no produce la intersección de las dos vistas, sino su unión: los centros propios más todos los centros activos de todas las comunas.

Solución: resolver un único contexto por petición, por exclusión: si hay sesión se fija el contexto de comuna, y si no la hay, el público. Nunca ambos. Es la regla de diseño que el Capítulo 4 enuncia al describir los tres contextos, y que el middleware aplica en un solo punto.

Lección: en un modelo de varios contextos sobre las mismas tablas, el riesgo no está en olvidar activar uno, que falla de forma ruidosa y visible –no se ve nada–, sino en activar dos, que falla en silencio y en la dirección peligrosa. Conviene que la elección del contexto sea estructuralmente excluyente, y no el resultado de acumular condiciones.

D.4 — Las claves foráneas ignoran RLS: ocultar una fila no impide referenciarla

Síntoma: una comuna podía inscribirse a sí misma como participante de un `SuperEvento` ajeno, sin invitación y sin haber visto nunca ese evento, conociendo solo su identificador numérico.

Causa raíz: la política de inserción sobre las participaciones solo exigía que la fila declarara la comuna activa (`municipality_id = current_tenant()`), bajo el supuesto de que un `SuperEvento` que la política de lectura le oculta a una comuna era, por eso mismo, inalcanzable para ella. El supuesto es incorrecto: PostgreSQL verifica las claves foráneas omitiendo RLS, por diseño, para no comprometer la integridad referencial. Que la comuna no pudiera leer el evento no le impedía apuntar a él, y como los identificadores son enteros correlativos, adivinar uno es trivial. La única era la cláusula `WITH CHECK` de la política, y era demasiado permisiva: dejaba a una comuna autoincorporarse y rompía el consentimiento explícito que exige RF9.

Solución: acotar la inserción a tres casos: el Super Administrador; una comuna que ya participa del `SuperEvento`, que es lo que le permite invitar a otras; y la primera inscripción de la comuna que originó el evento, limitada

a ese mismo evento y no a cualquiera (Anexo D.1). En el segundo caso, además, el valor de estado debe ser invitada: invitar solo puede dejar a la otra comuna pendiente, nunca participando; pasar a participando es el consentimiento de esa comuna y lo da ella sobre su propia fila. Aceptar una invitación dejó de ser un INSERT y pasó a ser un UPDATE del estado de la propia fila de participación, de modo que cada comuna decide solo sobre su fila y ninguna puede crearse una a su nombre en un evento ajeno.

Lección: en un esquema con RLS, que una fila sea invisible no es un control de acceso sobre quien puede referenciarla: ocultar no es proteger. Las comprobaciones de integridad omiten RLS, de modo que la decisión de quién puede apuntar a qué debe expresarse en la cláusula de escritura y no deducirse de lo que la lectura oculta. Es el reverso del desafío siguiente (Anexo D.5): allí, esa misma omisión de RLS en las verificaciones de unicidad era la única señal fiable disponible; aquí, la omisión en las de clave foránea dejaba pasar lo que la lectura creía haber ocultado.

D.5 — ON CONFLICT amplía el conjunto de políticas que deben cumplirse

Síntoma: el INSERT ... ON CONFLICT DO NOTHING que hacía idempotente una invitación fallaba con «new row violates row-level security policy», aunque la comuna sí tenía permiso para invitar. El mismo INSERT sin ON CONFLICT pasaba sin problema.

Causa raíz: con ON CONFLICT, PostgreSQL exige además que la fila nueva sea visible bajo la política de SELECT de esa tabla. La fila de una invitación lleva el identificador de la comuna invitada, que es justamente lo que la política de lectura le oculta a quien invita: una comuna puede crear esa fila, pero no verla. Se descartaron antes las políticas de inserción y actualización —el fallo persiste aun permitiéndolo todo en la primera y eliminando la segunda—, y la confirmación fue doble: una tabla de prueba con lectura abierta tolera ON CONFLICT, y la tabla real deja de fallar si se relaja su política de lectura.

Solución: insertar sin ON CONFLICT y tratar la violación de unicidad como «esa comuna ya estaba invitada», dentro de un punto de retorno para que un duplicado no aborte la transacción completa de la petición —recuérdese que todo el request vive en una sola transacción (Sección 5.2.4)—. Las verificaciones de unicidad de PostgreSQL sí omiten RLS, así que ese error es la única señal fiable disponible.

Lección: ON CONFLICT no es azúcar sintáctica sobre un INSERT cuando hay RLS de por medio. En una tabla donde se escriben filas que su propio autor no puede leer —y una invitación es exactamente eso— la cláusula deja de ser utilizable. Enlaza con la lección del primer desafío: el aislamiento se aplica en más lugares de los que uno tiene presentes al escribir la consulta. La restricción es propia del camino municipal: el Super Administrador sí puede usar ON CONFLICT porque pasa la política de lectura y ve la fila que escribe.

Anexo E — Galería de pantallas del PMV

Las siete pantallas que la extensión multi-tenant incorporó al frontend (Tabla 12), en el mismo orden en que esa tabla las enumera, seguidas de la pantalla heredada que la colaboración amplió. Se reproducen aquí en su estado normal de operación, sobre el entorno de datos sembrados descrito en la Sección 5.1. Las capturas del Capítulo 6 cumplen un propósito distinto –acreditar un comportamiento observable, generalmente contrastando dos comunas ante la misma acción– y no sustituyen a estas.

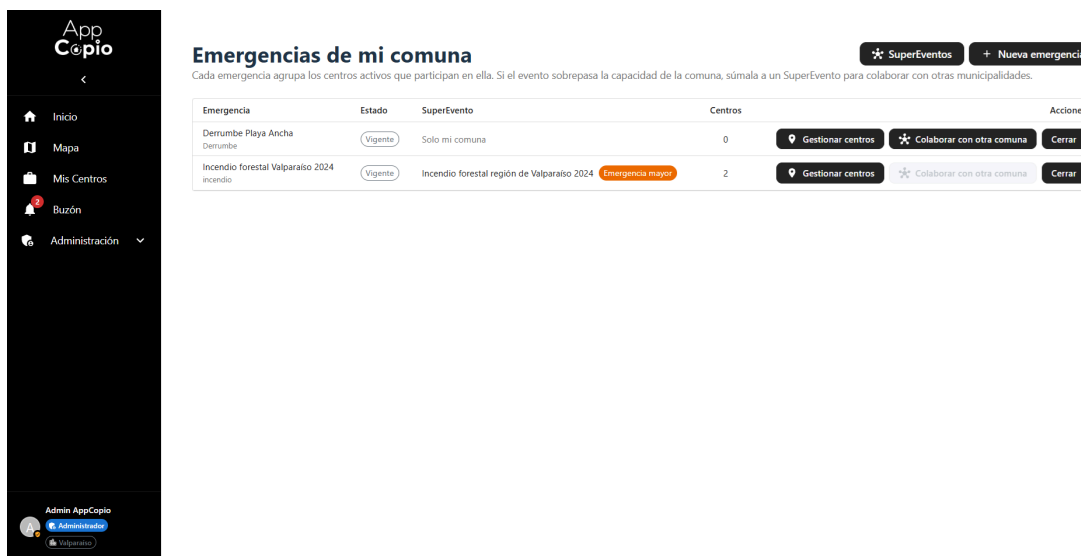


Figura 13: Emergencias de la comuna. Sesión de admin (Valparaíso): emergencias locales declaradas, con su estado y la acción de lanzar colaboración sobre una de ellas. Fuente: Elaboración propia.

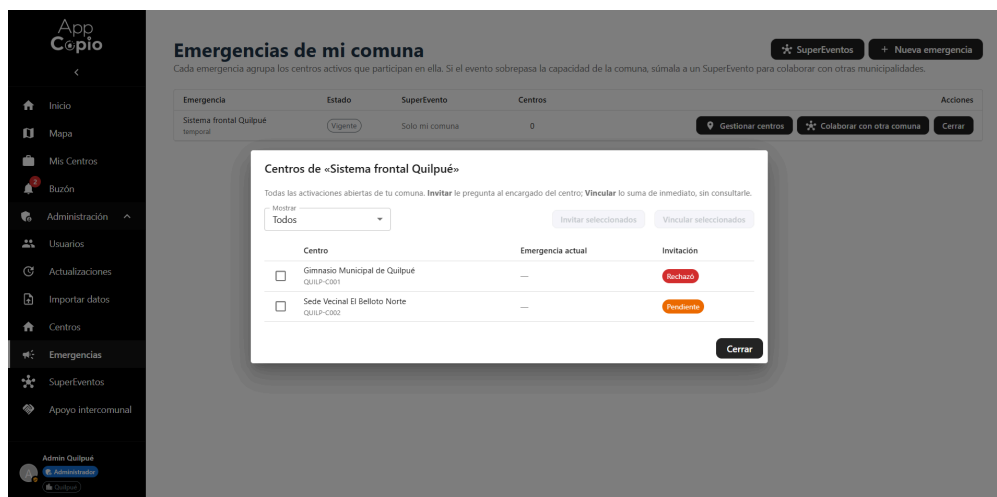


Figura 14: Diálogo de gestión de centros de una emergencia. Sesión de admin. quilpue (Quilpué): activaciones abiertas de la comuna con su estado frente a la emergencia –uno que rechazó y otro pendiente de responder–, e invitación o vinculación individual y en bloque. Fuente: Elaboración propia.

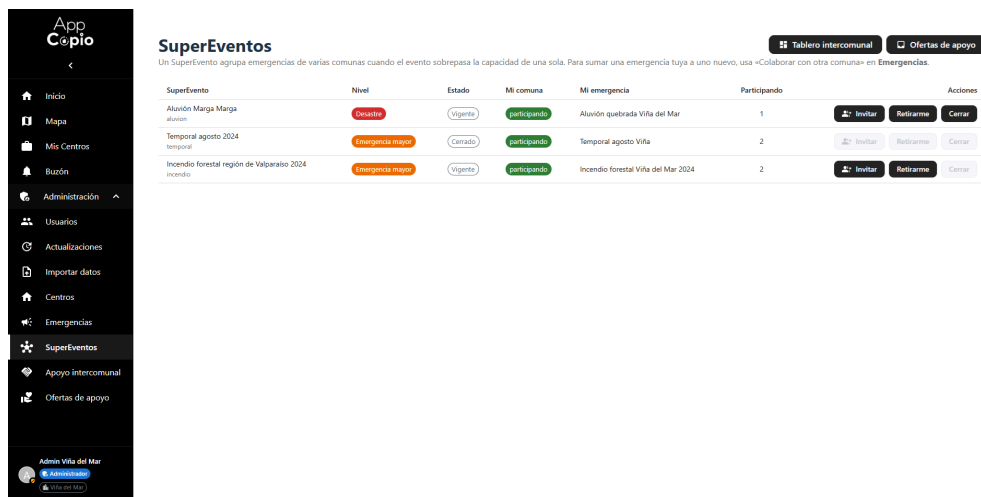


Figura 15: SuperEventos de la comuna. Sesión de admin.vina (Viña del Mar): los SuperEventos en que participa –dos vigentes y uno ya cerrado– con la emergencia local que aporta a cada uno. «Cerrar» solo está habilitado en el SuperEvento que la comuna originó. En los vigentes, «Retirarme» le permite abandonarlos sin cerrarlos.

Fuente: Elaboración propia.

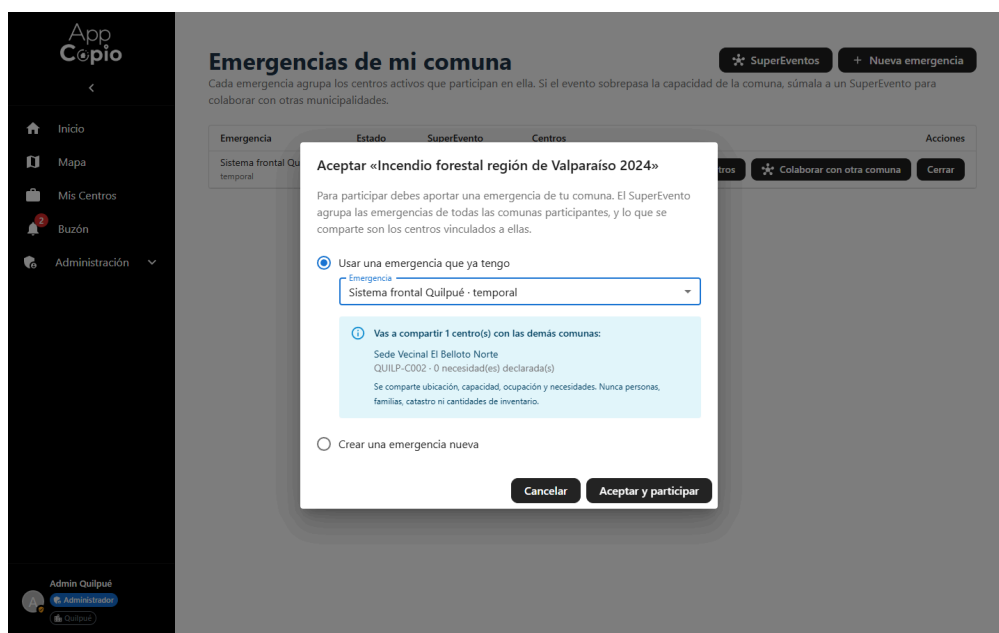
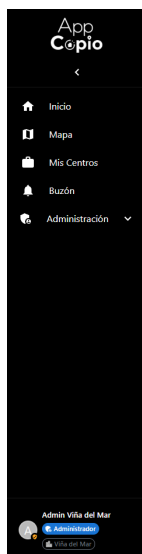


Figura 16: Diálogo de aceptación de una invitación a un SuperEvento. Sesión de admin.quilpué (Quilpué): la aceptación exige aportar una emergencia propia y muestra la vista previa de los centros que quedarán compartidos, tras aceptar el encargo de QUILP-C002 la convocatoria que aparecía pendiente en la Figura 14.

Fuente: Elaboración propia.



Tablero intercomunal

Centros activos de las otras comunas participantes y sus necesidades. Solo se comparten ubicación, capacidad, nivel de abastecimiento y prioridades: los datos de personas, familias e inventario nunca cruzan de comuna.

SuperEvento: Incendio forestal región de Valparaíso 2024

Comuna: [dropdown] Emergencia: [dropdown] Ítem necesitado: [dropdown] Prioridad mínima: Todas Ordenar por: Urgencia [dropdown] [Usar mi ubicación]

Mostrando 2 de 2 centros · ordenados por urgencia

Gimnasio Municipal de Valparaíso

Abastecimiento 0% · Capacidad 150 · abierto

Necesidades

Agua Embotellada (1.5L) · alto

Frascos (1.5 piezas) · medio

Pañales para Niños (Talla G) · bajo

[Ofrecer apoyo]

Liceo Bicentenario

Abastecimiento 0% · Capacidad 80 · abierto

Necesidades

Sin prioridades declaradas.

[Ofrecer apoyo]

Figura 17: Tablero intercomunal, vista de listado. Sesión de admin.vina (Viña del Mar): centros compartidos del SuperEvento con su comuna de origen, prioridades de necesidades y filtros. Fuente: Elaboración propia.



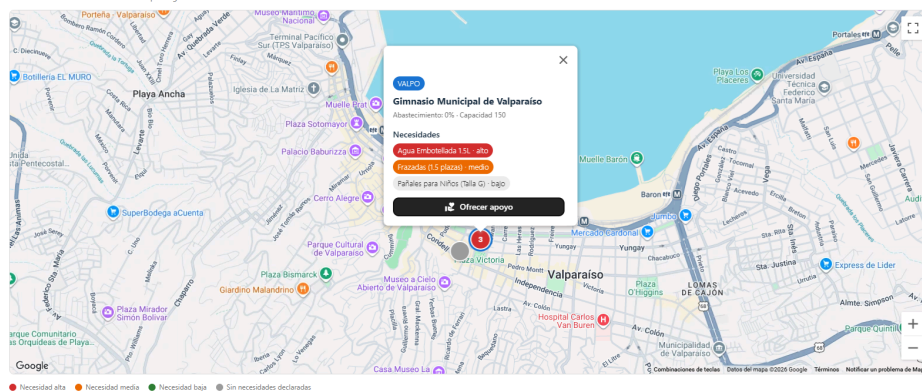
Tablero intercomunal

Centros activos de las otras comunas participantes y sus necesidades. Solo se comparten ubicación, capacidad, nivel de abastecimiento y prioridades: los datos de personas, familias e inventario nunca cruzan de comuna.

SuperEvento: Incendio forestal región de Valparaíso 2024

Comuna: [dropdown] Emergencia: [dropdown] Ítem necesitado: [dropdown] Prioridad mínima: Todas Ordenar por: Urgencia [dropdown] [Usar mi ubicación]

Mostrando 2 de 2 centros · ordenados por urgencia



The map view shows the location of the Gimnasio Municipal de Valparaíso. A pop-up window displays the center's details: Abastecimiento 0% · Capacidad 150. The needs section is identical to the list view: Agua Embotellada (1.5L) · alto, Frascos (1.5 piezas) · medio, and Pañales para Niños (Talla G) · bajo. A legend at the bottom indicates the color coding for needs: red for high, orange for medium, green for low, and grey for no needs declared.

Figura 18: Tablero intercomunal, vista de mapa: el mismo conjunto de centros y los mismos filtros de la Figura 17, sobre el eje geográfico. Cambiar de vista no altera el recorte, solo la forma de leerlo. Fuente: Elaboración propia.

App Copio

Inicio

Mapa

Mis Centros

Buzón

Administración

Usuarios

Actualizaciones

Importar datos

Centros

Emergencias

SuperEventos

Apoyo intercomunal

Ofertas de apoyo

Admin Viña del Mar

Administrador

Ofertas de apoyo

Ofrecimientos de recursos entre comunas participantes de un mismo SuperEvento.

RECIBIDAS

ENVIADAS

Destino	Centro	Item	Mensaje	SuperEvento	Estado	Acciones
Valparaíso	Gimnasio Municipal de Valparaíso	Agua Embotellada 1.5L	Tenemos 100 botellas disponibles para traslado.	Incendio forestal región de Valparaíso 2024	Borrador	Enviar Cancelar
Valparaíso	Gimnasio Municipal de Valparaíso	Agua Embotellada 1.5L	Tenemos 400 botellas de agua listas para traslado esta tarde.	Incendio forestal región de Valparaíso 2024	Pendiente	Cancelar
Valparaíso	Liceo Bicentenario	Frazadas (1.5 plazas)	Teníamos frazadas disponibles, pero ya se comprometieron.	Incendio forestal región de Valparaíso 2024	Cancelada	—

Los borradores son visibles solo para tu municipalidad; el administrador debe enviarlos para que lleguen al destino. Solo la comuna que ofrece puede cancelar; solo la que recibe puede aceptar o rechazar. Estás operando como Viña del Mar.

Figura 19: Bandeja de ofertas de apoyo. Sesión de admin.vina (Viña del Mar), pestaña de enviadas: cada oferta con su estado –incluido un borrador pendiente de aprobación interna– y las acciones disponibles según el lado y el estado desde el que se la observa. Fuente: Elaboración propia.

App Copio

Inicio

Mapa

Municipalidades

SuperEventos

Super Administrador

Super Administrador

Municipalidades

+ Nueva municipalidad

Cada municipalidad se crea junto a su primer administrador. El prefijo se usa para los identificadores de sus centros (por ejemplo VALPO-C001).

Comuna	Prefijo	Estado	Centros creados	Acciones
Concón	CONCO	Activa	3	Gestionar
Quilpué	QUILP	Activa	3	Gestionar
Valparaíso	VALPO	Activa	12	Gestionar
Viña del Mar	VINA	Activa	3	Gestionar

Figura 20: Panel de Super Administrador. Sesión de superadmin: listado de municipalidades, relevo de administradores desde «Gestionar»; el menú no ofrece acceso a centros ni datos operativos de comuna alguna. Fuente: Elaboración propia.



App Cepio

<

Inicio

Mapa

Municipalidades

SuperEventos

Super Administrador

SuperEventos

Un SuperEvento agrupa las emergencias de varias comunas cuando el evento sobrepasa la capacidad de una sola. Las emergencias siguen siendo locales: acá se coordinan, no se reemplazan.

SuperEvento	Nivel	Origen	Estado	Comunas	Acciones
Alusión Marga Marga	Creado	Creado por una comuna	Vigente	1	Invitar Agrupar Cerrar
Temporal agosto 2024	Emergencia mayor	Regional	Cerrado	2	Invitar Agrupar Cerrar
Incendio forestal región de Valparaíso 2024	Emergencia mayor	Regional	Vigente	2	Invitar Agrupar Cerrar

Emergencias sin SuperEvento

Emergencias locales abiertas que ninguna comuna ha sumado a un SuperEvento. Si dos o más son en realidad el mismo evento, agrúpalas en uno.

Emergencia	Comuna	Tipo	Centros vinculados
Sistema frontal Quilpué	QUILP	temporal	0

Figura 21: Panel de Super Administrador, vista de SuperEventos: creación de eventos, agrupación de emergencias existentes e invitación de comunas. Fuente: Elaboración propia.

App Cepio

<

Inicio

Mapa

Mis Centros

Buzón

Mis Turnos

Apoyo intercomunal

Ofertas de apoyo

Martina Tago

Mis Centros Asignados

Selecciona un centro para ver sus detalles y gestionar su inventario.

Mostrando 1 centro asignado

Liceo Bicentenario

ACTIVO

COMPARTIDO

Gestionar Inventario

Ver Detalles

Independencia 456 - Acopio

Emergencia: Incendio forestal Valparaíso 2024

Este centro se está compartiendo con otras comunas en «Incendio forestal región de Valparaíso 2024»

Figura 22: Pantalla heredada «Mis centros», ampliada por la colaboración. Sesión de martinalina (trabajador municipal, Valparaíso). Fuente: Elaboración propia.

Anexo F — Matriz de trazabilidad completa

Detalle requisito por requisito de cómo se acredita cada uno de los veinte requisitos definidos en el Capítulo 4, complementando la síntesis por modo de acreditación presentada en el Capítulo 6 (Tabla 13).

Requisito	Descripción resumida	Cómo se acredita
RF1	Múltiples municipalidades operan sobre la misma instancia, sin infraestructura propia.	Escenario 3 (alta de una comuna nueva sobre la instancia en marcha).
RF2	Rol de Super Administrador por sobre las municipalidades.	Escenario 3: incluida la comprobación, directo en la base, de que no ve ni modifica datos operativos, y la lista blanca de las tablas y funciones que lo admiten.
RF3	Cada municipalidad administra su propio registro de usuarios.	Escenarios 1 y 3.
RF4	Cada municipalidad mantiene su propio registro de centros.	Escenario 1.
RF5	Un administrador no ve centros de otras comunas salvo colaboración explícita.	Escenario 1.
RF6	Catálogos base compartidos, extensibles por comuna.	Escenario 4: los cuatro índices únicos parciales, y la prueba de que dos comunas crean una categoría homónima sin colisionar mientras el catálogo base sigue siendo único.
RF7	Las funcionalidades heredadas de la versión single-tenant siguen operando, acotadas a cada comuna.	Escenario 4: inventario, registro de residentes y prioridades de un centro responden 200 dentro de la propia comuna.
RF8	Existe un nivel de agrupación mayor a la activación individual, para centros de una o varias comunas.	Escenario 2 (el SuperEvento y las emergencias que agrupa).
RF9	La incorporación a la colaboración es explícita, nunca automática.	Escenario 2: incluida la prueba de que una comuna participante no puede inscribir a otra como participando. Escenario 5: la reinvitación tras un rechazo o un retiro es exclusiva del Super Administrador.
RF10	Visibilidad agregada del estado y necesidades de los centros del evento compartido.	Escenarios 2 y 5.
RF11	Una comuna declara una necesidad y otra responde con una oferta de apoyo sobre ella.	Escenario 2: una comuna ajena a la oferta no puede resolverla, un trabajador sin apoyo administrativo tampoco, y el borrador solo existe para la comuna que lo emitió. Escenario 5: el retiro de una comuna cancela sus ofertas sin resolver. El script no recorre las transiciones del ciclo (borrador, pendiente, aceptada); el estado sincronizado entre bandejas se acredita visualmente.
RF12	Los datos sensibles nunca cruzan de comuna, ni colaborando en un mismo evento.	Escenarios 1, 2 y 4.

RNF1	Aislamiento garantizado a nivel de base de datos.	Escenario 1.
RNF2	Extensión aditiva sobre el modelo de datos existente.	Por diseño (Capítulo 5) y por la continuidad funcional verificada en el Escenario 4.
RNF3	Sin sobrecarga significativa en las tablas de mayor volumen.	No validado empíricamente. Se argumenta por diseño (desnormalización del discriminador) y se identifica como trabajo futuro en el Capítulo 7.
RNF4	Mecanismos simples y estándar por sobre soluciones a medida.	Por diseño: RLS nativo de PostgreSQL, sin capa de federación propia (Sección 4.2).
RNF5	Sin duplicar infraestructura por cada comuna incorporada.	Escenario 3: las comunas conviven en un solo esquema de una sola base, y la comuna nueva se crea sobre la instancia en marcha sin aprovisionar nada.
RNF6	No se requiere migración de datos productivos.	Sección 4.4.
RNF7	La información compartida permite identificar origen y momento de publicación.	Escenarios 2 y 5: ninguna participación resuelta queda sin fecha de respuesta, y la comuna de origen y la fecha de cada oferta quedan garantizadas por el esquema (NOT NULL); las cancelaciones automáticas dejan registrado su motivo; sostenido por esquema con <code>created_by</code> , <code>invited_by</code> , <code>joined_at</code> , <code>responded_at</code> y <code>created_at</code> .
RNF8	Toda excepción al aislamiento estándar es acotada y de superficie mínima.	Por catálogo de funciones con privilegios elevados y sus <code>REVOKE ... FROM PUBLIC</code> (Sección 5.2.5), y por la lista blanca de políticas y funciones que admiten al Super Administrador (Escenario 3).

Tabla 21: Matriz de trazabilidad entre los requisitos del PMV y su forma de acreditación. Fuente: Elaboración propia.

Anexo G — Evidencia visual de los escenarios de validación

Capturas de pantalla que acompañan a los cinco escenarios del Capítulo 6 como evidencia secundaria de la capa de interfaz. Se agrupan aquí para no interrumpir la lectura del capítulo, que remite a cada una donde corresponde. Las pantallas que ya aparecen en el Anexo E no se repiten: el capítulo remite directamente a esa galería.

G.1 — Escenario 1: aislamiento estricto de datos

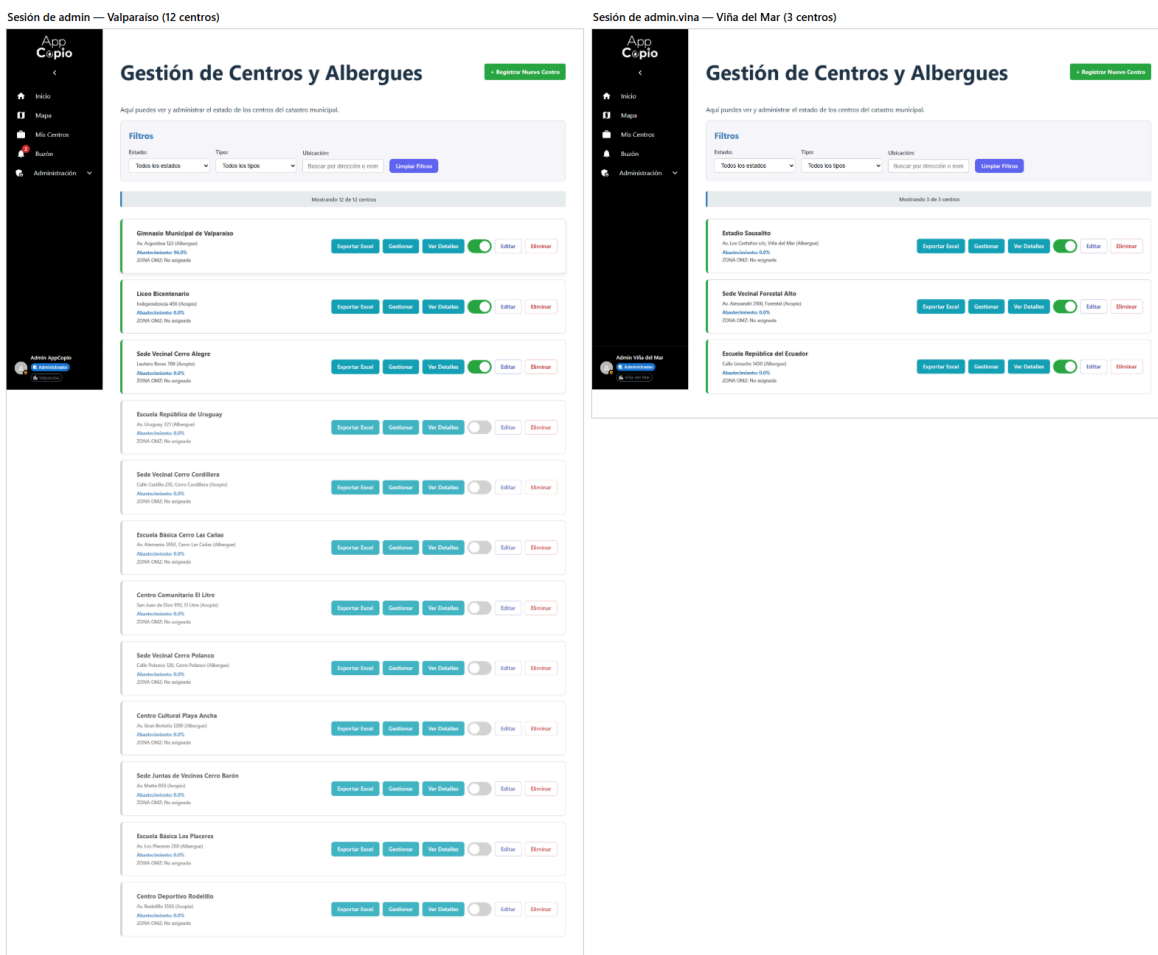


Figura 23: Listado «Mis Centros» en dos sesiones simultáneas: admin (Valparaíso, 12 de 12 centros) y admin.vina (Viña del Mar, 3 de 3). Ningún centro aparece en las dos. Fuente: Elaboración propia.

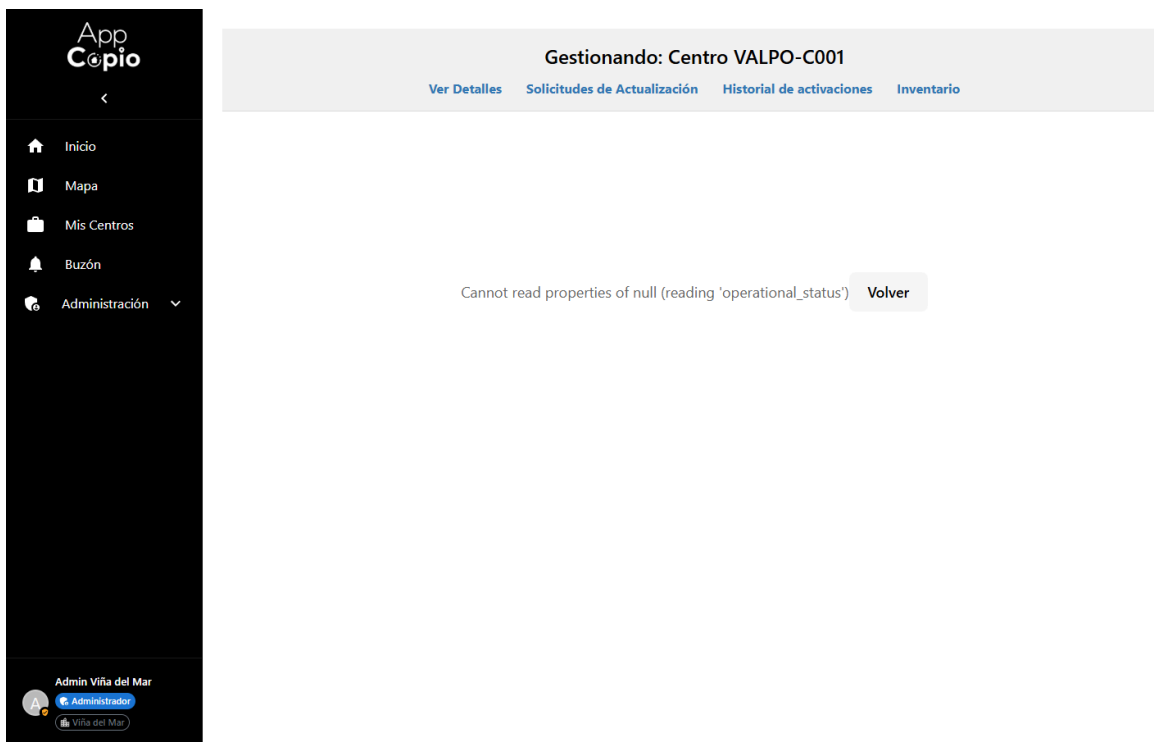


Figura 24: Intento de alcanzar por URL un centro de Valparaíso desde la sesión de Viña del Mar: la API responde 404 (comprobado por el script, Tabla 22) y la interfaz, que no maneja ese caso de forma amigable (deuda heredada del frontend), no muestra ningún dato del centro ajeno. Fuente: Elaboración propia.

G.2 — Escenario 2: colaboración inter-municipal con consentimiento

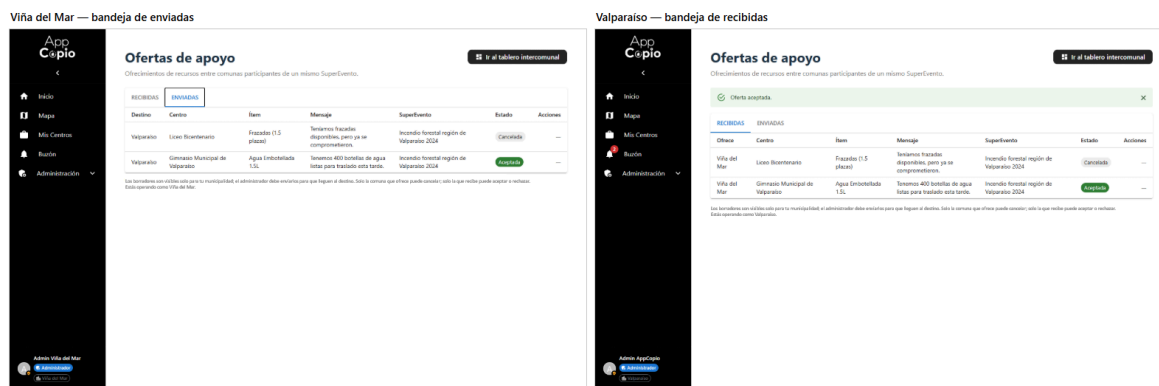


Figura 25: Una misma oferta de apoyo vista desde las dos comunas: bandeja de enviadas de Viña del Mar y bandeja de recibidas de Valparaíso, con el estado sincronizado entre ambas. Fuente: Elaboración propia.

G.3 — Escenario 4: continuidad funcional y canal público

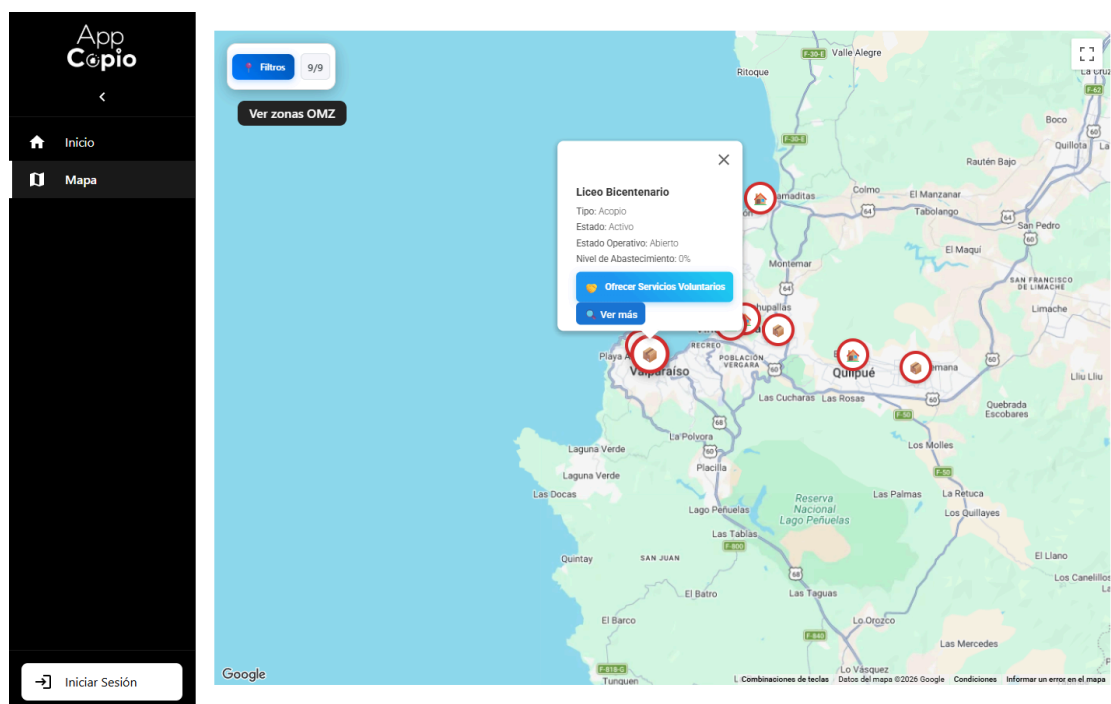


Figura 26: Mapa público consultado sin sesión iniciada: centros activos de varias comunas y la ficha reducida de uno de ellos, que muestra solo tipo, estado, estado operativo y nivel de abastecimiento; las necesidades prioritizadas se despliegan desde «Ver más». Fuente: Elaboración propia.

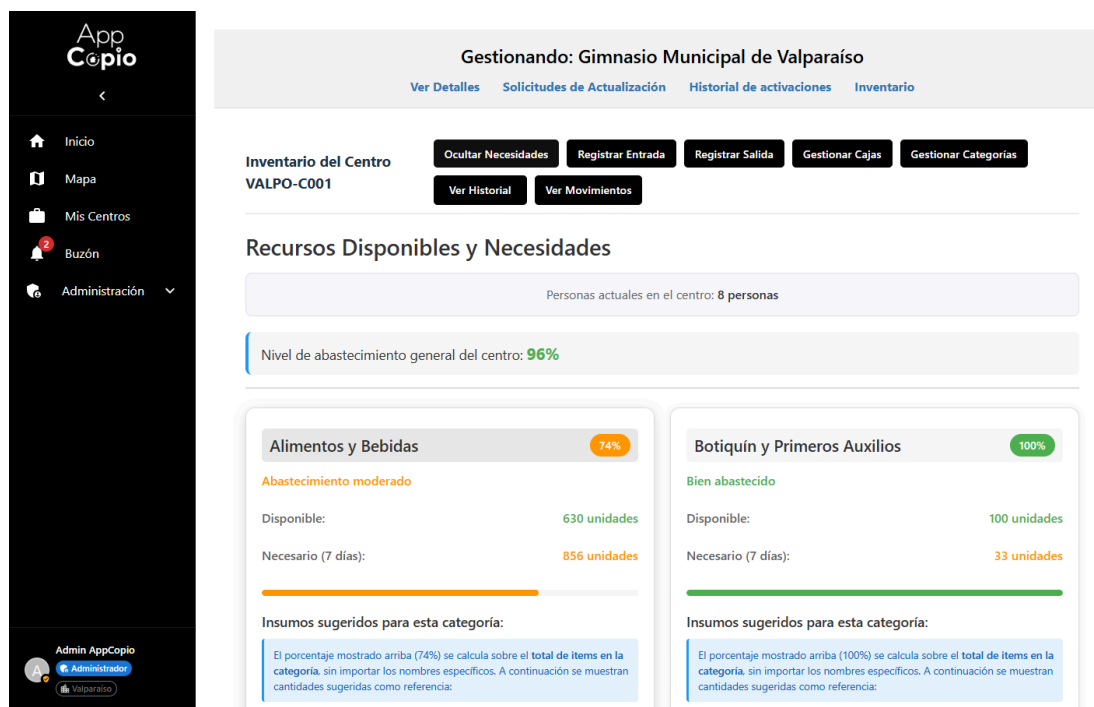


Figura 27: Módulo de inventario de un centro operando con normalidad dentro de la sesión de su comuna, sin regresiones respecto de la versión single-tenant. Fuente: Elaboración propia.

G.4 — Escenario 5: cierre, retiro y reingreso a la colaboración

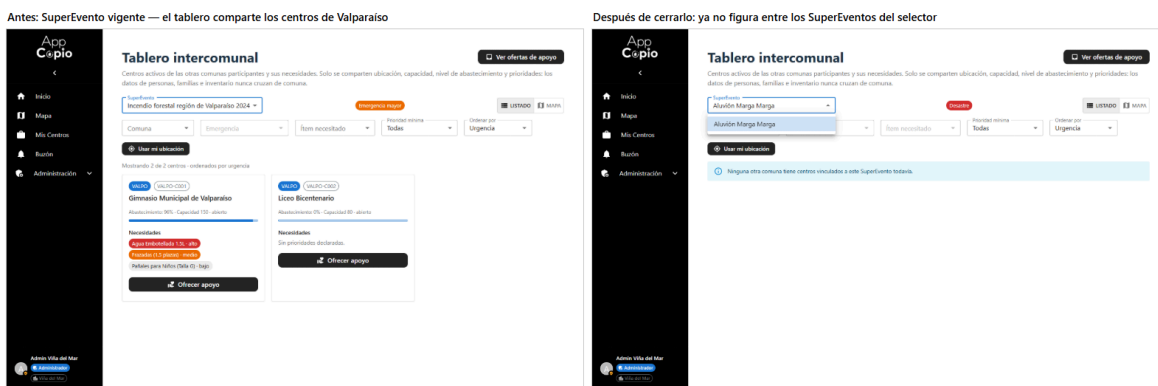


Figura 28: Efecto del cierre en la interfaz, desde la sesión de `admin.vina` (Viña del Mar): antes, el SuperEvento 1 («Incendio forestal región de Valparaíso 2024») comparte los centros de Valparaíso con sus necesidades prioritizadas; después de cerrarlo, deja de figurar entre los SuperEventos seleccionables del tablero. Tras el cierre, solo permanece seleccionable el SuperEvento 2 («Aluvión Marga Marga»), que aún no tiene centros vinculados.

Fuente: Elaboración propia.

Anexo H — Procedimientos de los escenarios de validación

Pasos de ejecución de cada uno de los cinco escenarios del Capítulo 6, contra el entorno y los usuarios de prueba descritos en ese capítulo (Tabla 14). Cada procedimiento se divide según el tipo de evidencia que produce: las *comprobaciones del script* son las que ejecuta `scripts/validar_multitenant.sh` una única vez, sobre el entorno recién recreado. El *recorrido en la interfaz* es lo que se ejecuta a mano en la aplicación y queda registrado en una captura (Anexo G y Anexo E).

H.1 — Escenario 1: aislamiento estricto de datos

Comprobaciones del script (evidencia primaria; resultados en Tabla 15 y Tabla 22):

1. Consultar la base como el rol de aplicación `appcopio_app`, sin fijar contexto de tenant, y verificar que ese rol no puede eludir RLS, que el backend no se conecta como superusuario y que ninguna tabla con `municipality_id` queda sin RLS habilitado y forzado.
2. Iniciar sesión por API como `admin` (Valparaíso) y `admin.vina` (Viña del Mar), contar los centros de cada una y comprobar que ningún identificador aparece en ambas.
3. Desde la sesión de Viña del Mar, solicitar por identificador un centro de Valparaíso, primero para leerlo y luego para editarlo.
4. Con el tenant de Viña del Mar fijado, consultar las personas de Valparaíso y el almacén de tokens de sesión.
5. Contrastar los padrones de usuarios de ambas comunas, verificando que cada una administra uno propio no vacío y que ninguna cuenta aparece en las dos; y verificar que las notificaciones sembradas tienen destinatario y tipo.

Recorrido en la interfaz (evidencia secundaria, Anexo G):

1. Iniciar sesión simultánea como `admin` y `admin.vina` en dos navegadores independientes y confirmar que cada uno lista únicamente los centros de su comuna (Figura 23).
2. Desde la sesión de Viña del Mar, abrir por URL un centro de Valparaíso y confirmar que no se muestra ningún dato suyo (Figura 24).

H.2 — Escenario 2: colaboración inter-municipal con consentimiento

Comprobaciones del script (evidencia primaria; resultados en Tabla 16 y Tabla 23):

1. Verificar que Quilpué, invitada al SuperEvento 1 («Incendio forestal región de Valparaíso 2024») pero aún sin responder, figura como invitada y que sus prioridades no aparecen en el tablero de las comunas participantes.
2. Consultar el tablero desde cada comuna participante y comprobar que ve los centros de la otra; inspeccionar la respuesta campo por campo, contrastándola contra la lista de campos que el diseño autoriza a cruzar – tanto a nivel del centro como de cada necesidad declarada–.
3. Como comuna no participante, solicitar el tablero por API; y como comuna no invitada, intentar inscribirse escribiendo directamente contra la base, sorteando por completo la aplicación.
4. Ejercitar las tres reglas del modelo de participación: aceptar sin aportar una emergencia, aceptar aportando la emergencia de otra comuna y aportar una segunda emergencia propia al mismo SuperEvento.
5. Comprobar que invitar no es privilegio de quien originó el SuperEvento: una comuna participante que no lo creó suma a una tercera; que esa facultad no permite inscribirla como participando; y que una comuna puede originar su propio SuperEvento e ingresar a él en un solo acto, escribiendo directamente contra la base.
6. Contrastar el reparto de roles dentro de una comuna llamando la API directamente: el trabajador municipal y el trabajador con apoyo administrativo acceden al tablero; el contacto ciudadano no; y el trabajador sin apoyo no puede declarar una emergencia ni resolver una oferta.
7. Contar las ofertas de Valparaíso en cada bandeja, comprobar que el borrador emitido por una comuna no es visible para la destinataria y que una comuna ajena a la oferta no puede resolverla.
8. Contrastar, sobre las notificaciones sembradas de Valparaíso (una invitación y una oferta), que llegan únicamente al administrador y al trabajador con apoyo administrativo, y que un trabajador sin ese privilegio no recibe ninguna. El sembrado reproduce a mano ese reparto de destinatarios; la creación del aviso por las funciones `notify_*` no la ejerce el script y se observa solo en la interfaz.

Recorrido en la interfaz (evidencia secundaria):

1. Abrir el tablero intercomunal desde `admin.vina` y confirmar que ve los centros activos de Valparaíso con sus necesidades priorizadas (Figura 17).
2. Como `admin.quilpue`, abrir la invitación pendiente y confirmar que el diálogo exige aportar una emergencia de la propia comuna antes de aceptar (Figura 16).
3. Aceptar la oferta como `admin` (Valparaíso) y confirmar que el cambio de estado se refleja en la bandeja de enviadas de `admin.vina` (Figura 25).

H.3 — Escenario 3: gobernanza de administración

Comprobaciones del script (evidencia primaria; resultados en Tabla 17):

1. Con sesión de `superadmin` por API, solicitar el listado de centros y el de municipalidades.
2. Consultar la base como el rol de aplicación con el contexto de Super Administrador fijado: contar las filas de las tablas operativas y de personas, compararlas con su conteo sin restricción de RLS para descartar que el cero sea trivial, e intentar modificar centros y personas (RF2). Verificar además que sí ve a los usuarios de las comunas.
3. Comprobar, con una lista blanca, qué políticas y funciones de la base admiten al Super Administrador y que coinciden exactamente con lo que su rol administra.
4. Contar los esquemas y las bases de datos existentes, para verificar que las cuatro comunas sembradas conviven en un único esquema de una única base.
5. Dar de alta una municipalidad nueva sobre la instancia en marcha y comprobar que nace sin centros heredados de ninguna otra comuna, con su propio correlativo de identificadores partiendo de cero y con su administrador atado únicamente a ella; eliminarla después y verificar que no queda rastro (RF1, RNF5).

Recorrido en la interfaz (evidencia secundaria):

1. Iniciar sesión como superadmin y confirmar que el menú no ofrece, en ninguna pantalla, acceso a centros, activaciones ni datos operativos de comuna alguna (Figura 20).

H.4 — Escenario 4: continuidad funcional y canal público**Comprobaciones del script** (evidencia primaria; resultados en Tabla 18):

1. Con sesión de Valparaíso por API, solicitar sobre un centro activo de esa comuna el inventario, el registro de residentes y las prioridades de necesidades, y confirmar que responden 200.
2. Solicitar el mapa público sin sesión y contrastar el número de centros que devuelve contra el número de centros activos que registra la base, y el número de comunas representadas contra el de comunas con centros activos.
3. Inspeccionar campo por campo la respuesta anónima, contrastándola contra la lista de campos que el canal público tiene autorizado exponer, y verificar que no contiene ninguno fuera de ella.
4. Verificar que existen los cuatro índices únicos parciales que separan el catálogo global del propio de cada comuna.
5. Crear, dentro de una misma transacción y cambiando de contexto de tenant entre medio, una categoría con idéntico nombre en dos comunas distintas; e intentar después duplicar un nombre del catálogo base.

Recorrido en la interfaz (evidencia secundaria):

1. Abrir el mapa público sin iniciar sesión y confirmar que muestra centros de las cuatro comunas sembradas y que la ficha de un centro expone solo los datos reducidos (Figura 26).
2. Con sesión de Valparaíso, abrir el módulo de inventario de un centro activo y confirmar que opera con normalidad (Figura 27).

H.5 — Escenario 5: cierre, retiro y reingreso a la colaboración

Comprobaciones del script (evidencia primaria; resultados en Tabla 19). Las de retiro, reinvitación y cierre ejecutan contra la base las mismas sentencias que el backend, sin invocar sus endpoints:

1. Confirmar que el SuperEvento 3 («Temporal agosto 2024»), cerrado desde el sembrado y en que participan Viña del Mar y Concón, conserva sus emergencias locales abiertas.
2. Solicitar su tablero intercomunal por API como una de las comunas que sí participa de él.
3. Consultar directamente contra la base, con el contexto de tenant de esa misma comuna, la función que resuelve el conjunto de centros compartidos.
4. Intentar emitir una oferta de apoyo dirigida a un centro de ese SuperEvento.
5. Retirar a Valparaíso del SuperEvento 1 («Incendio forestal región de Valparaíso 2024») y comprobar que sus ofertas sin resolver –el borrador que emitió y la pendiente que recibió– se cancelan solas, cada una con el motivo que corresponde; retirar a Viña y comprobar que solo se cancela su oferta pendiente y no las ya resueltas; y comprobar que ninguna cancelación hecha a mano lleva motivo de cierre automático.
6. Como comuna participante, intentar reinvitar a otra que rechazó (Concón, en el SuperEvento 2, «Aluvión Marga Marga»); como Super Administrador, reinvitar a la que rechazó y a la que se retiró, y comprobar que no pisa a una que ya participa ni a una con la invitación pendiente.
7. Cerrar el SuperEvento 2 («Aluvión Marga Marga») como su comuna originaria (Viña) participando, como la misma comuna ya retirada, como una comuna que no lo originó y como Super Administrador.

Recorrido en la interfaz (evidencia secundaria):

1. Con sesión de admin.vina, abrir el tablero intercomunal sobre el SuperEvento 1 y confirmar que comparte los centros de Valparaíso; cerrar luego ese SuperEvento y comprobar que deja de figurar entre los SuperEventos seleccionables del tablero (Figura 28).

Anexo I — Verificaciones por escenario

Detalle completo de las verificaciones de `scripts/validar_multitenant.sh` correspondientes a los escenarios 1 y 2 (Sección 6.1 y Sección 6.2). Las de los escenarios 3, 4 y 5 se presentan completas en el propio capítulo, por ser menos numerosas.

I.1 — Escenario 1

Verificación	Resultado esperado	Resultado obtenido	Estado
Base de datos			
rolsuper / rolbypassrls del rol appcopio_app	false / false	false / false	PASS
Conexiones activas del backend como superusuario postgres	0	0	PASS
Tablas con municipality_id sin RLS habilitado y forzado	0	0	PASS
SELECT COUNT(*) FROM Centers como appcopio_app, sin tenant fijado	0	0	PASS
SELECT COUNT(*) FROM Persons como appcopio_app, sin tenant fijado	0	0	PASS
Filas del almacén de tokens de sesión, consultado con tenant fijado	0	0	PASS
Notificaciones sembradas sin destinatario o sin tipo	0 / 0	0 / 0	PASS
Aislamiento entre comunas			
Conteo de centros por sesión: Valparaíso vs. Viña del Mar	12 / 3	12 / 3	PASS
Identificadores de centro comunes a ambas comunas	0	0	PASS
Lectura de un centro de otra comuna por identificador (código HTTP / contenido de la respuesta)	404 / sin-datos	404 / sin-datos	PASS
Edición de un centro de otra comuna por identificador	bloqueado	bloqueado	PASS
Personas de Valparaíso consultadas con el tenant de Viña del Mar	0	0	PASS
Cuentas de usuario comunes a ambas comunas	ninguno	ninguno	PASS
Cada comuna administra un padrón propio no vacío	si	si	PASS

Tabla 22: Verificaciones de aislamiento, Escenario 1. Fuente: Elaboración propia.

Nota

bloqueado: la operación fue rechazada —con un error o con cero filas afectadas, según el caso— y no modificó ningún dato.
sin-datos: la respuesta no contiene ningún dato del recurso solicitado.

ninguno: no existe ninguna cuenta presente en ambas comunas.
sí: ambas comunas tienen cuentas de usuario propias.

I.2 — Escenario 2

Verificación	Resultado esperado	Resultado obtenido	Estado
Roles			
Trabajador municipal solicita el tablero	200	200	PASS
Trabajador con apoyo administrativo solicita el tablero	200	200	PASS
Contacto ciudadano solicita el tablero	403	403	PASS
Trabajador sin apoyo declara una emergencia	403	403	PASS
Trabajador sin apoyo resuelve una oferta de su comuna	403	403	PASS
SuperEventos y colaboración			
Tablero del SuperEvento, consultado por cada una de las dos comunas (centros visibles en cada una)	2 / 2	2 / 2	PASS
Comuna que no participa solicita el tablero	403	403	PASS
Campos del tablero fuera de los autorizados por el diseño	exacto	exacto	PASS
Campos de cada necesidad declarada fuera de ítem y prioridad	exacto	exacto	PASS
Estado de participación de la comuna invitada que aún no responde	invitada	invitada	PASS
Prioridades de esa comuna visibles en el tablero antes de aceptar	0	0	PASS
Comuna no invitada intenta inscribirse escribiendo en la base (debe rechazarlo una política de RLS)	bloqueado	bloqueado	PASS
Una comuna origina su propio SuperEvento e ingresa a él	permitido	permitido	PASS
Activaciones vigentes sin emergencia disponibles para vincular	2	2	PASS
Ofertas recibidas / enviadas visibles para Valparaíso	2 / 3	2 / 3	PASS
Ofertas de Valparaíso a Viña visibles para Viña (3 enviadas, una de ellas en borrador)	2	2	PASS
Aceptar un SuperEvento sin aportar una emergencia	400	400	PASS
Aceptar aportando la emergencia de otra comuna	404	404	PASS

Aportar una segunda emergencia propia al mismo SuperEvento (debe rechazarlo el índice único parcial)	bloqueado	bloqueado	PASS
El Super Administrador intenta crear una emergencia	403	403	PASS
Una comuna participante inscribe a otra como participando (debe rechazarlo una política de RLS)	bloqueado	bloqueado	PASS
Una comuna participante no originaria invita a otra	permitido	permitido	PASS
Participaciones resueltas sin fecha de respuesta registrada	0	0	PASS
Comuna ajena intenta resolver una oferta que no le corresponde (por la API)	bloqueado	bloqueado	PASS
Notificaciones			
Tipos de aviso recibidos por el administrador de la comuna destino	super_event_invitation, support_offer	super_event_invitation, support_offer	PASS
Tipos de aviso recibidos por el trabajador con apoyo administrativo	super_event_invitation, support_offer	super_event_invitation, support_offer	PASS
Avisos intercomunales recibidos por un trabajador sin apoyo	0	0	PASS

Tabla 23: Verificaciones de colaboración inter-municipal, Escenario 2. Fuente: Elaboración propia.

Nota

exacto: el conjunto de campos devuelto coincide con la lista autorizada, sin campos adicionales.

bloqueado: la operación fue rechazada –con un error o con cero filas afectadas, según el caso– y no modificó ningún dato.

permitido: la operación se ejecutó sin error y luego se revirtió, sin dejar datos.

Anexo J — Resultado de ejecución script de validación multi-tenant

```
teja@martin-PC: ~/Documents/c/GitHub/AppCopio-MultiTenant (appcopio-multi-tenant)
$ docker compose down -v && docker compose up -d && bash scripts/validar_multitenant.sh

=== Validación multi-tenant de AppCopio ===

Preparación
PASS el backend responde | esperado: 401 | obtenido: 401
OK base recién sembrada
OK sesiones iniciadas

Base de datos
PASS appcopio_app existe y no puede saltarse RLS | esperado: false,false | obtenido: false,false
PASS el backend NO se conecta como superusuario | esperado: 0 | obtenido: 0
PASS toda tabla con municipality_id tiene RLS habilitado y forzado | esperado: 0 | obtenido: 0
PASS sin contexto de tenant, Centros no devuelve ninguna fila | esperado: 0 | obtenido: 0
PASS sin contexto de tenant, Personas no devuelve ninguna fila | esperado: 0 | obtenido: 0
PASS RefreshTokens sigue sellada incluso con tenant fijado | esperado: 0 | obtenido: 0
PASS ninguna notificación quedó sin destinatario | esperado: 0 | obtenido: 0
PASS ninguna notificación quedó sin kind | esperado: 0 | obtenido: 0
PASS todas las comas comparten un solo esquema | esperado: 1 | obtenido: 1
PASS y una sola base de datos | esperado: 1 | obtenido: 1

Aislamiento entre comunas
PASS Valparaíso ve sus 12 centros | esperado: 12 | obtenido: 12
PASS Viña ve sus 3 centros | esperado: 3 | obtenido: 3
PASS ningún centro se cruza entre comunas | esperado: 0 | obtenido: 0
PASS Viña no puede leer un centro de Valparaíso (responde 404 y sin datos) | esperado: 404/sin-datos | obtenido: 404/sin-datos
PASS Viña no puede editar un centro de Valparaíso | esperado: bloqueado | obtenido: bloqueado
PASS las personas de otra comuna no se filtran | esperado: 0 | obtenido: 0
PASS ninguna cuenta de usuario se cruza entre comunas | esperado: ninguno | obtenido: ninguno
PASS y cada comuna administra su propio padrón | esperado: si | obtenido: si

Roles
PASS el Super Admin NO ve centros | esperado: 0 | obtenido: 0
PASS el Super Admin sí ve las municipalidades | esperado: varias | obtenido: varias
PASS como Super Administrador, las tablas operativas y de personas no devuelven ninguna fila | esperado: 0 | obtenido: 0
PASS y esas mismas tablas sí tienen datos sembrados | esperado: si | obtenido: si
PASS como Super Administrador, tampoco puede modificar centros ni personas | esperado: 0 | obtenido: 0
PASS lo que sí ve son los usuarios de las comunas, que es lo que administra | esperado: si | obtenido: si
PASS las únicas tablas cuyas políticas admiten al Super Administrador | esperado: emergencies,municipal_zones,municipalities,supereventparticipants,
superevents,users | obtenido: emergencies,municipal_zones,municipalities,supereventparticipants,superevents,users
PASS y las únicas funciones que lo contemplan | esperado: notify_super_event_invitation,super_event_participants_of
| obtenido: notify_super_event_invitation,super_event_participants_of
PASS un Trabajador Municipal accede al tablero intercomunal | esperado: 200 | obtenido: 200
PASS un Trabajador CON apoyo admin también accede al tablero | esperado: 200 | obtenido: 200
PASS un Contacto Ciudadano NO accede al tablero | esperado: 403 | obtenido: 403
PASS un Trabajador sin apoyo no puede declarar una emergencia | esperado: 403 | obtenido: 403
PASS un Trabajador sin apoyo no puede resolver una oferta de su comuna | esperado: 403 | obtenido: 403

Catálogos compartidos y propios
PASS existen los índices que separan el catálogo global del propio | esperado: 4 | obtenido: 4
PASS dos comunas pueden crear una categoría con el mismo nombre | esperado: permitido | obtenido: permitido
PASS pero ninguna puede duplicar una categoría del catálogo base | esperado: bloqueado | obtenido: bloqueado

SuperEventos y colaboración
PASS el tablero del SEI funciona en ambos sentidos | esperado: 2/2 | obtenido: 2/2
PASS una comuna que no participa recibe 403 | esperado: 403 | obtenido: 403
PASS el tablero expone exactamente los campos autorizados, y ninguno más | esperado: exacto | obtenido: exacto
PASS y cada necesidad declarada, solo ítem y prioridad | esperado: exacto | obtenido: exacto
PASS Quilpú está invitada al SEI, no participando | esperado: invitada | obtenido: invitada
PASS sus prioridades NO se comparten mientras no acepte | esperado: 0 | obtenido: 0
PASS una comuna no invitada no puede autoinscribirse en un SuperEvento ajeno | esperado: bloqueado | obtenido: bloqueado
PASS una comuna puede autocrear su SuperEvento e inscribirse (vía 2: originado por una comuna) | esperado: permitido | obtenido: permitido
PASS Quilpú tiene activaciones sueltas para vincular | esperado: 2 | obtenido: 2
PASS Valparaíso ve 2 ofertas recibidas | esperado: 2 | obtenido: 2
PASS Valparaíso ve 3 ofertas enviadas (incluido su borrador) | esperado: 3 | obtenido: 3
PASS el borrador es invisible para la municipalidad destino | esperado: 2 | obtenido: 2
PASS el SuperEvento cerrado conserva sus emergencias abiertas | esperado: 2 | obtenido: 2
PASS y aun así no comparte ningún centro | esperado: 409 | obtenido: 409
PASS la revocación se aplica en el motor, no solo en la API | esperado: 0 | obtenido: 0
PASS un SuperEvento cerrado no admite ofertas nuevas | esperado: 409 | obtenido: 409
PASS aceptar un SuperEvento sin iniciar emergencia es rechazado | esperado: 400 | obtenido: 400
PASS ni se puede aportar la emergencia de otra comuna | esperado: 404 | obtenido: 404
PASS una comuna solo aporta una emergencia por SuperEvento | esperado: bloqueado | obtenido: bloqueado
PASS el Super Administrador no puede crear emergencias | esperado: 403 | obtenido: 403
PASS una comuna participante no puede inscribir a otra como 'participando' | esperado: bloqueado | obtenido: bloqueado
PASS una comuna participante no originaria puede invitar a otra | esperado: permitido | obtenido: permitido
PASS al retirarse una comuna se cancelan sus ofertas sin resolver, enviadas y recibidas | esperado: destino_retirada,origen_retirada
| obtenido: destino_retirada,origen_retirada
PASS una oferta cancelada por un usuario no lleva motivo de cierre automático | esperado: 0 | obtenido: 0
PASS el retiro no cancela las ofertas ya resueltas | esperado: 1 | obtenido: 1
PASS una comuna no puede reinvitar a otra que ya rechazó | esperado: 0 | obtenido: 0
PASS el Super Administrador reinvita a una comuna que rechazó | esperado: 1 | obtenido: 1
PASS y también a una que se retiró | esperado: 1 | obtenido: 1
PASS pero no pisa a una comuna que ya está participando | esperado: 0 | obtenido: 0
PASS ni a una que sigue con la invitación pendiente | esperado: 0 | obtenido: 0
PASS la comuna originaria participando puede cerrar su SuperEvento | esperado: 1 | obtenido: 1
PASS la originaria que se retiró ya no puede cerrarlo | esperado: 0 | obtenido: 0
PASS una comuna que no lo originó tampoco puede cerrarlo | esperado: 0 | obtenido: 0
PASS el Super Administrador siempre puede cerrarlo | esperado: 1 | obtenido: 1
PASS toda participación resuelta registra cuándo se respondió | esperado: 0 | obtenido: 0
PASS una comuna ajena no puede cambiar el estado de una oferta | esperado: bloqueado | obtenido: bloqueado

Notificaciones
PASS el admin de Valparaíso recibe la oferta y la invitación | esperado: super_event_invitation,support_offer
| obtenido: super_event_invitation,support_offer
PASS el apoyo admin recibe lo mismo que el admin | esperado: super_event_invitation,support_offer | obtenido:
super_event_invitation,support_offer
PASS el trabajador sin apoyo NO recibe avisos intercomunales | esperado: 0 | obtenido: 0

Continuidad funcional y canal público
PASS el inventario de un centro sigue operando dentro de la comuna | esperado: 200 | obtenido: 200
PASS el registro de residentes sigue operando dentro de la comuna | esperado: 200 | obtenido: 200
PASS las prioridades de necesidades siguen operando dentro de la comuna | esperado: 200 | obtenido: 200
PASS el mapa público muestra exactamente los centros activos | esperado: 9 | obtenido: 9
PASS el mapa público cubre todas las comunas con centros activos | esperado: 4 | obtenido: 4
PASS el mapa público expone exactamente los campos autorizados, y ninguno más | esperado: exacto | obtenido: exacto

Alta de una municipalidad nueva
PASS el Super Administrador da de alta una comuna sobre la instancia en marcha | esperado: creada | obtenido: creada
PASS la comuna nueva parte sin centros propios | esperado: 0 | obtenido: 0
PASS su correlativo de centros parte de cero | esperado: 0 | obtenido: 0
PASS su administrador queda atado solo a esa comuna | esperado: 1 | obtenido: 1
PASS y la comprobación no deja rastro en la base | esperado: 0 | obtenido: 0

Higiene de sesión
PASS no quedan transacciones abiertas sin actividad | esperado: 0 | obtenido: 0
PASS no hubo ningún intento de envío de correo | esperado: 0 | obtenido: 0

=====
TODO OK - 87 comprobaciones
```

Figura 29: Salida de scripts/validar_multitenant.sh: 87 de 87 comprobaciones con resultado PASS. Fuente:

Elaboración propia.