

**UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAISO – CHILE**



**EVALUACIÓN DE CALIDAD DE SOFTWARE SEGÚN EL
MODELO ISO/IEC 25010: CASO APPCOPIO**

Tito Vicente Orellana Barra

**MEMORIA PARA OPTAR AL TÍTULO
DE INGENIERIA CIVIL INFORMATICA**

Profesor Guía: Mauricio Figueroa

Resumen: Chile es un país expuesto a desastres naturales recurrentes, cuya gestión post-emergencia a nivel municipal depende de herramientas tecnológicas fragmentadas. AppCopio es una aplicación web progresiva diseñada para centralizar la administración de albergues y centros de acopio en tiempo real, pero su desarrollo no contó con un proceso formal de aseguramiento de calidad. Este trabajo evalúa la calidad de AppCopio utilizando el modelo ISO/IEC 25010, analizando seis características: fiabilidad, capacidad de interacción, adecuación funcional, seguridad, eficiencia de desempeño y flexibilidad. Se diseñó y ejecutó un plan de pruebas basado en ISO/IEC 25040, combinando análisis estático del código, auditorías automatizadas y prueba con una usuaria representativa del municipio de Valparaíso. Los resultados demuestran que AppCopio posee un rendimiento sobresaliente y una sincronización offline, pero presenta vulnerabilidades críticas en el control de acceso, dependencias con fallos de seguridad conocidos y un acoplamiento rígido a una sola municipalidad. Se concluye que el sistema es un prototipo funcional viable, pero requiere una refactorización estructural antes de su despliegue con datos reales. Se propone un plan de mejoras priorizado en tres fases para avanzar hacia un nivel de calidad compatible con un despliegue institucional seguro y escalable.

Palabras Clave: ISO 25010, Calidad de Software, Gestión de Emergencias, Fiabilidad, Usabilidad, AppCopio.



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: EVALUACIÓN DE CALIDAD DE SOFTWARE SEGÚN EL MODELO ISO/IEC 25010: CASO APPCOPIO

Nombre del candidato(a): Tito Vicente Orellana Barra

Carrera / Grado: Ingeniería Civil Informática

Campus: Casa Central

Departamento: Departamento de Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Mauricio Figueroa Colarte, 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 periodo 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: 27/07/2026

Firma:

Estudiante o Candidato(a):

Fecha: 27/07/2026

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.

1 Introducción

Chile es un país caracterizado por una alta frecuencia de eventos catastróficos, donde incendios, terremotos y aluviones impactan regularmente a la población. En este escenario, la capacidad de respuesta de las autoridades, específicamente de las municipalidades y sus equipos de emergencia, es vital para mitigar el impacto en la comunidad. La literatura académica respalda este rol fundamental, señalando que los gobiernos locales constituyen la primera línea de respuesta y gestión del riesgo debido a su vinculación directa e inmediata con el territorio afectado [14]. Sin embargo, la gestión actual de la ayuda post-emergencia presenta deficiencias tecnológicas significativas.

Actualmente, muchos equipos municipales dependen de herramientas ofimáticas genéricas y desconectadas, como Google Workspace u Office365, para coordinar tareas críticas. Esta fragmentación de la información, sumada a la falta de canales oficiales centralizados, genera duplicidad de esfuerzos y retrasos en la toma de decisiones. Diagnósticos institucionales advierten que la coordinación ante emergencias a nivel local padece de una escasa interoperabilidad digital, provocando que la logística dependa de metodologías manuales en lugar de datos integrados en tiempo real [15]. Asimismo, evaluaciones a nivel nacional enfatizan la urgencia de dotar a los municipios de plataformas técnicas especializadas, ya que la excesiva centralización de los sistemas actuales limita la autonomía y agilidad de los equipos en terreno [16].

Como respuesta a esta necesidad se construyó AppCopio, un piloto municipal concebido como una solución tecnológica integral diseñada para centralizar la gestión de albergues y centros de acopio, facilitando la coordinación en tiempo real entre la municipalidad, los encargados y la comunidad. No obstante, desarrollar una aplicación para este contexto implica desafíos que van más allá de la funcionalidad básica: el software debe operar en condiciones de alta incertidumbre, estrés humano y degradación de la infraestructura tecnológica.

El problema central radica en que, hasta el momento, el desarrollo de AppCopio no ha contado con un proceso formal de aseguramiento de calidad. Al estar destinado a gestionar emergencias reales, existe una incertidumbre crítica respecto a cómo se comportará ante escenarios de conectividad intermitente, si su interfaz es lo suficientemente intuitiva para voluntarios bajo presión, o si los datos sensibles de los damnificados están debidamente protegidos. Un fallo en su funcionamiento puede tener consecuencias humanas directas.

Esta falta de evaluación rigurosa impide validar si el software es lo suficientemente robusto, fiable y seguro para ser adoptado por distintas comunas o ser escalado a nivel nacional. Para abordar esta problemática, el presente trabajo propone realizar una auditoría técnica y funcional focalizada utilizando como marco de referencia la norma ISO/IEC 25010, la cual provee un modelo que permite descomponer el software en características medibles y objetivas para identificar vulnerabilidades y oportunidades de mejora.

Considerando la naturaleza crítica de AppCopio y en concordancia con los requerimientos del sistema, se han seleccionado y priorizado las siguientes características de calidad para esta evaluación:

- **Fiabilidad:** Debido a la inestabilidad de las redes en emergencias, se evaluará la tolerancia a fallos, la disponibilidad y, crucialmente, la recuperabilidad de los datos tras procesos de sincronización offline/online.
- **Capacidad de Interacción (Usabilidad):** Dado que los usuarios (voluntarios, encargados) operan bajo presión, se medirá la aprendizabilidad, la operabilidad de la interfaz, la protección frente a errores de usuario y la inclusividad para personas con distintos niveles de habilidad digital. Cabe destacar que la evaluación de este último punto corresponderá a una auditoría automatizada preliminar de accesibilidad, y no a una validación exhaustiva bajo estándares WCAG con usuarios con discapacidad.
- **Adecuación Funcional:** Se verificará la completitud funcional para cubrir las necesidades de los tres perfiles de usuario (DIDECO, Encargados, Comunidad) y la corrección funcional de la información mostrada en tiempo real.
- **Seguridad:** Se auditará la confidencialidad (roles y permisos), la integridad de los datos ante posibles corrupciones y la autenticidad de los usuarios críticos.
- **Eficiencia de Desempeño:** Se analizarán los tiempos de respuesta, enfocándose en el comportamiento temporal durante la carga y sincronización de datos.
- **Flexibilidad:** Se evaluará la escalabilidad y adaptabilidad del sistema para determinar su viabilidad de implementación a nivel nacional.

A través de esta evaluación, se busca obtener un diagnóstico preciso del estado actual del software y generar un plan de mejoras priorizado que contribuya a fortalecer la calidad de AppCopio para la gestión de emergencias.

1.1 Objetivos

El objetivo general es evaluar la calidad del software AppCopio conforme a los criterios definidos en la norma ISO/IEC 25010, identificando oportunidades de mejora en escenarios representativos de uso. Este se compone de los siguientes objetivos específicos:

- Diseñar un plan para medir la calidad del software en base a los criterios relevantes de ISO/IEC 25010.
- Ejecutar el plan de medición de calidad.
- Analizar resultados para detectar oportunidades de mejora.
- Crear un plan de mejoras de calidad para AppCopio.

1.2 Estructura del informe

En el capítulo N°2 se presentan las bases teóricas relacionadas con la gestión de emergencias y la evaluación de calidad de software bajo la norma ISO/IEC 25000. En el capítulo N°3 se describe el caso de estudio AppCopio. En el capítulo N°4 se diseña el plan de evaluación, definiendo la metodología, las métricas y las pruebas específicas. A continuación, en el capítulo N°5 se exponen los resultados obtenidos tras la ejecución de dichas pruebas. En el capítulo N°6 se realiza un análisis crítico de los hallazgos para diagnosticar el estado del software. Posteriormente, en el capítulo N°7 se presenta el plan de futuras mejoras priorizado y, finalmente, el capítulo N°8 contiene las conclusiones del trabajo realizado.

2 Marco Conceptual

2.1 Contexto: Gestión de Emergencias en Chile

Chile posee una institucionalidad dedicada a la Gestión del Riesgo de Desastres (GRD), encabezada por SENAPRED (ex-ONEMI) y operativizada a nivel local por las municipalidades a través de unidades como DIDECO (Dirección de Desarrollo Comunitario) [17]. En la fase de "Respuesta", la coordinación de albergues y centros de acopio es crítica. Sin embargo, la literatura e informes institucionales indican que la gestión de información en esta etapa suele ser fragmentada, dependiendo de herramientas ofimáticas no especializadas que dificultan la toma de decisiones en tiempo real y la trazabilidad de la ayuda [15].

2.2 Calidad del Software

En la ingeniería de software, la calidad se define no como un atributo absoluto, sino como el grado en que un sistema, componente o proceso cumple con los requisitos especificados y las expectativas del cliente o usuario. Según Pressman, la calidad del software es un proceso eficaz que se aplica para crear un producto útil y que aporta valor medible a quienes lo utilizan [1].

El aseguramiento de la calidad (SQA) implica un patrón planificado de acciones necesarias para proporcionar la confianza adecuada de que el producto satisface los requerimientos técnicos y funcionales. Dado que la valoración de calidad puede ser subjetiva, el uso de estándares internacionales es fundamental para establecer una línea base objetiva de evaluación.

2.3 Modelos de Calidad

Un modelo de calidad es un marco de trabajo que descompone el concepto abstracto de "calidad" en un conjunto de características y subcaracterísticas medibles. Históricamente han destacado modelos como McCall (1977), Boehm (1978) e ISO 9126 (1991). Actualmente una referencia ampliamente utilizada es la familia ISO/IEC 25000, conocida como SQuaRE (System and Software Quality Requirements and Evaluation), por ser la evolución más completa y detallada para la ingeniería de software moderna [2].

2.4 ISO/IEC 25000 (SQuaRE)

La norma ISO/IEC 25000 tiene como objetivo principal crear un marco de trabajo común para evaluar la calidad del producto de software. Esta familia se divide en:

- ISO/IEC 2500n: División de gestión de calidad.
- ISO/IEC 2501n: División de modelos de calidad (donde reside la ISO 25010).
- ISO/IEC 2502n: División de medición de calidad.
- ISO/IEC 2503n: División de requisitos de calidad.
- ISO/IEC 2504n: División de evaluación de calidad.

2.5 Modelo de Calidad del Producto ISO/IEC 25010:2023

Este estándar define un modelo de calidad del producto compuesto por nueve características principales [3]. Para efectos de esta memoria, y considerando la naturaleza crítica de AppCopio como sistema de gestión de emergencias, se han seleccionado las siguientes características como prioritarias para la evaluación:

- **Fiabilidad:** Capacidad de un sistema para desempeñar sus funciones especificadas bajo condiciones determinadas durante un período de tiempo. En el contexto de AppCopio, esto es crítico debido a la necesidad de operar en zonas con infraestructura de comunicaciones degradada (tolerancia a fallos y disponibilidad).
- **Capacidad de Interacción (Usabilidad):** Grado en que un producto permite a los usuarios alcanzar sus objetivos con efectividad, eficiencia y satisfacción. Dado que los voluntarios operan bajo estrés y con distintos niveles de alfabetización digital, la "protección frente a errores de usuario" y la "aprendizabilidad" son vitales.
- **Adecuación Funcional:** Capacidad del producto para proporcionar funciones que satisfacen las necesidades declaradas e implícitas. Se centra en la completitud funcional y la corrección de los datos.
- **Seguridad:** Protección de la información y los datos para que personas no autorizadas no puedan leerlos o modificarlos. Incluye la confidencialidad (roles), integridad y autenticidad.
- **Eficiencia de Desempeño:** Comportamiento temporal y utilización de recursos del sistema bajo condiciones específicas. Es crucial para garantizar tiempos de carga rápidos en dispositivos utilizados en terreno.
- **Flexibilidad:** Capacidad del producto para adaptarse a cambios en sus requisitos o entorno. Se evalúa la escalabilidad para determinar si el sistema puede expandirse desde un nivel comunal a uno nacional.

2.6 Proceso de Evaluación ISO/IEC 25040

La norma ISO/IEC 25040 establece el proceso de referencia para llevar a cabo la evaluación de la calidad del software [4]. Este proceso consta de cinco actividades principales que guiarán la metodología de este trabajo:

- Establecer los requisitos de evaluación: Definir el alcance, los objetivos y las métricas de calidad.
- Especificar la evaluación: Seleccionar las métricas e instrumentos (herramientas de prueba).
- Diseñar la evaluación: Definir el plan de pruebas y los procedimientos logísticos.
- Ejecutar la evaluación: Realizar las mediciones y obtener datos.

- Concluir la evaluación: Analizar resultados y generar informes de mejora.

2.7 Arquitecturas Web para Alta Disponibilidad (PWA)

En el contexto de gestión de emergencias, la continuidad operativa ante la pérdida de conectividad es un requisito no funcional crítico. Para abordar esto, la industria ha adoptado el modelo de Aplicaciones Web Progresivas (PWA). Una PWA es una aplicación web que utiliza capacidades modernas del navegador para ofrecer una experiencia de usuario similar a la de una aplicación nativa, incluyendo la capacidad de funcionar offline [8].

2.7.1 Service Workers y Gestión de Red

El componente núcleo de una PWA es el Service Worker, un script que el navegador ejecuta en segundo plano, separado de la página web principal. Técnicamente, actúa como un proxy de red programable que se sitúa entre la aplicación web, el navegador y la red. Su función principal en la fiabilidad del sistema es interceptar las peticiones de red y decidir cómo responder a ellas. Esto permite implementar estrategias de caché avanzadas, tales como:

- Cache-First: Servir recursos estáticos (imágenes, scripts) directamente desde el dispositivo para una carga instantánea.
- Network-First: Intentar obtener datos frescos del servidor, y si falla, recurrir a una copia guardada localmente. Esta tecnología es la que permite que el software siga siendo operativo visualmente (Tolerancia a fallos) incluso cuando el servidor es inalcanzable.

2.7.2 Persistencia de Datos en el Cliente (IndexedDB)

Para garantizar la "Recuperabilidad" de la información generada sin conexión, el almacenamiento local simple (LocalStorage) es insuficiente debido a sus límites de tamaño y naturaleza síncrona. En su lugar, se utilizan bases de datos transaccionales en el navegador como IndexedDB. Este mecanismo permite implementar el patrón de Sincronización en Segundo Plano (Background Sync). Cuando un usuario realiza una acción offline (ej. registrar un damnificado), la aplicación no envía la petición HTTP inmediatamente, sino que la almacena en una "Cola de Mutaciones" (Mutation Queue) dentro de IndexedDB. Un proceso observador detecta cuándo se restablece la conexión y procesa la cola, enviando los datos al servidor secuencialmente para asegurar la integridad referencial [18].

2.8 Métricas de Rendimiento Web (Web Performance)

El rendimiento de una aplicación web no es solo una cuestión de velocidad técnica, sino de percepción de usuario. Google ha estandarizado un conjunto de métricas conocidas como Core Web Vitals, que son esenciales para evaluar la experiencia de usuario en condiciones de red degradadas [6].

2.8.1 First Contentful Paint (FCP)

El FCP mide el tiempo transcurrido desde que la página comienza a cargar hasta que cualquier parte del contenido de la página se renderiza en la pantalla [7]. Es un indicador crítico de la velocidad de carga percibida. Un FCP bajo (idealmente menor a 1.8 segundos) confirma al usuario que la solicitud está siendo procesada, reduciendo la tasa de abandono en situaciones de estrés.

2.8.2 Concepto de Bundle Size (Empaquetado)

En aplicaciones modernas de página única (SPA), todo el código JavaScript necesario para ejecutar la aplicación se agrupa en archivos conocidos como Bundles. El tamaño de este empaquetado (Bundle Size) impacta directamente en la eficiencia: un archivo demasiado grande no solo tarda más en descargarse en redes lentas, sino que bloquea el hilo principal del procesador mientras el navegador lo analiza, compila y ejecuta [19]. Mantener un tamaño reducido es vital para garantizar la operatividad en dispositivos de escritorio o portátiles con hardware limitado.

2.9 Métricas de Calidad Interna del Software

La "Flexibilidad" y "Mantenibilidad" del software se evalúan mediante el análisis de su estructura interna, sin necesidad de ejecutarlo.

2.9.1 Análisis Estático y Complejidad Ciclomática

El análisis estático es el examen del código fuente mediante herramientas automatizadas para detectar patrones de error y deuda técnica. Una de las métricas más relevantes es la Complejidad Ciclomática, que mide el número de caminos linealmente independientes a través del código fuente de un programa [5]. Una complejidad alta indica que una función tiene demasiada lógica condicional (if, else, loops), lo que incrementa exponencialmente la probabilidad de defectos y dificulta su escalabilidad o modificación futura.

2.9.2 Codificación Rígida (Hardcoding)

La codificación rígida se refiere a la práctica de incrustar datos de entrada directamente en el código fuente del programa, en lugar de obtenerlos de fuentes externas o configuración. En sistemas escalables, la presencia de valores "quemados" (ej. coordenadas geográficas fijas o nombres de ciudades específicos) representa una violación al principio de "Adaptabilidad", ya que impide el despliegue del software en nuevos territorios sin modificar y recompilar el código.

2.9.3 Arquitecturas Multi-Tenant

Para que un sistema de software sea verdaderamente escalable, su diseño de base de datos debe soportar el crecimiento horizontal. Una arquitectura Multi-Tenant es un patrón de diseño donde una única instancia de la aplicación y la base de datos sirve a múltiples clientes u organizaciones independientes [20]. En el contexto municipal, esto implica estructurar el Modelo Entidad-Relación de manera que los datos de distintas comunas compartan la misma infraestructura, pero se mantengan lógicamente aislados mediante claves foráneas (ej. identificadores de municipalidad), evitando la necesidad de desplegar y mantener servidores físicos separados para cada nueva organización que adopte el sistema.

2.10 Seguridad en el Desarrollo Web Moderno

2.10.1 Gestión de Vulnerabilidades en Dependencias

El desarrollo de software moderno se basa fuertemente en la reutilización de código a través de librerías de terceros (paquetes). Sin embargo, esto introduce riesgos de seguridad heredados. La gestión de vulnerabilidades implica el escaneo automatizado del árbol de dependencias para identificar componentes con fallos de seguridad conocidos (CVE - Common Vulnerabilities and Exposures), un diccionario de vulnerabilidades mantenido a nivel mundial para estandarizar la identificación de amenazas cibernéticas [21]. Esto asegura que la plataforma no exponga vectores de ataque conocidos.

2.10.2 Control de Acceso Basado en Roles (RBAC)

Para cumplir con el criterio de "Confidencialidad", las aplicaciones deben implementar mecanismos de autorización robustos. El modelo RBAC (Role-Based Access Control) restringe el acceso a la red basándose en el rol de cada usuario dentro de la organización [10]. En arquitecturas web stateless, esto se implementa comúnmente verificando los permisos contenidos dentro de un token de sesión (como JWT) antes de renderizar una vista o procesar una petición en el servidor.

2.11 Capacidad de Interacción y Accesibilidad Web

En el desarrollo de interfaces modernas, la "Capacidad de Interacción" (o Usabilidad) determina el éxito en la adopción del sistema por parte de los usuarios finales, especialmente en contextos de emergencia donde no hay tiempo para capacitaciones extensas. Dos componentes teóricos críticos en este ámbito son la aprendizabilidad y la inclusividad.

2.11.1 Aprendizabilidad en Sistemas Críticos

La aprendizabilidad se refiere al grado en que un usuario sin experiencia previa puede comprender y utilizar el sistema de manera efectiva en su primer intento [22]. En la ingeniería de software, esto no se mide mediante métricas estáticas, sino a través de evaluaciones empíricas, tales como la observación directa o grabada de usuarios representativos del dominio (ej. trabajadores sociales o de emergencia) ejecutando flujos de trabajo críticos. Minimizar el tiempo de descubrimiento de las funciones es vital para reducir la fricción operativa.

2.11.2 Pautas de Accesibilidad Web (WCAG)

La inclusividad tecnológica está estandarizada internacionalmente a través de las Pautas de Accesibilidad al Contenido en la Web (WCAG) [9]. Estas pautas definen criterios técnicos rigurosos, como el contraste mínimo de colores, la navegabilidad mediante periféricos alternativos y el uso de atributos semánticos (etiquetas ARIA), para garantizar que la plataforma sea operable por personas con diversas capacidades visuales o motrices. El cumplimiento de estas normativas se cuantifica mediante herramientas de auditoría automatizada que simulan el análisis del Árbol de Accesibilidad del navegador.

2.12 Trabajos Previos

La evaluación de calidad de software mediante el modelo ISO/IEC 25010 ha sido aplicada en diversos contextos académicos e industriales. A continuación, se presentan trabajos previos relevantes que justifican y enmarcan la metodología adoptada en esta memoria.

2.12.1 Métricas de Código para Mantenibilidad

En [11] se realizó un estudio comparativo de herramientas de análisis estático para medir métricas básicas de mantenibilidad, bajo los acuerdos de la norma ISO/IEC 9126. El trabajo recopiló métricas de diversas fuentes, clasificándolas en subcaracterísticas de analizabilidad, cambiabilidad, estabilidad y testabilidad. Sus hallazgos son directamente aplicables a AppCopio: la Complejidad Ciclomática y las líneas de código por función emergen como los indicadores más robustos y ampliamente soportados por herramientas automatizadas.

2.12.2 Evaluación de Calidad en Aplicaciones Web con ISO/IEC 25010

En [12] se aplicó el modelo ISO/IEC 25010 para evaluar un sistema de información web en el sector salud, contexto análogo al de AppCopio en cuanto a la criticidad operativa y la diversidad de usuarios. Los autores

seleccionaron las características de usabilidad, fiabilidad y seguridad, argumentando que en sistemas de uso crítico estas dimensiones determinan de forma directa la adopción y la continuidad operativa. Sus resultados mostraron que la ausencia de pruebas de tolerancia a fallos constituye el riesgo más significativo para la continuidad del servicio, conclusión que refuerza la priorización de Fiabilidad en el presente trabajo [12].

2.12.3 Aplicación de ISO/IEC 25010 en Tecnologías Emergentes

En [13] se investigó la aplicación del modelo ISO/IEC 25010 en dos aplicaciones desarrolladas con herramientas de realidad aumentada (Vuforia y Wikitude). El estudio se enfocó en la Eficiencia de Desempeño, subcaracterísticas de comportamiento temporal y utilización de recursos, con el objetivo de comparar ambas plataformas previo a la selección de una de ellas para producción. Sus métricas (tiempo de respuesta, tiempo de espera y uso de memoria) y sus umbrales de aceptación constituyeron un precedente metodológico adoptado parcialmente en este trabajo para las pruebas de Rendimiento de Carga Inicial y Peso del Empaquetado Inicial.

3 Diseño del Plan de Evaluación

Este capítulo tiene como propósito definir el alcance, la metodología y los objetivos específicos de la evaluación de calidad que se aplicará al software AppCopio, describiendo en detalle la entidad objetivo y las características críticas seleccionadas.

3.1 Metodología

Para estructurar la evaluación de calidad de AppCopio, se aplicaron las directrices del proceso de evaluación definido en la norma ISO/IEC 25040. Su aplicación en este proyecto se traduce en las siguientes fases metodológicas:

1. **Establecer los requisitos de evaluación:** Identificar a AppCopio como la entidad objetivo y determinar que la evaluación se centrará en seis características de la norma ISO/IEC 25010, debido a la naturaleza crítica de la gestión de emergencias.
2. **Especificar la evaluación:** Seleccionar métricas técnicas (ej. Core Web Vitals, Complejidad Ciclométrica) y definir los umbrales de aceptación para cada una, adaptándolos a un entorno web de escritorio.
3. **Diseñar la evaluación:** Establecer los planes de prueba, definiendo si serán ejecutados mediante herramientas de análisis estático/dinámico automatizado o mediante validación manual simulando el uso real.
4. **Ejecutar la evaluación:** Aplicar los instrumentos de medición sobre un entorno local de AppCopio compilado en modo producción y recopilar los datos.
5. **Concluir la evaluación:** Analizar los resultados obtenidos para elaborar un diagnóstico y un plan de futuras mejoras.

3.2 Descripción del Caso de Estudio: AppCopio

El software AppCopio fue desarrollado para centralizar la gestión de albergues y coordinar la ayuda humanitaria en situaciones post-catástrofe. El sistema opera como una plataforma web que permite la interacción y el flujo de información en tiempo real entre la Municipalidad, los encargados de centros desplegados en terreno y la comunidad afectada.

Para dar soporte a esta operatividad, el sistema está construido bajo una arquitectura cliente-servidor, componiéndose de los siguientes elementos técnicos:

- **Frontend y Arquitectura Cliente:** La interfaz de usuario está desarrollada utilizando la librería React 19 y TypeScript, empleando Material UI v7 para el diseño responsivo de componentes y Vite como herramienta de empaquetado.

- **Backend y Base de Datos:** La lógica de negocio expone una API RESTful construida sobre un entorno de ejecución Node.js utilizando el framework Express. La persistencia de la información es administrada por un motor relacional PostgreSQL 15.
- **Roles de Usuario (RBAC):** La seguridad y operatividad del sistema se estructura mediante un Control de Acceso Basado en Roles que clasifica a los usuarios en cuatro perfiles principales: el "Administrador" (equipo de emergencia social de DIDECO con control global), el "Trabajador Municipal" (operador y encargado de centro responsable de la gestión en terreno), el "Contacto Ciudadano" (perfil preventivo vinculado a un inmueble o centro específico, como un dirigente vecinal que actúa como enlace local y recibe notificaciones) y el "Usuario Público" (cualquier persona de la comunidad, incluyendo damnificados, voluntarios o donantes, que accede para consumir la información del mapa interactivo en tiempo real).
- **Módulos Principales:** Las funcionalidades del software se dividen en módulos clave para la emergencia: un módulo de Gestión de Centros (creación, activación y georreferenciación), un módulo de Inventario (control de ingresos/salidas de insumos), un módulo de Registro de Damnificados (mediante la Ficha FIBE), y un buzón de Notificaciones y Actualizaciones.
- **Despliegue:** Si bien su arquitectura está diseñada para un despliegue en infraestructuras de nube pública, para efectos del control de calidad y auditoría del presente trabajo, el sistema se evalúa desplegado en un entorno local compilado en modo producción (Local Production Build).

Debido al contexto crítico en el que debe ser utilizado, donde la infraestructura de telecomunicaciones suele ser inestable, el software fue construido con un enfoque en la resiliencia operativa. Su característica principal es la capacidad de mantener la continuidad del servicio frente a la pérdida de conectividad a internet. Para lograr esto, la aplicación monitorea de forma autónoma el estado de la red; ante una caída, transiciona a un modo de funcionamiento offline que retiene la interfaz gráfica y permite a los usuarios seguir registrando datos. Esta información se almacena localmente en el dispositivo y se sincroniza de forma automática con el servidor central una vez que la conexión se restablece.

3.3 Objetivos de Evaluación AppCopio

El estándar ISO/IEC 25010 define nueve características de calidad, sin embargo, para esta evaluación se han descartado tres de ellas por razones de prioridad y contexto operativo.

Debido a que AppCopio es una herramienta de coordinación informativa y no controla maquinaria ni presenta riesgos directos para la integridad física de las personas, la característica de Protección (Safety) queda descartada. Por otro lado, dado que el sistema actualmente no coexiste ni intercambia datos con otros softwares (y su única integración visual con Google Maps fue validada en el desarrollo), la Compatibilidad quedó excluida del alcance de la evaluación. Finalmente, aunque se cuenta con acceso al código fuente, la Mantenibilidad no fue considerada en este plan, priorizando el esfuerzo en evaluar áreas con un impacto operativo más directo en terreno (como la fiabilidad y usabilidad), dejándola como recomendación para auditorías futuras.

Dada la criticidad del sistema, un fallo operativo en AppCopio puede significar la pérdida de información vital o el retraso en la entrega de ayuda. Por ello, la evaluación se centró en solo seis características. A continuación, se justifican los objetivos específicos para cada una.

3.3.1 Objetivos de Fiabilidad

Esta evaluación se enfoca en el modo offline y los mecanismos de sincronización del sistema. Considerando que en un escenario de emergencia la infraestructura de telecomunicaciones suele colapsar, el objetivo es validar empíricamente que la arquitectura PWA de AppCopio sea tolerante a fallos de red, asegurando que el usuario

pueda seguir ingresando datos de manera local y que el mecanismo de IndexedDB recupere y sincronice el 100% de esta información al reconectar, buscando asegurar que ningún registro de ayuda humanitaria se pierda.

3.3.2 Objetivos de Capacidad de Interacción (Usabilidad)

Esta dimensión se centra en el comportamiento de un usuario representativo en escenarios simulados y en la protección de los datos frente a errores operacionales. Dado que los operadores del sistema en terreno (encargados de albergues y voluntarios) a menudo no poseen formación técnica avanzada y trabajarán en contextos de alta presión, el objetivo es evaluar la operabilidad de la interfaz de escritorio y verificar que el sistema posea mecanismos de prevención de errores estrictos (como impedir el registro de stock o inventarios negativos), evitando que un error humano comprometa la logística de la municipalidad.

3.3.3 Objetivos de Adecuación Funcional

La evaluación pone su foco en la cobertura de las necesidades críticas para DIDECO y la comunidad. Partiendo de la premisa de que el software debe hacer exactamente aquello para lo que fue diseñado, el objetivo es comprobar la completitud del sistema frente a las Historias de Usuario esenciales definidas en las etapas tempranas del proyecto, garantizando que las tres vistas principales (gestión global para DIDECO, ingreso de datos para Encargados y mapa público para la Comunidad) expongan información matemáticamente correcta en tiempo real.

3.3.4 Objetivos de Seguridad

Esta característica se orienta al control de roles y a la protección de datos sensibles. Debido a que AppCopio maneja información altamente confidencial, como los datos personales de los damnificados recopilados en la Ficha FIBE (Ficha Básica de Emergencia, instrumento oficial del Ministerio de Desarrollo Social para catastrar e identificar las necesidades de las familias afectadas tras una catástrofe) [23], el objetivo es auditar la confidencialidad verificando que el Control de Acceso Basado en Roles (RBAC) funciona correctamente, impidiendo que un perfil público o de voluntario acceda a rutas administrativas, además de verificar la ausencia de vulnerabilidades críticas en las librerías base del software.

3.3.5 Objetivos de Eficiencia de Desempeño

El foco de esta medición se establece en los tiempos de carga y en la utilización de los recursos de red. Aunque la evaluación está acotada a navegadores de escritorio, las conexiones a internet post-catástrofe suelen ser lentas e inestables, por lo que el objetivo es medir las métricas vitales de rendimiento web (Core Web Vitals) para verificar que el empaquetado inicial (Bundle Size) sea ligero y que la vista principal cargue rápidamente, evitando bloqueos operativos.

3.3.6 Objetivos de Flexibilidad

Esta dimensión se enfoca en la viabilidad del escalamiento de la plataforma a nivel nacional. El objetivo es evaluar estáticamente la calidad interna del código fuente a través de su complejidad ciclomática y asegurar su adaptabilidad, verificando la ausencia de variables geográficas fijas (*hardcoded*) en la lógica de negocio que pudiesen amarrar el funcionamiento del software exclusivamente a la comuna de Valparaíso utilizada para el levantamiento de requisitos original.

4 Diseño de la Evaluación

En este capítulo se detalla el plan de evaluación de calidad diseñado para el software AppCopio. Se define la metodología híbrida utilizada (combinando herramientas automáticas y validación manual), las métricas específicas seleccionadas para cada característica de la norma ISO/IEC 25010 y los umbrales de aceptación, enfocándose en su operación como página web.

4.1 Pruebas de Fiabilidad

Para el área de fiabilidad, el diseño contempló evaluar tres casos relacionados con la operación de la plataforma ante inestabilidad de red: la sincronización de datos post-desconexión, la tolerancia a fallos en vistas no cacheadas y la sensibilidad del sistema para detectar caídas de internet mediante ping continuo.

4.1.1 Integridad de Sincronización (Recuperabilidad)

- **Objetivo:** Verificar que los datos creados sin conexión se almacenen localmente y se envíen al servidor sin pérdida al reconectar.
- **Procedimiento:**
 1. Autenticarse en la plataforma web desde un computador con conexión estable.
 2. Abrir las herramientas de desarrollador (DevTools) del navegador y activar la opción "Offline" en la pestaña Network.
 3. Completar el formulario y crear 3 registros nuevos de usuarios.
 4. Dirigirse a la pestaña Application > IndexedDB y verificar visualmente que los 3 registros se encuentren encolados.
 5. Desactivar la opción "Offline" para simular el retorno de la conexión.
 6. Monitorear la consola de red y consultar la base de datos PostgreSQL del servidor para confirmar que los 3 registros se insertaron correctamente sin duplicidad.
- **Criterio de Aceptación:** 100% de los datos se sincronizan sin pérdida ni duplicidad.

4.1.2 Navegación Offline en Vistas No Cacheadas (Tolerancia a Fallos)

- **Objetivo:** Comprobar si el Service Worker (arquitectura PWA) es capaz de servir la interfaz de una ruta que el usuario aún no ha visitado.
- **Procedimiento:**
 1. Limpiar completamente los datos de navegación y caché del navegador (Clear Site Data).
 2. Iniciar sesión y cargar únicamente la página del Home inicial.
 3. Desconectar físicamente el computador de la red Wi-Fi/Ethernet.
 4. Hacer clic en una sección interna no visitada previamente (ej. "Gestión de Inventario").
 5. Observar el comportamiento del navegador.
- **Criterio de Aceptación:** La vista debe cargar su estructura base (App Shell) sin mostrar el error genérico de "Sin conexión a internet" del navegador.

4.1.3 Sensibilidad de Detección de Red (Disponibilidad)

- **Objetivo:** Validar que el mecanismo de polling en segundo plano detecte la caída de la conexión en el tiempo esperado según su configuración.
- **Procedimiento:**
 1. Ingresar a AppCopio con conexión a internet estable.
 2. Utilizar DevTools para bloquear específicamente el dominio de monitoreo (google.com/favicon.ico), manteniendo el resto de la red activa.
 3. Iniciar un cronómetro en el momento exacto del bloqueo.

4. Observar la interfaz gráfica y detener el cronómetro cuando el sistema notifique visualmente el estado Offline.
- **Criterio de Aceptación:** El sistema debe transicionar a modo offline en un máximo de 20 segundos (tolerando 2 intentos fallidos de 10 segundos).

Las condiciones de aprobación se ven en la Tabla 1.

Tabla 1. Pruebas Fiabilidad
Fuente. Elaboración propia

Prueba	Condición Aprobación	Subcaracterística 25010	ISO/IEC
Integridad de Sincronización	Se aprueba si el 100% de los datos se sincronizan sin pérdida ni duplicidad.	Recuperabilidad	
Navegación Offline en Vistas No Cacheadas	Se aprueba si la vista carga correctamente y permite su uso básico.	Tolerancia a Fallos	
Sensibilidad de Detección de Red	Transición a offline $\leq$ 20 segundos.	Disponibilidad	

4.2 Pruebas de Capacidad de Interacción

Las pruebas se diseñaron para evaluar la usabilidad de la interfaz de escritorio. Para obtener métricas fidedignas de aprendizabilidad, se contempló el uso de material audiovisual de pruebas preliminares grabadas con actores reales del dominio.

4.2.1 Usabilidad con Usuario Representativo (Aprendizabilidad)

- **Objetivo:** Medir el tiempo de ejecución y la facilidad con la que un usuario no técnico, representativo del dominio operativo, aprende a utilizar las funcionalidades clave del sistema en su primer acercamiento.
 - **Procedimiento:**
 1. Solicitar al usuario representativo de DIDEKO realizar una batería de 8 tareas críticas dentro de la plataforma, sin otorgar asistencia técnica directa.
 2. Cronometrar el tiempo desde la entrega de la instrucción hasta la ejecución exitosa de cada tarea.
 3. Observar y registrar si existen titubeos prolongados, errores lógicos o si se requiere asistencia externa.
 4. Al finalizar cada tarea, aplicar un instrumento de evaluación (Cuestionario de Satisfacción) solicitando al usuario calificar su percepción de "Tiempo" y "Facilidad" utilizando una escala de Likert de 1 a 7 (donde 1 es muy insatisfecho/difícil y 7 es muy satisfecho/fácil).
 - **Criterio de Aceptación:** El usuario logra ubicar las funciones y completar los flujos en menos de 2 minutos por tarea sin abandonarla, obteniendo calificaciones de satisfacción iguales o superiores a 5/7.
- Nota metodológica:** Se exceptúan de la restricción de tiempo de 2 minutos aquellas tareas cuya naturaleza de negocio exija el ingreso masivo de datos (formularios extensos). En estos casos, la prueba se considerará exitosa si el usuario completa el flujo sin requerir asistencia técnica y sin presentar bloqueos de comprensión.

4.2.2 Operabilidad en Entorno Desktop (Operabilidad)

- **Objetivo:** Verificar que la interfaz se adapte correctamente a distintos tamaños de monitores de escritorio.
- **Procedimiento:**
 1. Abrir la plataforma en un navegador de escritorio.
 2. Ajustar la ventana a resoluciones de 1366x768 (Laptop estándar) y 1920x1080 (Monitor Full HD).
 3. Navegar por el mapa interactivo y las tablas de datos.
 4. Inspeccionar visualmente en búsqueda de errores de diagramación.
- **Criterio de Aceptación:** 0 elementos superpuestos, ocultos o desbordamientos horizontales indeseados.

4.2.3 Validación de Errores de Usuario (Protección)

- **Objetivo:** Confirmar que el sistema impide proactivamente que el estrés del usuario se traduzca en datos corruptos.
- **Procedimiento:**
 1. Ingresar al formulario de actualización de stock de un centro.
 2. Ingresar caracteres alfabéticos en el campo de "Cantidad".
 3. Ingresar un número negativo (ej. "-50").
 4. Intentar forzar el envío del formulario mediante clics repetidos.
- **Criterio de Aceptación:** El sistema bloquea el envío y despliega alertas visuales explícitas sobre el error.

4.2.4 Auditoría de Accesibilidad Web (Inclusividad)

- **Objetivo:** Validar técnicamente que la plataforma cumple con estándares básicos de legibilidad y contraste.
- **Procedimiento:**
 1. Abrir DevTools en Google Chrome.
 2. Ejecutar un reporte de Lighthouse seleccionando la categoría "Accessibility" para entorno Desktop.
 3. Extraer el puntaje final proporcionado por la herramienta.
- **Criterio de Aceptación:** Puntuación $\geq 85/100$.

Las condiciones de aprobación se ven en la Tabla 2.

Tabla 2. Métricas Capacidad de Interacción
Fuente. Elaboración propia

Métrica	Condición Aprobación	Subcaracterística ISO/IEC 25010
Usabilidad con Usuario Representativo	Complejidad sin asistencia técnica, con tiempo inferior a 2 minutos por tarea, salvo tareas extensas de ingreso masivo de datos, y satisfacción $\geq 5/7$.	Aprendizabilidad
Operabilidad en Entorno Desktop	0 errores de renderizado UI.	Operabilidad
Validación de Errores de Usuario	Bloqueo al 100% de datos ilógicos.	Protección frente a errores
Auditoría de Accesibilidad Web	Score Lighthouse $\geq 85/100$.	Inclusividad

4.3 Pruebas de Adecuación Funcional

Las pruebas se estructuraron para verificar empíricamente que el software cumple con las funciones vitales exigidas para la gestión de la emergencia.

4.3.1 Cobertura del Producto Mínimo Viable (Compleitud)

- **Objetivo:** Garantizar que las necesidades de los 3 actores (DIDECO, Encargados, Comunidad) estén cubiertas.
- **Procedimiento:**
 1. Extraer las Historias de Usuario catalogadas como "Esenciales" de la documentación del proyecto.
 2. Elaborar una lista de chequeo.
 3. Navegar por el sistema como administrador y verificar que cada historia de la lista pueda ser ejecutada en la plataforma real.
- **Criterio de Aceptación:** 100% de cumplimiento en las historias esenciales.

4.3.2 Exactitud Matemática de Inventarios (Corrección)

- **Objetivo:** Asegurar que los cálculos mostrados en tiempo real no presenten anomalías que afecten la logística.
- **Procedimiento:**
 1. Seleccionar un centro de acopio y anotar su stock actual de un insumo (ej. Agua = 100).
 2. Realizar transacción de ingreso sumando 50 unidades.
 3. Realizar transacción de salida restando 30 unidades.
 4. Calcular el valor esperado manualmente ($100 + 50 - 30 = 120$).
 5. Comparar el cálculo con el valor expuesto en la interfaz y en la base de datos.
- **Criterio de Aceptación:** 0 discrepancias numéricas.

Las condiciones de aprobación se ven en la Tabla 3.

Tabla 3. Métricas de Adecuación Funcional
Fuente. Elaboración propia

Métrica	Condición Aprobación	Subcaracterística 25010	ISO/IEC
Cobertura del Producto Mínimo Viable	100% Historias Esenciales presentes.	Compleitud Funcional	
Exactitud Matemática de Inventarios	Cálculos exactos sin anomalías.	Corrección Funcional	

4.4 Pruebas de Seguridad

Las pruebas se estructuraron para auditar la plataforma contra accesos indebidos, protección de la Ficha FIBE y vulnerabilidades subyacentes.

4.4.1 Control de Acceso a Rutas (Confidencialidad)

- **Objetivo:** Verificar el correcto funcionamiento del sistema RBAC (Control Basado en Roles).
- **Procedimiento:**
 1. Iniciar sesión utilizando credenciales de un rol no administrativo (Voluntario).
 2. Modificar la URL en la barra de direcciones intentando acceder a /admin o /gestion-usuarios.

3. Pulsar Enter para forzar la petición.
- **Criterio de Aceptación:** El sistema debe denegar el acceso devolviendo un error HTTP 403 o redirigiendo a la pantalla de inicio.

4.4.2 Auditoría de Dependencias (Integridad)

- **Objetivo:** Identificar vulnerabilidades conocidas (CVE) en las librerías de terceros utilizadas por el proyecto.
- **Procedimiento:**
 1. Abrir la terminal de comandos en la carpeta raíz del código fuente.
 2. Ejecutar el comando estándar npm audit.
 3. Revisar el reporte generado por el gestor de paquetes.
- **Criterio de Aceptación:** 0 reportes de nivel "High" (Alto) o "Critical" (Crítico).

4.4.3 Invalidación de Sesiones Web (Autenticidad)

- **Objetivo:** Garantizar que, una vez cerrado el sistema, no sea posible reingresar utilizando un token antiguo.
- **Procedimiento:**
 1. Iniciar sesión y realizar una acción básica (ej. cargar mapa).
 2. Presionar la opción "Cerrar Sesión" (Logout).
 3. Utilizar el botón "Atrás" del navegador para volver a la vista protegida.
 4. Intentar realizar una petición de actualización de datos.
- **Criterio de Aceptación:** El sistema debe expulsar al usuario al detectar que la sesión ya no es válida.

Las condiciones de aprobación se ven en la Tabla 4.

Tabla 4. Pruebas Seguridad
Fuente. Elaboración propia

Prueba	Condición Aprobación	Subcaracterística 25010	ISO/IEC
Control de Acceso a Rutas	Redirección o Error 403 comprobado.	Confidencialidad	
Auditoría de Dependencias	0 vulnerabilidades Altas/Críticas.	Integridad	
Invalidación de Sesiones Web	Prohibición total de uso post-logout.	Autenticidad	

4.5 Pruebas de Eficiencia de Desempeño

El diseño metodológico contempló utilizar métricas automatizadas para asegurar que los tiempos de respuesta del sistema no bloqueen el trabajo en terreno.

4.5.1 Rendimiento de Carga Inicial (Comportamiento Temporal)

- **Objetivo:** Medir el tiempo exacto que tarda el navegador en pintar el primer contenido útil en pantalla.
- **Procedimiento:**
 1. Abrir Google Lighthouse sobre un entorno local compilado en modo producción (Local Production Build)
 2. Seleccionar la categoría "Performance" con la configuración de dispositivo "Desktop".
 3. Ejecutar la auditoría de rendimiento.

4. Registrar la métrica denominada *First Contentful Paint* (FCP).
- **Criterio de Aceptación:** El valor de FCP debe ser inferior a 1.2 segundos (Estándar Google para redes estables).

4.5.2 Peso del Empaquetado Inicial (Utilización de Recursos)

- **Objetivo:** Asegurar que la descarga de la aplicación no sature el ancho de banda en conexiones de emergencia.
- **Procedimiento:**
 1. Abrir las herramientas de desarrollador (DevTools) en la pestaña *Network*.
 2. Desactivar la caché del navegador (*Disable cache*).
 3. Recargar la página inicial del sistema y registrar el total de Megabytes (MB) transferidos por la red correspondientes a los recursos base (JS, CSS, HTML).
- **Criterio de Aceptación:** El peso total de transferencia debe ser menor o igual a 2.0 MB.

Las condiciones de aprobación se ven en la Tabla 5.

Tabla 5. Pruebas eficiencia de desempeño
Fuente. Elaboración propia

Prueba	Condición Aprobación	Subcaracterística ISO/IEC 25010
Rendimiento de Carga Inicial	FCP <= 1.2 segundos.	Comportamiento Temporal
Peso del Empaquetado Inicial	Peso total <= 2.0 MB	Utilización de Recursos

4.6 Pruebas de Flexibilidad

Evaluación estructural del código y la base de datos para determinar si el software puede soportar su expansión a todo Chile.

4.6.1 Complejidad Ciclomática (Escalabilidad de Código)

- **Objetivo:** Detectar funciones complejas que dificultarían la mantenibilidad futura.
- **Procedimiento:**
 1. Configurar y ejecutar la regla complexity de la herramienta ESLint sobre la carpeta de controladores del backend.
 2. Listar todas las funciones que superen el límite establecido de caminos independientes.
- **Criterio de Aceptación:** 0 funciones con complejidad ciclomática superior a 15.

4.6.2 Detección de Código Hardcoded (Adaptabilidad)

- **Objetivo:** Verificar que el sistema no contenga datos geográficos quemados que lo aten a la ciudad de Valparaíso.
- **Procedimiento:**
 1. Realizar una búsqueda global en el repositorio de texto mediante comandos o IDE.
 2. Buscar cadenas como ciudades, nombres o coordenadas GPS específicas.
 3. Descartar los archivos de "datos semilla" (seeders) y focalizarse solo en la lógica de negocio (.ts, .tsx).
- **Criterio de Aceptación:** 0 ocurrencias de variables geográficas estáticas en el código operativo.

4.6.3 Soporte de Arquitectura Multi-Tenant (Escalabilidad de BD)

- **Objetivo:** Evaluar si el diseño actual de PostgreSQL permite soportar múltiples municipalidades en un solo servidor de manera aislada.
- **Procedimiento:**
 1. Inspeccionar estáticamente el Modelo Entidad-Relación (DER) del proyecto.
 2. Buscar la existencia de una tabla maestra Municipalidades.
 3. Verificar si las entidades núcleo (Usuarios, Centros, Inventario) poseen una llave foránea (ej. municipalidad_id) para filtrar los datos por cliente.
- **Criterio de Aceptación:** Presencia de la estructura Multi-Tenant para aislamiento de datos comunales.

Las condiciones de aprobación se ven en la Tabla 6.

Tabla 6. Pruebas de flexibilidad.
Fuente. Elaboración propia

Prueba	Condición Aprobación	Subcaracterística ISO/IEC 25010
Complejidad Ciclomática	0 funciones con más de 15 caminos.	Escalabilidad
Detección de Código Hardcoded	0 variables geográficas fijas.	Adaptabilidad
Soporte de Arquitectura Multi-Tenant	Posibilidad de manejo de multi municipalidades.	Escalabilidad

4.7 Cronograma ejecución de pruebas

La totalidad de las pruebas se planificó para ejecutarse en un lapso de 10 días, priorizando las pruebas automáticas al inicio para obtener métricas bases rápidas. Las tareas se distribuyen como se muestra en la Tabla 7.

Tabla 7. Cronograma ejecución de pruebas
Fuente. Elaboración propia

Actividad	Días 1-2	Días 3-4	Días 5-7	Días 8-10
Configuración y Análisis Estático (Automática)	X			
Auditoría Rendimiento Desktop (Automática)	X			
Fiabilidad y Detección Offline (Manual)		X		
Seguridad y Validación (Híbrida)			X	
Usabilidad y Accesibilidad (Manual)			X	
Análisis de hallazgos e informe				X

5 Realización del Plan de Evaluación

En este capítulo se presentan los datos crudos y resultados obtenidos tras la ejecución de las pruebas definidas en el Capítulo 4. Para garantizar la validez de las métricas de rendimiento y arquitectura, las pruebas fueron ejecutadas en un entorno local compilado en modo producción (Local Production Build), aislando así el sistema de latencias externas o tiempos de arranque en frío propios de los servidores gratuitos en la nube.

5.1 Resultados de Fiabilidad

5.1.1 Integridad de Sincronización (Recuperabilidad)

Para validar el mecanismo de persistencia local, la prueba se ejecutó sobre el módulo de "Gestión de Usuarios", evaluando la capacidad del sistema para encolar peticiones HTTP de tipo POST.

Tras ejecutar el protocolo de desconexión definido, el sistema capturó y encoló exitosamente tres peticiones POST de creación de usuarios en la base de datos local IndexedDB:

Tabla 8. Peticiones encoladas en IndexedDB

Fuente. Elaboración propia

ID	Método	URL	Estado
58291bab-0ced-410b-aaa-cb8638de069c	POST	/users	pending
71a6487f-0c5f-47be-802d-520d768348ed	POST	/users	pending
0b20304a-2f06-4377-ad9f-1ceab6c3ce94	POST	/users	pending

Al restablecer la red, el motor de sincronización procesó la cola automáticamente. Se verificó en la base de datos PostgreSQL la inserción de los tres registros de manera exacta.

Estado: Aprobado (100% de datos sincronizados sin pérdida ni duplicidad).

5.1.2 Navegación Offline en Vistas No Cacheadas (Tolerancia a Fallos)

Esta prueba evaluó la capacidad del Service Worker para servir la "cáscara" de la aplicación (App Shell) en rutas que no habían sido almacenadas en caché previamente por el usuario, simulando un corte de red abrupto en terreno.

La navegación interna operó correctamente sin conexión gracias al enrutador SPA, mostrando la estructura base de las vistas y manteniendo los elementos pre-cacheados. Sin embargo, al forzar una recarga completa (F5) en estado offline, la arquitectura PWA colapsó, exponiendo el error nativo del navegador ("ERR_INTERNET_DISCONNECTED") por falta de un Navigation Fallback en el Service Worker.

Estado: Parcialmente aprobado. Si bien el enrutador interno de la aplicación soporta la navegación offline inicial, la falta de una estrategia de Navigation Fallback en el Service Worker provoca una falla crítica (caída total del sistema) si el usuario recarga la página accidentalmente. Este hallazgo será abordado en el plan de mejoras.

5.1.3 Sensibilidad de Detección de Red (Disponibilidad)

Esta prueba verificó la eficacia del monitor de conectividad activo (Active Polling), el cual determina el estado real del sistema frente a intermitencias de red sin depender exclusivamente de los eventos nativos del navegador (los cuales suelen dar falsos positivos).

Al bloquear las peticiones del monitor de conectividad, el sistema detectó la caída y transicionó al estado *Offline* en exactamente 20 segundos (tras dos intentos fallidos de validación). Al retirar el bloqueo, el sistema demoró 9 segundos en recuperar el estado *Online* y reactivar la sincronización.

Extracto de logs de sistema (Detección Offline):

```
[ConnectivityMonitor] Ping fallido (1/2) TypeError: Failed to fetch
[ConnectivityMonitor] Máximo de intentos alcanzado, marcando OFFLINE
[ConnectivityMonitor] Estado: online → offline
[ConnectivityState] Sistema OFFLINE detectado por: ConnectivityMonitor
```

Estado: Aprobado. El sistema detecta y transiciona exitosamente entre estados de red, cumpliendo con la tolerancia configurada y garantizando la disponibilidad de la interfaz mediante una detección temprana de inestabilidad.

5.2 Resultados de Capacidad de Interacción

5.2.1 Usabilidad con Usuario Representativo (Aprendizabilidad)

Para la evaluación de esta prueba se analizó la grabación de una sesión de prueba guiada (Usability Testing) con las siguientes características:

- **Perfil del Usuario:** Trabajadora Social perteneciente a la Sección de Emergencia Social de DIDECO, Municipalidad de Valparaíso.
- **Cargo/Rol en el sistema:** Trabajador Municipal / Administrador.
- **Experiencia Previa:** Conocimiento experto en el dominio de emergencias sociales (operativo), sin conocimientos técnicos avanzados en sistemas TI.
- **Lugar y Duración de la Prueba:** La sesión se realizó en modalidad presencial en el Lab UX de la Universidad Federico Santa María con una duración total de aproximadamente 35 minutos.
- **Instrumento Aplicado:** Pauta de observación con cronometraje por tarea y cuestionario de satisfacción post-tarea basado en una escala Likert (1 al 7) para evaluar el tiempo percibido, la facilidad de uso y la información de apoyo.

Ejecución y Observaciones: La usuaria fue sometida a una batería de 8 tareas críticas. Todas las tareas fueron completadas exitosamente sin requerir intervención técnica del moderador. Los valores exactos y las calificaciones otorgadas para cada tarea se encuentran detallados en la Tabla A.1 de los Anexos. A continuación, se presentan los hallazgos cualitativos más relevantes obtenidos por tarea:

- **Flujos Exitosos (Alta Aprendizabilidad):** Las tareas de gestión básica (1, 3, 4, 5 y 6) se completaron en tiempos óptimos (entre 45 y 75 segundos), demostrando una curva de aprendizaje sumamente rápida. La usuaria otorgó la calificación máxima (7/7) al no encontrar fricciones en estos flujos.
- **Excepción de Tiempo por Carga de Datos (Tarea 2):** El registro de 240 segundos en la "Creación de centro de acopio" no responde a un problema de usabilidad, sino a la longitud intrínseca del requerimiento. El sistema solicita el ingreso de una alta volumetría de información (alrededor de 50 parámetros evaluando estados del centro en escala de 1 a 5). La usuaria completó el formulario sin asistencia ni bloqueos de comprensión, sugiriendo únicamente el uso de tipografía en negrita para los títulos del formulario para agilizar la lectura, lo que explica la evaluación de 6/7.
- **Fricciones de Interfaz (Búsqueda Visual):** En la tarea 7 (Ficha FIBE), se observó un leve retraso visual (110s). La usuaria indicó dificultad para ubicar el módulo en la interfaz inicial, lo que sugiere que este botón debería estar posicionado más alto en la jerarquía visual para facilitar su acceso durante emergencias (calificación 6/7).
- **Fricción Crítica de Lógica (UX de Notificaciones):** La tarea 8 presentó la mayor discrepancia. Al buscar la alerta de un centro desactivado, la usuaria se dirigió intuitivamente al "Buzón de notificaciones" principal, pero el sistema despliega estas alertas en el detalle interno de cada centro. Este choque entre el modelo mental de la usuaria y la arquitectura del sistema generó confusión, reduciendo la calificación a 5/7.
- **Intención de Uso:** En la entrevista final, la profesional confirmó que el sistema es viable y que lo utilizaría en su trabajo ante la necesidad de coordinar un albergue.

Estado: Aprobado con Observaciones. El sistema valida su "Aprendizabilidad", permitiendo a una usuaria no técnica operar las funciones clave sin asistencia en su primer acercamiento. Se justifica el tiempo extendido en la creación de centros por la densidad de los datos requeridos por el negocio. No obstante, el plan de mejoras

deberá contemplar el ajuste de la experiencia de usuario (UX) en la bandeja de notificaciones administrativas, la jerarquía visual del módulo FIBE y la inclusión de negritas en formularios extensos.

5.2.2 Operabilidad en Entorno Desktop (Operabilidad)

Esta prueba evaluó la adaptabilidad gráfica de la interfaz de usuario frente a diferentes resoluciones de pantalla estándar en equipos de escritorio, asegurando que la información crítica no se oculte ni se deforme.

La interfaz gráfica (grilla y contenedores flexibles) se adaptó de forma fluida a las resoluciones probadas (1366x768 y 1920x1080). No se registraron desbordamientos horizontales, elementos superpuestos ni textos truncados que afectaran el uso.

Estado: Aprobado. La interfaz gráfica demuestra una alta madurez en su diseño responsivo para entornos de escritorio, garantizando la operabilidad sin importar el hardware de visualización del usuario.

5.2.3 Validación de Errores de Usuario (Protección)

Esta prueba evaluó la robustez de los formularios del sistema, verificando si previenen el ingreso de datos corruptos y si proveen una retroalimentación clara que guíe al usuario al cometer un error, factor crítico dado el contexto de estrés en emergencias.

El sistema protegió con éxito la base de datos, bloqueando el 100% de los envíos con datos negativos o incompletos. No obstante, se detectaron fallos de interfaz: el selector numérico permite temporalmente ingresar el valor "0" causando comportamientos visuales erráticos, se carece de validación para exigir contraseñas seguras, y los errores por colisión de datos (ej. correo duplicado) exponen el código HTTP 409 nativo en lugar de brindar un mensaje comprensible al usuario.

Estado: Aprobado con Observaciones. Aunque el sistema protege exitosamente la integridad de la base de datos frente a datos matemáticamente imposibles, la carencia de validación en las contraseñas y la exposición de códigos de error HTTP directos al usuario final degradan significativamente la "Operabilidad" y requerirán mitigación inmediata en el plan de mejoras.

5.2.4 Auditoría de Accesibilidad Web (Inclusividad)

Esta prueba evaluó el cumplimiento técnico de las Pautas de Accesibilidad al Contenido en la Web (WCAG) utilizando la herramienta Google Lighthouse, con el fin de garantizar que la plataforma sea inclusiva y operable por usuarios que dependan de tecnologías de asistencia (como lectores de pantalla) o que posean discapacidades visuales. Los resultados obtenidos y las deficiencias detectadas en la muestra representativa se resumen en la Tabla A.2 de los anexos.

Estado: Desaprobado. Al presentar dos de sus tres vistas principales por debajo del umbral de aceptación (85/100), la plataforma presenta barreras restrictivas de accesibilidad. La corrección de las paletas de colores, la inyección de atributos semánticos ARIA en los formularios y la corrección de la jerarquía del DOM deberán ser priorizadas en el plan de mejoras.

5.3 Resultados de Adecuación Funcional

5.3.1 Cobertura del Producto Mínimo Viable (Compleitud)

Esta prueba verificó que el sistema implementa todas las funciones críticas prometidas en las etapas tempranas del proyecto para satisfacer a los tres actores clave (DIDECO, Encargados, Comunidad).

Se verificó la existencia y operatividad del 100% de las historias de usuario catalogadas como "Esenciales" visibles en la Tabla A.3 de los anexos. No se detectaron rutas rotas ni botones inoperativos.

Estado: Aprobado. El software cumple íntegramente con el alcance funcional exigido para su operación base.

5.3.2 Exactitud Matemática de Inventarios (Corrección)

El objetivo fue validar que las operaciones aritméticas de entrada/salida de stock sean exactas y se reflejen correctamente en la interfaz, previniendo errores de cálculo en la logística de distribución.

Los cálculos matemáticos de entrada y salida operaron con total exactitud en la base de datos (0 discrepancias detectadas). Sin embargo, se identificó un fallo crítico de reactividad en el frontend: los indicadores gráficos y niveles de abastecimiento no se actualizan tras procesar la mutación, obligando al usuario a recargar la página manualmente para reflejar el *stock* real.

Estado: Aprobado en capa de datos, desaprobado en actualización de interfaz. Si bien el motor de cálculo es 100% exacto, la deficiente gestión del estado global en el frontend rompe la promesa de "monitoreo en tiempo real" y requiere una refactorización para implementar un mecanismo de re-renderizado automático.

5.4 Resultados de Seguridad

5.4.1 Control de Acceso a Rutas (Confidencialidad)

El objetivo de esta prueba fue auditar el Control de Acceso Basado en Roles (RBAC), garantizando que los usuarios con menores privilegios no puedan visualizar ni alterar información administrativa confidencial. Aunque la interfaz oculta correctamente los menús administrativos según el rol asignado, el enrutador de la aplicación carece de validación activa (*Route Guards*).

Durante la ejecución, se comprobó esta escalada de privilegios horizontal iniciando sesión con el rol de "Trabajador Municipal" (perfil sin privilegios globales) e ingresando manualmente la ruta protegida `/admin/updates` en la barra del navegador. En lugar de interceptar la navegación y aplicar el rechazo esperado mediante un código HTTP 403 (*Forbidden*), el comportamiento observado fue la carga completa de la interfaz administrativa, permitiendo la visualización y alteración de datos dinámicos confidenciales de forma indebida (ver **Figura 1 en los Anexos** para la captura de pantalla de este acceso).

Estado: Desaprobado (Fallo Crítico). La plataforma es altamente vulnerable a accesos no autorizados mediante referencias directas a URLs. Ocultar los enlaces visuales en el menú no reemplaza la necesidad de implementar *Route Guards* estables en el código, convirtiendo este hallazgo en la máxima prioridad para el plan de mejoras.

5.4.2 Auditoría de Dependencias (Integridad)

Esta prueba tuvo como objetivo identificar vulnerabilidades conocidas (CVE) introducidas en el sistema a través del árbol de dependencias de terceros, evaluando tanto el entorno del servidor (Backend) como el cliente (Frontend).

El entorno del servidor (*backend*) arrojó un estado saludable con solo 1 vulnerabilidad de nivel Bajo. Por el contrario, el cliente (*frontend*) registró 5 vulnerabilidades (4 Altas y 1 Crítica) asociadas a dependencias heredadas como `jspdf` y `serialize-javascript`, abriendo vectores graves para Inyección de Código (XSS) y Ejecución Remota de Código (RCE).

Estado: Desaprobado. El sistema no cumple con la política de cero vulnerabilidades críticas. La presencia de fallas de inyección en el generador de PDFs y RCE en el empaquetador PWA exponen a la plataforma y a sus usuarios. Requerirá la actualización forzada de paquetes (`npm audit fix --force`) o la sustitución de librerías en el plan de mejoras.

5.4.3 Invalidación de Sesiones Web (Autenticidad)

El objetivo de esta prueba fue garantizar que el proceso de "Cerrar Sesión" destruya efectivamente las credenciales de acceso del usuario, evitando que una persona no autorizada pueda utilizar el historial del navegador (botón "Atrás") para reingresar a las vistas protegidas en un dispositivo compartido.

Tras ejecutar el cierre de sesión, se intentó reingresar a las vistas protegidas utilizando el historial del navegador (botón "Atrás"). El sistema interceptó correctamente la navegación, validó la destrucción del *token* y forzó una redirección automática hacia el entorno público.

Estado: Aprobado. El sistema maneja de forma segura y robusta el ciclo de vida final de la sesión, protegiendo la autenticidad de las operaciones.

5.5 Resultados de Eficiencia de Desempeño

5.5.1 Rendimiento de Carga Inicial (Comportamiento Temporal)

Esta prueba evaluó la velocidad de renderizado de la interfaz gráfica, un factor crítico para evitar el abandono de la plataforma por parte de usuarios operando bajo estrés en redes móviles degradadas. Para aislar la prueba de latencias externas de servidores y garantizar la rigurosidad metodológica de la medición, se evaluó el indicador First Contentful Paint (FCP) utilizando un entorno local compilado para producción (Local Production Build).

Las especificaciones exactas del entorno de medición (detalladas en la Tabla A.4), junto con los reportes completos generados por la herramienta de auditoría, se encuentran disponibles en la sección de Anexos.

Estado: Aprobado. El sistema se encuentra holgadamente por debajo del límite crítico establecido de 1.2 segundos, asegurando una percepción de velocidad instantánea para el usuario.

5.5.2 Peso del Empaquetado Inicial (Utilización de Recursos)

El objetivo de esta prueba fue verificar que la compilación de la aplicación de página única (SPA) no genere archivos de código (bundles) excesivamente pesados que saturen el ancho de banda.

La transferencia máxima de datos por red (sin utilizar caché) fue de 0.76 MB en la vista de mayor peso, cumpliendo con las exigencias para redes móviles limitadas al situarse muy por debajo del umbral de tolerancia de 2.0 MB.

Estado: Aprobado. El peso total de la plataforma es extremadamente ligero, situándose muy por debajo del límite máximo tolerado de 2.0 MB. Esto garantiza su viabilidad operativa mediante conexiones móviles compartidas (3G/4G).

5.6 Resultados de Flexibilidad

5.6.1 Complejidad Ciclomática (Escalabilidad de Código)

Esta métrica evaluó la calidad estructural interna del código fuente, factor determinante para asegurar que el sistema pueda ser mantenido, modificado y escalado por nuevos equipos de desarrollo sin introducir errores.

De 1.863 funciones analizadas estáticamente, el 91.9% presentó baja complejidad (promedio general de 2.6). No obstante, se detectaron 11 funciones y componentes críticos que superan el límite de 15 caminos independientes estipulado. El detalle exhaustivo de estos componentes, incluyendo su función, archivo de origen, nivel de complejidad y riesgo técnico asociado, se encuentra documentado en la Tabla A.5 de los Anexos. Como hallazgo principal, el cuello de botella arquitectónico más grave se registró en el componente *CenterResidentsPage*, alcanzando una Complejidad Ciclomática de 73.

Estado: Desaprobado. Aunque la base del código es sana, la existencia de componentes monolíticos que quintuplican el umbral máximo permitido ($CC \leq 15$) rompe el principio de mantenibilidad y exige una refactorización urgente (división en subcomponentes) antes de escalar la plataforma a nivel nacional.

5.6.2 Detección de Código Hardcoded (Adaptabilidad)

El objetivo fue verificar la ausencia de acoplamiento geográfico en la lógica de negocio, asegurando que la plataforma pueda ser desplegada en cualquier ciudad de Chile sin necesidad de alterar el código fuente.

Se identificó acoplamiento geográfico explícito en la lógica de negocio. El componente visual de mapas centra su vista inicialmente en coordenadas fijas de la ciudad de Valparaíso (lat: -33.04, lng: -71.61), y los servicios de carga de archivos (CSV) contienen ubicaciones de ejemplo quemadas en el código.

Estado: Desaprobado. El sistema no es agnóstico a la ubicación. Para ser adoptado por otras comunas, se requiere parametrizar estas variables trasladándolas a la base de datos o a configuraciones de entorno.

5.6.3 Soporte de Arquitectura Multi-Tenant (Escalabilidad de BD)

El objetivo final de esta evaluación fue analizar la viabilidad arquitectónica de expandir la plataforma para que sea utilizada simultáneamente por múltiples municipalidades de Chile bajo una misma infraestructura de servidores.

La inspección del Modelo Entidad-Relación comprobó un diseño bajo un enfoque *Single-Tenant* estricto. Las tablas transaccionales carecen de llaves foráneas organizacionales (ej. `municipalidad_id`) que permitan aislar lógicamente los datos de diferentes municipios dentro de una misma infraestructura.

Estado: Desaprobado. La base de datos fue diseñada bajo un enfoque *Single-Tenant* estricto. Para que otra municipalidad utilice el sistema en la actualidad, sería necesario desplegar un servidor y una base de datos completamente independientes para cada cliente, lo cual es ineficiente a nivel de infraestructura. La transición hacia una arquitectura Multi-Tenant será un pilar fundamental en las propuestas de mejora.

6 Análisis de Resultados

En este capítulo se presenta el análisis crítico y la interpretación de los datos obtenidos durante la ejecución del plan de evaluación. El objetivo es determinar si AppCopio cumple con los estándares de calidad necesarios para operar en un contexto de emergencia representativo y evaluar su viabilidad de escalamiento, traduciendo las métricas técnicas en impactos operativos.

6.1 Análisis de Fiabilidad

La fiabilidad general del sistema presenta un desempeño dual: altamente robusto en su lógica interna, pero frágil en ciertos aspectos de la experiencia de usuario. Por un lado, la arquitectura de sincronización en segundo plano (IndexedDB y Mutation Queue) demostró una efectividad del 100%, garantizando que ningún registro de dañado se pierda en zonas sin cobertura. Asimismo, el monitor de red personalizado logró mitigar las deficiencias nativas de los navegadores, otorgando al sistema una consciencia situacional exacta sobre el estado de la conectividad (transición a offline en 20 segundos).

Sin embargo, la tolerancia a fallos falló críticamente ante el comportamiento humano esperado bajo estrés. Si bien la aplicación soporta la navegación offline mediante el enrutador de React, la carencia de una página de respaldo (Navigation Fallback) en el Service Worker provoca que una recarga manual (tecla F5) colapse el sistema por completo. En un contexto de emergencia, donde los usuarios tienden a recargar las páginas cuando perciben lentitud, este fallo técnico se traduce en una pérdida total de la herramienta operativa hasta que retorne el internet.

6.2 Análisis de Capacidad de Interacción

La usabilidad del sistema es altamente favorable para la operatividad básica, pero presenta barreras excluyentes a nivel técnico. Las pruebas empíricas realizadas con el usuario clave de DIDECO validaron una curva de aprendizaje sumamente rápida: la usuaria logró crear centros, modificar inventarios y gestionar perfiles

sin capacitación previa, evaluando estas tareas con calificaciones de excelencia (7/7). Asimismo, el diseño fluido y libre de desbordamientos visuales en la vista de escritorio contribuye a esta percepción positiva.

No obstante, el análisis revela que la interfaz no está preparada para la diversidad funcional. Los puntajes reprobatorios en la auditoría de accesibilidad (promedio 84.6/100) evidencian que el bajo contraste de colores y la falta de semántica (etiquetas ARIA) impiden que voluntarios con discapacidades visuales o usuarios de lectores de pantalla utilicen la plataforma de forma autónoma. Adicionalmente, las fricciones encontradas en el sistema de notificaciones para administradores y la retroalimentación de errores crudos (ej. Error 409) elevan la carga cognitiva del usuario, lo cual es contraproducente en escenarios de crisis donde la claridad debe ser absoluta.

6.3 Análisis de Adecuación Funcional

El sistema cumple con el 100% de las historias de usuario esenciales presentes y los cálculos de distribución de ayuda son matemáticamente exactos a nivel de base de datos. Esto asegura que DIDECO no tomará decisiones logísticas basadas en datos corruptos en el servidor.

El problema radica en la reactividad de la interfaz. Al no actualizar los gráficos y niveles de abastecimiento en tiempo real tras una transacción, se rompe la promesa de "monitoreo en vivo". Operativamente, si un encargado ingresa 50 raciones de agua y la pantalla no refleja el cambio de inmediato, este podría asumir que el sistema falló y registrar la entrada nuevamente, duplicando el inventario, o forzar una recarga (F5) que, de estar offline, colapsaría la aplicación (como se evidenció en la Fiabilidad). Por ende, aunque la función matemática es correcta, la adecuación funcional de la interfaz es deficiente.

6.4 Análisis de Seguridad

El análisis sitúa a la seguridad como el eslabón más débil de la plataforma, representando un riesgo inminente para la privacidad de los damnificados. Si bien el ciclo de vida de la sesión se gestiona correctamente (cumpliendo el criterio de autenticidad al prevenir el secuestro de sesiones en dispositivos compartidos), la evaluación evidenció el incumplimiento directo de los criterios de aceptación definidos para la confidencialidad y la integridad. El sistema presenta dos vulnerabilidades críticas que lo inhabilitan categóricamente para un despliegue en producción, veredicto que se sustenta en la evidencia técnica y capturas documentadas en la sección de Anexos.

En primer lugar, el fallo en el criterio de confidencialidad se manifiesta en un "Control de Acceso Roto". Tal como demuestra la evidencia, un usuario de bajo privilegio puede escalar horizontalmente hacia vistas administrativas evadiendo por completo el modelo RBAC debido a la ausencia de Guardianes de Rutas (*Route Guards*). En segundo lugar, el criterio de integridad fue desaprobado al detectarse dependencias obsoletas en el cliente que abren vectores de ataque comprobados para la Inyección de Código (XSS) y la Ejecución Remota de Código (RCE). Manejar información estatal altamente sensible, como los datos personales de las familias afectados contenidos en las Fichas FIBE, bajo estas condiciones vulnerables, viola los principios de protección de datos y expone a la municipalidad a brechas severas.

6.5 Análisis de Eficiencia de Desempeño

La eficiencia de desempeño es la característica más sobresaliente y madura del proyecto. La decisión arquitectónica de compilar el sistema mediante herramientas modernas (Vite) resultó en un peso de empaquetado inicial ultraligero (inferior a 0.8 MB) y un tiempo de renderizado casi instantáneo (FCP de 0.7 segundos).

En el contexto geográfico de Valparaíso donde los centros de acopio en los cerros suelen padecer de conectividad móvil intermitente o redes compartidas, esta optimización garantiza que la herramienta cargará y será operativa sin consumir excesivo ancho de banda, eliminando los tiempos muertos de espera y maximizando la eficiencia de los voluntarios en terreno.

6.6 Análisis de Flexibilidad

El análisis estructural demostró que AppCopio, en su estado actual, no posee la flexibilidad necesaria para ser escalado a nivel nacional. La arquitectura fue diseñada bajo el paradigma de un sistema hecho a la medida para la Municipalidad de Valparaíso.

La codificación rígida de variables geográficas y, fundamentalmente, la ausencia de una base de datos Multi-Tenant impiden que otras comunas adopten el software sin que el equipo de desarrollo deba clonar, modificar el código y levantar infraestructuras de servidores totalmente independientes por cada nuevo cliente, incurriendo en altos costos operativos. A nivel de código fuente, la presencia de 11 funciones monolíticas con complejidad ciclomática crítica (hasta 73 caminos independientes) indica una alta deuda técnica que dificultará severamente el trabajo de futuros desarrolladores que intenten agregar nuevas características al sistema.

6.7 Evaluación Global de Calidad

A partir del análisis de las seis características de la norma ISO/IEC 25010, se concluye que AppCopio es un Producto Mínimo Viable (PMV) altamente eficiente y funcional, pero inmaduro para su despliegue comercial o escalamiento nacional.

Sus fortalezas radican en su excepcional rendimiento web y su capacidad para operar sin conexión a internet, resolviendo el problema núcleo de la gestión logística post-catástrofe. Sin embargo, los fallos críticos detectados en el Control de Acceso (Seguridad), la falta de reactividad en la interfaz (Adecuación Funcional) y su fuerte acoplamiento a una sola municipalidad (Flexibilidad) exigen una intervención técnica inmediata. El sistema aprueba condicionalmente como prototipo operativo para una comuna controlada, pero requiere ejecutar un plan de refactorización estructural antes de manipular datos reales de damnificados.

7 Plan de Mejoras

Basado en el análisis diagnóstico detallado en el capítulo anterior, este apartado propone un plan de acción técnico para mitigar las vulnerabilidades encontradas y preparar a AppCopio para su escalamiento a nivel nacional. Las propuestas abordan directamente los fallos críticos de seguridad, la inmadurez de la arquitectura de datos y las fricciones en la experiencia de usuario.

7.1 Mejoras de Fiabilidad

El mecanismo de sincronización diferida ha demostrado ser robusto, pero el colapso del sistema ante una recarga manual sin conexión representa un riesgo operativo inaceptable.

Implementación de Navigation Fallback: Se debe reconfigurar el Service Worker para interceptar las peticiones de navegación de tipo text/html. Si la red falla y el recurso no está en la caché estricta, el Service Worker debe retornar una vista pre-cacheada genérica o el mismo App Shell para permitir que el enrutador de React tome el control y dibuje la interfaz offline de manera elegante, evitando que el navegador nativo muestre el error de desconexión.

7.2 Mejoras de Usabilidad y Funcionalidad

Las deficiencias encontradas en esta área afectan tanto a la inclusión de los voluntarios como a la confianza operativa de los administradores.

Refactorización de Gestión de Estado Global: Para solucionar la falta de actualización en tiempo real de los inventarios, se debe abandonar la gestión de estado local aislada. Se propone implementar un manejador de estado global asíncrono como React Query o SWR. Estas librerías permiten invalidar automáticamente la caché del frontend tras ejecutar una petición POST/PUT (mutación), forzando a los componentes visuales (gráficos

de abastecimiento) a volver a renderizarse instantáneamente con los datos actualizados de la base de datos sin necesidad de recargar la página.

Corrección de Accesibilidad: Se debe realizar un rediseño menor en la paleta de colores del *framework* Material UI para asegurar que todos los botones y textos cumplan con el ratio de contraste mínimo de 4.5:1. Adicionalmente, es obligatorio inyectar el atributo `aria-label` en todos los componentes interactivos abstractos (switches, iconos) y corregir la jerarquía del DOM para que los elementos de lista (`<li>`) posean un contenedor semántico adecuado.

Rediseño de Experiencia de Usuario (UX) Administrativa: Se debe reestructurar la jerarquía visual del Dashboard, elevando la posición de acceso al formulario FIBE para garantizar su disponibilidad inmediata en emergencias. Asimismo, es imperativo unificar la bandeja de notificaciones: los avisos críticos, como la activación o desactivación de centros de acopio, deben despacharse directamente al buzón general del administrador, en lugar de quedar ocultos exclusivamente en la vista de detalles del centro afectado.

Gestión Amigable de Excepciones: Para disminuir la carga cognitiva, se debe implementar un interceptor de respuestas en el frontend que capture los errores crudos devueltos por el servidor (como el Error 409 detectado en las pruebas) y los traduzca en alertas contextuales y comprensibles para un usuario no técnico, indicando claramente cómo resolver el conflicto.

Mejora de Legibilidad en Formularios: Incorporar diferenciación tipográfica (uso de negritas) en los títulos de las distintas secciones de los formularios más extensos (como el de creación de centros de acopio). Esto agilizará el escaneo visual de la información y reducirá la fatiga durante los ingresos masivos de datos.

7.3 Mejoras de Seguridad

La seguridad de AppCopio requiere intervención de máxima urgencia para proteger los datos sensibles y la integridad del servidor.

Implementación de Route Guards: El "Control de Acceso Roto" debe solucionarse de inmediato creando un componente de orden superior en React. Este componente debe interceptar la navegación, leer la carga útil del token JWT almacenado y verificar el rol del usuario. Si el rol es insuficiente, debe forzar una redirección al inicio de forma programática.

Sanitización de Dependencias Críticas: Se debe ejecutar una actualización forzada (`npm audit fix --force`) y evaluar la migración de la librería `jspdf` (actualmente con vulnerabilidades de inyección y ejecución remota de código) hacia alternativas más modernas y seguras en el ecosistema Node.js, como `pdfkit` o delegar la generación del PDF estrictamente al entorno aislado del backend.

Políticas Estrictas de Entrada de Datos: Incorporar validaciones mediante expresiones regulares (Regex) para forzar contraseñas seguras (mínimo 8 caracteres, alfanuméricas) y reemplazar los errores técnicos directos (HTTP 409) por mensajes amigables y semánticos que guíen al usuario (ej. "El correo ingresado ya pertenece a otro voluntario").

7.4 Mejoras de Eficiencia y Arquitectura

Para cumplir con la visión de implementar AppCopio en todas las municipalidades de Chile, la estructura del código y la base de datos deben ser abstraídas de la ciudad de Valparaíso.

Migración a Arquitectura Multi-Tenant: La base de datos PostgreSQL debe ser rediseñada. Se debe crear una entidad central Municipalidades. Todas las tablas transaccionales y maestras (Usuarios, Centros, Inventario, Damnificados) deben incorporar una llave foránea `municipalidad_id`. En el backend, se debe implementar una capa de filtrado global para que cada consulta a la base de datos incluya implícitamente el identificador del municipio del usuario que realiza la petición, garantizando el aislamiento lógico de los datos en un servidor único.

Desacoplamiento de Variables Geográficas: Todas las coordenadas estáticas detectadas en los componentes deben ser extraídas y trasladadas a variables de entorno o integradas como parámetros de configuración descargados desde la base de datos según el municipio autenticado.

Reducción de Deuda Técnica: Las vistas monolíticas, específicamente CenterResidentsPage (Complejidad Ciclomática de 73), deben ser refactorizadas utilizando el principio de responsabilidad única. La vista debe fragmentarse en subcomponentes atómicos y la lógica de filtrado debe aislarse en Hooks personalizados independientes.

7.5 Priorización de Implementación

Dada la cantidad de mejoras propuestas tras la evaluación, es fundamental establecer una hoja de ruta técnica que maximice el valor operativo minimizando el tiempo de desarrollo. Para ello, las acciones correctivas se han priorizado utilizando una matriz estratégica que evalúa el nivel de impacto, el esfuerzo requerido, el riesgo técnico mitigado, las responsabilidades y el criterio estricto de validación. Esta planificación se estructura en tres fases de ejecución detalladas en la Tabla 9.

Tabla 9. Matriz de Priorización de Mejoras y Hoja de Ruta Técnica.

Fuente. Elaboración propia

Fase y Acción Propuesta	Impacto	Esfuerzo	Riesgo Mitigado	Responsable	Criterio de Validación
FASE 1: Tareas Críticas de Rápido Impacto (Quick Wins)					
1. Implementación de Route Guards	Alto	Medio	Escalada de privilegios y acceso no autorizado a datos confidenciales (Ficha FIBE).	Desarrollo Frontend	Intento de acceso directo a /admin devuelve rechazo (HTTP 403) o redirección.
2. Resolución de dependencias vulnerables	Alto	Bajo	Inyección de Código (XSS) y Ejecución Remota (RCE) en el cliente.	Desarrollo Frontend	Comando npm audit arroja cero vulnerabilidades de severidad Alta o Crítica.
3. Navigation Fallback en Service Worker	Alto	Bajo	Colapso de la aplicación (pantalla de error nativa) al recargar vistas sin conexión.	Desarrollo Frontend	Recarga (F5) en estado offline muestra una vista de resguardo sin perder el entorno de la PWA.
FASE 2: Proyectos Estructurales					
4. Migración a BD Multi-Tenant	Alto	Alto	Imposibilidad de escalar la plataforma a nivel nacional; aislamiento de datos.	Arquitectura / Backend	Base de datos implementa llaves foráneas organizacionales y el sistema filtra por municipio.
5. Gestión de estado con React Query	Alto	Alto	Decisiones logísticas erróneas en DIDECO por gráficos desactualizados.	Desarrollo Frontend	Gráficos y niveles de inventario se actualizan automáticamente tras cada mutación sin recargar.
6. Refactorización CenterResidentsPage	Alto	Alto	Inmantenibilidad del código fuente; alta probabilidad de introducir bugs.	Desarrollo Frontend	Análisis estático confirma que todas las funciones resultantes poseen

					Complejidad Ciclomática ≤ 15 .
FASE 3: Optimización y Accesibilidad					
7. Atributos ARIA y paleta de colores	Medio	Bajo	Exclusión operativa de usuarios con discapacidades visuales o en condiciones de alta luminosidad.	Diseño UI / Frontend	Herramienta Lighthouse certifica un puntaje de Accesibilidad $\geq 90/100$ en todas las vistas.
8. Unificación de notificaciones	Medio	Bajo	Confusión operativa y fricción en la experiencia de usuarios administradores.	Diseño UX / Frontend	Pruebas de usabilidad de validación superan la calificación de 6/7 en facilidad de uso.
9. Manejo de errores en formularios	Medio	Bajo	Frustración del usuario por exposición de errores HTTP nativos (ej. 409).	Desarrollo Frontend	Sistema captura el error y despliega mensajes amigables y descriptivos en la interfaz.

8 Conclusiones

La presente memoria tuvo como objetivo evaluar la calidad del software AppCopio, una plataforma crítica para la gestión de emergencias, utilizando el estándar internacional ISO/IEC 25010. A través del diseño y ejecución de un plan de pruebas riguroso, se logró reemplazar la incertidumbre operativa por un diagnóstico empírico, objetivo y cuantificable, respaldado por las evidencias técnicas y reportes analíticos recopilados en la sección de Anexos.

El análisis de los resultados permite concluir que los objetivos planteados fueron abordados de manera sustantiva, permitiendo generar un diagnóstico técnico y un plan de mejoras. La evaluación demostró que AppCopio es una aplicación completamente funcional desarrollada como un piloto municipal con un núcleo tecnológico excepcionalmente ágil, pero que actualmente carece de la madurez arquitectónica y de seguridad necesaria para un despliegue masivo a nivel nacional.

En cuanto a sus fortalezas, el sistema resolvió con éxito el mayor desafío técnico de los entornos de catástrofe: la conectividad degradada. La implementación de una arquitectura de Aplicación Web Progresiva (PWA) garantizó métricas de rendimiento sobresalientes (con tiempos de carga de 0.7 segundos y un empaquetado inferior a 1 MB) y demostró una sincronización offline efectiva en las pruebas ejecutadas mediante IndexedDB. Asimismo, la prueba con una usuaria representativa del municipio de Valparaíso confirmó que la plataforma posee una alta aprendizabilidad, facilitando su adopción por parte de equipos de emergencia bajo presión.

Sin embargo, someter el sistema a un escrutinio profundo reveló vulnerabilidades que invalidan su uso inmediato en producción con datos reales. La detección de un fallo crítico en el Control de Acceso (RBAC), que permitía a usuarios sin privilegios acceder a rutas administrativas, junto con el uso de dependencias vulnerables y la deficiente gestión del estado global en la interfaz, demostraron el peligro de omitir procesos formales de Aseguramiento de Calidad (SQA) durante el ciclo de desarrollo. Además, el fuerte acoplamiento del código a la ciudad de Valparaíso y la ausencia de una base de datos Multi-Tenant confirmaron que el sistema requiere una refactorización profunda antes de ser ofrecido a otras municipalidades.

El valor fundamental de este trabajo de título radica en la prevención de riesgos. Al descubrir fallos de seguridad críticos y cuellos de botella arquitectónicos en una etapa de auditoría, se identificaron de forma

temprana riesgos que habrían comprometido un eventual despliegue, evitando escenarios que pudiesen exponer información sensible de los damnificados o provocar un colapso logístico en medio de una emergencia real.

Finalmente, este trabajo no solo diagnostica el estado actual de AppCopio, sino que entrega un Plan de Mejoras priorizado y técnicamente fundamentado. El trabajo futuro inmediato consiste en la ejecución de este plan; de la correcta implementación de estas acciones de remediación y de su posterior validación técnica dependerá la transición definitiva de AppCopio desde su estado actual de piloto funcional hacia una solución de nivel empresarial, estandarizada, segura y lista para centralizar la ayuda humanitaria de manera robusta.

Agradecimientos

Quisiera expresar mi más profundo agradecimiento a mi familia por ser mi pilar fundamental de apoyo durante todos mis años de estudio, brindándome la fuerza y motivación necesarias para alcanzar esta meta profesional.

Agradezco a mi profesor guía, Mauricio Figueroa, por su orientación, exigencia y constante disposición para guiar el desarrollo de esta memoria, compartiendo su experiencia para llevar este trabajo al más alto estándar de la ingeniería.

A la Universidad Técnica Federico Santa María y a mis compañeros, por los años de formación, las largas jornadas de estudio y por forjar en mí el rigor característico de nuestra casa de estudios.

Finalmente, un agradecimiento especial a los equipos de emergencia y trabajadores sociales de la Municipalidad de Valparaíso que colaboraron en este proyecto, cuyo incansable trabajo en terreno es la verdadera inspiración detrás de soluciones tecnológicas como AppCopio.

Referencias

- [1] R. S. Pressman, Ingeniería de software: un enfoque práctico, 7 ed., México: McGraw-Hill, 2010.
- [2] ISO25000.com, «La familia de normas ISO/IEC 25000,» AQCLab, 2024. [En línea]. Available: <https://iso25000.com/index.php/normas-iso-25000>. [Último acceso: 10 04 2025].
- [3] ISO/IEC, «ISO/IEC 25010:2023 - Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models,» International Organization for Standardization, Ginebra, Suiza, 2023.
- [4] ISO/IEC, «ISO/IEC 25040:2024 - Software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Evaluation process,» International Organization for Standardization, Ginebra, Suiza, 2024.
- [5] T. McCabe, «A Complexity Measure,» IEEE Transactions on Software Engineering, Vols. SE-2, pp. 308-320, 31 December 1976.
- [6] Google web.dev, «Web Vitals,» 2020. [En línea]. Available: <https://web.dev/articles/vitals>. [Último acceso: 14 04 2025].
- [7] Google web.dev, «First Contentful Paint (FCP),» 2023. [En línea]. Available: <https://web.dev/articles/fcp>. [Último acceso: 14 04 2025].
- [8] MDN Web Docs, «Progressive web apps (PWAs),» Mozilla Foundation, 2025. [En línea]. Available: https://developer.mozilla.org/es/docs/Web/Progressive_web_apps. [Último acceso: 14 04 2025].
- [9] World Wide Web Consortium (W3C), «Web Content Accessibility Guidelines (WCAG) Overview,» Web Accessibility Initiative (WAI), 2023. [En línea]. Available: <https://www.w3.org/WAI/standards-guidelines/wcag/>. [Último acceso: 14 04 2025].
- [10] INCITS, «Information technology – Role Based Access Control (INCITS 359-2012),» American National Standards Institute (ANSI), 2012. [En línea]. Available: <https://blog.ansi.org/ansi/role-based-access-control-rbac-incits-359/>. [Último acceso: 14 04 2025].
- [11] J. P. Correia y M. Shepperd, «Software quality: the elusive target,» IEEE Software, vol. 38, n° 1, pp. 18-21, 2021.

- [12] R. Quispe-Vilchez, R. Mamani-Apaza y M. Quispe-Quispe, «Evaluación de calidad de software de un sistema de información web del sector salud aplicando el estándar ISO/IEC 25010,» Revista Ibérica de Sistemas e Tecnologías de Informação, n° E50, pp. 285-298, 2022.
- [13] J. Rodríguez-Flores, L. M. Sánchez-Adame y J. Urquiza-Fuentes, «Evaluación de la eficiencia de desempeño de aplicaciones de realidad aumentada bajo el estándar ISO/IEC 25010,» Programación Matemática y Software, vol. 12, n° 3, pp. 1-14, 2020.
- [14] M. Bustos, «Capacidades institucionales y vulnerabilidad territorial frente a desastres en la escala local,» Instituto de Estudios Urbanos y Territoriales UC, Santiago, Chile, 2020. [En línea]. Available: <https://estudiosurbanos.uc.cl/wp-content/uploads/2022/04/TESIS-PVZ.pdf>. [Último acceso: 14 06 2026].
- [15] Banco Interamericano de Desarrollo (BID), «Chile frente al desafío de los desastres naturales: del riesgo estructural a la resiliencia inteligente,» BID, Washington, D.C., EE. UU., 2023. [En línea]. Available: <https://nuevociclopolitico.cl/ensayos/chile-frente-al-desafio-de-los-desastres-naturales-del-riesgo-estructural-a-la-resiliencia-inteligente/>. [Último acceso: 14 06 2026].
- [16] Asociación de Municipalidades de Chile (AMUCH), «Estudio AMUCH: Fortalezas y desafíos del sistema municipal en la Gestión del Riesgo,» Dirección de Estudios AMUCH, Santiago, Chile, 2024. [En línea]. Available: https://amuch.cl/wp-content/uploads/2025/08/Fortalezas-y-desafios-del-sistema-municipal-en-la-Gestion-del-Riesgo-2-3_compressed.pdf. [Último acceso: 14 06 2026].
- [17] Ministerio del Interior y Seguridad Pública, «Ley 21364: Establece el Sistema Nacional de Prevención y Respuesta ante Desastres, sustituye la Oficina Nacional de Emergencia por el Servicio Nacional de Prevención y Respuesta ante Desastres, y adecúa normas que indica,» Biblioteca del Congreso Nacional de Chile, Santiago, 2021.
- [18] MDN Web Docs, «Background Synchronization API,» Mozilla Foundation, 2024. [En línea]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Background_Synchronization_API. [Último acceso: 24 05 2026].
- [19] Google web.dev, «Reduce JavaScript payloads with code splitting,» 2020. [En línea]. Available: <https://web.dev/articles/reduce-javascript-payloads-with-code-splitting>. [Último acceso: 24 05 2026].
- [20] Microsoft Azure, «Architectural approaches for multi-tenant solutions,» Microsoft Learn, 2022. [En línea]. Available: <https://learn.microsoft.com/en-us/azure/architecture/guide/multitenant/approaches>. [Último acceso: 24 05 2026].
- [21] The MITRE Corporation, «CVE - Common Vulnerabilities and Exposures,» 2024. [En línea]. Available: <https://cve.mitre.org/>. [Último acceso: 24 05 2026].
- [22] J. Nielsen, «Usability 101: Introduction to Usability,» Nielsen Norman Group, 2012. [En línea]. Available: <https://www.nngroup.com/articles/usability-101-introduction-to-usability/>. [Último acceso: 24 05 2026].
- [23] Ministerio de Desarrollo Social y Familia, «Ficha básica de emergencia (FIBE) - Ventanilla Única Social,» Gobierno de Chile, 2024. [En línea]. Available: <https://www.ventanillaunicasocial.gob.cl/ficha/329/ficha-basica-emergencia>. [Último acceso: 20 05 2026].

Anexos

Tabla A.1. Resultados de sesión de prueba guiada.

Fuente. Elaboración propia

Nº	Tarea Solicitada	Tiempo (seg)	Asistencia Requerida	Error Observado / Fricción	Calificación (Likert 1-7)
1	Creación de usuario	75s	Ninguna	Ninguno	7/7
2	Creación de centro de acopio	240s	Ninguna	Sugiere utilizar negrita para los títulos del formulario	6/7
3	Eliminación de centro de acopio	45s	Ninguna	Ninguno	7/7
4	Asignación de centro	60s	Ninguna	Ninguno	7/7
5	Actualización de inventario: Agua	60s	Ninguna	Ninguno	7/7
6	Actualización de inventario: Aseo	60s	Ninguna	Ninguno	7/7
7	Ingreso de damnificados (Ficha FIBE)	110s	Ninguna	Retraso visual (dificultad para ubicar el módulo en la interfaz).	6/7
8	Revisar notificación de centro desactivado	100s	Ninguna	Discrepancia lógica (buscó en el buzón principal en lugar del detalle del centro).	5/7

Tabla A.2. Resultados de Auditoría de Accesibilidad por Vista.

Fuente. Elaboración propia

Vista Evaluada	Puntaje	Errores Principales Detectados	Criterio WCAG Afectado
Inicio (Home)	89/100	Errores menores de contraste visual en textos secundarios.	WCAG 1.4.3 (Contraste Mínimo)
Gestión de Centros	82/100	Bajo contraste en insignias de estado; ausencia de etiquetas aria-label en switches; elementos de lista () huérfanos; saltos estructurales en encabezados.	WCAG 1.4.3 (Contraste), WCAG 4.1.2 (Nombre y Función), WCAG 1.3.1 (Información y Relaciones)
Gestión de Inventario	83/100	Bajo contraste en botones de acción ("Exportar", "Gestionar"); ausencia de etiquetas aria-label en selectores de "Prioridad".	WCAG 1.4.3 (Contraste), WCAG 4.1.2 (Nombre y Función)

Tabla A.3. Historias de usuarios definidas como esenciales.

Fuente. Elaboración propia

ID	Categoría	Descripción de la Historia de Usuario
H01	Esencial (E)	Como trabajador de emergencias necesito poder registrar posibles centros y toda su información relacionada para poder tener la información cargada en caso de que ocurra una emergencia.
H05	Esencial (E)	Como trabajador de emergencias necesito aprobar una incidencia pendiente para poder actualizar el catastro preventivo y mantener la validez del albergue o centro.
H06	Esencial (E)	Como trabajador de emergencias necesito rechazar las incidencias para filtrar las solicitudes en caso de que no sean pertinentes o no correspondan a necesidades reales del centro.
H08	Esencial (E)	Como líder de la sección de emergencias necesito desactivar centros para indicar que ya no están disponibles o utilizables operativamente.
H11	Esencial (E)	Como encargado de centro necesito mantener actualizado el inventario de mi centro una vez llega ayuda o más insumos para poder conocer los recursos disponibles y poder comenzar a distribuirla entre los damnificados.
H12	Esencial (E)	Como damnificado/voluntario/miembro de una ONG necesito visualizar en el mapa el estado actualizado de los centros de acopio y albergues para poder estar informado del panorama actual.
H14	Esencial (E)	Como encargado de un centro necesito poder ver el estado actual del inventario de mi albergue comunitario para conocer los recursos disponibles y ayudar y atender a los damnificados con ello.

Figura 1. Acceso no autorizado al panel de administración por ausencia de Guardianes de Rutas.

Fuente. Elaboración propia

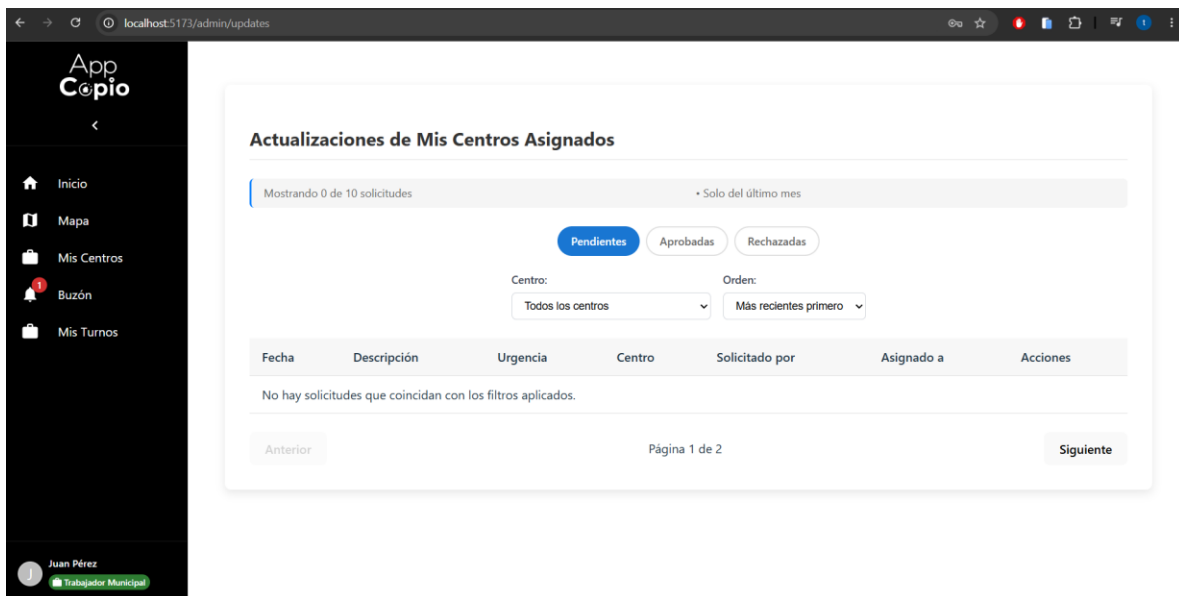


Figura 2. Reporte de rendimiento Lighthouse - Vista de Inicio.
Fuente. Elaboración propia

There were issues affecting this run of Lighthouse:

- There may be stored data affecting loading performance in this location: IndexedDB. Audit this page in an incognito window to prevent those resources from affecting your scores.



Performance

Values are estimated and may vary. The [performance score is calculated](#) directly from these metrics. [See calculator.](#)

▲ 0–49 50–89 90–100



METRICS

[Expand view](#)

First Contentful Paint

0.7 s

Largest Contentful Paint

0.9 s

Figura 3. Reporte de rendimiento Lighthouse - Vista Gestión de Centros.
Fuente. Elaboración propia

There were issues affecting this run of Lighthouse:

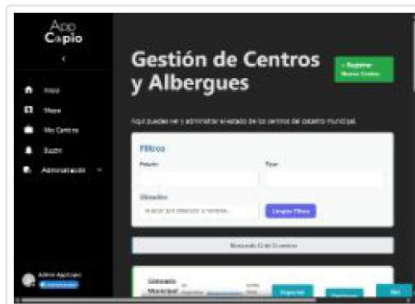
- There may be stored data affecting loading performance in this location: IndexedDB. Audit this page in an incognito window to prevent those resources from affecting your scores.



Performance

Values are estimated and may vary. The [performance score is calculated](#) directly from these metrics. [See calculator.](#)

▲ 0–49 50–89 90–100



METRICS

[Expand view](#)

First Contentful Paint

0.7 s

Largest Contentful Paint

1.0 s

Figura 4. Reporte de rendimiento Lighthouse - Vista Detalle de Centro.
Fuente. Elaboración propia

There were issues affecting this run of Lighthouse:

- There may be stored data affecting loading performance in this location: IndexedDB. Audit this page in an incognito window to prevent those resources from affecting your scores.



Performance

Values are estimated and may vary. The [performance score is calculated](#) directly from these metrics. [See calculator.](#)

▲ 0–49 50–89 90–100



METRICS

[Expand view](#)

First Contentful Paint

0.7 s

Largest Contentful Paint

1.1 s

Tabla A.4. Configuración del Entorno de Prueba de Rendimiento.

Fuente. Elaboración propia

Parámetro	Configuración Utilizada
Herramienta de Medición	Google Lighthouse 13.2.0 (Integrado en DevTools)
Entorno de Despliegue	Local Production Build (Vite)
Navegador	Chromium 149.0.0.0, Modo Incógnito
Hardware del Equipo	Procesador Intel Core i5-11400, 24GB RAM, Windows 11
Configuración Lighthouse	Modo: Navegación / Dispositivo: Desktop Emulado (Sin <i>throttling</i> de red simulado)
Fecha de Medición	14 de junio de 2026

Tabla A.5. Funciones y Componentes con Complejidad Ciclomática Crítica (CC > 20).

Fuente. Elaboración propia

Función/ Componente	Archivo (Ruta)	Complejidad Ciclomática (CC)	Riesgo Técnico Detectado
CenterResidentsPage	/pages/CenterResidentsPage/CenterResidentsPage.tsx	73	Función gigante con múltiples responsabilidades; dificulta pruebas unitarias y mantenimiento.
DatabaseDetailPage	/pages/Databases/DatabaseDetailPage.tsx	65	Lógica compleja de gestión de datos fuertemente acoplada a la vista.
InventoryPage	/pages/InventoryPage/InventoryPage.tsx	52	Múltiples flujos de manejo de inventario centralizados en un único bloque de código.
ROLE_ID_TMO	/pages/UpdatesPage/UpdatesPage.tsx	38	Alta anidación condicional acoplada a constantes de configuración.
PAGE_SIZE	/pages/UpdatesPage/UpdatesPage.tsx	38	Variables de estado que incrementan exponencialmente los caminos independientes de renderizado.
UpdatesPage	/pages/UpdatesPage/UpdatesPage.tsx	38	Gestión compleja de solicitudes de actualización sin división de subcomponentes.
CenterManagementPage	/pages/CenterManagementPage/CenterManagementPage.tsx	30	Exceso de filtros y opciones procesadas sincrónicamente en el mismo archivo.
CenterDetailsPage	/pages/CenterDetailsPage/CenterDetailsPage.tsx	26	Lógica de visualización de detalles con múltiples renderizados condicionales por secciones.
CreateCenterPage	/pages/CreateCenterPage/CreateCenterPage.tsx	25	Formulario multi-paso altamente complejo operando bajo un mismo estado local.
steps	/pages/CenterEditPage/CenterEditPage.tsx	21	Lógica anidada para la navegación entre pasos de edición.
CenterEditPage	/pages/CenterEditPage/CenterEditPage.tsx	21	Edición monolítica con validaciones múltiples procesadas simultáneamente.