



Desarrollo de un sistema multiplataforma para la gestión colaborativa de proyectos
comunitarios en Bucaramanga, orientada a la participación ciudadana y la
transparencia, 2026

Desarrollo Tecnológico

Jonatahn González Caballero
1005328900

UNIDADES TECNOLÓGICAS DE SANTANDER

Facultad de Ciencias Naturales e Ingenierías

Ingeniería de Sistemas

Bucaramanga 3 de octubre del 2026



Desarrollo de un sistema multiplataforma para la gestión colaborativa de proyectos
comunitarios en Bucaramanga, orientada a la participación ciudadana y la
transparencia, 2026

Desarrollo Tecnológico

Jonatahn González Caballero
1005328900

Trabajo de Grado para optar al título de
Ingeniero de Sistemas

DIRECTOR

Liz Katherinne Castellanos Rodríguez

Grupo de investigación – GRIIS

UNIDADES TECNOLÓGICAS DE SANTANDER

Facultad de Ciencias Naturales e Ingenierías
Ingeniería de Sistemas

Bucaramanga 3 de octubre del 2026

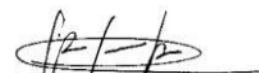
F-DC-125

INFORME FINAL DE TRABAJO DE GRADO EN MODALIDAD DE PROYECTO
DE INVESTIGACIÓN, DESARROLLO TECNOLÓGICO, MONOGRAFÍA,
EMPRENDIMIENTO Y SEMINARIO

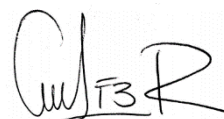
VERSIÓN: 2.0

Nota de Aceptación

Este informe final de trabajo de grado,
En modalidad Desarrollo Tecnológico, fue APROBADO
En cumplimiento de uno de los requisitos exigidos por las
Unidades Tecnológicas de Santander para optar el
Título de Ingeniero de Sistemas,
Según acta No. 26 del 20 de 10 del 2026,
Del Comité de Trabajo de Grado



Firma del Evaluador



Firma del Director

TABLA DE CONTENIDO

RESUMEN EJECUTIVO.....	10
INTRODUCCIÓN.....	12
1. DESCRIPCIÓN DEL TRABAJO DE INVESTIGACIÓN	15
1.1. PLANTEAMIENTO DEL PROBLEMA	15
1.2. JUSTIFICACIÓN.....	17
1.3. OBJETIVOS	18
1.3.1. OBJETIVO GENERAL	18
1.3.2. OBJETIVOS ESPECÍFICOS.....	18
1.4. ESTADO DEL ARTE.....	19
2. MARCO REFERENCIAL	23
2.1. MARCO TEÓRICO	23
2.2. MARCO CONCEPTUAL	37
2.3. MARCO LEGAL	46
2.4. MARCO HISTÓRICO	55
2.5. MARCO AMBIENTAL	57
3. DISEÑO DE LA INVESTIGACIÓN.....	58
3.1. DIAGNÓSTICO INICIAL Y ORIGEN DE LA PROPUESTA.....	58
3.2. MÉTODO DE INVESTIGACIÓN	62
3.3. TÉCNICAS E INSTRUMENTOS DE RECOLECCIÓN DE INFORMACIÓN.....	63
3.4. POBLACIÓN Y MUESTRA	63
3.5. OPERACIONALIZACIÓN DE LAS VARIABLES	64
3.6. CONSIDERACIONES ÉTICAS	65
3.7. PROCEDIMIENTO Y FASES DEL PROYECTO	67
4. DESARROLLO DEL TRABAJO DE GRADO	69
4.1. LEVANTAMIENTO Y ANÁLISIS DE LOS REQUISITOS	69
4.1.1. REQUISITOS FUNCIONALES	69
4.1.2. REQUISITOS NO FUNCIONALES.....	72
4.1.3. HISTORIAS DE USUARIO	75
4.1.4. CASOS DE USO	79

4.2.	DISEÑO DE LA ARQUITECTURA DEL SISTEMA.....	80
4.3.	DISEÑO DE LA BASE DE DATOS	85
4.4.	WIREFRAME	87
4.5.	DESARROLLO DE LOS MÓDULOS DEL SISTEMA.....	88
4.5.1.	PREPARACIÓN DEL ENTORNO DE DESARROLLO.....	90
4.5.2.	IMPLEMENTACIÓN DE LA BASE DE DATOS Y SUPABASE	92
4.5.3.	IMPLEMENTACIÓN DE LOS MÓDULOS.....	120
4.6.	DESPLIEGUE DEL SISTEMA	128
4.6.1.	DESPLIEGUE DE LA APLICACIÓN ANDROID	128
4.6.2.	DESPLIEGUE DE LA APLICACIÓN WEB.....	130
4.7.	PRUEBAS Y VALIDACIÓN	131
4.7.1.	PRUEBAS FUNCIONALES	131
4.7.2.	PRUEBAS DE RENDIMIENTO Y CONCURRENCIA	138
4.7.3.	PRUEBAS DE USABILIDAD.....	143
4.8.	GUÍA DE INSTALACIÓN PARA REPRODUCCIÓN DEL SISTEMA	150
4.8.1.	REQUISITOS PREVIOS	150
4.8.2.	CLONAR Y PREPARAR EL PROYECTO	150
4.8.3.	ESTRUCTURA DEL ARCHIVO .ENV	151
4.8.4.	CREACIÓN DEL PROYECTO EN SUPABASE Y EJECUCIÓN DEL SCRIPT SQL.....	151
4.9.	RESTRICCIONES Y CONSIDERACIONES TÉCNICAS	152
<u>5.</u>	<u>RESULTADOS</u>	<u>153</u>
5.1.	OBJETIVO ESPECÍFICO 1: ANÁLISIS DE REQUISITOS.....	154
5.2.	OBJETIVO ESPECÍFICO 2: DISEÑO DE LA ARQUITECTURA Y DEL MODELO DE DATOS... 154	154
5.3.	OBJETIVO ESPECÍFICO 3: DESARROLLO DEL SISTEMA MULTIPLATAFORMA	155
5.4.	OBJETIVO ESPECÍFICO 4: AUTENTICACIÓN Y CONTROL DE ACCESO	157
5.5.	OBJETIVO ESPECÍFICO 5: EVALUACIÓN CON USUARIOS FINALES	158
5.6.	ANÁLISIS DE RESULTADOS	164
<u>6.</u>	<u>CONCLUSIONES</u>	<u>167</u>
<u>7.</u>	<u>RECOMENDACIONES</u>	<u>169</u>
<u>8.</u>	<u>REFERENCIAS BIBLIOGRÁFICAS</u>	<u>171</u>
<u>9.</u>	<u>APÉNDICES</u>	<u>183</u>
<u>10.</u>	<u>ANEXOS.....</u>	<u>191</u>

LISTA DE FIGURAS

Figura 1. Primera pregunta de encuesta	59
Figura 2. Segunda pregunta de encuesta	60
Figura 3. Tercera pregunta de encuesta	60
Figura 4. Cuarta pregunta de encuesta.....	61
Figura 5. Quinta pregunta de encuesta	62
Figura 6. Diagrama de Casos de Uso	79
Figura 7. Diagrama de la Arquitectura del sistema.....	81
Figura 8. Diagrama de la arquitectura lógica del sistema	83
Figura 9. Diagrama arquitectura física del sistema.....	84
Figura 10. Diagrama del diseño de la base de datos.....	86
Figura 11. Wireframe para su versión móvil	87
Figura 12. Wireframe para su versión web	88
Figura 13. Creación de una nueva Organización en Supabase.....	92
Figura 14. Especificaciones generales del plan gratis en Supabase	93
Figura 15. Especificaciones detalladas del plan gratis en Supabase.....	93
Figura 16. Creación de la tabla de roles.....	96
Figura 17. Creación de la tabla usuarios	97
Figura 18. Creación de la tabla categorías	98
Figura 19. Creación de la tabla propuestas.....	99
Figura 20. Creación de la tabla propuestas_archivos.....	100
Figura 21. Creación de la tabla de votaciones.....	100
Figura 22. Creación de la tabla de las opciones de votación.....	101
Figura 23. Creación de la tabla registros de votos	102
Figura 24. Creación de la tabla de votos	102
Figura 25. Creación de la tabla de proyectos	103
Figura 26. Creación de la tabla actualizaciones de proyectos	104
Figura 27. Creación de la tabla proyecto archivos.....	105
Figura 28. Creación de la tabla notificaciones.....	105
Figura 29. Creación de la tabla auditoria.....	106
Figura 30. Función de sincronización autenticación y perfil.....	107
Figura 31. Función actualización automática marca temporal	108
Figura 32. Función porcentaje de avance de un proyecto	108
Figura 33. Política del estado pendiente	110
Figura 34. Función de emitir voto	111
Figura 35. Función de emitir voto 2	112
Figura 36. Creación de la vista de resultados de votación	113
Figura 37. Creación de la vista perfiles publicos	113
Figura 38. Creación política de seguridad para propuestas.....	114
Figura 39. Automatización de las votaciones programadas.....	115
Figura 40. Conexión con Supabase	116
Figura 41. Inicialización del usuario	117
Figura 42. Patrón y características del repositorio.....	118

Figura 43.	Configuración del proveedor de repositorio de propuestas.....	118
Figura 44.	Traducciones de errores en Supabase.....	119
Figura 45.	Gestión de cambios de autenticación mediante un StreamProvider	120
Figura 46.	Función crear propuesta del módulo de registros.....	121
Figura 47.	Interfaz registrar nueva propuesta.....	121
Figura 48.	Interfaz propuesta de la comunidad	122
Figura 49.	Función emitir voto del módulo votaciones.....	122
Figura 50.	Interfaz módulo de votaciones.....	123
Figura 51.	Interfaz resultados de una votación.....	123
Figura 52.	Función registrar avance del módulo proyectos	124
Figura 53.	Interfaz seguimiento de proyecto	124
Figura 54.	Interfaz registrar un avance de proyecto	125
Figura 55.	Función obtener métricas del módulo métricas	125
Figura 56.	Interfaz módulo métricas	126
Figura 57.	Interfaz módulo métricas parte gráfica	126
Figura 58.	Función Iniciar sesión del módulo autenticación.....	127
Figura 59.	Interfaz de la sección perfil del usuario	127
Figura 60.	Comando keytool	128
Figura 61.	Configuración de la firma en el proyecto	129
Figura 62.	Comando para generar el .apk.....	129
Figura 63.	Permiso de internet para su versión Android	130
Figura 64.	Comando para generar su versión web.....	130
Figura 65.	Publicación en Vercel.....	130
Figura 66.	Enrutamiento a una sola página.....	131
Figura 67.	Creación de una propuesta por un usuario.....	158
Figura 68.	Usuario en la pantalla inicial de la aplicación	159
Figura 69.	Usuario admin en la sección de métricas	160
Figura 70.	Registro de un usuario en la aplicación	161
Figura 71.	Manual de Usuario 1	221
Figura 72.	Manual de Usuario 2	221
Figura 73.	Manual de Usuario 3	222
Figura 74.	Manual de Usuario 4	222
Figura 75.	Manual de Usuario 5	223
Figura 76.	Manual de Usuario 6	223
Figura 77.	Manual de Usuario 7	224
Figura 78.	Manual de Usuario 8	224
Figura 79.	Manual de Usuario 9	225
Figura 80.	Manual de Usuario 10	225
Figura 81.	Manual de Usuario 11	226
Figura 82.	Manual de Usuario 12	226
Figura 83.	Manual de Usuario 13	227
Figura 84.	Manual de Usuario 14	227
Figura 85.	Manual de Usuario 15	228
Figura 86.	Política de Tratamiento de Datos Personales.....	229
Figura 87.	Documento Política de Tratamiento de Datos Personales.....	229
Figura 88.	Evidencias aceptación Políticas de Tratamiento de Datos Personales.....	230

Figura 89. Formato Autorización Uso de Imagen	231
--	-----

LISTA DE TABLAS

Tabla 1.	<i>Operacionalización de las variables del estudio.....</i>	64
Tabla 2.	<i>Componentes del sistema</i>	84
Tabla 3.	<i>Fases del desarrollo según el historial de commits</i>	89
Tabla 4.	<i>Entorno de Desarrollo</i>	90
Tabla 5.	<i>Paquetes directos (dependencias).....</i>	90
Tabla 6.	<i>Conceptos de partes del sistema.....</i>	95
Tabla 7.	<i>Características de la tabla roles.....</i>	96
Tabla 8.	<i>Características de la tabla Usuarios</i>	97
Tabla 9.	<i>Características de la tabla categorías</i>	98
Tabla 10.	<i>Características de la tabla propuestas</i>	99
Tabla 11.	<i>Características de la tabla propuestas archivos</i>	100
Tabla 12.	<i>Características de la tabla votaciones.....</i>	101
Tabla 13.	<i>Características de la tabla de las opciones de votación</i>	101
Tabla 14.	<i>Características de la tabla registros de votos.....</i>	102
Tabla 15.	<i>Características de la tabla votos</i>	103
Tabla 16.	<i>Características de la tabla proyectos</i>	103
Tabla 17.	<i>Características de la tabla actualizaciones de proyectos</i>	104
Tabla 18.	<i>Características de la tabla proyectos archivos</i>	105
Tabla 19.	<i>Características de la tabla notificaciones</i>	105
Tabla 20.	<i>Características de la tabla de auditoría.....</i>	106
Tabla 21.	<i>Permisos de usuario según el Rol</i>	109
Tabla 22.	<i>Funciones de apoyo</i>	111
Tabla 23.	<i>Políticas de acceso a los buckets de almacenamiento</i>	114
Tabla 24.	<i>Matriz de validación para pruebas funcionales</i>	131
Tabla 25.	<i>Alcance de la medición del objetivo 5</i>	137
Tabla 26.	<i>Escenarios de la prueba</i>	139
Tabla 27.	<i>Tiempos de respuesta con un solo usuario.....</i>	139
Tabla 28.	<i>Concurrencia: lectura simultánea de propuestas</i>	139
Tabla 29.	<i>Concurrencia: votación simultánea e intento de voto doble</i>	140
Tabla 30.	<i>Bitácora de defectos</i>	142
Tabla 31.	<i>Protocolo de tareas.....</i>	144
Tabla 32.	<i>Protocolo de tareas.....</i>	146
Tabla 33.	<i>Resultados por participante</i>	146
Tabla 34.	<i>Tabla de aceptación escala SUS</i>	147
Tabla 35.	<i>Resumen de los resultados</i>	147
Tabla 36.	<i>Respuestas por ítem.....</i>	147
Tabla 37.	<i>Matriz de cumplimiento por objetivo específico</i>	153
Tabla 38.	<i>Resultados de las pruebas funcionales por área.....</i>	156
Tabla 39.	<i>Resultados de la encuesta de evaluación (n = 10).....</i>	161
Tabla 40.	<i>Resultados de la medición posterior (n = 10)</i>	163
Tabla 41.	<i>Cumplimiento de los entregables</i>	164

RESUMEN EJECUTIVO

El objetivo fue desarrollar un sistema multiplataforma (web y móvil) para la gestión colaborativa de proyectos comunitarios en Bucaramanga durante 2026, con Flutter y Supabase, para apoyar la participación ciudadana, la transparencia y la toma de decisiones.

Se siguió un enfoque cuantitativo, descriptivo y aplicado, con los métodos deductivo y de análisis y técnicas de revisión documental, observación y encuesta. Las fases fueron diagnóstico (encuesta en línea a 15 personas), levantamiento de requisitos, modelamiento, diseño de la arquitectura y de la base de datos, implementación, pruebas y validación.

Se obtuvo una plataforma que funciona en navegador web y en Android desde una misma base de código, con Supabase y PostgreSQL. Tiene cuatro módulos principales (propuestas, votación digital, seguimiento de proyectos y métricas de participación) y tres complementarios (notificaciones, administración de usuarios y auditoría). Las 86 pruebas funcionales ejecutadas dieron el resultado esperado. En la validación, 10 miembros de la comunidad del Mirador de Arenales usaron la aplicación y respondieron un cuestionario: 8 de 10 (80 %) se declararon satisfechos o muy satisfechos, con lo que se alcanzó el umbral fijado, y la facilidad de uso tuvo la menor aceptación (70 %).

Se concluyó que las reglas críticas, ubicadas en la base de datos y no en Flutter, se aplican también a peticiones ajenas a la interfaz. Separar la participación de la elección en dos tablas permitió el voto único y secreto frente a los roles de la aplicación, pero no frente a quien acceda directamente a la base de datos. La

validación mostró una aceptación inicial en un solo grupo de 10 personas y no permite afirmar efectos sobre la participación, la transparencia ni la eficiencia.

PALABRAS CLAVE: Participación ciudadana, gestión colaborativa, desarrollo multiplataforma, transformación digital, proyectos comunitarios.

INTRODUCCIÓN

La transformación digital ha generado nuevas posibilidades para fortalecer la relación entre la ciudadanía y las instituciones, especialmente mediante herramientas que facilitan el acceso a la información, la participación y el seguimiento de los procesos públicos. En este contexto, el gobierno abierto plantea la necesidad de promover mecanismos de transparencia, rendición de cuentas, participación ciudadana y colaboración entre diferentes actores para la generación de valor público (Naser, 2018). Asimismo, la participación ciudadana constituye un elemento estratégico para la construcción de procesos de gestión pública más inclusivos y orientados a las necesidades de la población (Naser et al., 2021).

En América Latina, la incorporación de tecnologías digitales en los procesos participativos representa una oportunidad para ampliar los mecanismos tradicionales de interacción entre ciudadanía e instituciones. La Comisión Económica para América Latina y el Caribe señala que la transformación digital puede contribuir al desarrollo de procesos participativos y de gobernanza colaborativa, siempre que las herramientas implementadas respondan a criterios de inclusión, legitimidad, accesibilidad y sostenibilidad (Torres et al., 2025). De esta manera, las tecnologías de información pueden funcionar como instrumentos para facilitar la comunicación, la deliberación y el seguimiento de iniciativas colectivas.

En el contexto de Bucaramanga, la participación ciudadana constituye un componente reconocido dentro de la gestión pública municipal. La Alcaldía de Bucaramanga dispone de espacios destinados a informar sobre mecanismos y acciones de participación ciudadana durante las etapas de diagnóstico, formulación, implementación, evaluación y seguimiento de las acciones públicas (Alcaldía de

Bucaramanga, s. f.). Sin embargo, la existencia de estos mecanismos no elimina la necesidad de desarrollar soluciones tecnológicas que permitan integrar en un mismo entorno procesos relacionados con la presentación de propuestas, participación en decisiones, seguimiento de iniciativas y consulta de información.

A partir de esta situación surgió la necesidad de desarrollar una solución tecnológica que contribuyera a centralizar y organizar los procesos asociados con la gestión colaborativa de proyectos comunitarios urbanos. El aporte de este trabajo radicó en integrar diferentes mecanismos de participación dentro de una plataforma web y móvil, con el fin de facilitar la interacción de los usuarios con las iniciativas comunitarias y proporcionar información que permitiera realizar seguimiento a su desarrollo. La propuesta contempló módulos para la recolección de propuestas, votación digital, seguimiento de proyectos y análisis de métricas de participación, con el propósito de fortalecer la transparencia y apoyar la toma de decisiones colectivas.

Para la implementación de la solución se empleó un método de desarrollo de software de carácter computacional, estructurado en las etapas de levantamiento de requisitos, análisis, diseño, desarrollo, pruebas y validación. La construcción de la plataforma utilizó Flutter, tecnología que permitía desarrollar aplicaciones multiplataforma a partir de una misma base de código (Flutter, s. f.-b). Como plataforma de servicios backend se utilizó Supabase, que proporcionaba funcionalidades de autenticación, autorización e integración con bases de datos PostgreSQL, y permitía gestionar la información y el acceso a los recursos del sistema (Supabase, 2026a).

En consecuencia, el presente trabajo buscó aportar una alternativa tecnológica orientada a la gestión colaborativa de proyectos comunitarios en contextos urbanos

de Bucaramanga, que articula los principios de participación ciudadana y transformación digital con herramientas de desarrollo de software multiplataforma. El desarrollo y la validación de la plataforma permitieron determinar su funcionamiento y pertinencia frente a las necesidades identificadas, así como establecer oportunidades de mejora para futuras implementaciones.

1. DESCRIPCIÓN DEL TRABAJO DE INVESTIGACIÓN

1.1. PLANTEAMIENTO DEL PROBLEMA

En los contextos urbanos de Bucaramanga, la gestión de proyectos comunitarios presenta limitaciones estructurales derivadas de la insuficiencia de mecanismos integrados que promuevan una participación ciudadana efectiva y una transparencia real en los procesos de decisión. Esta debilidad surge principalmente de la persistencia de procesos manuales y el uso de herramientas digitales fragmentadas, lo que genera baja trazabilidad, escasa articulación entre actores y dificultades para realizar seguimiento continuo de las iniciativas. Según la Comisión Económica para América Latina y el Caribe (CEPAL, 2022), la limitada integración de tecnologías digitales en la gestión local restringe la capacidad de incidencia ciudadana y debilita los procesos de gobernanza participativa. Asimismo, la restringida disponibilidad de sistemas que incorporen mecanismos de votación verificables y análisis básico de métricas de participación agrava la desinformación y reduce la representatividad de las iniciativas comunitarias. Como consecuencia, estas limitaciones erosionan la confianza mutua entre la ciudadanía y las autoridades locales, lo que reduce la legitimidad de los proyectos comunitarios y obstaculiza el avance hacia una gobernanza más inclusiva y eficiente en el territorio bumangués (Zea, 2023).

Estas limitaciones afectan directamente tres dimensiones evaluables del proceso de gestión comunitaria: el nivel de participación ciudadana, la transparencia en la toma de decisiones y la eficiencia en la gestión y seguimiento de los proyectos comunitarios, las cuales constituyen variables clave para analizar el impacto de soluciones tecnológicas orientadas a la transformación digital en contextos urbanos.

En este problema intervienen cuatro actores. Los ciudadanos de los barrios proponen iniciativas, votan para priorizarlas y esperan información sobre su avance; al depender de procesos manuales y canales dispersos, su participación es difícil de registrar y verificar. Los líderes comunales articulan a la comunidad con las instituciones, recogen las necesidades del sector y hacen seguimiento a los proyectos; sin un registro único y trazable, su gestión depende de la información que logren consolidar por su cuenta. Las entidades municipales reciben las iniciativas, deciden sobre ellas y responden por su ejecución; la fragmentación de los canales les dificulta conocer el respaldo real de cada propuesta y rendir cuentas sobre su avance. El administrador de la plataforma gestiona los usuarios, las propuestas, las votaciones y los proyectos, y garantiza que el sistema opere con reglas de acceso definidas. Los tres primeros actores comparten el problema de la baja trazabilidad y la escasa transparencia; el administrador es quien opera la solución que se propone.

¿De qué manera el desarrollo de un sistema multiplataforma para la gestión colaborativa de proyectos comunitarios incide en los niveles de transparencia, participación ciudadana y eficiencia en la toma de decisiones colectivas en contextos urbanos de Bucaramanga durante el año 2026?

1.2. JUSTIFICACIÓN

La propuesta se planteó como respuesta a las limitaciones identificadas en los procesos de gestión de proyectos comunitarios en contextos urbanos de Bucaramanga, donde la insuficiencia de herramientas tecnológicas integradas restringe la transparencia, la participación ciudadana y la eficiencia en la toma de decisiones colectivas. En este sentido, se identificaron la necesidad de desarrollar una solución tecnológica que permita estructurar, centralizar y optimizar dichos procesos mediante el uso de sistemas digitales accesibles y escalables.

Desde el punto de vista tecnológico, la implementación de un sistema multiplataforma basada en herramientas como Flutter (s. f.-b) y Supabase (s. f.-c) posibilita la integración de funcionalidades orientadas a la recolección de propuestas, votación digital, seguimiento de proyectos y análisis de métricas, lo que permite mejorar la trazabilidad, disponibilidad de la información y soporte para la toma de decisiones basada en datos. A nivel social, se esperó que la propuesta contribuya al fortalecimiento de la participación ciudadana activa y al incremento de la confianza en los procesos comunitarios, al promover entornos más transparentes e inclusivos (Jiménez & Arce, 2024).

En términos económicos, la digitalización de estos procesos reduce costos operativos asociados a la gestión manual de la información y optimiza el uso de recursos en la ejecución de proyectos comunitarios (Vargas & Cruz, 2024). Desde la perspectiva ambiental, la disminución del uso de papel y la optimización de procesos contribuyen a prácticas más sostenibles (Gutiérrez & Romero, 2024). Asimismo, el valor teórico del proyecto radica en la aplicación de conceptos de transformación digital y gobernanza participativa en contextos locales, y aporta al desarrollo de soluciones tecnológicas con impacto social.

La relevancia de este proyecto para la Institución Universitaria Tecnológica de Santander (UTS) se fundamenta en su contribución al Grupo de Investigación en Ingeniería de Software (GRIIS), innovación tecnológica y transformación digital, lo que fortalece la formación académica mediante la aplicación práctica de conocimientos en un contexto real. De igual forma, el desarrollo del proyecto permite al estudiante consolidar competencias en diseño e implementación de sistemas distribuidos, y aporta soluciones aplicables a problemáticas sociales actuales.

1.3. OBJETIVOS

1.3.1. OBJETIVO GENERAL

Desarrollar un sistema multiplataforma (web y móvil) para la gestión colaborativa de proyectos comunitarios en Bucaramanga, mediante el uso de tecnologías como Flutter y Supabase, con el fin de mejorar la participación ciudadana, la transparencia y la eficiencia en la toma de decisiones durante el año 2026.

1.3.2. OBJETIVOS ESPECÍFICOS

1. Analizar los requerimientos funcionales y no funcionales del sistema, a partir de la caracterización de los procesos de participación ciudadana, con el fin de definir el alcance de la solución.
2. Diseñar la arquitectura del sistema y el modelo mediante un enfoque basado en servicios, que garantice escalabilidad, seguridad y disponibilidad de la información.
3. Desarrollar el sistema multiplataforma con los módulos de gestión de propuestas, votación digital, seguimiento de proyectos y visualización de métricas, mediante el uso de Flutter y Supabase.

4. Implementar mecanismos de autenticación, control de acceso y gestión de datos que aseguren la integridad, confidencialidad y disponibilidad de la información.
5. Evaluar el desempeño de la plataforma mediante pruebas funcionales y de usabilidad, con el fin de verificar su funcionamiento y medir su impacto en la participación y transparencia.

1.4. ESTADO DEL ARTE

En el contexto de la transformación digital, los sistemas de participación ciudadana han adquirido relevancia como mecanismos para fortalecer la gobernanza, la transparencia y la toma de decisiones colectivas. Estas herramientas permiten la interacción directa entre ciudadanos y entidades, y facilitan la recolección de propuestas, la deliberación y la votación digital en entornos urbanos. Según Sierra Fructuoso et al. (2024), los sistemas digitales de participación se han consolidado como instrumentos clave del gobierno abierto, y promueven la colaboración y el acceso a la información en procesos democráticos.

A nivel aplicado, existen diversas soluciones tecnológicas orientadas a la gestión colaborativa. Un ejemplo es la plataforma de participación ciudadana desarrollada por Smart City Soft (s. f.), la cual integra funcionalidades de consulta pública, interacción comunitaria y acceso multiplataforma, y permite a los ciudadanos participar desde dispositivos móviles y web.

Consul es una plataforma open-source líder en participación ciudadana, utilizada en más de 250 ciudades y organizaciones a nivel mundial, entre ellos varios países de América Latina como Uruguay, Brasil y Colombia. Permite a los ciudadanos iniciar debates públicos, presentar propuestas de desarrollo urbano,

participar en votaciones y realizar presupuestos participativos. Su arquitectura se basa en Ruby on Rails y PostgreSQL, lo que la hace robusta, pero requiere servidores dedicados y conocimientos avanzados de despliegue. Ofrece funcionalidades como verificación de usuarios, seguimiento de implementación de proyectos y exportación de datos abiertos. Sin embargo, no es nativamente móvil y su curva de aprendizaje para administradores locales puede ser alta en contextos de bajo presupuesto. Representa un referente consolidado en democracia digital, aunque su escalabilidad en ciudades medianas depende de adaptaciones locales significativas (Consul Project, 2025).

CitizenLab es una plataforma de democracia digital que facilita la co-creación comunitaria mediante propuestas, discusiones, votaciones y priorización de proyectos. Cuenta con interfaz web y móvil responsive, disponible en más de 20 idiomas, y ha sido implementada en más de 300 gobiernos y organizaciones. Su versión open-source (disponible en GitHub bajo licencia AGPL) permite hospedaje propio, aunque las funcionalidades premium requieren pago. Incluye herramientas de engagement como encuestas, presupuestos participativos y análisis de datos. Se destaca por su enfoque en inclusión y usabilidad, pero su modelo SaaS limita la adopción en municipios con recursos reducidos. Es un ejemplo claro de cómo las plataformas modernas integran participación en todo el ciclo de proyectos comunitarios (CitizenLab, 2025).

Elector es una aplicación móvil de votación electrónica desarrollada completamente con Flutter y Firebase como backend. Incluye un proceso de verificación en tres pasos, encriptación end-to-end y visualización de resultados en tiempo real. Los usuarios pueden registrar propuestas y emitir un voto único de forma segura, con historial de votaciones. Su diseño mobile-first la hace accesible desde dispositivos básicos, lo que demuestra la viabilidad de Flutter para sistemas

electorales y de participación. Sin embargo, al usar Firebase, depende de la infraestructura de Google y puede generar costos a medida que escala. Sirve como referencia técnica directa para implementar votación segura y auditable en aplicaciones multiplataforma (Ghale, 2025).

Decidim es una plataforma de participación ciudadana originada en Barcelona y adaptada en múltiples países de América Latina. Permite crear procesos participativos, asambleas, iniciativas ciudadanas, consultas y presupuestos participativos. Sus componentes modulares incluyen propuestas, votaciones, encuestas, reuniones presenciales, seguimiento de resultados y comentarios. Está desarrollada en Ruby on Rails y enfatiza transparencia, trazabilidad y exportación de datos abiertos. Su modelo de gobernanza comunitaria (Metadecidim) permite que los propios usuarios mejoren el código. Es una de las soluciones más completas para gestión integral de proyectos comunitarios, aunque su implementación requiere recursos técnicos que no siempre están disponibles en ciudades medianas (Decidim, 2019).

DemocracyOS es una plataforma argentina open-source diseñada para deliberación y votación ciudadana sobre legislación. Permite a los ciudadanos debatir proyectos de ley, proponer modificaciones y votar de forma vinculante o consultiva. Incluye herramientas de crowd law-making, seguimiento de iniciativas y visualización de resultados. Se centra en la transparencia y en cerrar la brecha entre representantes y ciudadanía. Ha sido utilizada en varios países de América Latina y se integra fácilmente con procesos institucionales. Su enfoque en deliberación profunda la diferencia de plataformas más orientadas a votación rápida, aunque su interfaz web tradicional limita la adopción móvil en entornos de baja conectividad (Democracy Technologies, s. f.).

SupaPoll es una aplicación de votación en tiempo real desarrollada con Next.js y Supabase, con soporte adicional para votación vía SMS mediante Twilio. Permite crear encuestas o propuestas, votar desde web o mensaje de texto y visualizar resultados actualizados en vivo. Utiliza Supabase Realtime para actualizaciones instantáneas y base de datos PostgreSQL. Demuestra la potencia de Supabase para sistemas colaborativos escalables y de bajo costo. Incluye verificación de usuarios y webhooks avanzados. Aunque es más simple que plataformas completas de civic tech, sirve como ejemplo práctico de cómo Supabase habilita funcionalidades de votación y seguimiento en tiempo real con tecnologías modernas (Hookdeck, 2023).

Flutter Voting App es una aplicación móvil de votación desarrollada íntegramente con Flutter y Provider para gestión de estado. Permite a los usuarios votar por candidatos o propuestas, visualizar conteo en tiempo real, gráficos de resultados y declaración automática de ganadores. Utiliza Material Design 3 y arquitectura limpia. Aunque es un proyecto de demostración, ilustra perfectamente las capacidades de Flutter para crear interfaces intuitivas y multiplataforma en sistemas de participación. Se puede extender fácilmente con backend como Supabase. Representa un ejemplo accesible y de código abierto para implementar votación y seguimiento en aplicaciones móviles orientadas a proyectos comunitarios (Islam, 2024).

Sin embargo, se identifica un vacío en soluciones que integren sistemas multiplataforma de bajo costo, orientadas específicamente a contextos urbanos intermedios como Bucaramanga, con enfoque en accesibilidad, escalabilidad y facilidad de implementación.

2. MARCO REFERENCIAL

2.1. Marco teórico

Ingeniería de software

La ingeniería de software es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, la operación y el mantenimiento del software (ISO/IEC/IEEE, 2017b). Abarca la especificación de requisitos, el diseño, la implementación, la validación y la evolución del sistema, así como la gestión del proceso y de los cambios (Pressman & Maxim, 2020; Sommerville, 2016). Su propósito es obtener productos que cumplan requisitos definidos y que puedan evaluarse con criterios de calidad.

Desarrollo de software

El desarrollo de software comprende las actividades que convierten las necesidades de los usuarios en un sistema ejecutable: especificación, diseño e implementación, validación y evolución (Sommerville, 2016). Estas actividades se organizan según un modelo de proceso, que puede ser secuencial, incremental o ágil. La elección del modelo depende de las características y las restricciones del proyecto (Pressman & Maxim, 2020).

Desarrollo multiplataforma

El desarrollo multiplataforma construye aplicaciones que se ejecutan en varios sistemas operativos o entornos a partir de una base de código compartida,

total o parcial. La literatura distingue enfoques como el web, el híbrido, el interpretado y el compilado, que difieren en rendimiento, acceso a las funciones del dispositivo y experiencia de usuario (Biørn-Hansen et al., 2018). La razón de ser de estos enfoques es la fragmentación de dispositivos y plataformas móviles, que obliga a mantener una versión por plataforma cuando se usa el desarrollo nativo (Biørn-Hansen et al., 2018).

Aplicaciones web y móviles

Una aplicación web se usa mediante un navegador y se comunica con servidores remotos por el protocolo HTTP (Fielding & Taylor, 2002). Una aplicación móvil se instala en un dispositivo, como un teléfono o una tableta, y puede acceder a recursos propios del equipo, entre ellos la cámara y las notificaciones (Biørn-Hansen et al., 2018). Ambos tipos de aplicación pueden consultar servicios remotos y bases de datos para procesar y almacenar la información.

Flutter

Flutter es un conjunto de herramientas de código abierto de Google para construir interfaces de usuario y aplicaciones compiladas para móvil, web y escritorio a partir de una sola base de código. Usa el lenguaje Dart y organiza la interfaz como un árbol de widgets, componentes que describen la apariencia y el comportamiento de cada elemento de la pantalla (Flutter, s. f.-a).

Dart

Dart es un lenguaje de programación desarrollado por Google, de tipado estático y orientado a objetos, que es el lenguaje de Flutter. Incluye soporte para

programación asíncrona y puede compilarse a código nativo y a JavaScript, de modo que la misma lógica se ejecuta en móvil y en web (Dart, s. f.).

Frontend

El frontend es la parte de una aplicación con la que interactúa el usuario. Corresponde a la capa de presentación del sistema, que muestra la información, recibe las acciones de la persona y las envía a los servicios que contienen la lógica y los datos (Sommerville, 2016).

Backend

El backend agrupa los componentes que procesan las solicitudes, aplican las reglas de negocio y administran los datos: servidores de aplicación, servicios de autenticación, interfaces de programación y almacenamiento. Corresponde a las capas de lógica de aplicación y de gestión de datos de una arquitectura por capas (Sommerville, 2016). Separarlo del frontend permite asignar responsabilidades distintas a cada componente y modificar uno sin rehacer el otro.

API

Una API, o interfaz de programación de aplicaciones, define las operaciones que un componente de software ofrece a otros y el formato de las solicitudes y respuestas, sin exponer su implementación interna. En la web, las APIs suelen seguir el estilo REST, en el que los recursos se identifican con URL y se manipulan con los métodos del protocolo HTTP (Fielding & Taylor, 2002). Supabase genera de forma automática una API REST sobre las tablas de la base de datos (Supabase, s. f.-b).

Supabase

Supabase es una plataforma de código abierto que ofrece, sobre una base de datos PostgreSQL, servicios de autenticación, una API generada automáticamente, almacenamiento de archivos, funciones y datos en tiempo real. PostgreSQL es el componente central y el resto de los servicios se conecta con ella (Supabase, s. f.-a).

PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional de código abierto (PostgreSQL Global Development Group, s. f.-e). Admite restricciones, transacciones, índices, funciones, disparadores y políticas de seguridad por fila, que permiten aplicar reglas directamente en la base de datos (PostgreSQL Global Development Group, s. f.-b, s. f.-c, s. f.-f).

Base de datos

Una base de datos es una colección de datos relacionados que representa algún aspecto del mundo real y que se diseña, construye y puebla con un propósito específico (Elmasri & Navathe, 2016). Su administración se delega en un sistema gestor, que controla cómo se almacenan, consultan y modifican los datos.

Modelo relacional

El modelo relacional fue propuesto por Codd (1970) y representa los datos como relaciones, que los sistemas implementan como tablas de filas y columnas.

Las asociaciones entre tablas se expresan con claves primarias y foráneas, y las restricciones de integridad limitan los valores válidos (Silberschatz et al., 2020).

Sistema gestor de bases de datos

Un sistema gestor de bases de datos (SGBD) es el software que permite definir, crear, consultar, actualizar y administrar una base de datos. También controla el acceso, la ejecución concurrente de operaciones y la recuperación ante fallos (Elmasri & Navathe, 2016; Silberschatz et al., 2020). PostgreSQL es el SGBD que usa Supabase para almacenar la información (Supabase, s. f.-a).

Integridad de datos

La integridad de datos es la propiedad por la cual la información almacenada se mantiene exacta, consistente y válida. En una base de datos relacional se aplica con restricciones (claves primarias, claves foráneas, valores únicos, no nulos y condiciones de verificación) y con funciones y disparadores que rechazan los cambios inválidos (Silberschatz et al., 2020; PostgreSQL Global Development Group, s. f.-b).

Clave primaria y clave foránea

Una clave primaria es un atributo o conjunto de atributos que identifica de forma única cada fila de una tabla. Una clave foránea es un atributo que referencia la clave primaria, o una clave candidata, de otra tabla. Con las claves foráneas se mantiene la integridad referencial, es decir, que no existan referencias a registros que no existen (Silberschatz et al., 2020; PostgreSQL Global Development Group, s. f.-b).

Índices

Un índice es una estructura auxiliar que permite al gestor localizar filas sin recorrer toda la tabla. Acelera las consultas que filtran, ordenan o relacionan por las columnas indexadas, pero ocupa espacio y debe actualizarse en cada inserción, modificación o eliminación (PostgreSQL Global Development Group, s. f.-d).

Transacciones

Una transacción es un conjunto de operaciones sobre una base de datos que se ejecuta como una unidad lógica: se aplican todas o ninguna. Sus propiedades se resumen en la sigla ACID (atomicidad, consistencia, aislamiento y durabilidad), introducida por Härder y Reuter (1983) y adoptada por los gestores relacionales (Silberschatz et al., 2020). En PostgreSQL, las sentencias entre BEGIN y COMMIT forman una transacción, y ROLLBACK las revierte (PostgreSQL Global Development Group, s. f.-g).

Funciones de PostgreSQL

Una función de PostgreSQL es una rutina almacenada en la base de datos que recibe parámetros, ejecuta una lógica definida en un lenguaje como SQL o PL/pgSQL y devuelve un resultado (PostgreSQL Global Development Group, s. f.-c). Permite encapsular operaciones y reglas en el servidor, de modo que se apliquen sin importar qué cliente invoque la base de datos.

Triggers

Un trigger, o disparador, hace que la base de datos ejecute automáticamente una función cuando ocurre un evento sobre una tabla, como una inserción, una actualización o una eliminación. Puede activarse antes o después del evento (PostgreSQL Global Development Group, s. f.-h). Se usa para mantener datos derivados, registrar cambios o validar reglas que deben cumplirse en toda modificación.

Autenticación

La autenticación es el proceso por el cual una persona o aplicación demuestra su identidad a un sistema, por ejemplo con una contraseña o un autenticador criptográfico, antes de recibir acceso a un recurso (Temoshok et al., 2025). Al asociar cada operación con una identidad verificada, permite atribuir y controlar las acciones realizadas.

Autenticación en Supabase

Supabase Auth es el servicio de autenticación de Supabase. Gestiona el registro de usuarios, los inicios de sesión y las sesiones, y emite tokens que identifican al usuario en las solicitudes posteriores. Esa identidad puede consultarse desde las políticas de acceso de la base de datos para decidir qué operaciones corresponden a cada usuario (Supabase, 2026a).

Autorización

La autorización es el proceso que decide qué acciones y qué recursos puede usar una identidad ya autenticada. La autenticación establece quién hace la solicitud y la autorización determina qué puede hacer. Puede basarse en roles (Ferraiolo et

al., 2001), en atributos del usuario y del recurso o en reglas de política (Hu et al., 2014).

Políticas de seguridad

Una política de control de acceso es un conjunto de reglas que establece quién puede acceder a qué recurso y bajo qué condiciones (Hu et al., 2014). En una base de datos, estas reglas pueden considerar al usuario autenticado, el registro solicitado y la operación que se intenta, y aplicarse en la capa más cercana a los datos (PostgreSQL Global Development Group, s. f.-f).

Row Level Security (RLS)

Row Level Security (RLS), o seguridad a nivel de fila, es un mecanismo de PostgreSQL que restringe, mediante políticas, las filas que cada usuario puede consultar o modificar en una tabla. Las políticas pueden diferenciar las operaciones SELECT, INSERT, UPDATE y DELETE, y el motor de la base de datos las aplica antes de devolver o cambiar las filas (PostgreSQL Global Development Group, s. f.-f). En Supabase, RLS es el mecanismo principal para controlar el acceso a los datos desde las aplicaciones cliente (Supabase, 2026b).

Políticas RLS en Supabase

Las políticas RLS de Supabase son reglas definidas con RLS sobre las tablas de PostgreSQL. Usan la identidad del usuario autenticado y los valores de cada fila para permitir o rechazar una operación. Por esa razón, el control de acceso no depende de las restricciones de la interfaz: se aplica sobre los datos aunque la solicitud no provenga de la aplicación (Supabase, 2026b).

Seguridad de la información

La seguridad de la información es la protección de la información y de los sistemas que la contienen frente al acceso, el uso, la divulgación, la alteración o la destrucción no autorizados, con el fin de preservar su confidencialidad, integridad y disponibilidad (National Institute of Standards and Technology [NIST], 2004). Estas propiedades se implementan con controles técnicos, procedimientos de gestión y controles de acceso.

Interfaz de usuario

La interfaz de usuario es el conjunto de elementos con los que una persona interactúa con un sistema: controles, formularios, menús, mensajes y mecanismos de navegación. Su diseño determina cómo se presentan las funciones y los datos y cómo se reciben las acciones del usuario (Shneiderman et al., 2016).

Experiencia de usuario

La experiencia de usuario comprende las percepciones y respuestas de una persona que resultan del uso, o del uso previsto, de un sistema, producto o servicio. Incluye las emociones, preferencias, comportamientos y logros del usuario antes, durante y después del uso (International Organization for Standardization [ISO], 2019).

Diseño responsivo

El diseño responsivo es un enfoque de diseño web, propuesto por Marcotte (2010), en el que una misma página adapta su distribución al tamaño y la resolución de la pantalla mediante rejillas flexibles, imágenes escalables y reglas de estilo condicionadas por el ancho del dispositivo. Con este enfoque una misma interfaz puede usarse en computadores, tabletas y teléfonos.

Control de versiones

El control de versiones es un sistema que registra los cambios sobre un conjunto de archivos a lo largo del tiempo. Permite recuperar versiones anteriores, comparar modificaciones y coordinar el trabajo de varias personas sobre el mismo proyecto (Chacon & Straub, 2014).

Git

Git es un sistema de control de versiones distribuido: cada copia de trabajo contiene el historial completo del proyecto. Permite registrar cambios mediante commits, crear ramas para desarrollar líneas de trabajo paralelas y fusionarlas después (Chacon & Straub, 2014).

GitHub

GitHub es una plataforma de desarrollo colaborativo que permite alojar repositorios basados en Git y proporcionar herramientas para gestionar proyectos de software. Entre sus funciones se encuentran el almacenamiento de código, administración de ramas, revisión de cambios, seguimiento de incidencias y colaboración entre desarrolladores. Los repositorios permiten conservar un historial de las modificaciones realizadas mediante Git (GitHub, s. f.).

Metodologías ágiles

Las metodologías ágiles son enfoques de desarrollo que entregan el software por incrementos funcionales, en ciclos cortos, con revisión frecuente del producto y adaptación a los cambios en los requisitos. Sus principios se formularon en el Manifiesto por el Desarrollo Ágil de Software (Beck et al., 2001) y Sommerville (2016) las presenta como una forma de desarrollo incremental.

Scrum

Scrum es un marco de trabajo para desarrollar y sostener productos complejos. Organiza el trabajo en Sprints, periodos de duración fija, y define el Product Backlog, el Sprint Backlog y el Incremento como artefactos, junto con eventos y responsabilidades para el equipo. Se basa en la transparencia, la inspección y la adaptación (Schwaber & Sutherland, 2020).

Arquitectura de software

La arquitectura de software de un sistema es el conjunto de estructuras necesarias para razonar sobre él: los elementos de software, las relaciones entre ellos y las propiedades de ambos (Bass et al., 2021). La norma ISO/IEC/IEEE 42010 la define como los conceptos o propiedades fundamentales de un sistema en su entorno, plasmados en sus elementos, sus relaciones y los principios de su diseño y evolución (ISO/IEC/IEEE, 2022). La arquitectura condiciona atributos de calidad como el rendimiento, la seguridad, la disponibilidad y la mantenibilidad (Bass et al., 2021; Sommerville, 2016).

Arquitectura cliente-servidor

En la arquitectura cliente-servidor, el sistema se organiza en servicios que ofrecen los servidores y que consumen los clientes a través de una red. El cliente presenta la información y envía las solicitudes; el servidor las procesa y administra los recursos (Sommerville, 2016). Una variante habitual es la arquitectura por capas, en la que cada capa ofrece servicios a la capa superior (Sommerville, 2016).

Computación en la nube

La computación en la nube es un modelo que permite acceder por red, bajo demanda, a un conjunto compartido de recursos informáticos configurables, como redes, servidores, almacenamiento, aplicaciones y servicios, que pueden aprovisionarse y liberarse con poco esfuerzo de gestión (Mell & Grance, 2011).

Sistemas de información

Un sistema de información es un conjunto de componentes interrelacionados que recopilan, procesan, almacenan y distribuyen información para apoyar la toma de decisiones y el control en una organización (Laudon & Laudon, 2022). Tiene dimensiones de gestión, de organización y de tecnología, por lo que no se reduce al software.

Gestión de proyectos

La gestión de proyectos aplica conocimientos, métodos y herramientas a las actividades de un proyecto para cumplir sus objetivos. Incluye la planificación, la

gestión de recursos, el seguimiento del avance y la medición del desempeño (Project Management Institute [PMI], 2021; Sommerville, 2016).

Proyectos comunitarios

Los proyectos comunitarios son iniciativas desarrolladas para atender necesidades, problemas u oportunidades identificadas en una comunidad mediante la intervención coordinada de diferentes actores. Su gestión puede comprender procesos de identificación de necesidades, formulación de propuestas, priorización, ejecución, seguimiento y evaluación. La participación de los integrantes de la comunidad puede intervenir en diferentes etapas de estos procesos (Osorio-Pérez, 2023).

Participación ciudadana digital

La participación ciudadana digital es el uso de tecnologías digitales para que las personas se informen, formulen propuestas, expresen opiniones, colaboren o incidan en decisiones sobre asuntos colectivos. Las plataformas pueden registrar las intervenciones de los participantes y dar acceso a la información de cada proceso (Torres et al., 2025).

Votación digital

La votación digital es el registro y el recuento de las preferencias de los participantes mediante sistemas informáticos. La Recomendación CM/Rec(2017)5 del Consejo de Europa define el voto electrónico como el uso de medios electrónicos para emitir y contar los votos, y exige respetar los principios del sufragio universal, igual, libre y secreto (Comité de Ministros del Consejo de Europa, 2017). En una

plataforma, esto obliga a definir cómo se identifica a los participantes, cómo se limita el voto a uno por persona y cómo se protegen los registros.

Métricas e indicadores de participación

Una métrica es una medida cuantitativa de un atributo de una entidad, y un indicador estima o evalúa un atributo a partir de un modelo de análisis y de criterios de decisión definidos (ISO/IEC/IEEE, 2017a). En este trabajo, las métricas de participación son conteos como el número de propuestas registradas, de votos emitidos y de usuarios registrados, y los indicadores son relaciones entre ellos, como el porcentaje de usuarios activos que votaron. Todo indicador debe documentar los datos que usa, su fórmula y la forma de interpretarlo.

Transformación digital

La transformación digital es un proceso en el que una organización responde a cambios disruptivos con tecnologías digitales y modifica su estrategia, estructura, procesos y cultura para crear valor (Vial, 2019). No equivale a sustituir actividades manuales por herramientas informáticas. En el sector público colombiano, la Política de Gobierno Digital orienta el uso de las TIC para generar valor público y fortalecer la relación entre ciudadanía y Estado (Decreto 1008 de 2018, 2018).

Calidad del software

La calidad del software es el grado en que un producto satisface los requisitos y las necesidades de sus usuarios en un contexto de uso específico. La norma ISO/IEC 25010:2023 define un modelo de calidad del producto con nueve características: adecuación funcional, eficiencia de desempeño, compatibilidad, capacidad de interacción, fiabilidad, seguridad, mantenibilidad, flexibilidad y

protección frente a daños (safety) (ISO/IEC, 2023). La capacidad de interacción es la característica que la edición de 2011 llamaba usabilidad, y la flexibilidad sustituye a la portabilidad. En este trabajo se conserva el término usabilidad, empleado en los requisitos no funcionales. La evaluación de la calidad requiere actividades de verificación y validación durante el ciclo de vida (Sommerville, 2016).

2.2. Marco conceptual

Comunidad

Una comunidad corresponde a un conjunto de personas que mantienen relaciones de interacción y comparten un espacio, intereses, necesidades o situaciones de carácter colectivo. En este trabajo, la comunidad es el ámbito en el que se identifican necesidades y se generan iniciativas para atender situaciones de interés común. Sus integrantes pueden asumir diferentes niveles de participación en la formulación, priorización y seguimiento de dichas iniciativas (Osorio-Pérez, 2023).

Gestión colaborativa

La gestión colaborativa es el proceso mediante el cual diferentes personas o actores intervienen de manera coordinada en actividades orientadas a un objetivo compartido. En el ámbito público, Ansell y Gash (2008) la describen como un arreglo en el que una o más entidades públicas involucran directamente a actores no estatales en un proceso de decisión colectiva formal, orientado al consenso y deliberativo (Ansell & Gash, 2008). Su aplicación implica distribuir la participación entre los involucrados y facilitar el intercambio de información durante las etapas del proceso.

En la gestión de proyectos comunitarios, este concepto comprende actividades como la formulación de iniciativas, su valoración o priorización, la participación en decisiones y el seguimiento de los avances. La plataforma se plantea como un medio para organizar digitalmente estas actividades y centralizar la información asociada con cada iniciativa.

Participación ciudadana

La participación ciudadana comprende la intervención de las personas en asuntos relacionados con su comunidad y en procesos mediante los cuales pueden expresar intereses, necesidades, propuestas o decisiones. Puede presentarse en diferentes momentos, desde la identificación de una problemática hasta el seguimiento de las acciones desarrolladas para atenderla (Defensoría del Pueblo de Colombia, s. f.). Arnstein (1969) distingue grados de participación, que van de la información y la consulta hasta el control ciudadano.

En el contexto de este trabajo, la participación se relaciona con las acciones que los ciudadanos pueden realizar dentro de la plataforma, particularmente el registro de propuestas, la participación en votaciones y la consulta del avance de los proyectos.

Participación digital

La participación digital se refiere a la intervención de las personas en procesos colectivos mediante tecnologías y plataformas digitales. Comprende acciones como registrar información, formular propuestas, expresar preferencias, consultar resultados y acceder a información sobre procesos participativos (Torres et al., 2025).

La participación digital permite trasladar determinadas actividades que tradicionalmente pueden realizarse de manera presencial o mediante mecanismos fragmentados hacia un entorno en el que las interacciones y los registros pueden gestionarse de forma centralizada.

Actor

En el modelado de sistemas, un actor es un rol que desempeña una persona u otro sistema que interactúa con el sistema en estudio (Object Management Group [OMG], 2017). La identificación de actores permite establecer quiénes intervienen en los procesos que el sistema representa y qué tipo de interacción mantienen con sus funcionalidades.

En la plataforma pueden existir actores con diferentes responsabilidades, de acuerdo con las funciones que desempeñen dentro de la gestión de los proyectos comunitarios.

Usuario

El usuario es la persona que interactúa con un producto, sistema o servicio (ISO, 2019). En la plataforma, accede a la información o ejecuta las funcionalidades disponibles según los permisos asignados, por ejemplo consultar proyectos, registrar propuestas, participar en votaciones o seguir iniciativas.

La identificación del usuario permite asociar las operaciones realizadas dentro del sistema con una cuenta determinada y aplicar las condiciones de acceso correspondientes.

Rol

En el control de acceso basado en roles, un rol es una función dentro de una organización a la que se asocian permisos; los usuarios obtienen esos permisos al ser asignados al rol (Ferraiolo et al., 2001; Sandhu et al., 1996). En la plataforma, la definición de roles permite diferenciar las funciones disponibles para cada tipo de participante y restringir ciertas operaciones.

La utilización de roles permite organizar el acceso a las funcionalidades de acuerdo con las responsabilidades definidas para cada tipo de usuario. De esta manera, las acciones disponibles para un participante de la comunidad pueden diferir de aquellas asignadas a un usuario encargado de tareas administrativas.

Propuesta

En este trabajo, una propuesta es una iniciativa planteada por uno o varios participantes para atender una necesidad, resolver una problemática o ofrecer una alternativa a la comunidad. Contiene la información que permite describir la situación identificada, la alternativa planteada y los demás datos necesarios para su evaluación.

Dentro del proceso de gestión comunitaria, las propuestas constituyen una entrada para actividades posteriores de revisión, priorización, votación o transformación en proyectos, de acuerdo con las reglas establecidas para cada proceso.

Proyecto comunitario

Un proyecto comunitario es una iniciativa orientada a atender una necesidad o problemática identificada dentro de una comunidad mediante un conjunto organizado de actividades. Su desarrollo puede involucrar diferentes participantes, recursos, responsables, etapas y mecanismos de seguimiento (Osorio-Pérez, 2023).

Para efectos de la plataforma, el proyecto constituye la unidad principal sobre la cual se organiza información relacionada con su descripción, estado, avances, participantes y demás elementos definidos durante el levantamiento de requisitos (Osorio-Pérez, 2023).

Votación

La votación es el procedimiento mediante el cual los integrantes de un grupo expresan su preferencia entre alternativas definidas y se cuentan esas preferencias para obtener un resultado. En Colombia, la Constitución establece el voto como un derecho y un deber ciudadano (Constitución Política de Colombia, 1991, art. 258).

Dentro de la gestión de iniciativas comunitarias, la votación puede utilizarse para priorizar propuestas o apoyar determinadas decisiones colectivas. El sistema debe asociar cada participación con las condiciones de identificación y validación definidas para garantizar que los registros correspondan a las reglas establecidas.

Seguimiento del proyecto

El seguimiento de un proyecto consiste en comparar el estado real de las actividades, los plazos y los entregables con lo planificado, para detectar

desviaciones y decidir correcciones (PMI, 2021; Sommerville, 2016). Incluye el estado actual, los avances, las actividades realizadas y las fechas relevantes.

El seguimiento permite mantener información actualizada sobre el desarrollo de los proyectos y proporcionar a los participantes un medio para consultar su situación dentro del sistema.

Estado del proyecto

El estado del proyecto indica la etapa del ciclo de gestión en la que se encuentra una iniciativa. Los estados deben corresponder a las etapas definidas para el proceso. En la plataforma, los proyectos pasan por los estados planeado, en ejecución, finalizado y suspendido, y las propuestas por pendiente, en revisión, aprobada, rechazada y convertida en proyecto.

La utilización de estados permite organizar la información de los proyectos y facilitar la consulta de su situación actual. La cantidad y denominación de los estados dependerán de las reglas establecidas para el sistema.

Métrica de participación

Una métrica de participación es un valor obtenido por conteo o medición de la actividad de los participantes en un proceso, como las propuestas registradas, los votos emitidos o los usuarios registrados (ISO/IEC/IEEE, 2017a).

Las métricas permiten transformar registros de actividad en valores que pueden ser consultados y comparados para describir el comportamiento de la participación dentro de la plataforma.

Indicador de participación

Un indicador de participación es una medida que se construye con una o más métricas, según un modelo de cálculo y un criterio de interpretación definidos. A diferencia de un registro individual, produce información agregada, por ejemplo el porcentaje de usuarios activos que votaron (ISO/IEC/IEEE, 2017a).

En la plataforma, los indicadores pueden utilizarse para presentar información agregada sobre la actividad de los participantes, siempre que previamente se definan las variables utilizadas y el método de cálculo correspondiente.

Transparencia

La transparencia se refiere a la disponibilidad y accesibilidad de la información relacionada con los procesos, decisiones y resultados de una actividad, dentro de las condiciones de acceso establecidas. La Ley 1712 de 2014 regula el derecho de acceso a la información pública y presume pública la información en poder de los sujetos obligados, con las excepciones que la propia ley establece (Ley 1712 de 2014, 2014). En los proyectos comunitarios, se relaciona con la posibilidad de consultar información sobre las iniciativas, sus estados, avances y resultados de los procesos participativos.

La transparencia dentro de un sistema digital debe coexistir con las restricciones de acceso necesarias para proteger información que no deba ser pública. Por esta razón, la información disponible debe diferenciarse de aquella que requiere permisos específicos.

Trazabilidad

La trazabilidad es la capacidad de rastrear la historia, la aplicación o la ubicación de un objeto (ISO, 2015). En ingeniería de software, la trazabilidad de requisitos es la capacidad de describir y seguir la vida de un requisito en ambos sentidos, desde su origen hasta su implementación y uso (Gotel & Finkelstein, 1994).

En la gestión de proyectos comunitarios, la trazabilidad puede aplicarse al registro de propuestas, cambios de estado, actualizaciones de información y resultados de procesos participativos, de acuerdo con los mecanismos definidos para el sistema.

Anonimato mediante alias

El uso de alias consiste en identificar públicamente a un participante con un nombre distinto del real. Técnicamente es una forma de seudonimización: el tratamiento de los datos personales de modo que ya no puedan atribuirse a una persona concreta sin información adicional, que se mantiene separada y sujeta a medidas técnicas y organizativas (Reglamento (UE) 2016/679, 2016, art. 4). En la normativa colombiana, un dato personal es la información vinculada o que pueda asociarse a personas naturales determinadas o determinables (Ley 1581 de 2012, 2012, art. 3). Por eso el alias oculta la identidad frente a otros participantes, pero no convierte los datos en anónimos.

En los procesos de propuesta y participación, el uso de alias puede emplearse para reducir la exposición de la identidad frente a otros participantes,

mientras que los mecanismos administrativos pueden conservar la información necesaria para la gestión y control del sistema de acuerdo con las reglas definidas.

Gestión de propuestas

La gestión de propuestas comprende el conjunto de actividades relacionadas con la creación, el registro, la consulta, la organización, la revisión y el seguimiento de las iniciativas planteadas por los participantes. Estructura las propuestas desde su registro inicial hasta las etapas definidas para su evaluación o priorización. Dentro del sistema, requiere conservar información que permita identificar cada iniciativa, consultar su contenido y determinar su situación dentro del proceso.

Toma de decisiones colectivas

La toma de decisiones colectivas es el proceso mediante el cual un grupo selecciona una alternativa o define una acción común. Puede apoyarse en deliberaciones, consultas, priorización o votaciones. En la gobernanza colaborativa, ese proceso es formal, busca el consenso y es deliberativo (Ansell & Gash, 2008).

En el contexto de los proyectos comunitarios, este proceso relaciona la participación de los integrantes de la comunidad con las decisiones asociadas a las iniciativas propuestas. La plataforma proporciona mecanismos digitales para registrar determinadas intervenciones y consultar información relacionada con los resultados.

Gestión colaborativa de proyectos comunitarios

La gestión colaborativa de proyectos comunitarios comprende el conjunto de actividades mediante las cuales diferentes actores intervienen en la formulación, priorización, desarrollo y seguimiento de iniciativas orientadas a atender necesidades de una comunidad (Ansell & Gash, 2008; Osorio-Pérez, 2023). Integra los elementos que estructuran el proceso abordado en el trabajo: participación de los actores, formulación de propuestas, toma de decisiones, gestión de proyectos, seguimiento y consulta de información.

La plataforma se orienta a proporcionar un entorno digital en el que estas actividades puedan organizarse y relacionarse dentro de un mismo sistema y en el que se mantienen diferenciadas las funciones de los participantes y los procesos asociados a cada proyecto.

2.3. Marco Legal

Constitución Política de Colombia de 1991

La Constitución Política de Colombia constituye el fundamento jurídico superior del ordenamiento nacional y establece derechos relacionados directamente con el propósito de la solución propuesta. El artículo 15 reconoce el derecho a la intimidad, al buen nombre y a conocer, actualizar y rectificar la información que haya sido recopilada sobre las personas en bancos de datos y archivos (Constitución Política de Colombia, 1991, art. 15). Por otra parte, el artículo 40 reconoce el derecho de los ciudadanos a participar en la conformación, ejercicio y control del poder político (Constitución Política de Colombia, 1991, art. 40).

Estas disposiciones tienen relación con el proyecto debido a que la plataforma requiere gestionar información asociada a los usuarios y, al mismo tiempo, pretende facilitar mecanismos digitales de participación y toma de

decisiones colectivas. En consecuencia, el diseño de la solución debe contemplar medidas que permitan proteger la información personal de los participantes y establecer mecanismos de participación acordes con los derechos reconocidos constitucionalmente.

Ley 1581 de 2012 – Protección de datos personales

La Ley 1581 de 2012 establece disposiciones generales para la protección de datos personales y desarrolla el derecho de las personas a conocer, actualizar y rectificar la información que haya sido recopilada sobre ellas. La norma establece principios y condiciones aplicables al tratamiento de datos personales por parte de entidades públicas y privadas (Ley 1581 de 2012, 2012).

Su relación con el proyecto es directa debido a que la plataforma requiere registrar y administrar información de los usuarios para permitir su autenticación, participación en propuestas, votaciones y seguimiento de actividades. Por esta razón, la recolección, almacenamiento, consulta y utilización de los datos deberán responder a finalidades determinadas y legítimas, y deberán garantizar los derechos de los titulares y aplicar las medidas necesarias para proteger la información.

Decreto 1377 de 2013 – Reglamentación de la protección de datos personales

El Decreto 1377 de 2013 reglamenta parcialmente la Ley 1581 de 2012 y desarrolla aspectos relacionados con la autorización para el tratamiento de datos personales, las políticas de tratamiento, los derechos de los titulares, las transferencias y transmisiones de información y el principio de responsabilidad demostrada (Decreto 1377 de 2013, 2013).

En el proyecto, esta reglamentación debe considerarse al definir los mecanismos mediante los cuales se informa a los usuarios sobre el tratamiento de sus datos y, cuando corresponda, se obtiene su autorización. También resulta relevante para establecer las políticas de tratamiento de información y los procedimientos mediante los cuales los usuarios puedan ejercer los derechos reconocidos por la legislación colombiana.

Decreto 1074 de 2015 – Reglamentación del sector Comercio, Industria y Turismo en materia de protección de datos

El Decreto 1074 de 2015 compila disposiciones reglamentarias del sector Comercio, Industria y Turismo e incorpora disposiciones relacionadas con la protección de datos personales y la reglamentación de la Ley 1581 de 2012 (Decreto 1074 de 2015, 2015).

Su consideración dentro del proyecto permite complementar las obligaciones relacionadas con el tratamiento de información personal. En particular, debe ser tenido en cuenta al establecer las políticas y procedimientos de tratamiento de datos, así como las responsabilidades asociadas a quienes intervengan en la recolección, almacenamiento o utilización de la información administrada por la plataforma.

Ley 1757 de 2015 – Participación ciudadana

La Ley 1757 de 2015 establece disposiciones relacionadas con la promoción y protección del derecho a la participación democrática y regula diferentes mecanismos e instancias de participación ciudadana. Entre otros aspectos,

desarrolla disposiciones relacionadas con mecanismos como la consulta popular, el cabildo abierto y otros instrumentos de participación (Ley 1757 de 2015, 2015).

Esta norma presenta una relación directa con el propósito funcional del proyecto, debido a que la plataforma busca proporcionar medios digitales para facilitar la participación de los integrantes de una comunidad en la presentación de propuestas, priorización de iniciativas y toma de decisiones colectivas. Sin embargo, los mecanismos de votación implementados en la plataforma deben entenderse como herramientas tecnológicas de participación dentro del alcance del proyecto y no necesariamente como equivalentes jurídicos a los mecanismos de participación ciudadana regulados formalmente por la ley.

Ley 1712 de 2014 – Transparencia y acceso a la información pública

La Ley 1712 de 2014 regula el derecho de acceso a la información pública y establece disposiciones orientadas a garantizar la transparencia, la publicidad y el acceso a la información bajo las condiciones previstas por el ordenamiento jurídico (Ley 1712 de 2014, 2014).

Su relación con el proyecto se encuentra en la necesidad de facilitar la consulta de información relacionada con las propuestas, proyectos comunitarios, estados de avance, resultados de participación e indicadores generados por la plataforma. No obstante, la publicación de información deberá realizarse con respeto a las excepciones y restricciones establecidas por la legislación, especialmente aquellas relacionadas con la protección de datos personales. Por tanto, transparencia no implica la exposición pública de toda la información almacenada en el sistema.

Ley 1341 de 2009 y Ley 1978 de 2019 – Tecnologías de la Información y las Comunicaciones

La Ley 1341 de 2009 establece principios y disposiciones generales relacionados con las Tecnologías de la Información y las Comunicaciones en Colombia. Entre sus objetivos se encuentra promover el acceso y uso de las TIC, así como su aprovechamiento para el desarrollo social, económico y político (Ley 1341 de 2009, 2009). La norma ha sido modificada, entre otras disposiciones, por la Ley 1978 de 2019, mediante la cual se modernizó el sector TIC y se actualizaron diferentes competencias y disposiciones relacionadas con este ámbito (Ley 1978 de 2019, 2019).

Estas normas se relacionan con el proyecto debido a que la solución propuesta corresponde a una aplicación tecnológica multiplataforma destinada a facilitar procesos comunitarios mediante herramientas digitales. Su consideración permite contextualizar jurídicamente el uso de tecnologías, aplicaciones y servicios digitales como medios para promover el acceso y aprovechamiento de las TIC.

Decreto 1078 de 2015 – Política de Gobierno Digital

El Decreto 1078 de 2015 contiene disposiciones del sector de Tecnologías de la Información y las Comunicaciones y establece, entre otros aspectos, lineamientos relacionados con la Política de Gobierno Digital. Esta política contempla el aprovechamiento de las TIC para generar valor público, fortalecer la relación entre ciudadanía y Estado y promover la transformación digital. También incorpora elementos relacionados con seguridad de la información, arquitectura y servicios digitales (Ministerio de Tecnologías de la Información y las Comunicaciones, s. f.).

Esta disposición constituye un referente para el proyecto especialmente cuando la plataforma pueda utilizarse como herramienta de apoyo a procesos de interacción entre ciudadanía y entidades públicas. Su relación es particularmente pertinente por los componentes de participación, colaboración, acceso a información y uso de tecnologías digitales contemplados en la solución.

Decreto 620 de 2020 – Servicios ciudadanos digitales

El Decreto 620 de 2020 establece lineamientos generales para el uso y operación de los servicios ciudadanos digitales y desarrolla disposiciones relacionadas con aspectos como autenticación digital, interoperabilidad, seguridad y privacidad de la información. Entre sus principios se encuentra la privacidad y seguridad desde el diseño y por defecto, así como la protección de la información durante su tratamiento (Decreto 620 del 2 de mayo de 2020, 2020).

Para el proyecto, esta norma constituye un referente especialmente útil en materia de diseño seguro y protección de la información, debido a que la plataforma contempla mecanismos de registro, autenticación, gestión de usuarios y tratamiento de datos. No obstante, su aplicación directa dependerá de que la solución sea implementada dentro del ámbito de los servicios ciudadanos digitales o por sujetos comprendidos en su ámbito de aplicación.

Ley 1273 de 2009 – Protección de la información y de los datos

La Ley 1273 de 2009 modificó el Código Penal colombiano para incorporar disposiciones relacionadas con la protección de la información y de los datos. La norma contempla conductas que afectan sistemas informáticos y datos, entre ellas

el acceso abusivo a sistemas informáticos, la interceptación de datos, el daño informático y otras conductas relacionadas con la seguridad de la información (Ley 1273 de 2009, 2009).

Su relación con el proyecto se encuentra en la necesidad de establecer mecanismos técnicos que reduzcan el riesgo de accesos no autorizados, alteraciones indebidas, pérdida de información o afectaciones a los sistemas que soportan la plataforma. Por esta razón, aspectos como la autenticación, el control de acceso, la gestión de permisos y las políticas de seguridad deben considerarse durante el desarrollo de la solución.

Ley 527 de 1999 – Mensajes de datos y firmas digitales

La Ley 527 de 1999 define y reglamenta el acceso y uso de los mensajes de datos, el comercio electrónico y las firmas digitales, además de establecer disposiciones relacionadas con las entidades de certificación. La norma reconoce efectos jurídicos a determinadas actuaciones realizadas mediante medios electrónicos y establece condiciones relacionadas con la generación, almacenamiento, conservación y comunicación de mensajes de datos (Ley 527 de 1999, 1999).

Esta disposición resulta pertinente para el proyecto debido a que las acciones realizadas dentro de la plataforma, tales como el registro de propuestas, participación en procesos digitales y almacenamiento de información, pueden generar registros electrónicos. Su consideración permite establecer criterios jurídicos para el tratamiento y conservación de información generada mediante medios digitales. Las disposiciones específicas sobre firma digital solamente serán aplicables en caso de que la plataforma incorpore mecanismos de este tipo.

Ley 23 de 1982 y Ley 1915 de 2018 – Derechos de autor y protección del software

La Ley 23 de 1982 establece el régimen general de derechos de autor en Colombia y protege las obras comprendidas dentro de su ámbito. La legislación contempla, entre otras materias, los derechos de los autores y las facultades relacionadas con la reproducción, transformación y comunicación de las obras (Ley 23 de 1982, 1982). La Ley 1915 de 2018 modificó la Ley 23 de 1982 y estableció disposiciones adicionales en materia de derecho de autor y derechos conexos (Ley 1915 de 2018, 2018).

Estas normas son relevantes para el proyecto porque el desarrollo de la plataforma genera elementos susceptibles de protección mediante derechos de autor, particularmente el código fuente, determinados componentes gráficos, documentación y otros recursos originales. Asimismo, durante el desarrollo deberán respetarse las condiciones de uso y las licencias correspondientes a bibliotecas, paquetes, recursos gráficos y demás componentes de terceros utilizados en la aplicación.

Decreto 1360 de 1989 – Protección del software

El Decreto 1360 de 1989 reglamenta la inscripción del soporte lógico o software en el Registro Nacional del Derecho de Autor y reconoce el soporte lógico como una creación propia del dominio literario, de conformidad con la Ley 23 de 1982. También establece elementos que pueden comprender el soporte lógico, entre ellos el programa de computador, la descripción del programa y el material auxiliar (Decreto 1360 de 1989, 1989).

Su relación con el proyecto se encuentra en la protección jurídica del software desarrollado. En consecuencia, el código fuente y los demás elementos que constituyan creaciones originales deberán gestionarse con respeto a los derechos de sus autores y las condiciones aplicables a su utilización, modificación, distribución o registro.

Consideraciones sobre privacidad, alias y protección de la identidad

Debido a que la plataforma contempla mecanismos de participación y votación, el diseño debe diferenciar entre la información utilizada internamente para identificar y autenticar a un usuario y la información que puede ser visible para los demás participantes. La protección de los datos personales exige limitar el tratamiento y divulgación de la información a las finalidades establecidas y aplicar medidas adecuadas de seguridad.

En este contexto, el uso de alias para identificar públicamente a los participantes constituye una medida de diseño orientada a reducir la exposición de su identidad frente a otros usuarios. El alias no elimina por sí mismo la condición de dato personal cuando pueda asociarse con una persona identificable, por lo que la información que permita relacionar el alias con la identidad real deberá mantenerse bajo controles de acceso adecuados. De esta manera, el sistema puede permitir la participación bajo una identidad pública limitada, mientras conserva internamente la información necesaria para la administración, seguridad y control del sistema.

En consecuencia, la arquitectura de la plataforma deberá contemplar mecanismos de autenticación, autorización y control de acceso que permitan diferenciar la información administrativa de aquella que puede ser consultada por

los participantes. Esta separación resulta especialmente relevante para los procesos de votación y participación, en los cuales debe protegerse la identidad del usuario sin perder las capacidades necesarias para administrar la plataforma y garantizar la integridad de los registros.

2.4. Marco Histórico

Antes de la incorporación de tecnologías digitales, la gestión de proyectos comunitarios y la participación ciudadana se apoyaban en reuniones presenciales, documentos físicos y mecanismos tradicionales de consulta. Esas dinámicas permitían la interacción directa, pero dificultaban conservar registros, consultar información histórica y hacer seguimiento continuo a las decisiones (Montez & Rodríguez, 2025). Con la expansión de las TIC, estos procesos incorporaron herramientas digitales para comunicar, almacenar información y recibir la participación de los usuarios (Boté-Vericad, 2025). Arnstein (1969) propuso una escala de ocho peldaños para clasificar la participación, que va de la manipulación y la información al control ciudadano.

En Colombia, la Constitución Política de 1991 definió al Estado como democrático y participativo (art. 1), reconoció el derecho de los ciudadanos a participar en la conformación, el ejercicio y el control del poder político (art. 40) y enumeró como mecanismos de participación el voto, el plebiscito, el referendo, la consulta popular, el cabildo abierto, la iniciativa legislativa y la revocatoria del mandato (art. 103) (Constitución Política de Colombia, 1991). La Ley 134 de 1994 reguló esos mecanismos (Ley 134 de 1994, 1994). En 2012, la Ley 1581 estableció el régimen general de protección de datos personales (Ley 1581 de 2012, 2012). La Ley 1712 de 2014 reguló el derecho de acceso a la información pública (Ley 1712 de 2014, 2014) y la Ley 1757 de 2015 dictó disposiciones sobre la promoción y protección del derecho a la participación democrática (Ley 1757 de 2015, 2015).

Las plataformas digitales de participación de código abierto aparecieron a mediados de la década de 2010. El Ayuntamiento de Madrid desarrolló en 2015 el software CONSUL para su plataforma Decide Madrid (Consul Democracy Foundation, s. f.), y en febrero de 2016 el Ayuntamiento de Barcelona y la comunidad de Decidim crearon la plataforma Decidim (Ajuntament de Barcelona, s. f.). En Colombia, el Decreto 1008 de 2018 estableció la Política de Gobierno Digital, que sustituyó a la estrategia de Gobierno en Línea dentro del Decreto 1078 de 2015 (Decreto 1008 de 2018, 2018), y el Decreto 620 de 2020 fijó los lineamientos de los servicios ciudadanos digitales (Decreto 620 del 2 de mayo de 2020, 2020).

El desarrollo multiplataforma siguió la expansión de los teléfonos inteligentes. La fragmentación de dispositivos y plataformas móviles llevó a desarrollar enfoques web, híbridos, interpretados y compilados para compartir código entre sistemas operativos (Bjørn-Hansen et al., 2018). En 2010, Marcotte propuso el diseño web responsivo para adaptar una misma página a distintos tamaños de pantalla (Marcotte, 2010). Facebook liberó el código de React Native el 26 de marzo de 2015 (Alpert, 2015) y Google publicó el 4 de diciembre de 2018 Flutter 1.0, su primera versión estable, para iOS y Android (Sneath, 2018). Con Flutter 2, publicado el 3 de marzo de 2021, el soporte para la web pasó al canal estable (Announcing Flutter 2, 2021).

En el lado de los datos, PostgreSQL descende del proyecto POSTGRES de la Universidad de California en Berkeley, cuya implementación comenzó en 1986, y adoptó su nombre actual en 1996 (PostgreSQL Global Development Group, s. f.-a). Supabase combina sobre PostgreSQL los servicios de autenticación, API y seguridad por filas que usa este trabajo (Supabase, s. f.-a).

2.5. Marco ambiental

El proyecto es una aplicación de software y no interviene directamente sobre el aire, el agua, el suelo ni la biodiversidad: no implica obras, extracción de recursos naturales ni vertimientos. La Ley 99 de 1993 creó el Ministerio del Medio Ambiente como organismo rector de la gestión ambiental y organizó el Sistema Nacional Ambiental (Ley 99 de 1993, 1993). La misma ley exige licencia ambiental para las obras y actividades que puedan causar deterioro grave a los recursos naturales o modificaciones notorias del paisaje (Ley 99 de 1993, 1993), condición que, por su naturaleza, no corresponde a una plataforma digital como esta.

Los efectos ambientales del proyecto son indirectos y provienen del consumo eléctrico de los equipos con que se desarrolla y se usa la aplicación, y de los servidores donde se aloja. Según la International Energy Agency (IEA, 2025), los centros de datos consumieron cerca de 415 TWh en 2024, alrededor del 1,5 % de la electricidad mundial. El consumo atribuible a esta plataforma no se midió y no se dispone de datos para estimarlo. Los computadores y dispositivos móviles utilizados son aparatos eléctricos y electrónicos cuyos residuos tienen manejo diferenciado y deben gestionarse al final de su vida útil conforme a la Ley 1672 de 2013, que fija los lineamientos de la política pública de gestión integral de estos residuos (Ley 1672 de 2013, 2013).

En sentido favorable, gestionar propuestas, votaciones y seguimiento de proyectos en formato digital puede reducir el uso de papel frente a los procesos manuales (Gutiérrez & Romero, 2024). En este trabajo esa reducción no se cuantificó, por lo que se plantea como una posibilidad y no como un resultado.

3. DISEÑO DE LA INVESTIGACIÓN

La investigación se desarrolló bajo un enfoque cuantitativo, debido a que contempla la recolección y análisis de información susceptible de ser expresada mediante frecuencias, porcentajes e indicadores relacionados con las necesidades de los usuarios, la participación ciudadana, la transparencia de los procesos y la utilización de una plataforma para la gestión de proyectos comunitarios. Este enfoque permitió organizar los resultados obtenidos mediante instrumentos estructurados y utilizarlos como insumo para la definición de requerimientos y para la posterior evaluación de la solución tecnológica desarrollada.

El tipo de investigación es descriptivo, debido a que se buscó identificar y caracterizar las condiciones, necesidades y dificultades relacionadas con la gestión de proyectos comunitarios y los procesos de participación de los usuarios. El carácter descriptivo se evidencia en la recopilación de información sobre las dificultades existentes, las funcionalidades consideradas necesarias, las preferencias respecto a los medios de acceso y la disposición de los posibles usuarios para participar mediante una herramienta digital. De manera complementaria, el proyecto tiene un carácter aplicado, debido a que los resultados obtenidos durante la investigación fueron utilizados como insumo para diseñar y desarrollar una solución tecnológica orientada a la problemática identificada.

3.1. Diagnóstico inicial y origen de la propuesta

Como parte de la etapa inicial del proyecto se realizó una encuesta dirigida a obtener una aproximación a la percepción de posibles usuarios sobre la gestión de proyectos comunitarios y el uso de herramientas digitales para apoyar estos

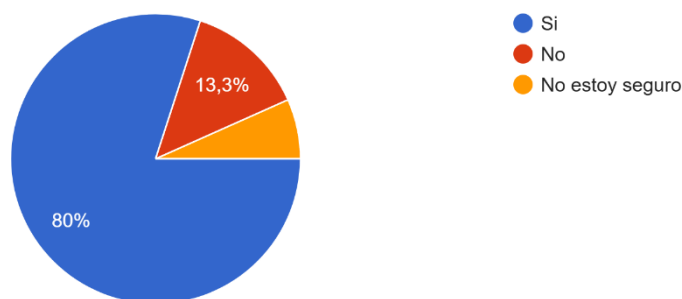
procesos. La información obtenida mediante este instrumento sirvió como uno de los elementos que motivaron y orientaron el inicio de la propuesta de desarrollo de la plataforma.

La encuesta contó con 15 respuestas. De los participantes, 12 personas (80 %) indicaron que sí existen dificultades para organizar, gestionar o realizar seguimiento a proyectos comunitarios, mientras que 2 personas (13,3 %) respondieron que no y 1 persona (6,7 %) manifestó no estar segura.

Figura 1. Primera pregunta de encuesta

¿Considera usted que en su comunidad existen dificultades para organizar, gestionar o dar seguimiento a proyectos comunitarios?

15 respuestas



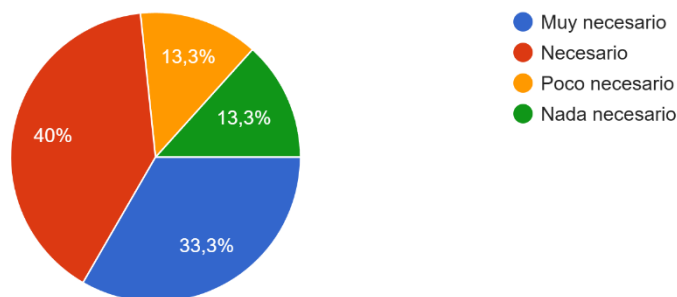
Fuente: Autor

En relación con la necesidad de una herramienta digital, 5 participantes (33,3 %) la consideraron muy necesaria y 6 (40 %) necesaria. Estos resultados permitieron identificar una percepción favorable hacia la utilización de herramientas digitales como apoyo para la gestión de este tipo de proyectos.

Figura 2. Segunda pregunta de encuesta

¿Qué tan necesario considera el uso de una herramienta digital que facilite la gestión de proyectos comunitarios (propuestas, seguimiento, participación ciudadana, etc.)?

15 respuestas



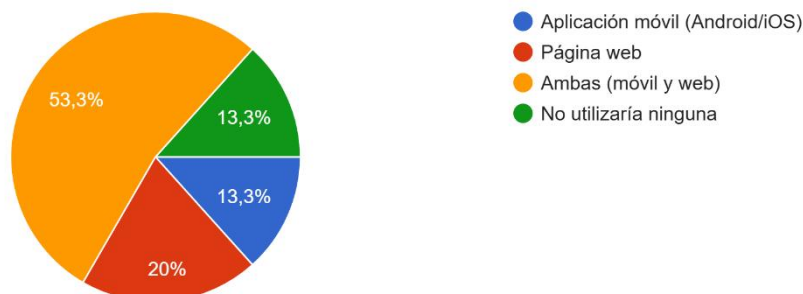
Fuente: Autor

La información recopilada también permitió orientar las características generales de la solución. Ocho participantes (53,3 %) manifestaron preferir una herramienta disponible tanto en dispositivos móviles como en la web, mientras que 3 (20 %) indicaron preferir únicamente una página web y 2 (13,3 %) una aplicación móvil.

Figura 3. Tercera pregunta de encuesta

¿En qué tipo de plataforma preferiría utilizar esta herramienta?

15 respuestas



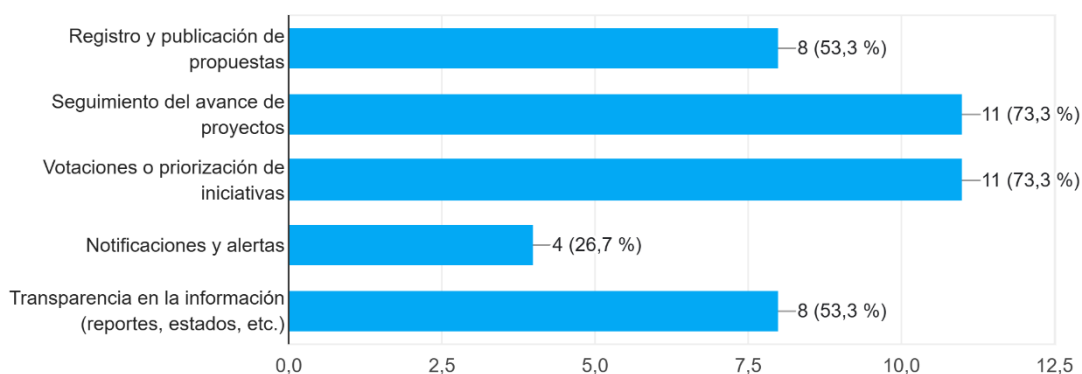
Fuente: Autor

Respecto a las funcionalidades que los participantes consideran prioritarias en una aplicación de este tipo, 11 personas (73,3 %) seleccionaron el seguimiento del avance de los proyectos y las votaciones o priorización de iniciativas. Asimismo, 8 participantes (53,3 %) indicaron el registro y publicación de propuestas y la transparencia de la información. Los resultados muestran que las funciones relacionadas con el seguimiento y la participación en la toma de decisiones son las de mayor interés entre los encuestados.

Figura 4. Cuarta pregunta de encuesta

¿Cuáles de las siguientes funcionalidades considera más importantes en una aplicación de este tipo?

15 respuestas



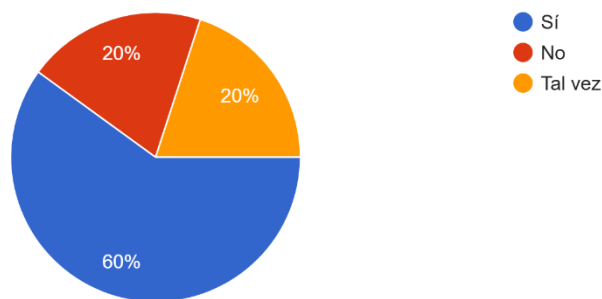
Fuente: Autor

En cuanto a la disposición para participar mediante una plataforma digital, 9 participantes (60 %) respondieron afirmativamente, 3 (20 %) indicaron que tal vez participarían y 3 (20 %) respondieron negativamente.

Figura 5. Quinta pregunta de encuesta

¿Estaría dispuesto(a) a utilizar una plataforma digital para participar activamente en proyectos comunitarios (opinar, votar, proponer ideas)?

15 respuestas



Fuente: Autor

Los resultados de esta encuesta no se consideran representativos de toda la población urbana de Bucaramanga, debido al número de participantes y a que no se estableció un procedimiento de muestreo probabilístico. Su función dentro del proyecto corresponde al diagnóstico inicial y a la identificación de necesidades y percepciones que sirvieron como referencia para orientar el desarrollo de la propuesta tecnológica.

3.2. Método de investigación

El método empleado fue deductivo y de análisis. El método deductivo permitió partir de conceptos, fundamentos y criterios relacionados con la participación ciudadana, la gestión colaborativa, la transformación digital, el desarrollo de software multiplataforma y la gestión de proyectos para establecer los elementos que fueron considerados en la solución propuesta.

El método de análisis permitió descomponer la información obtenida durante el diagnóstico y la revisión documental, se identificaron las características de los

procesos relacionados con la gestión de proyectos comunitarios y se estableció los requerimientos que deberán ser atendidos por el sistema. Este método también fue utilizado durante la revisión de los resultados de las pruebas y de la validación de la plataforma.

3.3. Técnicas e instrumentos de recolección de información

Para la recopilación de información se empleó las técnicas de revisión documental y encuesta.

La revisión documental permitió consultar fuentes académicas, normativas y técnicas relacionadas con participación ciudadana, gestión de proyectos comunitarios, desarrollo de aplicaciones multiplataforma, protección de datos, seguridad de la información y tecnologías utilizadas para el desarrollo de la solución. Esta técnica estableció los fundamentos que orientarán las diferentes etapas del proyecto.

La encuesta fue utilizada para recopilar información de posibles usuarios respecto a las dificultades percibidas en la gestión de proyectos comunitarios, la necesidad de una herramienta digital, las funcionalidades consideradas relevantes, la preferencia de acceso mediante web o dispositivos móviles y la disposición para participar mediante una plataforma digital. El cuestionario utilizado en el diagnóstico inicial constituye uno de los instrumentos que permitió orientar la propuesta y definir características generales de la solución.

3.4. Población y muestra

El trabajo se dirige a habitantes de contextos urbanos de Bucaramanga que participan o podrían participar en proyectos comunitarios. No se contó con un listado de esa población, por lo que se trabajó con dos grupos seleccionados sin procedimiento probabilístico. El primero está formado por las 15 personas que

respondieron la encuesta de diagnóstico, aplicada mediante un formulario en línea. El segundo, por los 10 miembros de la comunidad del Mirador de Arenales que usaron la aplicación durante la validación. La participación fue voluntaria y por conveniencia. Los resultados describen a estos dos grupos y no se generalizan a la población de Bucaramanga.

3.5. Operacionalización de las variables

El enfoque cuantitativo requiere que cada variable se exprese como un indicador medible. La Tabla 1 define las cuatro variables del estudio, el indicador con el que se mide cada una, el instrumento que produce el dato, la forma de cálculo y el lugar de los resultados donde se reporta. La participación, la transparencia y la eficiencia son las dimensiones que plantea el problema de investigación; la usabilidad corresponde a la evaluación con usuarios del objetivo específico 5.

Tabla 1.
Operacionalización de las variables del estudio

Variable	Definición operacional	Indicador	Instrumento y fuente	Cálculo	Objetivo 5 y resultados
Participación	Intervención de los usuarios en las propuestas y las votaciones de la plataforma	Nivel de participación en votaciones	Panel de métricas (función obtener_mtricas) y pruebas PF-77 y PF-78	(Usuarios distintos con voto registrado ÷ usuarios activos) × 100, con un decimal	Se verificó que el panel lo calcula (Tabla 25); no se interpreta como efecto
		Disposición declarada a participar	Encuesta diagnóstica, pregunta 5 (n = 15)	(Respuestas afirmativas ÷ total de respuestas) × 100: $9 \div 15 = 60 \%$	Diagnóstico inicial (Figura 5): intención declarada, no conducta observada
Transparencia	Disponibilidad de la información de propuestas, votaciones y proyectos, y registro de las acciones administrativas	Proporción de pruebas de consulta y de registro superadas	Matriz de pruebas funcionales (Tabla 24): PF-27, PF-35, PF-52, PF-53, PF-64, PF-65 y PF-80	(Pruebas con resultado esperado ÷ pruebas ejecutadas) × 100: $7 \div 7 = 100 \%$	Tabla 25: verifica que la información está disponible, no cómo la percibe el usuario

Variable	Definición operacional	Indicador	Instrumento y fuente	Cálculo	Objetivo 5 y resultados
Eficiencia	Ejecución, sin intervención manual, de la activación y el cierre de votaciones y del cálculo del avance de los proyectos	Proporción de procesos automáticos verificados	Matriz de pruebas funcionales: PF-46, PF-47, PF-48, PF-58 y PF-60	(Pruebas con resultado esperado ÷ pruebas ejecutadas) × 100: $5 \div 5 = 100 \%$	Tabla 25 : verifica la automatización; no se midieron tiempos
Usabilidad	Grado en que usuarios específicos pueden operar la aplicación para lograr sus objetivos con eficacia, eficiencia y satisfacción (ISO/IEC, 2023)	Satisfacción general (P1), recomendación (P2), facilidad de uso (P3) y funcionamiento (P4)	Cuestionario de cuatro preguntas cerradas con escala tipo Likert; 10 participantes	(Respuestas «Positivo» y «Muy positivo» ÷ respuestas válidas) × 100; media en escala de 1 a 5. Umbral del proyecto: $P1 \geq 80 \%$	Tabla 39 : P1 = 80 %; P3 = 70 %

Nota: Tabla elaborada por el autor. PF = prueba funcional; P1 a P4 = preguntas del cuestionario de evaluación.

Los indicadores se calculan sobre los datos de esta prueba y describen al grupo evaluado. No miden un cambio, porque no existe una medición anterior a la puesta en marcha del sistema ni un periodo de uso prolongado. Por eso el objetivo específico 5 se evalúa en funcionamiento y usabilidad, y la Tabla **25** precisa qué se midió en cada variable. En participación, el estudio verifica que el panel de métricas calcule el indicador (PF-77 y PF-78); los valores que se registren durante la prueba no se interpretan como un efecto del sistema.

3.6. Consideraciones Éticas

La participación fue voluntaria en los dos grupos: las 15 personas que respondieron la encuesta de diagnóstico mediante un formulario en línea y los 10 miembros de la comunidad del Mirador de Arenales que usaron la aplicación durante la validación. El tratamiento de los datos personales se rigió por la Ley 1581 de 2012 (2012) y el Decreto 1377 de 2013 (Decreto 1377 de 2013, 2013), que requieren la autorización previa e informada del titular, y por el artículo 15 de la Constitución,

que protege la intimidad y el derecho a conocer, actualizar y rectificar la información personal (Constitución Política de Colombia, 1991, art. 15).

El consentimiento de los usuarios de la aplicación se obtuvo en el registro. Al crear la cuenta, el sistema muestra una ventana emergente con los términos y la política de privacidad, y el registro solo se completa si la persona marca la casilla de aceptación. Esa aceptación se tomó como la autorización para tratar los datos que la aplicación recoge: nombre, correo electrónico, teléfono, alias, avatar e imágenes.

Para proteger esos datos, la base de datos aplica políticas de seguridad por filas, de modo que cada rol solo accede a la información que le corresponde. El correo y el teléfono de una persona no son visibles para otros usuarios, y las propuestas muestran el alias en lugar del nombre completo cuando el usuario lo ha definido. El voto se guarda sin el identificador del usuario, en una tabla separada del registro de quién votó, y la verificación del correo electrónico está activa. La posibilidad de vincular identidad y voto mediante las marcas de tiempo se describe como limitación en las recomendaciones. Los resultados de la encuesta de diagnóstico y del cuestionario de validación se presentan de forma agregada.

Los participantes de las dos encuestas fueron informados de manera oral que sus respuestas son anónimas y se usaran como datos de información para análisis estadístico.

Las Figuras que muestran a usuarios durante las pruebas, cada persona que aparece firmó un formato de autorización de uso de imagen, que limita el uso a fines académicos (informe, revisión, evaluación y sustentación), excluye el uso comercial. Los formatos firmados se incluyen en los anexos.

3.7. Procedimiento y fases del proyecto

El desarrollo del proyecto se organizó en fases relacionadas con los objetivos planteados, de manera que cada etapa generó resultados que sirvan como insumo para la siguiente.

Fase 1. Levantamiento y análisis de los requisitos.

A partir de la información obtenida en la fase de diagnóstico se estableció los requerimientos funcionales y no funcionales del sistema. En esta etapa se definieron los usuarios, roles, procesos y funcionalidades principales, entre ellas la gestión de propuestas, votaciones o priorización de iniciativas, seguimiento de proyectos y consulta de información relacionada con la participación.

Fase 2. Modelamiento y diseño de la solución.

En esta fase se realizó el modelamiento del sistema y el diseño de la solución tecnológica. Se estableció la arquitectura del sistema, la estructura de la base de datos, los modelos y diagramas correspondientes y el diseño de las interfaces de usuario. También se definieron los mecanismos de autenticación, autorización y control de acceso necesarios para proteger la información administrada por la plataforma.

Fase 3. Desarrollo e implementación.

En esta etapa se desarrolló la plataforma con Flutter para la construcción de la aplicación destinada a los entornos web y móvil. Supabase fue utilizado como servicio de backend para la gestión de la base de datos, autenticación, control de acceso y demás servicios requeridos por la aplicación. Git y Github para el control de versiones y repositorio del código. La implementación comprendió los módulos

definidos durante la etapa de requisitos, principalmente la gestión y publicación de propuestas, votación o priorización de iniciativas, seguimiento de proyectos y consulta de métricas de participación.

Fase 4. Pruebas funcionales.

Una vez implementados los componentes principales se realizaron pruebas funcionales. Las pruebas funcionales permitieron verificar que las funcionalidades desarrolladas respondan a los requerimientos establecidos.

Fase 5. Validación de la solución.

En esta fase se recopiló información de usuarios potenciales mediante instrumentos estructurados para valorar aspectos relacionados con el funcionamiento, facilidad de uso y pertinencia de las funcionalidades implementadas. Los resultados obtenidos permitieron identificar aspectos que requieran ajustes y estableció el grado de cumplimiento de los requerimientos definidos durante las primeras fases.

4. DESARROLLO DEL TRABAJO DE GRADO

4.1. Levantamiento y Análisis de los Requisitos

4.1.1. Requisitos Funcionales

Gestión de usuarios y autenticación

- El sistema deberá permitir a los ciudadanos registrarse mediante un formulario con información básica de identificación y credenciales de acceso.
- El sistema deberá permitir a los usuarios iniciar sesión mediante correo electrónico y contraseña.
- El sistema deberá validar las credenciales proporcionadas antes de conceder acceso a las funcionalidades privadas.
- El sistema deberá permitir a los usuarios recuperar su contraseña mediante un mecanismo de recuperación asociado a su correo electrónico.
- El sistema deberá permitir al ciudadano consultar y actualizar la información de su perfil.
- El sistema deberá permitir cerrar la sesión de forma segura.
- El sistema deberá diferenciar los permisos de acceso de acuerdo con el rol del usuario.

Gestión de propuestas comunitarias

- El sistema deberá permitir a los ciudadanos registrar propuestas comunitarias.
- El sistema deberá permitir ingresar título, descripción, categoría y demás información definida para una propuesta.

- El sistema deberá permitir adjuntar evidencias o imágenes relacionadas con una propuesta.
- El sistema deberá asignar automáticamente un estado inicial a cada propuesta registrada.
- El sistema deberá permitir al ciudadano consultar las propuestas que ha registrado.

Gestión de votaciones

- El sistema deberá permitir al administrador crear procesos de votación asociados a propuestas comunitarias.
- El sistema deberá permitir configurar las fechas de inicio y finalización de una votación.
- El sistema deberá permitir a los ciudadanos consultar las votaciones disponibles.
- El sistema deberá permitir a los ciudadanos emitir su voto durante el periodo establecido.
- El sistema deberá impedir que un ciudadano emita más de un voto en una misma votación.
- El sistema deberá registrar la participación del usuario en el proceso de votación y mantener la confidencialidad de su elección, de acuerdo con las reglas definidas para el proyecto.
- El sistema deberá cerrar automáticamente una votación cuando finalice el periodo establecido.
- El sistema deberá calcular los resultados de las votaciones.
- El sistema deberá permitir consultar los resultados de las votaciones finalizadas.
- El administrador deberá poder consultar y gestionar los procesos de votación.

Gestión y seguimiento de proyectos

- El sistema deberá permitir al administrador convertir o registrar propuestas seleccionadas como proyectos comunitarios.
- El sistema deberá permitir registrar información básica de los proyectos comunitarios.
- El sistema deberá permitir asignar responsables a los proyectos.
- El sistema deberá permitir definir y actualizar el estado de un proyecto.
- El sistema deberá permitir actualizar el porcentaje de avance de un proyecto.
- El sistema deberá permitir registrar avances, novedades y observaciones de los proyectos.
- El sistema deberá permitir adjuntar evidencias relacionadas con el avance de un proyecto.
- El sistema deberá permitir a los ciudadanos consultar los proyectos comunitarios publicados.
- El sistema deberá permitir consultar el estado y porcentaje de avance de los proyectos.

Métricas y reportes

- El sistema deberá recopilar información relacionada con la participación de los usuarios.
- El sistema deberá generar métricas relacionadas con usuarios, propuestas, votaciones y proyectos.
- El sistema deberá presentar las métricas mediante indicadores y elementos gráficos.
- El administrador deberá poder consultar reportes de participación ciudadana.

- El sistema deberá permitir consultar estadísticas de propuestas y votaciones.
- El sistema deberá permitir exportar información estadística cuando corresponda.

Notificaciones y comunicación

- El sistema deberá generar notificaciones relacionadas con eventos relevantes para el usuario.
- El sistema deberá notificar al ciudadano sobre cambios en sus propuestas.
- El sistema deberá notificar eventos relacionados con votaciones en las que pueda participar.
- El sistema deberá permitir consultar las notificaciones recibidas.

Administración del sistema

- El sistema deberá proporcionar al administrador un panel de gestión.
- El administrador deberá poder consultar usuarios registrados.
- El administrador deberá poder gestionar categorías de propuestas.
- El administrador deberá poder consultar el historial de acciones relevantes realizadas en el sistema.
- El sistema deberá restringir las funciones administrativas exclusivamente a usuarios con permisos correspondientes.

4.1.2. Requisitos No Funcionales

Usabilidad

- La interfaz deberá proporcionar una navegación clara y consistente entre las diferentes funcionalidades.

- Las funcionalidades principales deberán poder ser utilizadas sin requerir conocimientos técnicos especializados.
- Los formularios deberán proporcionar mensajes claros sobre errores y validaciones.
- La interfaz deberá mantener una estructura visual consistente entre las versiones web y móvil.

Compatibilidad y multiplataforma

- El sistema deberá estar disponible mediante una aplicación web y una aplicación móvil desarrolladas bajo una estrategia multiplataforma.
- La interfaz deberá adaptarse a diferentes tamaños y orientaciones de pantalla.
- La aplicación deberá mantener una experiencia de usuario consistente entre las plataformas soportadas.

Seguridad

- Las credenciales de los usuarios deberán almacenarse mediante mecanismos seguros de autenticación.
- El sistema deberá implementar control de acceso basado en roles.
- La información transmitida entre cliente y servidor deberá utilizar mecanismos de comunicación segura.
- El sistema deberá proteger la información personal de los usuarios frente a accesos no autorizados.
- El sistema deberá garantizar que un usuario no pueda acceder a funcionalidades correspondientes a otro rol.
- La información relacionada con los procesos de votación deberá gestionarse de acuerdo con las reglas de privacidad definidas para el proyecto.

Rendimiento

- Las operaciones habituales de consulta deberán presentar tiempos de respuesta adecuados para una interacción fluida.
- El sistema deberá optimizar la carga de imágenes y archivos para evitar tiempos de espera innecesarios.
- El sistema deberá ser capaz de soportar múltiples usuarios concurrentes dentro de la capacidad definida para el proyecto.

Disponibilidad y confiabilidad

- El sistema deberá mantener la disponibilidad de los servicios durante los periodos establecidos para su utilización.
- El sistema deberá gestionar adecuadamente errores de conexión o fallos de los servicios externos.
- El sistema deberá evitar la pérdida de información durante las operaciones de registro y actualización.

Mantenibilidad

- El software deberá desarrollarse con una arquitectura modular que facilite su mantenimiento y evolución.
- El código fuente deberá seguir convenciones de organización, nomenclatura y documentación.
- Las funcionalidades deberán estar desacopladas para facilitar modificaciones futuras.
- La solución deberá permitir incorporar nuevas funcionalidades sin requerir modificaciones extensivas en los módulos existentes.

Escalabilidad

- La arquitectura deberá permitir incrementar la cantidad de usuarios y datos gestionados sin requerir una modificación completa de la solución.
- La infraestructura deberá permitir ampliar los recursos disponibles de acuerdo con el crecimiento de la plataforma.

Privacidad y tratamiento de datos

- El sistema deberá limitar la visualización de información personal de acuerdo con el rol y los permisos del usuario.
- El sistema deberá aplicar mecanismos de protección para la información personal almacenada.
- El tratamiento de los datos personales deberá realizarse conforme a la normativa colombiana aplicable.

4.1.3. Historias de usuario

EPICA 1. Autenticación y Gestión de Usuarios

HU-1 Registro de usuario

Como ciudadano interesado en participar en proyectos comunitarios, quiero registrarme en la plataforma con mis datos personales, para acceder a las funcionalidades del sistema y participar en las actividades comunitarias.

HU-2 Inicio de sesión

Como ciudadano registrado, quiero iniciar sesión con mi correo y contraseña, para acceder de forma segura a mi cuenta y utilizar la plataforma.

HU-3 Recuperar contraseña

Como usuario registrado, quiero recuperar mi contraseña mediante mi correo electrónico, para volver a acceder a mi cuenta cuando la haya olvidado.

EPICA 2. Gestión de Propuestas

HU-4 Registrar propuesta

Como ciudadano, quiero registrar una propuesta comunitaria con su nombre, descripción, categoría y ubicación, para presentar iniciativas que beneficien a mi comunidad.

HU-5 Editar propuesta

Como ciudadano, quiero modificar una propuesta antes de que sea aprobada, para corregir errores o actualizar la información.

HU-6 Consultar estado de propuesta

Como ciudadano, quiero conocer el estado de mis propuestas, para saber si fueron aprobadas, rechazadas o continúan en revisión.

EPICA 3. Votaciones

HU-7 Consultar votaciones activas

Como ciudadano, quiero visualizar las votaciones disponibles, para participar en la toma de decisiones comunitarias.

HU-8 Emitir voto

Como ciudadano, quiero votar por una propuesta, para expresar mi opinión dentro del proceso democrático de selección

HU-9 Consultar resultados

Como ciudadano, quiero visualizar los resultados de las votaciones finalizadas, para conocer las decisiones tomadas por la comunidad.

EPICA 4. Seguimiento de Proyectos

HU-10 Consultar proyectos

Como ciudadano, quiero visualizar los proyectos comunitarios, para conocer las iniciativas que se encuentran en desarrollo.

HU-11 Consultar avance del proyecto

Como ciudadano, quiero conocer el progreso de un proyecto, para hacer seguimiento a su ejecución.

EPICA 5. Perfil

HU-12 Actualizar perfil

Como ciudadano, quiero modificar mi información personal, para mantener mis datos actualizados.

EPICA 6. Administración

HU-13 Gestionar usuarios

Como administrador, quiero consultar y administrar los usuarios registrados, para garantizar el correcto funcionamiento de la plataforma.

HU-14 Aprobar propuestas

Como administrador, quiero aprobar o rechazar propuestas ciudadanas, para asegurar que únicamente se publiquen iniciativas válidas.

HU-15 Crear votaciones

Como administrador, quiero crear procesos de votación, para permitir la participación ciudadana en la toma de decisiones.

HU-16 Gestionar proyectos

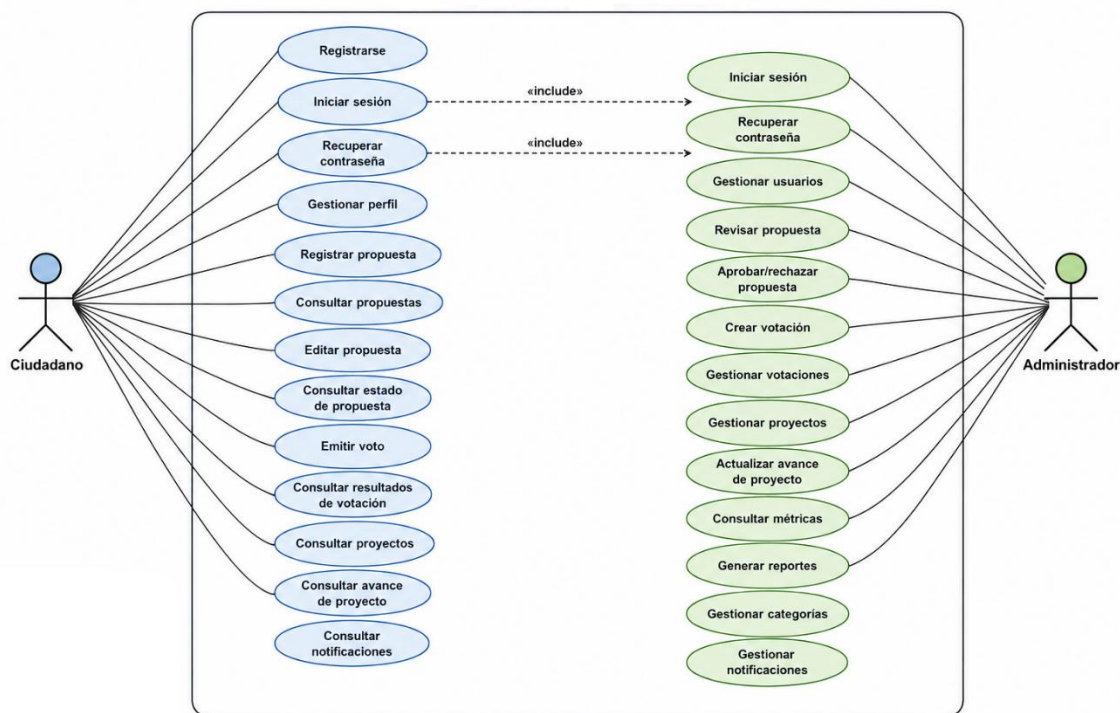
Como administrador, quiero actualizar el estado de los proyectos comunitarios, para mantener informada a la ciudadanía sobre su ejecución.

HU-17 Consultar métricas

Como administrador, quiero visualizar indicadores de participación ciudadana, para evaluar el desempeño de la plataforma y apoyar la toma de decisiones.

4.1.4. Casos de Uso

Figura 6. Diagrama de Casos de Uso



Fuente: Autor

El ciudadano puede registrarse, iniciar sesión, gestionar su perfil, registrar y consultar propuestas, participar en votaciones, consultar proyectos, revisar sus avances y recibir notificaciones.

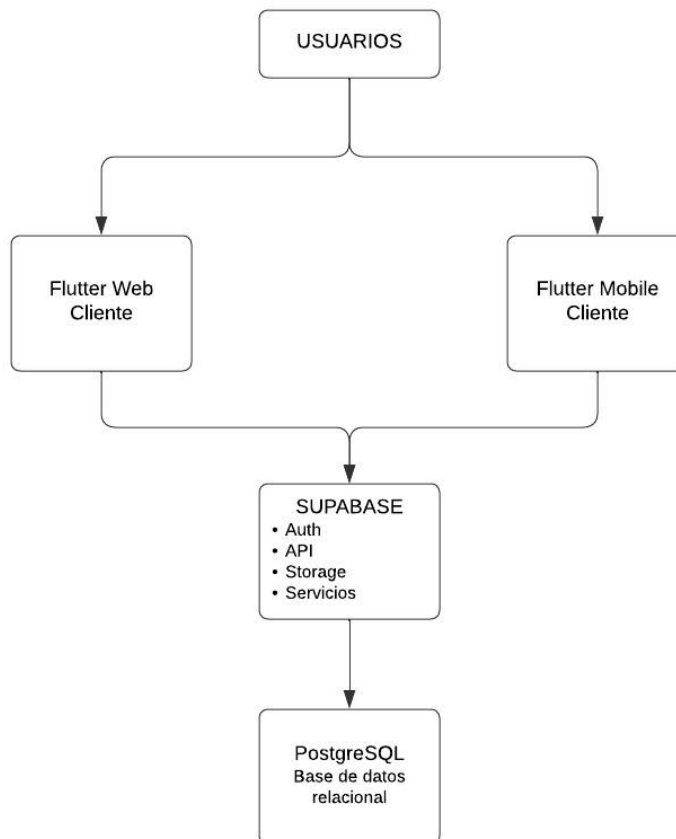
El administrador dispone de funciones relacionadas con la gestión y supervisión de la plataforma, como administrar usuarios, revisar y aprobar o rechazar propuestas, crear y gestionar votaciones, administrar proyectos, actualizar sus avances, consultar métricas y generar reportes. También puede gestionar categorías y notificaciones.

Las relaciones <<include>> representadas en el diagrama indican que determinadas acciones requieren la ejecución de otra funcionalidad. En este caso, el inicio de sesión y la recuperación de contraseña se relacionan con los procesos correspondientes de autenticación.

4.2. Diseño de la arquitectura del sistema

Para el desarrollo de la plataforma de gestión colaborativa de proyectos comunitarios se propone una arquitectura basada en el modelo cliente-servidor, con Flutter como tecnología para el desarrollo de las interfaces multiplataforma y Supabase como plataforma de servicios backend. La arquitectura busca facilitar la comunicación entre los usuarios y los servicios del sistema; con ella se pueden gestionar la autenticación, las propuestas comunitarias, los procesos de votación, el seguimiento de proyectos y las métricas de participación. La solución estará estructurada en diferentes capas y componentes, con el propósito de mantener una separación entre la interfaz de usuario, la lógica de aplicación y la gestión de datos.

Figura 7. Diagrama de la Arquitectura del sistema



Fuente: Autor; elaborada en <https://lucid.co/es/lucidchart>

Arquitectura lógica: Para el proyecto se propone una arquitectura lógica dividida en tres capas principales. Las solicitudes del usuario recorren las capas en ese orden y las respuestas regresan por el mismo camino.

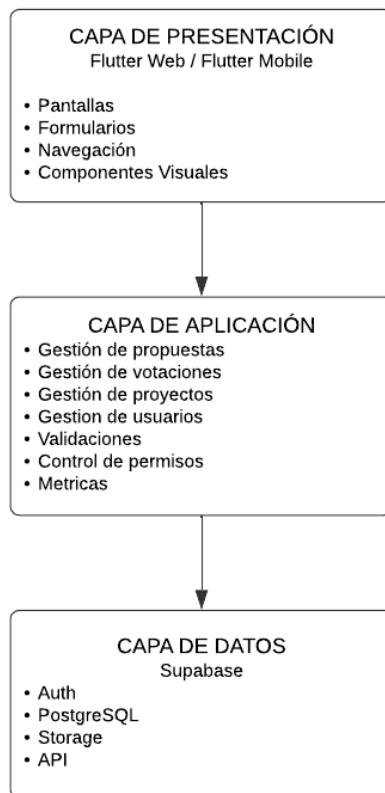
La capa de presentación correspondió a la aplicación desarrollada en Flutter, en sus versiones web y móvil, que comparten la misma base de código. Incluyó las pantallas, los formularios para ingresar información, la navegación entre vistas y los componentes visuales reutilizables. Es la única capa con la que interactúa el

usuario: recibe sus acciones y muestra los resultados que le entrega la capa de aplicación.

La capa de aplicación agrupó la lógica de los módulos del sistema: gestión de propuestas, de votaciones, de proyectos y de usuarios. También incluyó las validaciones, que verifican los datos antes de almacenarlos, y el control de permisos, que define qué acciones puede realizar cada usuario. Las métricas de participación se calcularon en esta capa a partir de la información registrada.

La capa de datos estuvo a cargo de Supabase y se compuso de cuatro servicios. Auth gestionó el registro y la autenticación de los usuarios. PostgreSQL almacenó la información de los módulos en una base de datos relacional. Storage guardó los archivos asociados a esa información. La API expuso estos servicios a las capas superiores, de modo que la aplicación accedió a los datos solo a través de ella.

Figura 8. Diagrama de la arquitectura lógica del sistema



Fuente: Autor; elaborada en <https://lucid.co/es/lucidchart>

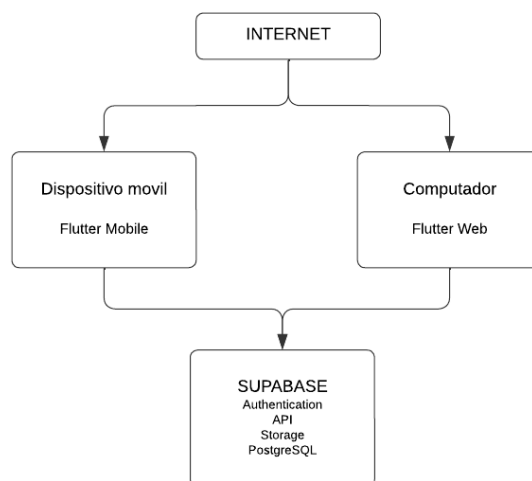
Arquitectura física: La arquitectura física constó de dos tipos de cliente y un servicio en la nube, conectados a través de Internet.

El primer nodo fue el dispositivo móvil Android, donde se ejecutó la aplicación Flutter Mobile. El segundo fue el computador, desde el cual el usuario accedió con un navegador a la aplicación Flutter Web, alojada en Vercel. El navegador descargó la aplicación desde Vercel y, una vez cargada, se comunicó con Supabase de la misma manera que el cliente móvil.

El tercer nodo fue Supabase, que concentró los servicios de autenticación, API, almacenamiento de archivos y la base de datos PostgreSQL. Ambos clientes se conectaron directamente a este nodo, sin un servidor intermedio propio, por lo que compartieron la misma base de datos y los mismos usuarios sin importar la plataforma.

Toda la comunicación entre los clientes y Supabase se realizó mediante HTTPS, con solicitudes a la API de Supabase e intercambio de datos en formato JSON. La autenticación se hizo con Supabase Auth, y las solicitudes posteriores a la API incluyeron el token de sesión del usuario.

Figura 9. Diagrama arquitectura física del sistema



Fuente: Autor; elaborada en <https://lucid.co/es/lucidchart>

Componentes del sistema: Los principales componentes identificados para la solución son los siguientes

Tabla 2.
Componentes del sistema

Componente	Responsabilidad
------------	-----------------

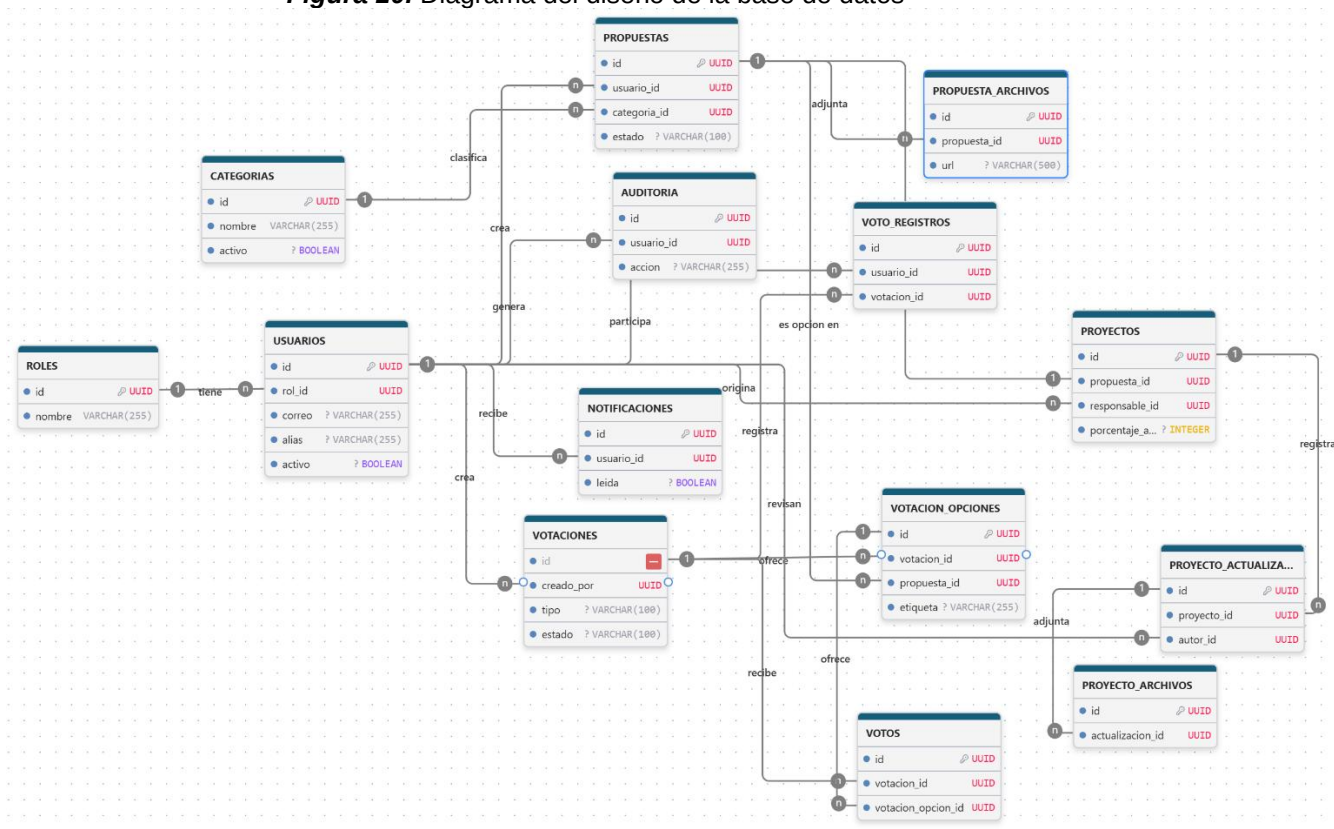
Aplicación Flutter Web	Proporcionar acceso a la plataforma mediante navegadores web.
Aplicación Flutter Mobile	Proporcionar acceso mediante dispositivos móviles.
Módulo de autenticación	Gestionar registro, inicio de sesión, recuperación de contraseña y sesiones.
Módulo de usuarios	Gestionar perfiles y roles.
Módulo de propuestas	Registrar, consultar, modificar y gestionar propuestas comunitarias.
Módulo de votaciones	Crear, administrar y participar en procesos de votación.
Módulo de proyectos	Gestionar proyectos y realizar seguimiento de su ejecución.
Módulo de notificaciones	Informar a los usuarios sobre eventos relevantes.
Módulo de métricas	Procesar y presentar indicadores de participación.
Módulo administrativo	Proporcionar herramientas de administración del sistema.
Supabase Auth	Gestionar la autenticación de usuarios.
Supabase Storage	Almacenar imágenes y archivos asociados a propuestas y proyectos.
PostgreSQL	Almacenar y relacionar la información del sistema.

Nota: Tabla elaborada por el autor a partir del análisis para el diseño de la arquitectura del sistema

4.3. Diseño de la base de datos

Para el desarrollo del sistema se utilizará PostgreSQL, mediante la plataforma Supabase, el diseño se realizará bajo un modelo relacional, con el fin de garantizar la integridad, consistencia y disponibilidad de la información, así como reducir la redundancia de datos mediante la definición de claves primarias, claves foráneas y restricciones.

Figura 10. Diagrama del diseño de la base de datos



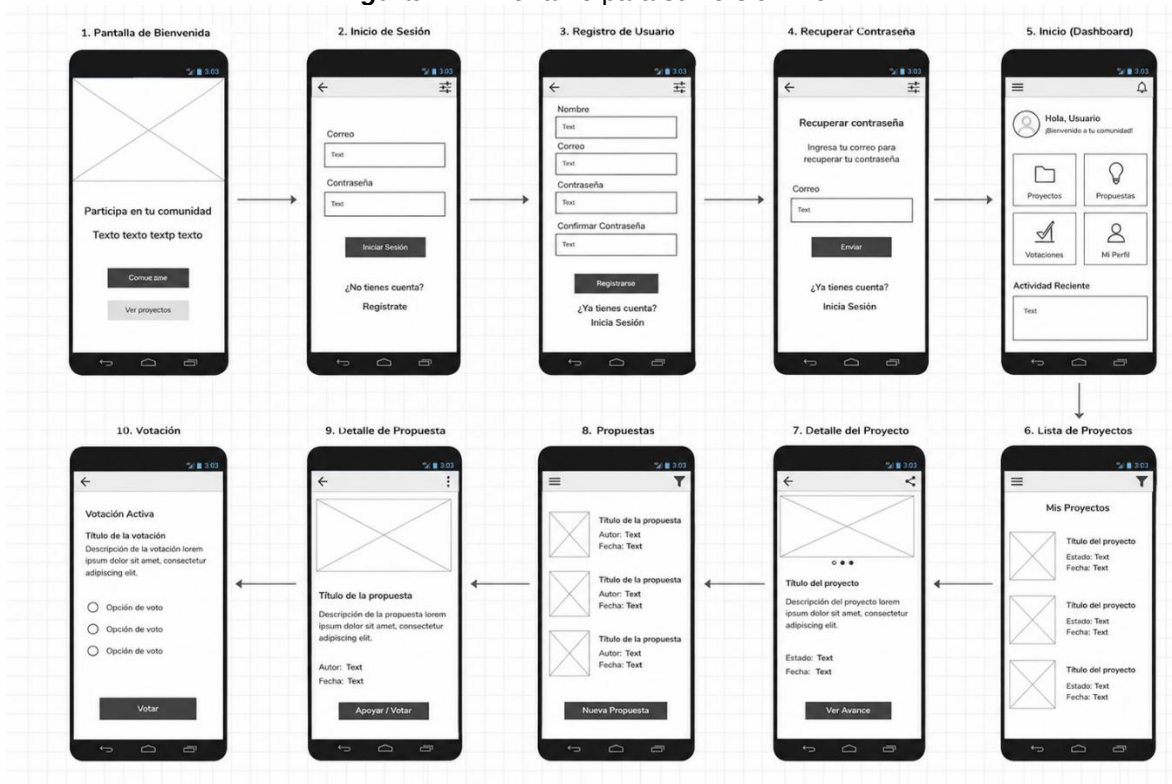
1

Fuente: Autor; Elaborado en <https://www.drawdb.app/>

¹ Nota: En esta elaboración se manejaron campos de tipo VARCHAR, para el entregable final se pasaron los campos a TEXT

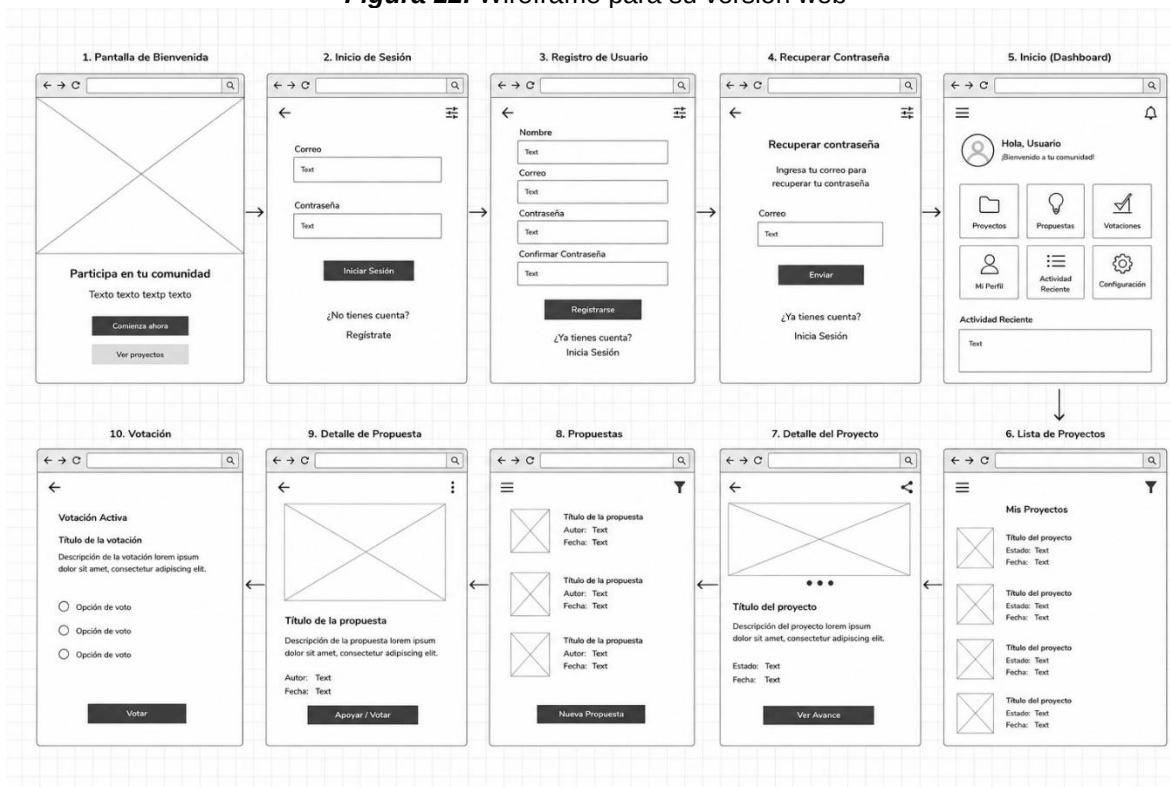
4.4. Wireframe

Figura 11. Wireframe para su versión móvil



Fuente: Autor

Figura 12. Wireframe para su versión web



Fuente: Autor

4.5. Desarrollo de los módulos del sistema

El desarrollo de la plataforma siguió un enfoque incremental e iterativo. El diagnóstico, el levantamiento de requisitos y el diseño de la arquitectura y de la base de datos se describen en la sección 3 y en las secciones 4.1 a 4.4. La implementación se hizo en ciclos sucesivos, cada uno dedicado a un módulo o a un conjunto de ajustes. En cada ciclo se programó la funcionalidad, se ejecutó la aplicación para comprobar su comportamiento, se corrigieron los errores encontrados y el resultado se registró en un commit.

El proyecto fue desarrollado por una sola persona, por lo que no se definieron los roles ni los eventos de Scrum. Del enfoque ágil se tomaron la redacción de los

requisitos como historias de usuario agrupadas en épicas y la entrega por incrementos funcionales.

Tabla 3.

Fases del desarrollo según el historial de commits

Fase	Periodo	Actividad	Commits
1	28 de agosto de 2026	Creación del repositorio, proyecto Flutter base e instalación de dependencias	0a8c76d, 91fff43, 41b6e08
2	29 de agosto de 2026	Estructura de carpetas, conexión con Supabase y diseño inicial	0db4f0f, 07ad894
3	31 de agosto de 2026	Módulo de propuestas	d6c15e1
4	Del 31 de agosto al 1 de septiembre de 2026	Módulo de votaciones y ajuste para votaciones Sí/No	ec6fbdb, 9dfd08c
5	1 de septiembre de 2026	Módulo de proyectos	2b59a64
6	2 de septiembre de 2026	Módulo de notificaciones	31720ea
7	3 de septiembre de 2026	Panel administrativo y corrección del manejo de usuarios inactivos	a317286
8	5 de septiembre de 2026	Módulo de métricas y auditoría de registros	b3879fc, fdc9d06
9	8 de septiembre de 2026	Correcciones en propuestas, perfiles y notificaciones	97a0698, 1c7984c
10	Del 16 al 17 de septiembre de 2026	Métricas por categoría y ciclo de pruebas	1da7f60, 5135034
11	Del 19 al 21 de septiembre de 2026	Rediseño de la interfaz y corrección de errores	c9eba34, 80d49d6
12	Del 22 al 27 de septiembre de 2026	Empaquetado para Android, permiso de internet, actualización de la base de datos y corrección de la recuperación de contraseña	5f5a7d6, 42b6bac, 2235f63, bf5e5c8, c23c0d5

Nota: Tabla elaborada por el autor a partir del historial de commits del repositorio

4.5.1. Preparación del entorno de desarrollo

Para iniciar el desarrollo de la plataforma se realizó la configuración del entorno de trabajo y de las herramientas necesarias para implementar, almacenar, probar y controlar las diferentes partes del sistema. La selección de estas herramientas estuvo relacionada con la arquitectura definida para el proyecto y con el objetivo de disponer de una solución multiplataforma para los entornos web y móvil.

Tabla 4.
Entorno de Desarrollo

Herramienta	Versión	Rol en el proyecto
Flutter SDK	3.44.4 (canal stable)	Framework principal
Dart SDK	3.12.2	Lenguaje del proyecto
Android SDK	36.1.0	Compilación y emulación Android
Android Emulator	36.3.10.0	Pruebas en emulador
JDK	Bundled con Android Studio	Herramientas de compilación de Gradle
Supabase	Plan Free	Backend (Auth, Postgres, Storage, Edge Functions)
Sistema operativo de desarrollo	Windows 11 (25H2)	---

Nota: Tabla elaborada por el autor

Tabla 5.
Paquetes directos (dependencias)

Paquete	Versión	Rol
supabase_flutter	2.17.2	Cliente de Supabase (Auth, Postgrest, Storage, Realtime)
flutter_riverpod	3.4.3	Gestión de estado
go_router	18.0.1	Enrutamiento y protección de rutas
flutter_dotenv	6.0.1	Carga de variables de entorno
image_picker	1.2.3	Selección de imágenes (propuestas, evidencias, avatar)
fl_chart	1.2.0	Gráficas del panel de métricas
google_fonts	8.2.1	Tipografía Poppins
url_launcher	6.3.3	Apertura de enlaces externos (política de datos, recursos)
cupertino_icons	1.0.9	Set de íconos por defecto de Flutter

Nota: Tabla elaborada por el autor

Flutter fue utilizado como framework principal para el desarrollo de la aplicación, lo que permitió construir las interfaces y funcionalidades de la plataforma a partir de una misma base de código para los diferentes entornos. El desarrollo se realizó con Dart como lenguaje de programación, empleado para la construcción de la lógica de la aplicación, componentes de interfaz y comunicación con los servicios del sistema.

Para la gestión de los servicios de backend se configuró un proyecto en Supabase, utilizado como plataforma para administrar los servicios requeridos por la aplicación. Dentro de este entorno se configuró la base de datos PostgreSQL, en la cual se estableció la estructura necesaria para almacenar la información relacionada con usuarios, propuestas, proyectos, votaciones, avances y registros de participación.

Para el control de versiones se utilizó Git, con GitHub como repositorio remoto. El historial de la rama principal contiene 24 commits realizados entre el 28 de agosto y el 27 de septiembre de 2026. La verificación durante esta etapa consistió en ejecutar la aplicación y comprobar el comportamiento de lo implementado. El historial muestra el carácter iterativo del proceso, porque varios módulos se corrigieron después de implementados:

- El módulo de votaciones se ajustó el día siguiente a su desarrollo para soportar votaciones de tipo Sí/No.
- Los módulos de propuestas, perfiles y notificaciones recibieron correcciones los días 6 y 8 de septiembre.
- La interfaz se modificó el 19 y el 21 de septiembre.

- La base de datos se actualizó el 25 de septiembre con las modificaciones surgidas durante el desarrollo.
- El flujo de recuperación de contraseña se corrigió en dos ocasiones, el 22 y el 27 de septiembre.

Finalmente, se utilizaron las herramientas disponibles en el entorno de desarrollo para ejecutar y comprobar la aplicación durante su construcción. Las pruebas se realizaron progresivamente sobre las funcionalidades implementadas, y se verificaron aspectos como la navegación entre interfaces, el funcionamiento de los formularios, la comunicación con Supabase, el almacenamiento y consulta de información y el comportamiento de los módulos desarrollados.

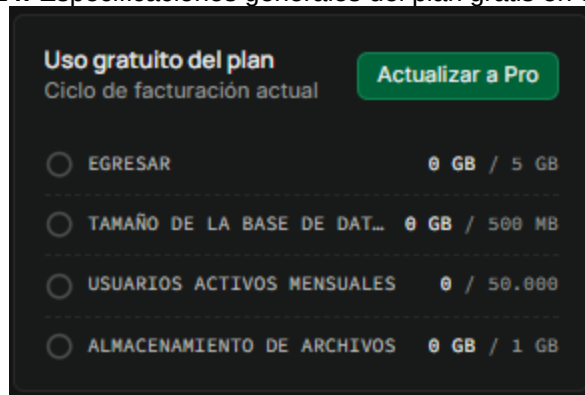
4.5.2. Implementación de la base de datos y Supabase

Al momento de la creación de la cuenta de Supabase se escogió el plan gratuito, el cual es suficiente para el proyecto.

Figura 13. Creación de una nueva Organización en Supabase

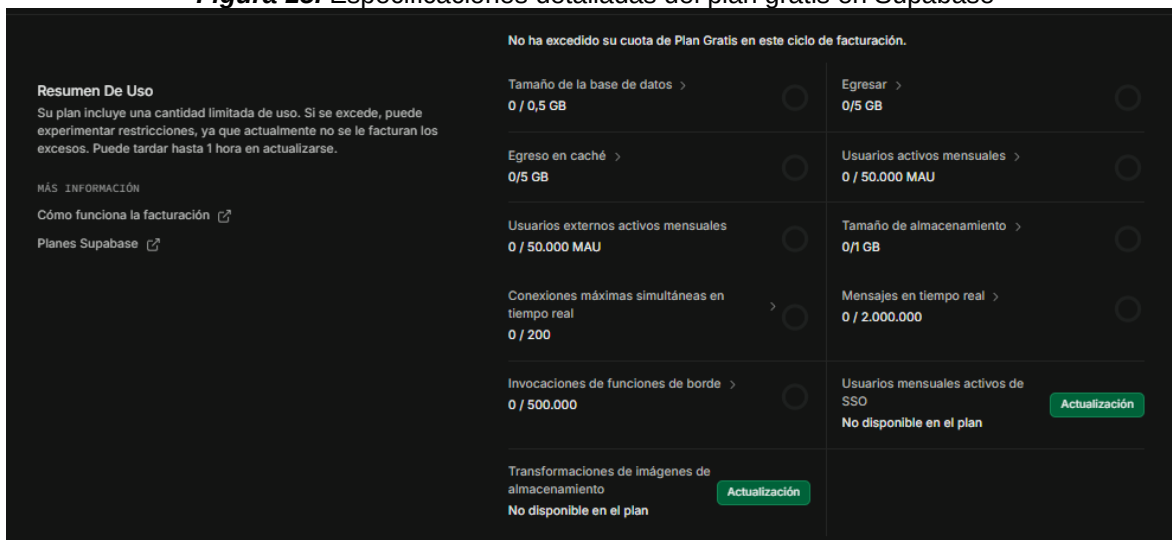
Fuente: Autor

Figura 14. Especificaciones generales del plan gratis en Supabase



Fuente: Autor

Figura 15. Especificaciones detalladas del plan gratis en Supabase



Fuente: Autor

Este modelo establece la estructura de datos sobre la cual se organiza el sistema y permite gestionar de manera relacionada la información de usuarios, propuestas, votaciones y seguimiento de proyectos. Su diseño e implementación responde al entregable establecido para el proyecto, lo que garantiza que la base de datos se encuentre estructurada y documentada de acuerdo con las necesidades funcionales definidas.

El modelo de datos se construyó a partir del análisis de los procesos funcionales definidos para el sistema, entre ellos el registro de ciudadanos, el ciclo de vida de las propuestas, la votación digital, la conversión de propuestas en proyectos y el seguimiento de sus avances. A partir de estos procesos se identificaron las entidades necesarias y posteriormente se establecieron las relaciones entre ellas, de acuerdo con la información que debe ser almacenada y consultada en cada etapa.

Como mecanismo de identificación, las tablas utilizan “UUID” como clave primaria en lugar de identificadores enteros autoincrementales. Esta decisión permite evitar identificadores secuenciales que puedan facilitar la enumeración de registros desde el cliente, permite generar identificadores sin depender de una secuencia centralizada y facilita la integración con Supabase Auth, debido a que los usuarios autenticados cuentan con un identificador “UUID” en “auth.users”. De esta manera, la estructura de identificación de los usuarios puede mantenerse consistente entre el sistema de autenticación y las tablas propias de la aplicación.

Durante el diseño también se analizó la forma de almacenar los votos. Inicialmente se contempló una única tabla que relacionara directamente al usuario con la opción seleccionada. Sin embargo, este diseño permitiría establecer fácilmente qué opción fue seleccionada por un determinado ciudadano para cualquier usuario que tuviera acceso a los registros correspondientes. Debido a que el sistema contempla procesos de participación mediante votación, se consideró necesario evitar que la estructura de la base de datos permitiera asociar directamente la identidad del participante con su elección.

Por esta razón, el diseño final separa la información relacionada con la participación de aquella correspondiente a la opción seleccionada. Una estructura

registra que un ciudadano realizó su participación y permite controlar que no se registre más de un voto en el mismo proceso, mientras que otra almacena la opción seleccionada sin mantener una relación directa que permita identificar públicamente al participante. Esta separación busca reducir la exposición de la identidad de los votantes y mantener diferenciadas las funciones de control de participación y almacenamiento de las elecciones realizadas.

Tabla 6.

Conceptos de partes del sistema

Entidad	Propósito conceptual
roles	Catálogo cerrado de los dos roles del sistema.
usuarios	Perfil público de cada persona registrada: datos de contacto, alias público, avatar, y el rol y estado de la cuenta. Vinculada 1:1 con el usuario de autenticación gestionado por Supabase Auth.
categorías	Clasificación temática de las propuestas (infraestructura vial, espacios públicos, seguridad ciudadana, etc.), gestionable por el administrador.
propuestas	Iniciativa presentada por un ciudadano
propuesta_archivos	Imágenes de soporte adjuntas a una propuesta.
votaciones	Proceso de votación digital sobre una o varias propuestas, con ventana temporal definida y dos modalidades: selección múltiple o Sí/No.
votacion_opciones	Cada alternativa disponible dentro de una votación, vinculada a la propuesta que representa.
voto_registros	Certificado de participación: registra que un ciudadano determinado votó en una votación determinada.
votos	La elección emitida, sin ninguna columna que identifique directamente al votante. Su fecha puede coincidir con la de voto_registros
proyectos	Obra o iniciativa en ejecución, originada a partir de una propuesta aprobada
proyecto_actualizaciones	Bitácora de avances de un proyecto, con su porcentaje de progreso asociado.
proyecto_archivos	Evidencias fotográficas de un avance de proyecto, con el mismo patrón de referencia a Storage que propuesta_archivos.
notificaciones	Mensajes internos generados automáticamente ante eventos relevantes del sistema

	(aprobación de propuesta, apertura de votación, actualización de proyecto).
auditoria	Registro inmutable de las acciones administrativas sensibles, con su autor, la entidad afectada y detalle contextual.

Nota: Tabla elaborada por el autor a partir del análisis de las partes del sistema

4.5.2.1 Creación de la Base de Datos

- Roles

Figura 16. Creación de la tabla de roles

```

1
2 create extension if not exists pgcrypto;
3
4 -- 1. ROLES
5
6 create table public.roles (
7     id      uuid primary key default gen_random_uuid(),
8     nombre text not null unique check (nombre in ('ciudadano','administrador'))
9 );
10
11 insert into public.roles (nombre) values ('ciudadano'), ('administrador');
12
13

```

Fuente: Autor

Tabla 7.
Características de la tabla roles

Columna	Tipo	Restricciones	Descripción
id	uuid	PK	Identificador del rol
nombre	text	NOT NULL, UNIQUE, CHECK	Nombre del rol; restringido a los dos valores válidos del sistema

Nota: Tabla elaborada por el autor a partir del análisis para la base de datos

- Usuarios

Figura 17. Creación de la tabla usuarios

```

13
14 -- 2. USUARIOS
15
16 create table public.usuarios (
17     id          uuid primary key references auth.users(id) on delete cascade,
18     rol_id      uuid not null references public.roles(id),
19     nombre_completo text not null,
20     correo      text not null unique,
21     telefono    text,
22     avatar_url  text,
23     alias       text check (alias is null or (length(alias) between 2 and 40)),
24     activo      boolean not null default true,
25     fecha_registro timestampz not null default now()
26 );

```

Fuente: Autor

Tabla 8.
Características de la tabla Usuarios

Columna	Tipo	Restricciones	Descripción
id	uuid	PK, FK → auth.users(id), ON DELETE CASCADE	Comparte el mismo identificador que la cuenta de autenticación; si esta se elimina, el perfil se elimina con ella
rol_id	uuid	NOT NULL, FK → roles(id)	Rol asignado al usuario
nombre_completo	text	NOT NULL	Nombre legal del usuario
correo	text	NOT NULL, UNIQUE	Correo electrónico, espejo del usado en auth.users
telefono	text	---	Teléfono de contacto
avatar_url	text	---	URL pública de la imagen de perfil en Storage
alias	text	CHECK longitud 2–40	Nombre público alternativo, mostrado en propuestas en lugar del nombre completo cuando está definido
activo	boolean	NOT NULL, DEFAULT true	Habilita o inhabilita la cuenta a nivel de aplicación

fecha_registro	timestampz	NOT NULL, DEFAULT now()	Fecha de creación del perfil
notificaciones_externas_activas	bool	---	Activar o desactivar las notificaciones
fecha_aceptacion_terminos	timestampz	NOT NULL, DEFAULT now()	Fecha de aceptación de los términos, va a la par de la fecha_registro

Nota: Tabla elaborada por el autor a partir del análisis para la base de datos

- **Categorías**

Figura 18. Creación de la tabla categorías

```

68 |
69 -- 3. CATEGORIAS
70 |
71 create table public.categorias (
72   id          uuid primary key default gen_random_uuid(),
73   nombre      text not null unique,
74   descripcion text,
75   activo      boolean not null default true
76 );
77 |

```

Fuente: Autor

Tabla 9.
Características de la tabla categorías

Columna	Tipo	Restricciones	Descripción
id	uuid	PK	Identificador de la categoría
nombre	text	NOT NULL, UNIQUE	Nombre de la categoría
descripcion	text	---	Descripción ampliada, opcional
activo	boolean	NOT NULL, DEFAULT true	Determina si aparece disponible al crear una nueva propuesta

Nota: Tabla elaborada por el autor a partir del análisis para la base de datos

- **Propuestas**

Figura 19. Creación de la tabla propuestas

```
80 create table public.propuestas (
81     id                uuid primary key default gen_random_uuid(),
82     usuario_id        uuid not null references public.usuarios(id),
83     categoria_id       uuid not null references public.categorias(id),
84     titulo            text not null,
85     descripcion        text not null,
86     ubicacion         text,
87     estado            text not null default 'pendiente'
88     | | | | | check (estado in ('pendiente','en_revision','aprobada','rechazada','convertida_proyecto')),
89     observaciones      text,
90     revisado_por       uuid references public.usuarios(id),
91     fecha_revision     timestampz,
92     fecha_creacion     timestampz not null default now(),
93     fecha_actualizacion timestampz not null default now()
94 );
95
```

Fuente: Autor

Tabla 10.

Características de la tabla propuestas

Columna	Tipo	Restricciones	Descripción
id	uuid	PK	Identificador de la propuesta
usuario_id	uuid	NOT NULL, FK → usuarios(id)	Autor de la propuesta
categoria_id	uuid	NOT NULL, FK → categorias(id)	Categoría temática
titulo	text	NOT NULL	Título de la propuesta
descripcion	text	NOT NULL	Descripción detallada
ubicacion	text	---	Ubicación referencial en la ciudad (opcional)
estado	text	NOT NULL, CHECK	Estado dentro de su ciclo de vida
observaciones	text	---	Comentario del revisor al aprobar o rechazar
revisado_por	uuid	FK → usuarios(id)	Administrador que realizó la revisión
fecha_revision	timestampz	---	Momento en que se aprobó o rechazó
fecha_creacion	timestampz	NOT NULL, DEFAULT now()	Momento de creación
fecha_actualizacion	timestampz	NOT NULL, DEFAULT now()	Se actualiza automáticamente en cada modificación

Nota: Tabla elaborada por el autor a partir del análisis para la base de datos

- **Propuesta archivos**

Figura 20. Creación de la tabla propuestas_archivos

```
110 create table public.propuesta_archivos (
111     id          uuid primary key default gen_random_uuid(),
112     propuesta_id uuid not null references public.propuestas(id) on delete cascade,
113     url         text not null,
114     tipo        text,
115     fecha_subida timestampz not null default now()
116 );
```

Fuente: Autor

Tabla 11.

Características de la tabla propuestas_archivos

Columna	Tipo	Restricciones	Descripción
id	uuid	PK	Identificador del archivo
propuesta_id	uuid	NOT NULL, FK, ON DELETE CASCADE	Propuesta a la que pertenece la imagen
url	text	NOT NULL	Ruta del objeto dentro del bucket propuestas de Storage
tipo	text	---	Tipo de archivo
fecha_subida	timestampz	NOT NULL, DEFAULT now()	Momento de la subida

Nota: Tabla elaborada por el autor a partir del análisis para la base de datos

- **Votaciones**

Figura 21. Creación de la tabla de votaciones

```
120 create table public.votaciones (
121     id          uuid primary key default gen_random_uuid(),
122     titulo      text not null,
123     descripcion text,
124     fecha_inicio timestampz not null,
125     fecha_fin   timestampz not null,
126     estado      text not null default 'programada'
127     |         | check (estado in ('programada','activa','finalizada','cancelada')),
128     tipo        text not null default 'seleccion_multiple'
129     |         | check (tipo in ('seleccion_multiple','si_no')),
130     creado_por  uuid not null references public.usuarios(id),
131     fecha_creacion timestampz not null default now(),
132     check (fecha_fin > fecha_inicio)
133 );
```

Fuente: Autor

Tabla 12.

Características de la tabla votaciones

Columna	Tipo	Restricciones	Descripción
id	uuid	PK	Identificador de la votación
titulo	text	NOT NULL	Título de la votación
descripcion	text	---	Descripción del propósito de la votación
fecha_inicio	timestampz	NOT NULL	Apertura programada
fecha_fin	timestampz	NOT NULL, CHECK > fecha_inicio	Cierre programado
estado	text	NOT NULL, CHECK	Programada, activa, finalizada o cancelada.
tipo	text	NOT NULL, CHECK	Selección múltiple entre varias propuestas, o Sí/No sobre una sola
creado_por	uuid	NOT NULL, FK → usuarios(id)	Administrador que creó la votación
fecha_creacion	timestampz	NOT NULL, DEFAULT now()	Momento de creación del registro

Nota: Tabla elaborada por el autor a partir del análisis para la base de datos

- Votación Opciones**

Figura 22. Creación de la tabla de las opciones de votación

```

137 create table public.votacion_opciones (
138     id          uuid primary key default gen_random_uuid(),
139     votacion_id  uuid not null references public.votaciones(id) on delete cascade,
140     propuesta_id uuid not null references public.propuestas(id),
141     etiqueta     text not null default '',
142     orden        int not null default 0,
143     unique (votacion_id, propuesta_id, etiqueta)
144 );
145

```

Fuente: Autor

Tabla 13.

Características de la tabla de las opciones de votación

Columna	Tipo	Restricciones	Descripción
id	uuid	PK	Identificador de la opción
votacion_id	uuid	NOT NULL, FK, ON DELETE CASCADE	Votación a la que pertenece
propuesta_id	uuid	NOT NULL, FK → propuestas(id)	Propuesta que representa esta opción

etiqueta	text	NOT NULL, DEFAULT "	Vacía en selección múltiple; "Sí" o "No" en votaciones de ese tipo
orden	int	NOT NULL, DEFAULT 0	Orden de presentación en la interfaz

Nota: Tabla elaborada por el autor a partir del análisis para la base de datos

• Voto Registros

Figura 23. Creación de la tabla registros de votos

```
148 create table public.voto_registros (
149     id          uuid primary key default gen_random_uuid(),
150     usuario_id  uuid not null references public.usuarios(id),
151     votacion_id uuid not null references public.votaciones(id),
152     fecha_voto  timestampz not null default now(),
153     unique (usuario_id, votacion_id)
154 );
```

Fuente: Autor

Tabla 14.

Características de la tabla registros de votos

Columna	Tipo	Restricciones	Descripción
id	uuid	PK	Identificador del registro de participación
usuario_id	uuid	NOT NULL, FK → usuarios(id)	Ciudadano que participó
votacion_id	uuid	NOT NULL, FK → votaciones(id)	Votación en la que participó
fecha_voto	timestampz	NOT NULL, DEFAULT now()	Momento de la participación

Nota: Tabla elaborada por el autor a partir del análisis para la base de datos

• Votos

Figura 24. Creación de la tabla de votos

```
158 create table public.votos (
159     id          uuid primary key default gen_random_uuid(),
160     votacion_id  uuid not null references public.votaciones(id),
161     votacion_opcion_id uuid not null references public.votacion_opciones(id),
162     fecha        timestampz not null default now()
163 );
```

Fuente: Autor

Tabla 15.

Características de la tabla votos

Columna	Tipo	Restricciones	Descripción
id	uuid	PK	Identificador del voto
votacion_id	uuid	NOT NULL, FK → votaciones(id)	Votación a la que pertenece
votacion_opcion_id	uuid	NOT NULL, FK → votacion_opciones(id)	Opción elegida
fecha	timestampz	NOT NULL, DEFAULT now()	Momento del voto

Nota: Tabla elaborada por el autor a partir del análisis para la base de datos

- **Proyectos**

Figura 25. Creación de la tabla de proyectos

```

225 create table public.proyectos (
226     id                uuid primary key default gen_random_uuid(),
227     propuesta_id      uuid unique references public.propuestas(id),
228     responsable_id    uuid not null references public.usuarios(id),
229     titulo            text not null,
230     descripcion       text,
231     estado            text not null default 'planeado'
232     | | | | | | | | | | check (estado in ('planeado','en_ejecucion','finalizado','suspendido')),
233     porcentaje_avance int not null default 0 check (porcentaje_avance between 0 and 100),
234     fecha_inicio      date,
235     fecha_fin_estimada date,
236     fecha_creacion    timestampz not null default now(),
237     check (fecha_fin_estimada is null or fecha_inicio is null or fecha_fin_estimada >= fecha_inicio)
238 );
239

```

Fuente: Autor

Tabla 16.

Características de la tabla proyectos

Columna	Tipo	Restricciones	Descripción
id	uuid	PK	Identificador del proyecto
propuesta_id	uuid	UNIQUE, FK → propuestas(id)	Propuesta de origen; única para impedir crear dos proyectos desde la misma propuesta
responsable_id	uuid	NOT NULL, FK → usuarios(id)	Administrador responsable del seguimiento
titulo	text	NOT NULL	Título del proyecto
descripcion	text	---	Descripción del alcance

estado	text	NOT NULL, CHECK	Planeado, en ejecución, finalizado o suspendido
porcentaje_avance	int	NOT NULL, CHECK 0-100	Porcentaje de avance, sincronizado automáticamente
fecha_inicio	date	---	Fecha calendario de inicio
fecha_fin_estimada	date	CHECK ≥ fecha_inicio	Fecha estimada de cierre
fecha_creacion	timestampz	NOT NULL, DEFAULT now()	Momento de conversión desde la propuesta

Nota: Tabla elaborada por el autor a partir del análisis para la base de datos

- **Proyecto Actualizaciones**

Figura 26. Creación de la tabla actualizaciones de proyectos

```

242 create table public.proyecto_actualizaciones (
243     id          uuid primary key default gen_random_uuid(),
244     proyecto_id uuid not null references public.proyectos(id) on delete cascade,
245     autor_id    uuid not null references public.usuarios(id),
246     descripcion text not null,
247     porcentaje_avance int check (porcentaje_avance between 0 and 100),
248     fecha       timestampz not null default now()
249 );

```

Fuente: Autor

Tabla 17.

Características de la tabla actualizaciones de proyectos

Columna	Tipo	Restricciones	Descripción
id	uuid	PK	Identificador de la actualización
proyecto_id	uuid	NOT NULL, FK, ON DELETE CASCADE	Proyecto al que pertenece
autor_id	uuid	NOT NULL, FK → usuarios(id)	Administrador que registró el avance
descripcion	text	NOT NULL	Descripción del avance realizado
porcentaje_avance	int	CHECK 0-100	Nuevo porcentaje, si el avance lo incluye
fecha	timestampz	NOT NULL, DEFAULT now()	Momento del registro

Nota: Tabla elaborada por el autor a partir del análisis para la base de datos

- **Proyecto Archivos**

Figura 27. Creación de la tabla proyecto archivos

```
271 -- 12. PROYECTO_ARCHIVOS
272
273 create table public.proyecto_archivos (
274     id          uuid primary key default gen_random_uuid(),
275     actualizacion_id uuid not null references public.proyecto_actualizaciones(id) on delete cascade,
276     url         text not null,
277     tipo        text,
278     fecha_subida timestampz not null default now()
279 );
```

Fuente: Autor

Tabla 18.

Características de la tabla proyectos archivos

Columna	Tipo	Restricciones	Descripción
id	uuid	PK	Identificador del archivo
actualizacion_id	uuid	NOT NULL, FK, ON DELETE CASCADE	Actualización a la que pertenece la evidencia
url	text	NOT NULL	Ruta del objeto en el bucket evidencias de Storage
tipo	text	---	Tipo de archivo
fecha_subida	timestampz	NOT NULL, DEFAULT now()	Momento de la subida

Nota: Tabla elaborada por el autor a partir del análisis para la base de datos

- **Notificaciones**

Figura 28. Creación de la tabla notificaciones

```
283 create table public.notificaciones (
284     id          uuid primary key default gen_random_uuid(),
285     usuario_id  uuid not null references public.usuarios(id) on delete cascade,
286     titulo      text not null,
287     mensaje     text not null,
288     tipo        text,
289     leida       boolean not null default false,
290     fecha       timestampz not null default now()
291 );
```

Fuente: Autor

Tabla 19.

Características de la tabla notificaciones

Columna	Tipo	Restricciones	Descripción
id	uuid	PK	Identificador de la notificación

usuario_id	uuid	NOT NULL, FK, ON DELETE CASCADE	Destinatario
titulo	text	NOT NULL	Título breve
mensaje	text	NOT NULL	Contenido del mensaje
tipo	text	---	Clasificación (aprobación, rechazo, votación activa, etc.).
leida	boolean	NOT NULL, DEFAULT false	Estado de lectura
fecha	timestampz	NOT NULL, DEFAULT now()	Momento de generación

Nota: Tabla elaborada por el autor a partir del análisis para la base de datos

- Auditoria**

Figura 29. Creación de la tabla auditoria

```

293 -- 14. AUDITORIA
294
295 create table public.auditoria (
296     id          uuid primary key default gen_random_uuid(),
297     usuario_id  uuid not null references public.usuarios(id),
298     accion      text not null,
299     entidad     text not null,
300     entidad_id  uuid,
301     detalle     jsonb,
302     fecha       timestampz not null default now()
303 );

```

Fuente: Autor

Tabla 20.

Características de la tabla de auditoría

Columna	Tipo	Restricciones	Descripción
id	uuid	PK	Identificador del registro de auditoría
usuario_id	uuid	NOT NULL, FK → usuarios(id)	Autor de la acción auditada
accion	text	NOT NULL	Nombre de la acción
entidad	text	NOT NULL	Tabla afectada
entidad_id	uuid	---	Identificador del registro afectado
detalle	jsonb	---	Información contextual de estructura variable según la acción
fecha	timestampz	NOT NULL, DEFAULT now()	Momento de la acción

Nota: Tabla elaborada por el autor a partir del análisis para la base de datos

4.5.2.2 Funciones y disparadores de integridad de datos

- Sincronización entre autenticación y perfil

Figura 30. Función de sincronización autenticación y perfil

```

28 create or replace function public.handle_new_user()
29 returns trigger
30 language plpgsql
31 security definer set search_path = public
32 as $$
33 declare
34     v_rol_ciudadano uuid;
35 begin
36     select id into v_rol_ciudadano from public.roles where nombre = 'ciudadano';
37
38     insert into public.usuarios (id, rol_id, nombre_completo, correo)
39     values (
40         new.id,
41         v_rol_ciudadano,
42         coalesce(new.raw_user_meta_data ->> 'nombre_completo', ''),
43         new.email
44     );
45     return new;
46 end;
47 $$;
48
49 create trigger on_auth_user_created
50 after insert on auth.users
51 for each row execute function public.handle_new_user();

```

Fuente: Autor

Este mecanismo garantiza que todo registro nuevo en auth.users tenga siempre una fila correspondiente en usuarios, asignada por defecto al rol de menor privilegio. Se ejecuta con SECURITY DEFINER porque el disparador necesita insertar en public.usuarios como consecuencia de una operación sobre el esquema auth, a la que el flujo de registro del cliente no tiene acceso directo de escritura.

- Actualización automática de la marca temporal

Figura 31. Función actualización automática marca temporal

```

96 create or replace function public.set_fecha_actualizacion()
97 returns trigger language plpgsql as $$
98 begin
99     new.fecha_actualizacion = now();
100     return new;
101 end;
102 $$;
103
104 create trigger trg_propuestas_actualizacion
105 before update on public.propuestas
106 for each row execute function public.set_fecha_actualizacion();

```

Fuente: Autor

Esta función mantiene fecha_actualizacion de modo que refleje siempre el momento real de la última modificación de una propuesta, sin depender de que el cliente la envíe manualmente en cada actualización.

- **Sincronización del porcentaje de avance de un proyecto**

Figura 32. Función porcentaje de avance de un proyecto

```

251 create or replace function public.actualizar_avance_proyecto()
252 returns trigger
253 language plpgsql
254 security definer set search_path = public
255 as $$
256 begin
257     if new.porcentaje_avance is not null then
258         update public.proyectos
259         set porcentaje_avance = new.porcentaje_avance,
260             estado = case when new.porcentaje_avance = 100 then 'finalizado' else estado end
261         where id = new.proyecto_id;
262     end if;
263     return new;
264 end;
265 $$;
266
267 create trigger trg_actualizar_avance
268 after insert on public.proyecto_actualizaciones
269 for each row execute function public.actualizar_avance_proyecto();

```

Fuente: Autor

Con esta función se evita que proyectos.porcentaje_avance se desincronice de su historial de actualizaciones cada vez que se registra un nuevo avance con

porcentaje, el proyecto se actualiza en la misma operación, y si el avance alcanza el 100%, el proyecto se marca automáticamente como finalizado.

4.5.2.3 Estructura de Seguridad

Matriz consolidada de políticas

Tabla 21.

Permisos de usuario según el Rol

Tabla	Ciudadano	Administrador
roles	Lectura del catálogo completo	Igual que ciudadano
usuarios	Lectura y edición únicamente de su propia fila; no puede alterar su propio rol ni activo	Lectura y gestión de cualquier fila
categorías	Solo lectura de las categorías activas	Lectura y gestión completa
propuestas	Lectura de las propias y de las de estado aprobada; creación y edición de las propias, solo mientras estén en pendiente y su cuenta esté activa	Lectura y gestión completa
propuesta_archivos	Según la visibilidad de la propuesta asociada; subida solo en propuestas propias y pendientes	Gestión completa
votaciones / votacion_opciones	Solo lectura	Gestión completa
voto_registros	Lectura únicamente de su propia participación	Lectura de la participación de cualquiera
votos	Sin acceso de ningún tipo	Sin acceso para ningún rol de la aplicación; ni siquiera el administrador tiene una política de lectura sobre esta tabla
proyectos / actualizaciones / archivos	Solo lectura	Gestión completa
notificaciones	Lectura y marcado de leídas únicamente de las propias	Igual que ciudadano
auditoria	Sin acceso	Solo lectura

Nota: Tabla elaborada por el autor a partir del análisis de los permisos

La ausencia total de políticas sobre votos es la pieza central del diseño de privacidad del voto, no es que el acceso esté restringido a un subconjunto de columnas o filas, es que no existe ningún camino de lectura directa para ningún rol

de la aplicación, el único acceso autorizado a resultados agregados es a través de la vista resultados_votacion.

Ejemplos representativos de política

Este es un ejemplo de una política simple el cual tiene un patrón repetido en la mayoría de las tablas, el resto de políticas se agregarán en los anexos.

Figura 33. Política del estado pendiente

```
388 create policy propuestas_update_propia on public.propuestas
389 for update to authenticated
390 using (usuario_id = auth.uid() and estado = 'pendiente')
391 with check (usuario_id = auth.uid() and estado = 'pendiente');
```

Fuente: Autor

La condición no se limita a verificar propiedad (usuario_id = auth.uid()): también exige que la propuesta siga en estado pendiente (lo que impide editar una propuesta ya revisada) y que la cuenta del usuario esté activa. Un ciudadano cuya cuenta fue desactivada por un administrador queda bloqueado de esta operación.

Alcance de la protección del voto. La separación entre voto_registros y votos protege la elección frente a los roles de la aplicación (ciudadano y administrador), que no tienen política de lectura sobre votos. No protege frente a quien tenga acceso directo a la base de datos con privilegios de propietario, como el titular del proyecto en Supabase. Además, ambas tablas asignan la fecha con now(), que dentro de una transacción devuelve la misma marca de tiempo; como emitir_voto() inserta en las dos tablas en una sola transacción, las fechas coinciden y permitirían emparejar la participación con la elección. En esta versión el riesgo se declara como limitación, y la mitigación se plantea en las recomendaciones.

4.5.2.4 Funciones de apoyo

Tabla 22.
Funciones de apoyo

Función	Propósito
es_admin()	Determina si el usuario autenticado tiene rol de administrador; evita la recursión de políticas sobre usuarios
usuario_activo()	Determina si la cuenta del usuario autenticado está activa; usada como condición adicional en las políticas de escritura de propuestas y en emitir_voto()
prevenir_escalada_privilegios()	Disparador BEFORE UPDATE sobre usuarios que impide que cualquier actualización, modifique rol_id o activo, salvo que quien la ejecute sea administrador o el acceso provenga directamente de la base de datos

Nota: Tabla elaborada por el autor

- **Emitir Voto**

Figura 34. Función de emitir voto

```

165 create or replace function public.emitir_voto(p_votacion_id uuid, p_opcion_id uuid)
166 returns void
167 language plpgsql
168 security definer set search_path = public
169 as $$
170 declare
171     v_estado text;
172     v_inicio timestamptz;
173     v_fin timestamptz;
174     v_valida boolean;
175 begin
176     if auth.uid() is null then
177         raise exception 'Debe iniciar sesión para votar';
178     end if;
179
180     select estado, fecha_inicio, fecha_fin
181     into v_estado, v_inicio, v_fin
182     from public.votaciones
183     where id = p_votacion_id
184     for update;
185

```

Fuente: Autor

Figura 35. Función de emitir voto 2

```

186 if not found then
187     raise exception 'La votación no existe';
188 end if;
189
190 if v_estado <> 'activa' or now() < v_inicio or now() > v_fin then
191     raise exception 'La votación no está activa en este momento';
192 end if;
193
194 select exists (
195     select 1 from public.votacion_opciones
196     where id = p_opcion_id and votacion_id = p_votacion_id
197 ) into v_valida;
198
199 if not v_valida then
200     raise exception 'La opción no pertenece a esta votación';
201 end if;
202
203 insert into public.voto_registros (usuario_id, votacion_id)
204 values (auth.uid(), p_votacion_id);
205
206 insert into public.votos (votacion_id, votacion_opcion_id)
207 values (p_votacion_id, p_opcion_id);
208 end;
209 $$;

```

Fuente: Autor

El control de duplicados no depende únicamente de la restricción UNIQUE de voto_registros. La función valida explícitamente el estado y la ventana temporal de la votación antes de escribir. Por ello, un intento de voto fuera de horario falla con un mensaje explicativo y no con un error genérico de base de datos. Los permisos de escritura directa sobre votos y voto_registros están revocados para el rol authenticated (REVOKE INSERT, UPDATE, DELETE). Esta función es el único camino de escritura posible.

4.5.2.5 Vistas del modelo

- Resultados votación

Figura 36. Creación de la vista de resultados de votación

```
212 create view public.resultados_votacion as
213 select
214     vo.votacion_id,
215     vo.id as votacion_opcion_id,
216     p.titulo as propuesta_titulo,
217     count(v.id) as total_votos,
218     vo.etiqueta,
219     case when vo.etiqueta <> '' then vo.etiqueta else p.titulo end as opcion_titulo
220 from public.votacion_opciones vo
221 left join public.votos v on v.votacion_opcion_id = vo.id
222 join public.propuestas p on p.id = vo.propuesta_id
223 group by vo.votacion_id, vo.id, p.titulo, vo.etiqueta;
```

Fuente: Autor

Esta vista no se declaró con security_invoker, por lo que se ejecuta con los privilegios de quien la creó. Por esta razón puede leer internamente la tabla votos para calcular el conteo. A quien la consulta solo se le concede GRANT SELECT sobre la vista.

- **Perfiles públicos**

Figura 37. Creación de la vista perfiles publicos

```
447 create view public.perfiles_publicos as
448 select id, alias, nombre_completo
449 from public.usuarios
450 where activo = true;
```

Fuente: Autor

La tabla usuarios restringe la lectura de la fila de un tercero para proteger su correo y teléfono, pero el sistema necesita mostrar el nombre o alias del autor de una propuesta a cualquier ciudadano. Esta vista expone únicamente las tres columnas no sensibles, para cualquier usuario activo.

- **Almacenamiento de archivos**

Los archivos imágenes de propuestas, evidencias de avance, avatares no residen en PostgreSQL, se almacenan en buckets de Supabase Storage, cuya metadata

(storage.objects) está sujeta al mismo mecanismo de RLS que las tablas del modelo.

Tabla 23.

Políticas de acceso a los buckets de almacenamiento

Bucket	Visibilidad	Quién sube	Quién elimina
avatares	Público (lectura libre)	El propio usuario, en su carpeta	El propio usuario
propuestas	Privado - visible solo si la propuesta asociada es del propio usuario, está aprobada, o el usuario es administrador	El autor de la propuesta, mientras esté pendiente	El autor o un administrador
evidencias	Privado - visible para cualquier autenticado	Solo administrador	Solo administrador

Nota: Tabla elaborada por el autor

Figura 38. Creación política de seguridad para propuestas

```

531 create policy propuestas_files_select on storage.objects
532   for select to authenticated
533   using (
534     bucket_id = 'propuestas'
535     and exists (
536       select 1 from public.propuestas p
537       where p.id::text = (storage.foldername(name))[2]
538         and (p.usuario_id = auth.uid() or p.estado = 'aprobada' or public.es_admin())
539     )
540 );

```

Fuente: Autor

Esta política replica, sobre el archivo, exactamente la misma condición de visibilidad que la tabla propuestas aplica sobre el registro al que pertenece, de modo que se evita que una imagen quede accesible para alguien que no podría ver la propuesta misma.

Automatización mediante tareas programadas

El único proceso que se ejecuta sin intervención de un cliente es la transición automática del ciclo de vida de una votación, mediante la extensión pg_cron

Figura 39. Automatización de las votaciones programadas

```
507 select cron.schedule(  
508   'actualizar-votaciones-programadas',  
509   '* * * * *',  
510   $$select public.actualizar_votaciones_programadas();$$  
511 );
```

Fuente: Autor

Esta tarea corre cada minuto y aplica tres reglas: activa una votación programada cuya fecha de inicio ya llegó, finaliza una votación activa cuya fecha de fin ya pasó, y como caso límite finaliza directamente una votación que seguía programada pero cuya ventana completa ya transcurrió sin haber sido activada a tiempo

4.5.2.6 Conexión con supabase

Toda la comunicación entre la aplicación Flutter y el backend pasa por un único paquete: supabase_flutter. Las credenciales nunca están escritas directamente en el código fuente. Viven en un archivo .env el cual está excluido de git mediante .gitignore, empaquetado como un asset de Flutter y leído en tiempo de ejecución con flutter_dotenv

Figura 40. Conexión con Supabase

```

4  class SupabaseConfig {
5      SupabaseConfig._();
6
7      static String get url {
8          final value = dotenv.env['SUPABASE_URL'];
9          if (value == null || value.isEmpty) {
10             throw StateError(
11                 'SUPABASE_URL no esta definida. Verifica que exista un archivo ',
12             );
13         }
14         return value;
15     }
16
17     static String get anonKey {
18         final value = dotenv.env['SUPABASE_ANON_KEY'];
19         if (value == null || value.isEmpty) {
20             throw StateError(
21                 'SUPABASE_ANON_KEY no está definida. Verifica tu archivo .env.',
22             );
23         }
24         return value;
25     }
26 }

```

Fuente: Autor

La única credencial que la app conoce es la clave anon la cual es pública nunca la service_role, que puede saltarse toda la seguridad. Esto es seguro porque la protección real del sistema no depende de mantener esta clave en secreto, sino de las políticas de Row Level Security descritas en la documentación del modelo de datos: aunque alguien extrajera la clave anon del build, seguiría sujeto exactamente a las mismas restricciones que cualquier usuario de la app.

Inicialización del cliente

El cliente se inicializa una sola vez, al arrancar la aplicación, antes de construir cualquier interfaz

Figura 41. Inicialización del usuario

```
Run | Debug | Profile
Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  usePathUrlStrategy();
  await dotenv.load(fileName: '.env');

  await Supabase.initialize(
    url: SupabaseConfig.url,
    anonKey: SupabaseConfig.anonKey,
    authOptions: const FlutterAuthClientOptions(
      authFlowType: AuthFlowType.implicit,
    ),
  );

  runApp(const ProviderScope(child: ComunidadBgaApp()));
}
```

Fuente: Autor

El paquete expone una única instancia global que gestiona por sí sola la sesión, sus tokens y su renovación automática.

Patrón de repositorio

Cada dominio funcional (autenticación, propuestas, votaciones, proyectos, notificaciones, perfil, administración, métricas) tiene su propia clase de repositorio, que recibe el cliente por inyección de dependencias y encapsula todas las llamadas a Supabase de ese dominio

Figura 42. Patrón y características del repositorio

```

8  class ProposalsRepository {
9      ProposalsRepository(this._client);
10     final SupabaseClient _client;
11
12     static const _selectConLookups =
13         '*, categorias(nombre), propuesta_archivos(id, url)';
14
15     Future<List<Propuesta>> _conNombresAutor(List<Propuesta> propuestas) async {
16         if (propuestas.isEmpty) return propuestas;
17         final autorIds = propuestas.map((p) => p.usuarioId).toSet().toList();
18         final perfiles =
19             await _client.from('perfiles_publicos').select('id, alias, nombre_completo').inFilter('id', autorIds);
20         final nombrePorId = <String, String>{
21             for (final p in perfiles)
22                 p['id'] as String: (((p['alias'] as String?)?.isEmpty ?? false)
23                     ? p['alias'] as String
24                     : (p['nombre_completo'] as String? ?? 'Ciudadano')),
25         };
26         return propuestas.map((p) => p.copyWith(autorNombre: nombrePorId[p.usuarioId] ?? 'Ciudadano')).toList();
27     }
28
29     Future<List<Categoria>> listarCategorias() async {
30         final data = await _client
31             .from('categorias')
32             .select('id, nombre')
33             .eq('activo', true)
34             .order('nombre');
35         return data.map((m) => Categoria.fromMap(m)).toList();
36     }

```

Fuente: Autor

Figura 43. Configuración del proveedor de repositorio de propuestas

```

8  final proposalsRepositoryProvider = Provider<ProposalsRepository>((ref) {
9      return ProposalsRepository(ref.watch(supabaseClientProvider));
10 }); // Provider

```

Fuente: Autor

Tres canales de comunicación con el backend

Consultas directas para operaciones simples de lectura/escritura, siempre sujetas a las políticas de RLS de la tabla correspondiente, el sistema nunca asume que una escritura tendrá éxito solo porque el formulario la envió; el resultado depende de si la política lo permite.

Llamadas RPC, validaciones cruzadas, o que combina varias escrituras que deben ocurrir de forma atómica, es el canal usado para emitir un voto, aprobar una

propuesta, crear una votación, registrar un avance de proyecto, etc. En estos casos, la validación vive en la función de PostgreSQL, no en Flutter; el cliente solo la invoca y reacciona a su resultado o a su excepción.

Storage de archivos como imágenes de propuestas, evidencias de proyectos, avatares. Los buckets privados como propuestas y evidencias se resuelven con URLs firmadas de vigencia corta (createSignedUrl). El bucket público (avatares) usa getPublicUrl directamente

Manejo de errores centralizado

Ningún widget captura excepciones crudas de Supabase. Todas pasan por un traductor único.

Figura 44. Traducciones de errores en Supabase

```
5  AppException mapSupabaseError(Object error) {
6      if (error is AppException) return error;
7      if (error is AuthException) {
8          return AppException(_mensajeAuth(error.message));
9      }
10     if (error is PostgreSQLException) {
11         return AppException(_mensajePostgreSQLException(error));
12     }
13     if (error is StorageException) {
14         return AppException('No se pudo procesar el archivo: ${error.message}');
15     }
16     return const AppException('Ocurrió un error de conexión. Intenta de nuevo.');
```

Fuente: Autor

Sesión reactiva y enrutamiento protegido

La verificación de sesión no se realiza de forma manual en cada pantalla; se gestiona mediante un flujo reactivo de autenticación. Un flujo (stream) expone los cambios de autenticación en tiempo real.

Figura 45. Gestión de cambios de autenticación mediante un StreamProvider

```
12 final authStateChangesProvider = StreamProvider<AuthState>((ref) {
13     return ref.watch(authRepositoryProvider).authStateChanges;
14 });
```

Fuente: Autor

El enrutador se suscribe al mismo flujo para decidir, en cada navegación, si redirigir a /login, a /home o bloquear una ruta administrativa. De este modo, iniciar o cerrar sesión desde cualquier punto de la aplicación reconfigura automáticamente la navegación disponible.

4.5.3. Implementación de los Módulos

Durante el desarrollo, la verificación obligatoria de correo electrónico se mantuvo desactivada para agilizar las pruebas. Antes de considerar el sistema listo para usuarios reales, esta opción se reactivó en Authentication → Providers → Email, con lo que se cierra la posibilidad de registrar una cuenta con un correo ajeno sin confirmarlo. Se documenta el desarrollo de los cinco módulos principales del sistema.

4.5.3.1 Módulo de Registro y Publicación de Propuestas

La implementación del módulo de propuestas permitió registrar, consultar y visualizar las iniciativas comunitarias. Mediante inserciones y consultas directas sobre la tabla propuestas, protegidas por las políticas de Row Level Security que restringen la edición a propuestas propias en estado pendiente; las imágenes se suben al bucket privado propuestas de Storage y se referencian por su ruta. El formulario desarrollado en Flutter valida los datos ingresados y posteriormente realiza la inserción correspondiente en la tabla propuestas de Supabase; la propuesta queda disponible de inmediato para el proceso de revisión administrativa.

Figura 46. Función crear propuesta del módulo de registros

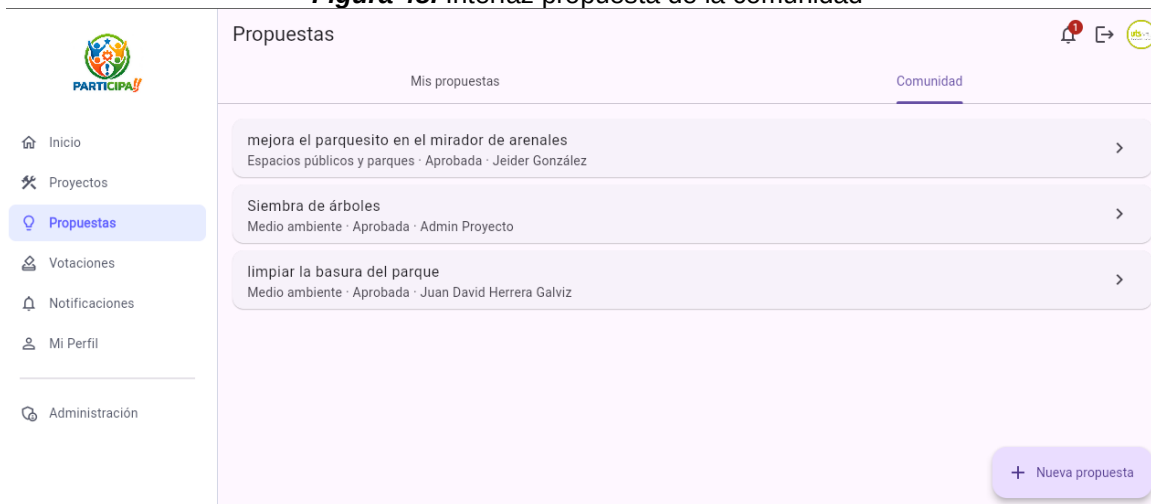
```
72 Future<String> crearPropuesta({
73     required String categoriaId,
74     required String titulo,
75     required String descripcion,
76     String? ubicacion,
77 }) async {
78     final userId = _client.auth.currentUser!.id;
79     final data = await _client
80         .from('propuestas')
81         .insert({
82             'usuario_id': userId,
83             'categoria_id': categoriaId,
84             'titulo': titulo,
85             'descripcion': descripcion,
86             'ubicacion': ubicacion,
87         })
88         .select('id')
89         .single();
90     return data['id'] as String;
91 }
```

Fuente: Autor

Figura 47. Interfaz registrar nueva propuesta

Fuente: Autor

Figura 48. Interfaz propuesta de la comunidad



Fuente: Autor

4.5.3.2 Módulo de Votaciones

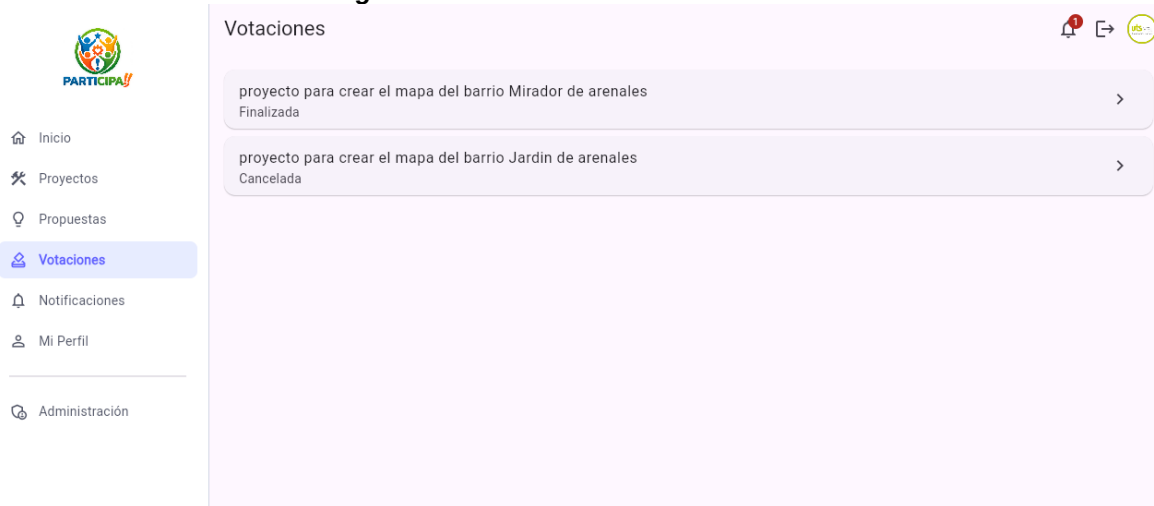
La implementación del módulo de votaciones permitió priorizar propuestas de forma digital. Garantizó que cada ciudadano participe una única vez sin que su elección individual quede expuesta a los roles de la aplicación, incluido el administrador. El voto no se inserta directamente: se invoca la función `emitir_voto()` mediante `.rpc()`, que valida y registra en una sola transacción atómica. Los resultados se consultan como una lectura normal sobre la vista agregada.

Figura 49. Función emitir voto del módulo votaciones

```
51 Future<void> emitirVoto({required String votacionId, required String opcionId}) async {
52     await _client.rpc('emitir_voto', params: {
53         'p_votacion_id': votacionId,
54         'p_opcion_id': opcionId,
55     });
56 }
```

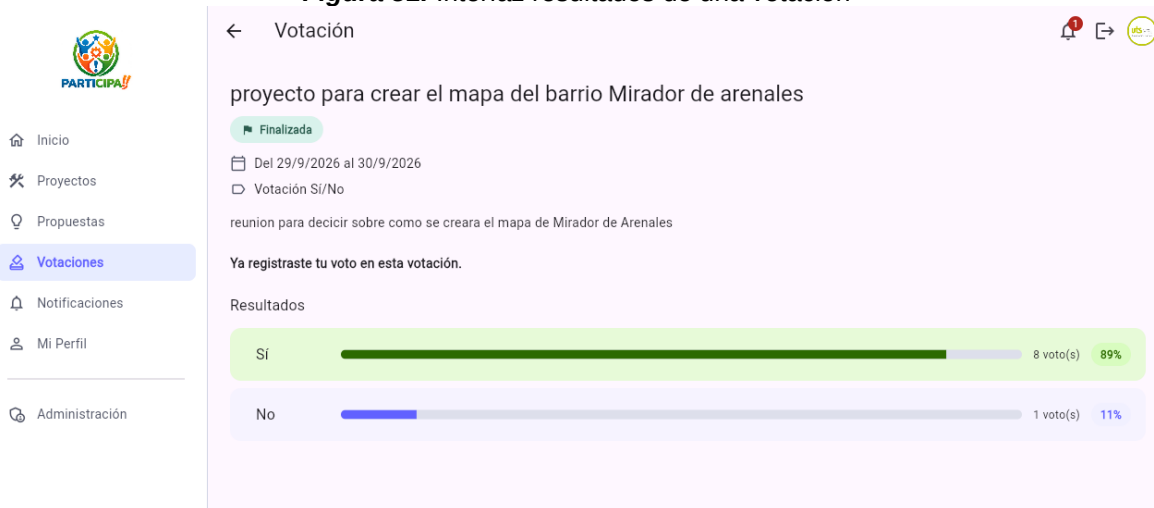
Fuente: Autor

Figura 50. Interfaz módulo de votaciones



Fuente: Autor

Figura 51. Interfaz resultados de una votación



Fuente: Autor

4.5.3.3 Módulo de Seguimiento de Proyectos

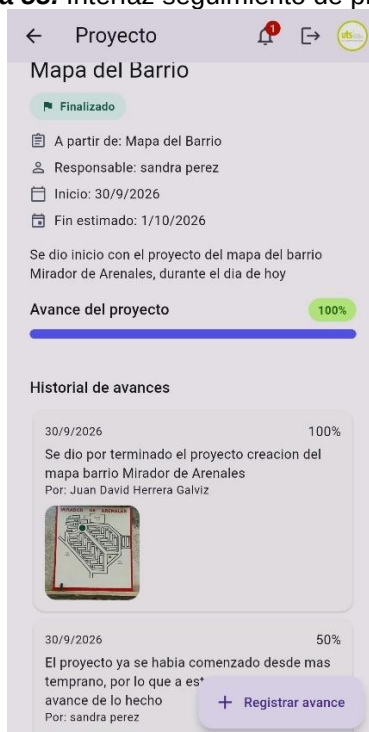
La conversión y el registro de avances se realizan mediante funciones .rpc(), mientras que la consulta del historial usa .select() con relaciones anidadas hacia el autor y las evidencias; el porcentaje del proyecto se sincroniza automáticamente mediante un disparador de base de datos, no desde el cliente.

Figura 52. Función registrar avance del módulo proyectos

```
65  Future<String> registrarAvance({
66      required String proyectoId,
67      required String descripcion,
68      int? porcentajeAvance,
69  }) async {
70      final id = await _client.rpc('registrar_avance_proyecto', params: {
71          'p_proyecto_id': proyectoId,
72          'p_descripcion': descripcion,
73          'p_porcentaje_avance': porcentajeAvance,
74      });
75      return id as String;
76  }
77
```

Fuente: Autor

Figura 53. Interfaz seguimiento de proyecto



Fuente: Autor

Figura 54. Interfaz registrar un avance de proyecto

Fuente: Autor

4.5.3.4 Módulo de Métricas de Participación

Una única llamada `.rpc('obtener_metricas')` devuelve un objeto jsonb con todos los indicadores ya calculados, con lo que el cliente no tiene que hacer diez consultas separadas ni sumarlas por su cuenta. El acceso está restringido exclusivamente al rol de administrador dentro de la propia función

Figura 55. Función obtener métricas del módulo métricas

```
10 Future<Metricas> obtenerMetricas() async {
11   final data = await _client.rpc('obtener_metricas');
12   return Metricas.fromMap(data as Map<String, dynamic>);
13 }
```

Fuente: Autor

Figura 56. Interfaz módulo métricas



Fuente: Autor

Figura 57. Interfaz módulo métricas parte gráfica



Fuente: Autor

4.5.3.5 Módulo de Autenticación y Perfiles

Se comunica con Supabase a través del módulo supabase.auth para todo lo relacionado con credenciales, y mediante consultas normales sobre usuarios para los datos de perfil, el avatar se sube al bucket público avatares de Storage.

Figura 58. Función Iniciar sesión del módulo autenticación

```
24 Future<void> iniciarSesion({required String correo, required String password}) async {
25   await _client.auth.signInWithPassword(email: correo, password: password);
26
27   final userId = _client.auth.currentUser?.id;
28   if (userId != null) {
29     final perfil = await _client.from('usuarios').select('activo').eq('id', userId).maybeSingle();
30     if (perfil != null && perfil['activo'] == false) {
31       await _client.auth.signOut();
32       throw const AppException('Tu cuenta ha sido desactivada. Contacta a un administrador.');
```

Fuente: Autor

Figura 59. Interfaz de la sección perfil del usuario

Fuente: Autor

4.6. Despliegue del Sistema

El despliegue se diseñó con prioridad en la simplicidad y el control directo sobre la infraestructura, de acuerdo con el alcance de un proyecto de grado. La aplicación web se compila localmente y el resultado estático se sube a Vercel. No se delegó la compilación a un entorno de integración continua, porque este tendría que instalar el SDK completo de Flutter en cada despliegue. Esa pieza de infraestructura no se justifica para un sistema que se despliega pocas veces y no de forma continua. La aplicación Android se compila y firma localmente y genera un instalador (.apk) que no depende de la publicación en una tienda de aplicaciones para poder evaluarse.

4.6.1. Despliegue de la aplicación Android

Todo build de producción de Android debe ir firmado criptográficamente con una clave propia del desarrollador, distinta de la clave de depuración que usa el entorno de pruebas. Se generó una clave de carga (upload key) mediante la herramienta keytool del JDK:

Figura 60. Comando keytool

```
PS C:\Users\Usuario\Documents\A_PROYECTO DE GRADO\Proyecto\proyecto_final> keytool -genkey -v -keystore "participa-keystore.jks" -keyalg RSA -keysize 2048 -validity 10000 -alias upload
```

Fuente: Autor

Esta clave y sus credenciales se referencian desde android/key.properties, un archivo excluido explícitamente del control de versiones, ya que su exposición comprometería la capacidad de publicar actualizaciones legítimas de la aplicación.

Configuración de la firma en el proyecto

El archivo de construcción de Gradle del módulo de la aplicación se configuró para aplicar automáticamente esta firma a cualquier compilación de tipo release:

Figura 61. Configuración de la firma en el proyecto

```

37 signingConfigs {
38     create("release") {
39         keyAlias = keystoreProperties["keyAlias"] as String?
40         keyPassword = keystoreProperties["keyPassword"] as String?
41         storeFile = keystoreProperties["storeFile"]?.let { file(it) }
42         storePassword = keystoreProperties["storePassword"] as String?
43     }
44 }
45
46 buildTypes {
47     release {
48         // TODO: Add your own signing config for the release build.
49         // Signing with the debug keys for now, so `flutter run --release` works.
50         signingConfig = signingConfigs.getByName("release")
51     }
52 }
53 }

```

Fuente: Autor

Generación de instalable

Figura 62. Comando para generar el .apk

```

o\proyecto_final> flutter build apk --release

```

Fuente: Autor

El resultado, en build/app/outputs/flutter-apk/app-release.apk, es un instalador directo que no requiere la Google Play Store, adecuado para distribución controlada sin pasar por un proceso de publicación formal.

Permiso de red, un ajuste necesario tras la primera compilación

Durante las pruebas del instalador generado se detectó que la aplicación no lograba conectarse al backend, mientras que el mismo código funcionaba sin problema en modo de depuración. La causa, Flutter añade automáticamente el permiso de Internet a un manifiesto auxiliar de depuración, invisible durante el desarrollo, que no existe en una compilación de producción, esta debe declarar el permiso explícitamente en el manifiesto principal

Figura 63. Permiso de internet para su versión Android

```
1 <manifest xmlns:android="http://schemas.android.com/apk/res/android">
2
3 <uses-permission android:name="android.permission.INTERNET"/>
4
5 <application
```

Fuente: Autor

4.6.2. Despliegue de la aplicación Web

Generación del build estático

Flutter Web compila a un conjunto de archivos estáticos (HTML, JavaScript y recursos) mediante:

Figura 64. Comando para generar su versión web

```
\proyecto_final> flutter build web --release
```

Fuente: Autor

Publicación en Vercel

El resultado se publicó en Vercel, una plataforma de hosting de sitios estáticos con HTTPS automático y sin costo para este nivel de uso. El despliegue se realiza con su interfaz de línea de comandos:

Figura 65. Publicación en Vercel

```
\proyecto_final> vercel deploy build\web --prod
```

Fuente: Autor

Enrutamiento de una aplicación de una sola página (SPA)

Dado que la aplicación usa go_router con URLs reales fue necesario instruir a Vercel para que cualquier ruta no reconocida como archivo físico sirva igualmente el punto de entrada de la aplicación, y que el enrutador de Flutter la resuelva del lado del cliente:

Figura 66. Enrutamiento a una sola página

```
▲ vercel.json > ...
1  {
2    "rewrites": [
3      { "source": "/(.*)", "destination": "/index.html" }
4    ]
5  }
```

Fuente: Autor

Sin esta regla, refrescar la página en una ruta interna o compartir ese enlace directamente resultaría en un error 404, porque el servidor intentaría ubicar un archivo físico con ese nombre en lugar de delegar la navegación a la aplicación.

4.7. Pruebas y Validación

4.7.1. Pruebas Funcionales

Con el fin de validar el funcionamiento del sistema, se realizaron pruebas funcionales las cuales se llevaron a cabo en dispositivos Android y entornos web.

Tabla 24.

Matriz de validación para pruebas funcionales

ID	Funcionalidad	Prueba	Resultado esperado	Resultado
PF-01	Registro	Registrar usuario con datos válidos	Usuario creado correctamente, con rol "ciudadano" asignado por defecto	Correcto
PF-02	Registro	Registrar usuario con campos obligatorios vacíos	El sistema muestra las validaciones correspondientes y no envía la solicitud	Correcto
PF-03	Registro	Registrar usuario con un correo ya existente	El sistema muestra un mensaje de error que indica que la cuenta ya existe	Correcto
PF-04	Registro	Registrar usuario con contraseñas que no coinciden	El sistema muestra el error de confirmación antes de enviar	Correcto
PF-05	Inicio de sesión	Ingresar credenciales correctas	Se permite el acceso y se redirige al Dashboard	Correcto
PF-06	Inicio de sesión	Ingresar credenciales incorrectas	El sistema muestra "Correo o contraseña incorrectos"	Correcto
PF-07	Inicio de sesión	Iniciar sesión con una cuenta	El sistema muestra "Tu cuenta ha sido desactivada..." y no otorga acceso	Correcto

		desactivada por un administrador		
PF-08	Inicio de sesión	Cerrar y reabrir la app tras haber iniciado sesión	La sesión persiste; no se solicita volver a iniciar sesión	Correcto
PF-09	Recuperación de contraseña	Solicitar enlace de recuperación con un correo registrado	Llega el correo con el enlace de recuperación	Correcto
PF-10	Recuperación de contraseña	Abrir el enlace recibido	Se abre el formulario de nueva contraseña, no la pantalla de inicio	Correcto
PF-11	Recuperación de contraseña	Establecer una nueva contraseña	La contraseña se actualiza; se puede iniciar sesión con la nueva	Correcto
PF-12	Cierre de sesión	Cerrar sesión desde cualquier pantalla	Se invalida la sesión y se redirige a la pantalla de inicio	Correcto
PF-13	Protección de rutas	Un ciudadano intenta acceder manualmente a /admin	El sistema redirige automáticamente a /home	Correcto
PF-14	Protección de rutas	Un usuario sin sesión intenta acceder a una ruta protegida	El sistema redirige a /login	Correcto
PF-15	Perfil	Consultar el correo electrónico propio	Se muestra el correo, marcado como no editable	Correcto
PF-16	Perfil	Editar el alias público	El alias se guarda y se refleja como autor en las propuestas propias	Correcto
PF-17	Perfil	Dejar el alias vacío	Las propuestas propias muestran el nombre completo como respaldo	Correcto
PF-18	Perfil	Editar el número de teléfono	El teléfono se guarda correctamente	Correcto
PF-19	Perfil	Cambiar la foto de avatar	El avatar se actualiza en el perfil y en el resto de la app	Correcto
PF-20	Perfil	Tocar el avatar propio	Se abre la imagen ampliada con zoom, sin activar el cambio de foto	Correcto
PF-21	Propuestas (ciudadano)	Crear una propuesta con datos válidos	La propuesta queda registrada en estado "pendiente"	Correcto
PF-22	Propuestas (ciudadano)	Crear una propuesta con campos obligatorios vacíos	El sistema muestra las validaciones correspondientes	Correcto
PF-23	Propuestas (ciudadano)	Adjuntar imágenes al crear una propuesta	Las imágenes se suben y son visibles en el detalle	Correcto
PF-24	Propuestas (ciudadano)	Editar una propuesta propia mientras está "pendiente"	Los cambios se guardan correctamente	Correcto
PF-25	Propuestas (ciudadano)	Intentar editar una propuesta propia ya aprobada o rechazada	El sistema no permite la edición	Correcto

PF -26	Propuestas (ciudadano)	Consultar "Mis propuestas"	Se listan todas las propuestas propias, sin importar su estado	Correcto
PF -27	Propuestas (ciudadano)	Consultar la pestaña "Comunidad"	Solo se listan propuestas de estado "aprobada", de cualquier autor	Correcto
PF -28	Propuestas (ciudadano)	Intentar consultar el detalle de una propuesta pendiente de otro usuario (vía URL directa)	El sistema no muestra la información	Correcto
PF -29	Propuestas (ciudadano)	Tocar una imagen de una propuesta	Se abre la imagen ampliada con zoom	Correcto
PF -30	Propuestas (administrador)	Consultar la cola de propuestas pendientes/en revisión	Se listan correctamente, con el autor visible	Correcto
PF -31	Propuestas (administrador)	Marcar una propuesta como "en revisión"	El estado cambia y el ciudadano pierde la posibilidad de editarla	Correcto
PF -32	Propuestas (administrador)	Aprobar una propuesta	El estado cambia a "aprobada"; el autor recibe una notificación	Correcto
PF -33	Propuestas (administrador)	Rechazar una propuesta sin indicar motivo	El sistema no permite continuar sin el motivo	Correcto
PF -34	Propuestas (administrador)	Rechazar una propuesta con motivo	El estado cambia a "rechazada"; la notificación incluye el motivo	Correcto
PF -35	Propuestas (administrador)	Consultar la auditoría tras aprobar/rechazar	Se registran las acciones aprobar_propuesta/rechazar_propue sta	Correcto
PF -36	Categorías (administrador)	Crear una nueva categoría	La categoría queda disponible en el formulario de nueva propuesta	Correcto
PF -37	Categorías (administrador)	Editar una categoría existente	Los cambios se reflejan correctamente	Correcto
PF -38	Categorías (administrador)	Desactivar una categoría	Deja de aparecer en el formulario de nueva propuesta, sin afectar propuestas existentes	Correcto
PF -39	Votaciones (administrador)	Crear una votación de selección múltiple con dos o más propuestas aprobadas	La votación se crea en estado "programada"	Correcto
PF -40	Votaciones (administrador)	Intentar crear una votación de selección múltiple con menos de dos propuestas	El sistema no lo permite	Correcto
PF -41	Votaciones (administrador)	Crear una votación de tipo Sí/No con una propuesta	La votación se crea con las dos opciones generadas automáticamente	Correcto

PF -42	Votaciones (administrador)	Intentar crear una votación Sí/No con más de una propuesta	El sistema no lo permite	Correcto
PF -43	Votaciones (administrador)	Activar una votación manualmente	El estado cambia a "activa"; los ciudadanos reciben notificación	Correcto
PF -44	Votaciones (administrador)	Finalizar una votación manualmente	El estado cambia a "finalizada"; los resultados quedan visibles	Correcto
PF -45	Votaciones (administrador)	Cancelar una votación "programada"	El estado cambia a "cancelada"	Correcto
PF -46	Votaciones (automatización)	Programar una votación con fecha de inicio próxima, sin activarla manualmente	La votación se activa sola al llegar la hora programada	Correcto
PF -47	Votaciones (automatización)	Dejar que una votación activa llegue a su fecha de fin	La votación se finaliza sola, sin intervención manual	Correcto
PF -48	Votaciones (automatización)	Programar una votación cuya ventana completa ya venció antes de revisarla	Pasa directamente a "finalizada", sin quedar atascada en "programada"	Correcto
PF -49	Votaciones (ciudadano)	Votar en una votación activa	El voto se registra; se muestra confirmación	Correcto
PF -50	Votaciones (ciudadano)	Intentar votar una segunda vez en la misma votación	El sistema rechaza el segundo intento	Correcto
PF -51	Votaciones (ciudadano)	Intentar votar en una votación "programada" o "finalizada"	El sistema no permite emitir el voto	Correcto
PF -52	Votaciones (ciudadano)	Consultar resultados después de haber votado	Se muestran los conteos agregados por opción	Correcto
PF -53	Votaciones (ciudadano)	Consultar resultados de una votación finalizada sin haber participado	Los resultados son visibles igualmente	Correcto
PF -54	Votaciones (seguridad)	Consultar la tabla votos mediante la API con una cuenta de ciudadano y con una de administrador	No se puede leer ninguna fila; los roles de la aplicación no tienen forma de relacionar identidad con elección	Correcto

PF -55	Proyectos (administrador)	Convertir una propuesta aprobada en proyecto	El proyecto se crea; la propuesta pasa a "convertida_proyecto"	Correcto
PF -56	Proyectos (administrador)	Intentar convertir una propuesta no aprobada	El sistema no lo permite	Correcto
PF -57	Proyectos (administrador)	Intentar convertir la misma propuesta una segunda vez	El sistema no lo permite (ya no aparece disponible)	Correcto
PF -58	Proyectos (administrador)	Registrar un avance con un nuevo porcentaje	El porcentaje del proyecto se sincroniza automáticamente	Correcto
PF -59	Proyectos (administrador)	Intentar registrar un porcentaje menor al avance actual	El sistema no lo permite	Correcto
PF -60	Proyectos (administrador)	Registrar un avance que llega al 100%	El proyecto pasa automáticamente a estado "finalizado"	Correcto
PF -61	Proyectos (administrador)	Intentar registrar un nuevo avance en un proyecto ya finalizado	El sistema no lo permite	Correcto
PF -62	Proyectos (administrador)	Adjuntar evidencias a un avance	Las evidencias se suben y son visibles con zoom	Correcto
PF -63	Proyectos (administrador)	Cambiar el estado de un proyecto (suspender/reanuda r)	El estado se actualiza correctamente	Correcto
PF -64	Proyectos (ciudadano)	Consultar el listado general de proyectos	Se listan todos los proyectos, con su estado y avance	Correcto
PF -65	Proyectos (ciudadano)	Consultar el detalle e historial de avances de un proyecto	Se muestra el historial completo, ordenado del más reciente al más antiguo	Correcto
PF -66	Notificaciones	Aprobar o rechazar una propuesta	El autor recibe la notificación interna correspondiente	Correcto
PF -67	Notificaciones	Activar una votación	Todos los ciudadanos activos reciben la notificación	Correcto
PF -68	Notificaciones	Finalizar una votación	Solo quienes votaron reciben la notificación	Correcto
PF -69	Notificaciones	Registrar un avance de proyecto	El autor de la propuesta original recibe la notificación	Correcto
PF -70	Notificaciones	Tocar una notificación no leída	Se marca como leída y navega a la sección correspondiente	Correcto
PF -71	Notificaciones	Usar "Marcar todas como leídas"	El contador de notificaciones no leídas se limpia	Correcto
PF -72	Administración (usuarios)	Cambiar el rol de otro usuario	El rol se actualiza correctamente	Correcto

PF -73	Administración (usuarios)	Activar/desactivar la cuenta de otro usuario	El estado se actualiza correctamente	Correcto
PF -74	Administración (usuarios)	Intentar modificar la propia cuenta desde el panel de administración	El sistema no lo permite	Correcto
PF -75	Administración (usuarios)	Iniciar sesión con una cuenta recién desactivada	El acceso es rechazado	Correcto
PF -76	Administración (usuarios)	Con una sesión ya abierta antes de la desactivación, intentar votar o crear una propuesta	La operación es rechazada por el sistema	Correcto
PF -77	Métricas	Consultar el panel de métricas	Se muestran los diez indicadores esperados con valores coherentes	Correcto
PF -78	Métricas	Consultar las gráficas de propuestas, votaciones y proyectos por estado	Los totales coinciden con las tarjetas numéricas	Correcto
PF -79	Métricas	Usar "Copiar reporte"	El resumen de texto se copia correctamente al portapapeles	Correcto
PF -80	Auditoría	Consultar el panel de auditoría como administrador	Se listan cronológicamente las acciones administrativas registradas	Correcto
PF -81	Auditoría	Intentar acceder a la auditoría con una cuenta ciudadana	El acceso es rechazado	Correcto
PF -82	Seguridad (RLS)	Intentar leer, mediante una consulta directa a la API, las propuestas pendientes de otro usuario	La consulta no devuelve resultados	Correcto
PF -83	Seguridad (RLS)	Intentar modificar el propio rol_id mediante una actualización directa a la tabla usuarios	La operación es rechazada	Correcto
PF -84	Multiplataforma	Usar la aplicación instalada en un dispositivo Android	Todas las funcionalidades operan correctamente	Correcto
PF -85	Multiplataforma	Usar la aplicación desde un navegador (Web)	Todas las funcionalidades operan correctamente	Correcto

PF-86	Multiplataforma	Reducir el ancho de la ventana en Web por debajo de la resolución de escritorio	La barra lateral desaparece y cada pantalla usa su propia navegación	Correcto ²
-------	-----------------	---	--	-----------------------

Nota: Tabla elaborada por el autor a partir las pruebas funcionales realizadas

El objetivo específico 5 plantea evaluar el desempeño de la plataforma y medir su impacto en la participación y la transparencia. En este trabajo, la medición se limitó a lo observable durante una prueba con usuarios: el periodo de uso fue corto y no se contó con una medición previa a la puesta en marcha del sistema. Las variables, los indicadores y su forma de cálculo se definieron en la Tabla 1. La siguiente tabla indica qué se midió en cada dimensión, con qué fuente y qué permite afirmar.

Tabla 25.

Alcance de la medición del objetivo 5

Dimensión	Qué se midió	Fuente	Qué permite afirmar
Participación	Actividad registrada por el sistema durante la prueba y disposición a participar declarada en el diagnóstico	Panel de métricas; encuesta diagnóstica	Describe la participación del grupo de prueba; no la compara con un periodo anterior
Transparencia	Que el estado de las propuestas, los resultados de las votaciones y los avances de los proyectos sean consultables, y que las acciones administrativas queden registradas	PF-27, PF-35, PF-52, PF-53, PF-64, PF-65, PF-80	Verifica que la información está disponible; no mide cómo la percibe el usuario
Eficiencia	Activación y cierre de votaciones y cálculo del avance de proyectos sin intervención manual	PF-46 a PF-48, PF-58, PF-60	Verifica la automatización; no se midieron tiempos
Aceptación y usabilidad	Cuestionario de cuatro preguntas cerradas	10 participantes	Describe la percepción de este grupo

² Nota: Las pruebas funcionales fueron ejecutadas el 28 de septiembre del 2026, alternando entre su versión móvil y su versión web de escritorio (Chrome).

Nota: Tabla elaborada por el autor

4.7.2. Pruebas de Rendimiento y Concurrencia

Se obtuvo evidencia medible de dos requisitos no funcionales del sistema: el tiempo de respuesta de sus operaciones principales y su comportamiento cuando varios usuarios operan al mismo tiempo, con atención especial a la integridad del proceso de votación bajo carga simultánea.

Como criterio de referencia se adoptó el límite de 1 segundo descrito por Nielsen (1993), por debajo del cual el usuario percibe que el sistema responde sin interrumpir su flujo de pensamiento.

Se desarrolló un script (`pruebas_rendimiento.mjs`) automatizado en Node.js que utiliza el cliente oficial `@supabase/supabase-js`, el mismo mecanismo de acceso que emplea la aplicación. Las pruebas se ejecutaron contra el entorno de producción de Supabase (plan gratuito) el 10 de octubre de 2026.

El script creó 20 usuarios de prueba con el correo ya confirmado, mediante la API de administración, y utilizó una votación dedicada exclusivamente a pruebas. Al finalizar, elimina los usuarios, propuestas y votos que generó.

Cada operación se cronometra con un reloj de alta resolución, desde el envío de la solicitud hasta recibir la respuesta completa. El tiempo incluye, por tanto, la latencia de red entre el equipo de pruebas y el servidor. Para cada operación se calculan la media, la mediana, el percentil 95 (método del rango más cercano), el mínimo y el máximo, además del número de errores.

Tabla 26.
Escenarios de la prueba

Escenario	Descripción
A. Un usuario	Inicio de sesión de los 20 usuarios en secuencia, y 30 repeticiones secuenciales de cada operación: listar propuestas de la comunidad, listar proyectos, consultar resultados de una votación y consultar métricas (administrador). La creación de propuesta se repitió 15 veces.
B1. Lectura concurrente	5, 10 y 20 usuarios consultando propuestas aprobadas de forma simultánea, en tres rondas por escenario.
B2. Votación simultánea	20 usuarios emiten su voto al mismo tiempo sobre la misma votación. Cinco de ellos lanzan además un segundo intento simultáneo, para un total de 25 llamadas a la función emitir_voto.

Nota: Tabla elaborada por el autor

Tras la ráfaga de votos se verificó la integridad contando, con el rol de servicio, las filas nuevas en la tabla votos y los registros únicos en voto_registros.

Resultados

Tabla 27.
Tiempos de respuesta con un solo usuario

Operación	n	Errores	Media (ms)	Mediana (ms)	P95 (ms)	Mín (ms)	Máx (ms)
Inicio de sesión	20	0	230	225	260	204	276
Listar propuestas de la comunidad	30	0	151	142	195	131	297
Listar proyectos	30	0	150	143	182	133	228
Consultar resultados de una votación	30	0	152	151	170	132	177
Crear propuesta	15	0	157	148	263	132	263
Consultar métricas (administrador)	30	0	151	144	191	130	275

Nota: Tabla elaborada por el autor a partir de los resultados

Con muestras pequeñas, el percentil 95 coincide o se aproxima al valor máximo, por lo que ambos deben leerse juntos.

Tabla 28.
Concurrencia: lectura simultánea de propuestas

Escenario	n	Errores	Media (ms)	Mediana (ms)	P95 (ms)	Mín (ms)	Máx (ms)	Duración total (ms)
-----------	---	---------	------------	--------------	----------	----------	----------	---------------------

5 usuarios simultáneos (3 rondas)	15	0	225	172	493	129	493	990
10 usuarios simultáneos (3 rondas)	30	0	286	228	695	129	697	1810
20 usuarios simultáneos (3 rondas)	60	0	223	179	480	131	490	1019

Nota: Tabla elaborada por el autor a partir de los resultados

Tabla 29.

Concurrencia: votación simultánea e intento de voto doble

Indicador	Esperado	Obtenido
Llamadas simultáneas a emitir voto	25	25
Votos aceptados	20	20
Intentos duplicados rechazados	5	5
Filas nuevas en la tabla votos	20	20
Registros de participación únicos	20	20

Nota: Tabla elaborada por el autor a partir de los resultados

La ráfaga completa se resolvió en 453 ms. La latencia de los 20 votos aceptados fue de 313 ms de media, 292 ms de mediana, 444 ms en el percentil 95, con un mínimo de 219 ms y un máximo de 453 ms. Los cinco intentos rechazados devolvieron el mismo mensaje: violación de la restricción única voto_registros_usuario_id_votacion_id_key.

Análisis

Tiempos con un solo usuario. Las seis operaciones medidas respondieron sin errores y con medianas entre 142 y 225 ms. Las operaciones de datos (lecturas, creación de propuesta y consulta de métricas) tuvieron medias casi idénticas, entre 150 y 157 ms, y su mínimo osciló entre 130 y 133 ms. Esa uniformidad sugiere que, con el volumen de datos de la prueba, el tiempo está dominado por la latencia de red más que por el procesamiento en la base de datos, aunque esta medición no permite confirmarlo. El resultado es relevante para el módulo de métricas: la

consulta que agrega información de varias tablas en una sola llamada (mediana de 144 ms) respondió tan rápido como una lectura simple (142 ms). El inicio de sesión fue la operación más lenta (mediana de 225 ms), algo esperable en una operación que verifica credenciales, aunque tampoco puede aislarse la causa a partir de estos datos.

Lectura concurrente. Las 105 solicitudes simultáneas se completaron sin errores. Con respecto a las lecturas individuales (medianas de 142 a 151 ms), bajo concurrencia las medianas subieron a un rango de 172 a 228 ms, y el máximo observado fue de 697 ms. No se observó una degradación progresiva con el número de usuarios: el escenario de 10 usuarios resultó más lento que el de 20 (media de 286 frente a 223 ms). Con tres rondas por escenario y una sola ejecución, esa diferencia no puede atribuirse a la carga, y es consistente con la variabilidad natural de la red.

Votación simultánea. Los resultados confirman el comportamiento descrito en la documentación del modelo de datos. De las 25 llamadas, se aceptaron exactamente 20 (un voto por usuario) y se rechazaron exactamente los 5 segundos intentos. La tabla votos recibió 20 filas y voto_registros 20 participaciones únicas, sin votos duplicados ni registros huérfanos. El mensaje de rechazo identifica la restricción UNIQUE como el mecanismo que detuvo el voto doble, es decir, el control opera a nivel de base de datos incluso cuando ambos intentos llegan al mismo tiempo.

La latencia de un voto (mediana de 292 ms) fue aproximadamente el doble que la de una lectura. La función emitir_voto bloquea la fila de la votación mientras valida y registra el voto, de modo que los votos simultáneos sobre una misma votación se procesan en cola; los datos son consistentes con ese diseño, aunque la

prueba no mide el bloqueo directamente. El bloqueo es por votación, no global, por lo que votaciones distintas no se bloquean entre sí. El comportamiento con un número de votantes simultáneos mayor a 20 sobre una misma votación no fue evaluado.

Limitaciones

- Los resultados corresponden a una sola ejecución, realizada desde un único equipo y una única conexión de red. Incluyen la latencia de esa conexión y no representan a usuarios en otras redes o en dispositivos móviles.
- El máximo de 20 usuarios simultáneos estuvo acotado por el límite de ráfagas de inicio de sesión de Supabase Auth. Por eso estas pruebas verifican un comportamiento correcto bajo concurrencia moderada, pero no constituyen una prueba de estrés ni identifican el punto de saturación del sistema.
- Las pruebas se realizaron sobre el plan gratuito de Supabase, con recursos limitados; los resultados podrían variar en otros planes.
- Las mediciones corresponden al volumen de datos existente al momento de la prueba. No se evaluó el comportamiento con volúmenes mayores.
- Se midió el tiempo de respuesta de la API, no el tiempo de renderizado de la interfaz ni la carga de imágenes; la creación de propuesta se midió sin adjuntar archivos.
- Solo se evaluaron lecturas concurrentes y votación simultánea. No se probaron otras escrituras simultáneas, como la creación de propuestas por varios usuarios a la vez.

Bitácora de defectos

Tabla 30.
Bitácora de defectos

Fecha	Módulo	Hallazgo	Corrección (commit)	Verificación
-------	--------	----------	------------------------	--------------

31 ago–1 sep	Votaciones	La primera versión no soportaba votaciones	Se añadió el tipo Sí/No (ec6fbdb, 9dfd08c)	PF-41, PF-42
3-sep	Administración	Manejo incorrecto de usuarios inactivos	Corrección en el panel administrativo (a317286)	PF-07, PF-75
6 y 8 sep	Propuestas, perfiles, notificaciones	Errores detectados tras implementar los módulos	97a0698, 1c7984c	Pruebas de los tres módulos
19–21 sep	Interfaz	Rediseño y corrección de errores	c9eba34, 80d49d6	PF-84 a PF-86
22 y 27 sep	Autenticación	La recuperación de contraseña no funcionó como se esperaba, y se corrigió en dos ocasiones	Dos correcciones del flujo	PF-09 a PF-11
22–27 sep	Android	Falta del permiso de acceso a internet en el empaquetado	Permiso agregado	PF-84
25-sep	Base de datos	Ajustes del esquema surgidos durante el desarrollo	Actualización de la base	Pruebas de regresión†
Sin fecha registrada	Votaciones automáticas	Votaciones con ventana vencida sin haberse activado quedaban atascadas en "programada"	Regla adicional en la tarea programada que las finaliza	PF-48
Fase de diseño	Privacidad del voto	Una sola tabla de votos permitía asociar usuario y opción	Separación en voto_registros y votos	PF-54
Revisión final	Privacidad del voto	votos.fecha y voto_registros.fecha_voto toman la misma marca de tiempo	Sin corrección en esta versión; declarada como limitación y recomendación	

Nota: Tabla elaborada por el autor a partir de los errores durante el desarrollo

4.7.3. Pruebas de Usabilidad

El protocolo de tareas se aplicó a 4 participantes (P01 a P04) y el cuestionario SUS fue respondido por 10 (P01 a P10). Los participantes se identificaron con códigos para mantener el anonimato.

Protocolo de tareas

Cada participante realizó siete tareas representativas de los módulos del sistema. El facilitador leyó el escenario sin nombrar botones ni pantallas, y un observador registró los resultados. El tiempo máximo por tarea es el límite tras el cual se consideraría fracaso.

Tabla 31.

Protocolo de tareas

ID	Escenario	Criterio de éxito	Tiempo máx.
T1	Crear una cuenta en Participa y entrar a la aplicación	Llega al panel principal tras aceptar la política de datos	5 min
T2	Registrar una propuesta para arreglar un parque del barrio, adjuntando una foto	La propuesta aparece en "Mis propuestas" como pendiente, con imagen	6 min
T3	Averiguar el estado de la propuesta recién creada	Indica correctamente el estado	2 min
T4	Buscar una propuesta aprobada por la comunidad e indicar quién la publicó	Indica el autor de una propuesta aprobada	3 min
T5	Emitir el voto en una votación abierta	Aparece la confirmación del voto	3 min
T6	Averiguar el porcentaje de avance de un proyecto y su última actualización	Indica el porcentaje y la última actualización correctos	4 min
T7	Configurar un alias para que aparezca como autor de las propuestas	El alias queda guardado en el perfil	3 min

Nota: Tabla elaborada por el autor

Métricas registradas por tarea:

- Resultado: éxito sin ayuda (E), éxito asistido (EA, cuando el facilitador tuvo que intervenir) o fracaso (F, por abandono o tiempo máximo agotado).
- Tiempo: segundos desde que termina la lectura del escenario hasta que se cumple el criterio de éxito.

- Errores: acciones que no acercan a la meta y deben corregirse (pantalla equivocada, formulario rechazado por validación, opción incorrecta). Pedir ayuda no se cuenta como error y se refleja en el resultado EA.

Cuestionario SUS

Se aplicó la escala SUS (Brooke, 1996) al terminar las tareas: diez afirmaciones en escala de 1 (totalmente en desacuerdo) a 5 (totalmente de acuerdo). La redacción en español es una adaptación del instrumento original, con “aplicación” en lugar de “sistema”. No se comparó con la versión validada en español (Sevilla-Gonzalez et al., 2020).

Cálculo del puntaje. En los ítems impares se resta 1 a la respuesta; en los pares se resta la respuesta a 5. Los diez aportes se suman y el total se multiplica por 2.5, con lo que se obtiene un puntaje de 0 a 100 (que no es un porcentaje). El puntaje del grupo es el promedio de los puntajes individuales, reportado con su desviación estándar muestral y un intervalo de confianza del 95% basado en la distribución t de Student.

Interpretación. Se utilizaron tres referencias de la literatura: el promedio de la industria, cercano a 68 (Sauro & Lewis, 2016); los rangos de aceptabilidad de Bangor et al. (2008), donde menos de 50 es no aceptable, de 50 a 70 es marginal y más de 70 es aceptable; y la escala de adjetivos de Bangor et al. (2009), asignando el adjetivo cuya media está más cerca del puntaje.

Resultados

Tabla 32.
Protocolo de tareas

Tarea	n	E	EA	F	Éxito sin ayuda	Tiempo mediano (s)	Tiempo medio (s)	Rango (s)	Errores
T1 Registro e inicio de sesión	4	3	1	0	75%	75.5	93.8	45 a 179	0
T2 Crear propuesta con imagen	4	3	1	0	75%	79.5	88.5	71 a 124	0
T3 Consultar estado de la propuesta	4	3	1	0	75%	12.0	13.0	5 a 23	0
T4 Encontrar propuesta aprobada y su autor	4	3	1	0	75%	11.0	13.5	6 a 26	0
T5 Votar	4	4	0	0	100%	11.5	15.3	4 a 34	0
T6 Consultar avance de un proyecto	4	2	2	0	50%	54.5	54.3	30 a 78	0
T7 Configurar alias	4	2	2	0	50%	18.5	29.8	9 a 73	0
Global	28	20	8	0	71%	28.0	44.0	4 a 179	0

Nota: Tabla elaborada por el autor

Tabla 33.
Resultados por participante

Participante	Tareas con asistencia	Tiempo total (s)	Puntaje SUS
P01	0 de 7	170	82.5
P02	6 de 7 (todas salvo T5)	520	67.5
P03	1 de 7 (T7)	302	70.0
P04	1 de 7 (T6)	240	67.5

Nota: Tabla elaborada por el autor

Escala SUS

Tabla 34.

Tabla de aceptación escala SUS

Participante	Puntaje	Rango de aceptabilidad
P01	82.5	Aceptable
P02	67.5	Marginal
P03	70.0	Marginal
P04	67.5	Marginal
P05	80.0	Aceptable
P06	85.0	Aceptable
P07	72.5	Aceptable
P08	77.5	Aceptable
P09	70.0	Marginal

Nota: Tabla elaborada por el autor

Tabla 35.

Resumen de los resultados

Estadístico	Valor
Participantes (n)	10
SUS medio	76.25
Desviación estándar (muestral)	7.93
Mediana	75.0
Mínimo / máximo	67.5 / 90.0
Intervalo de confianza del 95%	70.58 a 81.92
Adjetivo más cercano	Bueno
Aceptabilidad	Aceptable
Frente al promedio de referencia (68)	Por encima

Nota: Tabla elaborada por el autor

Tabla 36.

Respuestas por ítem

Ítem	Tema	Media respuesta de	Aporte medio (0 a 4)	Rango
1	Uso frecuente	4.3	3.3	3 a 5
2	Complejidad innecesaria	1.7	3.3	1 a 2
3	Facilidad de uso	2.7	1.7	2 a 3
4	Necesidad de apoyo técnico	1.7	3.3	1 a 2
5	Integración de funciones	4.4	3.4	4 a 5
6	Inconsistencia	1.5	3.5	1 a 2
7	Aprendizaje rápido	3.4	2.4	1 a 5
8	Incomodidad de uso	2.0	3.0	1 a 5
9	Confianza al usar	4.5	3.5	4 a 5
10	Necesidad de aprender antes	1.9	3.1	1 a 3

Nota: Tabla elaborada por el autor

Análisis

Los 28 intentos se completaron, sin ningún fracaso, y 20 de ellos (71%) sin ayuda del facilitador. Ningún intento se acercó al tiempo máximo: el mayor consumo fue el de T1, con 179 s de 300 s, y en el resto de tareas el peor tiempo estuvo por debajo del 45% del límite.

La función de votación, central en el sistema, fue la más sencilla: los cuatro participantes la completaron sin ayuda, con una mediana de 11.5 s. Las tareas de consulta T3 y T4 tuvieron medianas de 12 y 11 s, y en cada una un participante necesitó asistencia. Las tareas de formulario, registro (T1) y creación de propuesta (T2), fueron las más largas, con medianas de 75.5 y 79.5 s, lo esperable por la cantidad de datos que se ingresan, y en cada una un participante necesitó ayuda.

Las tareas con más asistencias fueron T6 y T7, con 2 de 4 participantes asistidos en cada una. T6 destaca además por su tiempo: con una mediana de 54.5 s, tardó entre 4 y 5 veces más que las otras tareas de consulta (T3, T4 y T5, con medianas de 11 a 12 s). Esto señala la consulta del avance de un proyecto y la configuración del alias como los puntos de la interfaz con mayor margen de mejora. La hoja de observaciones no incluye notas cualitativas, por lo que no queda documentado qué causó cada asistencia; por esa razón estos resultados no permiten afirmar el motivo de la dificultad.

Usabilidad percibida (SUS)

El puntaje SUS medio fue de 76.25, que corresponde al adjetivo “Bueno” (el más cercano entre “Bueno”, con 71.4, y “Excelente”, con 85.5) y se ubica en el rango de aceptabilidad “Aceptable”. El intervalo de confianza del 95% (70.58 a 81.92) no incluye el promedio de referencia de 68, de modo que el puntaje es consistente con

una usabilidad superior al promedio. Su límite inferior supera por muy poco el umbral de 70 que separa lo marginal de lo aceptable.

A nivel individual, 6 de los 10 participantes puntuaron por encima de 70 y 4 en el rango marginal (dos con 67.5 y dos con 70.0). Ninguno puntuó por debajo de 67.5.

La lectura por ítem tiene carácter exploratorio, porque el SUS está pensado como puntaje global y la muestra es pequeña. Los ítems mejor valorados fueron la confianza al usar la aplicación (ítem 9) y la ausencia de inconsistencias (ítem 6), con aportes medios de 3.5, seguidos de la integración de funciones (3.4). La menor valoración fue la de facilidad de uso (ítem 3), con un aporte medio de 1.7: ningún participante respondió más de 3 (neutral) a esa afirmación. El aprendizaje rápido (ítem 7) fue el segundo más bajo (2.4) y el más disperso, con respuestas de 1 a 5. Es decir, los participantes percibieron la aplicación como coherente, bien integrada y sin necesidad de apoyo técnico, pero no la calificaron de fácil de usar de forma explícita. Esto es consistente con las ocho asistencias observadas en las tareas.

Entre los cuatro participantes que hicieron ambas pruebas, quien no necesitó ayuda (P01) obtuvo el puntaje más alto (82.5) y quien más ayuda necesitó (P02) obtuvo uno de los más bajos (67.5). Es una observación descriptiva sobre cuatro casos, no una correlación.

Limitaciones

- Tamaño de la muestra. El protocolo de tareas se aplicó a 4 participantes, por debajo de los 5 que suelen recomendarse para identificar problemas de usabilidad (Nielsen & Landauer, 1993), y el SUS a 10. Los resultados son descriptivos: proporciones como 3 de 4

son muy imprecisas, y los 28 intentos provienen de solo 4 personas, por lo que no son observaciones independientes.

- Precisión del SUS. Con 10 participantes, el intervalo de confianza tiene un margen de error de ± 5.7 puntos.
- Muestra por conveniencia. Los participantes no representan a toda la población de Bucaramanga.
- Rol evaluado. Solo se evaluó la experiencia del ciudadano; las funciones del administrador no se sometieron a pruebas de usabilidad.
- Condiciones de la prueba. Las tareas se realizaron en sesiones guiadas, con datos de prueba preparados previamente, y los tiempos se cronometraron manualmente.
- Observaciones cualitativas. No se registraron notas sobre las causas de las asistencias, lo que limita la identificación de mejoras concretas.
- Instrumento. La redacción del SUS en español es una adaptación propia, no la versión validada.
- Alcance de la medición. El SUS mide usabilidad percibida. No mide el impacto en la participación y la transparencia, que el objetivo 5 también menciona y que requiere otra evidencia.

4.8. Guía de instalación para reproducción del sistema

4.8.1. Requisitos previos

- Flutter SDK (canal stable) y Git
- Android Studio, con el SDK de Android y un emulador o dispositivo físico
- Una cuenta de Supabase (nivel gratuito es suficiente)
- Node.js (únicamente si se desea desplegar la versión Web a Vercel)

4.8.2. Clonar y preparar el proyecto

```
git clone https://github.com/JonatahnGonzalezCaballero/proyectoFinal.git
cd participa
flutter pub get
```

4.8.3. Estructura del archivo .env

El proyecto no incluye credenciales en el código fuente. Crear un archivo .env en la raíz del proyecto (junto a pubspec.yaml) con esta estructura y sustituir cada valor por el correspondiente al proyecto de Supabase propio

```
SUPABASE_URL=
SUPABASE_ANON_KEY=
PASSWORD_RESET_REDIRECT_URL=
```

SUPABASE_URL y SUPABASE_ANON_KEY se obtienen en el dashboard de Supabase, en Settings → API, una vez creado el proyecto. PASSWORD_RESET_REDIRECT_URL debe apuntar a la URL donde se ejecutará la aplicación Web

Este archivo está excluido del control de versiones mediante .gitignore y nunca debe compartirse ni subirse a un repositorio público.

4.8.4. Creación del proyecto en Supabase y ejecución del script SQL

- Crear un proyecto nuevo en supabase.com/dashboard y anotar una contraseña segura para la base de datos.
- Ir a Database → Extensions, buscar pg_cron y habilitarla - este paso es manual y no puede hacerse por SQL; sin él, la última sección del script (automatización de votaciones) fallará.
- Ir a SQL Editor → New query, pegar el contenido completo del script de esquema consolidado (schema_final_participa.sql) y ejecutarlo.
- Verificar en Table Editor que las catorce tablas se crearon correctamente.

- e. En Authentication → Providers → Email, configurar según el entorno: desactivar “Confirm email” para agilizar pruebas de desarrollo, o mantenerlo activo para un entorno de producción real.

4.9. Restricciones y consideraciones técnicas

- Peso máximo de la app <70MB
- Funcionamiento 100% online
- Solo funcional su versión Android y versión Web

5. RESULTADOS

Los resultados se organizan según los cinco objetivos específicos del trabajo. La Tabla **37** resume el indicador, la evidencia y el grado de cumplimiento de cada uno. La evidencia de funcionamiento proviene de las pruebas funcionales de la Tabla **24** y de la encuesta aplicada a miembros de la comunidad del Mirador de Arenales.

Tabla 37.

Matriz de cumplimiento por objetivo específico

Objetivo específico	Indicador	Evidencia	Grado de cumplimiento
1. Analizar los requerimientos funcionales y no funcionales del sistema	Requisitos funcionales y no funcionales definidos y organizados	46 requisitos funcionales en 7 grupos; 28 no funcionales en 8 categorías; 17 historias de usuario en 6 épicas; diagrama de casos de uso (Figura 6)	Cumplido
2. Diseñar la arquitectura del sistema y el modelo de datos	Arquitectura y modelo de datos diseñados y documentados	Arquitectura lógica y física (Figuras 7 a 9); 13 componentes (Tabla 2); modelo relacional de 14 tablas (Tablas 7 a 20)	Cumplido en el diseño; la escalabilidad y la concurrencia no se probaron con pruebas de carga
3. Desarrollar el sistema multiplataforma con los módulos de propuestas, votación, seguimiento y métricas	Al menos cuatro módulos funcionales en web y móvil	PF-21 a PF-65, PF-77 a PF-79 y PF-84 a PF-86 (Tabla 38)	Cumplido para web y Android; sin versión iOS
4. Implementar mecanismos de autenticación, control de acceso y gestión de datos	Registro, inicio de sesión, roles y acceso a datos validados	PF-01 a PF-08, PF-12 a PF-14, PF-72 a PF-76, PF-82 y PF-83 (Tabla 38)	Cumplido
5. Evaluar el desempeño mediante pruebas funcionales y de usabilidad	Pruebas funcionales con resultado esperado; evaluación con al menos 10 usuarios y	86 de 86 pruebas funcionales (Tabla 38); encuesta con 10 participantes, P1 = 80 % (Tabla 39); alcance de la	Cumplido en funcionamiento y usabilidad; el impacto en la participación y la

Objetivo específico	Indicador	Evidencia	Grado de cumplimiento
	satisfacción general ≥ 80 %	medición (Tabla 25); Pruebas de Usabilidad (Tabla 32).	transparencia medida con una sola medición posterior.

Nota: Tabla elaborada por el autor a partir de los resultados de las pruebas funcionales y de la encuesta de evaluación

5.1. Objetivo específico 1: análisis de requisitos

Se levantaron y documentaron 46 requisitos funcionales, organizados en siete grupos: gestión de usuarios y autenticación, propuestas comunitarias, votaciones, proyectos, métricas y reportes, notificaciones y administración del sistema. También se definieron 28 requisitos no funcionales en ocho categorías: usabilidad, compatibilidad y multiplataforma, seguridad, rendimiento, disponibilidad y confiabilidad, mantenibilidad, escalabilidad, y privacidad y tratamiento de datos.

Los requisitos se tradujeron en 17 historias de usuario (HU-1 a HU-17), agrupadas en seis épicas. El diagrama de casos de uso (Figura **6**) delimitó las funciones del ciudadano y del administrador. Estos productos se describen en la sección 4.1 y fijaron el alcance de la solución.

5.2. Objetivo específico 2: diseño de la arquitectura y del modelo de datos

La arquitectura diseñada sigue el modelo cliente-servidor y se organiza en tres capas lógicas: presentación (aplicación Flutter para web y móvil, con una sola base de código), aplicación (lógica de los módulos, validaciones y control de permisos) y datos (Supabase, con los servicios Auth, PostgreSQL, Storage y API). En la arquitectura física, un dispositivo Android y un navegador web se conectan mediante HTTPS a Supabase, sin servidor intermedio propio; la aplicación web se aloja en Vercel (Figuras **7** a **9**). El diseño identificó 13 componentes con su responsabilidad (Tabla **2**).

Modelo de datos

Se implementó en PostgreSQL, sobre Supabase, un modelo relacional de 14 tablas: roles, usuarios, categorías, propuestas, archivos de propuestas, votaciones, opciones de votación, registros de participación, votos, proyectos, actualizaciones de proyecto, archivos de proyecto, notificaciones y auditoría. Cada tabla se documentó con sus columnas, tipos y restricciones. El modelo incluye, además:

- Tres funciones de apoyo y tres disparadores de integridad.
- Dos vistas: resultados de votación y perfiles públicos.
- Tres buckets de almacenamiento, cada uno con políticas de acceso propias.
- Una tarea programada que se ejecuta cada minuto.

La separación entre la tabla de registros de participación y la tabla de votos, junto con la ausencia de políticas de lectura sobre esta última, hizo posible el voto único sin que los roles de la aplicación puedan identificar la elección, lo que se comprobó en PF-54.

5.3. Objetivo específico 3: desarrollo del sistema multiplataforma

Plataforma funcional en entorno web y móvil

Se obtuvo una aplicación desarrollada en Flutter, con Supabase como servicio de backend, que opera desde un navegador web y desde un dispositivo Android a partir de una misma base de código. Las pruebas PF-84 y PF-85 verificaron que todas las funcionalidades operan en ambos entornos, y PF-86 comprobó que, al reducir el ancho de la ventana web, la barra lateral desaparece y cada pantalla pasa a su propia navegación. La versión para iOS no se construyó. La aplicación requiere conexión permanente a internet y su peso es menor a 70 MB.

Módulos funcionales

Se implementaron los cuatro módulos exigidos (propuestas, votación digital, seguimiento de proyectos y métricas de participación), además de los módulos de notificaciones, administración de usuarios y auditoría. Se ejecutaron 86 pruebas funcionales, de las cuales 86 dieron el resultado esperado. La siguiente tabla resume los resultados por área.

Tabla 38.

Resultados de las pruebas funcionales por área

Área	Pruebas	Correctas	Incorrectas
Autenticación y protección de rutas (PF-01 a PF-14)	14	14	0
Perfil (PF-15 a PF-20)	6	6	0
Propuestas y categorías (PF-21 a PF-38)	18	18	0
Votaciones (PF-39 a PF-54)	16	16	0
Proyectos (PF-55 a PF-65)	11	11	0
Notificaciones (PF-66 a PF-71)	6	6	0
Administración de usuarios (PF-72 a PF-76)	5	5	0
Métricas (PF-77 a PF-79)	3	3	0
Auditoría (PF-80 a PF-81)	2	2	0
Seguridad de acceso a datos (PF-82 a PF-83)	2	2	0
Multiplataforma (PF-84 a PF-86)	3	3	0
Total	86	86	0

Nota: Tabla elaborada por el autor a partir de la Tabla 24

En el módulo de propuestas, el ciudadano pudo crear propuestas con imágenes adjuntas y editarlas solo mientras estaban en estado pendiente. El administrador pudo aprobarlas o rechazarlas, y el sistema exigió un motivo para

rechazar (PF-33). Cada decisión generó una notificación al autor y un registro de auditoría (PF-32, PF-34 y PF-35).

En el módulo de votaciones, el sistema aceptó un solo voto por ciudadano en cada votación (PF-50) y rechazó votos en votaciones programadas o finalizadas (PF-51). Las tres reglas de la tarea programada se cumplieron: activación automática al llegar la fecha de inicio, cierre automático al llegar la fecha de fin, y paso directo a finalizada de las votaciones cuya ventana ya había vencido (PF-46 a PF-48). Al consultar la tabla de votos a través de la API con una cuenta de ciudadano y con una de administrador, no se obtuvo ninguna fila.

En el módulo de proyectos, el administrador pudo convertir propuestas aprobadas en proyectos, y el sistema impidió convertir una propuesta no aprobada o convertir dos veces la misma (PF-55 a PF-57). El porcentaje de avance se sincronizó con el historial de actualizaciones. No permitió registrar un porcentaje inferior al actual (PF-59), y al llegar a 100 % el proyecto pasó a finalizado (PF-60).

En el módulo de métricas, el panel mostró los diez indicadores previstos, y los totales de las gráficas coincidieron con los de las tarjetas numéricas (PF-77 y PF-78).

5.4. Objetivo específico 4: autenticación y control de acceso

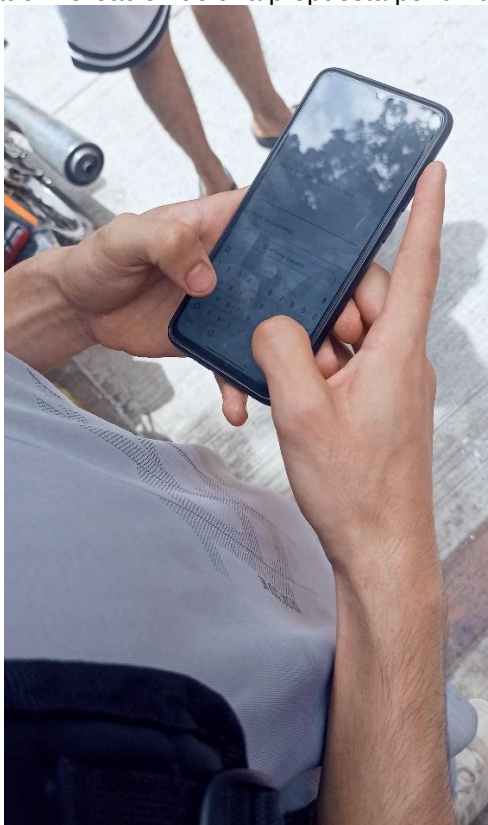
El registro, el inicio de sesión, el cierre de sesión y la persistencia de la sesión funcionaron según lo esperado. Todo usuario nuevo recibió el rol de ciudadano por defecto. Un ciudadano que intentó acceder a la ruta de administración fue redirigido a la pantalla principal, y un usuario sin sesión fue redirigido al inicio de sesión.

En las pruebas de seguridad, una consulta directa a la API no devolvió las propuestas pendientes de otro usuario, y la modificación del propio rol mediante una actualización directa fue rechazada. Una cuenta desactivada no pudo iniciar sesión, y una sesión abierta antes de la desactivación no pudo votar ni crear propuestas.

5.5. Objetivo específico 5: evaluación con usuarios finales

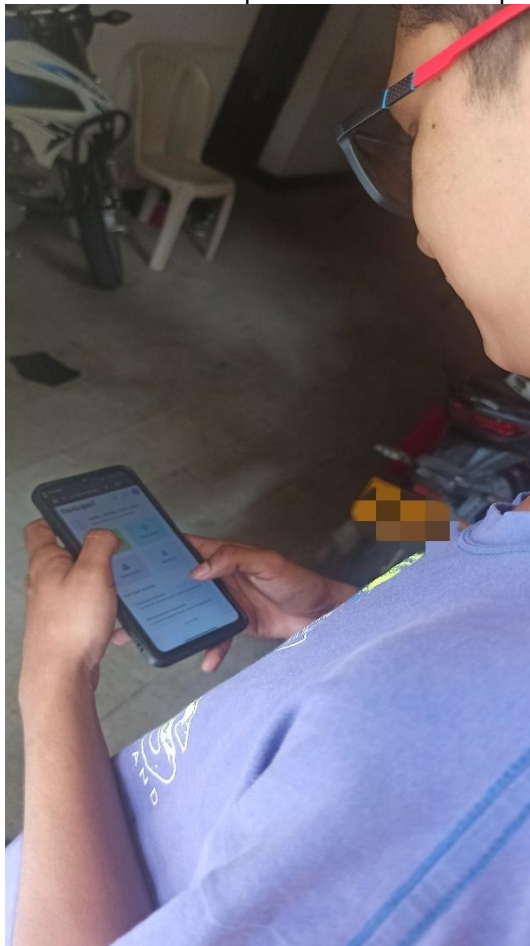
La evaluación se hizo con 10 miembros de la comunidad del Mirador de Arenales que usaron la aplicación. Se muestra a algunos usuarios durante las pruebas.

Figura 67. Creación de una propuesta por un usuario



Fuente: Autor

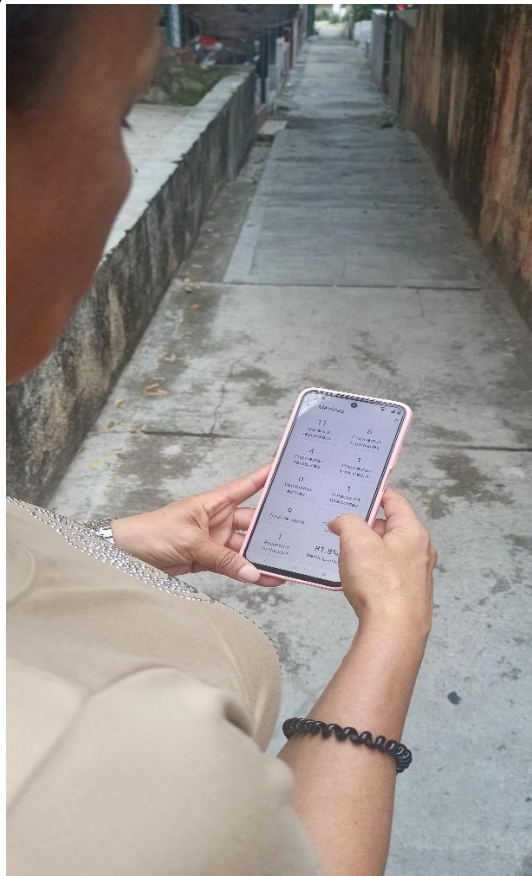
Figura 68. Usuario en la pantalla inicial de la aplicación



Fuente: Autor

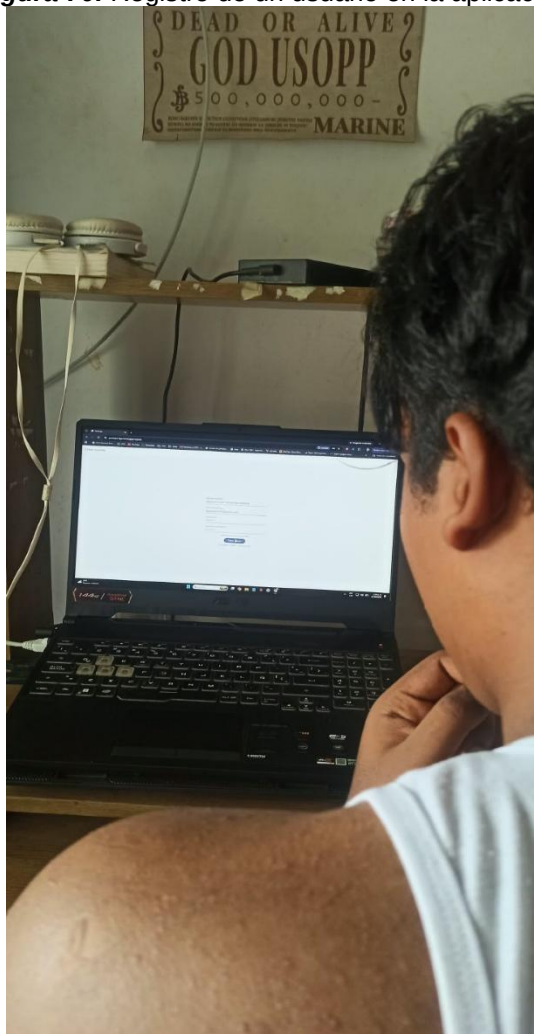
³ Nota: El usuario no dio permiso de mostrar la placa de su motocicleta, por lo que se editó para censurarla.

Figura 69. Usuario admin en la sección de métricas



Fuente: Autor

Figura 70. Registro de un usuario en la aplicación



Fuente: Autor

Después de usar la aplicación, los participantes respondieron un cuestionario de cuatro preguntas cerradas con escala tipo Likert. La siguiente tabla presenta la distribución de las respuestas.

Tabla 39.

Resultados de la encuesta de evaluación (n = 10)

Pregunta	Muy positivo	Positivo	Neutral	Negativo	Sin respuesta	Media (1 a 5)
P1. Satisfacción	6 (60 %)	2 (20 %)	2 (20 %)	0	0	4,4

con la experiencia general de uso						
P2. Probabilidad de recomendar la aplicación en su comunidad	2 (20 %)	5 (50 %)	1 (10 %)	0	2 (20 %)	4,1 (n = 8)
P3. Facilidad para usar las principales funciones	0	7 (70 %)	3 (30 %)	0	0	3,7
P4. Satisfacción con el funcionamiento de las funcionalidades	3 (30 %)	5 (50 %)	2 (20 %)	0	0	4,1 ⁴

Nota: Tabla elaborada por el autor a partir de las respuestas del formulario.

El indicador de satisfacción general definido para el proyecto (Tabla 1) corresponde a la P1, y se calculó como la proporción de participantes que respondieron “Satisfecho” o “Muy satisfecho”. Ocho de los diez participantes lo hicieron (80 %). La P4, sobre el funcionamiento de las funcionalidades, arrojó el mismo valor. Ningún participante dio una respuesta negativa en ninguna de las cuatro preguntas.

La facilidad de uso fue el aspecto con menor proporción de respuestas positivas (70 %): siete participantes calificaron las funciones como fáciles y tres como ni fáciles ni difíciles, y ninguno eligió “Muy fácil”. Los dos participantes que respondieron “Neutral” en P1 respondieron lo mismo en P4 y “Ni fácil ni difícil” en P3. El cuestionario no incluyó preguntas abiertas, por lo que no se dispone de información sobre las causas.

⁴ Nota: Muy positivo corresponde a “Muy satisfecho”, “Muy probable” o “Muy fácil”; Positivo, a “Satisfecho”, “Probable” o “Fácil”. Las medias se calcularon asignando valores de 1 a 5 a los niveles de cada escala. En P2 los porcentajes se calcularon sobre los 10 participantes; sobre las 8 respuestas válidas, 7 son positivas (87,5 %)

Medición posterior de percepción de transparencia y participación

Se aplicó un formulario de cinco preguntas cerradas con escala de acuerdo (totalmente en desacuerdo, en desacuerdo, ni de acuerdo ni en desacuerdo, de acuerdo, totalmente de acuerdo), codificada de 1 a 5. Las preguntas 1 a 3 miden transparencia y las preguntas 4 y 5, participación, se obtuvieron 10 respuestas. Por la imposibilidad de una medición previa, esta es una única medición posterior al uso.

Tabla 40.

Resultados de la medición posterior (n = 10)

Pregunta	Dimensión	Totalmente de acuerdo	De acuerdo	De acuerdo o más	Media
1. Estado de propuestas y proyectos	Transparencia	7	3	10 de 10 (100 %)	4,7
2. Resultados de votaciones claros	Transparencia	6	4	10 de 10 (100 %)	4,6
3. Confianza en el registro y conteo del voto	Transparencia	8	2	10 de 10 (100 %)	4,8
4. Proponer y votar sin reuniones presenciales	Participación	7	3	10 de 10 (100 %)	4,7
5. Mayor posibilidad de que la opinión sea tomada en cuenta	Participación	8	2	10 de 10 (100 %)	4,8

Nota: Tabla elaborada por el autor a partir de los resultados de la encuesta

Los 10 participantes respondieron “de acuerdo” o “totalmente de acuerdo” en las cinco preguntas. No hubo respuestas neutrales ni en desacuerdo. La media fue de 4,70 en transparencia y de 4,75 en participación, con una media global de 4,72 sobre 5. En transparencia, la confianza en que el voto se registra una vez y se cuenta correctamente (pregunta 3) obtuvo la media más alta (4,8). La claridad de los resultados de las votaciones (pregunta 2) obtuvo la más baja (4,6), con cuatro respuestas “de acuerdo” y seis “totalmente de acuerdo”. En participación, la posibilidad de que la opinión sea tomada en cuenta (pregunta 5) alcanzó 4,8 y la

facilidad para proponer y votar sin reuniones presenciales (pregunta 4), 4,7. Con el umbral de 80 % de respuestas favorables definido para la evaluación, ambas dimensiones se cumplen.

La medición es única y posterior al uso, por lo que describe la percepción de los participantes, pero no permite atribuir a la plataforma un cambio respecto de una situación anterior. La muestra es de 10 personas de una sola comunidad, de modo que los resultados no son generalizables. Las respuestas se concentran en los dos valores más altos de la escala, lo que deja poco margen para distinguir entre preguntas. Esa concentración puede deberse a la deseabilidad social, porque los participantes conocían al autor del sistema. Además, las respuestas se registraron en un intervalo de unos dos minutos, lo que indica una aplicación en una sola sesión, y el instrumento no incluyó preguntas abiertas que permitan explicar las valoraciones.

5.6. Análisis de resultados

La siguiente tabla compara cada entregable con la evidencia obtenida

Tabla 41.

Cumplimiento de los entregables

Entregable	Evidencia	Resultado
Plataforma funcional en web y móvil	PF-84 a PF-86	Cumplido para web y Android; sin versión iOS
Al menos cuatro módulos funcionales	PF-21 a PF-65 y PF-77 a PF-79	Cumplido: cuatro módulos más tres complementarios
Autenticación y control de acceso validado	PF-01 a PF-08, PF-12 a PF-14, PF-72 a PF-76, PF-82 y PF-83	Cumplido
Modelo de datos relacional documentado	14 tablas documentadas (Tablas 7 a 20)	Cumplido
Evaluación con al menos 10 usuarios y satisfacción $\geq 80\%$	Encuesta con 10 participantes; P1 = 80 %	Cumplido en el límite del umbral

Nota: Tabla elaborada por el autor

El 100 % de las pruebas funcionales dio el resultado esperado, y las pruebas de seguridad sobre el voto y los permisos se superaron sin excepciones dentro del alcance de PF-54. Esto indica que las funcionalidades definidas en los requisitos operan en las dos plataformas probadas.

En la evaluación con usuarios, la satisfacción general alcanzó el umbral del 80 % sin superarlo. Con diez participantes, cada respuesta representa diez puntos porcentuales, de modo que una respuesta distinta en P1 habría dejado el resultado en 70 %. La muestra es del tamaño mínimo exigido, proviene de una sola comunidad y no se seleccionó con un procedimiento probabilístico, por lo que los resultados describen la percepción de este grupo y no pueden generalizarse a la población de Bucaramanga.

Las pruebas funcionales verificaron el funcionamiento, y los indicadores de la prueba y el cuestionario describen la actividad y la aceptación del grupo participante.

En las pruebas de rendimiento las operaciones principales de la plataforma respondieron en menos de un segundo, con y sin usuarios concurrentes, y el proceso de votación mantuvo su integridad bajo envío simultáneo, incluyendo el bloqueo del voto doble. Estos resultados aportan evidencia a los requisitos de rendimiento y concurrencia a la escala evaluada, y las limitaciones anteriores delimitan hasta dónde puede extenderse esa afirmación. El script pruebas_rendimiento.mjs se incluyen como anexo para permitir la reproducción de las pruebas.

Para las pruebas de usabilidad la evaluación aporta evidencia de que los ciudadanos pudieron completar todas las tareas del protocolo (28 de 28), la mayoría

sin ayuda, y que perciben la plataforma con una usabilidad “buena” y aceptable (SUS de 76.25, IC 95% de 70.58 a 81.92). La función de votación fue la mejor resuelta. Las oportunidades de mejora identificadas son la consulta del avance de un proyecto, la configuración del alias y la facilidad de uso percibida. El tamaño reducido de la muestra de tareas y la falta de observaciones cualitativas limitan el alcance de estas conclusiones, y deben validarse con más participantes antes de generalizarlas. Los datos originales y los cálculos se incluyen como anexo en el archivo evaluacion_usabilidad_participa.xlsx.

6. CONCLUSIONES

El análisis de requisitos, que produjo requisitos funcionales y no funcionales, historias de usuario y un diagrama de casos de uso, delimitó el alcance del sistema. Sobre ese alcance se diseñó una arquitectura cliente-servidor en tres capas, con Flutter como capa de presentación y Supabase como único backend. Al no existir un servidor intermedio propio, las versiones web y móvil comparten la misma base de datos y los mismos usuarios.

Flutter permitió construir las versiones web y Android a partir de una misma base de código, con las mismas funcionalidades en ambos entornos y una interfaz que se adapta al ancho de la ventana en la web. La versión para iOS no se construyó y la aplicación requiere conexión permanente a internet.

El diseño inicial del voto usaba una sola tabla que relacionaba al usuario con la opción elegida, de modo que cualquier rol con lectura sobre ella habría podido saber qué eligió cada ciudadano. Separar la participación de la elección en dos tablas, y no definir ninguna política de lectura sobre la segunda, permitió que el voto único y el voto secreto coexistieran frente a los roles de la aplicación, incluido el administrador, y que los resultados se consultaran solo mediante una vista agregada. Esta protección no alcanza a quien tenga acceso directo a la base de datos: las dos tablas asignan la fecha con `now()` y, como `emitir_voto()` inserta en ambas dentro de una misma transacción, las marcas de tiempo coinciden y permitirían emparejar participación y elección.

Ubicar las validaciones críticas en funciones de PostgreSQL y en políticas de seguridad por filas, y no en Flutter, hizo que las reglas se aplicaran también a

peticiones que no provenían de la interfaz, como las de una sesión que seguía abierta después de desactivar la cuenta. Por la misma razón, incluir la clave pública en el cliente no amplía lo que un usuario puede hacer: el límite lo establecen las políticas.

La transición automática de las votaciones se resolvió con una tarea programada en la base de datos. Su diseño reveló un caso no previsto: las votaciones cuya ventana vencía sin haberse activado quedaban en estado programada. Se resolvió con una regla adicional que las finaliza directamente.

En la evaluación de usabilidad, los participantes valoraron de forma favorable la experiencia general y el funcionamiento de las funcionalidades, y ninguno dio una respuesta negativa. La facilidad de uso fue el aspecto con menor proporción de respuestas positivas. Como el grupo evaluado proviene de una sola comunidad y el cuestionario solo tuvo preguntas cerradas, no fue posible identificar las causas; las evaluaciones posteriores deben incluir preguntas abiertas.

7. RECOMENDACIONES

Eliminar la columna fecha de la tabla votos o reducir su precisión, porque las tablas votos y voto_registros asignan la fecha con now() y la función emitir_voto() inserta en ambas dentro de la misma transacción. Las marcas de tiempo coinciden y permiten vincular identidad y elección a quien tenga acceso directo a la base de datos.

Incorporar una verificación de identidad en el registro. Hoy basta un correo y una contraseña, lo que no impide que una persona cree varias cuentas y vote más de una vez. Una alternativa es articular la autenticación con los servicios ciudadanos digitales del Decreto 620 de 2020, ya referenciado en el marco legal. Incorporar en el registro el aviso de tratamiento de datos personales, la autorización del titular y una política de tratamiento publicada en la aplicación, de acuerdo con la Ley 1581 de 2012.

Añadir pruebas de carga y una revisión de seguridad de las políticas de acceso. Los requisitos no funcionales de rendimiento y concurrencia no tienen evidencia de prueba, porque la matriz de validación cubre solo funcionalidad.

Habilitar consulta sin conexión de propuestas y proyectos, de modo que la votación sea la única operación que exija red. El estado del arte ya plantea la baja conectividad como una barrera de adopción.

Generar la versión para iOS. Requiere un equipo Apple y una cuenta de desarrollador, y debe presupuestarse. También conviene añadir notificaciones push, porque las actuales son solo mensajes internos de la aplicación o dos correos

por hora por parte de Supabase, y una exportación real de los reportes de métricas, ya que hoy solo se copia un resumen al portapapeles.

Para la apropiación del sistema, acordar con la entidad o la organización comunitaria quién administra la plataforma, quién asume el costo del servicio y cómo se capacita a los administradores.

8. REFERENCIAS BIBLIOGRÁFICAS

Ajuntament de Barcelona. (s. f.). *Decidim*.

<https://ajuntament.barcelona.cat/innovaciodemocratica/es/decidim>

Alcaldía de Bucaramanga. (s. f.). *Participa*. <https://www.bucaramanga.gov.co/participa/>

Alpert, S. (2015, 26 de marzo). *Introducing React Native*. React Blog.

<https://legacy.reactjs.org/blog/2015/03/26/introducing-react-native.html>

Announcing Flutter 2. (2021, 3 de marzo). Google Developers Blog.

<https://developers.googleblog.com/announcing-flutter-2/>

Ansell, C., & Gash, A. (2008). Collaborative governance in theory and practice. *Journal of Public Administration Research and Theory*, 18(4), 543–571.

<https://doi.org/10.1093/jopart/mum032>

Arnstein, S. R. (1969). A ladder of citizen participation. *Journal of the American Institute of Planners*, 35(4), 216–224. <https://doi.org/10.1080/01944366908977225>

Bangor, A., Kortum, P. T., & Miller, J. T. (2008). An empirical evaluation of the System Usability Scale. *International Journal of Human-Computer Interaction*, 24(6), 574–594. <https://doi.org/10.1080/10447310802205776>

Bangor, A., Kortum, P. T., & Miller, J. T. (2009). Determining what individual SUS scores mean: Adding an adjective rating scale. *Journal of Usability Studies*, 4(3), 114–123.

https://uxpajournal.org/wp-content/uploads/sites/7/pdf/JUS_Bangor_May2009.pdf

Bass, L., Clements, P., & Kazman, R. (2021). *Software architecture in practice* (4.^a ed.). Addison-Wesley.

- Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). *Manifesto for agile software development*. <https://agilemanifesto.org/>
- Biørn-Hansen, A., Grønli, T.-M., & Ghinea, G. (2018). A survey and taxonomy of core concepts and research challenges in cross-platform mobile development. *ACM Computing Surveys*, 51(5), Artículo 108. <https://doi.org/10.1145/3241739>
- Boté-Vericad, J. (2025, 30 de mayo). *Innovación y transformación digital en información y documentación*. <https://ojs.letras.up.pt/index.php/paginasueb/article/view/15293>
- Brooke, J. (1996). SUS: A “quick and dirty” usability scale. En P. W. Jordan, B. Thomas, B. A. Weerdmeester, & I. L. McClelland (Eds.), *Usability evaluation in industry* (pp. 189–194). Taylor & Francis.
- Chacon, S., & Straub, B. (2014). *Pro Git* (2.^a ed.). Apress. <https://git-scm.com/book/es/v2>
- CitizenLab. (2025). *CitizenLab open source repository* [Software]. GitHub. <https://github.com/CitizenLabDotCo/citizenlab>
- Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, 13(6), 377–387. <https://doi.org/10.1145/362384.362685>
- Comisión Económica para América Latina y el Caribe. (2022, 18 de noviembre). *Agenda digital para América Latina y el Caribe (eLAC2024)* (LC/CMSI.8/5). <https://hdl.handle.net/11362/48497>
- Comité de Ministros del Consejo de Europa. (2017). *Recommendation CM/Rec(2017)5 of the Committee of Ministers to member States on standards for e-voting*. [https://search.coe.int/cm/Pages/result_details.aspx?Reference=CM/Rec\(2017\)5](https://search.coe.int/cm/Pages/result_details.aspx?Reference=CM/Rec(2017)5)

F-DC-125

INFORME FINAL DE TRABAJO DE GRADO EN MODALIDAD DE PROYECTO
DE INVESTIGACIÓN, DESARROLLO TECNOLÓGICO, MONOGRAFÍA,
EMPRENDIMIENTO Y SEMINARIO

VERSIÓN: 2.0

Constitución Política de Colombia. (1991). Secretaría del Senado de la República.

http://www.secretariassenado.gov.co/senado/basedoc/constitucion_politica_1991.html

Consul Democracy Foundation. (s. f.). *About the project.*

<https://consuldemocracy.org/about-us/>

Consul Project. (2025). *The most complete citizen participation tool for an open, transparent and democratic government.* <https://consuldemocracy.org/>

Dart. (s. f.). *Dart overview.* <https://dart.dev/overview>

Decidim. (2019). *Decidim is a digital platform for citizen participation.* <https://decidim.org/>

Decreto 620 del 2 de mayo de 2020. (2020, 2 de mayo). Agencia Presidencial de Cooperación Internacional. <https://www.apccolombia.gov.co/normativa/decreto-620-del-2-de-mayo-de-2020>

Decreto 1008 de 2018. (2018, 14 de junio). Función Pública, Gestor Normativo. <https://www1.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=86902>

Decreto 1074 de 2015, “Decreto Único Reglamentario del Sector Comercio, Industria y Turismo”. (2015, 26 de mayo). Superintendencia de Industria y Comercio. <https://sedeelectronica.sic.gov.co/transparencia/normativa/decreto-1074-de-2015-decreto-unico-reglamentario-del-sector-comercio-industria-y-turismo>

Decreto 1360 de 1989. (1989, 23 de junio). Alcaldía Mayor de Bogotá. <https://www.alcaldiabogota.gov.co/sisjur/normas/Norma1.jsp?i=10575>

Decreto 1377 de 2013. (2013, 27 de junio). Alcaldía Mayor de Bogotá. <https://www.alcaldiabogota.gov.co/sisjur/normas/Norma1.jsp?i=53646>

F-DC-125

INFORME FINAL DE TRABAJO DE GRADO EN MODALIDAD DE PROYECTO
DE INVESTIGACIÓN, DESARROLLO TECNOLÓGICO, MONOGRAFÍA,
EMPRENDIMIENTO Y SEMINARIO

VERSIÓN: 2.0

Defensoría del Pueblo de Colombia. (s. f.). *Participa.*

<https://www.defensoria.gov.co/participacion-ciudadana>

Democracy Technologies. (s. f.). *DemocraciaOS.* [https://democracy-](https://democracy-technologies.org/tool/democraciaos/)

[technologies.org/tool/democraciaos/](https://democracy-technologies.org/tool/democraciaos/)

Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of database systems* (7.^a ed.). Pearson.

Ferraiolo, D. F., Sandhu, R., Gavrila, S., Kuhn, D. R., & Chandramouli, R. (2001). Proposed

NIST standard for role-based access control. *ACM Transactions on Information and System Security*, 4(3), 224–274. <https://doi.org/10.1145/501978.501980>

Fielding, R. T., & Taylor, R. N. (2002). Principled design of the modern Web architecture.

ACM Transactions on Internet Technology, 2(2), 115–150. <https://doi.org/10.1145/514183.514185>

Flutter. (s. f.-a). *Flutter architectural overview.*

<https://docs.flutter.dev/resources/architectural-overview>

Flutter. (s. f.-b). *Flutter documentation.* <https://docs.flutter.dev/>

Ghale, A. (2025). *Elector: Online election voting mobile application* [Software]. GitHub.

<https://github.com/anishghale007/elector>

GitHub. (s. f.). *¿Qué es GitHub?* GitHub Docs. [https://docs.github.com/es/get-started/start-](https://docs.github.com/es/get-started/start-your-journey/what-is-github)

[your-journey/what-is-github](https://docs.github.com/es/get-started/start-your-journey/what-is-github)

Gotel, O. C. Z., & Finkelstein, A. C. W. (1994). An analysis of the requirements traceability

problem. En *Proceedings of the First International Conference on Requirements Engineering* (pp. 94–101). IEEE. <https://doi.org/10.1109/ICRE.1994.292398>

Gutiérrez, W. G., & Romero, W. (2024). *La transformación digital como herramienta en la modernización de la gestión de proyectos* [Trabajo de grado, Corporación

F-DC-125

INFORME FINAL DE TRABAJO DE GRADO EN MODALIDAD DE PROYECTO
DE INVESTIGACIÓN, DESARROLLO TECNOLÓGICO, MONOGRAFÍA,
EMPRENDIMIENTO Y SEMINARIO

VERSIÓN: 2.0

Universitaria Minuto de Dios]. Repositorio UNIMINUTO.

<https://repository.uniminuto.edu/server/api/core/bitstreams/2ecddaff-1e3b-4fe6-a2b1-d044ac0b5ece/content>

Härder, T., & Reuter, A. (1983). Principles of transaction-oriented database recovery. *ACM Computing Surveys*, 15(4), 287–317. <https://doi.org/10.1145/289.291>

Hookdeck. (2023). *SupaPoll: Next.js voting application with Supabase* [Software]. GitHub. <https://github.com/hookdeck/supapoll>

Hu, V. C., Ferraiolo, D., Kuhn, R., Schnitzer, A., Sandlin, K., Miller, R., & Scarfone, K. (2014). *Guide to attribute based access control (ABAC) definition and considerations* (NIST Special Publication 800-162). National Institute of Standards and Technology. <https://doi.org/10.6028/NIST.SP.800-162>

International Energy Agency. (2025). *Energy and AI*. <https://www.iea.org/reports/energy-and-ai/executive-summary>

International Organization for Standardization. (2015). *Quality management systems — Fundamentals and vocabulary* (ISO 9000:2015).

International Organization for Standardization. (2019). *Ergonomics of human-system interaction - Part 210: Human-centred design for interactive systems* (ISO 9241-210:2019).

Islam, S. (2024). *Flutter voting app* [Software]. GitHub. <https://github.com/siraajul/voting>

ISO/IEC. (2023). *Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Product quality model* (ISO/IEC 25010:2023). <https://www.iso.org/standard/78176.html>

F-DC-125

INFORME FINAL DE TRABAJO DE GRADO EN MODALIDAD DE PROYECTO
DE INVESTIGACIÓN, DESARROLLO TECNOLÓGICO, MONOGRAFÍA,
EMPRENDIMIENTO Y SEMINARIO

VERSIÓN: 2.0

ISO/IEC/IEEE. (2017a). *Systems and software engineering - Measurement process*
(ISO/IEC/IEEE 15939:2017).

ISO/IEC/IEEE. (2017b). *Systems and software engineering - Vocabulary* (ISO/IEC/IEEE
24765:2017).

ISO/IEC/IEEE. (2022). *Software, systems and enterprise - Architecture description*
(ISO/IEC/IEEE 42010:2022).

Jiménez, J. R., & Arce, J. R. (2024). *Gobernanza digital inclusiva: Plataforma colaborativa
para la participación ciudadana* [Trabajo de grado, Universidad EAN]. Repositorio
Universidad EAN.

[https://repository.universidadean.edu.co/server/api/core/bitstreams/d564a606-fcc3-
4291-b4ee-1eb78cc067f6/content](https://repository.universidadean.edu.co/server/api/core/bitstreams/d564a606-fcc3-4291-b4ee-1eb78cc067f6/content)

Laudon, K. C., & Laudon, J. P. (2022). *Management information systems: Managing the
digital firm* (17.^a ed.). Pearson.

Ley 23 de 1982. (1982, 28 de enero). Función Pública, Gestor Normativo.
<https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=3431>

Ley 99 de 1993. (1993, 22 de diciembre). Secretaría del Senado de la República.
http://www.secretariasenado.gov.co/senado/basedoc/ley_0099_1993.html

Ley 134 de 1994. (1994, 31 de mayo). Congreso de la República de Colombia. Diario Oficial
No. 41.373.
http://www.secretariasenado.gov.co/senado/basedoc/ley_0134_1994.html

Ley 527 de 1999. (1999, 18 de agosto). Congreso de la República de Colombia. Diario
Oficial No. 43.673.
http://www.secretariasenado.gov.co/senado/basedoc/ley_0527_1999.html

F-DC-125

INFORME FINAL DE TRABAJO DE GRADO EN MODALIDAD DE PROYECTO
DE INVESTIGACIÓN, DESARROLLO TECNOLÓGICO, MONOGRAFÍA,
EMPRENDIMIENTO Y SEMINARIO

VERSIÓN: 2.0

Ley 1273 de 2009. (2009, 5 de enero). Congreso de la República de Colombia.

http://www.secretariasenado.gov.co/senado/basedoc/ley_1273_2009.html

Ley 1341 de 2009. (2009, 30 de julio). Congreso de la República de Colombia.

http://www.secretariasenado.gov.co/senado/basedoc/ley_1341_2009.html

Ley 1581 de 2012. (2012, 17 de octubre). Congreso de la República de Colombia. Diario

Oficial

No.

48.587.

http://www.secretariasenado.gov.co/senado/basedoc/ley_1581_2012.html

Ley 1672 de 2013. (2013, 19 de julio). Función Pública, Gestor Normativo.

<https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=53825>

Ley 1712 de 2014. (2014, 6 de marzo). Congreso de la República de Colombia.

http://www.secretariasenado.gov.co/senado/basedoc/ley_1712_2014.html

Ley 1757 de 2015. (2015, 6 de julio). Congreso de la República de Colombia.

http://www.secretariasenado.gov.co/senado/basedoc/ley_1757_2015.html

Ley 1915 de 2018. (2018, 12 de julio). Alcaldía Mayor de Bogotá.

<https://www.alcaldiabogota.gov.co/sisjur/normas/Norma1.jsp?i=97476&dt=S>

Ley 1978 de 2019. (2019, 25 de julio). Función Pública, Gestor Normativo.

<https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=98210>

Marcotte, E. (2010, 25 de mayo). *Responsive web design*. A List Apart.

<https://alistapart.com/article/responsive-web-design/>

Mell, P., & Grance, T. (2011). *The NIST definition of cloud computing* (NIST Special

Publication 800-145). National Institute of Standards and Technology.

<https://doi.org/10.6028/NIST.SP.800-145>

Ministerio de Tecnologías de la Información y las Comunicaciones. (s. f.). *Compilación jurídica del MINTIC - Decreto Único Reglamentario del Sector de Tecnologías de la Información y las Comunicaciones (Decreto 1078 de 2015)*. Avance Jurídico Casa Editorial S.A.S.

https://normograma.mintic.gov.co/mintic/compilacion/dudst_decreto_unico_reglamentario_sector_tecnologias_informacion_comunicaciones_decreto_1078_2015.html

Montez, S. C., & Rodríguez, M. J. R. (2025). Evolución de la participación ciudadana en la gestión pública: Tendencias y desafíos. *I+D Internacional Revista Científica y Académica*, 4(1), 1–22. <https://doi.org/10.63636/3078-1639.v4.n1.24>

Naser, A. (2018, 22 y 23 de octubre). *Gobernando el futuro: Los ejes de un nuevo Estado para la ciudadanía* [Presentación]. Jornadas Planificación para el Desarrollo con Visión de Futuro, CEPAL, Santiago, Chile.

https://www.cepal.org/sites/default/files/events/files/panel_3_presentacion_3_alejandra_naser.pdf

Naser, A., Williner, A., & Sandoval, C. (2021). *Participación ciudadana en los asuntos públicos: Un elemento estratégico para la Agenda 2030 y el gobierno abierto* (LC/TS.2020/184). Comisión Económica para América Latina y el Caribe (CEPAL).

https://repositorio.cepal.org/bitstream/handle/11362/46645/S2000907_es.pdf

National Institute of Standards and Technology. (2004). *Standards for security categorization of federal information and information systems* (FIPS PUB 199).

<https://doi.org/10.6028/NIST.FIPS.199>

Nielsen, J. (1993). *Usability engineering*. Academic Press.

Nielsen, J., & Landauer, T. K. (1993). A mathematical model of the finding of usability problems. En *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems* (pp. 206–213). ACM.
<https://doi.org/10.1145/169059.169166>

Object Management Group. (2017). *OMG Unified Modeling Language (OMG UML)* (Versión 2.5.1; formal/17-12-05). <https://www.omg.org/spec/UML/2.5.1/>

Osorio-Pérez, O. (2023). El concepto de comunidad en los proyectos de intervención comunitaria: Disonancias, opacidades y rupturas. *Alteridades*, 33(65), 61–72.
<https://doi.org/10.24275/uam/izt/dcsh/alteridades/2023v33n65/Osorio>

PostgreSQL Global Development Group. (s. f.-a). *A brief history of PostgreSQL*. PostgreSQL Documentation. <https://www.postgresql.org/docs/current/history.html>

PostgreSQL Global Development Group. (s. f.-b). *Constraints*. PostgreSQL Documentation. <https://www.postgresql.org/docs/current/ddl-constraints.html>

PostgreSQL Global Development Group. (s. f.-c). *CREATE FUNCTION*. PostgreSQL Documentation. <https://www.postgresql.org/docs/current/sql-createfunction.html>

PostgreSQL Global Development Group. (s. f.-d). *Indexes*. PostgreSQL Documentation. <https://www.postgresql.org/docs/current/indexes.html>

PostgreSQL Global Development Group. (s. f.-e). *PostgreSQL: About*. <https://www.postgresql.org/about/>

PostgreSQL Global Development Group. (s. f.-f). *Row security policies*. PostgreSQL Documentation. <https://www.postgresql.org/docs/current/ddl-rowsecurity.html>

PostgreSQL Global Development Group. (s. f.-g). *Transactions*. PostgreSQL Documentation. <https://www.postgresql.org/docs/current/tutorial-transactions.html>

PostgreSQL Global Development Group. (s. f.-h). *Triggers*. PostgreSQL Documentation.

<https://www.postgresql.org/docs/current/triggers.html>

Pressman, R. S., & Maxim, B. R. (2020). *Software engineering: A practitioner's approach* (9.^a ed.). McGraw-Hill Education.

Project Management Institute. (2021). *A guide to the project management body of knowledge (PMBOK® guide)* (7.^a ed.). Project Management Institute.

Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo, de 27 de abril de 2016, relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos. (2016). Diario Oficial de la Unión Europea, L 119, 1–88. <https://eur-lex.europa.eu/legal-content/ES/TXT/?uri=CELEX:32016R0679>

Sandhu, R. S., Coyne, E. J., Feinstein, H. L., & Youman, C. E. (1996). Role-based access control models. *Computer*, 29(2), 38–47. <https://doi.org/10.1109/2.485845>

Sauro, J., & Lewis, J. R. (2016). *Quantifying the user experience: Practical statistics for user research* (2.^a ed.). Morgan Kaufmann.

Schwaber, K., & Sutherland, J. (2020). *The Scrum guide*. <https://scrumguides.org/scrum-guide.html>

Sevilla-Gonzalez, M. D. R., Moreno Loaeza, L., Lazaro-Carrera, L. S., Bourguet Ramirez, B., Vázquez Rodríguez, A., Peralta-Pedrero, M. L., & Almeda-Valdes, P. (2020). Spanish version of the System Usability Scale for the assessment of electronic tools: Development and validation. *JMIR Human Factors*, 7(4), Artículo e21161. <https://doi.org/10.2196/21161>

Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., & Elmqvist, N. (2016). *Designing the user interface: Strategies for effective human-computer interaction* (6.^a ed.). Pearson.

Sierra Fructuoso, J., González Salcedo, A., & Moreno Martínez, C. (2024, 21 de noviembre). *Plataformas de participación ciudadana online: Una propuesta metodológica para su evaluación* [Ponencia]. GT. 3 La plasmación de valores en la Administración Pública: Gobernanza ética y desarrollo sostenible, Instituto Complutense de Ciencia de la Administración, Universidad Complutense de Madrid.
<https://www.ucm.es/icca/plataformas-de-participacion-ciudadana>

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2020). *Database system concepts* (7.^a ed.). McGraw-Hill Education.

Smart City Soft. (s. f.). *Smart City Soft: Conectando gobiernos, empoderando ciudadanos*.
<https://smartcitysoft.com/es-home/>

Sneath, T. (2018, 4 de diciembre). *Flutter 1.0: Google's portable UI toolkit*. Google Developers Blog. <https://developers.googleblog.com/flutter-10-googles-portable-ui-toolkit/>

Sommerville, I. (2016). *Software engineering* (10.^a ed.). Pearson.

Supabase. (s. f.-a). *Architecture*. Supabase Docs.
<https://supabase.com/docs/guides/getting-started/architecture>

Supabase. (s. f.-b). *REST API*. Supabase Docs. <https://supabase.com/docs/guides/api>

Supabase. (s. f.-c). *Use Supabase with Flutter*. <https://supabase.com/docs/guides/getting-started/quickstarts/flutter>

F-DC-125

INFORME FINAL DE TRABAJO DE GRADO EN MODALIDAD DE PROYECTO
DE INVESTIGACIÓN, DESARROLLO TECNOLÓGICO, MONOGRAFÍA,
EMPRENDIMIENTO Y SEMINARIO

VERSIÓN: 2.0

Supabase. (2026a, 25 de septiembre). *Auth*. Supabase Docs.

<https://supabase.com/docs/guides/auth>

Supabase. (2026b, 25 de septiembre). *Row Level Security*. Supabase Docs.

<https://supabase.com/docs/guides/database/postgres/row-level-security>

Temoshok, D., Proud-Madruga, D., Choong, Y.-Y., Galluzzo, R., Gupta, S., LaSalle, C., Lefkovitz, N., & Regenscheid, A. (2025). *Digital identity guidelines* (NIST Special Publication 800-63-4). National Institute of Standards and Technology.

<https://doi.org/10.6028/NIST.SP.800-63-4>

Torres, V., Naser, A., & Valenzuela, P. (2025). *Gobernanza colaborativa y procesos de deliberación democrática: Transformación digital en la participación* (LC/TS.2025/48). Comisión Económica para América Latina y el Caribe (CEPAL).

<https://hdl.handle.net/11362/84485>

Vargas, M. D., & Cruz, C. P. M. (2024). Impacto de la digitalización en la eficiencia de las cooperativas. *Trascender, Contabilidad y Gestión*, 9(27), 2–30.

<https://doi.org/10.36791/tcg.v9i27.265>

Vial, G. (2019). Understanding digital transformation: A review and a research agenda. *The Journal of Strategic Information Systems*, 28(2), 118–144.

<https://doi.org/10.1016/j.jsis.2019.01.003>

Zea, S. T. (2023). *Confianza ciudadana: Retos y oportunidades en el sector público colombiano* [Trabajo de grado, Universidad Militar Nueva Granada]. Repositorio

UMNG. <https://repository.umng.edu.co/server/api/core/bitstreams/8afee611-5f5c-482f-a58f-aaa708159d68/content>

9. APÉNDICES

APÉNDICE A: Cuestionario de diagnóstico

1. ¿Considera usted que en su comunidad existen dificultades para organizar, gestionar o dar seguimiento a proyectos comunitarios?

- ☐ Si
- ☐ No
- ☐ No estoy seguro

2. ¿Qué tan necesario considera el uso de una herramienta digital que facilite la gestión de proyectos comunitarios (propuestas, seguimiento, participación ciudadana, etc.)?

- ☐ Muy necesario
- ☐ Necesario
- ☐ Poco necesario
- ☐ Nada necesario

3. ¿Cuáles de las siguientes funcionalidades considera más importantes en una aplicación de este tipo? (Selección múltiple)

- ☐ Registro y publicación de propuestas
- ☐ Seguimiento del avance de proyectos
- ☐ Votaciones o priorización de iniciativas
- ☐ Notificaciones y alertas
- ☐ Transparencia en la información (reportes, estados, etc.)

4. ¿En qué tipo de plataforma preferiría utilizar esta herramienta?

- ☐ Aplicación móvil (Android/iOS)

- ☐ Página web
- ☐ Ambas (móvil y web)
- ☐ No utilizaría ninguna

5. ¿Estaría dispuesto(a) a utilizar una plataforma digital para participar activamente en proyectos comunitarios (opinar, votar, proponer ideas)?

- ☐ Sí
- ☐ No
- ☐ Tal vez

APÉNDICE B: Encuesta de satisfacción del sistema

1. ¿Qué tan satisfecho(a) está con la experiencia general de uso de la aplicación?

- ☐ Muy insatisfecho
- ☐ Insatisfecho
- ☐ Neutral
- ☐ Satisfecho
- ☐ Muy satisfecho

2. ¿Qué tan probable es que recomiende esta aplicación a otras personas de su comunidad?

- ☐ Nada probable
- ☐ Poco probable
- ☐ Neutral
- ☐ Probable
- ☐ Muy probable

3. ¿Qué tan fácil le resultó utilizar las principales funciones de la aplicación?

- ☐ Muy difícil
- ☐ Difícil
- ☐ Ni fácil ni difícil
- ☐ Fácil
- ☐ Muy fácil

4. ¿Qué tan satisfecho(a) está con el funcionamiento de las diferentes opciones y funcionalidades disponibles en la aplicación?

- ☐ Muy insatisfecho
- ☐ Insatisfecho
- ☐ Neutral
- ☐ Satisfecho
- ☐ Muy satisfecho

APÉNDICE C: Encuesta de usabilidad de la aplicación

1. Creo que me gustaría usar esta aplicación con frecuencia.

- ☐ 1 — Totalmente en desacuerdo
- ☐ 2 — En desacuerdo
- ☐ 3 — Ni de acuerdo ni en desacuerdo
- ☐ 4 — De acuerdo
- ☐ 5 — Totalmente de acuerdo

2. Encontré la aplicación innecesariamente compleja.

- ☐ 1 — Totalmente en desacuerdo

F-DC-125

INFORME FINAL DE TRABAJO DE GRADO EN MODALIDAD DE PROYECTO
DE INVESTIGACIÓN, DESARROLLO TECNOLÓGICO, MONOGRAFÍA,
EMPRENDIMIENTO Y SEMINARIO

VERSIÓN: 2.0

- 2 — En desacuerdo
- 3 — Ni de acuerdo ni en desacuerdo
- 4 — De acuerdo
- 5 — Totalmente de acuerdo

3. Pensé que la aplicación era fácil de usar.

- 1 — Totalmente en desacuerdo
- 2 — En desacuerdo
- 3 — Ni de acuerdo ni en desacuerdo
- 4 — De acuerdo
- 5 — Totalmente de acuerdo

4. Creo que necesitaría el apoyo de una persona técnica para poder usar esta aplicación.

- 1 — Totalmente en desacuerdo
- 2 — En desacuerdo
- 3 — Ni de acuerdo ni en desacuerdo
- 4 — De acuerdo
- 5 — Totalmente de acuerdo

5. Encontré que las diversas funciones de la aplicación estaban bien integradas.

- 1 — Totalmente en desacuerdo
- 2 — En desacuerdo
- 3 — Ni de acuerdo ni en desacuerdo
- 4 — De acuerdo
- 5 — Totalmente de acuerdo

6. Pensé que había demasiada inconsistencia en esta aplicación.

- 1 — Totalmente en desacuerdo
- 2 — En desacuerdo
- 3 — Ni de acuerdo ni en desacuerdo
- 4 — De acuerdo
- 5 — Totalmente de acuerdo

7. Imagino que la mayoría de las personas aprenderían a usar esta aplicación muy rápidamente.

- 1 — Totalmente en desacuerdo
- 2 — En desacuerdo
- 3 — Ni de acuerdo ni en desacuerdo
- 4 — De acuerdo
- 5 — Totalmente de acuerdo

8. Encontré la aplicación muy incómoda de usar.

- 1 — Totalmente en desacuerdo
- 2 — En desacuerdo
- 3 — Ni de acuerdo ni en desacuerdo
- 4 — De acuerdo
- 5 — Totalmente de acuerdo

9. Me sentí muy seguro/a al usar la aplicación.

- 1 — Totalmente en desacuerdo
- 2 — En desacuerdo
- 3 — Ni de acuerdo ni en desacuerdo
- 4 — De acuerdo
- 5 — Totalmente de acuerdo

10. Necesité aprender muchas cosas antes de poder empezar a usar la aplicación.

- 1 — Totalmente en desacuerdo
- 2 — En desacuerdo
- 3 — Ni de acuerdo ni en desacuerdo
- 4 — De acuerdo
- 5 — Totalmente de acuerdo

APÉNDICE D: Encuesta de Transparencia y Participación

1. La plataforma me permite saber en qué estado se encuentra cada propuesta o proyecto (en votación, aprobado, en ejecución).

- Totalmente en desacuerdo
- En desacuerdo
- Ni de acuerdo ni en desacuerdo
- De acuerdo
- Totalmente de acuerdo

2. Los resultados de las votaciones se muestran de forma clara y fácil de entender.

- Totalmente en desacuerdo
- En desacuerdo
- Ni de acuerdo ni en desacuerdo
- De acuerdo
- Totalmente de acuerdo

3. Confío en que mi voto se registra una sola vez y se cuenta correctamente.

- Totalmente en desacuerdo

- ☐ En desacuerdo
- ☐ Ni de acuerdo ni en desacuerdo
- ☐ De acuerdo
- ☐ Totalmente de acuerdo

4. La plataforma me facilita proponer iniciativas y votar sin depender de reuniones presenciales.

- ☐ Totalmente en desacuerdo
- ☐ En desacuerdo
- ☐ Ni de acuerdo ni en desacuerdo
- ☐ De acuerdo
- ☐ Totalmente de acuerdo

5. Con la plataforma siento que mi opinión tiene más posibilidades de ser tenida en cuenta en las decisiones de la comunidad.

- ☐ Totalmente en desacuerdo
- ☐ En desacuerdo
- ☐ Ni de acuerdo ni en desacuerdo
- ☐ De acuerdo
- ☐ Totalmente de acuerdo

APÉNDICE E: Planilla para las pruebas de usabilidad

Mediante una planilla de Excel se realizaron la evaluación de usabilidad del sistema, dicha planilla esta alojada en Drive para su revisión

F-DC-125

INFORME FINAL DE TRABAJO DE GRADO EN MODALIDAD DE PROYECTO
DE INVESTIGACIÓN, DESARROLLO TECNOLÓGICO, MONOGRAFÍA,
EMPRENDIMIENTO Y SEMINARIO

VERSIÓN: 2.0

<https://docs.google.com/spreadsheets/d/1hNbsnGm5DjxogtO8KMWog-rdJnj4EaVF/edit?usp=sharing&ouid=107782864437052607009&rtpof=true&sd=true>
e

ELABORADO POR:
Docencia

REVISADO POR:
Sistema Integrado de Gestión

APROBADO POR: Líder del Sistema Integrado de Gestión
FECHA APROBACIÓN: Octubre de 2023

10. ANEXOS

ANEXO A: Script SQL completo

Un archivo .sql el cual contiene todo el script para la creación de la base de datos junto con sus triggers y las políticas RLS.

```
create extension if not exists pgcrypto;

-- 1. ROLES
create table if not exists public.roles (
    id          uuid primary key default gen_random_uuid(),
    nombre text not null unique check (nombre in ('ciudadano','administrador'))
);

insert into public.roles (nombre) values ('ciudadano'), ('administrador')
on conflict (nombre) do nothing;

-- 2. USUARIOS
create table if not exists public.usuarios (
    id          uuid primary key references auth.users(id) on
delete cascade,
    rol_id      uuid not null references public.roles(id),
    nombre_completo text not null,
    correo      text not null unique,
    telefono    text,
    avatar_url  text,
    alias       text check (alias is null or (length(alias)
between 2 and 40)),
    activo      boolean not null default true,
    notificaciones_externas_activas boolean not null default true,
    fecha_registro timestampz not null default now(),
    fecha_aceptacion_terminos timestampz not null default now()
);

-- 3. FUNCIONES HELPER
create or replace function public.es_admin()
returns boolean
```

```
language sql
stable
security definer set search_path = public
as $$
    select exists (
        select 1
        from public.usuarios u
        join public.roles r on r.id = u.rol_id
        where u.id = auth.uid() and r.nombre = 'administrador'
    );
$$;

create or replace function public.handle_new_user()
returns trigger
language plpgsql
security definer set search_path = public
as $$
declare
    v_rol_ciudadano uuid;
begin
    select id into v_rol_ciudadano from public.roles where nombre = 'ciudadano';
    insert into public.usuarios (id, rol_id, nombre_completo, correo,
fecha_aceptacion_terminos)
values (
    new.id,
    v_rol_ciudadano,
    coalesce(new.raw_user_meta_data ->> 'nombre_completo', ''),
    new.email,
    case when (new.raw_user_meta_data ->> 'acepto_terminos')::boolean is true
then now() else null end
);
    return new;
end;
$$;

create or replace function public.usuario_activo()
returns boolean
language sql
stable
security definer set search_path = public
as $$
```

```

    select coalesce((select activo from public.usuarios where id = auth.uid()),
false);
$$;

-- 4. SINCRONIZACIÓN AUTH - PERFIL
create or replace function public.handle_new_user()
returns trigger
language plpgsql
security definer set search_path = public
as $$
declare
    v_rol_ciudadano uuid;
begin
    select id into v_rol_ciudadano from public.roles where nombre = 'ciudadano';
    insert into public.usuarios (id, rol_id, nombre_completo, correo)
    values (
        new.id,
        v_rol_ciudadano,
        coalesce(new.raw_user_meta_data ->> 'nombre_completo', ''),
        new.email
    );
    return new;
end;
$$;

drop trigger if exists on_auth_user_created on auth.users;
create trigger on_auth_user_created
    after insert on auth.users
    for each row execute function public.handle_new_user();

-- 5. ANTI-ESCALADA DE PRIVILEGIOS (usuarios)
create or replace function public.prevenir_escalada_privilegios()
returns trigger
language plpgsql
security definer set search_path = public
as $$
begin
    if auth.uid() is null then
        return new;
    end if;

```

```

if new.rol_id <> old.rol_id and not public.es_admin() then
    raise exception 'No tiene permisos para cambiar su rol';
end if;
if new.activo <> old.activo and not public.es_admin() then
    raise exception 'No tiene permisos para cambiar el estado de la cuenta';
end if;
return new;
end;
$$;

drop trigger if exists trg_usuarios_prevenir_escalada on public.usuarios;
create trigger trg_usuarios_prevenir_escalada
    before update on public.usuarios
    for each row execute function public.prevenir_escalada_privilegios();

-- 6. CATEGORIAS
create table if not exists public.categorias (
    id          uuid primary key default gen_random_uuid(),
    nombre      text not null unique,
    descripcion text,
    activo      boolean not null default true
);

insert into public.categorias (nombre, descripcion) values
    ('Infraestructura vial', 'Vías, andenes, puentes peatonales, señalización'),
    ('Espacios públicos y parques', 'Parques, plazas, zonas verdes y de
recreación'),
    ('Seguridad ciudadana', 'Iluminación, cámaras, frentes de seguridad'),
    ('Medio ambiente', 'Arborización, manejo de residuos, cuencas'),
    ('Cultura y educación', 'Bibliotecas, casas de cultura, actividades
educativas'),
    ('Servicios públicos', 'Acueducto, alcantarillado, energía, gas')
on conflict (nombre) do nothing;

-- 7. PROPUESTAS
create table if not exists public.propuestas (
    id          uuid primary key default gen_random_uuid(),
    usuario_id  uuid not null references public.usuarios(id),
    categoria_id uuid not null references public.categorias(id),
    titulo      text not null,
    descripcion text not null,

```

```
ubicacion          text,
estado             text not null default 'pendiente'
                  check (estado in
('pendiente','en_revision','aprobada','rechazada','convertida_proyecto')),
observaciones      text,
revisado_por       uuid references public.usuarios(id),
fecha_revision     timestampz,
fecha_creacion     timestampz not null default now(),
fecha_actualizacion timestampz not null default now()
);

create or replace function public.set_fecha_actualizacion()
returns trigger language plpgsql as $$
begin
    new.fecha_actualizacion = now();
    return new;
end;
$$;

drop trigger if exists trg_propuestas_actualizacion on public.propuestas;
create trigger trg_propuestas_actualizacion
    before update on public.propuestas
    for each row execute function public.set_fecha_actualizacion();

-- 8. PROPUESTA_ARCHIVOS
create table if not exists public.propuesta_archivos (
    id                uuid primary key default gen_random_uuid(),
    propuesta_id      uuid not null references public.propuestas(id) on delete
cascade,
    url               text not null,
    tipo              text,
    fecha_subida      timestampz not null default now()
);

-- 9. VOTACIONES
create table if not exists public.votaciones (
    id                uuid primary key default gen_random_uuid(),
    titulo            text not null,
    descripcion       text,
    fecha_inicio      timestampz not null,
    fecha_fin         timestampz not null,
```

```

estado      text not null default 'programada'
            check (estado in
('programada','activa','finalizada','cancelada')),
tipo        text not null default 'seleccion_multiple'
            check (tipo in ('seleccion_multiple','si_no')),
creado_por   uuid not null references public.usuarios(id),
fecha_creacion timestampz not null default now(),
            check (fecha_fin > fecha_inicio)
);

-- 10. VOTACION OPCIONES
create table if not exists public.votacion_opciones (
  id          uuid primary key default gen_random_uuid(),
  votacion_id uuid not null references public.votaciones(id) on delete cascade,
  propuesta_id uuid not null references public.propuestas(id),
  etiqueta    text not null default '',
  orden       int not null default 0,
  unique (votacion_id, propuesta_id, etiqueta)
);

-- 11. VOTO REGISTROS
create table if not exists public.voto_registros (
  id          uuid primary key default gen_random_uuid(),
  usuario_id  uuid not null references public.usuarios(id),
  votacion_id uuid not null references public.votaciones(id),
  fecha_voto  timestampz not null default now(),
  unique (usuario_id, votacion_id)
);

-- 12. VOTOS (elección anónima – sin usuario_id)
create table if not exists public.votos (
  id          uuid primary key default gen_random_uuid(),
  votacion_id uuid not null references public.votaciones(id),
  votacion_opcion_id uuid not null references public.votacion_opciones(id),
  fecha       timestampz not null default now()
);

-- Único punto de entrada para votar
create or replace function public.emitir_voto(p_votacion_id uuid, p_opcion_id
uuid)
returns void

```

```
language plpgsql
security definer set search_path = public
as $$
declare
    v_estado text;
    v_inicio timestampz;
    v_fin timestampz;
    v_valida boolean;
begin
    if auth.uid() is null then
        raise exception 'Debe iniciar sesión para votar';
    end if;

    if not public.usuario_activo() then
        raise exception 'Su cuenta está desactivada y no puede participar en
votaciones';
    end if;

    select estado, fecha_inicio, fecha_fin into v_estado, v_inicio, v_fin
        from public.votaciones where id = p_votacion_id for update;

    if not found then
        raise exception 'La votación no existe';
    end if;

    if v_estado <> 'activa' or now() < v_inicio or now() > v_fin then
        raise exception 'La votación no está activa en este momento';
    end if;

    select exists (
        select 1 from public.votacion_opciones
        where id = p_opcion_id and votacion_id = p_votacion_id
    ) into v_valida;

    if not v_valida then
        raise exception 'La opción no pertenece a esta votación';
    end if;

    insert into public.voto_registros (usuario_id, votacion_id)
    values (auth.uid(), p_votacion_id);
```

```

insert into public.votos (votacion_id, votacion_opcion_id)
values (p_votacion_id, p_opcion_id);
end;
$$;

-- Vista de resultados agregados
create or replace view public.resultados_votacion as
select
    vo.votacion_id,
    vo.id as votacion_opcion_id,
    p.titulo as propuesta_titulo,
    count(v.id) as total_votos,
    vo.etiqueta,
    case when vo.etiqueta <> '' then vo.etiqueta else p.titulo end as
opcion_titulo
from public.votacion_opciones vo
left join public.votos v on v.votacion_opcion_id = vo.id
join public.propuestas p on p.id = vo.propuesta_id
group by vo.votacion_id, vo.id, p.titulo, vo.etiqueta;

-- Función de creación de votaciones
create or replace function public.crear_votacion(
    p_titulo text,
    p_descripcion text,
    p_fecha_inicio timestamptz,
    p_fecha_fin timestamptz,
    p_tipo text,
    p_propuesta_ids uuid[]
)
returns uuid
language plpgsql
security definer set search_path = public
as $$
declare
    v_votacion_id uuid;
    v_propuesta_id uuid;
    v_orden int := 0;
begin
    if not public.es_admin() then
        raise exception 'No tiene permisos para crear votaciones';
    end if;

```

```

if p_tipo not in ('seleccion_multiple','si_no') then
    raise exception 'Tipo de votación no válido';
end if;

if p_tipo = 'seleccion_multiple' and (array_length(p_propuesta_ids,1) is null
or array_length(p_propuesta_ids,1) < 2) then
    raise exception 'Debe seleccionar al menos dos propuestas';
end if;

if p_tipo = 'si_no' and (array_length(p_propuesta_ids,1) is null or
array_length(p_propuesta_ids,1) <> 1) then
    raise exception 'Una votación de Sí/No debe tener exactamente una
propuesta';
end if;

insert into public.votaciones (titulo, descripcion, fecha_inicio, fecha_fin,
tipo, creado_por)
values (p_titulo, p_descripcion, p_fecha_inicio, p_fecha_fin, p_tipo,
auth.uid())
returning id into v_votacion_id;

if p_tipo = 'si_no' then
    v_propuesta_id := p_propuesta_ids[1];
    insert into public.votacion_opciones (votacion_id, propuesta_id, etiqueta,
orden)
values (v_votacion_id, v_propuesta_id, 'Sí', 0);
    insert into public.votacion_opciones (votacion_id, propuesta_id, etiqueta,
orden)
values (v_votacion_id, v_propuesta_id, 'No', 1);
else
    foreach v_propuesta_id in array p_propuesta_ids loop
        insert into public.votacion_opciones (votacion_id, propuesta_id, orden)
values (v_votacion_id, v_propuesta_id, v_orden);
        v_orden := v_orden + 1;
    end loop;
end if;

insert into public.auditoria (usuario_id, accion, entidad, entidad_id,
detalle)
values (auth.uid(), 'crear_votacion', 'votaciones', v_votacion_id,

```

```

        jsonb_build_object('titulo', p_titulo, 'tipo', p_tipo, 'propuestas',
p_propuesta_ids));

    return v_votacion_id;
end;
$$;

create or replace function public.actualizar_estado_votacion(p_votacion_id uuid,
p_nuevo_estado text)
returns void
language plpgsql
security definer set search_path = public
as $$
begin
    if not public.es_admin() then
        raise exception 'No tiene permisos para modificar votaciones';
    end if;

    if p_nuevo_estado not in ('activa','finalizada','cancelada') then
        raise exception 'Estado no válido';
    end if;

    update public.votaciones set estado = p_nuevo_estado where id = p_votacion_id;

    if not found then
        raise exception 'La votación no existe';
    end if;

    insert into public.auditoria (usuario_id, accion, entidad, entidad_id,
detalle)
    values (auth.uid(), 'modificar_votacion', 'votaciones', p_votacion_id,
        jsonb_build_object('nuevo_estado', p_nuevo_estado));
end;
$$;

-- Transición automática de estado según fecha
create or replace function public.actualizar_votaciones_programadas()
returns void
language plpgsql
security definer set search_path = public
as $$

```

```
begin
  update public.votaciones
  set estado = 'activa'
  where estado = 'programada'
    and now() >= fecha_inicio
    and now() < fecha_fin;

  update public.votaciones
  set estado = 'finalizada'
  where estado = 'activa'
    and now() >= fecha_fin;

  update public.votaciones
  set estado = 'finalizada'
  where estado = 'programada'
    and now() >= fecha_fin;
end;
$$;

-- 13. PROYECTOS
create table if not exists public.proyectos (
  id                uuid primary key default gen_random_uuid(),
  propuesta_id      uuid unique references public.propuestas(id),
  responsable_id    uuid not null references public.usuarios(id),
  titulo            text not null,
  descripcion        text,
  estado            text not null default 'planeado'
                  check (estado in
('planeado','en_ejecucion','finalizado','suspendido')),
  porcentaje_avance int not null default 0 check (porcentaje_avance between 0
and 100),
  fecha_inicio      date,
  fecha_fin_estimada date,
  fecha_creacion    timestamptz not null default now(),
  check (fecha_fin_estimada is null or fecha_inicio is null or
fecha_fin_estimada >= fecha_inicio)
);

-- 14. PROYECTO_ACTUALIZACIONES
create table if not exists public.proyecto_actualizaciones (
  id                uuid primary key default gen_random_uuid(),
```

```

    proyecto_id      uuid not null references public.proyectos(id) on delete
cascade,
    autor_id         uuid not null references public.usuarios(id),
    descripcion      text not null,
    porcentaje_avance int check (porcentaje_avance between 0 and 100),
    fecha            timestampz not null default now()
);

create or replace function public.actualizar_avance_proyecto()
returns trigger
language plpgsql
security definer set search_path = public
as $$
begin
    if new.porcentaje_avance is not null then
        update public.proyectos
            set porcentaje_avance = new.porcentaje_avance,
                estado = case when new.porcentaje_avance = 100 then 'finalizado' else
estado end
            where id = new.proyecto_id;
    end if;
    return new;
end;
$$;

drop trigger if exists trg_actualizar_avance on public.proyecto_actualizaciones;
create trigger trg_actualizar_avance
    after insert on public.proyecto_actualizaciones
    for each row execute function public.actualizar_avance_proyecto();

-- Conversión de propuesta a proyecto, registro de avances y cambio de estado.
create or replace function public.crear_proyecto_desde_propuesta(
    p_propuesta_id uuid,
    p_titulo text,
    p_descripcion text,
    p_fecha_inicio date default null,
    p_fecha_fin_estimada date default null
)
returns uuid
language plpgsql
security definer set search_path = public

```

```
as $$
declare
    v_proyecto_id uuid;
    v_estado_propuesta text;
begin
    if not public.es_admin() then
        raise exception 'No tiene permisos para crear proyectos';
    end if;

    select estado into v_estado_propuesta from public.propuestas where id =
p_propuesta_id;

    if v_estado_propuesta is null then
        raise exception 'La propuesta no existe';
    end if;

    if v_estado_propuesta <> 'aprobada' then
        raise exception 'Solo se puede crear un proyecto a partir de una propuesta
aprobada';
    end if;

    insert into public.proyectos (propuesta_id, responsable_id, titulo,
descripcion, fecha_inicio, fecha_fin_estimada)
    values (p_propuesta_id, auth.uid(), p_titulo, p_descripcion, p_fecha_inicio,
p_fecha_fin_estimada)
    returning id into v_proyecto_id;

    update public.propuestas set estado = 'convertida_proyecto' where id =
p_propuesta_id;

    insert into public.auditoria (usuario_id, accion, entidad, entidad_id,
detalle)
    values (auth.uid(), 'crear_proyecto', 'proyectos', v_proyecto_id,
        jsonb_build_object('propuesta_id', p_propuesta_id, 'titulo',
p_titulo));

    return v_proyecto_id;
end;
$$;

create or replace function public.registrar_avance_proyecto(
```

```
p_proyecto_id uuid,
p_descripcion text,
p_porcentaje_avance int default null
)
returns uuid
language plpgsql
security definer set search_path = public
as $$
declare
    v_actualizacion_id uuid;
    v_estado_actual text;
    v_porcentaje_actual int;
begin
    if not public.es_admin() then
        raise exception 'No tiene permisos para registrar avances';
    end if;

    select estado, porcentaje_avance into v_estado_actual, v_porcentaje_actual
    from public.proyectos where id = p_proyecto_id;

    if v_estado_actual is null then
        raise exception 'El proyecto no existe';
    end if;

    if v_estado_actual = 'finalizado' then
        raise exception 'El proyecto ya está finalizado y no admite más avances';
    end if;

    if p_porcentaje_avance is not null and (p_porcentaje_avance < 0 or
p_porcentaje_avance > 100) then
        raise exception 'El porcentaje debe estar entre 0 y 100';
    end if;

    if p_porcentaje_avance is not null and p_porcentaje_avance <
v_porcentaje_actual then
        raise exception 'El porcentaje no puede ser menor al avance actual de % por
ciento', v_porcentaje_actual;
    end if;

    insert into public.proyecto_actualizaciones (proyecto_id, autor_id,
descripcion, porcentaje_avance)
```

```
values (p_proyecto_id, auth.uid(), p_descripcion, p_porcentaje_avance)
returning id into v_actualizacion_id;
-- trg_actualizar_avance sincroniza proyectos.porcentaje_avance.

insert into public.auditoria (usuario_id, accion, entidad, entidad_id,
detalle)
values (auth.uid(), 'actualizar_proyecto', 'proyectos', p_proyecto_id,
        jsonb_build_object('actualizacion_id', v_actualizacion_id,
'porcentaje_avance', p_porcentaje_avance));

return v_actualizacion_id;
end;
$$;

create or replace function public.actualizar_estado_proyecto(p_proyecto_id uuid,
p_nuevo_estado text)
returns void
language plpgsql
security definer set search_path = public
as $$
begin
    if not public.es_admin() then
        raise exception 'No tiene permisos para modificar proyectos';
    end if;

    if p_nuevo_estado not in ('planeado','en_ejecucion','finalizado','suspendido')
then
        raise exception 'Estado no válido';
    end if;

    update public.proyectos set estado = p_nuevo_estado where id = p_proyecto_id;

    if not found then
        raise exception 'El proyecto no existe';
    end if;

    insert into public.auditoria (usuario_id, accion, entidad, entidad_id,
detalle)
values (auth.uid(), 'actualizar_proyecto', 'proyectos', p_proyecto_id,
        jsonb_build_object('nuevo_estado', p_nuevo_estado));
end;
```

```

$$;

-- 15. PROYECTO_ARCHIVOS
create table if not exists public.proyecto_archivos (
  id          uuid primary key default gen_random_uuid(),
  actualizacion_id  uuid not null references public.proyecto_actualizaciones(id)
on delete cascade,
  url         text not null,
  tipo        text,
  fecha_subida timestampz not null default now()
);

-- 16. NOTIFICACIONES
create table if not exists public.notificaciones (
  id          uuid primary key default gen_random_uuid(),
  usuario_id  uuid not null references public.usuarios(id) on delete cascade,
  titulo      text not null,
  mensaje     text not null,
  tipo        text,
  leida       boolean not null default false,
  fecha       timestampz not null default now()
);

-- Generación automática de notificaciones ante eventos del sistema
create or replace function public.notificar_cambio_estado_propuesta()
returns trigger
language plpgsql
security definer set search_path = public
as $$
begin
  if new.estado = 'aprobada' and old.estado <> 'aprobada' then
    insert into public.notificaciones (usuario_id, titulo, mensaje, tipo)
    values (new.usuario_id, 'Propuesta aprobada',
           format('Tu propuesta "%s" fue aprobada.', new.titulo),
           'propuesta_aprobada');
  elsif new.estado = 'rechazada' and old.estado <> 'rechazada' then
    insert into public.notificaciones (usuario_id, titulo, mensaje, tipo)
    values (new.usuario_id, 'Propuesta rechazada',
           format('Tu propuesta "%s" fue rechazada. Motivo: %s',
                 new.titulo, coalesce(new.observaciones, 'No especificado')),
           'propuesta_rechazada');

```

```

end if;
return new;
end;
$$;

drop trigger if exists trg_notificar_propuesta on public.propuestas;
create trigger trg_notificar_propuesta
after update on public.propuestas
for each row execute function public.notificar_cambio_estado_propuesta();

create or replace function public.notificar_cambio_estado_votacion()
returns trigger
language plpgsql
security definer set search_path = public
as $$
begin
if new.estado = 'activa' and old.estado <> 'activa' then
insert into public.notificaciones (usuario_id, titulo, mensaje, tipo)
select u.id, 'Nueva votación disponible',
format('La votación "%s" ya está abierta. ¡Participa!', new.titulo),
'votacion_activa'
from public.usuarios u
join public.roles r on r.id = u.rol_id
where r.nombre = 'ciudadano' and u.activo = true;
elsif new.estado = 'finalizada' and old.estado <> 'finalizada' then
insert into public.notificaciones (usuario_id, titulo, mensaje, tipo)
select vr.usuario_id, 'Votación finalizada',
format('La votación "%s" finalizó. Consulta los resultados.',
new.titulo),
'votacion_finalizada'
from public.voto_registros vr
where vr.votacion_id = new.id;
end if;
return new;
end;
$$;

drop trigger if exists trg_notificar_votacion on public.votaciones;
create trigger trg_notificar_votacion
after update on public.votaciones
for each row execute function public.notificar_cambio_estado_votacion();

```

```
create or replace function public.notificar_actualizacion_proyecto()
returns trigger
language plpgsql
security definer set search_path = public
as $$
declare
    v_usuario_id uuid;
    v_proyecto_titulo text;
begin
    select p.usuario_id, pr.titulo into v_usuario_id, v_proyecto_titulo
    from public.proyectos pr
    join public.propuestas p on p.id = pr.propuesta_id
    where pr.id = new.proyecto_id;

    if v_usuario_id is not null then
        insert into public.notificaciones (usuario_id, titulo, mensaje, tipo)
        values (v_usuario_id, 'Actualización de proyecto',
            format('Hay una nueva actualización en el proyecto "%s".',
v_proyecto_titulo),
            'proyecto_actualizado');
    end if;
    return new;
end;
$$;

drop trigger if exists trg_notificar_actualizacion_proyecto on
public.proyecto_actualizaciones;
create trigger trg_notificar_actualizacion_proyecto
after insert on public.proyecto_actualizaciones
for each row execute function public.notificar_actualizacion_proyecto();

-- 17. AUDITORIA
create table if not exists public.auditoria (
    id          uuid primary key default gen_random_uuid(),
    usuario_id  uuid not null references public.usuarios(id),
    accion      text not null,
    entidad     text not null,
    entidad_id  uuid,
    detalle     jsonb,
    fecha       timestampz not null default now()
```

```
);

-- 18. VISTA DE PERFILES PÚBLICOS
create or replace view public.perfiles_publicos as
select id, alias, nombre_completo
from public.usuarios
where activo = true;

-- 19. FUNCIONES DE ADMINISTRACIÓN
create or replace function public.aprobar_propuesta(p_propuesta_id uuid,
p_observaciones text default null)
returns void
language plpgsql
security definer set search_path = public
as $$
begin
    if not public.es_admin() then
        raise exception 'No tiene permisos para aprobar propuestas';
    end if;

    update public.propuestas
    set estado = 'aprobada', observaciones = p_observaciones,
        revisado_por = auth.uid(), fecha_revision = now()
    where id = p_propuesta_id;

    if not found then
        raise exception 'La propuesta no existe';
    end if;

    insert into public.auditoria (usuario_id, accion, entidad, entidad_id,
detalle)
    values (auth.uid(), 'aprobar_propuesta', 'propuestas', p_propuesta_id,
        jsonb_build_object('observaciones', p_observaciones));
end;
$$;

create or replace function public.rechazar_propuesta(p_propuesta_id uuid,
p_observaciones text)
returns void
language plpgsql
security definer set search_path = public
```

```
as $$
begin
    if not public.es_admin() then
        raise exception 'No tiene permisos para rechazar propuestas';
    end if;

    if p_observaciones is null or length(trim(p_observaciones)) = 0 then
        raise exception 'Debe indicar el motivo del rechazo';
    end if;

    update public.propuestas
    set estado = 'rechazada', observaciones = p_observaciones,
        revisado_por = auth.uid(), fecha_revision = now()
    where id = p_propuesta_id;

    if not found then
        raise exception 'La propuesta no existe';
    end if;

    insert into public.auditoria (usuario_id, accion, entidad, entidad_id,
detalle)
    values (auth.uid(), 'rechazar_propuesta', 'propuestas', p_propuesta_id,
        jsonb_build_object('observaciones', p_observaciones));
end;
$$;

create or replace function public.admin_actualizar_usuario(
    p_usuario_id uuid,
    p_rol_id uuid,
    p_activo boolean
)
returns void
language plpgsql
security definer set search_path = public
as $$
begin
    if not public.es_admin() then
        raise exception 'No tiene permisos para gestionar usuarios';
    end if;

    if p_usuario_id = auth.uid() then
```

```

        raise exception 'No puede modificar su propia cuenta desde este panel';
    end if;

    update public.usuarios set rol_id = p_rol_id, activo = p_activo where id =
p_usuario_id;

    if not found then
        raise exception 'El usuario no existe';
    end if;

    insert into public.auditoria (usuario_id, accion, entidad, entidad_id,
detalle)
    values (auth.uid(), 'gestionar_usuarios', 'usuarios', p_usuario_id,
        jsonb_build_object('nuevo_rol_id', p_rol_id, 'nuevo_activo',
p_activo));
end;
$$;

create or replace function public.obtener_metricas()
returns jsonb
language plpgsql
stable
security definer set search_path = public
as $$
declare
    v_resultado jsonb;
begin
    if not public.es_admin() then
        raise exception 'No tiene permisos para consultar métricas';
    end if;

    select jsonb_build_object(
        'usuarios_registrados', (select count(*) from public.usuarios),
        'propuestas_registradas', (select count(*) from public.propuestas),
        'propuestas_pendientes', (select count(*) from public.propuestas where
estado = 'pendiente'),
        'propuestas_en_revision', (select count(*) from public.propuestas where
estado = 'en_revision'),
        'propuestas_aprobadas', (select count(*) from public.propuestas where estado
in ('aprobada', 'convertida_proyecto')),

```

```
'propuestas_rechazadas', (select count(*) from public.propuestas where
estado = 'rechazada'),
'propuestas_convertidas', (select count(*) from public.propuestas where
estado = 'convertida_proyecto'),
'votaciones_programadas', (select count(*) from public.votaciones where
estado = 'programada'),
'votaciones_activas', (select count(*) from public.votaciones where estado =
'activa'),
'votaciones_finalizadas', (select count(*) from public.votaciones where
estado = 'finalizada'),
'votaciones_canceladas', (select count(*) from public.votaciones where
estado = 'cancelada'),
'total_votos', (select count(*) from public.votos),
'proyectos_planeados', (select count(*) from public.proyectos where estado =
'planeado'),
'proyectos_activos', (select count(*) from public.proyectos where estado =
'en_ejecucion'),
'proyectos_finalizados', (select count(*) from public.proyectos where estado
= 'finalizado'),
'proyectos_suspendidos', (select count(*) from public.proyectos where estado
= 'suspendido'),
'proyectos_por_categoria', (
select coalesce(
jsonb_agg(jsonb_build_object('categoria', c.nombre, 'total',
conteo.total) order by conteo.total desc),
'[]'::jsonb
)
from (
select p.categoria_id, count(*) as total
from public.proyectos pr
join public.propuestas p on p.id = pr.propuesta_id
group by p.categoria_id
) conteo
join public.categorias c on c.id = conteo.categoria_id
),
'nivel_participacion', (
select case
when (select count(*) from public.usuarios where activo = true) = 0 then
0
else round(
100.0 * (select count(distinct usuario_id) from public.voto_registros)
```

```

        / (select count(*) from public.usuarios where activo = true), 1
    )
    end
)
) into v_resultado;

return v_resultado;
end;
$$;

-- 20. ÍNDICES
create index if not exists idx_propuestas_usuario          on
public.propuestas(usuario_id);
create index if not exists idx_propuestas_categoria        on
public.propuestas(categoria_id);
create index if not exists idx_propuestas_estado          on
public.propuestas(estado);
create index if not exists idx_propuesta_archivos_propuesta on
public.propuesta_archivos(propuesta_id);
create index if not exists idx_votos_opcion               on
public.votos(votacion_opcion_id);
create index if not exists idx_voto_registros_votacion    on
public.voto_registros(votacion_id);
create index if not exists idx_notificaciones_usuario     on
public.notificaciones(usuario_id, leida);
create index if not exists idx_auditoria_fecha            on
public.auditoria(fecha desc);
create index if not exists idx_proyectos_estado          on
public.proyectos(estado);

-- 21. ROW LEVEL SECURITY
alter table public.roles          enable row level security;
alter table public.usuarios       enable row level security;
alter table public.categorias     enable row level security;
alter table public.propuestas     enable row level security;
alter table public.propuesta_archivos enable row level security;
alter table public.votaciones     enable row level security;
alter table public.votacion_opciones enable row level security;
alter table public.voto_registros enable row level security;
alter table public.votos          enable row level security;
alter table public.proyectos      enable row level security;

```

```
alter table public.proyecto_actualizaciones enable row level security;
alter table public.proyecto_archivos enable row level security;
alter table public.notificaciones enable row level security;
alter table public.auditoria enable row level security;

drop policy if exists roles_select on public.roles;
create policy roles_select on public.roles for select to authenticated using
(true);

drop policy if exists usuarios_select on public.usuarios;
create policy usuarios_select on public.usuarios
for select to authenticated using (id = auth.uid() or public.es_admin());

drop policy if exists usuarios_update_propio on public.usuarios;
create policy usuarios_update_propio on public.usuarios
for update to authenticated using (id = auth.uid()) with check (id =
auth.uid());

drop policy if exists usuarios_admin_todo on public.usuarios;
create policy usuarios_admin_todo on public.usuarios
for all to authenticated using (public.es_admin()) with check
(public.es_admin());

drop policy if exists categorias_select on public.categorias;
create policy categorias_select on public.categorias for select to authenticated
using (true);

drop policy if exists categorias_admin on public.categorias;
create policy categorias_admin on public.categorias
for all to authenticated using (public.es_admin()) with check
(public.es_admin());

drop policy if exists propuestas_select on public.propuestas;
create policy propuestas_select on public.propuestas
for select to authenticated
using (usuario_id = auth.uid() or estado = 'aprobada' or public.es_admin());

drop policy if exists propuestas_insert on public.propuestas;
create policy propuestas_insert on public.propuestas
for insert to authenticated
with check (usuario_id = auth.uid() and public.usuario_activo());
```

```
drop policy if exists propuestas_update_propia on public.propuestas;
create policy propuestas_update_propia on public.propuestas
  for update to authenticated
  using (usuario_id = auth.uid() and estado = 'pendiente' and
public.usuario_activo())
  with check (usuario_id = auth.uid() and estado = 'pendiente' and
public.usuario_activo());

drop policy if exists propuestas_admin on public.propuestas;
create policy propuestas_admin on public.propuestas
  for all to authenticated using (public.es_admin()) with check
(public.es_admin());

drop policy if exists propuesta_archivos_select on public.propuesta_archivos;
create policy propuesta_archivos_select on public.propuesta_archivos
  for select to authenticated
  using (exists (
  select 1 from public.propuestas p
  where p.id = propuesta_id and (p.usuario_id = auth.uid() or p.estado =
'aprobada' or public.es_admin())
));

drop policy if exists propuesta_archivos_insert on public.propuesta_archivos;
create policy propuesta_archivos_insert on public.propuesta_archivos
  for insert to authenticated
  with check (
  public.usuario_activo()
  and exists (select 1 from public.propuestas p where p.id = propuesta_id and
p.usuario_id = auth.uid() and p.estado = 'pendiente')
);

drop policy if exists propuesta_archivos_admin on public.propuesta_archivos;
create policy propuesta_archivos_admin on public.propuesta_archivos
  for all to authenticated using (public.es_admin()) with check
(public.es_admin());

drop policy if exists votaciones_select on public.votaciones;
create policy votaciones_select on public.votaciones for select to authenticated
using (true);
```

```
drop policy if exists votaciones_admin on public.votaciones;
create policy votaciones_admin on public.votaciones
  for all to authenticated using (public.es_admin()) with check
(public.es_admin());

drop policy if exists votacion_opciones_select on public.votacion_opciones;
create policy votacion_opciones_select on public.votacion_opciones for select to
authenticated using (true);

drop policy if exists votacion_opciones_admin on public.votacion_opciones;
create policy votacion_opciones_admin on public.votacion_opciones
  for all to authenticated using (public.es_admin()) with check
(public.es_admin());

drop policy if exists voto_registros_select_propio on public.voto_registros;
create policy voto_registros_select_propio on public.voto_registros
  for select to authenticated using (usuario_id = auth.uid());

drop policy if exists voto_registros_select_admin on public.voto_registros;
create policy voto_registros_select_admin on public.voto_registros
  for select to authenticated using (public.es_admin());

revoke insert, update, delete on public.votos          from authenticated;
revoke insert, update, delete on public.voto_registros from authenticated;
grant execute on function public.emitir_voto(uuid, uuid) to authenticated;
grant select on public.resultados_votacion to authenticated;
grant select on public.perfiles_publicos to authenticated;

grant execute on function public.aprobar_propuesta(uuid, text) to authenticated;
grant execute on function public.rechazar_propuesta(uuid, text) to
authenticated;
grant execute on function public.crear_votacion(text, text, timestamptz,
timestamptz, text, uuid[]) to authenticated;
grant execute on function public.actualizar_estado_votacion(uuid, text) to
authenticated;
grant execute on function public.crear_proyecto_desde_propuesta(uuid, text,
text, date, date) to authenticated;
grant execute on function public.registrar_avance_proyecto(uuid, text, int) to
authenticated;
grant execute on function public.actualizar_estado_proyecto(uuid, text) to
authenticated;
```

```
grant execute on function public.admin_actualizar_usuario(uuid, uuid, boolean)
to authenticated;
grant execute on function public.obtener_metricas() to authenticated;

drop policy if exists proyectos_select on public.proyectos;
create policy proyectos_select on public.proyectos for select to authenticated
using (true);

drop policy if exists proyectos_admin on public.proyectos;
create policy proyectos_admin on public.proyectos
  for all to authenticated using (public.es_admin()) with check
(public.es_admin());

drop policy if exists proyecto_actualizaciones_select on
public.proyecto_actualizaciones;
create policy proyecto_actualizaciones_select on public.proyecto_actualizaciones
for select to authenticated using (true);

drop policy if exists proyecto_actualizaciones_admin on
public.proyecto_actualizaciones;
create policy proyecto_actualizaciones_admin on public.proyecto_actualizaciones
  for all to authenticated using (public.es_admin()) with check
(public.es_admin());

drop policy if exists proyecto_archivos_select on public.proyecto_archivos;
create policy proyecto_archivos_select on public.proyecto_archivos for select to
authenticated using (true);

drop policy if exists proyecto_archivos_admin on public.proyecto_archivos;
create policy proyecto_archivos_admin on public.proyecto_archivos
  for all to authenticated using (public.es_admin()) with check
(public.es_admin());

drop policy if exists notificaciones_select propias on public.notificaciones;
create policy notificaciones_select propias on public.notificaciones
  for select to authenticated using (usuario_id = auth.uid());

drop policy if exists notificaciones_update propias on public.notificaciones;
create policy notificaciones_update propias on public.notificaciones
  for update to authenticated using (usuario_id = auth.uid()) with check
(usuario_id = auth.uid());
```

```
drop policy if exists notificaciones_admin on public.notificaciones;
create policy notificaciones_admin on public.notificaciones
  for all to authenticated using (public.es_admin()) with check
(public.es_admin());

drop policy if exists auditoria_select_admin on public.auditoria;
create policy auditoria_select_admin on public.auditoria for select to
authenticated using (public.es_admin());

-- 22. STORAGE – BUCKETS Y POLÍTICAS
insert into storage.buckets (id, name, public) values
('avatares', 'avatares', true) on conflict (id) do nothing;
insert into storage.buckets (id, name, public) values ('propuestas',
'propuestas', false) on conflict (id) do nothing;
insert into storage.buckets (id, name, public) values ('evidencias',
'evidencias', false) on conflict (id) do nothing;

drop policy if exists avatares_insert on storage.objects;
create policy avatares_insert on storage.objects
  for insert to authenticated
  with check (bucket_id = 'avatares' and (storage.foldername(name))[1] =
auth.uid()::text);

drop policy if exists avatares_update on storage.objects;
create policy avatares_update on storage.objects
  for update to authenticated
  using (bucket_id = 'avatares' and (storage.foldername(name))[1] =
auth.uid()::text)
  with check (bucket_id = 'avatares' and (storage.foldername(name))[1] =
auth.uid()::text);

drop policy if exists avatares_select on storage.objects;
create policy avatares_select on storage.objects
  for select to authenticated using (bucket_id = 'avatares');

drop policy if exists avatares_delete on storage.objects;
create policy avatares_delete on storage.objects
  for delete to authenticated
  using (bucket_id = 'avatares' and (storage.foldername(name))[1] =
auth.uid()::text);
```

```
drop policy if exists propuestas_files_select on storage.objects;
create policy propuestas_files_select on storage.objects
  for select to authenticated
  using (
    bucket_id = 'propuestas'
    and exists (
      select 1 from public.propuestas p
      where p.id::text = (storage.foldername(name))[2]
      and (p.usuario_id = auth.uid() or p.estado = 'aprobada' or
public.es_admin())
    )
  );

drop policy if exists propuestas_files_insert on storage.objects;
create policy propuestas_files_insert on storage.objects
  for insert to authenticated
  with check (bucket_id = 'propuestas' and (storage.foldername(name))[1] =
auth.uid()::text);

drop policy if exists propuestas_files_update on storage.objects;
create policy propuestas_files_update on storage.objects
  for update to authenticated
  using (bucket_id = 'propuestas' and (storage.foldername(name))[1] =
auth.uid()::text)
  with check (bucket_id = 'propuestas' and (storage.foldername(name))[1] =
auth.uid()::text);

drop policy if exists propuestas_files_delete on storage.objects;
create policy propuestas_files_delete on storage.objects
  for delete to authenticated
  using (bucket_id = 'propuestas' and ((storage.foldername(name))[1] =
auth.uid()::text or public.es_admin()));

drop policy if exists evidencias_select on storage.objects;
create policy evidencias_select on storage.objects
  for select to authenticated using (bucket_id = 'evidencias');

drop policy if exists evidencias_insert on storage.objects;
create policy evidencias_insert on storage.objects
  for insert to authenticated
```

```
with check (bucket_id = 'evidencias' and public.es_admin());

drop policy if exists evidencias_update on storage.objects;
create policy evidencias_update on storage.objects
  for update to authenticated
  using (bucket_id = 'evidencias' and public.es_admin())
  with check (bucket_id = 'evidencias' and public.es_admin());

drop policy if exists evidencias_delete on storage.objects;
create policy evidencias_delete on storage.objects
  for delete to authenticated
  using (bucket_id = 'evidencias' and public.es_admin());

-- 23. AUTOMATIZACIÓN (pg_cron)
select cron.schedule(
  'actualizar-votaciones-programadas',
  '* * * * *',
  $$select public.actualizar_votaciones_programadas();$$
) where not exists (
  select 1 from cron.job where jobname = 'actualizar-votaciones-programadas'
);
```

ANEXO B: Manual de Usuario

Manual elaborado en Canva, en el cual se explica el uso del sistema multiplataforma, esto con el fin de orientar al usuario de su correcta utilización, el link corresponde al alojamiento de dicho manual y será usado como complemento:

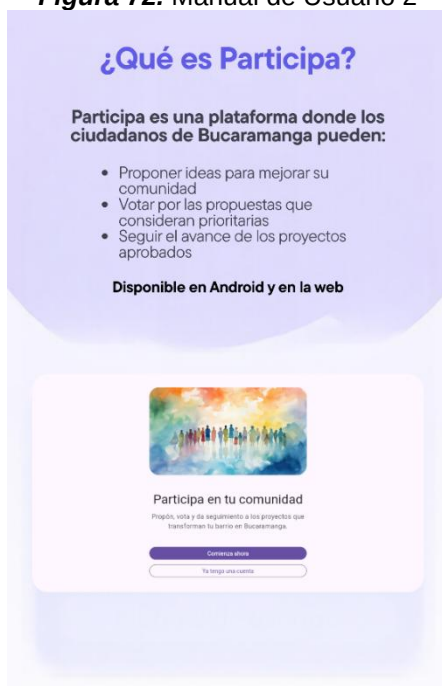
<https://drive.google.com/file/d/1nGEN2osX1vkExh7V4ZVJy56NkAGUzcvD/view?usp=sharing>

Figura 71. Manual de Usuario 1



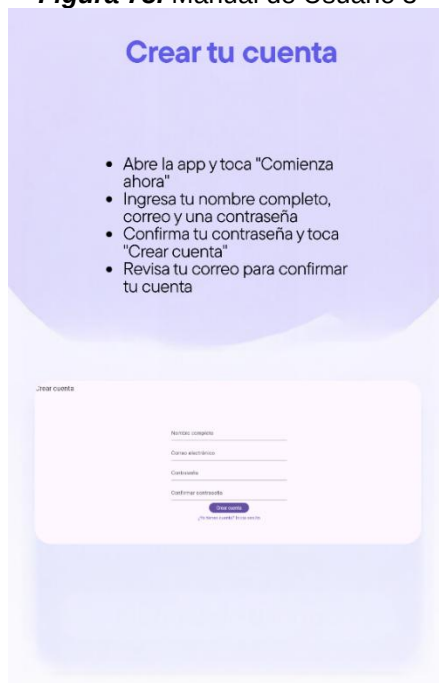
Fuente: Autor

Figura 72. Manual de Usuario 2



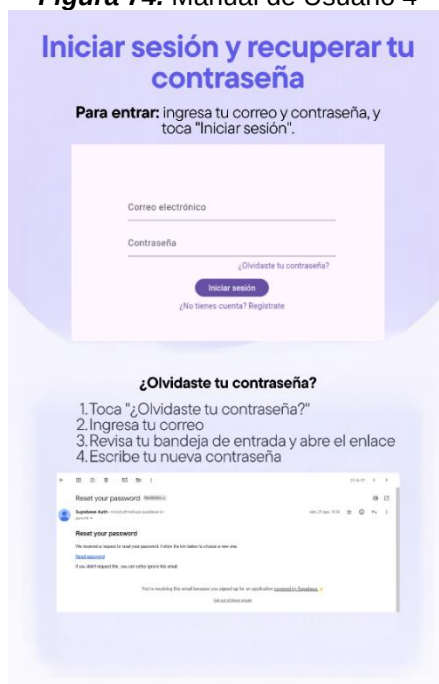
Fuente: Autor

Figura 73. Manual de Usuario 3



Fuente: Autor

Figura 74. Manual de Usuario 4



Fuente: Autor

Figura 75. Manual de Usuario 5



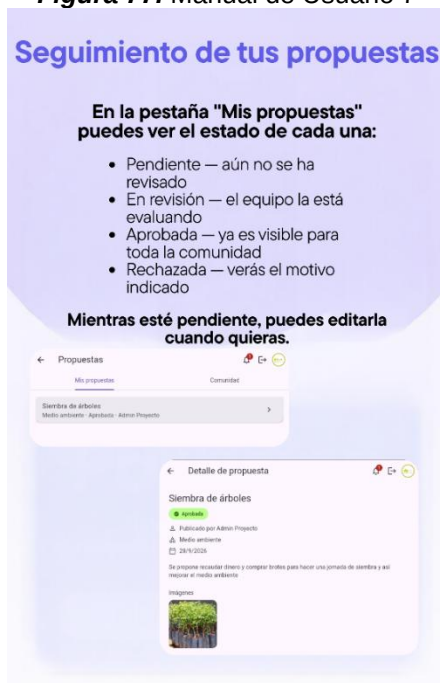
Fuente: Autor

Figura 76. Manual de Usuario 6



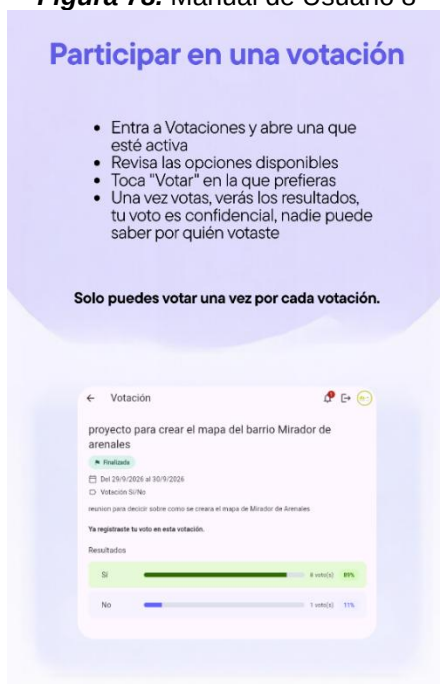
Fuente: Autor

Figura 77. Manual de Usuario 7



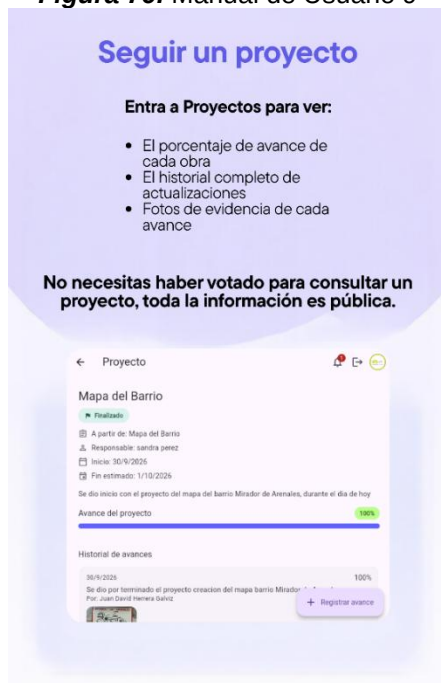
Fuente: Autor

Figura 78. Manual de Usuario 8



Fuente: Autor

Figura 79. Manual de Usuario 9



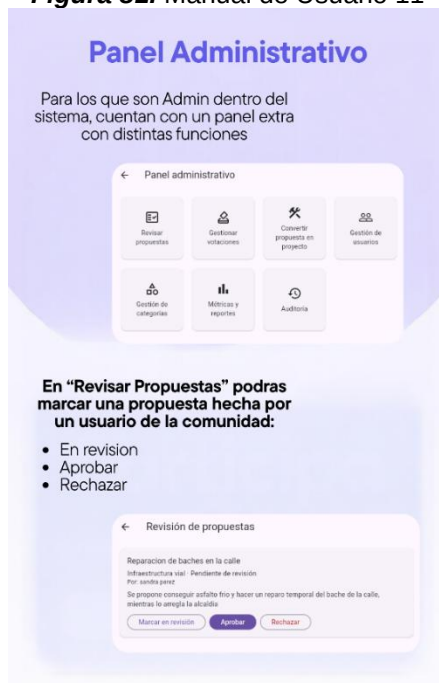
Fuente: Autor

Figura 80. Manual de Usuario 10



Fuente: Autor

Figura 81. Manual de Usuario 11



Fuente: Autor

Figura 82. Manual de Usuario 12



Fuente: Autor

Figura 83. Manual de Usuario 13



Fuente: Autor

Figura 84. Manual de Usuario 14



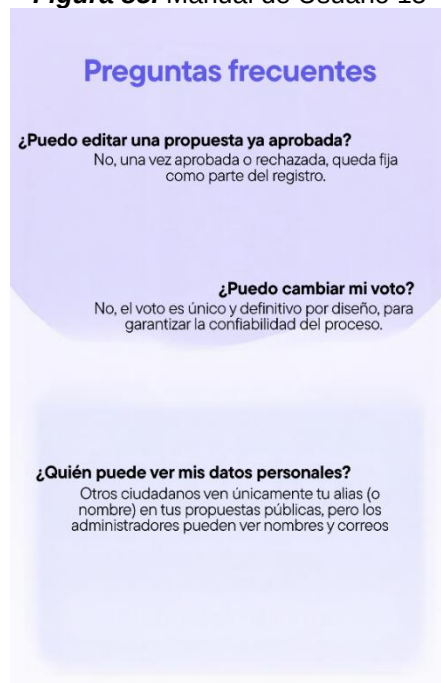
Fuente: Autor

ELABORADO POR:
Docencia

REVISADO POR:
Sistema Integrado de Gestión

APROBADO POR: Líder del Sistema Integrado de Gestión
FECHA APROBACIÓN: Octubre de 2023

Figura 85. Manual de Usuario 15



Fuente: Autor

ANEXO C: Repositorio de GitHub

Todo el código fue alojado en el GitHub personal del autor:

<https://github.com/JonatahnGonzalezCaballero/proyectoFinal.git>

ANEXO D: Consentimiento informado y la autorización de tratamiento de datos personales de los participantes del sistema

Al momento de crear una cuenta en el sistema, saltará una ventana emergente, pedirá marcar la casilla “Acepto los términos y la Política de Tratamiento de Datos Personales” el cual cada usuario puede revisar (dándole clic al link) para luego continuar con la creación de la cuenta.

Figura 86. Política de Tratamiento de Datos Personales

← Crear cuenta

Tratamiento de datos personales

Para usar Participa necesitamos tu autorización para el tratamiento de tus datos personales (nombre, correo, teléfono y la información que registres), conforme a la Ley 1581 de 2012 de Colombia.

[Ver la Política de Tratamiento de Datos Personales](#)

☒ Acepto los términos y la Política de Tratamiento de Datos Personales

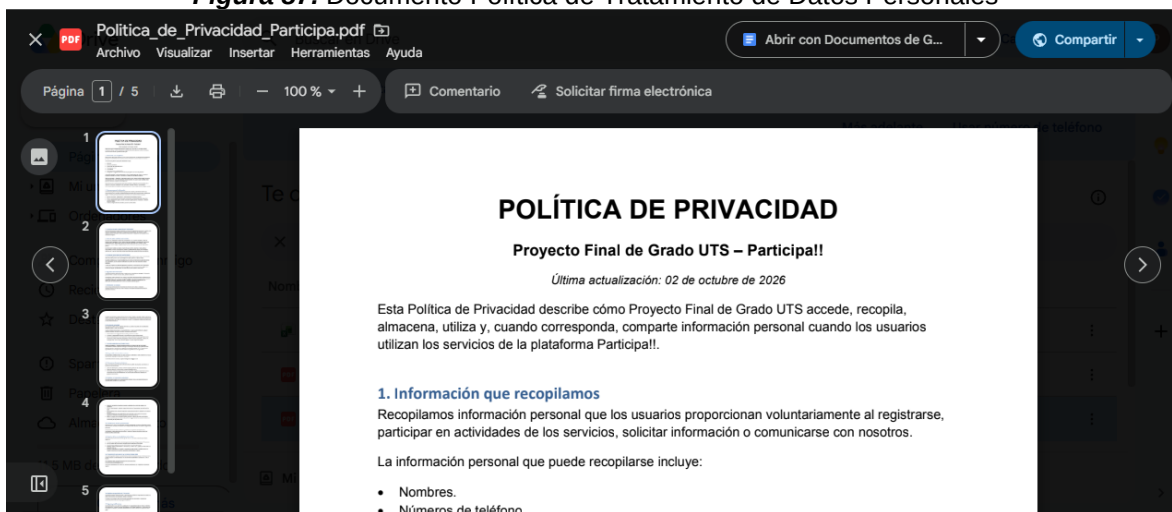
No acepto [Continuar](#)

¿Ya tienes cuenta? [Inicia sesión](#)

Fuente: Autor

El archivo se encuentra disponible y alojado en el drive del correo proyectofinaldegrado2026@gmail.com

Figura 87. Documento Política de Tratamiento de Datos Personales



Fuente: Autor

https://drive.google.com/file/d/1RLycscfrCwhHdk8q5PiUlnDoL3xOvt_G/view?usp=sharing

Figura 88. Evidencias aceptación Políticas de Tratamiento de Datos Personales

<pre> 1 select fecha_registro, fecha_aceptacion_terminos 2 from usuarios; </pre>	
Results Chart	
fecha_registro	fecha_aceptacion_terminos
2026-09-27 21:41:14.162137+00	2026-09-28 02:48:39.863504+00
2026-09-28 03:06:31.370098+00	2026-09-28 03:06:31.370098+00
2026-09-28 14:51:37.885975+00	2026-09-28 14:51:37.885975+00
2026-09-29 21:20:35.377954+00	2026-09-29 21:20:35.377954+00
2026-09-28 02:48:39.863504+00	2026-09-28 02:48:39.863504+00
2026-09-28 12:22:02.575392+00	2026-09-28 12:22:02.575392+00
2026-09-27 17:52:22.132799+00	2026-09-27 17:52:22.132799+00
2026-09-27 23:16:38.699338+00	2026-09-27 23:16:38.699338+00
2026-09-27 17:42:00.560232+00	2026-09-27 17:42:00.560232+00
2026-09-30 03:50:27.313633+00	2026-09-30 03:50:27.313633+00
2026-09-28 03:09:23.678173+00	2026-09-28 03:09:23.678173+00

Fuente: Autor

ANEXO E: Tratamiento de Datos y Fines Estadísticos de la Encuesta

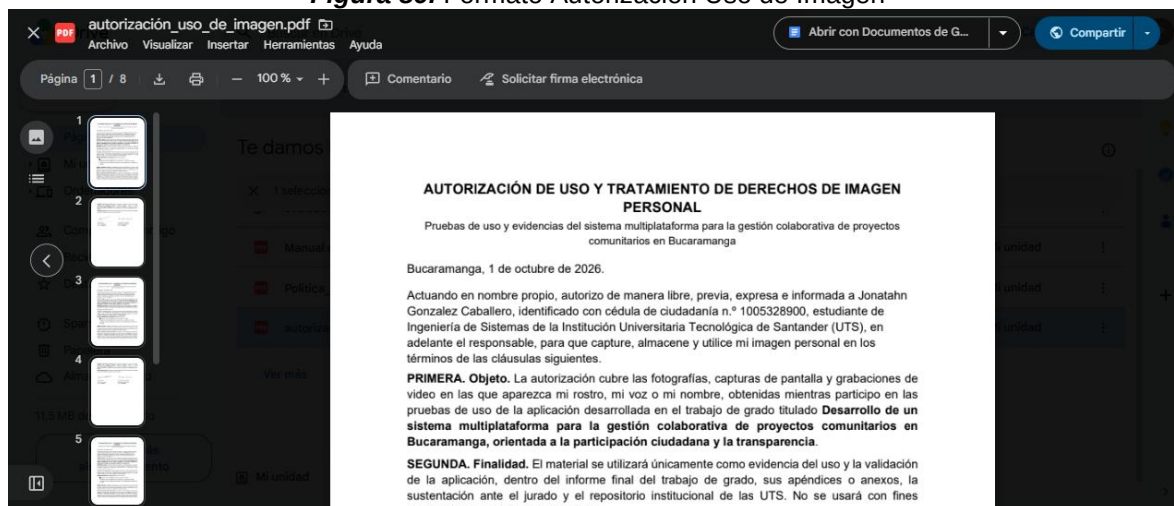
Se les informó de manera oral a los participantes de las encuestas que las respuestas enviadas son totalmente anónimas, con lo que se garantiza que no se

ha registrado ninguna información personal y que los datos obtenidos se procesarán únicamente para análisis estadístico.

ANEXO F: Autorización de uso de imagen

Para las evidencias de uso del sistema se colocaron cuatro usuarios participantes, a cada uno de ellos se les entregó de manera digital un formato de autorización de uso de imagen, el cual los cuatro usuarios firmaron; Se unió los cuatro formatos en un solo PDF y se subió al drive del correo proyectofinaldegrado2026@gmail.com

Figura 89. Formato Autorización Uso de Imagen



Fuente: Autor

<https://drive.google.com/file/d/1vdLFWxc6BpUlnVDGZHnD7ca1ql6F7Fy4/vi?usp=sharing>

ANEXO G: Script pruebas_rendimiento.mjs

```
// Pruebas de rendimiento y concurrencia - Plataforma "Participa"
//
// Mide (1) tiempos de respuesta de las operaciones principales y
// (2) el comportamiento con varios usuarios simultáneos, incluyendo
// la integridad del conteo de votos y el bloqueo del voto duplicado.
//
```

ELABORADO POR:
Docencia

REVISADO POR:
Sistema Integrado de Gestión

APROBADO POR: Líder del Sistema Integrado de Gestión
FECHA APROBACIÓN: Octubre de 2023

```
// Uso: ver variables de entorno requeridas más abajo.
// Salida: tablas en consola + resultados_rendimiento.md/.json
//
// IMPORTANTE: SUPABASE_SERVICE_ROLE_KEY se usa solo para crear y borrar
// los usuarios de prueba. Únicamente como variable de entorno de
// esta terminal; no se debe guardar en archivos ni la subir a git.

import { createClient } from '@supabase/supabase-js';
import { performance } from 'node:perf_hooks';
import { writeFileSync } from 'node:fs';

const env = process.env;
const requeridas = [
  'SUPABASE_URL', 'SUPABASE_ANON_KEY', 'SUPABASE_SERVICE_ROLE_KEY',
  'ADMIN_EMAIL', 'ADMIN_PASSWORD', 'VOTACION_QA_ID', 'OPCION_QA_ID',
];
const faltan = requeridas.filter((k) => !env[k]);
if (faltan.length) {
  console.error('Faltan variables de entorno:', faltan.join(', '));
  process.exit(1);
}

const N = Number(env.N_USUARIOS ?? 20);
const REPS = Number(env.REPETICIONES ?? 30);
const DUP = 5; // usuarios que intentan votar dos veces a la vez
if (N < DUP + 1 || N > 25) {
  console.error(`N_USUARIOS debe estar entre ${DUP + 1} y 25 (límite de ráfaga
de inicios de sesión de Supabase Auth).`);
  process.exit(1);
}

const opts = { auth: { persistSession: false, autoRefreshToken: false } };
const nuevoCliente = () => createClient(env.SUPABASE_URL, env.SUPABASE_ANON_KEY,
opts);
const servicio = createClient(env.SUPABASE_URL, env.SUPABASE_SERVICE_ROLE_KEY,
opts);

// ----- utilidades estadísticas -----
const percentil = (arr, p) => {
  const s = [...arr].sort((a, b) => a - b);
  return s[Math.max(0, Math.ceil((p / 100) * s.length) - 1)];
}
```

```
};
const resumen = (ms, errores = 0) => {
  if (!ms.length) return { n: 0, errores };
  return {
    n: ms.length,
    errores,
    media_ms: Math.round(ms.reduce((s, x) => s + x, 0) / ms.length),
    mediana_ms: Math.round(percentil(ms, 50)),
    p95_ms: Math.round(percentil(ms, 95)),
    min_ms: Math.round(Math.min(...ms)),
    max_ms: Math.round(Math.max(...ms)),
  };
};

async function cronometrar(fn) {
  const t0 = performance.now();
  try {
    const r = await fn();
    if (r?.error) throw r.error;
    return { ms: performance.now() - t0, ok: true, r };
  } catch (e) {
    return { ms: performance.now() - t0, ok: false, error: e };
  }
}

async function repetir(operacion, fn, veces = REPS) {
  const ms = [];
  let errores = 0;
  for (let i = 0; i < veces; i++) {
    const r = await cronometrar(fn);
    r.ok ? ms.push(r.ms) : errores++;
  }
  return { operacion, ...resumen(ms, errores) };
}

const revisar = (res, etiqueta) => {
  if (res?.error) console.error(`Limpieza (${etiqueta}):`, res.error.message);
};

// ----- ejecución -----
const resultados = {
```

```

fecha: new Date().toISOString(),
usuarios_simulados: N,
repeticiones: REPS,
tiempos: [],
concurrency: [],
votos: {},
};
const inicio = new Date(Date.now() - 120000).toISOString(); // margen por
diferencia de relojes
const usuarios = [];
const propuestasCreadas = [];

try {
  // Preparación: N usuarios de prueba ya confirmados (se miden sus inicios de
  sesión)
  const marca = Date.now();
  for (let i = 0; i < N; i++) {
    const email = `rendimiento_${marca}_${i}@example.com`;
    const { data, error } = await servicio.auth.admin.createUser({
      email,
      password: 'Prueba123456',
      email_confirm: true,
      user_metadata: { nombre_completo: `Prueba Rendimiento ${i}` },
    });
    if (error) throw error;
    usuarios.push({ id: data.user.id, email, cliente: nuevoCliente() });
  }

  const tiemposLogin = [];
  let erroresLogin = 0;
  for (const u of usuarios) {
    const r = await cronometrar(() =>
      u.cliente.auth.signInWithPassword({ email: u.email, password:
'Prueba123456' }));
    r.ok ? tiemposLogin.push(r.ms) : erroresLogin++;
  }
  resultados.tiempos.push({ operacion: 'Inicio de sesión',
...resumen(tiemposLogin, erroresLogin) });

  const admin = nuevoCliente();
  {

```

```
const { error } = await admin.auth.signInWithPassword({
  email: env.ADMIN_EMAIL, password: env.ADMIN_PASSWORD,
});
if (error) throw error;
}

// Parte A: tiempos de respuesta con un solo usuario
const c0 = usuarios[0].cliente;
const { data: cats, error: errCat } = await
c0.from('categorias').select('id').limit(1);
if (errCat || !cats?.length) throw errCat ?? new Error('No hay categorías
creadas');
const categoriaId = cats[0].id;

resultados.tiempos.push(await repetir('Listar propuestas de la comunidad', ()
=>
  c0.from('propuestas')
    .select('*', categorias(nombre), propuesta_archivos(id, url))
    .eq('estado', 'aprobada')
    .order('fecha_creacion', { ascending: false })));

resultados.tiempos.push(await repetir('Listar proyectos', () =>
  c0.from('proyectos')
    .select('*', propuestas(titulo), usuarios(nombre_completo))
    .order('fecha_creacion', { ascending: false })));

resultados.tiempos.push(await repetir('Consultar resultados de una votación',
() =>
  c0.from('resultados_votacion').select().eq('votacion_id',
env.VOTACION_QA_ID)));

resultados.tiempos.push(await repetir('Crear propuesta', async () => {
  const r = await c0.from('propuestas').insert({
    usuario_id: usuarios[0].id,
    categoria_id: categoriaId,
    titulo: 'Propuesta de prueba de rendimiento',
    descripcion: 'Registro generado automáticamente para medir tiempos de
respuesta.',
  }).select('id').single();
  if (r.data) propuestasCreadas.push(r.data.id);
  return r;
});
```

```

    }, Math.min(REPS, 15)));

    resultados.tiempos.push(await repetir('Consultar métricas (administrador)', ()
=>
    admin.rpc('obtener_metricas')));

    // Parte B1: lectura con usuarios simultáneos
    const niveles = [5, 10, N].filter((n, i, a) => n <= N && a.indexOf(n) === i);
    for (const nivel of niveles) {
        const ms = [];
        let errores = 0;
        const t0 = performance.now();
        for (let ronda = 0; ronda < 3; ronda++) {
            const lote = await Promise.all(usuarios.slice(0, nivel).map((u) =>
                cronometrar(() => u.cliente.from('propuestas')
                    .select('*', categorias(nombre)).eq('estado', 'aprobada'))));
            lote.forEach((r) => (r.ok ? ms.push(r.ms) : errores++));
        }
        resultados.concurrencia.push({
            escenario: `${nivel} usuarios simultáneos leyendo propuestas (3 rondas)`,
            ...resumen(ms, errores),
            duracion_total_ms: Math.round(performance.now() - t0),
        });
    }

    // Parte B2: votos simultáneos + intento de voto doble simultáneo
    const contarVotos = async () => {
        const { count, error } = await servicio.from('votos')
            .select('*', { count: 'exact', head: true })
            .eq('votacion_id', env.VOTACION_QA_ID);
        if (error) throw error;
        return count;
    };
    const votosAntes = await contarVotos();

    const t0 = performance.now();
    const llamadas = [];
    usuarios.forEach((u, i) => {
        const votar = () => cronometrar(() => u.cliente.rpc('emitir_voto', {
            p_votacion_id: env.VOTACION_QA_ID, p_opcion_id: env.OPCION_QA_ID,
        }));
    });

```

```

    llamadas.push(votar());
    if (i < DUP) llamadas.push(votar()); // segundo intento simultáneo del mismo
usuario
  });
  const lote = await Promise.all(llamadas);
  const duracion = performance.now() - t0;

  const exitosos = lote.filter((r) => r.ok);
  const rechazados = lote.filter((r) => !r.ok);
  const votosDespues = await contarVotos();
  const { count: registros } = await servicio.from('voto_registros')
    .select('*', { count: 'exact', head: true })
    .eq('votacion_id', env.VOTACION_QA_ID)
    .in('usuario_id', usuarios.map((u) => u.id));

  resultados.votos = {
    llamadas_simultaneas: lote.length,
    exitosos_esperados: N,
    exitosos_obtenidos: exitosos.length,
    rechazados_esperados: DUP,
    rechazados_obtenidos: rechazados.length,
    votos_nuevos_en_tabla: votosDespues - votosAntes,
    registros_de_participacion: registros,
    duracion_total_ms: Math.round(duracion),
    latencia_exitosos: resumen(exitosos.map((r) => r.ms)),
    mensajes_de_rechazo: [...new Set(rechazados.map((r) => r.error?.message ??
String(r.error)))],
    integridad_correcta:
      exitosos.length === N && votosDespues - votosAntes === N && registros ===
N,
  };

  reportar();
} catch (e) {
  console.error('La prueba falló:', e.message ?? e);
  process.exitCode = 1;
} finally {
  try {
    revisar(await servicio.from('votos').delete().eq('votacion_id',
env.VOTACION_QA_ID).gte('fecha', inicio), 'votos');
    if (usuarios.length) {

```

```

        revisar(await servicio.from('voto_registros').delete().in('usuario_id',
usuarios.map((u) => u.id)), 'voto_registros');
    }
    if (propuestasCreadas.length) {
        revisar(await servicio.from('propuestas').delete().in('id',
propuestasCreadas), 'propuestas');
    }
    for (const u of usuarios) {
        revisar(await servicio.auth.admin.deleteUser(u.id), 'usuario');
    }
    console.log('Limpieza finalizada: usuarios, propuestas y votos de prueba
eliminados.');
```

```

    } catch (e) {
        console.error('Error durante la limpieza, revísala manualmente:', e.message
?? e);
    }
}

function reportar() {
    console.log('\nTIEMPOS DE RESPUESTA (un usuario)');
    console.table(resultados.tiempos);
    console.log('\nCONCURRENCIA (lectura)');
    console.table(resultados.concurrencia);
    console.log('\nVOTACIÓN SIMULTÁNEA');
    console.log(resultados.votos);

    writeFileSync('resultados_rendimiento.json', JSON.stringify(resultados, null,
2));

    const v = (x) => x ?? '-';
    const fila = (r) =>
        `| ${r.operacion ?? r.escenario} | ${v(r.n)} | ${v(r.errores)} |
${v(r.media_ms)} | ${v(r.mediana_ms)} | ${v(r.p95_ms)} | ${v(r.min_ms)} |
${v(r.max_ms)} | `;
    const cabecera =
        '| Operación / escenario | n | Errores | Media (ms) | Mediana (ms) | P95
(ms) | Mín (ms) | Máx (ms) |\n|---|---|---|---|---|---|---|---|';
    const b = resultados.votos;
    const md = [
        `# Resultados de pruebas de rendimiento y concurrencia`,
        ``
    ],

```

```
`Fecha de ejecución: ${resultados.fecha}`,
`Usuarios simulados: ${N} | Repeticiones por operación: ${REPS}`,
``,
`## Tiempos de respuesta (un usuario)`,
cabecera,
...resultados.tiempos.map(fila),
``,
`## Concurrencia: lectura simultánea`,
cabecera,
...resultados.concurrencia.map(fila),
``,
`## Concurrencia: votación simultánea e intento de voto doble`,
`| Indicador | Esperado | Obtenido |`,
`|---|---|---|`,
`| Llamadas simultáneas a emitir_voto | ${N + DUP} |
${b.llamadas_simultaneas} |`,
`| Votos aceptados | ${N} | ${b.exitosos_obtenidos} |`,
`| Intentos duplicados rechazados | ${DUP} | ${b.rechazados_obtenidos} |`,
`| Filas nuevas en la tabla votos | ${N} | ${b.votos_nuevos_en_tabla} |`,
`| Registros de participación únicos | ${N} |
${b.registros_de_participacion} |`,
``,
`Duración total de la ráfaga: ${b.duracion_total_ms} ms. Integridad
correcta: ${b.integridad_correcta ? 'sí' : 'NO'}.`,
``,
].join('\n');
writeFileSync('resultados_rendimiento.md', md);
console.log('\nArchivos generados: resultados_rendimiento.md y
resultados_rendimiento.json');
}
```