

PROFESOR PATROCINANTE:

ANIBAL EDMUNDO FAUNDEZ DEL RIO

ESCUELA DE INFORMÁTICA Y TELECOMUNICACIONES

**SISTEMA DE UNIDAD TERRITORIAL
MIRADOR DE VOLCANES IV**

Proyecto Asignatura Proceso de Portafolio de Titulo

Para optar

al título de **Ingeniero en Informática**

MARIA SALDIVIA – CRISTAL HINOSTROZA – CECILIA VERA

PUERTO MONTT – CHILE

2025

DEDICATORIA

Dedicamos este trabajo a nuestras familias, quienes fueron nuestra mayor fuente de fuerza, motivación y acompañamiento durante todo este proceso. Su apoyo incondicional, su paciencia y su confianza en nosotras hicieron posible llegar hasta esta etapa.

A nuestros seres queridos que ya no están, pero cuya memoria y enseñanzas nos dieron la fuerza para continuar incluso en los momentos más difíciles. Su presencia permanece en cada logro, en cada avance y en cada paso que damos.

Y finalmente, nos lo dedicamos a nosotras mismas, por haber sido capaces de perseverar, de levantarnos pese a las dificultades y de avanzar con determinación y compromiso. Este proyecto es el reflejo de nuestro esfuerzo, resiliencia y crecimiento.

AGRADECIMIENTOS

Queremos expresar nuestro profundo agradecimiento a nuestras familias, quienes nos acompañaron en cada etapa de este proyecto con paciencia, comprensión y apoyo incondicional. Su confianza y respaldo fueron fundamentales para continuar avanzando incluso en momentos de dificultad.

De manera especial, reconocemos el compromiso del equipo de trabajo: María Saldivia, Cecilia Vera y Cristal Hinostroza, por su responsabilidad, compromiso y disposición constante. A pesar de los desafíos personales, situaciones familiares, problemas de salud y cargas emocionales enfrentadas durante el semestre, cada integrante aportó su esfuerzo y dio lo mejor de sí para lograr un trabajo cohesionado y significativo.

Agradecemos también a todas las personas del entorno cercano que, de una u otra forma, colaboraron brindando palabras de ánimo, espacios de estudio, apoyo emocional o simplemente comprensión en los momentos más demandantes del proyecto.

Finalmente, extendemos nuestro agradecimiento a la comunidad y a la Junta de Vecinos que inspiraron este proyecto, recordándonos la importancia de desarrollar tecnologías que aporten al bienestar social y a la participación ciudadana.

SUMARIO

El presente informe sintetiza el desarrollo y resultados del proyecto APT, orientado a diseñar, implementar y evaluar un sistema digital de apoyo para la gestión comunitaria de una Junta de Vecinos. A lo largo del documento se presentan los fundamentos del problema identificado, los objetivos planteados, el diseño metodológico aplicado y el proceso de implementación técnica realizado en sus distintas etapas.

El proyecto integra múltiples módulos funcionales —gestión de actividades vecinales, postulaciones, requerimientos ciudadanos, finanzas comunitarias, emisión de certificados y publicación de noticias— permitiendo mejorar la comunicación interna, la participación de los socios y la eficiencia administrativa de la directiva. El informe también recoge reflexiones del trabajo colaborativo, las limitaciones encontradas y recomendaciones para la continuidad y mejora futura del sistema.

A través de este documento se evidencia la aplicación de competencias profesionales relacionadas con análisis de requerimientos, desarrollo de software, diseño de interfaces, manejo de base de datos y trabajo en equipo, consolidando una experiencia formativa integral que contribuye al desarrollo académico y profesional de las integrantes del grupo.

ÍNDICE

DEDICATORIA	2
AGRADECIMIENTOS	3
SUMARIO	4
ÍNDICE	4
ÍNDICE DE TABLAS	6
ÍNDICE DE FIGURAS	7
ÍNDICE DE ANEXOS	8
1. ANTECEDENTES GENERALES	9
1.1. Introducción	9
1.2. Descripción del Proyecto	9
1.2.1. Tema	9
1.2.2. Áreas de Desempeño	9
1.2.3. Competencias o Unidades de Competencias	9
1.3. Fundamentación Proyecto APT	10
1.3.1. Relevancia del proyecto APT	10
1.3.2. Descripción de Proyecto APT	10
1.3.3. Pertinencia del proyecto con el perfil de egreso	11
1.3.4. Relación con los intereses profesionales	12

1.3.5. Fortalezas y debilidades para desarrollar el proyecto APT	12
1.4. Planteamiento del problema	13
1.5. Objetivos	14
1.5.1. Objetivo General	14
1.5.2. Objetivos Específicos	14
2. DISEÑO METODOLÓGICO	16
2.1. ETAPA N°1	18
2.1.1. Etapa de Inicio – Levantamiento y análisis	18
2.2. ETAPA N°2	18
2.2.1. Etapa de Diseño	18
2.2.2. Arquitectura utilizada	19
2.3. ETAPA N°3	21
2.3.1. Desarrollo – Implementación	21
2.4. ETAPA N°4	22
2.4.1. Cierre – Validación y Documentación	22
3. RESULTADOS	23
3.1. Vista Vecino	23
3.2. Vista Directiva	24
4. CONCLUSIONES Y RECOMENDACIONES	25
4.1. Conclusiones	25
4.2. Limitaciones y Recomendaciones.	26
4.2.1. Recomendaciones organizacionales y de uso	27
BIBLIOGRAFÍA	28
LINKOGRAFÍA	28
ANEXO A: Diccionario de Datos	29
ANEXO B: Requerimientos del sistema	35
ANEXO C: Endpoints principales	35
ANEXO D: Pruebas Funcionales	40

ÍNDICE DE TABLA

Tabla 1. Diseño Metodológico

14

ÍNDICE DE FIGURAS

<i>Ilustración 1. Modelo BBDD</i>	17
<i>Ilustración 2. MockUp 1</i>	19
<i>Ilustración 3. MockUp 2</i>	19
<i>Ilustración 4. MockUp 3</i>	19
<i>Ilustración 5. MockUp 4</i>	19
<i>Ilustración 6. Capturas del sistema Vecino</i>	22
<i>Ilustración 7. Capturas del sistema Directiva</i>	23

ÍNDICE DE ANEXOS

<i>Anexos 1. Diccionario de Datos</i>	28
<i>Anexos 2. Requerimientos del sistema</i>	36
<i>Anexos 3. Endpoints principales</i>	37
<i>Anexos 4. Pruebas Funcionales</i>	41

1. ANTECEDENTES GENERALES

1.1. Introducción

En la actualidad, las organizaciones utilizan herramientas tecnológicas con el propósito de mejorar su organización interna, la comunicación con sus usuarios, así como reducir tiempos, costos y uso de recursos materiales. El Estado, las municipalidades y diversas instituciones han ido incorporando progresivamente sistemas de información para apoyar sus procesos, sin embargo, las juntas de vecinos suelen quedar rezagadas en este ámbito, a pesar del rol clave que cumplen en el desarrollo comunitario.

Una junta de vecinos es una organización comunitaria de carácter territorial que representa a las personas que residen en un mismo barrio. Su objetivo principal es promover el desarrollo de la comunidad, defender los intereses de los vecinos y velar por sus derechos, actuando como puente entre la ciudadanía y las autoridades. No obstante, muchas de estas organizaciones continúan gestionando sus actividades mediante procesos manuales, usando documentos físicos, planillas y comunicaciones informales, lo que dificulta la trazabilidad, la organización y la participación de la comunidad.

En este contexto, se propone el desarrollo del Sistema Unidad Territorial, una solución de software orientada a mejorar la gestión de una junta de vecinos tipo, de manera que pueda adaptarse a la realidad de distintas unidades vecinales en Chile. El sistema busca apoyar la administración de socios, certificados, proyectos, solicitudes y comunicaciones, facilitando el trabajo del directorio y mejorando la experiencia de los vecinos que interactúan con la organización.

1.2. Descripción del Proyecto

1.2.1. Tema

Diseño e implementación de un sistema web de gestión para una unidad territorial (junta de vecinos) que permita organizar y administrar sus procesos internos y los servicios ofrecidos a la comunidad.

1.2.2. Áreas de Desempeño

El proyecto se enmarca en las siguientes áreas de desempeño:

- Análisis y evaluación de soluciones informáticas.
- Desarrollo de software.

1.2.3. Competencias o Unidades de Competencias

Las principales competencias asociadas al desarrollo del proyecto son:

- Desarrollar una solución de software utilizando técnicas que permitan sistematizar el proceso de desarrollo y mantenimiento, asegurando el logro de los objetivos.

- Construir modelos de datos para soportar los requerimientos de la organización, de acuerdo con un diseño definido y escalable en el tiempo.
- Realizar pruebas de certificación tanto de los productos como de los procesos, utilizando buenas prácticas definidas por la industria.

1.3. Fundamentación Proyecto APT

1.3.1. Relevancia del proyecto APT

El proyecto APT es relevante ya que busca fortalecer los procesos de gestión de una junta de vecinos mediante una solución tecnológica accesible y adaptable. Su desarrollo permite optimizar la organización interna, mejorar la administración de información y facilitar la comunicación con los habitantes de la unidad territorial.

La propuesta aporta valor al centrarse en una organización que realiza gran parte de sus trámites de forma manual, lo que dificulta la trazabilidad y la eficiencia en las gestiones. La creación de un sistema web contribuye a ordenar los procesos, reducir tiempos administrativos y ofrecer a la comunidad una forma más clara y directa de acceder a servicios y solicitudes.

Además, el proyecto crea una instancia formativa significativa para los estudiantes, al permitir aplicar competencias de análisis, diseño, programación y pruebas en un contexto real. Esto genera un impacto positivo tanto en el desarrollo profesional como en la modernización de una estructura comunitaria que históricamente ha contado con recursos tecnológicos limitados.

1.3.2. Descripción de Proyecto APT

El Proyecto APT consiste en el desarrollo del Sistema Unidad Territorial, una plataforma web orientada a mejorar la gestión interna de una junta de vecinos mediante la digitalización y automatización de sus procesos principales. El proyecto abarca desde el levantamiento de requerimientos y el diseño de la solución, hasta el desarrollo modular, integración, pruebas y entrega final del sistema.

El sistema incluye los módulos definidos en la planificación y validados con el cliente, los cuales responden a las necesidades operativas detectadas:

- Gestión de Vecinos: Registro, administración y actualización de la información de los habitantes inscritos en la junta de vecinos.
- Certificados de Residencia: Solicitud en línea, revisión interna y emisión del certificado correspondiente.

- Proyectos Vecinales: Postulación de proyectos comunitarios, evaluación del directorio y seguimiento del estado de cada iniciativa.
- Comunicación y Noticias: Publicación de avisos, comunicados y noticias relevantes para los habitantes de la unidad territorial.
- Administración y Seguridad: Control de usuarios internos del sistema, incluyendo roles y permisos de acceso.
- Administración de Pagos y cuotas: Registro y visualización de pagos efectuados por conceptos asociados a la junta de vecinos, cuotas, pago certificados, etc. Permitiendo mantener un control básico de ingresos y su trazabilidad.

El desarrollo del proyecto se organiza de manera progresiva, integrando actividades de análisis detallado, diseño de arquitectura, modelamiento de datos, construcción por módulos, pruebas parciales y finales, además de las acciones de capacitación y entrega. La solución final busca ser accesible, clara y usable para los miembros del directorio o personal encargado, facilitando la gestión y la comunicación con la comunidad.

1.3.3. Pertinencia del proyecto con el perfil de egreso

El desarrollo del Sistema Unidad Territorial se alinea directamente con el perfil de egreso del Ingeniero en Informática, ya que integra competencias propias del diseño, construcción, evaluación y gestión de soluciones tecnológicas orientadas a resolver necesidades reales de una organización.

El proyecto exige aplicar conocimientos avanzados en análisis de requerimientos, modelamiento de datos, diseño de arquitecturas, programación web, integración de módulos, pruebas de software y control de calidad. Asimismo, requiere considerar aspectos de seguridad, usabilidad, comunicación con usuarios y documentación técnica, todos elementos que forman parte del desempeño esperado de un Ingeniero en Informática.

Además, el proyecto promueve el trabajo en entornos colaborativos, la planificación de actividades, la toma de decisiones técnicas y la capacidad de proponer soluciones eficientes y escalables. Esto fortalece habilidades transversales propias del perfil profesional, tales como liderazgo técnico, capacidad de adaptación, pensamiento crítico y responsabilidad en el desarrollo de sistemas que impactan en un entorno real.

De esta forma, el proyecto contribuye directamente al desarrollo de las competencias necesarias para el ejercicio profesional del Ingeniero en Informática, al permitir la participación en un proceso completo de construcción de software desde la concepción de la idea hasta su implementación final.

1.3.4. Relación con los intereses profesionales

El desarrollo de este proyecto se encuentra estrechamente relacionado con los intereses profesionales del equipo, ya que permitió aplicar y fortalecer competencias vinculadas al desarrollo de software, la gestión de información y la creación de soluciones tecnológicas orientadas a necesidades reales del entorno. Como grupo, compartimos el interés por diseñar y construir aplicaciones funcionales que faciliten procesos dentro de organizaciones, y la creación del módulo para la Junta de Vecinos representó una oportunidad concreta para trabajar en un contexto similar al que enfrentaremos en el ámbito laboral.

Este proyecto nos permitió profundizar en áreas clave de nuestra formación profesional, tales como el desarrollo frontend, backend, diseño de interfaces, consumo de APIs, manejo de estados, estructuración de bases de datos, validaciones y flujos de interacción centrados en el usuario. Estos elementos coinciden directamente con nuestras proyecciones futuras, tanto en el desarrollo web como en la gestión y análisis de sistemas informáticos.

Asimismo, el trabajo colaborativo requerido —incluyendo la planificación, distribución de tareas, uso de metodologías ágiles, versionamiento y resolución conjunta de problemas— responde a intereses profesionales comunes dentro del equipo, ya que todas buscamos desempeñarnos en entornos organizados, colaborativos y orientados a resultados. El proyecto brindó un espacio idóneo para ejercitarse estas competencias, demostrando que la tecnología puede contribuir de manera significativa a mejorar procesos comunitarios y administrativos.

En conjunto, este proyecto no sólo se relaciona con los intereses individuales de cada integrante, sino también con nuestra visión grupal de convertirnos en profesionales capaces de desarrollar soluciones tecnológicas pertinentes, escalables y centradas en las personas. La experiencia reforzó nuestro compromiso con el área de desarrollo de software y confirmó que queremos seguir avanzando en este campo dentro de nuestro futuro profesional.

1.3.5. Fortalezas y debilidades para desarrollar el proyecto APT

El plan de trabajo del proyecto “Unidad Territorial” se estructuró de manera progresiva, considerando las necesidades reales de la Junta de Vecinos y las capacidades del equipo de desarrollo. En una primera etapa, el grupo realizó el levantamiento de requerimientos y análisis del contexto, definiendo claramente los módulos prioritarios: gestión de socios, postulaciones, solicitud de certificados, actividades vecinales y administración interna. Posteriormente, se desarrollaron los artefactos de planificación técnica, incluyendo la arquitectura del sistema, el diseño de la base de datos, los bocetos de

interfaz y la definición de roles. Con estos insumos, el equipo organizó el trabajo en iteraciones, distribuyendo tareas de acuerdo con las habilidades individuales y las exigencias de cada fase. Durante la implementación, se llevó a cabo un ciclo continuo de desarrollo, pruebas y ajustes, integrando feedback de la directiva para asegurar que la solución fuese funcional, accesible y alineada con el funcionamiento real de la organización vecinal. Finalmente, el plan incluyó la integración completa del sistema, la documentación técnica, el proceso de pruebas formales y la preparación del despliegue, garantizando así un cierre ordenado del proyecto y una entrega coherente con los objetivos establecidos.

1.4. Planteamiento del problema

La Junta de Vecinos constituye una organización fundamental dentro de la comunidad, encargada de canalizar necesidades, coordinar actividades y fomentar la participación ciudadana. Sin embargo, en el diagnóstico inicial realizado por el equipo, se identificó que gran parte de sus procesos de gestión se desarrollan de manera manual, desorganizada o mediante herramientas dispersas, lo que dificulta la comunicación con los socios y la administración de actividades vecinales.

Actualmente, la Junta de Vecinos enfrenta problemas en la gestión de actividades comunitarias, entre ellos:

- Falta de un sistema centralizado para administrar proyectos y actividades, generando duplicidades, pérdida de información y dificultades para realizar seguimientos.
- Limitada visibilidad para los vecinos sobre las actividades disponibles, sus características, requisitos y fechas importantes.
- Procesos de postulación manuales o informales, que impiden registrar adecuadamente el interés o participación de los socios.
- Ausencia de herramientas intuitivas que permitan a la directiva mantener informado al barrio y tomar decisiones basadas en datos reales sobre participación comunitaria.
- Baja trazabilidad y transparencia en la gestión, lo que afecta la confianza y la eficiencia interna.

Este escenario genera barreras tanto para la directiva como para los vecinos, dificultando la planificación, organización y difusión de actividades comunitarias. Por una parte, la directiva debe invertir tiempo en coordinar procesos manuales y recopilar información dispersa; por otra, los vecinos carecen de un espacio confiable donde puedan conocer, postularse o manifestar interés en las actividades disponibles.

Frente a esta situación, se identificó la necesidad de desarrollar un módulo digital de gestión de actividades vecinales, que permitiera centralizar la información, automatizar procesos clave y mejorar la comunicación entre la directiva y la comunidad. Este problema no solo afecta la operación interna de la Junta, sino que también limita la participación social y el desarrollo de iniciativas comunitarias.

En consecuencia, el desafío que se aborda en este proyecto es la falta de una plataforma tecnológica que facilite la gestión completa de proyectos vecinales, permitiendo registrar actividades, habilitar postulaciones, visualizar intereses, ordenar la información y otorgar herramientas de apoyo tanto para la directiva como para los vecinos. Este problema constituye el punto de partida para el diseño y desarrollo de la solución propuesta.

1.5. Objetivos

1.5.1. Objetivo General

Diseñar y desarrollar un módulo digital de gestión de actividades vecinales para la Junta de Vecinos, que permita centralizar la información, facilitar la comunicación entre la directiva y los socios, y optimizar los procesos de postulación, registro de interés, administración de actividades y difusión comunitaria.

1.5.2. Objetivos Específicos

- Analizar y levantar los procesos actuales relacionados con la gestión de actividades dentro de la Junta de Vecinos, identificando necesidades, limitaciones y oportunidades de mejora.
- Diseñar una interfaz amigable y accesible tanto para socios como para miembros de la directiva, que permita visualizar actividades, postularse, mostrar interés y gestionar información relevante de manera intuitiva.
- Implementar un sistema de administración de actividades, que incluya la creación, edición, eliminación y actualización de proyectos vecinales según su tipo (Municipal o JJVV).
- Desarrollar la funcionalidad de postulación para socios, permitiendo registrar y gestionar solicitudes de participación en actividades promovidas por la Junta de Vecinos.
- Implementar un módulo de registro de interés para actividades municipales, facilitando la recopilación de información sobre la comunidad interesada.

- Desarrollar un panel de gestión para la directiva, que permita visualizar postulantes, actualizar estados, revisar historial y administrar la demanda y participación vecinal.
- Garantizar la persistencia y seguridad de la información mediante la integración con una base de datos adecuada y la aplicación de buenas prácticas en la gestión de datos personales.
- Validar el funcionamiento del módulo a través de pruebas técnicas y revisión por parte de los usuarios finales (vecinos y directiva), asegurando usabilidad, eficiencia y coherencia con los objetivos del proyecto.
- Promover la participación social, ofreciendo a los vecinos una plataforma transparente, actualizada y accesible desde distintos dispositivos.

2. DISEÑO METODOLÓGICO

Tabla 1. Diseño Metodológico

DISEÑO METODOLÓGICO		
Objetivos específicos	Etapas	Actividades
<i>Nº1. Analizar y levantar los procesos relacionados con la gestión de actividades dentro de la Junta de Vecinos, identificando necesidades, limitaciones y oportunidades de mejora.</i>	Nº1 Etapa de Inicio – Levantamiento y análisis	<ul style="list-style-type: none"> → Definir alcance y objetivos del proyecto. → Configuración del repositorio y entorno. → Asignación de roles y responsabilidades del equipo. → Levantamiento preliminar de requerimientos. → Análisis del problema y diagnóstico de procesos actuales. → Selección inicial de tecnologías. → Prototipo inicial de pantallas (bocetos). → Consolidación de requisitos funcionales y no funcionales. → Levantamiento detallado de requerimientos (validado con JJVV).
<i>Nº2. Diseñar una interfaz amigable y accesible, junto con la estructura técnica del sistema y su base de datos.</i>	Nº2 Etapa de Diseño	<ul style="list-style-type: none"> → Prototipado de pantallas y validaciones internas. → Diseño de arquitectura del sistema (frontend – backend – BD). → Validación del diseño con la Junta de Vecinos. → Diseño del esquema de base de datos preliminar y final. → Selección definitiva de herramientas. → Ajustes al prototipado antes del desarrollo.

<p><i>N°3. Implementar los módulos principales: administración de actividades, postulaciones, registro de interés y panel de gestión. Garantizar persistencia y seguridad.</i></p>	<p>N°3</p>	<p>Etapa de Desarrollo – Implementación</p> <ul style="list-style-type: none"> → Implementación de base de datos (creación de tablas y relaciones). → Desarrollo del módulo Gestión de Vecinos. → Desarrollo del módulo Certificados de Residencia. → Desarrollo del módulo Proyectos Vecinales (incluye municipal y JVV). → Desarrollo del módulo de Recursos Comunitarios. → Desarrollo del módulo Comunicación y Noticias. → Desarrollo del módulo Administración y Seguridad (roles, permisos). → Desarrollo del módulo Requerimientos Externos. → Integración parcial entre módulos. → QA parcial y corrección de errores.
<p><i>N°4. Validar, probar y asegurar el correcto funcionamiento del sistema, junto con promover la participación social mediante una plataforma accesible.</i></p>	<p>N°4</p>	<p>Etapa de Cierre – Validación y Documentación</p> <ul style="list-style-type: none"> → Capacitación a usuarios (directiva y socios selectos). → Migración de datos iniciales. → Pruebas de usabilidad con usuarios simulados. → Prueba piloto y retroalimentación. → Corrección de errores detectados en QA/Piloto. → Ajustes finales. → Entrega final y presentación.

Fuente: Elaboración propia.

2.1. ETAPA N°1

2.1.1. Etapa de Inicio – Levantamiento y análisis

Esta etapa tuvo como finalidad comprender el contexto inicial del proyecto, definir su alcance y levantar la información necesaria para construir una solución pertinente para la Junta de Vecinos. Se realizaron actividades orientadas a identificar las necesidades de los usuarios (tanto socios como directiva), los procesos actuales, sus limitaciones y las oportunidades de mejora.

Durante esta fase se ejecutaron actividades clave como la definición del alcance y objetivos generales del proyecto, la configuración del repositorio y el entorno de trabajo colaborativo, así como la asignación de roles dentro del equipo. Además, se llevó a cabo un levantamiento preliminar de requerimientos, complementado con el análisis de procesos internos de la Junta de Vecinos, permitiendo establecer una visión clara de los problemas que debía abordar la solución tecnológica.

También se elaboraron prototipos iniciales de pantallas y se seleccionaron las tecnologías que posteriormente serían utilizadas en el desarrollo. Esta etapa finalizó con la consolidación de los requisitos funcionales y no funcionales, los cuales guiaron la etapa de diseño y construcción del sistema.

2.2. ETAPA N°2

2.2.1. Etapa de Diseño

Esta etapa corresponde al núcleo conceptual del proyecto, donde se estructura el funcionamiento del sistema antes de pasar al desarrollo. En esta fase se trabajó directamente en los siguientes objetivos específicos:

- Analizar y levantar los procesos actuales de la Junta de Vecinos.
- Diseñar una interfaz clara, accesible y coherente con las necesidades del usuario.
- Modelar la base de datos definitiva para garantizar integridad, relaciones y seguridad.
- Definir la arquitectura del sistema, asegurando coherencia técnica y escalabilidad.
- Establecer flujos de interacción entre usuarios, módulos y componentes del sistema.

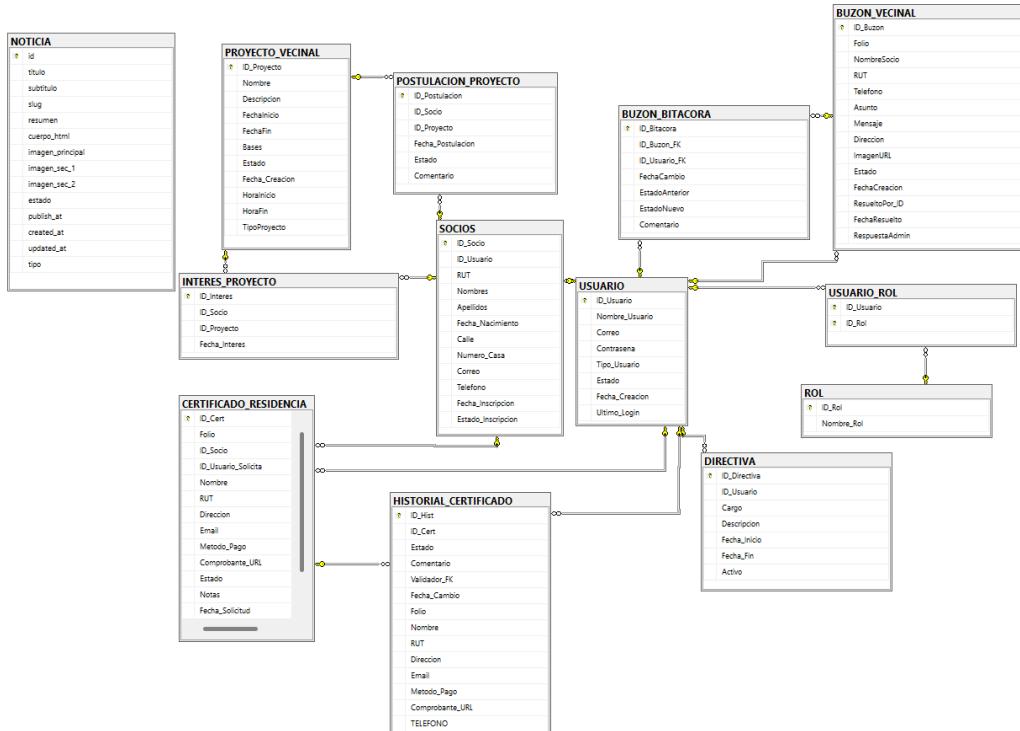


Ilustración 1. Modelo BBDD

Como parte del proceso de análisis y diseño del sistema, se construyó el Diccionario de Datos, donde se describen en detalle todas las tablas que componen la base de datos del sistema, incluyendo sus campos, tipos de datos, claves primarias, claves foráneas y relaciones internas.

El Diccionario de Datos completo se adjunta en el Anexo A de este informe

2.2.2. Arquitectura utilizada

La arquitectura del sistema se definió durante esta etapa, adoptando un modelo Cliente–Servidor con arquitectura en capas, propia de aplicaciones web modernas.

Componentes definidos:

→ Capa de Presentación (Frontend):

Construida con React, encargada de la interacción con el usuario mediante componentes reutilizables y rutas específicas para socios y directiva.

→ Capa Lógica o Servicio (Backend/API REST):

Implementada con Node.js + Express, donde se desarrollaron controladores organizados por

módulo (socios, actividades, postulaciones, certificación, buzón vecinal, noticias). Cada controlador maneja solicitudes HTTP de manera independiente y escalable.

→ Capa de Datos:

Gestionada con Microsoft SQL Server, un sistema de base de datos relacional robusto. El diseño final incluyó tablas normalizadas como: **SOCIOS**, **USUARIO**, **USUARIO_ROL**, **DIRECTIVA**, **PROYECTO_VECINAL**, **POSTULACION_PROYECTO**, **INTERES_PROYECTO**, **BUZON_VECINAL**, **CERTIFICADO_RESIDENCIA**, **HISTORIAL_CERTIFICADO**, **NOTICIA**, entre otras.

→ Modelo de Comunicación:

La comunicación entre frontend y backend se realizó mediante API REST, intercambiando información a través de solicitudes HTTP (GET, POST, PUT, DELETE) en formato JSON.

→ Seguridad y gestión de datos:

Integración de manejo de sesiones mediante token, validación de roles y almacenamiento seguro de credenciales.

Resultados de esta etapa:

- Modelo entidad–relación definitiva del sistema.
- Prototipos revisados y validados con usuarios internos (directiva).
- Arquitectura formal del sistema documentada y aplicada en el desarrollo.

- Flujos de navegación y casos de uso listos para implementación.

Ilustración 2. MockUp 1

Ilustración 3. MockUp 2

Bienvenido@ Directiva

Search Here

Nombre directiva

Hasta Socio

Bienvenido@ Directiva

Search Here

Nombre directiva

Hasta Socio

Bienvenido@ Directiva

Search Here

Nombre directiva

Hasta Socio

Ilustración 4. MockUp 3

Ilustración 5. MockUp 4

Bienvenido, para ser socio completa los siguientes datos

Al volverte socio podrás acceder a las postulaciones y podrás ingresar tus requerimientos

Nombres

Rut

Apellidos

Correo electrónico

Dirección

Comprobante de pago

Registrarme

Bienvenido, para ser socio completa los siguientes datos

Al volverte socio podrás acceder a las postulaciones y podrás ingresar tus requerimientos

Nombres

Rut

Apellidos

Correo electrónico

Dirección

Comprobante de pago

Registrarme

2.3. ETAPA N°3

2.3.1. Desarrollo – Implementación

En la etapa de desarrollo se materializó el diseño definido previamente, implementando de manera incremental los distintos módulos del sistema web de la Junta de Vecinos. A nivel de arquitectura se trabajó con una estructura por capas: una capa de presentación desarrollada en React, una capa de lógica de negocio implementada en Node.js con Express, y una capa de datos basada en SQL Server como gestor de base de datos relacional. Sobre este modelo se programaron los controladores, rutas y consultas necesarias para que cada módulo pudiera crear, leer, actualizar y eliminar información de manera consistente.

Durante esta etapa se construyeron los principales módulos funcionales. Para el rol socio, se implementó el módulo de *Requerimientos Vecinales*, que permite enviar solicitudes asociadas al entorno (seguridad, limpieza, iluminación, etc.), adjuntar evidencias y registrar automáticamente los datos personales del socio a partir de la tabla **SOCIOS**. También se desarrolló el módulo de Certificados de Residencia, que habilita a la directiva para emitir certificados, registrar su historial y mantener la trazabilidad de los documentos emitidos; y el módulo de Noticias y Comunicaciones, mediante el cual la directiva puede publicar avisos y actualizaciones visibles para los vecinos en el portal.

En paralelo se implementó el módulo de Gestión de Actividades Vecinales, que distingue entre actividades de tipo Municipal y JJVV. En este módulo se desarrolló la creación, edición y eliminación de proyectos, junto con la posibilidad de que los socios se postulen a las actividades de la Junta de Vecinos o registren interés en actividades municipales. Para la directiva, se programaron las vistas de administración de postulantes, cambio de estado (pendiente, aceptado, rechazado) y eliminación en cascada de postulaciones asociadas a un proyecto. Adicionalmente, se construyó el panel de finanzas, que permite registrar ingresos y egresos, calcular saldos y visualizar un resumen mediante tarjetas de indicadores y un gráfico de distribución de egresos por categoría, apoyando la transparencia y el control interno de la Junta.

Finalmente, durante esta etapa se integró el sistema de roles y navegación, de modo que, según el rol activo (socio, directiva o administrador), el usuario accede a menús y funcionalidades diferenciadas. Se realizaron pruebas técnicas iterativas (enrutamientos, validaciones de formularios, manejo de errores de API, formatos de fecha y moneda) y se ajustaron consultas SQL, controladores y componentes de interfaz para asegurar que el flujo completo —desde el inicio de sesión hasta la gestión de actividades, requerimientos, certificados, noticias y finanzas— funcionara de forma coherente.

Nota sobre evidencia técnica del desarrollo: Durante la Etapa N°3 se implementaron los distintos módulos funcionales del sistema, tanto en el frontend como en el backend, además de la integración con la base de datos SQL Server. Para no sobrecargar el cuerpo del informe con contenido técnico extenso, solamente se incorporaron en el Anexo C algunas capturas representativas del código de los endpoint más relevantes del backend.

2.4. ETAPA N°4

2.4.1. Cierre – Validación y Documentación

La etapa de cierre se orientó a validar el funcionamiento global del sistema y dejar evidencia formal del trabajo realizado. En primer lugar, se efectuaron pruebas integrales de extremo a extremo, verificando que las historias de usuario críticas se cumplieran: envío de requerimientos por parte de

socios y recepción en la Junta, emisión de certificados de residencia y registro en el historial, publicación y visualización de noticias, creación y administración de actividades vecinales con sus respectivas postulaciones e intereses, y registro/visualización de movimientos financieros en el panel de finanzas. Estas pruebas combinaron recorridos técnicos (revisión de respuestas HTTP, manejo de errores, integridad en base de datos) con ejercicios más cercanos al uso real por parte de la directiva y vecinos.

A partir de estas validaciones se realizaron ajustes finales: corrección de mensajes de error y confirmación, mejoras en la presentación de tablas y formularios, ajustes en algunos filtros y formatos (por ejemplo, fechas en formato local y montos en pesos chilenos), y pequeñas mejoras en la experiencia de usuario, como autocompletar datos del socio, mantener coherencia en la navegación lateral y destacar estados relevantes mediante colores y etiquetas. El objetivo fue dejar un sistema utilizable, entendible y alineado con la realidad operativa de la Junta de Vecinos.

En paralelo, se elaboró la documentación del proyecto. Esta incluyó la descripción de la arquitectura utilizada, el modelo de base de datos implementado (tablas como **SOCIOS**, **USUARIO**, **USUARIO_ROL**, **PROYECTO_VECINAL**, **POSTULACION_PROYECTO**, **REQUERIMIENTOS**, **CERTIFICADO_RESIDENCIA**, **NOTICIA**, **INTERES_PROYECTO**, entre otras), los principales endpoints de la API, y el detalle de los módulos desarrollados. Asimismo, se generó documentación orientada a usuarios no técnicos (directiva y socios), explicando de forma sencilla cómo enviar requerimientos, postular a actividades, revisar noticias, solicitar certificados y consultar la información financiera. Finalmente, se incorporó una reflexión grupal sobre los aprendizajes técnicos y profesionales obtenidos, cerrando así el ciclo metodológico del proyecto.

3. RESULTADOS

3.1. Vista Vecino

Ilustración 6.Capturas del sistema Vecino

The screenshots illustrate the following features:

- Nuestra identidad organizacional:** Shows the mission, who we are, and our vision.
- Solicita tu certificado de residencia:** A form to request a residence certificate.
- Actividades Vecinales Disponibles:** Lists available neighborhood activities like Bingo, Campeonato futsal, and Bingo bailable.
- Bienvenido:** A feedback form for reporting issues in the neighborhood.
- Solicita tu certificado de residencia:** Another view of the residence certificate request form.
- Solicitud enviada correctamente:** Confirmation message after submitting a request.
- Actividades Vecinales Disponibles:** Another view of the available neighborhood activities.
- Bienvenido:** Another view of the feedback form for reporting issues.
- Solicita tu certificado de residencia:** A third view of the residence certificate request form.

3.2. Vista Directiva

Ilustración 7. Capturas del sistema Directiva

The image displays 10 screenshots of the Directiva system interface, arranged in a grid:

- Solicitudes (Pendientes):** Shows a list of pending requests with columns for ID, Nombre, RUT, Fecha, Estado, and Acción.
- Detalles de la solicitud:** A modal showing detailed information for request C-00024, including the requester's details and a screenshot of the mobile application interface.
- Confirmar aprobación:** A confirmation dialog for approving request C-00024, with 'Aprobar' and 'Rechazar' buttons.
- Histórico de cambios:** A table showing the history of changes for two requests, with columns for ID, Nombre, RUT, Fecha, Estado, and Acciones.
- Gestión de Actividades Vecinales:** A form for adding a new community activity, including fields for activity name, date, time, and description.
- Publicación de noticias:** A form for publishing news, including fields for title, subtitle, main image, secondary image, publication date, and content.
- Panel de Finanzas:** A dashboard showing total income (\$0), total expenses (\$0), and net balance (\$0).
- Registrar Egreso:** A form for registering an expense, including fields for description, amount, category, and a 'Alta/Egreso' button.
- Socios Inscritos (6):** A table showing registered members with columns for Nombre, RUT, Dirección, Correo, and Teléfono.
- Postulantes (1):** A table showing applicants with columns for Nombre, RUT, Dirección, Correo, and Teléfono.
- Historial de Actividades:** A table showing activity history with columns for Fechas, Actividad, Estado, Tipo, and Acciones.

4. CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

El desarrollo del sistema web para la Junta de Vecinos permitió evidenciar la importancia de la tecnología como herramienta de organización comunitaria y participación ciudadana. La digitalización de procesos que antes se realizaban de manera informal —como la postulación a actividades, la emisión de certificados, el envío de requerimientos o la comunicación entre socios y directiva— generó mayor orden, trazabilidad y accesibilidad para todos los actores involucrados.

A nivel técnico, la implementación de una arquitectura basada en Node.js (backend), React (frontend) y SQL Server (base de datos) posibilitó construir un sistema modular, escalable y comprensible, donde cada funcionalidad corresponde a un módulo independiente pero conectado dentro de una misma plataforma. La distribución por roles (socio, directiva y administrador) permitió adaptar la experiencia de usuario a las necesidades reales de la institución.

El equipo desarrolló diversas capacidades durante el proyecto: levantamiento de requerimientos, diseño UI/UX, modelación de datos, programación full-stack, control de versiones, pruebas funcionales y documentación. Además, se trabajó con metodologías iterativas, lo cual permitió corregir errores sobre la marcha, adaptar funcionalidades según las pruebas realizadas y entregar un producto funcional y coherente con lo solicitado por la Junta de Vecinos.

Finalmente, el proyecto logró su objetivo principal: proporcionar a la comunidad una plataforma unificada que centraliza información, mejora la comunicación, ordena procesos internos y facilita la gestión territorial y administrativa de la Junta de Vecinos.

4.2. Limitaciones y Recomendaciones.

A lo largo del desarrollo del proyecto se identificaron diversas limitaciones que influyeron tanto en el alcance como en la velocidad de avance. La principal de ellas fue la disponibilidad de tiempo, ya que cada integrante del grupo enfrentó responsabilidades académicas, laborales y personales que dificultaron mantener un ritmo constante. A esto se sumó que, durante el semestre, algunas integrantes debieron afrontar situaciones complejas (como temas de salud mental y duelo), lo que obligó al equipo a reorganizar tareas, redistribuir esfuerzos y ajustar cronogramas. Pese a ello, la colaboración y el apoyo mutuo permitieron mantener continuidad en el trabajo y cumplir con los entregables clave.

Otra limitación relevante fue el alcance técnico del proyecto, especialmente en la integración completa de todos los módulos dentro de un mismo ecosistema. Aunque se logró desarrollar gran parte de la funcionalidad (proyectos vecinales, postulaciones, requerimientos, finanzas, comunicación, autenticación y certificaciones), el tiempo disponible restringió la inclusión de automatizaciones avanzadas (como notificaciones por correo o mensajería), optimizaciones de rendimiento o mejoras en accesibilidad para adultos mayores. Asimismo, algunas decisiones tecnológicas —como el manejo básico de sesiones o la ausencia de un sistema de permisos más complejo— se tomaron con el objetivo de priorizar la funcionalidad mínima completa (MVP) antes que la sofisticación arquitectónica.

Pese a estas limitaciones, el proyecto dejó en evidencia varias oportunidades de mejora que pueden implementarse a futuro. Entre las recomendaciones destacan:

Recomendaciones técnicas

- Fortalecer el sistema de autenticación mediante tokens más seguros, expiración controlada y permisos granulares para cada rol.
- Optimizar la base de datos mediante índices, triggers y validaciones para acompañar el crecimiento futuro del número de socios.
- Integrar notificaciones automáticas (correo o WhatsApp) para informar a los vecinos sobre estados de postulaciones, requerimientos o nuevas noticias.
- Agregar un módulo de reportes descargables en finanzas y certificaciones para profesionalizar los procesos internos.
- Mejorar la accesibilidad digital, considerando vecinos mayores o personas con dificultades visuales (tipografías más grandes, contraste, modos de lectura).

4.2.1. Recomendaciones organizacionales y de uso

- Difundir activamente la plataforma dentro de la comunidad para fomentar su uso como canal oficial de comunicación.
- Capacitar a la directiva en el uso del sistema, especialmente en módulos más administrativos como finanzas y gestión de actividades.

- Establecer responsables por módulo, para evitar confusiones y asegurar que los requerimientos se atiendan dentro de los tiempos esperados.
- Revisar y actualizar la información periódicamente, ya que la utilidad del sistema depende de que el contenido esté siempre vigente.
- Evaluar mejoras anuales, de acuerdo con la evolución de las necesidades comunitarias, permitiendo que el sistema crezca junto con la Junta de Vecinos.

BIBLIOGRAFÍA

El presente informe no requiere bibliografía externa, ya que todo el contenido fue elaborado a partir del análisis propio, la documentación del proyecto y los desarrollos realizados por el equipo.

No se utilizaron textos, artículos, páginas web ni documentos externos como referencia directa.

LINKOGRAFÍA

El presente informe no requiere linkografía externa, ya que todo el contenido fue elaborado a partir del análisis propio, la documentación del proyecto y los desarrollos realizados por el equipo.

No se utilizaron textos, artículos, páginas web ni documentos externos como referencia directa.

ANEXO A: Diccionario de Datos

Anexos 1. Diccionario de Datos

A continuación, se presenta el diccionario de datos correspondiente a la base de datos unidad_territorial, utilizada por el sistema web desarrollado para la Junta de Vecinos. Este anexo describe cada tabla, sus atributos, tipo de dato, claves y propósito dentro del sistema.

→ Tabla: USUARIO

Campo	Tipo	Clave	Descripción
ID_Usuario	INT	PK	Identificador único del usuario.
Nombre_Usuario	NVARCHAR		Nombre o alias del usuario para iniciar sesión.
Correo	NVARCHAR		Correo electrónico utilizado como credencial.
Contrasena	NVARCHAR		Contraseña cifrada del usuario.
Tipo_Usuario	NVARCHAR		Define el tipo básico de usuario (SOCIO, DIRECTIVA, ADMIN).
Estado	NVARCHAR		Estado de actividad del usuario (Activo, Inactivo).
Fecha_Creacion	DATETIME		Fecha de creación del usuario.
Ultimo_Login	DATETIME		Fecha del último inicio de sesión.

→ Tabla: ROL

Campo	Tipo	Clave	Descripción
ID_Rol	INT	PK	Identificador del rol.
Nombre_Rol	NVARCHAR	-	Nombre del rol dentro del sistema.

→ Tabla: USUARIO_ROL

Campo	Tipo	Clave	Descripción
ID_Usuario	INT	FK → USUARIO	Usuario asignado.
ID_Rol	INT	FK → ROL	Rol asociado al usuario.

→ Tabla: SOCIOS

Campo	Tipo	Clave	Descripción
ID_Socio	INT	PK	Identificador del socio.

ID_Usuario	INT	FK → USUARIO	Relación usuario ↔ socio.
RUT	VARCHAR		Identificación del vecino.
Nombres	NVARCHAR		Nombre(s) del socio.
Apellidos	NVARCHAR		Apellidos del socio.
Fecha_Nacimiento	DATE		Fecha de nacimiento.
Calle	NVARCHAR		Dirección del socio.
Numero_Casa	NVARCHAR		Número de la vivienda.
Correo	NVARCHAR		Correo del socio.
Telefono	NVARCHAR		Teléfono del socio.
Fecha_Inscripcion	DATETIME		Fecha de registro en la JJVV.
Estado_Inscripcion	NVARCHAR		Estado del proceso (Pendiente, Aprobado, Rechazado).

→ Tabla: DIRECTIVA

Campo	Tipo	Clave	Descripción
ID_Directiva	INT	PK	Identificador único.
ID_Usuario	INT	FK → USUARIO	Usuario que ocupa un cargo.
Cargo	NVARCHAR		Cargo dentro de la directiva (Presidente, Tesorero, etc.).
Descripcion	NVARCHAR		Detalles del cargo o responsabilidades.
Fecha_Inicio	DATETIME		Inicio del periodo.
Fecha_Fin	DATETIME		Fin del periodo.
Activo	BIT		Indica si el miembro sigue en funciones.

→ Tabla: PROYECTO_VECINAL

Campo	Tipo	Clave	Descripción
ID_Proyecto	INT	PK	Identificador del proyecto.

Titulo	NVARCHAR		Nombre del proyecto.
Descripcion	TEXT		Información detallada del proyecto.
FechaInicio	DATE		Fecha de inicio.
FechaFin	DATE		Fecha de término.
Bases	TEXT		Requisitos o bases para participar.
Estado	NVARCHAR		Estado (Abierto, En Revisión, Finalizado).
Fecha_Creacion	DATETIME		Fecha de creación.
HoralInicio	TIME		Horario de inicio (si aplica).
HoraFin	TIME		Horario de término.
TipoProyecto	NVARCHAR		(Municipal, JJVV).

→ Tabla: POSTULACION_PROYECTO

Campo	Tipo	Clave	Descripción
ID_Postulacion	INT	PK	Identificador de la postulación.
ID_Socio	INT	FK → SOCIOS	Socio postulante.
ID_Proyecto	INT	FK PROYECTO_VECINAL →	Proyecto al que postula.
Fecha_Postulacion	DATETIME		Fecha en que se registró.
Estado	NVARCHAR		Pendiente, Aceptado o Rechazado.
Comentario	TEXT		Motivo opcional.

→ Tabla: INTERES_PROYECTO

Campo	Tipo	Clave	Descripción
ID_Interes	INT	PK	Identificador del interés registrado.
ID_Socio	INT	FK → SOCIOS	Socio interesado.
ID_Proyecto	INT	FK PROYECTO_VECINAL →	Proyecto asociado.

Fecha_Interes	DATETIME		Fecha del registro de interés.
---------------	----------	--	--------------------------------

→ Tabla: CERTIFICADO_RESIDENCIA

Campo	Tipo	Clave	Descripción
ID_Cert	INT	PK	Identificador del certificado.
Folio	VARCHAR		Folio del certificado.
ID_Socio	INT	FK → SOCIOS	Solicitante.
ID_Usuario_Solicita	INT	FK → USUARIO	Usuario que realiza la emisión.
Nombre	NVARCHAR		Nombre del vecino.
RUT	VARCHAR		RUT del solicitante.
Direccion	NVARCHAR		Dirección del vecino.
Email	NVARCHAR		Correo del solicitante.
Metodo_Pago	NVARCHAR		Forma de pago.
Comprobante_URL	NVARCHAR		Archivo comprobante.
Estado	NVARCHAR		Estado del proceso.
Notas	TEXT		Comentarios del proceso.
Fecha_Solicitud	DATETIME		Fecha de emisión.

→ Tabla: HISTORIAL_CERTIFICADO

Campo	Tipo	Clave	Descripción
ID_Hist	INT	PK	Identificador del registro.
ID_Cert	INT	FK → CERTIFICADO_RESIDENCIA	Certificado asociado.
Estado	NVARCHAR		Estado previo o nuevo.
Comentario	TEXT		Observaciones.
Validador_FK	INT	FK → USUARIO	Usuario que validó el cambio.

Fecha_Cambio	DATETIME		Fecha de la modificación.
Folio	VARCHAR		Folio del certificado.
Nombre	NVARCHAR		Nombre del vecino.
RUT	VARCHAR		Identificación.
Direccion	NVARCHAR		Dirección.
Email	NVARCHAR		Correo.
Metodo_Pago	NVARCHAR		Método utilizado.
Comprobante_URL	NVARCHAR		Archivo comprobante.
Telefono	NVARCHAR		Número de contacto.

→ Tabla: NOTICIA

Campo	Tipo	Clave	Descripción
ID_Noticia	INT	PK	Identificador.
Titulo	NVARCHAR		Título de la noticia.
Subtitulo	NVARCHAR		Encabezado secundario.
Slug	VARCHAR		Ruta amigable para URL.
Resumen	NVARCHAR		Breve resumen.
Cuerpo_HTML	TEXT		Contenido principal.
Imagen_Principal	NVARCHAR		URL de la imagen principal.
Imagen_Sec_1	NVARCHAR		Imagen adicional 1.
Imagen_Sec_2	NVARCHAR		Imagen adicional 2.
Estado	NVARCHAR		Publicado, Borrador, Archivado.
Publish_at	DATETIME		Fecha de publicación.
Created_at	DATETIME		Creación.

Updated_at	DATETIME		Última modificación.
Tipo	NVARCHAR		(Aviso urgente, Proyecto, Noticia).

→ Tabla: BUZON_VECINAL

Campo	Tipo	Clave	Descripción
ID_Buzon	INT	PK	Identificador del aviso.
Folio	VARCHAR		Folio del requerimiento.
NombreSocio	NVARCHAR		Nombre del socio.
RUT	VARCHAR		Identificación.
Telefono	NVARCHAR		Número de contacto.
Asunto	NVARCHAR		Categoría del requerimiento.
Direccion	NVARCHAR		Ubicación del caso.
ImagenURL	NVARCHAR		Archivo adjunto.
Estado	NVARCHAR		Pendiente, Resuelto, Archivado.
FechaCreacion	DATETIME		Fecha del aviso.
ResueltoPor_ID	INT	FK → USUARIO	Usuario que lo resolvió.
FechaResuelto	DATETIME		Fecha de cierre.
RespuestaAdmin	TEXT		Respuesta de la directiva.

→ Tabla: BUZON_BITACORA

Campo	Tipo	Clave	Descripción
ID_Bitacora	INT	PK	Identificador.
ID_Buzon_FK	INT	FK → BUZON_VECINAL	Aviso asociado.
ID_Usuario_FK	INT	FK → USUARIO	Usuario que modifica.
FechaCambio	DATETIME		Fecha del cambio.
EstadoAnterior	NVARCHAR		Estado previo.
EstadoNuevo	NVARCHAR		Estado nuevo.

Comentario	TEXT		Justificación.
------------	------	--	----------------

→ Tabla: NOTIFICACION

Campo	Tipo	Clave
ID_Noticacion	INT	PK
ID_Usuario	INT	FK
Contenido	TEXT	
Medio	NVARCHAR	– (Email, WhatsApp, SMS)
Modulo_Origen	NVARCHAR	
Fecha_Envio	DATETIME	

ANEXO B: Requerimientos del sistema

Anexos 2. Requerimientos del sistema

[Matriz de Requerimientos.xlsx](#)

ANEXO C: Endpoints principales

Anexos 3.Endpoints principales

→ Gestión de roles:

```
//Aproueba un postulante cambiando su estado a 'Activo'.
export const aprobarSocio = async (req, res) => {
  const { idSocio } = req.params;
  try {
    const pool = await getPool();
    await pool.request()
      .input("idSocio", sql.Int, idSocio)
      .query(`
        UPDATE dbo.SOCIOS
        SET Estado_Inscripcion = 'Aprobado'
        WHERE ID_Socio = @idSocio
      `);
    res.status(200).json({ ok: true, message: "Socio aprobado exitosamente." });
  } catch (error) {
    console.error("Error al aprobar socio:", error);
    res.status(500).json({ ok: false, message: "Error al aprobar socio", error: error.message });
  }
};

//Rechaza (elimina) un postulante.
export const rechazarSocio = async (req, res) => {
  const { idSocio } = req.params;
  try {
    const pool = await getPool();
    await pool.request()
      .input("idSocio", sql.Int, idSocio)
      .query(`
        UPDATE dbo.SOCIOS
        SET Estado_Inscripcion = 'Rechazado'
        WHERE ID_Socio = @idSocio
      `);
    res.status(200).json({ ok: true, message: "Socio rechazado exitosamente." });
  } catch (error) {
    console.error("Error al rechazar socio:", error);
    res.status(500).json({ ok: false, message: "Error al rechazar socio", error: error.message });
  }
};
```

```

/* ELECCIÓN DE ROL */
export const chooseRole = async (req, res) => {
  const { usuarioId, rolElegido } = req.body || {};
  try {
    if (!usuarioId || !rolElegido) {
      return res.status(400).json({ ok: false, error: "Faltan datos" });
    }

    const rol = String(rolElegido).toUpperCase();
    const pool = await getPool();

    // Validar que ese rol esté asociado al usuario (sin "Cargo")
    const v = await pool
      .request()
      .input("idU", sql.Int, usuarioId)
      .input("rol", sql.NVarChar, rol).query(`
        SELECT TOP 1 u.ID_Usuario, u.Nombre_Usuario, u.Correo
        FROM dbo.USUARIO u
        WHERE u.ID_Usuario = @idU
        AND EXISTS (
          SELECT 1
          FROM dbo.USUARIO_ROL ur
          JOIN dbo.ROL r ON r.ID_Rol = ur.ID_Rol
          WHERE ur.ID_Usuario = u.ID_Usuario
          AND UPPER(r.Nombre_Rol) = @rol
        )
      `);

    if (v.recordset.length === 0) {
      return res
        .status(403)
        .json({ ok: false, error: "Rol no asignado al usuario" });
    }

    const u = v.recordset[0];
    const token = issueToken({
      sub: u.ID_Usuario,
      nombre: u.Nombre_Usuario,
      tipo_usuario: rol,
      roles: [rol],
    });

    return res.status(200).json({ ok: true, rol, token });
  } catch (err) {
    const full = JSON.stringify(err, Object.getOwnPropertyNames(err), 2);
    console.error(`[choose-role] error FULL: ${full}`);
    return res.status(500).json({ ok: false, error: "Error al elegir rol" });
  }
};

```

→ Endpoint de actividades:

```

// Crear una actividad
export const crearProyecto = async (req, res) => {
  console.log('Datos recibidos del frontend:', req.body);

  const {
    Nombre,
    Descripcion,
    Bases,
    FechaInicio,
    FechaFin,
    HoraInicio,
    HoraFin,
    TipoProyecto,
  } = req.body;

  try {
    if (!Nombre || !Descripcion || !FechaInicio || !FechaFin) {
      return res
        .status(400)
        .json({ error: "Faltan campos obligatorios (Nombre, Descripción, Fechas)" });
    }

    const pool = await getPool();

    const horaInicioValida = HoraInicio
      ? `${HoraInicio.length === 5 ? HoraInicio + ":00" : HoraInicio}`
      : null;

    const horaFinValida = HoraFin
      ? `${HoraFin.length === 5 ? HoraFin + ":00" : HoraFin}`
      : null;

    const fechaActual = new Date();
    const fechaFinProyecto = new Date(FechaFin);
    const estadoInicial =
      fechaFinProyecto < fechaActual ? "Finalizado" : "Abierto";

    console.log(`\nHoras normalizadas:`, [
      HoraInicio,
      HoraFin,
      horaInicioValida,
      horaFinValida,
    ]);

    const request = pool.request();
    request.input("Nombre", sql.NVarChar(128), Nombre);
    request.input("Descripcion", sql.NVarChar(400), Descripcion);
    request.input("Bases", sql.NVarChar(400), Bases || null);
    request.input("FechaInicio", sql.Date, FechaInicio);
    request.input("FechaFin", sql.Date, FechaFin);
    request.input("HoraInicio", sql.NVarChar(8), horaInicioValida);
    request.input("HoraFin", sql.NVarChar(8), horaFinValida);
    request.input("Estado", sql.NVarChar(20), estadoInicial);
    request.input("TipoProyecto", sql.NVarChar(30), TipoProyecto || "JWV");

    await request.query(
      `INSERT INTO PROYECTO_VECTINAL
      (Nombre, Descripcion, Bases, FechaInicio, FechaFin, HoraInicio, HoraFin, Estado, TipoProyecto)
      VALUES
      (@Nombre, @Descripcion, @Bases, @FechaInicio, @FechaFin, @HoraInicio, @HoraFin, @Estado, @TipoProyecto)
      `);

    console.log("Proyecto creado correctamente");
    res.status(201).json({ message: "Proyecto creado correctamente" });
  } catch (err) {
    console.error(` Error al crear proyecto: ${err}`);
    res.status(500).json({ error: err.message });
  }
};

//Obtener todos los proyectos
export const obtenerProyectos = async (req, res) => {
  try {
    const pool = await getPool();
    await pool.request().query(`
      UPDATE PROYECTO_VECTINAL
      SET Estado = 'Finalizado'
      WHERE FechaFin < GETDATE() AND Estado <> 'Finalizado';
    `);

    const result = await pool.request().query(`
      SELECT
        P.ID_Proyecto,
        P.Nombre,
        P.Descripcion,
        P.Bases,
        P.FechaInicio,
        P.FechaFin,
        CONVERT(VARCHAR(8), P.HoraInicio, 108) AS HoraInicio,
        CONVERT(VARCHAR(8), P.HoraFin, 108) AS HoraFin,
        P.Estado,
        P.Fecha_Creacion,
        P.TipoProyecto,
        -- Contador de interesados
        (SELECT COUNT(*)
         FROM INTERES_PROYECTO I
         WHERE I.ID_Proyecto = P.ID_Proyecto
        ) AS TotalInteres
      FROM PROYECTO_VECTINAL P
      ORDER BY P.FechaInicio ASC;
    `);

    res.json(result.recordset);
  } catch (err) {
    console.error(` Error al obtener proyectos: ${err}`);
    res.status(500).json({ error: "Error al obtener proyectos" });
  }
};

//Actualizar proyecto (fecha, hora, estado o bases)
export const actualizarProyecto = async (req, res) => {
  const { id } = req.params;
  const { FechaInicio, FechaFin, HoraInicio, HoraFin, Estado, Bases } = req.body;

  try {
    const pool = await getPool();
    const horaInicioNorm = normalizarHora(HoraInicio);
    const horaFinNorm = normalizarHora(HoraFin);

    const request = pool.request()
      .input("ID_Proyecto", sql.Int, id)
      .input("FechaInicio", sql.Date, FechaInicio || null)
      .input("FechaFin", sql.Date, FechaFin || null)
      .input("HoraInicio", sql.NVarChar(8), horaInicioNorm)
      .input("HoraFin", sql.NVarChar(8), horaFinNorm)
      .input("Estado", sql.NVarChar(20), Estado || null)
      .input("Bases", sql.NVarChar(400), Bases || null);

    await request.query(
      `UPDATE PROYECTO_VECTINAL
      SET
        FechaInicio = ISNULL(@FechaInicio, FechaInicio),
        FechaFin = ISNULL(@FechaFin, FechaFin),
        HoraInicio = ISNULL(@HoraInicio, HoraInicio),
        HoraFin = ISNULL(@HoraFin, HoraFin),
        Estado = ISNULL(@Estado, Estado),
        Bases = ISNULL(@Bases, Bases)
      WHERE ID_Proyecto = @ID_Proyecto
    `);

    res.json({ message: "Proyecto actualizado correctamente" });
  }
};

```

→ Endpoint de certificados:

```

async function enviarCorreoCertificadoBasico(cert, pdfBuffer, pdfFileName) {
  if (!cert?.Email) {
    console.warn(` No hay correo para este certificado, no se envia email.`);
    return { sent: false, reason: "NO_EMAIL" };
  }

  const destinatario = cert.Email;
  const nombre = cert.Nombre || "";
  const folio = cert.Folio || "";
  const rut = cert.RUT || "";
  const direccion = cert.Direccion || "";
  const fechaKey = new Date().toLocaleDateString("es-CL");

  const mailOptions = {
    from: `*Junta de Vecinos* <${process.env.MAIL_USER}>`,
    to: destinatario,
    subject: `Nuevo certificado de residencia para ${nombre} ${folio} ${rut}`,
    html: `

Este es tu certificado de residencia. Puedes imprimirlo y presentarlo en la junta de vecinos.



Nombre: ${nombre}



Folio: ${folio}



RUT: ${rut}



Dirección: ${direccion}



Fecha: ${fechaKey}



Este certificado es válido por un año.



Atentamente,



Junta de Vecinos

`,
    attachments: [pdfBuffer]
  };
}

```

→ Endpoint de finanzas:

```
//POST Crea un nuevo movimiento (Ingreso o Egreso)
export const crearMovimiento = async (req, res) => {
  // Obtenemos los datos del formulario (fronted)
  const { Tipo, Monto, Descripcion, Categoria, ID_Socio_FK } = req.body;
  const ID_Dire_FK = req.user.ID_Usuario || 1;

  // Validación simple
  if (!Tipo || !Monto || !Descripcion || !Categoria) {
    return res.status(400).json({ error: 'Faltan campos obligatorios' });
  }

  try {
    const pool = await getPool();
    await pool.request();
    .input('Tipo', sql.VarChar(10), Tipo)
    .input('Monto', sql.Decimal(10, 2), Monto)
    .input('Descripcion', sql.VarChar(255), Descripcion)
    .input('Categoria', sql.VarChar(50), Categoria)
    .input('ID_Socio_FK', sql.Int, ID_Socio_FK || null)
    .input('ID_Dire_FK', sql.Int, ID_Dire_FK);

    query = `INSERT INTO MOVIMIENTOS (Tipo, Monto, Descripcion, Categoria, ID_Socio_FK, ID_Dire_FK)
VALUES (@Tipo, @Monto, @Descripcion, @Categoria, @ID_Socio_FK, @ID_Dire_FK)`;
    res.status(201).json({ message: 'Movimiento registrado con éxito' });

  } catch (err) {
    console.error(`Error al crear movimiento: ${err}`);
    res.status(500).json({ error: err.message });
  }
};

//GET /Devuelve los totales (Saldo, Ingresos, Egresos)
export const obtenerDashboard = async (req, res) => {
  try {
    const pool = await getPool();
    const result = await pool.request()
      .query(`
SELECT
  -- 1. Saldo Neto (Ingresos - Egresos)
  ISNULL(SUM(CASE WHEN Tipo = 'Ingreso' THEN Monto ELSE -Monto END), 0) AS SaldoNeto,
  -- 2. Total de Ingresos (todo lo que entró)
  ISNULL(SUM(CASE WHEN Tipo = 'Ingreso' THEN Monto ELSE 0 END), 0) AS TotalIngresos,
  -- 3. Total de Egresos (todo lo que salió)
  ISNULL(SUM(CASE WHEN Tipo = 'Egreso' THEN Monto ELSE 0 END), 0) AS TotalEgresos
FROM MOVIMIENTOS
`);

    res.json(result.recordset[0]);
  } catch (err) {
    console.error(`Error al obtener dashboard: ${err}`);
    res.status(500).json({ error: err.message });
  }
};

//GET Devuelve una lista de los últimos 50 movimientos
export const obtenerMovimientos = async (req, res) => {
  try {
    const pool = await getPool();
    const result = await pool.request()
      .query(`
SELECT TOP 50
  M.ID_Movimiento,
  M.Tipo,
  M.Monto,
  M.Descripcion,
  M.Categoría,
  M.Fecha,
  U.Nombre_Usuario AS RegistradoPor, -- El admin que lo registró
  S.Nombres AS Socio -- El socio que pagó (si aplica)
FROM MOVIMIENTOS M
JOIN USUARIO U ON M.ID_Dire_FK = U.ID_Usuario
LEFT JOIN SOCIOS S ON M.ID_Socio_FK = S.ID_Socio
ORDER BY M.Fecha DESC
`);

    res.json(result.recordset);
  } catch (err) {
    console.error(`Error al obtener movimientos: ${err}`);
    res.status(500).json({ error: err.message });
  }
};
```

→ Endpoint de requerimientos:

```

//CREAR NUEVO REQUERIMIENTO (POST)
export const crearMensajeBuzon = async (req, res) => {
  const {
    nombre,
    socioNombre,
    rut_Socio,
    telefono,
    tipo,
    comentarlos,
    direccion,
  } = req.body;

  constImagenURL = req.file ? publicUrlFor(req.file.path) : null;
  const nombre_socio = socioNombre;
  const rut = rut_Socio;
  const asunto = tipo;
  const mensaje = comentarlos || "";

  if (!nombre || !asunto || !mensaje) {
    return res.status(400).json({ ok: false, message: "Faltan campos obligatorios." });
  }

  const pool = await getPool();
  const tx = new sql.Transaction(pool);

  try {
    await tx.begin();

    // Inserta en la tabla principal
    const reqBuzon = new sql.Request(tx);
    reqBuzon.input("NombreSocio", sql.VarChar(128), nombre);
    reqBuzon.input("RUT", sql.VarChar(12), rut || null);
    reqBuzon.input("Telefono", sql.VarChar(255), telefono || null);
    reqBuzon.input("Asunto", sql.NVarChar(200), asunto);
    reqBuzon.input("Mensaje", sql.NVarChar(sql.MAX), mensaje);
    reqBuzon.input("Direccion", sql.NVarChar(255), direccion || null);
    reqBuzon.input("ImagenURL", sql.NVarChar(400), imagenURL || null);

    const resultBuzon = await reqBuzon.query(
      `INSERT INTO dbo.BUZON_VECINAL
      (@NombreSocio, @RUT, @Telefono, @Asunto, @Mensaje, @Direccion, @ImagenURL, Estado)
      OUTPUT inserted.ID_Buzon, inserted.Folio, inserted.FechaCreacion, inserted.Estado
      VALUES
      (@NombreSocio, @RUT, @Telefono, @Asunto, @Mensaje, @Direccion, @ImagenURL, 'Pendiente')`;
    );

    const nuevoTicket = resultBuzon.recordset[0];

    // Inserta el primer registro en la Bitácora
    const reqBitacora = new sql.Request(tx);
    reqBitacora.input("ID_Buzon", sql.Int, nuevoTicket.ID_Buzon);
    reqBitacora.input("ID_Usuario", sql.Int, null);
    reqBitacora.input("EstadoAnterior", sql.VarChar(20), null);
    reqBitacora.input("EstadoNuevo", sql.VarChar(20), nuevoTicket.Estado);
    reqBitacora.input("Comentario", sql.NVarChar(1000), "Requerimiento a aviso creado por vecino.");

    await reqBitacora.query(
      `INSERT INTO dbo.BUZON_BITACORA
      (@ID_Buzon_FK, ID_Usuario_FK, EstadoAnterior, EstadoNuevo, Comentario)
      VALUES
      (@ID_Buzon_FK, @ID_Usuario_FK, @EstadoAnterior, @EstadoNuevo, @Comentario)`);

    await tx.commit();
    res.status(201).json({ ok: true, data: {
      Folio: nuevoTicket.Folio,
      Adjunto_URL: imagenURL
    } });
  } catch (error) {
    await tx.rollback();
    console.error("Error al crear mensaje de buzón:", error);
    res.status(500).json({ ok: false, message: "Error al guardar el mensaje", error: error.message });
  }
};

//LISTAR MENSAJES (GET)
export const listarMensajeBuzon = async (req, res) => {
  const { estando } = req.query;
  const pool = await getPool();
  const request = pool.request();
  let query = `SELECT * FROM dbo.BUZON_VECINAL`;
  const estadosValidos = ['Pendiente', 'En Resuelto'];
  if (estando && estadosValidos.includes(estando)) {
    query += ` WHERE Estado = @Estado`;
    request.input("Estado", sql.VarChar(20), estando);
  }
  query += ` ORDER BY FechaCreacion DESC`;

  const result = await request.query(query);
  res.status(200).json({ ok: true, data: result.recordset });
} catch (error) {
  console.error("Error al listar mensajes:", error);
  res.status(500).json({ ok: false, message: "Error al obtener los mensajes", error: error.message });
}

//CAMBIAR ESTADO (Función genérica para "En Revisión" o "Resuelto")
export const cambiarEstadoBuzon = async (req, res) => {
  const { id } = req.params; // ID_Buzon
  const { estandoNuevo, respuestaAdmin, idAdmin } = req.body;
  // Validación
  const estadosValidos = ['En Revisión', 'Resuelto'];
  if (estadosValidos.includes(estandoNuevo)) {
    return res.status(400).json({ ok: false, message: "Estado nuevo no válido." });
  }
  if (estandoNuevo === 'Resuelto' && !respuestaAdmin) {
    return res.status(400).json({ ok: false, message: "Se requiere un comentario para resolver el ticket." });
  }
  if (!idAdmin) {
    return res.status(400).json({ ok: false, message: "Se requiere un ID de administrador." });
  }
  const pool = await getPool();
  const tx = new sql.Transaction(pool);

  try {
    await tx.begin();
    // Obtiene el estado anterior
    const reqEstado = new sql.Request(tx);
    reqEstado.input("ID_Buzon", sql.Int, id);
    const estandoActual = await reqEstado.query(`SELECT Estado FROM dbo.BUZON_VECINAL WHERE ID_Buzon = @ID_Buzon`);

    if (estandoActual.recordset.length === 0) {
      throw new Error("Ticket no encontrado.");
    }
    const estandoAnterior = estandoActual.recordset[0].Estado;

    if (estandoAnterior === 'Resuelto') {
      throw new Error("El ticket ya está resuelto.");
    }
    // Actualizar la tabla principal
    const reqUpdate = new sql.Request(tx);
    reqUpdate.input("ID_Buzon", sql.Int, id);
    reqUpdate.input("Estado", sql.VarChar(20), estandoNuevo);
    reqUpdate.input("FechaResuelto", sql.DateTime, sysdate());
    reqUpdate.input("EstadoNuevo", sql.VarChar(20), respuestaAdmin);
    reqUpdate.input("Comentario", sql.NVarChar(1000), respuestaAdmin || null);
    reqUpdate.input("ResultadoPor_ID", sql.Int, idAdmin);

    let queryUpdate = `UPDATE dbo.BUZON_VECINAL
    SET
      Estado = @Estado,
      ResultadoPor_ID = @ResultadoPor_ID,
      RespuestaAdmin = COALESCE(@RespuestaAdmin, RespuestaAdmin)
    `;
    if (estandoNuevo === 'Resuelto') {
      queryUpdate += ` WHERE Estado = 'En Revisión'`;
    }
    queryUpdate += ` WHERE ID_Buzon = @ID_Buzon`;

    await reqUpdate.query(queryUpdate);
    // Insertar en la Bitácora
    const reqBitacora = new sql.Request(tx);
    reqBitacora.input("ID_Buzon_FK", sql.Int, id);
    reqBitacora.input("ID_Usuario_FK", sql.Int, idAdmin);
    reqBitacora.input("EstadoAnterior", sql.VarChar(20), estandoAnterior);
    reqBitacora.input("EstadoNuevo", sql.VarChar(20), estandoNuevo);
    reqBitacora.input("Comentario", sql.NVarChar(1000), respuestaAdmin || 'Cambiado a ${estandoNuevo}' );
  }
}

```

→ Endpoint de postulación:

```

import { sql, getPool } from "../pool.js";
// Crear una postulación
export const crearPostulacion = async (req, res) => {
  const { ID_Socio, ID_Proyecto, Comentario } = req.body;
  try {
    const pool = await getPool();

    await pool.request()
      .input("ID_Socio", sql.Int, ID_Socio)
      .input("ID_Proyecto", sql.Int, ID_Proyecto)
      .input("Comentario", sql.NChar(500), Comentario || "")
      .input("Estado", sql.NVarChar(20), "Pendiente")
      .query(`INSERT INTO POSTULACION_PROYECTO (ID_Socio, ID_Proyecto, Comentario, Estado, Fecha_Postulacion)
VALUES (@ID_Socio, @ID_Proyecto, @Comentario, @Estado, SYSDATETIME())`);

    res.status(201).json({ message: "Postulación registrada correctamente" });
  } catch (err) {
    console.error(`Error al crear postulación: ${err}`);
    res.status(500).json({ error: "Error al crear postulación" });
  }
};

//Obtener todas las postulaciones de un proyecto
export const obtenerPostulacionesPorProyecto = async (req, res) => {
  const { idProyecto } = req.params;
  try {
    const pool = await getPool();
    const result = await pool.request()
      .input("ID_Proyecto", sql.Int, idProyecto)
      .query(`
SELECT
  p.ID_Postulacion,
  p.ID_Proyecto,
  p.ID_Socio,
  p.Comentario,
  p.Estado,
  p.Fecha_Postulacion,
  s.Nombres,
  s.Apellidos,
  s.Carrera,
  s.Telefono,
  s.Rut
FROM POSTULACION_PROYECTO p
INNER JOIN SOCIOS s ON s.id_Socio = p.ID_Socio
WHERE p.ID_Proyecto = @ID_Proyecto
ORDER BY p.Fecha_Postulacion DESC
`);

    res.json(result.recordset);
  } catch (err) {
    console.error(`Error al obtener postulaciones: ${err}`);
    res.status(500).json({ error: "Error al obtener postulaciones" });
  }
};

// Actualizar estado de una postulación
export const actualizarEstadoPostulación = async (req, res) => {
  const idPostulacion = req.params.id;
  const { Estado } = req.body;

  console.log(`📝 Datos recibidos: ${idPostulacion}, ${Estado}`);

  try {
    if (!idPostulacion) {
      return res.status(400).json({ error: "Falta el ID de la postulación" });
    }

    if (!Estado || typeof Estado !== "string" || Estado.trim() === "") {
      return res.status(400).json({ error: "El campo Estado es obligatorio" });
    }

    const pool = await getPool();
    const result = await pool.request()
      .input("ID_Postulacion", sql.Int, idPostulacion)
      .input("Estado", sql.NVarChar(20), Estado)
      .query(`UPDATE dbo.POSTULACION_PROYECTO
SET Estado = @Estado
WHERE ID_Postulacion = @ID_Postulacion`);

    if (result.rowsAffected[0] === 0) {
      return res.status(404).json({ error: "Postulación no encontrada" });
    }

    res.json({ message: "Estado actualizado correctamente" });
  } catch (err) {
    console.error(`Error al actualizar postulación: ${err}`);
    res.status(500).json({ error: "Error interno del servidor" });
  }
};

```

ANEXO D: Pruebas Funcionales

Anexos 4. Pruebas Funcionales

→ Postulación Actividad de la junta de vecinos:

<div style="

→ Solicitar Certificado de residencia:

CERTIFICADO DE RESIDENCIA

Folio N°: C-00024

Fecha de emisión: 17-11-2025

La directiva de la Junta de Vecinos Mirador de Volcanes 4, RUT: 65.205.436-6 de la comuna de Puerto Montt.

Certifica conocer a Don(ña): **Carolina Paz Jara Orellana**
Rut: **18.591.409-7**

Como residente en calle o pasaje: **Costa Tenglo 1546**
del sector Mirador de Volcanes 4, comuna de Puerto Montt.

Este certificado tiene una vigencia de 30 días desde la fecha de emisión.

Heidi M. Orellana

PRESIDENTA JUNTA DE VECINOS
DIRECTIVA JUNTA DE VECINOS
MIRADOR DE VOLCANES 4