**Capstone**

**Junta de vecinos**

Integrantes: Miguel Medina, William López

[**Introducción**](#_xos6rzecoc08) **4**

[**Problema**](#_uw2dvm7l94fj) **5**

[**Solución y beneficios**](#_cgykyw1wndne) **8**

[**Solución propuesta**](#_ey0kj1ved1tj) **8**

[Principales funcionalidades de la solución:](#_1zksli2c7lis) 8

[Beneficios directos de la solución](#_uqe6yull11ou) 9

[**Alcance dentro del proyecto (MVP)**](#_1fa2n8z3hryd) **11**

[Fuera de alcance del MVP](#_j0cml3s1qmcr) 12

[Supuestos](#_x1rrvx45aiew) 12

[Restricciones](#_ch1tm029o7wb) 12

[**Metodología**](#_zd19e1jo62cn) **13**

[1. Justificación del uso de metodología](#_ng3rwxfhz8vd) 13

[2. Roles en el equipo](#_p7onr7g5pqrs) 13

[3. Artefactos de Scrum](#_m4unu92n2964) 13

[4. Ceremonias Scrum](#_4et2tpv2aaa1) 14

[5. Duración de Sprints y entregas](#_nk977wqrnprr) 14

[6. Herramientas de apoyo](#_68e1wofn8vyh) 14

[**Requerimientos**](#_qs71ypn3bs30) **15**

[Requerimientos funcionales (RF)](#_4eix4kb3icsl) 15

[Requerimientos no funcionales (RNF)](#_4ys6ux8umg3b) 17

[**Diagrama de clases**](#_ikbvurtuqwdl) **19**

[**Diagrama ERD**](#_161hhhcsr2gf) **21**

[Relación entre entidades](#_delk6hy5xkjj) 23

[Valor del ERD](#_u1nqlmk4s2hf) 23

[**Diagrama casos de uso**](#_66wqdxlbicwb) **24**

[Valor del diagrama](#_igfgrtch1ctg) 25

[Casos de uso (con IDs y breve descripción)](#_1g5yb143bm10) 26

[**Stack tecnológico**](#_elv3abeg98r8) **27**

[Frontend](#_r8vlbit774m9) 27

[Backend (API + vistas)](#_avevu6clla2z) 27

[Base de datos](#_3su9ym7trwml) 27

[Infraestructura](#_43sl899dub88) 28

[Funcionalidades de apoyo (agnósticas)](#_ppjwz6dqva0l) 28

[**Cronograma**](#_afb6hn4cfmvv) **29**

[**Cronograma (12 semanas, Scrum)**](#_1ig6ccav8pb) **29**

[Hitos globales](#_j8bfleaczufd) 29

[Plan por sprint (objetivo, tareas clave, entregables)](#_hbieqhafn0md) 29

[Sprint 0 (Semana 1) – Incepción y preparación](#_ig3tz7p2lmyc) 29

[Sprint 1 (Semanas 2–3) – Autenticación y padrón](#_co5tkvwaj2gn) 30

[Sprint 2 (Semanas 4–5) – Certificados (PDF + QR) y noticias](#_p557evn4fkuj) 30

[Sprint 3 (Semanas 6–7) – Notificaciones por email + transparencia de solicitudes](#_kxqsrke03k18) 30

[Sprint 4 (Semanas 8–9) – Recursos y reservas (anti-solape)](#_fwo6vc8rdy2l) 31

[Sprint 5 (Semanas 10–11) – Incidencias geolocalizadas + accesibilidad](#_dcjtvjnx4tm3) 31

[Sprint 6 (Semana 12) – Cierre, pruebas y documentación](#_dohnna7kkif6) 31

[Dependencias y buffers](#_p9emc3mci4tf) 33

[RACI (ligero)](#_e07pcy1u9poo) 33

[**Costos - Sistema unidad territorial**](#_nkgysqk6h7eu) **34**

[3. Costos de gestión de riesgos y pruebas](#_tdo8xyywmf5d) 35

[**Gestión de Riesgos**](#_jqrlhuq4p862) **36**

[1) Matriz de evaluación](#_shtjgvr4fh6g) 36

[**Plan de Pruebas**](#_omsfsgrbcjq0) **38**

[1) Estrategia](#_eksnpy1c56vh) 38

[2) Criterios de entrada/salida](#_hsxql1gz706g) 38

[3) Métricas de calidad](#_w8qpbmk2o46w) 39

[4) Herramientas](#_m1cd5mehuquy) 39

[6) Casos de prueba esenciales (detallados)](#_kigkezfw3d37) 41

[7) Severidad y SLA de defectos](#_nv16hlwarteo) 42

[8) Datos de prueba](#_v9fa5qb05mkh) 43

[9) Ambiente de pruebas](#_9held6fl43s2) 43

[**Conclusión**](#_j87urmsgs2ec) **44**

# **Introducción**

El presente plan de proyecto describe el diseño, desarrollo e implementación del **Sistema Unidad Territorial**, una solución tecnológica orientada a mejorar la gestión de las juntas de vecinos en Chile. Actualmente, muchas de estas organizaciones comunitarias enfrentan dificultades en la administración de trámites, la comunicación con los vecinos, la transparencia de procesos y el acceso equitativo a recursos y actividades.

El objetivo de este plan es establecer una ruta clara y estructurada para el desarrollo del sistema, detallando el problema a abordar, la solución propuesta, los beneficios esperados, el alcance definido, la metodología de trabajo, los requerimientos funcionales y no funcionales, los modelos de análisis y diseño, la tecnología a utilizar, el cronograma de ejecución, los costos asociados y las estrategias de riesgos y pruebas.

El sistema será implementado como un **sitio web responsivo**, desarrollado con **Django y PostgreSQL**, que permitirá a los vecinos y al directorio de la junta gestionar inscripciones, emitir certificados de residencia, publicar noticias, administrar reservas de recursos comunitarios, reportar incidencias con geolocalización y recibir notificaciones de manera digital. Además, se pondrá énfasis en la **accesibilidad digital**, con un modo inclusivo para adultos mayores, y en el cumplimiento de estándares de seguridad y calidad.

Este plan servirá como **documento guía** para los integrantes del equipo de desarrollo, los docentes supervisores y los futuros usuarios, asegurando que el proyecto avance con una visión compartida, metas claras y entregables verificables en cada etapa.

# **Problema**

En la actualidad, las juntas de vecinos en Chile enfrentan una serie de dificultades relacionadas con su gestión administrativa y la comunicación con los habitantes de su unidad territorial. Estas organizaciones, siendo la forma más directa de representación comunitaria, carecen en su mayoría de sistemas tecnológicos que permitan agilizar procesos, mejorar la transparencia y facilitar la participación ciudadana.  
  
**Principales problemas identificados:**

1. **Gestión manual y fragmentada:** La inscripción de vecinos, emisión de certificados de residencia y postulación a proyectos suele hacerse en papel o con herramientas básicas (Word, Excel, entre otras.), generando duplicidad de datos, errores y lentitud de transmisión.
2. **Falta de comunicación efectiva:** Los avisos, actividades y noticias se difunden en murales físicos o redes sociales informales, dificultando llegar a todos los vecinos, especialmente a adultos mayores o a quienes no usan redes sociales.
3. **Escasa transparencia y trazabilidad:** No existe un sistema centralizado que registre las solicitudes, aprobaciones o rechazos, lo que puede generar desconfianza entre los vecinos al no poder verificar el estado de sus trámites o postulaciones.
4. **Acceso limitado a recursos comunitarios:** La reserva de espacios como canchas, salones o plazas carece de un sistema de gestión en línea, lo que ocasiona conflictos por choques de horarios o falta de control.
5. **Falta de inclusión digital:** Los adultos mayores y vecinos con baja alfabetización digital quedan rezagados en el acceso a trámites y actividades
6. **Ausencia de un canal de incidencias territoriales:** Actualmente no existe una plataforma unificada para que los vecinos reporten problemas comunitarios (ejemplo: luminarias defectuosas, baches, basurales, etc.), dificultando la priorización y gestión de estas incidencias por parte del directorio.

**Impacto del problema:**

* Retraso en los trámites administrativos (certificados, inscripciones, proyectos, etc.).
* Baja participación en actividades vecinales.
* Conflictos por uso de recursos comunitarios.
* Pérdida de confianza y baja percepción de transparencia en la gestión.
* Exclusión de ciertos segmentos de la población (adultos mayores).
* Ineficiencia en la atención de problemas del barrio que afectan a la calidad de vida.

**Métricas y ejemplos:**

1. **Gestión manual y fragmentada:**

**Métrica:** Tiempo promedio para tramitar una inscripción o certificado.

**Ejemplo:** “Actualmente, la emisión de un certificado de residencia puede tardar entre **2 y 5 días hábiles**, dado que se procesa manualmente en papel o Excel, retrasando la entrega al vecino.”

1. **Falta de comunicación efectiva:**

**Métrica:** Canales utilizados.

**Ejemplo:** “De 50 vecinos, solo **15 marcan como visto el correo** de la junta de vecinos.”

1. **Escasa transparencia y trazabilidad:**

**Métrica:** Porcentaje de solicitudes sin seguimiento para el vecino.

**Ejemplo:** “En la última postulación a proyectos vecinales, un **60% de los postulantes** no recibió una confirmación clara de si su solicitud fue recibida ni en qué estado estaba.”

1. **Acceso limitado a recursos comunitarios:**

**Métrica:** Conflicto por reservas.

**Ejemplo:** “En promedio, se producen de **3 a 5 conflictos mensuales** por doble reserva de canchas o salones debido a la gestión manual en un cuaderno físico.”

1. **Falta de inclusión digital (adulto mayor)**

**Métrica:** Accesibilidad a trámites.

**Ejemplo:** “Actualmente, los adultos mayores deben **acudir físicamente** a la sede vecinal, lo que limita su participación.”

1. **Ausencia de canal de incidencias territoriales:**

**Métrica:** Número de incidencias comunitarias sin canal formal.

**Ejemplo:** “En promedio, se presentan **10 reclamos mensuales** (baches, luminarias apagadas, basurales, entre otras.), que se pierden en conversaciones informales que no tienen seguimiento.”

# **Solución y beneficios**

# **Solución propuesta**

El proyecto plantea el desarrollo de un **sistema web responsivo** para la gestión de las juntas de vecinos, diseñado con criterios de usabilidad, seguridad y accesibilidad. El sistema busca digitalizar y centralizar los procesos clave de la organización, permitiendo una gestión más eficiente, transparente e inclusiva.

## **Principales funcionalidades de la solución:**

1. **Gestión de vecinos y trámites en línea:**

* Registro y validación digital de vecinos.
* Solicitud y emisión de **certificados de residencia** con firma digital simple y código QR verificable.
* Padrón actualizado en tiempo real, reduciendo errores y duplicidades.

1. **Comunicación centralizada y multicanal**

* Módulo de **noticias y avisos** publicados en el sitio web.
* Notificaciones por **email** en el MVP.

1. **Transparencia y trazabilidad de solicitudes:**

* Seguimiento en línea de trámites (estado: recibido, en revisión, aprobado/rechazado).
* Registro histórico de solicitudes y resoluciones accesibles tanto para el directorio como para los vecinos.

1. **Gestión de recursos comunitarios:**

* Calendario digital para reservar canchas, salones o plazas.
* Sistema de aprobación/rechazo con notificaciones automáticas.
* Evita conflictos por doble reserva y mejora la utilización de los espacios.

1. **Inclusión digital para adultos mayores:**

* **Modo de accesibilidad:** alto contraste, tipografía ampliada y navegación simplificada.
* Posibilidad de asistencia guiada en el sitio (texto a voz básico o tutoriales visuales).
* Mantiene acceso presencial en paralelo, pero facilita a los adultos mayores realizar trámites en línea.

1. **Gestión de incidencias comunitarias con geolocalización**

* Módulo para que los vecinos reporten problemas (ejemplo: luminarias apagadas, baches, basurales, etc.) con fotos y ubicación en un mapa.
* Dashboard para el directorio con visualización de incidencias priorizadas por urgencia y frecuencia.
* Fomenta la participación activa y mejora la capacidad de respuesta de la junta.

## **Beneficios directos de la solución**

* **Agilidad en los trámites:**

**Indicador:** Reducción del tiempo promedio de emisión de certificados de residencia de **2-5 días a menos de 24 horas**.

**Beneficio:** Mayor eficiencia administrativa y satisfacción ciudadana.

* **Mejora en la comunicación con vecinos:**

**Indicador:** Aumento de la cobertura de comunicación oficial de un **40% a más del 80%** de los vecinos en los primeros 6 meses.

**Beneficio:** Mayor asistencia a actividades y mejor difusión de información relevante.

* **Transparencia en la gestión:**

**Indicador:** Disminución de solicitudes sin seguimiento informado de un **60% al 0%**.

**Beneficio:** Refuerzo de la confianza de la directiva vecinal.

* **Uso eficiente de recursos comunitarios:**

**Indicador:** Reducción de conflictos de doble reserva de **3-5 casos mensuales a 0**.

**Beneficio:** Aprovechamiento óptimo de canchas, salas y espacios comunitarios.

* **Inclusión de adultos mayores:**

**Indicador:** Aumento del acceso de adultos mayores a trámites en línea de **10% a 50% en el primer año**.

**Beneficio:** Participación más equitativa e integración de un grupo históricamente rezagado.

* **Gestión de incidencias territoriales:**

**Indicador:** Registro formal de al menos **80% de incidencias comunitarias** y reducción del tiempo de respuesta de **30 días promedio a 10-15 días**.

**Beneficio:** Mayor rapidez en la solución de problemas barriales y mejor calidad de vida en la comunidad, también aumentando la participación ciudadana.

# **Alcance dentro del proyecto (MVP)**

El sistema a desarrollar corresponde a un **sitio web responsivo** para la gestión de las juntas de vecinos, orientado a resolver los problemas identificados y entregar beneficios concretos a la comunidad. El alcance se define en dos niveles: **MVP (Producto Mínimo Viable)** y **futuras extensiones (fuera de alcance del presente proyecto, pero posibles de considerar en etapas posteriores).**

1. **Gestión de vecinos y trámites**
   * Registro y validación digital de vecinos.
   * Solicitud y emisión de **certificados de residencia con QR verificable**.
   * Padrón actualizado en tiempo real.
2. **Comunicación y noticias**
   * Módulo de publicación de **noticias, avisos y actividades**.
   * Notificaciones automáticas por **correo electrónico**.
3. **Transparencia y seguimiento de solicitudes**
   * Flujo de revisión y aprobación/rechazo de trámites.
   * Historial accesible de solicitudes y resoluciones.
4. **Gestión de recursos comunitarios**
   * Calendario digital de reservas (canchas, salones, plazas).
   * Sistema de aprobación/rechazo con notificación automática.
5. **Inclusión digital (adultos mayores)**
   * Interfaz accesible con **alto contraste, tipografía ampliada y navegación simplificada**.
   * Opción de asistencia guiada básica (texto a voz).
6. **Gestión de incidencias comunitarias**
   * Módulo para reportar incidencias con foto y geolocalización.
   * Dashboard para el directorio con priorización de incidencias.

## **Fuera de alcance del MVP**

* Integración con **WhatsApp Business API o SMS** (planificado como mejora futura).
* **Pagos electrónicos** para actividades o reservas.
* **Encuestas o votaciones digitales** de vecinos.
* **Portal de transparencia ampliado** (presupuestos, balances, actas públicas).
* **Firma electrónica avanzada** para resoluciones legales.
* Aplicación móvil nativa (el sistema será web responsivo).

## **Supuestos**

* Se dispone de un **hosting web y base de datos en la nube** de bajo costo (ejemplo: Render, Railway, Vercel).
* Los usuarios principales (directorio de la junta) contarán con **capacitación básica** para operar el sistema.
* Los vecinos tienen acceso mínimo a un navegador web desde PC o smartphone.

## **Restricciones**

* El proyecto se desarrollará en un plazo máximo de **12–14 semanas**.
* El equipo está compuesto por **dos integrantes**, lo que limita el alcance a funcionalidades críticas.
* El presupuesto contempla **costos mínimos de infraestructura** (uso de servicios gratuitos o de bajo costo en la nube).

# **Metodología**

Para el desarrollo del Sistema Unidad Territorial se empleará la **metodología de desarrollo Iterativa e Incremental**, adecuada para el proyecto.

Esta metodología permite dividir el trabajo en **ciclos cortos de 2 semanas**, denominados *iteraciones*, en los cuales se entrega una versión parcial y funcional del sistema. Cada iteración agrega nuevas funcionalidades sobre la base previamente desarrollada, asegurando progreso constante, validación continua y reducción de riesgos.

### **Justificación del uso de Iterativa-Incremental**

* Permite entregar versiones funcionales tempranas del sistema, facilitando la validación.
* Reduce el riesgo de errores acumulados, ya que cada iteración se prueba y documenta antes de continuar.
* Favorece la adaptabilidad: los módulos pueden priorizarse o ajustarse en función del tiempo disponible y las dificultades encontradas.

### **Roles en el equipo**

* **Equipo de desarrollo (2 integrantes):** responsables del análisis, diseño, codificación, pruebas y documentación del sistema.
* **Docentes supervisores:** actúan como revisores externos, validando el avance en cada iteración y asegurando el cumplimiento de objetivos académicos.

### **Estructura de trabajo en iteraciones**

* **Planificación de iteración:** se definen los módulos a implementar en el ciclo (ejemplo: padrón, certificados).
* **Desarrollo y pruebas:** se codifican las funcionalidades y se ejecutan pruebas unitarias, de integración y de validación.
* **Entrega incremental:** al cierre de la iteración se entrega un sistema ejecutable con las nuevas funciones añadidas.
* **Revisión y ajustes:** se documentan avances, se identifican problemas y se ajusta el plan para la siguiente iteración.

### 

### 

### **Entregables por iteración**

* Versión parcial del sistema con funcionalidades completas y probadas.
* Documentación técnica y de usuario actualizada.
* Evidencias de pruebas (reportes, capturas, logs).

# **Requerimientos**

## **Requerimientos funcionales (RF)**

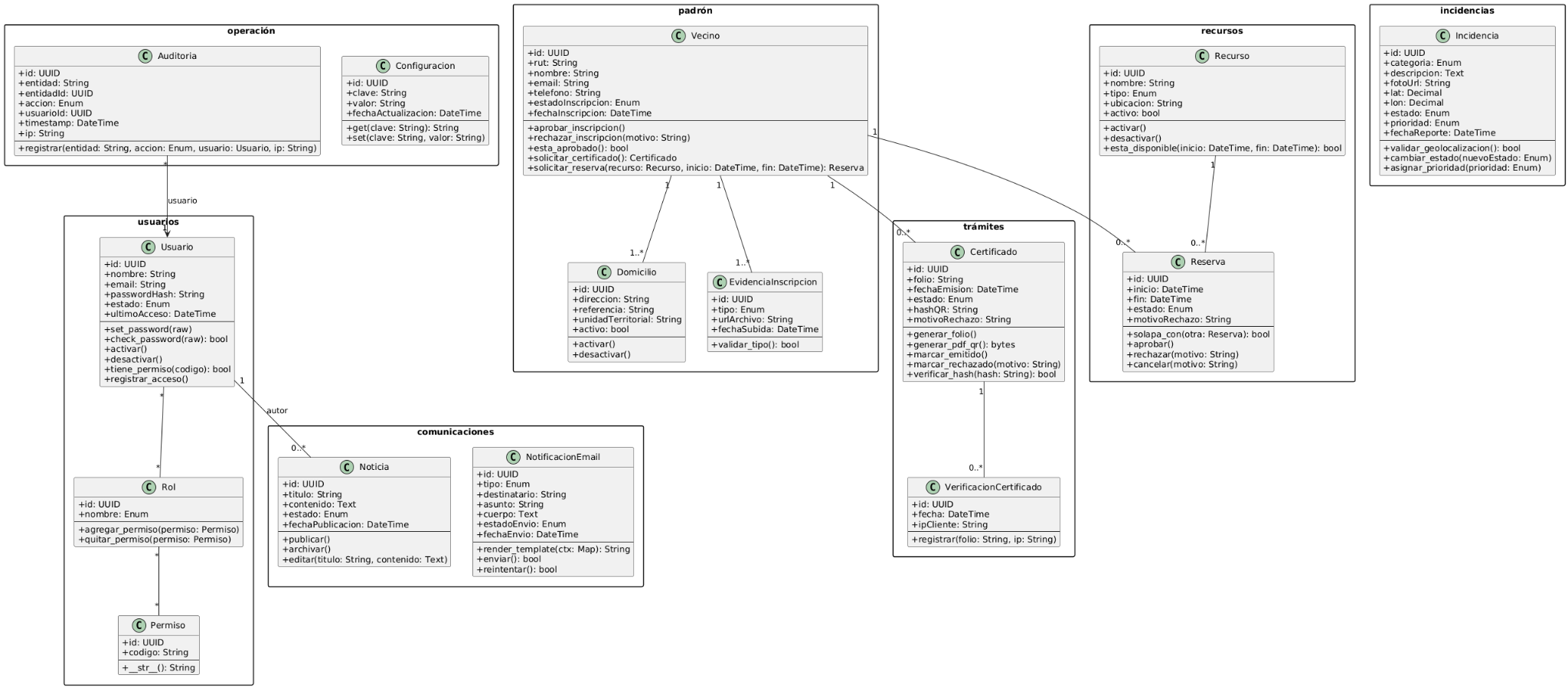
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Nombre** | **Descripción** | **Prioridad** | **Criterios de aceptación** |
| RF01 | Autenticación y roles (RBAC) | El sistema debe permitir autenticación de usuarios y asignación de roles. | Obligatorio | -Usuarios creados/inactivados. -Acceso restringido. -Recuperación de contraseña por email. |
| RF02 | Registro e inscripción de vecinos | El sistema debe permitir inscripción online de vecinos y validación del directorio. | Obligatorio | -Formulario público. -Flujo de aprobación. -Evidencias adjuntas seguras. |
| RF03 | Padrón y gestión de vecinos | Visualizar, buscar y actualizar el padrón de vecinos. | Obligatorio | -Búsqueda por RUT. -Historial de cambios. -Exportación CSV. |
| RF04 | Solicitud y emisión de certificados | Solicitar, revisar y emitir certificados en PDF con QR verificable. | Obligatorio | -Flujo completo.  -PDF con folio y QR. -Verificación vía endpoint. |
| RF05 | Notificaciones por email (MVP) | Enviar correos automáticos en eventos clave. | Obligatorio | -Plantillas.  -Log de envío. -Reintentos ante fallo. |
| RF06 | Módulo de noticias y avisos | Crear, editar, publicar y listar noticias/avisos. | Obligatorio | -Editor básico.  -Portada pública. -Registro de autor. |
| RF07 | Gestión de recursos y reservas | Administrar inventario de espacios y reservas con aprobación. | Obligatorio | -Calendario.  -Flujo de aprobación. -Evitar la doble reserva. |
| RF08 | Seguimiento de solicitudes | El vecino puede consultar el estado de sus solicitudes. | Obligatorio | -Vista de solicitudes. -Panel de gestión. -Exportación de reportes. |
| RF09 | Gestión de incidencias con geolocalización | Reportar incidencias con foto y ubicación en mapa. | Obligatorio | -Formulario con foto/mapa.  -Estados de incidencia.  -Tablero de gestión. |
| RF10 | Panel de indicadores (KPIs) | Mostrar métricas clave de gestión. | Deseable | -Widgets.  -Filtros por fecha. -Exportación snapshot. |
| RF11 | Accesibilidad modo adulto mayor | Interfaz accesible con alto contraste, fuentes grandes y asistencia básica. | Obligatorio | -Toggle accesible. -Contraste WCAG AA. -Texto a voz. |
| RF12 | Administración de contenidos y archivos | Cargar y gestionar archivos (actas, reglamentos, afiches). | Deseable | -Tipos permitidos. -Metadatos.  -Descarga pública. |
| RF13 | Bitácora de auditoría | Registrar acciones críticas de usuarios. | Obligatorio | -Log con usuario, acción, fecha y IP. -Filtros y exportación. |
| RF14 | Gestión de configuraciones básicas | Ajustar textos institucionales, datos de junta, plantillas de correo. | Deseable | -CRUD validado. -Rollback simple. |
| RF15 | Verificación pública de certificados | Permitir validar certificados por QR o folio. | Obligatorio | -Página pública /verify. -Estatus validado/invalidado. |

## **Requerimientos no funcionales (RNF)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Nombre** | **Descripción** | **Prioridad** | **Criterios de aceptación** |
| RNF01 | Rendimiento | El sistema debe responder en p95 < 800 ms para operaciones estándar. | Obligatorio | -Pruebas de carga. -Páginas pesadas < 1200 ms. |
| RNF02 | Disponibilidad | Disponibilidad mensual ≥ 99,0% en el MVP. | Deseable | -Monitoreo básico. -Ventanas de mantenimiento notificadas. |
| RNF03 | Seguridad | Cifrado TLS 1.2+, sesiones seguras, RBAC, protección OWASP. | Obligatorio | -Checklist sin hallazgos críticos.  -Hashing robusto. |
| RNF04 | Privacidad y datos personales | Consentimiento, minimización y eliminación de datos. | Obligatorio | -Consentimiento explícito.  -Función de eliminación/anonimización. |
| RNF05 | Usabilidad | SUS ≥ 80 en pruebas con usuarios del directorio. | Deseable | -Tareas clave en < 3 min por usuarios no técnicos. |
| RNF06 | Accesibilidad | Cumplir WCAG 2.2 AA en vistas principales. | Obligatorio | -Contrastes. -Navegación por teclado.  -Textos alternativos. |
| RNF07 | Compatibilidad/Portabilidad | Funciona en navegadores modernos y móviles. | Deseable | -Tests responsivos. -Compatibilidad con Chrome, Firefox, Edge, iOS, Android. |
| RNF08 | Mantenibilidad/CI-CD | Código con lintin, tests y pipeline CI/CD. | Deseable | -Cobertura backend ≥ 60%.  -Builds reproducibles con Docker. |

# **Diagrama de clases**

El diagrama de clases del **Sistema Unidad Territorial** representa la estructura estática del software, mostrando las entidades principales, sus atributos y las relaciones entre ellas. Este modelo permite visualizar cómo se organiza la información en el sistema y cómo interactúan los distintos módulos para dar soporte a los procesos de la junta de vecinos.



En el esquema se identifican **seis dominios clave**:

1. **Usuarios**: Gestiona la autenticación y los roles del sistema. Incluye las clases Usuario, Rol y Permiso, que permiten implementar un modelo de control de acceso basado en roles (RBAC).
2. **Padrón**: Contiene la información de los vecinos (Vecino), sus domicilios (Domicilio) y las evidencias que respaldan su inscripción (Evidencia/Inscripción). Un vecino aprobado en el padrón es requisito para acceder a trámites y reservas.
3. **Trámites**: Modela la emisión de certificados (Certificado) y su validación (Verificación/Certificado), asegurando unicidad de folio y trazabilidad mediante códigos QR.
4. **Comunicaciones**: Incluye la publicación de noticias (Noticia) y el envío de notificaciones por correo electrónico (Notificación/Email) a los vecinos.
5. **Recursos y Reservas**: Define los recursos comunitarios (Recurso) y su uso mediante reservas (Reserva), con reglas que eviten solapamientos en los mismos intervalos de tiempo.
6. **Incidencias**: Representa los reportes ciudadanos (Incidencia) con geolocalización y categorías predefinidas, permitiendo a la directiva gestionarlas y darles seguimiento.
7. **Operación**: Se incluyen mecanismos de auditoría (Auditoría) para registrar acciones críticas y configuración (Configuración) para parámetros del sistema.

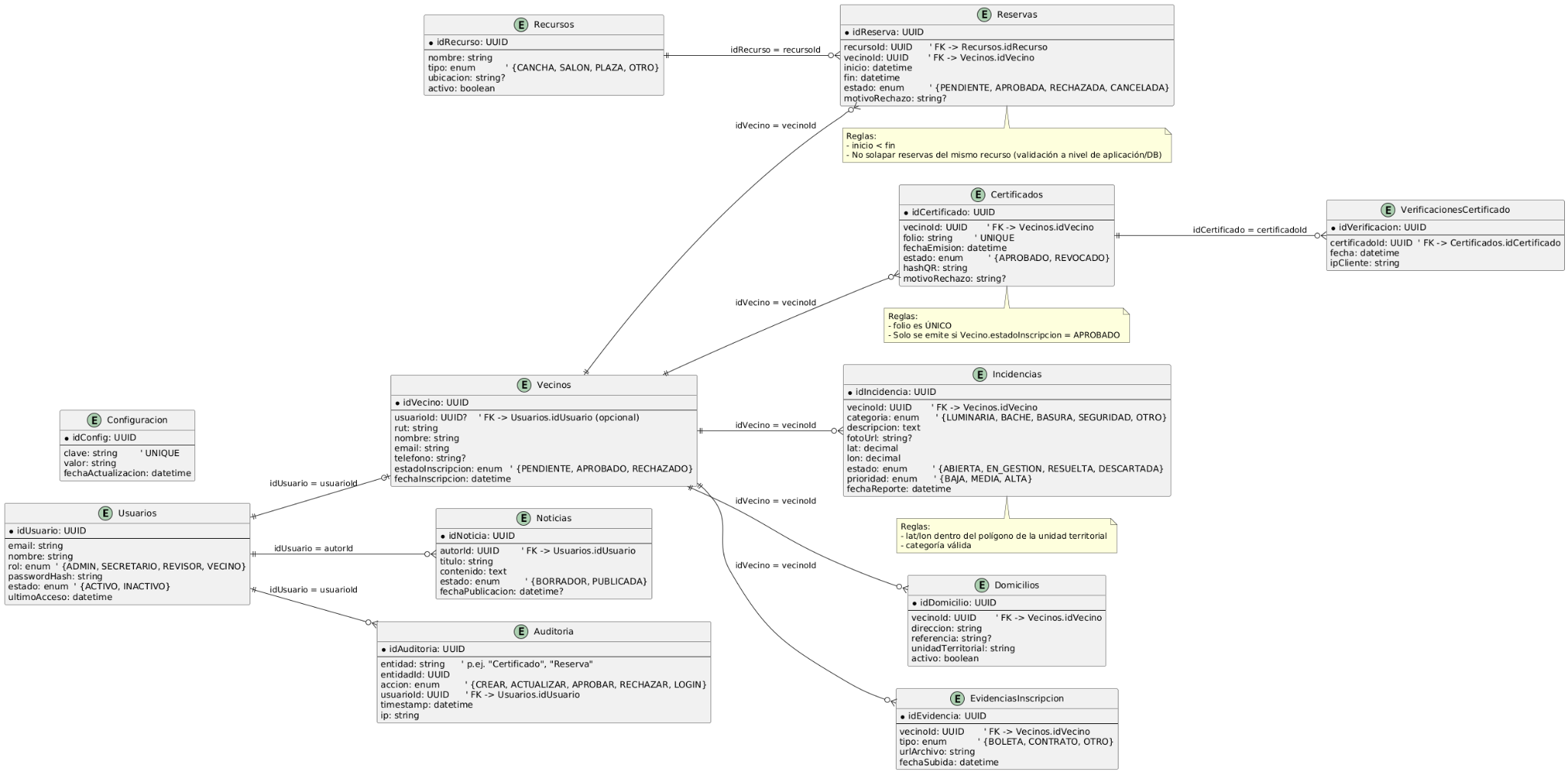
El diagrama destaca relaciones importantes como:

* **Vecino–Certificado** (un vecino puede tener varios certificados),
* **Recurso–Reserva** (un recurso admite múltiples reservas no solapadas),
* **Vecino–Incidencia** (un vecino puede reportar varias incidencias),
* **Usuario–Rol–Permiso** (para control de accesos y responsabilidades).

En conjunto, este modelo proporciona una **visión integral y modular** del sistema, garantizando consistencia de datos, seguridad en los accesos y trazabilidad de las operaciones, lo que respalda los objetivos funcionales y no funcionales definidos en el proyecto.

# **Diagrama ERD**

El **diagrama entidad–relación (ERD)** del Sistema Unidad Territorial muestra la estructura lógica de la base de datos en PostgreSQL, especificando tablas, atributos, claves primarias y relaciones entre entidades. Este modelo garantiza la **integridad referencial**, la consistencia de datos y la trazabilidad de los procesos definidos en los requerimientos.



En el esquema se distinguen las siguientes **entidades principales**:

1. **Usuarios**: Contiene credenciales, datos básicos, roles y estados de los usuarios que acceden al sistema. Se relaciona con la auditoría, noticias y el padrón de vecinos.
2. **Vecinos**: Registra la información personal y de inscripción de cada habitante. Incluye relaciones con domicilios, evidencias de inscripción, certificados, reservas e incidencias.
3. **Certificados**: Documentos oficiales vinculados a un vecino, con atributos como folio único, fecha de emisión, estado y hashQR. Se complementa con la tabla **VerificacionesCertificado**, que almacena consultas externas realizadas por terceros.
4. **Recursos y Reservas**:  
   * **Recursos**: representan canchas, salones u otros bienes comunitarios.
   * **Reservas**: gestionan el uso de los recursos, con validaciones para evitar solapamientos de horarios.
5. **Incidencias**: Reportes ciudadanos geolocalizados con categoría, descripción, coordenadas y estado. Permite la trazabilidad de problemas y su gestión por parte del directorio.
6. **Noticias**: Publicaciones creadas por usuarios administradores, visibles a la comunidad. Cada noticia está asociada a un autor (Usuarios).
7. **Evidencias/Inscripcion y Domicilios**: Complementan el padrón de vecinos con información de soporte (archivos subidos como comprobantes) y domicilios registrados.
8. **Auditoría y Configuración**:  
   * **Auditoría**: registra acciones críticas (crear, actualizar, aprobar, rechazar, login) con usuario, entidad y timestamp.
   * **Configuración**: almacena parámetros globales del sistema mediante clave–valor.

## **Relación entre entidades**

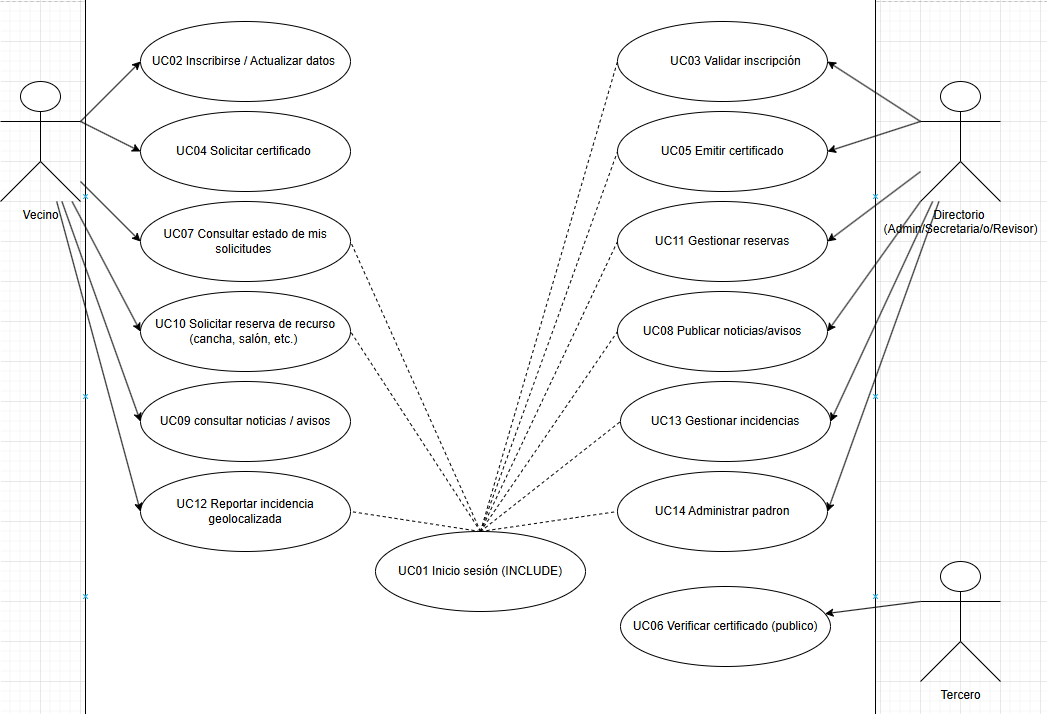
* **Un Usuario** puede estar vinculado a **un Vecino** y, a la vez, ser **autor de Noticias** o responsable en **Auditoría**.
* **Un vecino** puede tener múltiples **Certificados, Reservas, Incidencias, Domicilios y Evidencias**.
* **Un Certificado** pertenece a un **vecino** y puede ser verificado muchas veces en **Verificaciones/Certificado**.
* **Un Recurso** puede tener múltiples **Reservas**, y cada **Reserva** corresponde a un solo **Vecino**.
* Las **reglas de negocio** se refuerzan con restricciones:  
  + El folio de certificado debe ser **único**.
  + Las reservas no pueden **solaparse** en el mismo recurso/intervalo.
  + Una incidencia debe tener coordenadas válidas dentro del polígono de la unidad territorial.

## **Valor del ERD**

Este modelo permite asegurar que los procesos definidos en los casos de uso (inscripción, certificados, reservas, incidencias, noticias, notificaciones y auditoría) estén respaldados en la base de datos con integridad y consistencia. Además, facilita la escalabilidad, al estar normalizado y preparado para integraciones futuras (ej. notificaciones vía WhatsApp o votaciones comunitarias).

# **Diagrama casos de uso**

El **diagrama de casos de uso** del Sistema Unidad Territorial representa las principales interacciones entre los actores y el sistema, ilustrando los procesos que deben ser soportados por la aplicación. Su objetivo es **mostrar de forma clara qué funcionalidades ofrece el sistema y quiénes las utilizan**, asegurando que los requerimientos funcionales identificados se traduzcan en escenarios prácticos.



En el modelo se identifican tres actores principales:

1. **Vecino**: usuario que se inscribe en la junta de vecinos y utiliza el sistema para realizar trámites, consultar información, reservar recursos y reportar incidencias.
2. **Directorio (Admin/Secretario/Revisor)**: encargado de validar inscripciones, aprobar o rechazar solicitudes, gestionar reservas, publicar noticias, administrar el padrón y dar seguimiento a las incidencias.
3. **Tercero**: actor externo que solo interactúa con el sistema para verificar la validez de un certificado de residencia a través de un folio o código QR.

Los **casos de uso identificados** son:

* **UC01. Inicio de sesión (Include):** caso transversal que se incluye en todos los procesos que requieren autenticación y autorización.
* **UC02–UC14:** cubren funcionalidades específicas como inscripción, certificados, reservas, incidencias, noticias y administración del padrón.
* **UC06. Verificar certificado (público):** accesible directamente por un tercero sin necesidad de autenticación.

El diagrama refleja además la relación de **inclusión (<<include>>)** entre el inicio de sesión (UC01) y los demás casos protegidos, garantizando que solo usuarios autenticados puedan acceder a funcionalidades privadas.

## Valor del diagrama

Este modelo proporciona una **visión general de alto nivel** sobre cómo los distintos actores interactúan con el sistema, facilitando la comunicación entre desarrolladores, docentes y usuarios finales. Además, sirve como **punto de trazabilidad** para validar que todos los requerimientos funcionales (RF01–RF15) se encuentren representados en el sistema.

## 

## **Casos de uso (con IDs y breve descripción)**

* **UC01. Inicio Sesión** *(incluido por otros)*: Iniciar sesión para acceder a funciones privadas (RBAC).
* **UC02. Inscribirse / Actualizar datos (Vecino):** Enviar solicitud de inscripción y/o actualizar datos personales.
* **UC03. Validar inscripción (Directorio):** Revisar evidencias y aprobar/rechazar la inscripción del vecino.
* **UC04. Solicitar certificado de residencia (Vecino):** Crear solicitud de certificado.
* **UC05. Emitir certificado (Directorio):** Aprobar/rechazar solicitud y **generar PDF con QR**.
* **UC06. Verificar certificado (Tercero):** Validar la vigencia y datos mínimos del certificado vía QR/folio (acceso público).
* **UC07. Consultar estado de mis solicitudes (Vecino):** Ver seguimiento y resolución de trámites.
* **UC08. Publicar noticias/avisos (Directorio):** Crear/editar/publicar contenidos visibles en el portal.
* **UC09. Consultar noticias/avisos (Vecino):** Ver listados y detalle de noticias/avisos.
* **UC10. Solicitar reserva de recurso (Vecino):** Enviar solicitud de uso de canchas, salas, etc.
* **UC11. Gestionar reservas (Directorio):** Aprobar/rechazar solicitudes; evitar doble reserva.
* **UC12. Reportar incidencia geolocalizada (Vecino):** Registrar problema con foto+ubicación.
* **UC13. Gestionar incidencias (Directorio):** Priorizar, asignar estado y cerrar incidencias.
* **UC14. Administrar padrón (Directorio):** Buscar, editar y exportar datos del padrón.

# **Stack tecnológico**

## **Frontend**

* **Opción A (simple/rápida):** Sitio **server-rendered** con plantillas (HTML + CSS + JS).  
  + Framework de vistas: el propio del backend (p. ej., **Django templates**).
  + UI: Bootstrap/Tailwind.
  + Pros: menos piezas, curva rápida, ideal para MVP accesible.
* **Opción B (SPA/Interfaz rica):** **Framework JS** (React/Vue) consumiendo API.  
  + Pros: UX muy fluida; Contras: más complejidad (deploy, estado, SEO).

## **Backend (API + vistas)**

* **Framework web MVC** con autenticación, ORM y migraciones.  
  + Ejemplos: **Django**, Laravel, Spring Boot, .NET, etc.
  + Exponer **API REST**.

## **Base de datos**

* **Relacional** (recomendada para tu caso): **PostgreSQL**, MySQL/MariaDB u **Oracle**.
* **Cache/colas** (opcional): Redis, para notificaciones/colas.

## 

## **Infraestructura**

* **Contenedores:** Docker (dev y prod).
* **CI/CD:** GitHub Actions / GitLab CI.
* **Hosting:** proveedor cloud (Railway/Render/Fly.io) o on-premise.
* **Dominios/SSL:** reverse proxy con HTTPS.

## **Funcionalidades de apoyo (agnósticas)**

* **Email SMTP** (notificaciones).
* **Generación de PDF** y **código QR**.
* **Mapas** para incidencias (OpenStreetMap/Leaflet).
* **Accesibilidad:** cumplir WCAG 2.1/2.2 AA (alto contraste, teclado, textos alternativos).
* **Seguridad:** TLS, roles/permiso, protección XSS/CSRF/SQLi, auditoría.

# 

# 

# 

# **Arquitectura del Sistema**

## **Enfoque arquitectónico: Modelo–Vista–Controlador (MVC)**

El sistema adoptará una arquitectura **Modelo–Vista–Controlador (MVC)**, que separa responsabilidades en tres capas principales para favorecer mantenibilidad, pruebas y evolución incremental.

**Nota sobre Django (MTV):** Aunque Django utiliza el patrón **MTV (Model–Template–View)**, el mapeo con MVC es directo:

* **Model (MVC)** ↔ **Model (Django)**
* **View (MVC)** ↔ **Template (Django)**
* **Controller (MVC)** ↔ **View/URLs (Django)** (las “views” de Django actúan como controladores; el enrutador urls.py direcciona).

### **Objetivos**

* Aislar lógica de negocio y reglas de dominio en **Modelos/Servicios**.
* Mantener **Controladores** delgados (coordinan flujos, validan entrada/salida).
* Centralizar la **presentación** en **Vistas/Plantillas** accesibles y responsivas.

## **Capas y responsabilidades**

### **1) Modelo (Dominio + Persistencia)**

* **Entidades principales:** Usuario/Rol/Permiso, Vecino, Domicilio, Evidencia, Certificado, VerificaciónCertificado, Recurso, Reserva, Incidencia, Noticia, Auditoría, Configuración.
* **Responsabilidades:**
  + Reglas de negocio y validaciones (clean(), constraints a nivel de BD).
  + ORM Django (migraciones, relaciones, restricciones de unicidad/anti-solape).
  + Servicios de dominio (por ejemplo: generación de folio, cálculo de hash/QR, verificación de reservas).
* **Persistencia:** PostgreSQL.

### **2) Controlador (Coordinación de casos de uso)**

* **Elementos:** views.py (o Views de DRF si se expone API), urls.py, formularios/serializers.
* **Responsabilidades:**
  + Orquestar cada caso de uso (solicitar certificado, aprobar, emitir PDF+QR; crear reserva con anti-solape; registrar incidencia con geolocalización).
  + Validación de entrada/salida, manejo de estados, autorización por rol (RBAC).
  + Integraciones: envío de emails (SMTP), generación de PDF/QR, almacenamiento de evidencias.

### 

### **3) Vista (Presentación)**

* **Elementos:** plantillas Django (HTML + CSS + JS), componentes de UI (Bootstrap/Tailwind), páginas públicas (noticias, verificación de certificados) y privadas (paneles de gestión).
* **Responsabilidades:**
  + Renderizar datos ya preparados por el Controlador.
  + Accesibilidad (WCAG 2.2 AA): alto contraste, navegación por teclado, textos alternativos, modo adulto mayor.

## **Mecanismos transversales**

* **Seguridad:** autenticación, autorización por roles/grupos, CSRF, validación server-side, headers seguros, cifrado TLS.
* **Auditoría:** registro de acciones críticas (quién, qué, cuándo, IP).
* **Errores y logging:** captura centralizada, trazas, alertas en fallos de envío de correo o generación de PDF.
* **Configuración:** variables de entorno (secretos, SMTP, base de datos), perfiles dev/staging/prod.
* **Rendimiento:** consultas selectivas, índices en campos de búsqueda (RUT, folio, fechas), cache puntual si es necesario.
* **Pruebas:** unitarias (modelo/servicios), integración (views/API), E2E (flujos clave).

## **Flujo de solicitud típico (ejemplo: emitir certificado)**

1. **Vecino** crea solicitud de certificado (Vista → Controlador).
2. **Controlador** valida entrada, consulta **Modelo**, persiste solicitud y notifica recepción.
3. **Directorio** aprueba/rechaza (Controlador aplica reglas del **Modelo**).
4. **Servicio de certificados** genera **PDF + QR**, guarda evidencias y folio único (Modelo).
5. **Controlador** envía **email** al vecino y expone **/verify/<folio>** (Vista pública para terceros).

## **Justificación de MVC para este proyecto**

* **Encaja con Django** y su patrón MTV, manteniendo una separación clara de responsabilidades.
* **Favorece la iteración incremental:** cada módulo (padrón, certificados, reservas, incidencias) se incorpora sin afectar al resto.
* **Mejora pruebas y mantenimiento:** modelos y servicios probables con unit tests; controladores delgados; vistas simples.
* **Escalable:** permite exponer API REST sin romper la división por capas.

# **Cronograma**

## **Cronograma (12 semanas, Iterativa-Incremental)**

### **Hitos globales**

* **H0 – Inicio (semana 1):** Definición del alcance, plan de trabajo y configuración del entorno.
* **H1 – Versión base (semana 6):** Trámites y comunicación operativos.
* **H2 – Beta funcional (semana 10):** Reservas e incidencias completas.
* **H3 – Entrega final (semana 12):** Producto completo, pruebas y documentación.

### **Plan por iteración (objetivo, tareas clave, entregables)**

**Iteración 0 (Semana 1) – Incepción y preparación**

* Objetivo: Base del proyecto, lineamientos iniciales, backlog priorizado.
* Claves: Kick-off, setup del repositorio, configuración de entorno y base de datos.
* Entregables: Documento de plan, repositorio y pipeline, política de ramas.

**Iteración 1 (Semanas 2–3) – Autenticación y padrón**

* Objetivo: Implementar acceso seguro y gestión básica de vecinos.
* Claves: Autenticación de usuarios, modelos de vecinos y domicilios, flujo de inscripción.
* Entregables: Módulo de inscripción/validación y padrón con búsqueda/exportación.

**Iteración 2 (Semanas 4–5) – Certificados y noticias**

* Objetivo: Trámite de certificados y comunicación básica.
* Claves: Flujo solicitud–aprobación de certificados, generación de PDF con QR, módulo de noticias.
* Entregables: Certificados verificables + módulo de noticias.

**Iteración 3 (Semanas 6–7) – Notificaciones y seguimiento**

* Objetivo: Comunicación con vecinos y trazabilidad de trámites.
* Claves: Notificaciones por email, vista de “Mis solicitudes”, bitácora de auditoría.
* Entregables: Seguimiento end-to-end de trámites con notificaciones.

**Iteración 4 (Semanas 8–9) – Reservas comunitarias**

* Objetivo: Gestión de recursos comunitarios evitando solapamientos.
* Claves: Modelos de recursos y reservas, validación de conflictos, aprobación/rechazo.
* Entregables: Calendario digital de reservas operativo.

**Iteración 5 (Semanas 10–11) – Incidencias y accesibilidad**

* Objetivo: Reporte de incidencias con geolocalización y mejoras de inclusión digital.
* Claves: Módulo de incidencias en mapa, formulario con foto y ubicación, interfaz accesible.
* Entregables: Módulo de incidencias + modo accesible.

**Iteración 6 (Semana 12) – Cierre y documentación**

* Objetivo: Consolidar, probar y documentar el sistema.
* Claves: Pruebas completas, manual de usuario, guía de despliegue, presentación final.
* Entregables: Versión final del sistema + documentación y evidencias.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Iteración** | **Semanas** | **Objetivo** | **Entregables** | **Evidencias/Hito** |
| 0 | 1 | Incepción y preparación | Plan v1, repos/CI, Docker dev | Backlog priorizado, acta Kick-off |
| 1 | 2–3 | Auth + padrón | Inscripción/validación, padrón CRUD | Tests auth/CRUD, guía padrón |
| 2 | 4–5 | Certificados + noticias | PDF con QR, endpoint /verify, módulo de noticias | Swagger/E2E, demo interna |
| 3 | 6–7 | Notificaciones + seguimiento | Emails, “Mis solicitudes”, bitácora | Logs/envíos, **H1: MVP base** |
| 4 | 8–9 | Reservas (anti-solape) | Calendario, aprobación/rechazo | Tests anti-solape, manual |
| 5 | 10–11 | Incidencias + accesibilidad | Módulo de incidencias en mapa, modo accesible | SUS/encuestas, **H2: Beta** |
| 6 | 12 | Cierre y defensa | Release, ERS, manuales, video | Reportes pruebas, **H3: Final** |

## 

## 

## **Dependencias y buffers**

* **Dependencias:**
  + Reservas depende de auth/padrón (S1) y notificaciones (S3).
  + Incidencias depende de padrón (S1) y noticias opcionales (comunicación).
* **Buffer de riesgo:** 10–15% del tiempo de cada sprint (no planificar al 100%).
* **Criterios de “Hecho” (DoD):** código, pruebas, documentación mínima, desplegado en staging.

## **RACI (ligero, metodología iterativa-incremental)**

* **Equipo de desarrollo (Miguel / William):** Responsables de análisis, implementación, pruebas y documentación (R/A).
* **Docentes supervisores:** Revisan avances en los hitos H1/H2/H3, validan entregables y entregan retroalimentación (C/I).
* **Stakeholders secundarios (ej. junta piloto, si aplica):** Pueden entregar feedback opcional sobre la utilidad del sistema, pero no intervienen en la planificación (C).

# **Costos - Sistema unidad territorial**

**1. Infraestructura (Cloud / Hosting)**

|  |  |  |
| --- | --- | --- |
| **Concepto** | **Alternativa (MVP)** | **Estimación mensual** |
| Hosting Web/App | **Railway/Render (free/low tier)** | $0 – $10 USD |
| Base de Datos PostgreSQL | Incluida en hosting (tier free/low) | $0 – $10 USD |
| Almacenamiento archivos (evidencias/fotos) | S3-compatible (Cloudflare R2, 10GB free) | $0 |
| CDN/SSL | Cloudflare / Vercel SSL gratis | $0 |
| Dominio personalizado (opcional) | .cl o .com | $10 – $15 USD anual |

**Total Infraestructura (MVP):** $0 – $20 USD/mes.

**2. Herramientas de apoyo**

|  |  |  |
| --- | --- | --- |
| Herramienta | Uso | Costo |
| GitHub / GitHub Actions | Control de versiones + CI/CD | $0 (plan free) |
| Docker | Entorno dev/prod | $0 |
| Django + DRF + librerías (open source) | Backend + API | $0 |
| Bootstrap/Tailwind, Leaflet, qrcode, WeasyPrint | UI, mapas, QR, PDF | $0 |
| Mailtrap (desarrollo SMTP) | Testing correos | $0 (free tier) |
| SendGrid/Resend (prod SMTP) | Envío notificaciones reales | $0–$15 USD/mes según uso |
| Herramientas de pruebas (pytest, axe, Selenium/Playwright) | Testing | $0 |

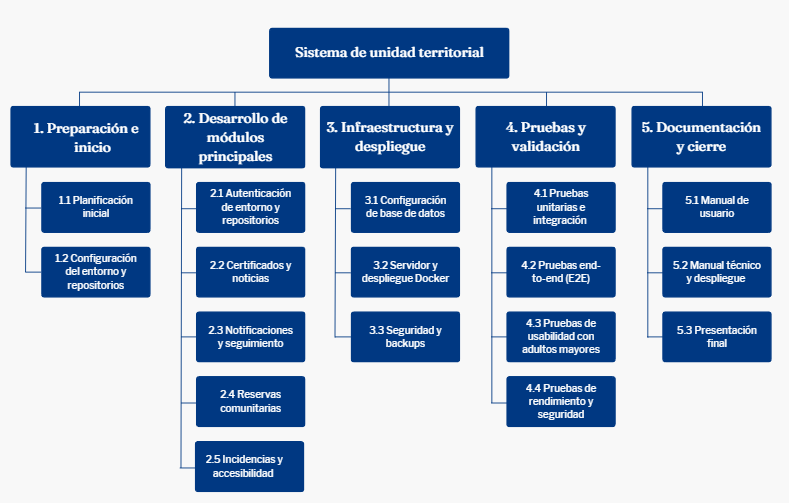
**Total Herramientas:** $0 – $15 USD/mes.

## **3. Costos de gestión de riesgos y pruebas**

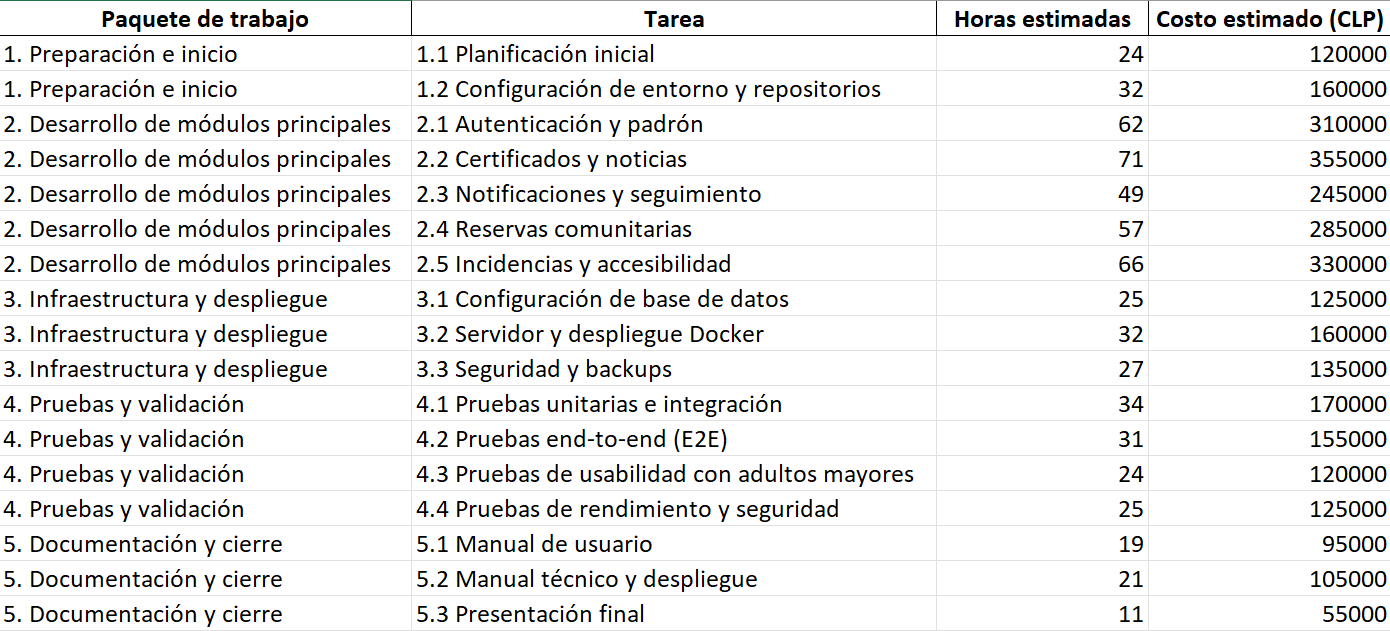
* **Tiempo de desarrollo (principal costo):**
  + 12 semanas × ~25 h/semana × 2 integrantes.
  + 600 - 620 horas hombre.
  + Valor referencial (si fuese proyecto real, $8.000–$12.000 CLP/hora → $2,880,000–$3,160,000 CLP).
* **Pruebas de usabilidad con vecinos/adultos mayores:**
  + 3–5 participantes, voluntarios.
  + Costo: $0 (si es colaboración) o simbólico (colación/transporte).
* **Mitigaciones de riesgos técnicos:**
  + Backups diarios (automatizado, incluido en hosting).
  + Control de versiones + pipeline (sin costo extra).
  + Plan B de hosting (migrar entre Railway, Render, Fly.io).
  + Pruebas automáticas + checklist OWASP básico.

## **4. Desglose de trabajo (EDT) y estimación de costos**

* **Estructura de desglose de trabajo (EDT):** Con el fin de obtener una estimación más precisa de los costos, se elaboró un **diagrama EDT** que organiza el proyecto en cinco paquetes principales: preparación e inicio, desarrollo de módulo, infraestructura y despliegue, pruebas y validación, y finalmente documentación y cierre. Cada paquete se divide en tareas específicas, a las cuales se les asignó una estimación de horas hombre según su complejidad y alcance.



* **Estimación de horas y costos por tarea:** El EDT permitió distribuir de manera proporcional las **610 horas hombre** planificadas. Aplicando una tarifa referencial de **$5.000 CLP/hora**, se obtiene un costo total estimado de **$3.050.000 CLP**,lo que se encuentra dentro del rango máximo establecido ($2.880.000 - $3.160.000).



**4. Resumen de costos**

* **Infraestructura:** $0–$20 USD/mes.
* **Herramientas:** $0–$15 USD/mes.
* **Tiempo de desarrollo (valor de referencia):** 610 horas hombre ($3.050.000 CLP en mercado).
* **Otros:** ~$15 USD anual.

# **Gestión de Riesgos**

## **1) Matriz de evaluación**

* **Probabilidad (P):** Baja (1), Media (2), Alta (3)
* **Impacto (I):** Bajo (1), Medio (2), Alto (3)
* **Severidad (S):** P × I → Bajo (1–2), Medio (3–4), Alto (6–9)

**2) Registro de riesgos**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Riesgo** | **Causa** | **Consecuencia** | **P** | **I** | **S** | **Señales tempranas** | **Mitigación (ex-ante)** | **Contingencia (ex-post)** | **Responsable** |
| R1 | Retraso por curva de librerías (PDF/QR/Leaflet) | Falta de experiencia | Bloqueo de funcionalidades clave | 2 | 3 | 6 | PRs lentos, issues abiertos >3 días | Prototipos tempranos (Spike) en S2; librería alternativa lista | Simplificar PDF (plantilla básica) y posponer estilos | William |
| R2 | Choque de reservas por error lógico | Validación incompleta | Conflictos y quejas de usuarios | 2 | 3 | 6 | Casos duplicados en QA | Doble validación: clean() + constraint SQL | Revisión manual por admin + script para detectar solapes | Miguel |
| R3 | Incidencias sin geolocalización válida | Datos erróneos | Tickets inutilizables | 2 | 2 | 4 | lat/lon fuera de rango | Validar polígono territorial en servidor | Corrección manual en tablero | William |
| R4 | Vulnerabilidades OWASP comunes | Inputs no validados | Exposición de datos | 1 | 3 | 3 | Hallazgos en checklist | Validación server-side + SecurityMiddleware | Hotfix + bloqueo temporal de endpoint | Miguel |
| R5 | Rotura de compatibilidad entre sprints | Cambios de esquema/API | Re-trabajo | 2 | 2 | 4 | Tests quebrándose | Migraciones controladas + contratos DRF | Feature flags y versión API | William |
| R6 | Falta de evidencias para la defensa | Olvido de reportes | Mala evaluación | 2 | 2 | 4 | Sprint sin adjuntos | Plantilla de evidencias por sprint | Recopilación final + anexo | Miguel |
| R7 | Retrasos por dependencias externas (SMTP/S3) | Corte del proveedor | Notificaciones fallan | 2 | 2 | 4 | Timeouts frecuentes | Retrys + colas y fallback SMTP | Reintentos manuales y reenvío | William |

**Buffers:** reservar 10–15% del tiempo por sprint para incertidumbres (riesgos R1, R4 y R8).

**Revisión:** actualizar matriz en cada Sprint Review (cambiar P/I/S según avance).

# **Plan de Pruebas**

## **1) Estrategia**

* **Niveles:**
  + **Unitarias** (modelo/servicios Django): validaciones clean(), reglas de negocio, utilidades.
  + **Integración** (DRF/ORM): endpoints clave y transacciones.
  + **E2E** (flujo completo): usuario → solicitud → aprobación → PDF+QR → verificación.
  + **UAT** (aceptación con usuarios): directiva y 3–5 vecinos/adultos mayores.
* **Tipos:** funcionales, accesibilidad, rendimiento, seguridad, compatibilidad (navegadores y móviles), datos (backup/restore).

## **2) Criterios de entrada/salida**

* **Entrada (para iniciar pruebas de sprint):**
  + Historias con criterios de aceptación definidos, endpoints/vistas listos en staging.
  + Datos semilla disponibles.
* **Salida (para dar por “Hecho”):**
  + Pruebas unitarias **verdes**, cobertura mínima **≥ 60%** backend (objetivo 70%).
  + **0 defectos críticos/altos** abiertos.
  + Evidencias: capturas, reporte pytest, logs de despliegue.

## 

## **3) Métricas de calidad**

* **Cobertura backend:** ≥ 60% (target 70%).
* **p95 de respuesta:** < **800 ms** (RNF01) en operaciones estándar.
* **SUS (usabilidad):** ≥ **80** con directiva (RNF05).
* **Accesibilidad:** WCAG 2.1/2.2 **AA** en vistas principales (RNF06).
* **Defectos por severidad:** 0 críticos al release.

## **4) Herramientas**

* **pytest**, **pytest-django**, **factory\_boy** (fixtures).
* **django-test + Client** para vistas/DRF.
* **Playwright** (opcional) para E2E.
* **axe-core** (revisión accesibilidad, manual + extensión).
* **k6/Locust** para carga básica (picos de 30–50 usuarios).
* **OWASP ZAP** (scan básico de seguridad).

**5) Plan de pruebas por área**

|  |  |  |  |
| --- | --- | --- | --- |
| **Área** | **Objetivo** | **Casos clave** | **Evidencia** |
| Autenticación/RBAC | Acceso por rol | Login, recuperación, acceso denegado (403) | Reporte pytest + capturas |
| Padrón (Vecino) | CRUD correcto | Alta/validación, edición, exportación CSV | Logs y CSV generado |
| Certificados | Emisión y verificación | Solicitar→aprobar; PDF con QR; /verify/<folio> | PDF generado + lectura QR |
| Noticias/Avisos | Publicación | Crear/editar/publicar; vista pública | Capturas de portada |
| Reservas | Anti-solape | Crear 2 reservas solapadas mismo recurso → bloquear | Prueba que falla/OK + constraint SQL |
| Incidencias | Geolocalización | Reporte con foto + lat/lon; tablero de gestión | Capturas mapa + listado |
| Notificaciones | Emails eventos | Envío en aprobación/rechazo; logs SMTP | Log de Mailtrap |
| Auditoría | Trazabilidad | Registro de acciones críticas | Export de logs |

## **6) Casos de prueba esenciales (detallados)**

**CT-001 – Emisión de certificado con QR**

* Pre: Vecino APROBADO, solicitud creada.
* Pasos: Aprobar solicitud → generar PDF → obtener folio/QR → abrir /verify/<folio>.
* Resultado esperado: PDF con folio/fecha/QR; verificación “Válido” vía web; registro en auditoría.

**CT-002 – Prevención de doble reserva**

* Pre: Recurso “Cancha A”; reserva A [10:00–11:00] aprobada.
* Pasos: Intentar crear reserva B [10:30–11:00] mismo recurso.
* Resultado: Bloqueo con mensaje de conflicto; no se crea registro (pasa validación de modelo y constraint en DB).

**CT-003 – Incidencia geolocalizada**

* Pre: Mapa operativo; polígono de unidad territorial configurado (server).
* Pasos: Cargar foto, indicar categoría, fijar marcador, enviar.
* Resultado: Incidencia guardada con lat/lon válidos; estado “Abierta”; visible en tablero.

**CT-004 – Modo accesible (adulto mayor)**

* Pre: Vista pública con toggle “Accesible”.
* Pasos: Activar modo; navegar por teclado.
* Resultado: Contraste cumple AA; tamaño ≥18px; foco visible; sin trampas de teclado.

**CT-005 – Seguridad básica**

* Pre: Usuario sin rol de aprobación.
* Pasos: Intentar acceder a vista de “Emitir certificado”.
* Resultado: 403 Forbidden; evento en bitácora.

**CT-006 – Rendimiento (smoke)**

* Pre: Entorno staging con datos semilla.
* Pasos: 30 usuarios concurrentes, 5 min, mix de operaciones (login, listar noticias, crear incidencia).
* Resultado: p95 < 800 ms; tasa de error < 1%.

## **7) Severidad y SLA de defectos**

* Crítico: caída, pérdida de datos, seguridad → corregir antes del release del sprint.
* Alto: bloquea función clave → corregir dentro del sprint.
* Medio: workaround disponible → planificar siguiente sprint.
* Bajo: cosmético → backlog.

## 

## **8) Datos de prueba**

* Semillas mínimas: 1 admin, 5 vecinos, 1 recurso, 3 incidencias.
* Archivos Dummy: evidencias PDF/JPG, fotos PNG de 200–500KB.
* Certificados de prueba: folios simulados y QR codificados a /verify/<folio>.

## **9) Ambiente de pruebas**

* Entorno staging igual a producción (PostgreSQL, configuración similar).
* Backups habilitados.
* SMTP configurado hacia Mailtrap.
* Logging y métricas activas.