**[nombre proyecto]**

**(SAD) Software Architecture Document**

**Versión 1.0**

# Tabla de Contenidos

**Introducción 3**

[1.1](#_heading=h.30j0zll) Contexto del Problema 3

[1.2](#_heading=h.1fob9te) Propósito 3

[1.3](#_heading=h.3znysh7) Ámbito 3

[1.4](#_heading=h.2et92p0) Definiciones, acrónimos y abreviaciones 3

[1.5](#_heading=h.tyjcwt) Referencias 3

[1.6](#_heading=h.3dy6vkm) Resumen ejecutivo 4

[**2**](#_heading=h.3whwml4) **Representación de la Arquitectura 5**

[2.1](#_heading=h.1t3h5sf) Representación 5

[**3**](#_heading=h.4d34og8) **Metas y Restricciones de la Arquitectura 6**

[A continuación se revisan las metas y restricciones de la arquitectura. 6](#_heading=h.2s8eyo1)

[3.1](#_heading=h.17dp8vu) Metas de la arquitectura 6

[3.2](#_heading=h.3rdcrjn) Restricciones de la Arquitectura 6

[3.3](#_heading=h.26in1rg) Otros antecedentes y consideraciones 6

[**4**](#_heading=h.lnxbz9) **Vista de Casos de Uso y Escenarios de Calidad 8**

[4.1](#_heading=h.35nkun2) Modelo de Casos de Uso 8

[4.2](#_heading=h.1ksv4uv) Especificación de Casos de Uso Relevantes 8

[4.3](#_heading=h.44sinio) Especificación de los Escenarios de Calidad Relevantes 9

[**Vista Lógica 11**](#_heading=h.2jxsxqh)

[4.4](#_heading=h.z337ya) Parte Estructural 11

[4.5](#_heading=h.3j2qqm3) Parte Dinámica 11

[**5**](#_heading=h.1y810tw) **Vista de Procesos 13**

[**6**](#_heading=h.4i7ojhp) **Vista de Implementación 14**

[**7**](#_heading=h.2xcytpi) **Vista de Despliegue 15**

[**8**](#_heading=h.1ci93xb) **Decisiones de Diseño y Selección de Alternativas 16**

# Identificación de Documento

|  |  |
| --- | --- |
| **Identificación** |  |
| **Proyecto** |  |
| **Versión** |  |

|  |  |
| --- | --- |
| **Documento mantenido por** |  |
| **Fecha de última revisión** |  |
| **Fecha de próxima revisión** |  |

|  |  |
| --- | --- |
| **Documento aprobado por** |  |
| **Fecha de última aprobación** |  |

# Historia de cambios

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Índice de figuras

Figura 1: Diagrama de Arquitectura General del Sistema

Figura 2: Diagrama de Componentes

Figura 3: Diagrama de Casos de Uso

Figura 4: Diagrama de Base de Datos

# Introducción

El presente documento describe la arquitectura de software del proyecto Sistema Unidad Territorial, orientado a mejorar la gestión de una junta de vecinos. Este sistema permitirá a la organización interna del directorio y a los vecinos registrar actividades, proyectos entre otros, a través de una plataforma web. La documentación tiene como objetivo guiar el desarrollo, garantizar la coherencia técnica y establecer los lineamientos para la implementación y mantenimiento del sistema.

# Alcance

El sistema permitirá:

Inscripción y gestión de vecinos.

Solicitud y emisión de certificados de residencia.

Postulación y gestión de proyectos vecinales.

Publicación de noticias y envío de notificaciones vía correo o WhatsApp.

Gestión de solicitudes de uso de espacios comunes (canchas, salas, plazas) mediante un calendario.

# Referencias

# Manuales de Django y DRF (Django Rest Framework).

# Documentación oficial de Bootstrap para el frontend.

# Guías de buenas prácticas de desarrollo web y seguridad de software.

# Normas de documentación UML para diagramas de arquitectura.

**Definiciones, acrónimos y abreviaciones**

|  |  |
| --- | --- |
| **ACRÓNIMO** | **DESCRIPCIÓN** |
| *CU* | Caso de Uso |
| *EC* | Escenario de Calidad |
| *DS* | Diagrama de secuencia |
| *MVC* | Modelo Vista Controlador |
| *MVT* | Modelo Vista Template |

# Arquitectura de Software

El sistema se basa en una arquitectura en capas:

1. Capa de Presentación (Frontend): Interfaz web responsiva desarrollada en HTML, CSS y JS.
2. Capa de Lógica de Negocio (Backend): Implementada en Django y Django REST Framework, gestionando reglas de negocio, autenticación y control de roles.
3. Capa de Persistencia: Base de datos relacional que almacena información de vecinos, proyectos, actividades y notificaciones.

Se prioriza escalabilidad y mantenibilidad, utilizando buenas prácticas de diseño y modularización.

# Objetivos y Restricciones de la Arquitectura

**Objetivos**

Garantizar la seguridad y privacidad de los datos de los vecinos.

Facilitar la integración de nuevas funcionalidades a futuro.

Asegurar un rendimiento aceptable para usuarios concurrentes.

**Restricciones**

El sistema debe funcionar sobre navegadores modernos.

El backend debe usar Django 5.x y la base de datos PostgreSQL.

Los usuarios deben autenticarse mediante correo y contraseña, con roles diferenciados (admin y vecino).

# Diagramas

Se incluirán los siguientes diagramas:

Diagrama de Arquitectura General: Vista de las capas del sistema y flujo de información.

Diagrama de Componentes: Módulos de frontend, backend y base de datos.

Diagrama de Casos de Uso: Funcionalidades de vecinos y administradores.

Diagrama

El contenido generado por IA puede ser incorrecto.

Diagrama de Base de Datos: Tablas, relaciones y atributos principales.

# Tamaño y desempeño

Número estimado de usuarios concurrentes: 25-50.

Número estimado de registros de vecinos: hasta 1.000.

Tiempo de respuesta promedio esperado: < 2 segundos por operación estándar.