RESIDENT House

Integrantes del Grupo: Laynus Ugarte
Cristian Salvo
Leonardo Castillo

Profesor: Ricardo Alberto Aravena Videla

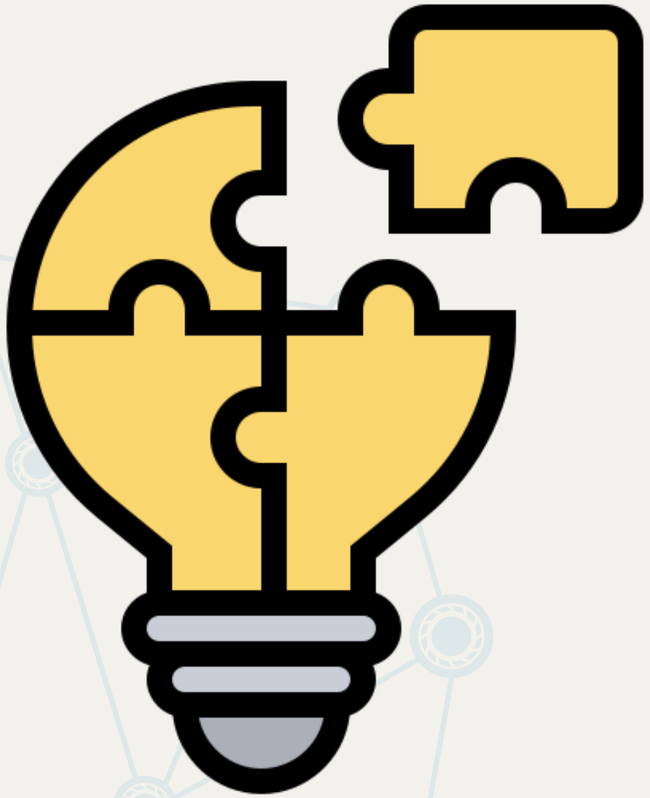
Problema y Contexto

En los condominios, la seguridad y la coordinación en la portería son aspectos fundamentales para garantizar el bienestar de los residentes. Sin embargo, los métodos que se utilizan actualmente —como llamadas telefónicas, timbres o avisos presenciales— presentan diversas limitaciones:

- Demoras en la autorización de visitas o entregas.
- Confusión y pérdida de información al no contar con un registro digital centralizado.
- Falta de trazabilidad y control, lo que dificulta la gestión por parte de la administración.
- Canales de comunicación dispersos (teléfono, WhatsApp, papel), que generan ineficiencias y errores.

Esta situación provoca inseguridad, desorganización y pérdida de tiempo tanto para los conserjes como para los residentes. Frente a este contexto surge la necesidad de una solución tecnológica moderna, que permita centralizar la comunicación, agilizar procesos y ofrecer un registro confiable y en tiempo real de todas las interacciones en la portería.

Solución Propuesta



“Resident House” es una aplicación móvil que conecta en tiempo real a residentes y conserjes, agilizando el registro de visitas, trabajadores, entregas y paquetes. Gracias a notificaciones instantáneas, los vecinos podrán autorizar o rechazar accesos desde su celular, mientras que el conserje llevará un control digital claro y organizado. Además, incluye un módulo de mensajería y un apartado especial de avisos comunitarios que fortalecen la colaboración vecinal.

OBJETIVOS

Objetivo general:

Desarrollar una aplicación móvil que mejore la comunicación y gestión entre residentes y conserjes en condominios.

Objetivos específicos:

• Implementar un sistema de registro digital de visitas y paquetería.

• Desarrollar un módulo de notificaciones push para residentes

• Crear un espacio de avisos comunitarios para emergencias, solicitudes y donaciones

Funcionalidades Principales

- Login diferenciado para residentes y conserjes.
- Registro digital de residentes, nuevos trabajadores.
- Notificaciones en tiempo real al residente.
- Aprobación o rechazo de ingreso de la visita o paquetería desde el celular.
- Bitácora digital de eventos para la administración.
- Módulo de mensajería segura entre conserje y residente.

Apartado Especial – Avisos Comunitarios:

- Publicar avisos de emergencias (robos, incendios, accidentes)
- Reportar objetos perdidos o hallados.
- Hacer solicitudes vecinales (reuniones, apoyo, reparaciones).
- Organizar campañas de donaciones o ayuda comunitaria.

CLIENTE O USUARIO OBJETIVO

- Residentes 🏡

Personas que habitan en el condominio y requieren un entorno seguro, organizado y eficiente. Buscan tener control sobre quién accede a su hogar y recibir notificaciones claras en tiempo real.

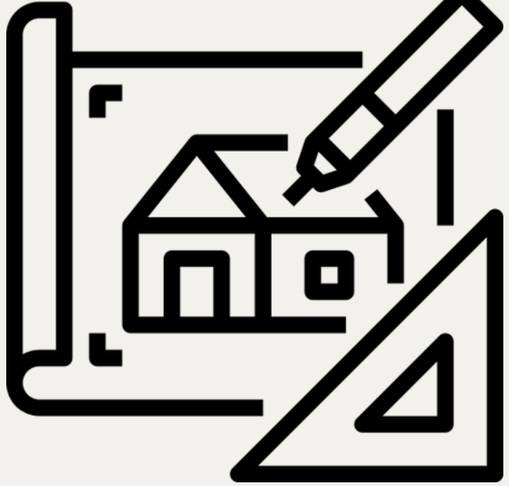
- Conserjes 👮‍♂️🛂

Encargados de la gestión diaria de accesos, visitas y paquetería. Necesitan una herramienta que simplifique su labor, reduzca errores y mantenga un registro confiable de cada evento.

- Administración 📊

Representa a la entidad que supervisa el condominio. Su interés está en contar con reportes automáticos, bitácoras digitales y trazabilidad de procesos, lo que facilita la toma de decisiones y garantiza la transparencia en la gestión.

Arquitectura propuesta.



Se eligió una arquitectura cliente–servidor en capas porque:

- Claridad y organización: separa la app móvil (Ionic + Angular) de los servicios backend, facilitando el mantenimiento y la escalabilidad.
- Eficiencia en la comunicación: los residentes y conserjes usan la app, que se conecta en tiempo real con el backend (Firebase, de momento) para manejar registros, accesos y notificaciones.
- Flexibilidad tecnológica: al trabajar con Firebase hoy, se obtiene rapidez de desarrollo, pero se mantiene la posibilidad de migrar más adelante a otra nube (ej. Oracle Cloud) sin reestructurar todo el sistema.
- Seguridad y control: el backend gestiona autenticación, roles y permisos, garantizando que cada actor (residente, conserje, administración) solo acceda a la información que le corresponde.
- Escalabilidad: la separación por capas permite agregar futuras funciones (chat, reportes avanzados, etc.) sin afectar la base ya implementada

ARQUITECTURA PROPUESTA

Capa de Presentación

Front-end / App môvil
Desarrollada en lonic + Angular
Accesible por residentes y conserjes

Capa de Lógica de Negocio

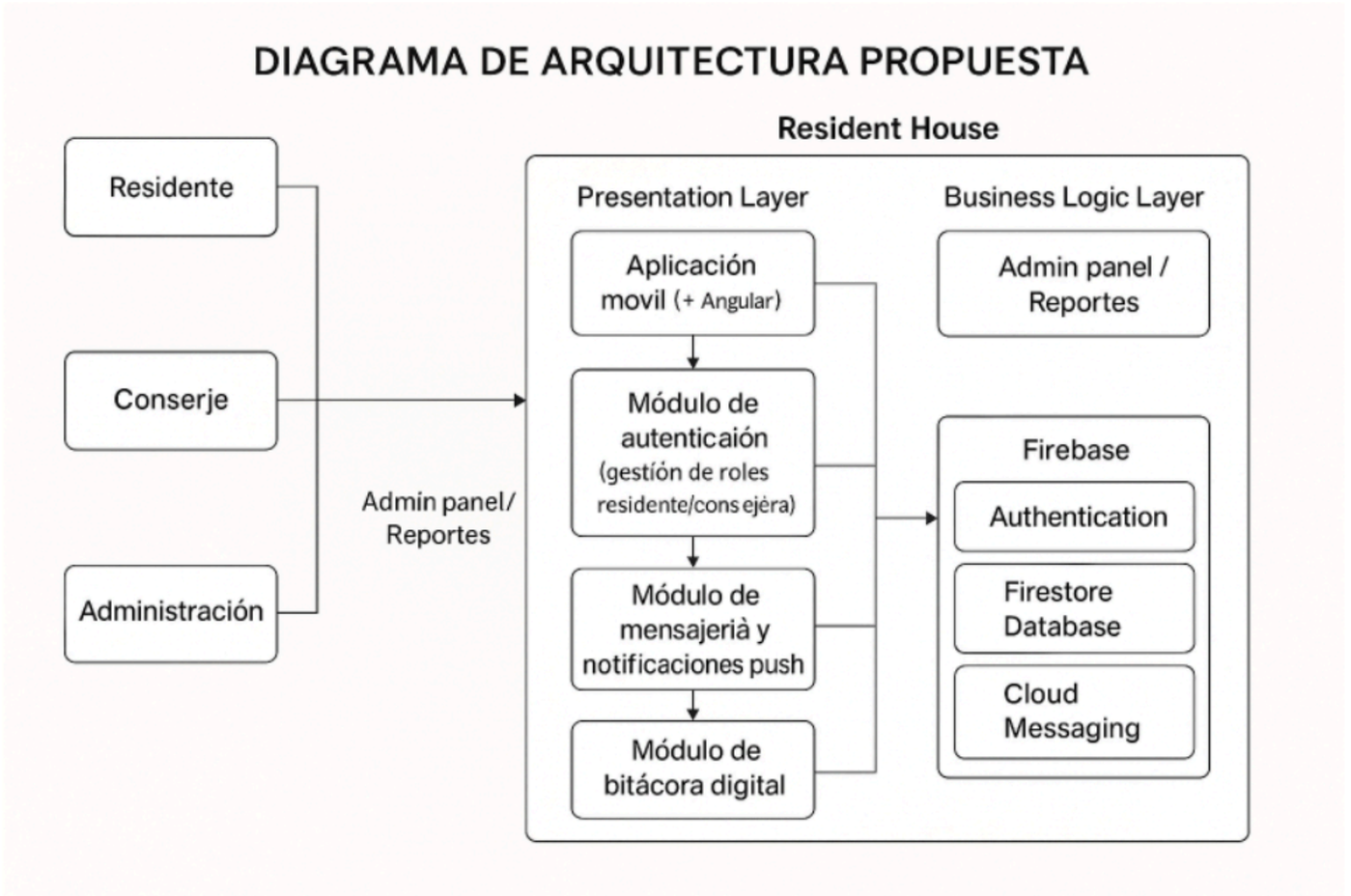
- Gestiôn de accesos
- Registro de visitas y paquetería
- Mensajería residenté-conserje
- Notificaciones en tiempo real

Capa de Datos

Back-end / BaaS
Firebase (De momento) para autenticación, base de datos en la nube y notificacione push

Capa de Administración

Panel de reportes e bitácora digítal para la administración del condominío



Lenguajes, frameworks y librerías.

Angular: framework frontend robusto para construir interfaces modernas y dinámicas.

Ionic: permite crear aplicaciones móviles híbridas de forma rápida y multiplataforma.

Capacitor: facilita la integración con funcionalidades nativas del dispositivo (notificaciones, cámara, etc.).

Node.js: entorno de ejecución que permite manejar el backend en JavaScript con alto rendimiento.

Firebase (de momento): servicios en la nube para autenticación, base de datos y notificaciones push.

Requerimientos funcionales y no funcionales.

Funcionales

Registro e inicio de sesión: acceso diferenciado para residentes y conserjes.

Gestión de visitas y paquetería: registro digital en tiempo real.

Bitácora digital: historial de accesos y eventos para la administración.

Módulo de mensajería: comunicación directa entre residentes y conserjes.

Notificaciones push: alertas instantáneas para accesos, avisos o emergencias.

No Funcionales

Escalabilidad: el sistema debe adaptarse al crecimiento de usuarios y datos.

Seguridad: uso de protocolos seguros para proteger información sensible.

Disponibilidad: acceso al sistema en cualquier momento y desde cualquier dispositivo.

Usabilidad: interfaz simple e intuitiva para usuarios no técnicos.

Mantenibilidad: facilidad para realizar mejoras y correcciones futuras.

Roadmap Proyecto Resident House

Fase 1 – Definición del Proyecto

- Kickoff
- Definición de actores, objetivos y requerimientos funcionalES
- Elaboración de casos de uso y diagramas

Fase 2 – Diseño y Prototipo inicial (Semanas 3–6)

- Diseno de arquitectura en capas (lonic + Angular +-Firebase, de momento)
- Creación de mockups y primeros flujos de usuario
- Desárrollo del prototipo inicial (MVP-1)

Fase 4 – Pruebas y Ajustes (Semanas 11–13)

- Pruebas de integración y de aceptación de usuario
- Auditoria de seguridad y datos
- Corrección de errores y mejoras
- Preparación de informes intermedios

Fase 3 – Desarrollo de Funcionalidades Avanzadas (Semanas 2-10)

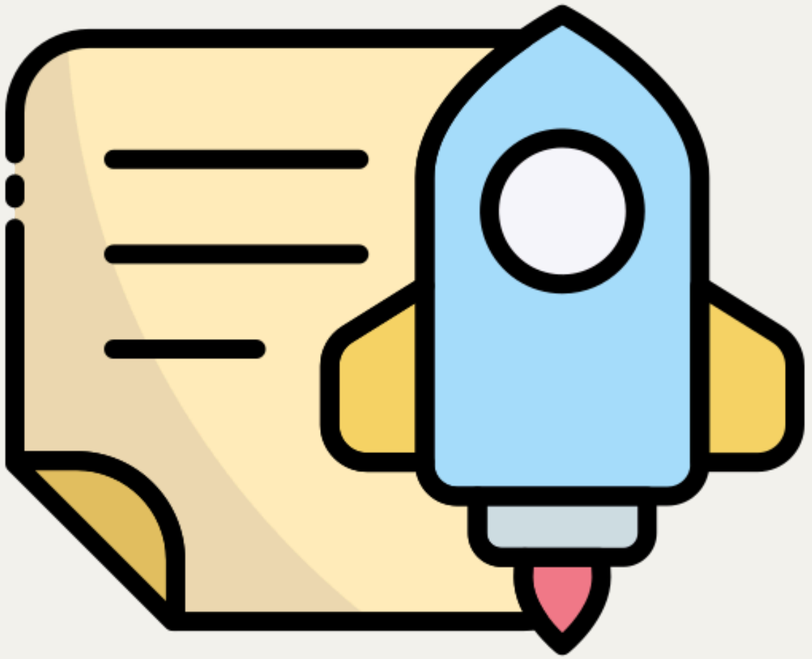
- Implementación de mensajeria residente-conserje
- Bitacora digital de eventos
- Modulo de avisos comunitarios
- Optimización de roles y seguridad

Fase 5 – Documentación y Entrega Final (Semanas 14–16)

- Redacción del informe final
- Presentación del proyecto (diapositivas + demo)
- Entrega del MVP con login, yisitas, paqucteria y notificaciones push
- Revisión y retroallmentación

Infraestructura de Despliegue

El sistema Resident House se implementará bajo un modelo cliente–servidor en la nube, asegurando acceso desde dispositivos móviles y navegadores web.



- Frontend: desarrollado con Ionic + Angular, desplegado como aplicación móvil para residentes y conserjes.
- Backend: gestionado en Firebase (de momento), ofreciendo servicios de autenticación, base de datos en tiempo real y notificaciones push.
- Base de datos: Firestore (Firebase), con capacidad de escalado automático y almacenamiento en la nube.
- Entornos definidos:
  - Desarrollo: para la creación y pruebas internas.
  - Pruebas: validación de funcionalidades en condiciones similares a producción.
  - Producción: despliegue final accesible para usuarios.
- Modelo de despliegue: en la nube, con acceso multicanal (móvil y escritorio mediante navegador).
- Estrategia de despliegue: inicial manual, con la posibilidad de incorporar en el futuro despliegues automatizados (CI/CD).

Escalabilidad y Mantenimiento

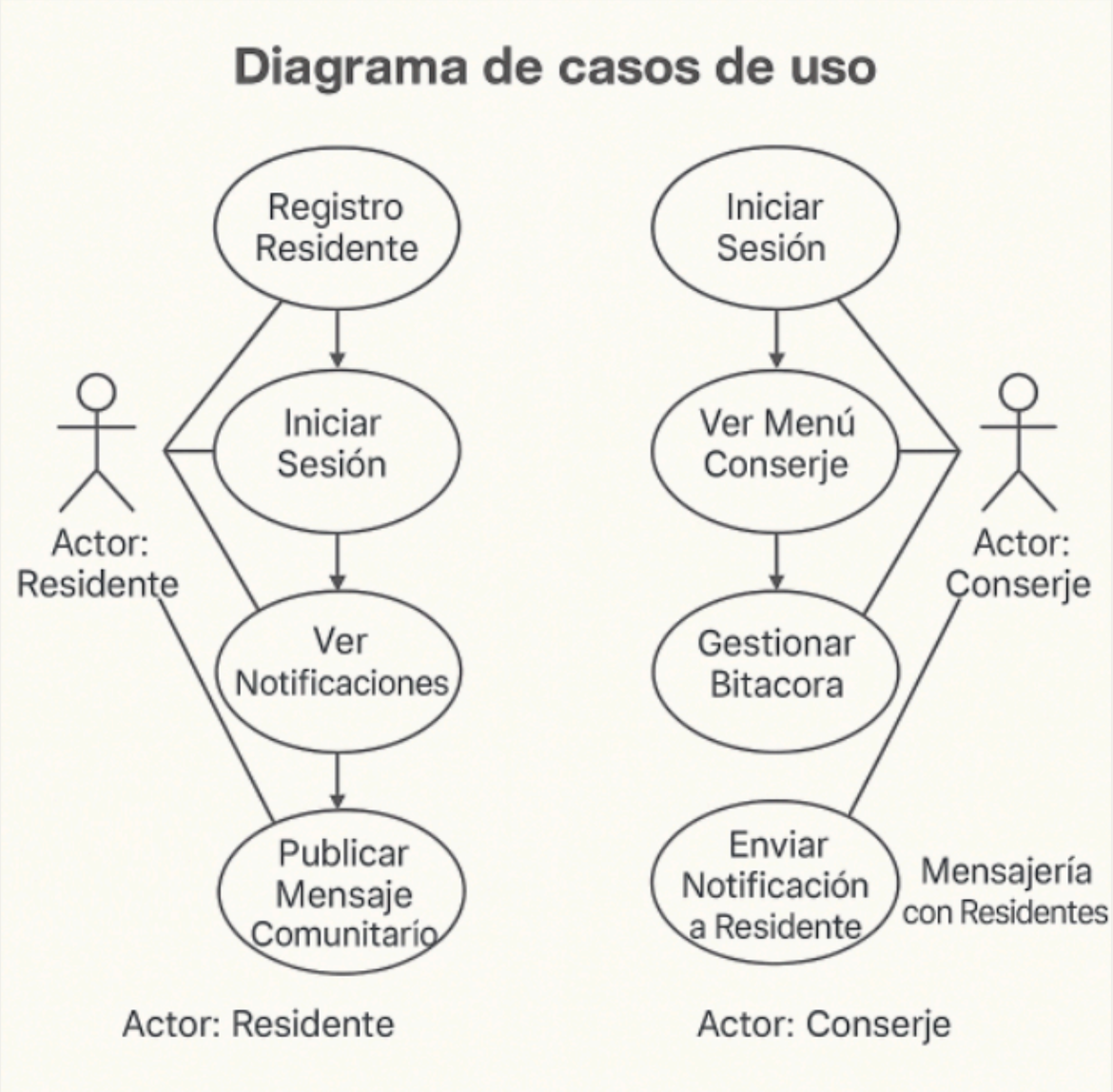
Las funciones de escalabilidad y mantenimiento en el proyecto Resident House se enfocan en garantizar que la aplicación crezca y se mantenga de forma estable en el tiempo.

Escalabilidad:
El sistema, al estar en la nube con Firebase (de momento), permite aumentar progresivamente el número de usuarios (residentes y conserjes) sin afectar el rendimiento.

Mantenimiento:
Se realizarán mejoras continuas, corrigiendo errores y optimizando procesos para mantener la aplicación estable y actualizada.
Plan de recuperación: se considerarán respaldos de datos y estrategias para la rápida recuperación en caso de fallos.
Nuevas funciones: la arquitectura modular facilita añadir características futuras como reportes avanzados o integración con cámaras de seguridad.

Monitoreo: se evaluará constantemente el uso y desempeño del sistema, asegurando que cumpla con los objetivos de seguridad y eficiencia.

DIAGRAMA CASOS DE USO



Modelo BPMN

Modelo Entidad Relacion

Diagrama MVP

Matriz R.a.c.i

Cronograma

Mockups/Diseños.