FICHA TÉCNICA – PROYECTO RESIDENT HOUSE

Versión: 1.0

Fecha: 15 Agosto 2025

Integrantes: Laynus Alexander Ugarte Campos,

Leonardo Castillo,

Cristian Salvo

Profesor Guía: Ricardo Alberto Aravena Videla

Sección: 705D

Sede: Duoc UC – Maipú

# 1. Introducción

El proyecto Resident House corresponde al desarrollo de una aplicación móvil orientada a mejorar la comunicación, el control de accesos y la seguridad en condominios privados compuestos por casas. Actualmente, estos procesos se gestionan mediante llamadas telefónicas, mensajes informales o registros manuales, lo que genera demoras, desorganización y falta de trazabilidad.  
  
La aplicación busca digitalizar los flujos de portería, optimizando la interacción entre residentes, conserjes y administradores, garantizando rapidez, seguridad y registro confiable de cada evento.

# 2. Objetivo de la Solicitud

Elaborar una ficha técnica completa y estructurada que defina el alcance, características técnicas y plan de trabajo del proyecto Resident House, sirviendo como entregable formal de la Fase 1 – Definición del Proyecto APT Capstone.

# 3. Alcance de la Ficha Técnica

Incluye:

- Contexto del problema.  
- Actores, objetivos y funcionalidades principales.  
- Arquitectura tecnológica propuesta.  
- Requerimientos funcionales y no funcionales.  
- Plan de trabajo y cronograma general.  
- MVP y criterios de aceptación.  
- Evidencias esperadas y firmas.

No incluye:

- Manuales de usuario finales.  
- Código fuente o pruebas unitarias.  
- Diseño gráfico definitivo de interfaces.

# 4. Información General del Sistema

|  |  |
| --- | --- |
| Nombre del Proyecto | Resident House |
| Objetivo Principal | Modernizar la comunicación y control de accesos en condominios de casas, digitalizando visitas, paquetería y avisos comunitarios. |
| Usuarios Destinatarios | Residentes, conserjes y administradores de condominios privados. |
| Responsables del Proyecto | Scrum Master: Laynus Ugarte · Líder Técnico: Leonardo [Apellido] · Desarrollo/QA: Cristian [Apellido] |

# 5. Descripción Funcional de Alto Nivel

## 5.1 Actores principales

- Residente: autoriza/rechaza visitas, recibe notificaciones push, consulta bitácora, se comunica con el conserje y publica avisos comunitarios.  
- Conserje: registra visitas y paquetería, controla estados de acceso, mantiene la bitácora.  
- Administrador: gestiona usuarios y unidades, genera reportes y difunde avisos globales.

## 5.2 Funcionalidades principales

|  |  |  |
| --- | --- | --- |
| Código | Funcionalidad | Descripción |
| F01 | Login y roles | Acceso diferenciado por rol: residente, conserje, administrador. |
| F02 | Registro de visitas | Creación y gestión de visitas con estados (pendiente, aprobado, rechazado, consumido). |
| F03 | Registro de paquetería | Control de recepción y entrega de paquetes con notificación al residente. |
| F04 | Autorización de acceso | El residente aprueba o rechaza solicitudes en tiempo real desde su dispositivo. |
| F05 | Mensajería interna | Chat en tiempo real entre residente y conserje. |
| F06 | Avisos comunitarios | Espacio colaborativo para emergencias, objetos perdidos o campañas. |
| F07 | Bitácora digital | Registro auditable de todas las acciones realizadas en el sistema. |
| F08 | Reportes básicos | Consulta por fechas, estados y unidades. |

(Diagrama de Casos de Uso – UML)

Diagrama

El contenido generado por IA puede ser incorrecto.  
👉 (Aquí va Diagrama de Actividad – UML)  
👉 (Aquí va Diagrama BPMN del proceso principal de visitas)

# 6. Reglas de Negocio

- Una visita solo puede transitar entre los estados definidos: pendiente → aprobado/rechazado → consumido/expirado.  
- Los residentes solo pueden gestionar información asociada a su unidad.  
- Todo acceso autorizado o rechazado queda registrado en la bitácora con fecha, hora y responsable.  
- Los conserjes no pueden aprobar visitas en nombre del residente.

# 7. Arquitectura Propuesta

Se plantea una arquitectura cliente–servidor en n-capas, con aplicación móvil construida en Ionic + Angular y un backend serverless en Firebase (de momento, sujeto a cambio).

|  |  |
| --- | --- |
| App móvil (Ionic + Angular) | Interfaz de usuario y lógica de presentación. |
| Firebase Authentication (de momento) | Manejo de usuarios y roles. |
| Firestore (de momento) | Almacenamiento de visitas, paquetería, mensajes y avisos. |
| Firebase Cloud Messaging (de momento) | Envío de notificaciones push. |
| Cloud Functions (de momento) | Validación de flujos y generación de logs de auditoría. |

(Diagrama de Arquitectura) Diagrama

El contenido generado por IA puede ser incorrecto.

# 8. Lenguajes, Frameworks y Librerías

|  |  |
| --- | --- |
| Lenguajes | TypeScript, HTML, CSS |
| Frameworks | Angular, Ionic |
| SDK/Librerías | Firebase (de momento), AngularFire, TailwindCSS (opcional) |
| Herramientas | GitHub, VS Code, Postman, Android Studio |

# 9. Modelo de Datos

👉 (Aquí va el Modelo de Datos — explicación de entidades, atributos y relaciones entre ellas)

# 10. Requerimientos

## 10.1 Funcionales

- RF1: Autenticación y control de acceso por rol.  
- RF2: Registro digital de visitas.  
- RF3: Registro digital de paquetería.  
- RF4: Autorización/rechazo de accesos por parte del residente.  
- RF5: Mensajería residente–conserje.  
- RF6: Publicación y consulta de avisos comunitarios.  
- RF7: Bitácora auditable de eventos.  
- RF8: Generación de reportes básicos.

## 10.2 No Funcionales

- RNF1 – Seguridad comunitaria: solo residentes autorizados pueden acceder al sistema y sus datos.  
- RNF2 – Disponibilidad offline parcial: debe ser posible registrar visitas sin internet, sincronizando después.  
- RNF3 – Escalabilidad modular: el sistema debe admitir nuevas funciones sin rehacer la aplicación.  
- RNF4 – Experiencia de usuario simple: las autorizaciones de visitas deben completarse en menos de 3 pasos.  
- RNF5 – Privacidad de datos: la información de residentes, visitas y paquetería no será visible fuera de la comunidad.

# 11. Seguridad y Cumplimiento

- Acceso diferenciado por rol.  
- Cifrado de datos en tránsito mediante HTTPS.  
- Reglas de acceso restringido por unidad en la base de datos.  
- Registro en bitácora de eventos críticos.  
- Cumplimiento con normativa de protección de datos personales (Ley 19.628 – Chile).

# 12. Plan de Trabajo y Cronograma

Fases:  
1. Semanas 1–2: Definición del proyecto, kickoff y casos de uso.  
2. Semanas 3–6: Diseño de arquitectura y primer prototipo (MVP-1).  
3. Semanas 7–10: Desarrollo de funcionalidades avanzadas (MVP-2).  
4. Semanas 11–13: Pruebas, auditoría y mejoras.  
5. Semanas 14–16: Documentación y entrega final.

👉 (Aquí va Cronograma Gantt)

Roadmap

Escala de tiempo

El contenido generado por IA puede ser incorrecto.

# 13. MVP – Producto Mínimo Viable

(El MVP es el Producto Mínimo Viable: la versión inicial del sistema que incluye solo las funcionalidades esenciales para demostrar su utilidad.)  
  
El MVP debe incluir:  
- Autenticación y login diferenciado.  
- Registro de visitas y paquetería.  
- Autorización de accesos en tiempo real.  
- Notificaciones push al residente.  
- Bitácora básica de eventos.

👉 (Aquí va Definición detallada de criterios de aceptación del MVP)

# 14. Matriz RACI

👉 (Aquí va la Matriz RACI — roles y responsabilidades)

# 15. Gestión de Riesgos

|  |  |  |
| --- | --- | --- |
| Riesgo | Impacto | Mitigación |
| Cambio de plataforma (Firebase vetado) | Medio | Plan alternativo con Node.js + MongoDB + JWT. |
| Retraso en desarrollo por coordinación del equipo | Alto | Sprints cortos, comunicación semanal. |
| Fallas en notificaciones push | Medio | Reintentos automáticos y registro en bitácora. |
| Cambios en requisitos de seguridad | Medio | Validación temprana con profesor guía. |

# 16. Evidencias Esperadas

- Documento de ficha técnica (este archivo).  
- Diagramas UML, BPMN y ER.  
- Mockups de pantallas principales.  
- Cronograma Gantt.  
- MVP desplegado con login, visitas, paquetería y notificaciones.  
- Informe y presentación final.

# 17. Firmas

Se solicita que la ficha técnica sea entregada en un plazo no mayor a \_\_\_\_\_\_\_\_ días hábiles desde la recepción de la presente solicitud.  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Firma del solicitante

# 18. Anexos

👉 (Aquí va Diagrama de Casos de Uso – UML)  
👉 (Aquí va Diagrama de Actividad – UML)  
👉 (Aquí va Diagrama BPMN – Proceso de Visitas)  
👉 (Aquí va Diagrama de Clases – UML)  
👉 (Aquí va Diagrama Entidad–Relación – ER)  
👉 (Aquí va Diagrama de Arquitectura)  
👉 (Aquí va Cronograma Gantt)  
👉 (Aquí va Mockups de interfaz)