

Especificación de Requisitos de Software (ERS)

Conforme al estándar IEEE Std 830-1998

Proyecto:

ConsorcioApp — Cliente Android

(nuevo repositorio: app Android + API compartida)

Fenix Software — ISPC

2026

Ficha del documento

Fecha	Revisión	Autor	Referencia de cambio
12/09/2026	1	Equipo completo	Creación de este ERS en un repositorio nuevo, dedicado exclusivamente al cliente Android (Java, JDK 17, compileSdk 34) y a los requisitos de la API que consume. El frontend web (Angular) mantiene su propia documentación completa en el repositorio original y no se duplica aquí.
14/10/2026	2	Equipo completo	Sincronización con el estado real del proyecto: RNF01 (minSdk 21), HU-02 (Activity implementada), HU-03 (SMTP y contraseña aleatoria), Backlog (endpoints y estados), Inconvenientes del Sprint 1, alcance (ajustes de backend), HU-12 (endpoint mi_perfil), CA-03.2 y RNF03 (medidas de seguridad), manejo del token JWT, URL de Swagger, arquitectura y estado de las tareas de backend.

Nota sobre el alcance de este documento

ConsortioApp ya cuenta con un frontend web funcional en Angular, completamente documentado (Requisitos, Historias de Usuario, Backlog y Actas) en el repositorio original del proyecto. Este documento pertenece a un repositorio nuevo, creado para desarrollar un cliente Android nativo para el mismo sistema. Para evitar duplicar y desincronizar información, este ERS cubre únicamente: (a) los requisitos de backend/API compartidos, incluidos aquí porque el cliente Android depende de ellos, y (b) los requisitos, historias de usuario y backlog específicos del cliente Android. El frontend web no se detalla en este documento; ver el repositorio original para su documentación completa.

Contenido

Nota: en Word, hacer clic derecho sobre el índice y elegir "Actualizar campos" para generar los números de página.

1. Introducción

Este documento es una Especificación de Requisitos de Software (ERS) para el cliente Android de ConsorcioApp, una plataforma de gestión de consorcios que centraliza la comunicación entre el Administrador y los residentes de un edificio. Se estructura según el estándar IEEE Práctica Recomendada para Especificaciones de Requisitos de Software ANSI/IEEE 830-1998.

1.1 Propósito

El presente documento define las especificaciones funcionales y no funcionales necesarias para el desarrollo del cliente Android de ConsorcioApp, y describe los requisitos de la API de backend (ya existente) de la que este cliente depende. Sirve como referencia técnica para el equipo de desarrollo y como base de comunicación con el cuerpo docente/evaluador de este repositorio.

1.2 Alcance

El sistema gestiona un único consorcio, administrado por un Administrador de Consorcio. El cliente Android de este repositorio debe permitir:

- Inicio de sesión con username y contraseña, obteniendo un token JWT de la API.
- Un dashboard adaptado al rol (Administrador o Residente).
- Alta de residentes vinculados a una unidad funcional (piso y departamento), desde el rol Administrador.
- Creación, seguimiento y cambio de estado de reclamos.
- Publicación y visualización de comunicados.
- Cambio de contraseña desde el perfil del residente.

Quedan fuera del alcance de este documento y de este repositorio:

- El frontend web en Angular: se mantiene y se sigue desarrollando en su propio repositorio, con su propia documentación.
- Cambios al modelo de datos del backend (se reutiliza el existente). Las reglas de negocio y de seguridad del backend sí se ajustaron en este repositorio donde el cliente Android lo requirió real y reserv a de amenities.

1.3 Personal involucrado

El equipo de este repositorio (consorcioMobile) se reorganizó respecto del equipo original: Gisela D'Andrea abandonó el proyecto. Se incorporaron Joaquín Romero y Nicolás Elías Calmucci como oyentes cruzados entre Proyecto Integrador II y Programación de Aplicaciones Móviles: cada uno cursa solo una de las dos materias, por lo que entre ambos equivalen a la participación de un integrante regular.

Nombre	Rol en el proyecto	Módulo asignado	Contacto
Nahuel Santiago CUFRE	Desarrollador/ CiberSec	Módulo Login	nahuucufre@gmail.com
Luis Gerardo Catalas	Desarrollador / Testing / Cybersecurity	Módulo Contactarnos	gerardocatalas@mi.unc.edu.ar
Luis Gastón Alonso	Desarrollador / Testing / Cybersecurity /	Módulo Perfil de Usuario	https://github.com/t0ng42013

Joaquin Romero	Desarrollador / Testing / Cybersecurity (oyente - solo cursa Aplicaciones Móviles)	Módulo Dashboard / Residente	joaquin.romero89@gmail.com www.github.com/JDR89
Ivan Ignacio Moreno Rivero	Desarrollador/Testin g	Módulo Comunicados	iv4n.m0reno@gmail.com
Verónica Cecilia Vargas	Desarrolladora/Testi ng	Módulo Reclamos	vargas.veronica55@gmail.com
Nicolás Elías Calmucci	Oyente(solo cursa Proyecto Integrador II)	Módulo Dashboard / Residente (apoyo)	calmuccin@gmail.com

1.4 Definiciones, acrónimos y abreviaturas

Término	Descripción
MVP	Producto Mínimo Viable (Minimum Viable Product)
ERS	Especificación de Requisitos de Software
RF / RNF	Requerimiento Funcional / Requerimiento No Funcional
HU	Historia de Usuario
CA	Criterio de Aceptación
SP	Story Point (unidad de estimación de esfuerzo)
JWT	JSON Web Token, estándar de autenticación de sesión provisto por la API
SMTP	Simple Mail Transfer Protocol, protocolo de envío de correo electrónico
DRF	Django REST Framework, framework con el que está construida la API
JDK 17	Java Development Kit versión 17, entorno de desarrollo Java de la app Android
compileSdk	Versión del SDK de Android contra la que se compila el proyecto (34 en este repositorio)
minSdk / targetSdk	Versión mínima soportada y versión objetivo de Android de la app
Activity / Fragment	Componentes de pantalla de una aplicación Android con vistas XML
Gradle	Sistema de build utilizado para compilar el proyecto Android

Unidad funcional	Unidad habitacional (piso y departamento) a la que se asocia un residente
------------------	---

1.5 Referencias

Título del documento	Referencia
IEEE Recommended Practice for Software Requirements Specifications	IEEE Std 830-1998
Repositorio y documentación del proyecto web original (Angular + Django)	https://github.com/Fenix-Software-2026/ConsortioApp
Documentación de la API (Swagger UI)	https://gaston.alwaysdata.net/api/docs
Android Developers Documentation	https://developer.android.com/docs
Java SE 17 Documentation	https://docs.oracle.com/en/java/javase/17/

1.6 Resumen

El resto del documento se organiza así: la sección 2 describe el producto desde la perspectiva del cliente Android y la API de la que depende; la sección 3 detalla los RF y RNF vigentes (backend, reutilizados sin cambios), las Historias de Usuario reformuladas para su implementación en Android, y el Backlog/planificación de sprints específico de este repositorio; el Anexo I incluye enlaces de referencia y materiales de diseño pendientes.

2. Descripción general

2.1 Perspectiva del producto

CConsortioApp mantiene una arquitectura Full Stack desacoplada. El backend es una API REST desarrollada en Django REST Framework, con persistencia en MySQL, documentada automáticamente mediante drf-spectacular (OpenAPI 3.0, con interfaces Swagger UI y ReDoc). La autenticación se resuelve mediante tokens JWT. Esta API ya existe, fue construida para el frontend web en Angular, y este repositorio la reutiliza sin modificarla.

El producto de este repositorio es un cliente Android nativo, desarrollado en Java (JDK 17, compileSdk 34) con vistas XML tradicionales (Activities/Fragments), que consume la misma API REST. El frontend web en Angular sigue existiendo como cliente independiente de la misma API, pero su desarrollo y documentación no forman parte de este repositorio.

Resuelto parcialmente (14/10/2026): la arquitectura MVVM está aplicada por ahora solo en el módulo Perfil (MiPerfilViewModel); el resto de los módulos, incluido Alta de Residentes, consume la API directamente desde la Activity, y extender MVVM queda sujeto al avance del equipo. Networking: Retrofit con OkHttp y conversor Gson (un interceptor agrega el token JWT a cada llamada).

2.2 Características de los usuarios

Tipo de usuario	Formación requerida	Actividades
Equipo de Desarrollo / Súper Admin	Conocimientos técnicos de administración de sistemas y bases de datos	Gestión del backend, la base de datos y creación inicial del Administrador del Consorcio
Administrador del Consorcio	Manejo básico de aplicaciones móviles	Alta de residentes, gestión de comunicados y resolución centralizada de reclamos desde la app Android
Residente	Manejo básico de aplicaciones móviles, sin conocimientos técnicos previos	Reporta reclamos propios y visualiza comunicados desde la app Android

2.3 Restricciones

Backend / API (existente, reutilizado sin cambios):

- Django 6.0.4 + Django REST Framework, base de datos MySQL.
- Documentación de API vía drf-spectacular (OpenAPI 3.0 / Swagger / ReDoc).
- La creación del Administrador del Consorcio solo puede hacerse desde el panel /admin de Django (RF01).

Cliente Android (alcance de este repositorio):

- Lenguaje: Java, JDK 17.
- compileSdk: 34 (Android 14).
- UI: vistas XML tradicionales (Activities/Fragments), sin Jetpack Compose.
- Consumo de la API REST de Django mediante HTTP.(<https://gaston.alwaysdata.net>)

3. Requisitos específicos

Los RF y RNF listados a continuación describen el comportamiento del sistema (API + cliente) heredado del proyecto original. En este repositorio, todo lo relativo a "la interfaz" o "el cliente" refiere específicamente a la app Android.

3.1 Requerimientos funcionales

ID	Nombre	Descripción	Prioridad	Actor
RF01	Inicialización del Administrador	El sistema debe permitir la creación del usuario único con rol Administrador exclusivamente a través del panel de administración de Django (/admin).	MUST	Equipo de Desarrollo
RF02	Alta de Residentes y Envío de credenciales	El sistema debe permitir al Administrador dar de alta un único usuario residente responsable por unidad funcional, vinculándolo a una unidad preexistente (Piso y Departamento). Al crearse, el sistema debe proveerle sus credenciales provisorias.	MUST	Administrador
RF03	Inicio de sesión y JWT	El sistema debe autenticar usuarios (Admin/Residente) con email y contraseña correctos, retornando un token JWT para la sesión.	MUST	Usuario Registrado
RF04	Control de acceso por roles	La app Android debe restringir o habilitar sus pantallas y acciones según el rol del token (Admin vs Residente).	MUST	Sistema
RF05	Crear reclamo con adjunto	El residente debe poder crear un reclamo con título, descripción y categoría desde la app Android. El sistema registrará la fecha_creacion automáticamente. Estado inicial: "Pendiente".	MUST	Residente
RF06	Listado de reclamos e Historial	La app Android debe listar reclamos paginados (10 por página). El Admin ve todos los del consorcio; el Residente solo los propios. Filtros por estado y categoría.	MUST	Ambos Roles
RF07	Ver detalle de reclamo	La app Android debe mostrar el detalle completo de un reclamo junto con su historial y la fecha_actualizacion correspondiente.	MUST	Ambos Roles
RF08	Cambiar estado de reclamo	El administrador debe poder actualizar, desde la app Android, el campo estado de cualquier reclamo siguiendo el flujo: Pendiente → En Proceso → Resuelto → Archivado.	MUST	Administrador

RF09	Publicar comunicados	El administrador debe poder crear, editar y eliminar comunicados desde la app Android, definiendo titulo, contenido, fecha_publicacion y si es_urgente (booleano).	MUST	Administrador
RF10	Ver comunicados en Feed	El residente debe poder visualizar los comunicados vigentes en la pantalla principal de la app Android, ordenados por fecha_publicacion.	MUST	Residente
RF11	Dashboard adaptado por rol	La app Android debe mostrar una pantalla principal con métricas/accesos rápidos específicos para el Admin y un feed simplificado para el Residente.	MUST	Ambos Roles
RF12	Módulo Perfil de Usuario	El residente debe poder ver su nombre de usuario y modificar, desde la app Android, la contraseña provisoria por una nueva clave segura.	SHOULD	Residente
RF13	Módulo Contactarnos	La app Android debe ofrecer una pantalla estática con los datos de contacto de la administración (teléfono, correo electrónico y horario de atención), sin formulario ni interacción con la API.	MUST	Residente

3.1.1 Requerimientos no funcionales

ID	Nombre	Categoría	Descripción	Prioridad
RNF 01	Compatibilidad Android	Compatibilidad	La app debe compilar con compileSdk 34 y funcionar correctamente desde Android 5.0 (minSdk 21, mínimo exigido por la cátedra en el ABP).	MUST
RNF 02	Usabilidad e Interfaz	Usabilidad	La interfaz debe seguir principios de diseño limpios y consistentes, garantizando una navegación intuitiva para la gestión de reclamos y comunicados.	SHOULD
RNF 03	Seguridad y Privacidad	Seguridad	El sistema debe garantizar la confidencialidad de los datos mediante el hashing de contraseñas en la base de datos y la restricción de endpoints según el rol del JWT. Medidas implementadas en el backend (14/10/2026): SECRET_KEY obligatoria desde variable de entorno (mínimo 50 caracteres); ALLOWED_HOSTS obligatorio cuando DEBUG está El sistema debe garantizar la confidencialidad de los datos mediante el hashing de contraseñas en la base de datos y la restricción de endpoints según el rol del JWT. Medidas implementadas en el backend	MUST

			(14/10/2026): SECRET_KEY obligatoria desde variable de entorno (mínimo 50 caracteres); ALLOWED_HOSTS obligatorio cuando DEBUG está	
RNF 04	Accesibilidad Operativa	Usabilidad	El sistema debe ser operable por usuarios finales (residentes) sin requerir conocimientos técnicos previos ni capacitaciones avanzadas.	SHOULD
RNF 05	Trazabilidad (Auditoría)	Auditoría	El sistema debe registrar de forma interna la identidad del usuario que realiza acciones críticas, como la creación de comunicados o los cambios de estado en los reclamos.	MUST

3.2 Historias de Usuario (cliente Android)

Historias de usuario reformuladas para su implementación en la app Android. Los criterios de aceptación (CA) del backend se mantienen sin cambios; los de interfaz se redactan en términos de pantallas Android.

Alcance del Sprint 1: todos los módulos marcados como "Hecho" en el Backlog corresponden únicamente a la creación de la interfaz (Activity/layout) de cada pantalla, sin conexión funcional a la API todavía (sin llamadas HTTP reales, sin manejo de JWT, sin persistencia de datos cargados por el usuario). La integración funcional completa de cada módulo con el backend se planifica para el Sprint 2 de este repositorio (ver 3.4).

HU-01 | Inicialización del Administrador del Consorcio

Como Equipo de Desarrollo, quiero dar de alta al administrador único en el panel de Django (/admin), para habilitar el acceso inicial a la gestión del consorcio.

- CA-01.1 El registro se realiza exclusivamente desde el backend protegido de Django (/admin).
- CA-01.2 Se debe ingresar obligatoriamente username, password y asignar el rol de "Administrador".
- CA-01.3 El sistema aplica hashing automático sobre la contraseña antes de guardarla.

Story Points: 2 | Prioridad: Must | Rol: Equipo de Desarrollo

HU-02 | Alta de Residentes y Vinculación

Como Administrador del Consorcio, quiero registrar a un nuevo residente y asociarlo a un departamento, desde la app Android, para habilitar su acceso a la plataforma y mantener actualizado el padrón del edificio.

- CA-02.1 Pantalla con campos obligatorios: nombre, apellido, nombre de usuario, email y selección de la unidad preexistente (piso y departamento); solo se ofrecen las unidades libres.
- CA-02.2 El sistema valida que el email sea único y que la unidad no esté ocupada por otro usuario activo (relación 1 a 1).
- CA-02.3 Al guardar, el usuario se crea con el rol de "residente".

Story Points: 5 | Prioridad: Must | Rol: Administrador

HU-03 | Provisión de Credenciales al Residente

Como Sistema, quiero notificar al nuevo residente sus credenciales de acceso provisorias, para que pueda ingresar por primera vez a la app.

- CA-03.1 El disparador ocurre inmediatamente después de que el Administrador ejecuta con éxito la HU-02.
- CA-03.2 El backend no expone la contraseña provisoria en la respuesta de creación: la genera de forma aleatoria (10 caracteres con minúscula, mayúscula, número y símbolo), la guarda hasheada y la envía por email al residente.
- CA-03.3 El envío de las credenciales se resuelve en el backend: Django envía el mail por SMTP al residente recién creado (implementado el 23/09/2026).

Story Points: 5 | Prioridad: Must | Rol: Sistema

HU-04 | Inicio de Sesión y Autenticación JWT

Como Usuario Registrado (Admin/Residente), quiero ingresar mis credenciales en la pantalla de Login de la app Android, para acceder de forma segura a mi panel correspondiente.

- CA-04.1 Pantalla de Login (Activity) con campos username/email y password.
- CA-04.2 El backend valida las credenciales contra la base de datos utilizando el hashing correspondiente.
- CA-04.3 Autenticación exitosa → el backend retorna un token JWT válido que almacena el rol y el id del usuario para manejar la sesión.

Story Points: 5 | Prioridad: Must | Rol: Usuario Registrado

HU-05 | Control de Navegación por Roles

Como Sistema, quiero leer el rol contenido en el token JWT, para restringir o permitir el acceso a las pantallas de la app según corresponda.

- CA-05.1 Si el usuario tiene rol "Administrador", visualiza las opciones de gestión y el panel administrativo.
- CA-05.2 Si el usuario tiene rol "Residente", se le muestra el feed de comunicados y sus reclamos propios.
- CA-05.3 Si un residente intenta acceder a una pantalla de administrador, el sistema debe bloquear la navegación y redirigirlo.

Story Points: 3 | Prioridad: Must | Rol: Sistema

HU-06 | Creación de Reclamos

Como Residente, quiero reportar un problema del edificio mediante un formulario simple en la app Android, para que el Administrador tome conocimiento y pueda resolverlo.

- CA-06.1 Formulario con campos título, descripción y un selector (Spinner) para la categoría (ej: Plomería, Electricidad, Gas).
- CA-06.2 Al enviar, el sistema setea automáticamente el estado en "Pendiente", guarda la fecha_creacion del día y vincula el reclamo al usuario y a su unidad.

Story Points: 5 | Prioridad: Must | Rol: Residente

HU-07 | Historial de Reclamos Propios

Como Residente, quiero ver un listado de los reclamos que cargué en el sistema, para hacer el seguimiento del estado de mis solicitudes.

- CA-07.1 Vista tipo lista (RecyclerView) que muestra título, categoría, estado y fecha_creacion.
- CA-07.2 El cliente filtra la consulta al backend enviando el ID del residente, asegurando que solo vea sus propios registros.
- CA-07.3 Los reclamos permanecen visibles e inalterables para el residente en modo lectura.

Story Points: 3 | Prioridad: Must | Rol: Residente

HU-08 | Panel de Control Global de Reclamos

Como Administrador del Consorcio, quiero visualizar la lista completa de todos los reclamos hechos en el edificio, para centralizar las tareas de mantenimiento y priorizar las urgencias.

- CA-08.1 Listado (RecyclerView) que muestra todos los reclamos existentes en la base de datos sin distinción de usuario.
- CA-08.2 Se muestra título, categoría, estado, fecha_creacion y la unidad (piso/depto) de origen.
- CA-08.3 Se incluyen filtros rápidos para segmentar por estado (ej: ver solo los "Pendientes").

Story Points: 5 | Prioridad: Must | Rol: Administrador

HU-09 | Gestión y Cambio de Estado de Reclamos

Como Administrador del Consorcio, quiero actualizar el estado de los reclamos a medida que se gestionan, para mantener informados a los residentes sobre el avance de sus soluciones.

- CA-09.1 Desde el detalle de cualquier reclamo, el Administrador tiene un control (Spinner) para modificar el campo estado.
- CA-09.2 Los estados permitidos siguen el flujo secuencial: Pendiente → En Proceso → Resuelto → Archivado.
- CA-09.3 Al confirmar el cambio, el backend actualiza automáticamente fecha_actualizacion con la hora del servidor.

Story Points: 3 | Prioridad: Must | Rol: Administrador

HU-10 | Publicación de Comunicados

Como Administrador del Consorcio, quiero redactar comunicados dirigidos a todo el edificio, para informar sobre asambleas, cortes de servicios o avisos importantes.

- CA-10.1 Formulario con campos obligatorios título, contenido y un control booleano para marcar si es urgente.
- CA-10.2 Al guardar, el backend asigna la fecha_publicacion del día y vincula el registro con el ID del administrador que lo publica.
- CA-10.3 El nuevo comunicado queda disponible en el feed de los residentes.

Story Points: 4 | Prioridad: Must | Rol: Administrador

HU-11 | Feed de Novedades del Edificio

Como Residente, quiero visualizar un feed de comunicados en mi pantalla principal, para estar al tanto de las novedades y avisos importantes del consorcio.

- CA-11.1 Sección destacada (RecyclerView) que renderiza los comunicados existentes.
- CA-11.2 Los comunicados se muestran ordenados de forma descendente por fecha_publicacion (los más nuevos primero).
- CA-11.3 Los comunicados con es_urgente en verdadero deben resaltar visualmente (ej. color de fondo o etiqueta de alerta).

Story Points: 3 | Prioridad: Must | Rol: Residente

HU-12 | Módulo Perfil de Usuario

Como Residente, quiero ver mi nombre de usuario y modificar mi contraseña desde una pantalla de perfil, para gestionar mis propios datos de acceso dentro de la app.

- CA-12.1 Pantalla de perfil que muestra el nombre del usuario autenticado.
- CA-12.2 Formulario para ingresar contraseña actual y nueva contraseña.
- CA-12.3 El backend valida la contraseña actual antes de aplicar el cambio (endpoint PATCH /api/usuario/mi_perfil/).
- CA-12.4 La nueva contraseña se guarda hasheada, igual que en el alta inicial.
- CA-12.5 Incluye un botón "Cerrar Sesión" que finaliza la sesión del usuario y lo devuelve a la pantalla de Login.
- CA-12.6 Si current_password está ausente o es incorrecta, el backend rechaza el cambio con un error de validación y conserva la contraseña anterior.
- CA-12.7 La aplicación muestra el resultado de la operación sin exponer current_password ni new_password en respuestas, logs o mensajes de error.
-

Story Points: 3 | Prioridad: Should | Rol: Residente

HU-13 | Módulo Contactarnos

Como Residente, quiero acceder a una pantalla con los datos de contacto de la administración, para poder comunicarme por teléfono o correo electrónico ante cualquier duda o urgencia.

- CA-13.1 Pantalla estática ("¿Necesitás ayuda?") accesible desde la app, sin formulario ni envío de datos.
- CA-13.2 Muestra teléfono de contacto, correo electrónico y horario de atención.
- CA-13.3 Incluye un botón "Volver" para regresar a la pantalla anterior.
- CA-13.4 Los datos de contacto no provienen de un endpoint de la API: son informativos y quedan fijos en la app (o configurables, a confirmar).

Story Points: 1 | Prioridad: Must (exigido por la cátedra) | Rol: Ambos Roles

3.3 Backlog

Tareas de backend (ya resueltas en el proyecto original; se listan como referencia porque la app Android depende de ellas): Actualización (14/10/2026): las tareas que figuraban pendientes se completaron en este repositorio. El cliente Android consume POST /api/login/, GET/POST /api/reclamos/, PATCH /api/reclamos/{id}/ (cambio de estado), GET/POST /api/comunicados/, GET/PUT/DELETE /api/comunicados/{id}/, GET /api/unidad/, GET/POST /api/usuario/ y GET/PATCH /api/usuario/mi_perfil/ .

ID Tarea	Descripción	SP	Estado
HU01-T1	Configurar modelos de Django según el DER (Usuario, Unidad)	1	Hecho 
HU01-T2	Crear SúperUsuario inicial (admin_consortio) vía comandos	1	Hecho 
HU02-T1	Endpoint POST /api/usuario/ (Solo Admin)	2	Hecho 
HU03-T1	Configurar librería SMTP en settings.py de Django	2	Hecho 
HU03-T2	Señales (signals) para envío automático de mail	3	Hecho 
HU04-T1	Autenticación Django y generación de JWT	2	Hecho 
HU06-T1	Modelo Reclamo con fechas y FKs	2	Hecho 
HU06-T2	Endpoint POST /api/reclamos/ con validación de rol	2	Hecho 
HU07-T1	Endpoint GET /api/reclamos/{id}/ (filtro por ID)	1	Hecho 
HU08-T1	Endpoint GET /api/reclamos/ (Solo Admin)	1	Hecho 
HU09-T1	Endpoint PATCH /api/reclamos/{id}/cambiar-estado/	1	Hecho 
HU10-T1	Modelo Comunicado con contenido y es_urgente	1	Hecho 
HU10-T2	Endpoints CRUD para /api/comunicados/ (Solo Admin)	2	Hecho 
HU11-T1	Endpoint GET /api/comunicados/activos/ (orden descendente)	1	En Progreso 
HU12-T1	Endpoint PATCH /api/usuario/mi_perfil/	2	Hecho 

Tareas de frontend Android — estado real relevado del tablero GitHub Projects del repositorio consorcioMobile (columnas: Backlog / Ready / In progress / Testing-QA / Done):

Issue #	Descripción	HU / Módulo	Responsable	Estado
#5	Crear/Configurar la Organización en GitHub	Infraestructura	Equipo	Hecho 
#6	Configurar el repositorio del proyecto	Infraestructura	Equipo	Hecho 
#7	Implementar estructura de Branching	Infraestructura	Equipo	Hecho 

#8	Configurar tablero Kanban	Infraestructura	Equipo	Hecho
#1	Implementar módulo Mi Perfil	HU-12 (Perfil de Usuario)	Alonso	Hecho
#12	Implementar módulo Login	HU-04	Cufre	Hecho
#2	Módulo Comunicados — Desarrollo vista Residente	HU-11	Moreno Rivero	Hecho
#3	Navegación: Vista Comunicados Residente	HU-11	Moreno Rivero	Hecho
#13	Implementar módulo Mis Reclamos	HU-06 / HU-07	Vargas	Hecho
#14	Implementar Dashboard Owner - Residente	HU-08 / HU-11	Romero / Calmucci	Hecho
#16	Implementar módulo Contactarnos	HU-13 (confirmado: pantalla estática de contacto)	Catalas	Hecho (pantalla estática; no requeriría más trabajo salvo que decidan hacer los datos dinámicos)
#4	Documentación y Casos de Uso: Módulo Comunicados Residentes	HU-11	Moreno Rivero	Hecho
#9	Crear y organizar Issues	General	Equipo completo	Hecho
#10	Crear y completar la Wiki	General	Equipo completo	Hecho
#11	Revisión final del repositorio - "Sprint 0 y 1"	General	Equipo completo	Cerrada el 30/09/2026 Hecho
#26	Ceremonia - Planning Poker - Evidencia 5	General	Equipo completo	Backlog
(a crear)	Pantalla de alta de residentes (rol Administrador)	HU-02	Alonso	Hecho
(a crear)	Mecanismo explícito de control de acceso por rol	HU-05	[COMPLETAR]	Falta crear Issue
(a crear)	Control de cambio de estado de reclamo (vista Admin)	HU-09	Vargas	Hecho
HU-10	Formulario de publicación de comunicados (vista Admin)	HU-10	Moreno Rivero	Hecho
#53	Modelo Comunicado y endpoints del CRUD	HU-10 / RF09	Moreno Rivero	Hecho

#54	Listado de comunicados del residente	HU-11 / RF10	Moreno Rivero	Hecho 
#55	Vista de detalle de comunicado	HU-11 / RF10	Moreno Rivero	Hecho 
#56	Historial dinámico en el panel admin - comunicados	HU-10 / RF09	Moreno Rivero	Hecho 
#57	Publicar comunicados desde el admin (alta)	HU-10 / RF09	Moreno Rivero	Hecho 
#58	Editar y borrar comunicados (admin)	HU-10 / RF09	Moreno Rivero	Hecho 
#59	Acceso al panel de comunicados del administrador	HU-05 / RF04	Moreno Rivero	Hecho 
#60	AUT-UNIT-01: Tests unitarios del módulo Comunicados	HU-11 / QA	Moreno Rivero	Hecho 
#61	AUT-NET-01: Tests de integración de red (MockWebServer)	HU-11 / QA	Moreno Rivero	Hecho 
#65	Acceso a comunicados sin token de autenticación	HU-05 / RNF03	Moreno Rivero	Hecho 
#82	ACC-01 a ACC-13: hallazgos de accesibilidad de Comunicados	HU-11 / RNF04	Moreno Rivero	Hecho 

3.4 Sprints

El proyecto ya atravesó dos sprints en su repositorio web original (Sprint 1: fundaciones del sistema, 16/04–20/04/2026; Sprint 2: funcionalidades principales sobre Angular y Django, planificado 21/04/2026, backlog técnico detallado con cierre 25/05–01/06/2026). Ese historial se documenta en el repositorio original y no se repite aquí; a continuación se planifica el primer sprint de este repositorio, dedicado al cliente Android. Avance (14/10/2026): además de Perfil y Alta de Residentes, el módulo de Comunicados quedó integrado de forma funcional con la API: feed del residente ordenado por fecha_publicacion, detalle de comunicado, alta desde el panel admin (POST), edición y borrado (PUT/DELETE) y control de acceso al panel. Ver Issues #53 a #59.

Sprint 1 de este repositorio — Cliente Android (Interfaces)

N° de sprint	1 (de este repositorio)
Sprint Backlog	Construcción de las interfaces (Activities/layouts) de los módulos Perfil, Login, Comunicados, Reclamos, Dashboard y Contactarnos, sin integración funcional a la API todavía
Responsabilidades	Ver tabla de Personal Involucrado (1.3): cada integrante tiene un módulo asignado dentro de consorcioMobile
Calendario	Planning: semana del 09/09/2026 (aprox.) — Cierre: la mayoría de los módulos se cerraron el 15/09/2026 (ver Issues #1-#16 en GitHub).
Inconvenientes	(1) Seguridad: el backend crea a los residentes con una contraseña fija por defecto ("123456") en vez de una generada aleatoriamente; queda pendiente resolverlo antes de avanzar (relacionado con RNF03 y con HU-03). (2) Indefinición de alcance: no quedó claro desde el Planning si este Sprint debía cubrir solo la vista Residente o también la de Administrador, lo que generó incertidumbre en la asignación de tareas — ver placeholder de HU-02/HU-09/HU-10 en el Backlog. (3) HU-03 (envío de credenciales por SMTP) sigue sin implementarse, arrastrado desde el repositorio original. Actualización (13/10/2026): los puntos (1) y (3) quedaron resueltos: el backend genera una contraseña aleatoria segura y envía las credenciales por SMTP (commits del 23/09/2026)

Sprint 2 de este repositorio (propuesto) — Integración funcional

N° de sprint	2 (de este repositorio)
Sprint Backlog	Conectar cada Activity ya maquetada con la API real: Login funcional (JWT), Perfil (traer username, cambiar contraseña, logout), Reclamos (crear/listar vía endpoints), Comunicados (feed real ordenado por fecha), Dashboard con datos dinámicos por rol; además, crear e implementar los Issues faltantes de HU-02, HU-05, HU-09 y HU-10 Avance (13/10/2026): Perfil (cambio de contraseña y cierre de sesión) y Alta de Residentes (HU-02) ya están implementados y conectados a la API.

Responsabilidades	Se mantiene la asignación de módulos establecida en el Sprint 1 (ver sección 1.3). Cada desarrollador es responsable de la integración de la UI de su módulo con la API REST, la gestión del token JWT correspondiente, el manejo de errores HTTP y la creación de las pruebas unitarias y de integración exigidas por QA
Calendario	Planning: 16/09/2026. Ejecución y Testing: 17/09/2026 al 12/10/2026. Cierre y Release: 13/10/2026 al 14/10/2026
Inconvenientes	Ciberseguridad: Durante la integración de los módulos, se detectaron vulnerabilidades en el backend original (escalamiento de privilegios en el Perfil y falta de validación de host). Se debió reasignar esfuerzo del equipo móvil hacia el backend para blindar los serializadores (evitando el cambio de rol/unidad) y configurar ALLOWED_HOSTS de forma segura. (2) Pruebas Automatizadas: La curva de aprendizaje inicial para implementar MockWebServer y pruebas instrumentadas en Android generó demoras menores en la fase de QA, lo que obligó a priorizar la automatización unitaria (JVM) sobre la integración de red en algunos módulos menos críticos.

3.5 Cierre formal del Sprint 1 — Evidencia de Aprendizaje 5

Checklist de cierre exigido para la evaluación del Sprint 1 de este repositorio (revisión a partir del 22/09/2026):

- Repositorio: rama main estable al cierre del Sprint 1; esquema de branching main / develop / feature por integrante o por HU.
- GitHub Projects: Milestone "Sprint 01" creado; Issues creados y asignados (ver Backlog, columna "Issue #"); tablero Kanban configurado en el repositorio consorcioMobile con columnas Backlog / Ready / In progress / Testing-QA / Done.
- Wiki: registro de ceremonias del Sprint 1, video de navegación de la app + capturas de Activities (Anexo II), página de Testing, sección de Ciberseguridad, avances individuales y de equipo, y página con la documentación de trabajo (actas, anotaciones).
- Este documento IEEE830 actualizado al cierre del Sprint 1.
- Ceremonia de Planning Poker registrada como Issue en GitHub, con voto individual de cada integrante antes del consenso y tabla de consenso final.

Anexo I

Diagrama Entidad-Relación (DER)

[Carpeta Documentación ConsorcioApp](#)

Mockups / Prototipos (Android)

[REVISAR / COMPLETAR: crear mockups/wireframes propios para las pantallas Android: Login, Dashboard Admin, Dashboard Residente, Alta de Reclamo, Feed de Comunicados, Alta de Residente, Perfil]

Referencias y enlaces

Recurso	Enlace
Repositorio y documentación del proyecto web original	https://github.com/Fenix-Software-2026/ConsorcioApp
Documentación de API (Swagger UI)	https://gaston.alwaysdata.net/api/docs
Documentación de API (ReDoc)	http://127.0.0.1:8000/api/redoc/

[REVISAR / COMPLETAR: agregar el repositorio de GitHub y el tablero de trabajo (Trello/GitHub Projects) específicos de este nuevo repositorio Android, una vez creados]