PLAN INTEGRAL DE DESARROLLO

APP DE SERVICIO TÉCNICO IT

Versión: 1.1 (Arquitectura Railway + Firebase)

---

1. RESUMEN DE INVERSIÓN Y RECURSOS

Recurso Humano: 1 Desarrollador Senior Full-stack (modalidad exclusiva).

Duración Total: 5 Meses (20 semanas / ~900–950 horas hombre).

Presupuesto de Desarrollo: USD 14.000 (alcance MVP + validación) con roadmap evolutivo posterior.

Costos Operativos (Año 1 – estimados): - Backend (Railway): USD 300–500 - Firebase (Auth + FCM + Storage): USD 200–300 - APIs externas (Maps, otros): USD 250

---

2. STACK TECNOLÓGICO DEFINITIVO

Frontend

- **Framework:** Flutter (desarrollo híbrido Android / iOS desde una única base de código)
- **Estrategia:** Híbrido con foco inicial en Android, sin bloqueo para iOS
- **Gestión de estado:** Bloc o Riverpod
- **Mapas:** Google Maps (Flutter plugin)

Backend

- **Plataforma:** Railway
- **Lenguaje:** Node.js (NestJS) o Python (FastAPI)
- **Arquitectura:** API REST
- **Base de datos:** PostgreSQL (Railway)

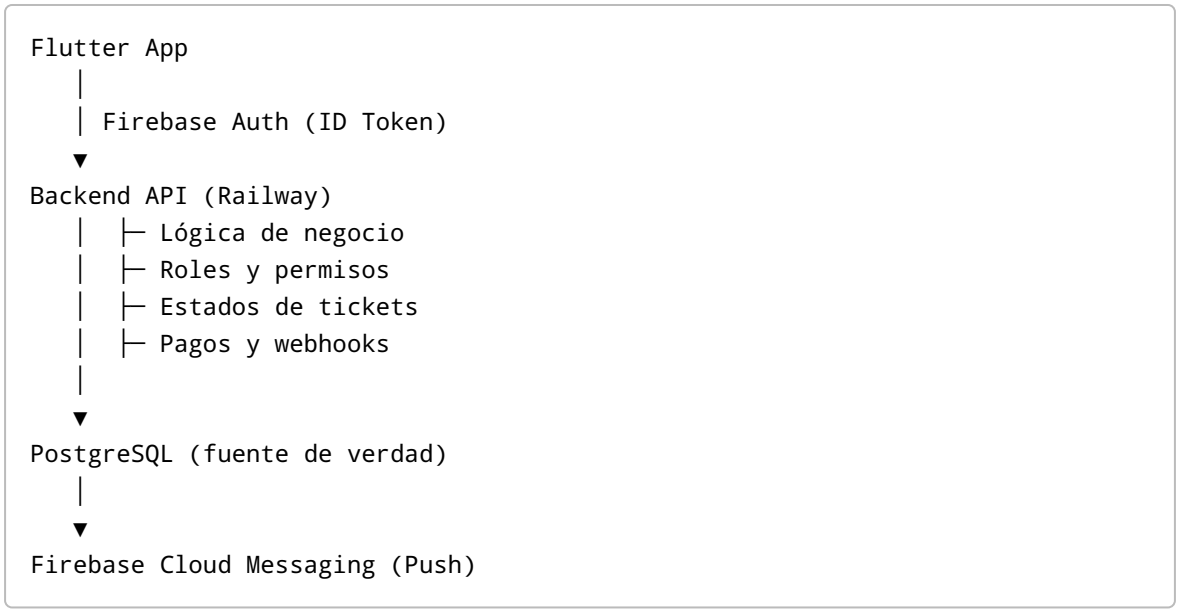
Servicios Externos

- **Autenticación:** Firebase Authentication
- **Notificaciones Push:** Firebase Cloud Messaging (FCM)
- **Storage de evidencias:** Firebase Storage (reemplazable por S3 compatible)
- **Pagos (Fase 3):** Mercado Pago – Checkout Pro + Webhooks

> **Principio arquitectónico:** Firebase se utiliza exclusivamente para identidad y mensajería. Toda la lógica de negocio, reglas, estados y auditoría residen en el backend desplegado en Railway.

---

3. ARQUITECTURA GENERAL DEL SISTEMA



---

4. GESTIÓN DE IDENTIDAD Y ROLES

Autenticación

- Login y sesión gestionados por Firebase Auth.
- El frontend envía el **Firebase ID Token** al backend en cada request.
- El backend valida el token usando Firebase Admin SDK.

Roles (definidos en base de datos)

- Cliente
- Técnico
- Administrador

> Los roles **no se gestionan en Firebase**, sino en la base de datos del backend para garantizar portabilidad y control.

---

5. MODELO DE NOTIFICACIONES

- El backend es el único responsable de decidir cuándo y a quién notificar.
- Firebase FCM se utiliza únicamente como canal de entrega.

Ejemplos de triggers: - Ticket creado → Push a técnicos de la zona - Ticket aceptado → Push al cliente - Ticket finalizado → Push + correo al administrador

---

6. CONTRATO DE API (RESUMEN)

Convenciones

- Base URL: /api/v1
- Autenticación: Firebase ID Token (Bearer)
- Backend valida roles, estados y transiciones

Endpoints principales

Contexto - GET /me

Cliente - POST /tickets - GET /tickets?mine=true - GET /tickets/{id} - POST /tickets/{id}/cancel - POST /tickets/{id}/sign (Fase 2)

Técnico - GET /tickets?status=abierto - POST /tickets/{id}/accept - POST /tickets/{id}/arrive - POST /tickets/{id}/start - POST /tickets/{id}/finish - POST /tickets/{id}/media (Fase 2)

Administrador - GET /tickets - POST /tickets/{id}/assign - POST /tickets/{id}/force-cancel - GET /tickets/{id}/events - GET /reports/tickets

---

7. CRONOGRAMA POR FASES (AJUSTADO)

FASE 1 – MVP OPERATIVO

Duración: 8 semanas | **Horas:** ~320

Enfoque: Digitalización completa del flujo de tickets con backend propio en Railway.

(Contenido funcional igual al documento original, adaptado al nuevo stack.)

FASE 2 – VALIDACIÓN Y CALIDAD

Duración: 4 semanas | **Horas:** ~200

Incluye: - Evidencia fotográfica (Firebase Storage) - Firma digital - Geolocalización básica - Reportes PDF - Auditoría de eventos (tabla `ticket_events`)

FASE 3 – ESCALABILIDAD Y MONETIZACIÓN

Duración: 8 semanas | **Horas:** ~350–400

Incluye: - Navegación GPS + ETA - Integración Mercado Pago (Checkout Pro) - Webhooks y estados de pago - Notificaciones avanzadas - Publicación en stores

---

7. MODELO DE DATOS (ALTO NIVEL)

Fase 1 (MVP)

- users
- roles
- user_roles
- tickets
- ticket_events (auditoría)
- devices (tokens FCM)

Fase 2 (Validación y Calidad)

- ticket_media (fotos antes/después)
- ticket_signatures (firma digital)
- ticket_locations (geolocalización)
- service_reports (PDF / comprobantes)

---

8. CONSIDERACIONES CLAVE DE ESCALABILIDAD

- API versionada (/v1)
- Lógica desacoplada del frontend
- Posibilidad de migrar backend fuera de Railway
- Firebase reemplazable en Auth y Storage

---

9. CONCLUSIÓN

El proyecto adopta una estrategia de desarrollo **híbrida con Flutter**, permitiendo:

- Reducir tiempos y costos de desarrollo
- Mantener una única base de código para Android e iOS
- Asegurar performance adecuada para aplicaciones operativas de campo
- Facilitar la escalabilidad futura del producto

La combinación **Flutter + Backend en Railway + Firebase (Auth y FCM)** proporciona un equilibrio óptimo entre velocidad de desarrollo, calidad técnica y control de la lógica de negocio.

El sistema queda preparado tanto para su uso interno como para una futura evolución a modelo SaaS o licenciamiento a terceros.