

# TECHNICAL DESIGN DOCUMENT (TDD)

## Plataforma de Gestión de Servicios Técnicos IT

Versión: 1.0

---

### 1. OBJETIVO DEL DOCUMENTO

Este documento define el diseño técnico completo del sistema de gestión de servicios técnicos IT, con el objetivo de servir como **referencia única y definitiva** para el desarrollo, mantenimiento y escalabilidad de la plataforma.

El TDD establece: - Arquitectura general - Decisiones tecnológicas - Modelo de datos - Contrato de API - Flujos críticos - Consideraciones de seguridad y escalabilidad

---

### 2. ALCANCE DEL SISTEMA

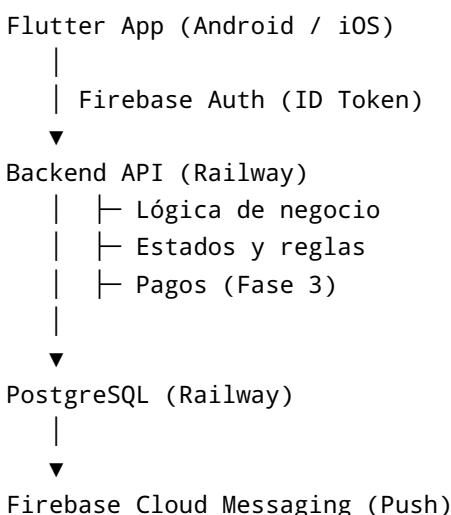
El sistema permite: - Gestión de tickets de servicio técnico - Operación en campo por técnicos - Supervisión y control por administradores - Evidencia, auditoría y reportes

Fuera de alcance inicial: - Gestión de inventario - Facturación fiscal - Integraciones ERP

---

### 3. ARQUITECTURA GENERAL

#### 3.1 Visión de alto nivel



### **3.2 Principios arquitectónicos**

- Backend como única fuente de verdad
  - Cliente sin lógica de negocio
  - Firebase usado solo como infraestructura
  - Estados controlados mediante máquina de estados
- 

## **4. STACK TECNOLÓGICO**

### **Frontend**

- Flutter (base de código única Android / iOS)
- Bloc o Riverpod para gestión de estado
- Google Maps Flutter Plugin

### **Backend**

- Node.js + NestJS (o Python + FastAPI)
- API REST
- PostgreSQL

### **Servicios externos**

- Firebase Authentication
  - Firebase Cloud Messaging
  - Firebase Storage
  - Mercado Pago (Checkout Pro – Fase 3)
- 

## **5. GESTIÓN DE IDENTIDAD Y SEGURIDAD**

### **5.1 Autenticación**

- Autenticación gestionada por Firebase Auth
- Uso de Firebase ID Token en cada request
- Validación del token en backend mediante Firebase Admin SDK

### **5.2 Autorización**

- Roles gestionados en base de datos
- Guards por rol y acción
- Validación estricta en backend

Roles: - Cliente - Técnico - Administrador

---

## 6. MODELO DE DATOS

### Entidades principales

- users
- roles
- user\_roles
- tickets
- ticket\_events
- devices

### Entidades Fase 2

- ticket\_media
- ticket\_signatures
- ticket\_locations
- service\_reports

El modelo garantiza trazabilidad completa y auditoría.

---

## 7. MÁQUINA DE ESTADOS DEL TICKET

### Estados válidos

- abierto
- asignado
- en\_camino
- en\_proceso
- finalizado
- cancelado

### Transiciones permitidas

```
abierto → asignado → en_camino → en_proceso → finalizado  
abierto → cancelado  
(asignado | en_camino | en_proceso) → cancelado (solo admin)
```

Cada transición: - Es validada por backend - Genera evento de auditoría - Dispara notificaciones

---

## 8. CONTRATO DE API (RESUMEN)

### Convenciones

- Base URL: /api/v1
- Auth: Bearer Firebase ID Token
- JSON

## **Endpoints clave**

- GET /me
- POST /tickets
- GET /tickets
- POST /tickets/{id}/accept
- POST /tickets/{id}/arrive
- POST /tickets/{id}/start
- POST /tickets/{id}/finish
- POST /tickets/{id}/cancel

(Admin y Fase 2 ver documento funcional)

---

## **9. NOTIFICACIONES**

- Decididas exclusivamente por backend
  - Enviadas vía Firebase Cloud Messaging
  - Asociadas a eventos de ticket
- 

## **10. DESPLIEGUE E INFRAESTRUCTURA**

### **Backend**

- Despliegue continuo en Railway
- Variables de entorno seguras

### **Base de datos**

- PostgreSQL administrado en Railway

### **Mobile**

- Builds Flutter para Android e iOS
  - Publicación en stores en Fase 3
- 

## **11. ESCALABILIDAD Y EVOLUCIÓN**

El diseño permite:

- Migrar backend fuera de Railway
- Reemplazar Firebase Storage
- Incorporar nuevos roles
- Evolucionar a modelo SaaS

---

## **12. RIESGOS Y MITIGACIONES**

Riesgo	Mitigación
Vendor lock-in	Servicios desacoplados

Riesgo	Mitigación
Errores de estado	Máquina de estados
Bugs de permisos	Guards backend
Crecimiento	API versionada

---

## 13. CONCLUSIÓN

Este TDD establece una base técnica sólida, escalable y mantenible, adecuada para un producto profesional de operación en campo, minimizando riesgos y facilitando la evolución futura del sistema.