Documento de Arquitectura de Software (DAS)

Versión: 1.0

Fecha: 26 de mayo de 2025

Responsables: Maria Jose Alvarez, Rene Figueroa, Carlos Navarro

# 1. Introducción

Este Documento de Arquitectura de Software (DAS) detalla la estructura y los componentes del sistema de Gestión de Rutas para Conductores. Incluye las decisiones arquitectónicas, patrones de diseño, diagramas y consideraciones para garantizar confiabilidad, escalabilidad y seguridad en producción.

# 2. Propósito

El propósito de este documento es proporcionar una guía clara y detallada de la arquitectura del sistema, sirviendo de referencia para el equipo de desarrollo, pruebas y operaciones.

# 3. Alcance

Cubre la vista lógica, de componentes, de despliegue y de datos, incluyendo diagramas de arquitectura, patrones utilizados y especificaciones de infraestructura.

# 4. Contexto del Sistema

El sistema se compone de dos aplicaciones front-end (web en Angular y móvil en Ionic Angular) y un backend serverless en Firebase (Authentication, Firestore, Cloud Functions y Storage).

# 5. Visión General de la Arquitectura

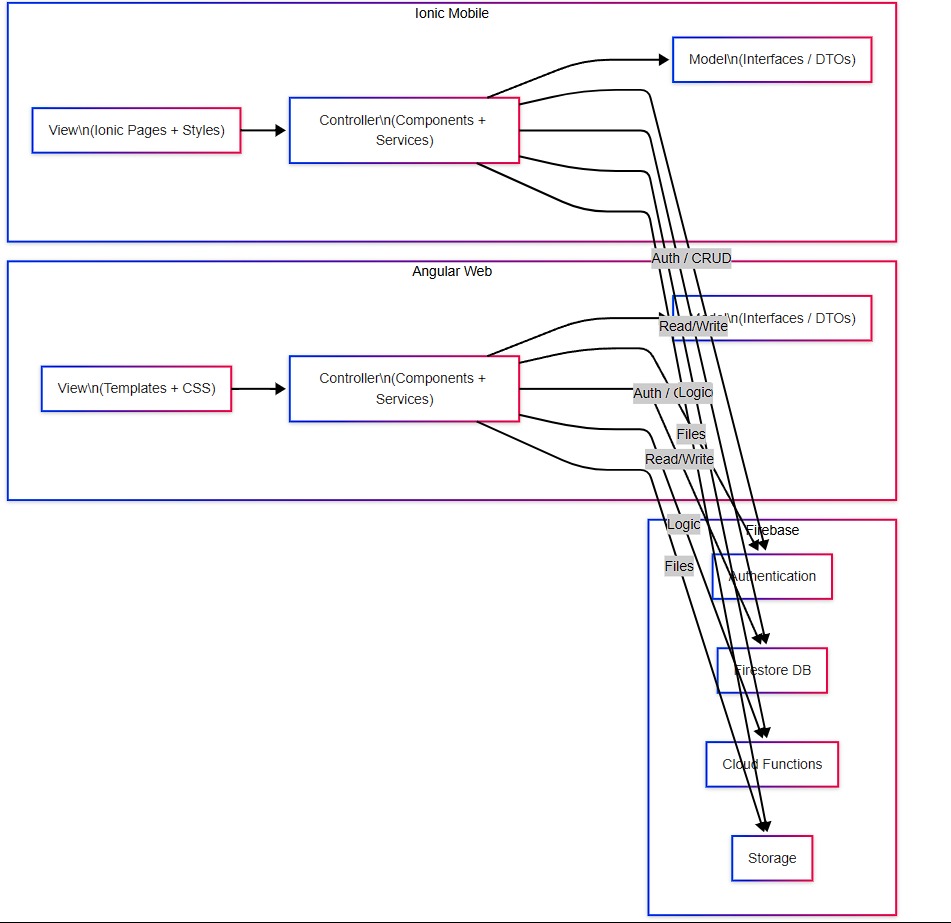
La arquitectura adopta un estilo cliente-servidor serverless, delegando la lógica de negocio a Cloud Functions y el almacenamiento de datos a Firestore. Se emplean patrones MVC en el front-end y servicios centralizados para acceso a datos.

Componentes principales:

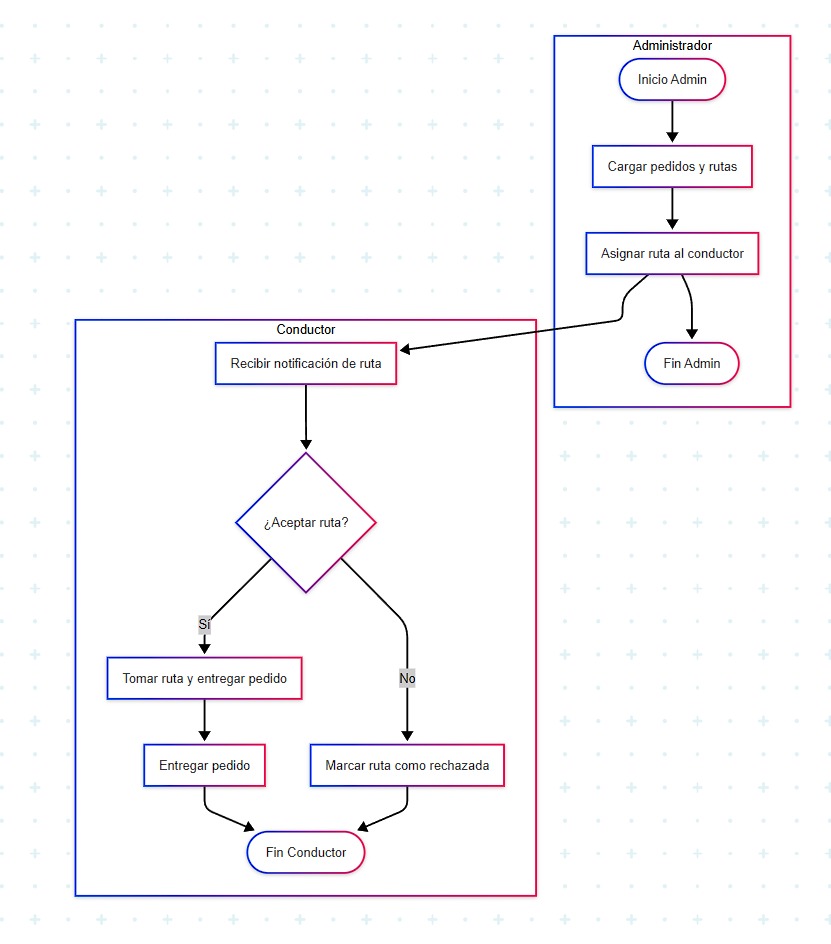
* Panel web (Angular) con módulos de autenticación, gestión de usuarios y asignación de rutas.
* App móvil (Ionic Angular) para conductores, con módulos de autenticación, visualización y gestión de pedidos.
* Backend en Firebase: Firestore como base de datos NoSQL en tiempo real, Authentication para seguridad, Cloud Functions para procesos back-end, y Storage para archivos.

# 6. Vistas de Arquitectura

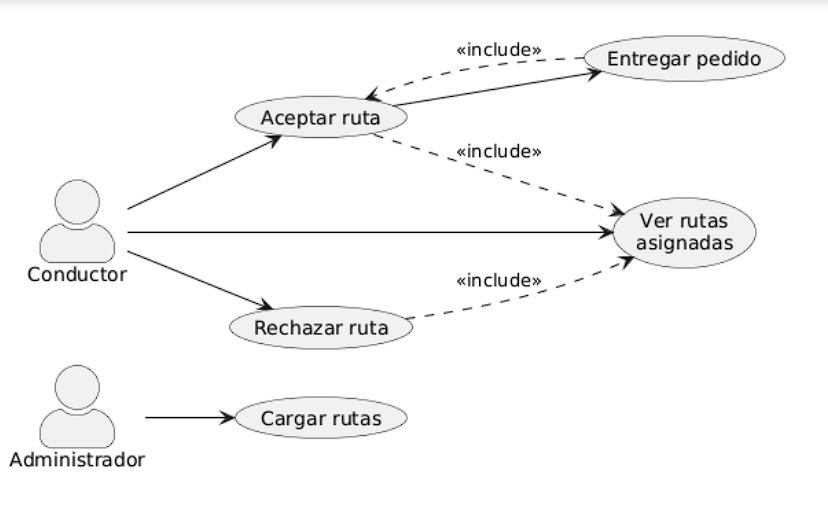
## 6.1 Diagrama MVC



## 6.2 Diagrama de Flujo de Datos



## 6.3 Diagrama de Casos de Uso



# 7. Diseño de Componentes

Cada aplicación front-end se organiza en capas: componentes UI, servicios de acceso a datos, modelos y utilidades. Las Cloud Functions implementan casos de uso críticos: asignación de rutas, notificaciones y procesamiento de estados.

# 8. Diseño de Datos

Firestore almacena colecciones principales: Usuarios, Conductores, Rutas, Pedidos. Cada documento de 'Pedidos' incluye campos: estado, timestamps, referencia a conductor y datos de cliente.

# 9. Seguridad

* Autenticación via Firebase Authentication con email/contraseña.
* Reglas de seguridad en Firestore basadas en claims de usuario (roles).
* TLS en todas las comunicaciones.
* Validación de accesos en Cloud Functions.

# 10. Despliegue

La aplicación web se despliega en Firebase Hosting. La app móvil se publica en Google Play o se distribuye vía Landing Page con APK. Cloud Functions y Firestore se gestionan desde Firebase Console.

# 11. Rendimiento y Escalabilidad

* Firestore indexado para consultas de rutas y pedidos.
* Funciones lite en Cloud Functions con cold start minimizado.
* Cache en el cliente para datos inmutables.
* Dimensionamiento automático de Firestore y escalado de funciones.

# 12. Mantenimiento y Monitoreo

* Logging centralizado en Cloud Logging.
* Alertas en Cloud Monitoring para errores críticos.
* Documentación actualizada en repo con diagramas y especificaciones.

# 13. Referencias

* Firebase Documentation
* Angular Style Guide
* Ionic Framework Docs
* Design Patterns: Elements of Reusable Object-Oriented Software