MANUAL TÉCNICO

Aplicación UberEats Clone

Sistema de Gestión de Pedidos y Entregas

Versión: 1.0

Fecha: 17 de junio de 2025

Documentación Técnica Completa del Proyecto

Tecnologías: Flutter • Firebase • Google Maps • Riverpod

Índice

1 Resumen Ejecutivo

La aplicación UberEats Clone es un sistema completo de gestión de pedidos y entregas que replica las funcionalidades principales de plataformas de delivery como UberEats. El proyecto implementa un ecosistema completo con roles diferenciados para clientes, tiendas y repartidores, incluyendo características avanzadas como análisis en tiempo real, gestión de inventario, sistema de pagos y comunicación integrada.

2 Objetivo del Proyecto

2.1 Objetivo General

Desarrollar una aplicación móvil multiplataforma que facilite la gestión integral de pedidos de comida, desde la selección de productos hasta la entrega final, proporcionando una experiencia de usuario fluida y herramientas de gestión empresarial robustas.

2.2 Objetivos Específicos

- Implementar un sistema de autenticación seguro con Firebase Auth
- Crear interfaces diferenciadas para clientes, tiendas y repartidores
- Desarrollar un sistema de pedidos en tiempo real con seguimiento GPS
- Integrar un sistema completo de gestión de pagos
- Implementar análisis de ventas y métricas para tiendas
- Crear un sistema de gestión de inventario automatizado
- Desarrollar notificaciones push y sistema de comunicación
- Garantizar escalabilidad y mantenibilidad del código

2.3 Parámetros de Evaluación

Criterio	**Especificación**
Funcionalidad	Sistema completo de pedidos y entregas
Usabilidad	Interfaz intuitiva y responsive
Rendimiento	Tiempo de respuesta ¡2 segundos
Seguridad	Autenticación robusta y datos encriptados
Escalabilidad	Arquitectura modular con Riverpod
Multiplataforma	Compatible Android e iOS
Tiempo Real	Actualizaciones instantáneas vía Firebase

Cuadro 1: Parámetros de evaluación del proyecto

3 Tecnologías y Lenguajes Utilizados

3.1 Lenguajes de Programación

- **Dart**: Lenguaje principal para desarrollo Flutter
- **JavaScript**: Para funciones de Firebase Cloud Functions
- **Kotlin/Java**: Para configuraciones nativas Android
- **Swift/Objective-C**: Para configuraciones nativas iOS

3.2 Framework Principal

- **Flutter 3.x**: Framework multiplataforma de Google
- **Dart SDK**: Entorno de desarrollo
- **Material Design**: Sistema de diseño de Google

3.3 Backend y Base de Datos

- **Firebase Firestore**: Base de datos NoSQL en tiempo real
- **Firebase Auth**: Sistema de autenticación
- **Firebase Storage**: Almacenamiento de archivos
- **Firebase Messaging**: Notificaciones push
- **Firebase Functions**: Lógica del servidor

3.4 Gestión de Estado y Arquitectura

- **Riverpod**: Gestión de estado reactiva
- **Provider Pattern**: Arquitectura de inyección de dependencias
- **Repository Pattern**: Abstracción de datos

3.5 APIs y Servicios Externos

- **Google Maps API**: Mapas y geolocalización
- **Places API**: Búsqueda de lugares
- **Geocoding API**: Conversión de coordenadas
- **Firebase Cloud Messaging**: Notificaciones

3.6 Librerías y Dependencias Principales

Librería	Versión	Propósito
flutter_riverpod	^2.4.9	Gestión de estado
firebase_core	^2.24.2	Configuración Firebase
cloud_firestore	^4.13.6	Base de datos
firebase_auth	^4.15.3	Autenticación
google_maps_flutter	^2.5.0	Mapas integrados
geolocator	^10.1.0	Geolocalización
image_picker	^1.0.4	Selección de imágenes
fl_chart	^0.65.0	Gráficos y análisis
firebase_messaging	^14.7.10	Notificaciones push
flutter_local_notifications	^16.3.0	Notificaciones locales
url_launcher	^6.2.2	Navegación externa
permission_handler	^11.1.0	Permisos del sistema

Cuadro 2: Principales dependencias del proyecto

4 Arquitectura del Sistema

4.1 Patrón de Arquitectura

El proyecto implementa una **arquitectura en capas** con separación clara de responsabilidades:

- **Capa de Presentación**: Screens y Widgets
- **Capa de Lógica de Negocio**: Providers y Notifiers
- **Capa de Datos**: Services y Repositories
- **Capa de Modelos**: Data Models y DTOs

4.2 Estructura de Directorios

```
lib/
|-- models/                    # Modelos de datos
|   |-- payment_model.dart
|   |-- analytics_model.dart
|   |-- inventory_model.dart
|   |-- notification_model.dart
|   |-- chat_model.dart
|   |-- order_model.dart
|-- services/                  # Servicios y APIs
|   |-- payment_service.dart
|   |-- notification_service.dart
|   |-- location_service.dart
|   |-- firebase_service.dart
```

```
|-- providers/                    # Gestion de estado
|   |-- payment_provider.dart
|   |-- analytics_provider.dart
|   |-- inventory_provider.dart
|   |-- auth_provider.dart
|-- screens/                      # Pantallas de la aplicacion
|   |-- customer/
|   |-- store/
|   |-- deliverer/
|-- widgets/                      # Componentes reutilizables
|-- utils/                        # Utilidades y helpers
```

Listing 1: Estructura del proyecto

5 Funcionalidades Implementadas

5.1 Sistema de Autenticación

- Registro y login con email/contraseña
- Autenticación con Google Sign-In
- Gestión de sesiones persistentes
- Roles diferenciados (cliente, tienda, repartidor)
- Recuperación de contraseñas

5.2 Gestión de Pedidos

- Catálogo de productos con filtros
- Carrito de compras interactivo
- Sistema de reordenar pedidos anteriores
- Seguimiento en tiempo real de pedidos
- Estados de pedido automatizados
- Historial completo de pedidos

5.3 Sistema de Pagos

- Múltiples métodos de pago (tarjeta, efectivo, digital)
- Validación de tarjetas con algoritmo de Luhn
- Cálculo automático de propinas (0 %, 10 %, 15 %, 20 %)
- Procesamiento seguro de transacciones

- Historial de pagos y reembolsos
- Integración preparada para gateways reales

5.4 Análisis y Métricas (Dashboard)

- Métricas de ventas en tiempo real
- Gráficos de ingresos con fl_chart
- Análisis de productos más vendidos
- Identificación de horas pico
- Tasas de finalización de pedidos
- Reportes por períodos personalizables

5.5 Gestión de Inventario

- Seguimiento de stock en tiempo real
- Alertas de inventario bajo
- Historial de movimientos de inventario
- Gestión de proveedores
- Costos y precios automatizados
- Reportes de rotación de productos

5.6 Sistema de Notificaciones

- Notificaciones push con Firebase Messaging
- Notificaciones locales por categorías
- Configuración de preferencias por usuario
- Notificaciones de pedidos, chat, inventario
- Horarios de silencio configurables
- Badges y contadores de notificaciones

5.7 Comunicación y Chat

- Sistema de chat tienda-cliente
- Integración con WhatsApp
- Mensajes pre-configurados

- Estados de mensaje (enviado, entregado, leído)
- Soporte para texto, imágenes y ubicación
- Notificaciones de nuevos mensajes

5.8 Seguimiento y Entrega

- Seguimiento GPS en tiempo real
- Mapas interactivos con Google Maps
- Rutas optimizadas para repartidores
- Estimación de tiempos de entrega
- Notificaciones de proximidad
- Confirmación de entrega con ubicación

6 Flujo de Lógica del Negocio

6.1 Flujo Principal del Cliente

1. **Autenticación**: Registro/Login con email o Google
2. **Exploración**: Búsqueda de tiendas y productos
3. **Selección**: Agregado de productos al carrito
4. **Configuración**: Dirección de entrega y método de pago
5. **Pago**: Procesamiento seguro de la transacción
6. **Seguimiento**: Monitoreo en tiempo real del pedido
7. **Comunicación**: Chat con la tienda si es necesario
8. **Recepción**: Confirmación de entrega
9. **Evaluación**: Calificación del servicio

6.2 Flujo de Gestión de Tienda

1. **Configuración**: Setup inicial de la tienda
2. **Inventario**: Gestión de productos y stock
3. **Pedidos**: Recepción y procesamiento de órdenes
4. **Preparación**: Actualización de estados de pedido
5. **Comunicación**: Chat con clientes cuando necesario

6. **Analytics**: Revisión de métricas y ventas
7. **Notificaciones**: Alertas de inventario y pedidos

6.3 Flujo del Repartidor

1. **Disponibilidad**: Marcado como disponible
2. **Asignación**: Recepción de pedidos asignados
3. **Recolección**: Navegación a la tienda
4. **Confirmación**: Pickup del pedido
5. **Entrega**: Navegación al cliente con GPS
6. **Comunicación**: Contacto con cliente vía WhatsApp
7. **Finalización**: Confirmación de entrega

7 Base de Datos y Modelos

7.1 Colecciones de Firestore

Colección	Propósito
users	Información de usuarios y perfiles
stores	Datos de tiendas y configuración
products	Catálogo de productos
orders	Pedidos y su estado
payments	Transacciones de pago
inventory	Stock y movimientos
notifications	Historial de notificaciones
chat_rooms	Conversaciones tienda-cliente
deliveries	Información de entregas
analytics	Métricas y estadísticas

Cuadro 3: Estructura de base de datos en Firestore

7.2 Modelos de Datos Principales

- **User Model**: Autenticación y perfil de usuario
- **Store Model**: Información y configuración de tiendas
- **Product Model**: Catálogo de productos con precios
- **Order Model**: Pedidos con estados y seguimiento
- **Payment Model**: Transacciones y métodos de pago

- **Inventory Model**: Stock y alertas de inventario
- **Analytics Model**: Métricas y KPIs
- **Notification Model**: Configuración de notificaciones

8 Configuración y Despliegue

8.1 Requisitos del Sistema

- **Flutter SDK**: 3.16.0 o superior
- **Dart SDK**: 3.2.0 o superior
- **Android Studio**: Para desarrollo Android
- **Xcode**: Para desarrollo iOS (macOS)
- **Firebase Project**: Configurado con todos los servicios
- **Google Maps API**: Claves de API configuradas

8.2 Comandos de Construccion

```
# Instalar dependencias
flutter pub get

# Analizar codigo
flutter analyze

# Ejecutar tests
flutter test

# Limpiar proyecto
flutter clean

# Construir APK debug
flutter build apk --debug

# Construir APK release
flutter build apk --release

# Ejecutar aplicacion
flutter run
```

Listing 2: Comandos principales

8.3 Configuracion de Firebase

1. Crear proyecto en Firebase Console

2. Habilitar Authentication (Email/Password y Google)
3. Configurar Firestore con reglas de seguridad
4. Activar Firebase Storage
5. Configurar Firebase Messaging
6. Descargar archivos de configuracion:
   - `google-services.json` para Android
   - `GoogleService-Info.plist` para iOS

9 Analisis de Costos

9.1 Costos de Desarrollo

Recurso	Tiempo	Descripción
Análisis y Diseño	40 horas	Planificación y wireframes
Desarrollo Frontend	120 horas	UI/UX y funcionalidades
Integración Backend	60 horas	Firebase y APIs
Testing y QA	30 horas	Pruebas y correcciones
Documentación	20 horas	Manual técnico y guías
Total	**270 horas**	**Tiempo total de desarrollo**

Cuadro 4: Estimación de tiempo de desarrollo

9.2 Costos de Infraestructura

Servicio	Costo Mensual	Límites
Firebase Firestore	$0 - $25	Según uso y lecturas/escrituras
Firebase Auth	Gratis	Hasta 50,000 usuarios
Firebase Storage	$0 - $10	Según almacenamiento usado
Google Maps API	$0 - $200	Según consultas de mapas
Firebase Messaging	Gratis	Notificaciones ilimitadas
Total Estimado	**$0 - $235**	**Para aplicación pequeña-mediana**

Cuadro 5: Costos estimados de servicios en la nube

9.3 Costos de Distribución

- **Google Play Store**: $25 (pago único para cuenta de desarrollador)
- **Apple App Store**: $99/año (membresía de desarrollador)
- **Certificados de Firma**: Incluidos en las membresías

10 Referencias Tecnicas

10.1 Documentacion Oficial

- Flutter Documentation: https://docs.flutter.dev/
- Firebase Documentation: https://firebase.google.com/docs
- Riverpod Documentation: https://riverpod.dev/
- Google Maps Flutter: https://pub.dev/packages/google_maps_flutter
- Material Design Guidelines: https://material.io/design

10.2 Librerias y Paquetes

- Pub.dev - Flutter Packages: https://pub.dev/
- Firebase Flutter Plugins: https://firebase.flutter.dev/
- FL Chart Documentation: https://github.com/imaNNeo/fl_chart
- Geolocator Plugin: https://pub.dev/packages/geolocator

10.3 APIs y Servicios

- Google Maps Platform: https://developers.google.com/maps
- Firebase Console: https://console.firebase.google.com/
- Google Cloud Console: https://console.cloud.google.com/

11 Consideraciones de Seguridad

11.1 Autenticacion y Autorizacion

- Tokens JWT manejados automaticamente por Firebase Auth
- Reglas de seguridad de Firestore por roles de usuario
- Validacion de permisos en el lado del servidor
- Sesiones seguras con renovacion automatica

11.2 Proteccion de Datos

- Encriptacion en transito (HTTPS/TLS)
- Encriptacion en reposo (Firebase)
- Validacion de datos de entrada

- Sanitizacion de datos sensibles
- Cumplimiento con mejores practicas de privacidad

12 Mantenimiento y Escalabilidad

12.1 Monitoreo

- Firebase Analytics para metricas de uso
- Crashlytics para reporte de errores
- Performance Monitoring para optimizacion
- Alertas automaticas de Firebase

12.2 Escalabilidad

- Arquitectura modular con Riverpod
- Firestore automaticamente escalable
- Caching estrategico en la aplicacion
- Optimizacion de consultas de base de datos
- Lazy loading de contenido

13 Conclusiones

El proyecto UberEats Clone representa una implementacion completa y robusta de un sistema de delivery de comida. La aplicacion cumple con todos los requisitos funcionales establecidos y proporciona una base solida para un producto comercial.

13.1 Fortalezas del Proyecto

- **Arquitectura Solida**: Separacion clara de responsabilidades
- **Tecnologias Modernas**: Stack tecnologico actualizado y mantenible
- **Escalabilidad**: Diseno preparado para crecimiento
- **Funcionalidad Completa**: Todas las caracteristicas de una app de delivery
- **Experiencia de Usuario**: Interfaces intuitivas y responsivas
- **Tiempo Real**: Actualizaciones instantaneas y seguimiento live

13.2 Oportunidades de Mejora

- Integracion con gateways de pago reales
- Implementacion de machine learning para recomendaciones
- Optimizacion adicional de rendimiento
- Expansion de funcionalidades de analisis
- Implementacion de tests automatizados mas extensivos

Este manual técnico documenta completamente la implementación del sistema UberEats Clone desarrollado con Flutter y Firebase.
