UNIVERSIDAD FRANCISCO GAVIDIA
Tecnología, Innovación y Calidad

MATERIA:

ANALIZANDO LAS NECESIDADES DE HARDWARE Y SOFTWARE

TÍTULO:

PARCIAL 3 – BALANCE DUAL APP.

MAESTRO:
ING. CARLOS BORIS MARTÍNEZ CALZADIA

ALUMNO:	**# CARNET**
ARBAIZA ORELLANA, FRANCISCO ALEXANDER	AO100422

Contenido

Nombre del proyecto y descripción breve

Balance Dual es una aplicación móvil diseñada para optimizar la gestión de las finanzas personales mediante una metodología clara y efectiva. Su característica central es la implementación de un sistema de doble cartera que maneja de manera rigurosa los flujos de capital, fomentando una disciplina financiera superior. La aplicación ofrece una experiencia de usuario fluida con una interfaz gráfica que presenta los datos financieros de manera legible.

Doble Cartera y Asistente IA:

Gastos: Destinada a la administración del presupuesto diario, semanal o mensual y el seguimiento detallado de las transacciones operativas y personales.

Ahorros: Exclusivamente reservada para objetivos financieros a corto, medio y largo plazo, garantizando que el capital de reserva no se mezcle con los gastos corrientes.

Asistente de Inteligencia Artificial: incorpora un asistente de IA inteligente que va más allá del mero registro. Esta herramienta analiza los patrones de gasto del usuario, identifica oportunidades de ahorro, genera recomendaciones financieras personalizadas y proyecta escenarios futuros, facilitando la toma de decisiones financieras informadas y estratégicas.

Plataforma No-Code elegida

- ✓ **Framework:** React Native con Expo (~54.0.20)
- ✓ **Lenguaje:** TypeScript
- ✓ **Base de Datos:** Firebase Firestore
- ✓ **Autenticación:** Firebase Authentication
- ✓ **IA:** Google Gemini API
- ✓ **Navegación:** React Navigation (Stack y Bottom Tabs)
- ✓ **Almacenamiento Local:** AsyncStorage
- ✓ **Estado Global:** React Context API

Descripción de la funcionalidad implementada

1. Autenticación (src/features/auth/)

- *Registro de Usuario*: Crea una cuenta con email y contraseña, inicializa las carteras en cero
- *Inicio de Sesión*: Autenticación con email y contraseña usando Firebase Auth
- *OTP por Email*: Sistema de verificación de código de 6 dígitos enviado por email
- *Persistencia de Sesión*: La sesión se mantiene usando AsyncStorage

Servicios:

- authService.login(): Iniciar sesión
- authService.register(): Registrar nuevo usuario
- authService.sendEmailOTP(): Enviar código OTP por email
- authService.verifyEmailOTP(): Verificar código OTP (máximo 3 intentos, expira en 5 minutos)
- authService.logout(): Cerrar sesión

2. Gestión de Carteras (src/features/wallets/)

- *Dual Wallet System*: Dos carteras independientes (Gastos y Ahorros)
- *Balance en Tiempo Real*: Los saldos se actualizan automáticamente con cada transacción
- *Visualización*: Tarjetas informativas mostrando el saldo de cada cartera y el total

Servicios:

- walletService.getUser(): Obtener datos del usuario y saldos de carteras
- walletService.updateWalletBalances(): Actualizar saldos de ambas carteras
- walletService.initializeUserWallets(): Inicializar carteras para nuevo usuario

3. Transacciones (src/features/transactions/)

- *Tipos de Transacción*:
  - *Ingreso (Income)*: Añade dinero a la cartera de gastos
  - *Gasto (Expense)*: Resta dinero de la cartera de gastos (valida saldo suficiente)
  - *Transferencia (Transfer)*: Mueve dinero entre las dos carteras
- *Historial Completo*: Lista todas las transacciones ordenadas por fecha (más reciente primero)
- *Validación de Saldos*: Previene transacciones que excedan el saldo disponible

Servicios:

- transactionService.createTransaction(): Crear nueva transacción y actualizar saldos
- transactionService.getUserTransactions(): Obtener todas las transacciones del usuario
- transactionService.getTransactionsByDateRange(): Filtrar transacciones por rango de fechas
- transactionService.getTransactionsByType(): Filtrar transacciones por tipo

4. Asistente de IA (src/features/ai-assistant/)

- *Consultas Inteligentes*: Haz preguntas sobre tus finanzas y recibe respuestas contextualizadas
- *Contexto Financiero*: El asistente tiene acceso a tus saldos y transacciones recientes
- *Sugerencias Automáticas*: Extrae recomendaciones financieras de las respuestas
- *Análisis Rápido*: Genera insights sobre tu actividad financiera

Servicios:

- geminiService.processQuery(): Procesa consulta del usuario con contexto financiero
- geminiService.getQuickInsights(): Genera insights rápidos sobre tus finanzas
- queryParser.parseResponse(): Parsea y formatea las respuestas de la IA

5. Navegación

- *AppNavigator*: Gestiona el flujo principal (Auth → Main)
- *AuthNavigator*: Pantallas de autenticación (Login, Register, OTP)
- *MainNavigator*: Navegación principal con tabs (Home, Transacciones, Asistente)
- *TransactionStack*: Stack de navegación para transacciones (Historial → Nueva Transacción)

Diagrama de infraestructura:

```
BalanceDualExpo/
├── src/
│   ├── core/
│   │   ├── config/          # Configuración de Firebase y Gemini
│   │   ├── context/         # Context providers (Auth, Wallet)
│   │   └── navigation/      # Navegadores de la app
│   ├── features/
│   │   ├── ai-assistant/    # Asistente de IA
│   │   ├── auth/            # Autenticación (Login, Register, OTP)
│   │   ├── transactions/    # Gestión de transacciones
│   │   └── wallets/         # Gestión de carteras
│   └── shared/
│       ├── components/      # Componentes reutilizables
│       ├── constants/       # Constantes y tema
│       ├── hooks/           # Custom hooks
│       ├── types/           # TypeScript types
│       └── utils/           # Utilidades
├── assets/                  # Imágenes e íconos
├── App.tsx                  # Componente principal
├── app.config.ts            # Configuración de Expo
├── package.json             # Dependencias
└── .env                     # Variables de entorno
```

Explicación Uso de IA

Antes de comenzar, asegúrate de tener instalado:

- *Node.js* (versión 18 o superior)
- *npm* o *yarn*
- *Expo CLI* (npm install -g expo-cli)
- Una cuenta de *Firebase* con un proyecto configurado
- Una cuenta de *Google Cloud* con la API de Gemini habilitada
- Un webhook de Google Apps Script para envío de emails OTP

Google Gemini API **(NO COMPARTIR ESTAS CREDENCIALES)**

GEMINI_API_KEY=tu_gemini_api_key

https://aistudio.google.com/app/u/2/api-keys

Webhook para envío de OTP por email (Google Apps Script)

EMAIL_OTP_WEBHOOK_URL=https://script.google.com/macros/s/tu_script_id/exec

Cómo obtener las credenciales de Firebase: **(NO COMPARTIR ESTAS CREDENCIALES)**

1. Ve a [Firebase Console](https://console.firebase.google.com/)
2. Selecciona o crea un proyecto
3. Ve a *Configuración del proyecto* (ícono de engranaje)
4. En la sección *Tus aplicaciones*, selecciona la app web o crea una nueva
5. Copia los valores de configuración y pégalos en tu .env

Cómo obtener la API Key de Gemini: (NO COMPARTIR ESTAS CREDENCIALES)

1. Ve a [Google AI Studio](https://makersuite.google.com/app/apikey)
2. Inicia sesión con tu cuenta de Google
3. Crea una nueva API Key
4. Copia la clave y pégala en GEMINI_API_KEY

Configurar Webhook para OTP por Email:

1. Crea un script en [Google Apps Script](https://script.google.com/)
2. Configura una función que reciba peticiones POST con email y otpCode
3. Implementa el envío de email usando MailApp.sendEmail()
4. Despliega el script como una aplicación web
5. Copia la URL del webhook en EMAIL_OTP_WEBHOOK_URL

Ejemplo de Google Apps Script:

```
javascript
function doPost(e) {
  const data = JSON.parse(e.postData.contents);
  const { email, otpCode } = data;

  MailApp.sendEmail({
    to: email,
    subject: 'Código de verificación - Balance Dual',
    htmlBody: `
      <h2>Código de Verificación</h2>
      <p>Tu código de verificación es: <strong>${otpCode}</strong></p>
      <p>Este código expira en 5 minutos.</p>
```

```
  `
  });

  return ContentService.createTextOutput(JSON.stringify({success: true}))
    .setMimeType(ContentService.MimeType.JSON);
}
```

Configuración de Firebase
Estructura de Firestore

Colección: users

typescript

```
{
  email: string;
  createdAt: Timestamp;
  spendingWallet: number;
  savingsWallet: number;
}
```

Colección: transactions

typescript

```
{
  userId: string;
  type: 'income' | 'expense' | 'transfer';
  amount: number;
  description: string;
  fromWallet?: 'spending' | 'savings';
  toWallet?: 'spending' | 'savings';
  createdAt: Timestamp;
}
```

Colección: emailOtps

```
typescript
{
  otp: string;
  attempts: number;
  expiresAt: number; // timestamp
  createdAt: Timestamp;
  usedAt?: Timestamp;
}
```

Índice de Firestore

Para optimizar las consultas de transacciones, se requiere un índice compuesto:

Colección: transactions
- Campo 1: userId (Ascendente)
- Campo 2: createdAt (Descendente)

Puedes crear el índice automáticamente usando este enlace cuando Firebase lo sugiera, o manualmente:
1. Ve a [Firebase Console](https://console.firebase.google.com/)
2. Selecciona tu proyecto
3. Ve a *Firestore Database* > *Índices*
4. Haz clic en *Crear Índice* y configura los campos mencionados

Validación de Formularios
- Validación de email y contraseña
- Validación de montos y descripciones
- Mensajes de error amigables en español

Cálculo de costos:

Inversión inicial (CAPEX)

INVERSION INICIAL (CAPEX)		
Concepto	**Costo**	**Detalle**
Laptop Desarrollador	1000	1 desarrollador
Herramientas IA	20	para asistir al desarrollador
Configuraciones Backend	300	firebase, licencias ide, etc
Costo API Asistencia IA	100	Gemini API comercial que incorpora al asistente
Muebles y equipos de oficina	75	
Total CAPEX	**1495**	

Gastos mensuales u operativos (OPEX)

GASTOS OPERATIVOS (OPEX)		
Concepto	**Costo**	**Detalle**
Desarrolladores	1000	salario
Servicios básicos	100	luz, agua, internet, teléfono
Firebase blaze	1.84	100k docs leidos y escritos, storage, hosting,etc
Servicios de API asistente	20	
Total OPEX	**1121.84**	

Punto de equilibrio estimado:

PUNTO DE EQUILIBRIO	
Punto de equilibrio = (CAPEX + (OPEX * 12)) / Ingreso por usuario	
Inversion Inicial	1495
Gastos Operativos	1121.84
Suscripción APP	2.49
Punto de Equilibrio	6006.859438

¿cuántos usuarios o clientes necesitarían para recuperar la inversión? **6006**

Escenario de rentabilidad

1. Costo de Suscripción de 2.49 (por opciones premium)
2. Anuncios al exceder 5 transacciones en gastos
3. Asociarme con Restaurantes, supermercados, farmacias, etc para promocionar descuentos.