💰 Finanzas Familia Vásquez - Documentación Técnica

Versión: 1.0.0 **Estado:** Producción (Self-Hosted)

📋 Descripción General

Aplicación web progresiva (PWA) diseñada para la gestión financiera familiar. Permite el registro de gastos de tarjeta de crédito, control de ingresos/egresos recurrentes (fijos) y conciliación bancaria. La aplicación está optimizada para dispositivos móviles ("Mobile First") y funciona en un entorno autohospedado para garantizar la privacidad total de los datos.

🏗️ Arquitectura del Sistema

El proyecto sigue una arquitectura **Monolítica Modular** basada en el stack T3 (simplificado), optimizada para despliegues ligeros en servidores Linux (Ubuntu).

Diagrama de Componentes

[Cliente (Navegador)] <---> [Next.js Server (App Router)] <---> [Prisma ORM] <---> [SQLite DB (Archivo Local)]

1. **Frontend (UI):** Renderizado híbrido (Server/Client Components). Interfaz reactiva construida con Tailwind CSS.
2. **Backend (API):** Server Actions de Next.js. Elimina la necesidad de una API REST separada, permitiendo comunicación directa y segura con la base de datos.
3. **Persistencia:** Base de datos relacional SQLite embebida. Los datos viven en un archivo `.db` dentro del servidor, facilitando backups simples (copiar y pegar).
4. **Contenerización:** Docker encapsula todo el entorno (Node.js, dependencias, DB) para que corra idéntico en desarrollo y producción.

🛠️ Tecnologías y Stack

Lenguajes & Frameworks

- **TypeScript:** Lenguaje principal. Provee tipado estático para reducir errores en tiempo de ejecución.
- **Next.js 14 (App Router):** Framework Full-Stack. Maneja el enrutamiento, la API y la optimización de imágenes/fuentes.
- **React 18:** Librería de UI para componentes interactivos.

Estilos & UI

- **Tailwind CSS:** Framework de utilidades para diseño rápido y responsive.
- **Lucide React:** Librería de iconos vectoriales ligeros y consistentes (+60 iconos implementados).

- **Recharts:** Librería de visualización de datos para los gráficos de barras y torta.

Base de Datos & Backend

- **SQLite:** Motor de base de datos SQL ligero y sin servidor.
- **Prisma ORM:** Capa de abstracción de base de datos. Permite interactuar con SQLite usando TypeScript en lugar de SQL puro.

Infraestructura

- **Docker:** Para crear la imagen del contenedor.
- **Docker Compose:** Para orquestar el servicio y gestionar volúmenes persistentes.

📂 Estructura de Datos (Schema)

La base de datos consta de 3 tablas principales definidas en `prisma/schema.prisma` :

1. **Transaction (Movimientos):**
   - Registra cada gasto individual.
   - Campos clave: `amount` (Monto), `isPaid` (Estado conciliación), `week` (Agrupación temporal), `sub` (Comercio/Detalle).
2. **Recurring (Fijos):**
   - Gestiona ingresos y gastos fijos mensuales.
   - Campos clave: `type` (Ingreso/Gasto), `owner` (Daniel/Gedalya/Ambos).
3. **Category (Configuración):**
   - Almacena la personalización de categorías e iconos.
   - Permite que la configuración visual persista entre sesiones.

🚀 Guía de Despliegue (Servidor Ubuntu)

Prerrequisitos

- Servidor Ubuntu 20.04/22.04.
- Docker y Docker Compose instalados.

Pasos de Instalación

1. **Transferir Archivos:** Copia la carpeta del proyecto al servidor (usando SCP o Git).
2. **Construir Contenedor:**

```
docker-compose up -d --build
```

3. **Verificar Estado:**

```
docker-compose logs -f
```

4. **Acceso:** La aplicación estará disponible en el puerto `3000` de tu servidor.

Mantenimiento y Backups

La base de datos se guarda en el volumen `./db-data`. Para hacer un backup, simplemente copia el archivo `dev.db` o la carpeta completa a una ubicación segura (S3, Google Drive, etc.).

✨ Funcionalidades Clave

1. **Conciliación de Tarjeta:** Módulo específico para marcar gastos como "Pagados" a la tarjeta de crédito.
2. **Gestión de Fijos:** Balance personal (Daniel vs Gedalya) de ingresos y gastos fijos.
3. **Importación IA:** Interfaz dedicada para pegar datos CSV generados por Inteligencia Artificial (Gemini/GPT) a partir de fotos de facturas.
4. **Dashboard Inteligente:** Gráficos con filtros temporales (Semana/Mes/Año) y navegación histórica.