

🤖 MiDinero - Contexto Completo para IAs (ChatGPT, Claude, etc.)

> **Copia este archivo completo cuando uses otra IA para continuar el proyecto**

📊 ¿Qué es MiDinero?

Aplicación web de **control de gastos personales con IA** que ayuda a personas del común a manejar su dinero de forma casual y amigable.

Características Principales

- ✅ Login/Registro con Supabase Auth
- ✅ Dashboard con gráficos interactivos
- ✅ Categorización automática de gastos usando IA
- ✅ Análisis inteligente de patrones de gasto
- ✅ Sugerencias personalizadas de ahorro
- ✅ Modo oscuro con tema verde (dinero/economía)
- ✅ Completamente responsive
- ✅ Presentación interactiva integrada

🏗️ Arquitectura Técnica

Frontend

```

React 18 + TypeScript + Vite

- └─ Tailwind CSS v4.0 (estilos)
- └─ Shadcn/ui (componentes UI)
- └─ Motion/React (animaciones)
- └─ Recharts (gráficos)
- └─ Lucide React (iconos)
- └─ React Hook Form + Zod (formularios)
- └─ Sonner (notificaciones toast)

...

### ### Backend (Arquitectura Híbrida)

...

#### 1. Supabase (Auth + PostgreSQL)

- └─ Autenticación de usuarios
- └─ Base de datos principal
- └─ Storage de archivos

#### 2. Kotlin + Spring Boot (API REST)

- └─ Puerto: 8080
- └─ Endpoints: /api/expenses, /api/categories
- └─ Gestión de gastos y categorías

#### 3. Python + FastAPI (Microservicio de IA)

- └─ Puerto: 8000
- └─ Endpoints: /api/ai/categorize, /api/ai/analyze
- └─ Machine Learning para categorización y análisis

...

### ### Base de Datos (PostgreSQL/Supabase)

```sql

-- Tablas principales

- users (Supabase Auth)

- expenses (id, user_id, amount, category, description, date, created_at)

- categories (id, name, icon, color, user_id)

- user_preferences (id, user_id, currency, language, theme)

...

📁 Estructura de Archivos Actual

...

MiDinero/

├─ App.tsx # Componente principal

├─ components/

| └─ auth/

| | └─ login-form.tsx # Formulario de login

| | └─ register-form.tsx # Formulario de registro

| └─ dashboard/

| | └─ ai-insights.tsx # Insights de IA

| | └─ currency-settings.tsx # Configuración de moneda

| | └─ expense-form.tsx # Formulario agregar gasto

| | └─ expense-list.tsx # Lista de gastos

| | └─ expense-summary.tsx # Resumen y gráficos

| └─ presentation/

| | └─ presentation-mode.tsx # Presentación interactiva

| └─ theme-provider.tsx # Provider de tema

| └─ theme-toggle.tsx # Toggle dark mode

| └─ ui/ # Componentes Shadcn (NO MODIFICAR)

```

├─ utils/
|   ├─ api.tsx          # API original (deprecated)
|   ├─ api-kotlin.tsx   # Cliente API Kotlin
|   ├─ hybrid-api.tsx   # API híbrida (USAR ESTE)
|   ├─ mock-auth.tsx    # Auth mock para testing
|   └─ supabase/
|       ├─ client.tsx    # Cliente Supabase
|       └─ info.tsx      # Info de configuración
├─ styles/
|   └─ globals.css       # Estilos globales + tema
├─ package.json          # Dependencias
├─ vite.config.ts        # Config Vite
└─ tsconfig.json         # Config TypeScript
...

```

🎨 Sistema de Diseño

Paleta de Colores (Tema Oscuro Verde)

```
``css
```

```
/* Verdes principales (dinero/economía) */
```

```
--primary: #10b981 /* emerald-500 */
```

```
--primary-dark: #059669 /* emerald-600 */
```

```
--primary-light: #34d399 /* emerald-400 */
```

```
/* Fondos oscuros */
```

```
--background: #0a0a0a /* Negro suave */
```

```
--surface: #111111 /* Superficie elevada */
```

```
--card: #1a1a1a    /* Cards */
```

```
/* Textos */
```

```
--foreground: #ffffff /* Texto principal */
```

```
--muted: #737373    /* Texto secundario */
```

```
/* Estados */
```

```
--success: #10b981
```

```
--warning: #f59e0b
```

```
--error: #ef4444
```

```
--info: #3b82f6
```

```
...
```

Tipografía

```
```css
```

```
/* NO usar clases de Tailwind para estas propiedades */
```

```
/* Están definidas en globals.css */
```

```
h1 { font-size: 2.5rem; font-weight: 700; }
```

```
h2 { font-size: 2rem; font-weight: 600; }
```

```
h3 { font-size: 1.5rem; font-weight: 600; }
```

```
body { font-size: 1rem; line-height: 1.5; }
```

```
...
```

### ### Componentes UI Disponibles (Shadcn)

- Button, Card, Dialog, Sheet, Tabs

- Form, Input, Select, Checkbox, Switch

- Alert, Toast (Sonner), Skeleton

- Table, Avatar, Badge, Progress

- Chart (wrapper de Recharts)
- Accordion, Collapsible, Dropdown
- Y 40+ más en /components/ui/

---

## ## 🍷 Endpoints de API

### ### Kotlin Backend (http://localhost:8080)

```typescript

// Gastos

GET /api/expenses // Obtener todos los gastos

POST /api/expenses // Crear gasto

PUT /api/expenses/{id} // Actualizar gasto

DELETE /api/expenses/{id} // Eliminar gasto

// Categorías

GET /api/categories // Obtener categorías

POST /api/categories // Crear categoría

PUT /api/categories/{id} // Actualizar categoría

DELETE /api/categories/{id} // Eliminar categoría

// Headers requeridos

Authorization: Bearer {supabase_token}

Content-Type: application/json

```

### ### Python IA Backend (http://localhost:8000)

```
``typescript
```

```
// Categorización automática
```

```
POST /api/ai/categorize
```

```
Body: { description: string, amount: number }
```

```
Response: { category: string, confidence: number }
```

```
// Análisis de patrones
```

```
POST /api/ai/analyze
```

```
Body: { expenses: Expense[], period: string }
```

```
Response: {
```

```
 trends: Trend[],
```

```
 insights: string[],
```

```
 suggestions: string[]
```

```
}
```

```
// Predicción de gastos
```

```
POST /api/ai/predict
```

```
Body: { userId: string, category: string }
```

```
Response: { predicted_amount: number, next_month: string }
```

```
// Headers requeridos
```

```
Authorization: Bearer {supabase_token}
```

```
Content-Type: application/json
```

```
...
```

```
Supabase
```

```
``typescript
```

```
// Ya configurado en /utils/supabase/client.tsx
```

```
import { supabase } from './utils/supabase/client';
```

```
// Auth
```

```
await supabase.auth.signUp({ email, password });
```

```
await supabase.auth.signInWithPassword({ email, password });
```

```
await supabase.auth.signOut();
```

```
// Database
```

```
const { data, error } = await supabase
```

```
 .from('expenses')
```


```
 .select('*')
```

```
 .eq('user_id', userId);
```

```
...
```

```

```

```
 Código de Ejemplo
```

```
Ejemplo 1: Agregar Gasto con IA
```

```
``typescript
```

```
// components/dashboard/expense-form.tsx
```

```
import { useState } from 'react';
```

```
import { hybridAPI } from '../utils/hybrid-api';
```

```
import { Button } from '../ui/button';
```

```
import { Input } from '../ui/input';
```

```
import { toast } from 'sonner@2.0.3';
```



```
export function ExpenseForm() {
 const [description, setDescription] = useState("");
 const [amount, setAmount] = useState("");
 const [loading, setLoading] = useState(false);

 const handleSubmit = async (e: React.FormEvent) => {
 e.preventDefault();
 setLoading(true);

 try {
 // 1. Categorizar con IA (Python)
 const aiResult = await hybridAPI.categorizeExpense({
 description,
 amount: parseFloat(amount)
 });

 // 2. Guardar en backend (Kotlin)
 await hybridAPI.createExpense({
 description,
 amount: parseFloat(amount),
 category: aiResult.category,
 date: new Date().toISOString()
 });

 toast.success('¡Gasto agregado! 🟢');
 setDescription("");
 setAmount("");
 } catch (error) {
 toast.error('Uy, algo salió mal 😞');
```

```

 } finally {
 setLoading(false);
 }
};

return (
 <form onSubmit={handleSubmit} className="space-y-4">
 <Input
 placeholder="¿En qué gastaste?"
 value={description}
 onChange={(e) => setDescription(e.target.value)}
 />
 <Input
 type="number"
 placeholder="¿Cuánto?"
 value={amount}
 onChange={(e) => setAmount(e.target.value)}
 />
 <Button type="submit" disabled={loading}>
 {loading ? 'Guardando...' : 'Agregar Gasto'}
 </Button>
 </form>
);
}
...

```

### ### Ejemplo 2: Gráfico de Gastos

```
``typescript
```

```

// components/dashboard/expense-chart.tsx

import { BarChart, Bar, XAxis, YAxis, Tooltip, ResponsiveContainer } from 'recharts';
import { Card } from '../ui/card';

interface ExpenseChartProps {
 data: Array<{ category: string; amount: number }>;
}

export function ExpenseChart({ data }: ExpenseChartProps) {
 return (
 <Card className="p-6">
 <h3 className="mb-4">Gastos por Categoría</h3>
 <ResponsiveContainer width="100%" height={300}>
 <BarChart data={data}>
 <XAxis dataKey="category" stroke="#737373" />
 <YAxis stroke="#737373" />
 <Tooltip
 contentStyle={{
 backgroundColor: '#1a1a1a',
 border: '1px solid #10b981'
 }}
 />
 <Bar dataKey="amount" fill="#10b981" radius={[8, 8, 0, 0]} />
 </BarChart>
 </ResponsiveContainer>
 </Card>
);
}

```

### ### Ejemplo 3: Hybrid API Helper

```
``typescript
// utils/hybrid-api.tsx
import { supabase } from './supabase/client';

const KOTLIN_API = 'http://localhost:8080';
const PYTHON_API = 'http://localhost:8000';

async function getAuthToken() {
 const { data: { session } } = await supabase.auth.getSession();
 return session?.access_token;
}

export const hybridAPI = {
 // Kotlin endpoints
 async getExpenses() {
 const token = await getAuthToken();
 const res = await fetch(`${KOTLIN_API}/api/expenses`, {
 headers: { 'Authorization': `Bearer ${token}` }
 });
 return res.json();
 },

 async createExpense(expense: Expense) {
 const token = await getAuthToken();
 const res = await fetch(`${KOTLIN_API}/api/expenses`, {
 method: 'POST',
```

```

headers: {
 'Authorization': `Bearer ${token}`,
 'Content-Type': 'application/json'
},
body: JSON.stringify(expense)
});
return res.json();
},

```

// Python AI endpoints

```

async categorizeExpense(data: { description: string; amount: number }) {
 const token = await getAuthToken();
 const res = await fetch(`${PYTHON_API}/api/ai/categorize`, {
 method: 'POST',
 headers: {
 'Authorization': `Bearer ${token}`,
 'Content-Type': 'application/json'
 },
 body: JSON.stringify(data)
 });
 return res.json();
},

```

```

async analyzeExpenses(expenses: Expense[]) {
 const token = await getAuthToken();
 const res = await fetch(`${PYTHON_API}/api/ai/analyze`, {
 method: 'POST',
 headers: {
 'Authorization': `Bearer ${token}`,

```

```
'Content-Type': 'application/json'
},
body: JSON.stringify({ expenses })
});
return res.json();
}
};
...

```

## 🎯 Tono y Estilo de la App

### ❌ NO Hacer (Formal)

...

"Se ha detectado un incremento del 23% en la categoría alimentación"

"Por favor, ingrese la descripción del gasto"

"Error: La operación no pudo completarse"

...

### ✅ Sí Hacer (Casual)

...

"¡Ey! Gastaste bastante en comida esta semana 🤪"

"¿En qué gastaste?"

"Uy, algo salió mal. Intenta de nuevo 🙌"

...

### Ejemplos de Mensajes

```
```typescript
```

```
// Insights de IA
```

```
"💡 Tip: Podrías ahorrar $250 si reduces un café por día"
```

```
"🇲🇪 Este mes gastaste menos que el anterior. ¡Vas bien! 🎉"
```

```
"⚠️ Cuidado, ya llevas $800 en salidas. Tal vez modera un poco 😬"
```

```
// Validaciones
```

```
"Ey, necesitas poner cuánto gastaste"
```

```
"El monto debe ser mayor a $0, ¡no regalamos dinero! 😂"
```

```
// Éxito
```

```
"¡Listo! Gasto agregado 💰"
```

```
"¡Categoría creada! 🎨"
```

```
"¡Guardado! Ahora relájate un poco 😌"
```

```
// Errores
```

```
"Uy, no pudimos guardar eso 😭"
```

```
"Algo falló. ¿Tienes internet? 🤖"
```

```
"Oops, hubo un error. Intenta de nuevo"
```

```
```
```

```

```

```
🚀 Comandos de Desarrollo
```

```
```bash
```

```
# Instalar dependencias
```

npm install

Ejecutar frontend (Puerto 5173)

npm run dev

Ejecutar backend Kotlin (Puerto 8080)

Ver: MiDinero-Guia-Desarrollo-Backend.md

Ejecutar backend Python (Puerto 8000)

Ver: MiDinero-Guia-Desarrollo-Backend.md

Build para producción

npm run build

Preview de build

npm run preview

...

🛠 Variables de Entorno (.env)

```env

# Supabase

VITE\_SUPABASE\_URL=https://tu-proyecto.supabase.co

VITE\_SUPABASE\_ANON\_KEY=tu-anon-key-aqui

# Backend URLs (opcional, defaults a localhost)

VITE\_KOTLIN\_API\_URL=http://localhost:8080



VITE\_PYTHON\_API\_URL=http://localhost:8000

# Modo de desarrollo

VITE\_DEV\_MODE=true

...

---

## 📦 Dependencias Clave

```json

{

"dependencies": {

"react": "^18.3.1",

"react-dom": "^18.3.1",

"tailwindcss": "^4.0.0",

"@supabase/supabase-js": "^2.x",

"motion": "^11.x", // Animaciones

"recharts": "^2.x", // Gráficos

"lucide-react": "latest", // Iconos

"react-hook-form": "^7.55.0",

"zod": "^3.x",

"sonner": "^2.0.3", // Toasts

"date-fns": "^3.x" // Manejo de fechas

}

}

```

---

## ## 🐛 Problemas Comunes y Soluciones

### ### 1. Error de CORS con Backend

```
``typescript
// Asegúrate de que el backend tenga CORS habilitado
// Kotlin: @CrossOrigin(origins = ["http://localhost:5173"])
// Python: app.add_middleware(CORSMiddleware, allow_origins=["*"])
...

```

### ### 2. Supabase Auth no funciona

```
``typescript
// Verifica que las URLs de redirección estén configuradas
// Supabase Dashboard → Authentication → URL Configuration
// Site URL: http://localhost:5173
...

```

### ### 3. Componentes Shadcn no aparecen

```
``typescript
// Asegúrate de importar desde ./components/ui/
import { Button } from './components/ui/button';
// NO: import { Button } from 'shadcn';
...

```

### ### 4. Estilos no se aplican

```
``typescript
// Verifica que globals.css esté importado en main.tsx
import './styles/globals.css';
...

```

---

## ## 📖 Documentación Adicional

El proyecto incluye documentación extensa:

- **\*\*MiDinero-Proyecto-Completo-Para-ChatGPT.md\*\*** - Contexto completo
- **\*\*MiDinero-Arquitectura-Backend-Completa.md\*\*** - Detalles de arquitectura
- **\*\*MiDinero-Guia-Desarrollo-Backend.md\*\*** - Setup de backends
- **\*\*GUIA-PRESENTACION-INTERACTIVA.md\*\*** - Modo presentación
- **\*\*GUIA-DESPLIEGUE.md\*\*** - Deploy a producción

---

## ## 🎓 Buenas Prácticas

### ### Código

1. ✅ Siempre usar TypeScript con tipos explícitos
2. ✅ Componentes funcionales con hooks
3. ✅ Separar lógica en custom hooks
4. ✅ Usar híbrida API (utils/hybrid-api.tsx)
5. ✅ Manejar loading y error states

### ### Estilos

1. ✅ NO usar: text-\*, font-\*, leading-\* de Tailwind
2. ✅ Usar componentes de /components/ui/
3. ✅ Mantener tema verde oscuro (#10b981)

4. ☒ Mobile-first responsive design

### ### UX

1. ☒ Tono casual y amigable
2. ☒ Usar emojis ocasionalmente 💰 💎 ✨
3. ☒ Feedback inmediato (toasts, loading)
4. ☒ Validaciones claras y útiles

---

## ## 🤖 Prompts Útiles para IAs

### ### Para ChatGPT/Claude

...

"Basándote en el contexto de MiDinero, agrega un componente de presupuestos mensuales que:

- Permita establecer un límite por categoría
- Muestre progreso visual (progress bar)
- Alerta cuando se acerque al límite (tono casual)
- Use el tema verde oscuro
- Se integre con la API híbrida"

...

...

"Mejora la función de análisis de IA en ai-insights.tsx para que detecte patrones de gasto recurrentes y sugiera optimizaciones. Mantén el tono casual de MiDinero"

...

...

"Crea un sistema de exportación de gastos a CSV/PDF que:

- Use los datos de Supabase
- Incluya filtros por fecha y categoría
- Tenga diseño consistente con MiDinero
- Botón de descarga con animación"

...

---

## ## Checklist de Features

### ### Implementadas

- [x] Login/Registro con Supabase
- [x] Dashboard con gráficos (Recharts)
- [x] Formulario de gastos
- [x] Lista de gastos
- [x] Categorización automática (IA)
- [x] Análisis de patrones
- [x] Modo oscuro verde
- [x] Responsive design
- [x] Presentación interactiva
- [x] API híbrida (Kotlin + Python)

### ### Pendientes/Sugeridas

- [ ] Sistema de presupuestos mensuales
- [ ] Notificaciones push

- [ ] Exportación CSV/PDF
- [ ] Recordatorios de gastos recurrentes
- [ ] Compartir gastos (grupos)
- [ ] Modo offline con sincronización
- [ ] Gráficos avanzados (tendencias, predicciones)
- [ ] Integración con bancos (Open Banking)

---

## ## 🎯 Resumen Ejecutivo

**\*\*MiDinero\*\*** es una app de control de gastos con IA, usando:

- **\*\*Frontend\*\***: React + TypeScript + Tailwind
- **\*\*Backend\*\***: Kotlin + Python + PostgreSQL/Supabase
- **\*\*Tema\*\***: Oscuro con verde (#10b981) - temática dinero
- **\*\*Tono\*\***: Casual, para gente del común
- **\*\*Features\*\***: Categorización IA, análisis, gráficos, sugerencias

**\*\*Archivos clave a revisar primero\*\***:

1. `/App.tsx` - Componente principal
2. `/utils/hybrid-api.tsx` - Cliente API
3. `/components/dashboard/expense-form.tsx` - Formulario principal
4. `/styles/globals.css` - Tema y estilos

**\*\*Siguiente paso sugerido\*\***: Agregar sistema de presupuestos mensuales

---

**\*\*Generado para uso con ChatGPT, Claude, Gemini, etc.\*\***

Versión: 1.0 | Última actualización: Octubre 2025