Especificação & Prompt — App de Gestão de Gastos (MVP Offline-First)

Autor: Rafael Ventura • Data gerada: 2025-09-05T17:27:36Z

1. Visão Geral

Construir uma aplicação web simples, bonita e moderna para gestão de gastos pessoais.

O MVP será offline-first utilizando armazenamento local do navegador (LocalStorage ou IndexedDB), com arquitetura em camadas preparada para conectar a uma API no futuro sem reescrever a UI.

Haverá login/registro já no MVP (conta local), visando reaproveitar preferências de usuário e facilitar a futura sincronização.

2. Escopo & Limites do MVP

• Usuário com conta local por navegador (sem multi-dispositivo/sync neste MVP).

• Moeda: BRL (R$). Datas e formatações pt-BR.

• Sem anexos de comprovantes inicialmente.

• Lançamentos manuais; recorrências podem ser pré-configuradas.

• Visualização de meses retroativos até a data de cadastro do app.

• Volume esperado: resistente a milhares de lançamentos; foco em performance no navegador.

3. Regras de Negócio

3.1 Transações

• Toda transação é ENTRADA (receita) ou SAÍDA (despesa) e possui: data, valor (>0), categoria. Campos opcionais: descrição, forma de pagamento, cartão, etiquetas.

• Edição e exclusão permitidas; manter updatedAt.

Filtros para visualizar bem

3.2 Salário & Receitas

• Salário fixo mensal configurável em “Configurações”.

• Receitas avulsas (não-recorrentes) são lançadas normalmente em “Lançamentos”.

3.3 Categorias

• Categorias são editáveis. Ao excluir uma categoria em uso:

– Opção A: excluir também os lançamentos associados (com confirmação forte).

– Opção B: reatribuir os lançamentos para outra categoria antes de remover.

• (Opcional) Categorias podem ter tipo sugerido (Receita/Despesa) para facilitar filtros.

3.4 Cartão de Crédito

• Lançamentos podem marcar “forma de pagamento = Cartão” e selecionar um “Cartão de crédito X”.

• Totais do mês somam gastos “normais” + gastos no cartão com base na DATA DO LANÇAMENTO (MVP).

• (Evolução futura) Modo “Consolidar por fatura”: cada cartão tem dia de fechamento/vencimento; relatórios mensais podem somar pela fatura, não pela data de compra.

3.5 Relatórios e Períodos

• Relatório Mensal: totais de receitas, despesas e saldo; distribuição por categorias; tabela de lançamentos filtrável.

• Relatório Anual: visão 12 meses, receitas, despesas, saldo, % poupança (saldo/receitas).

• Comparação com meses anteriores e destaques de “Top categorias (despesa)”.

3.6 Configurações

• Dia do mês em que recebe (impacta apenas destaque visual/atalhos).

• Tema claro/escuro.

• Salário fixo e modelos de recorrência (somente geração manual no MVP).

4. Requisitos Funcionais (Casos de Uso)

RF001 — Login/Registro (conta local)

• Registro com nome, e-mail (opcional no MVP) e senha local (hash no storage). Login simples.

• Objetivo: permitir preferências por usuário e preparar futura migração para API.

• Recuperação de senha pode ser local (pergunta-seed) no MVP; e-mail fica para o backend futuro.

RF002 — Cadastrar Transação

• Campos: tipo (Receita/Despesa), data, categoria, valor (R$), (opcionais) descrição, forma de pagamento, cartão.

• Validações: data válida, valor > 0, categoria existente.

• Após salvar, atualizar KPIs do período selecionado.

RF003 — Editar/Excluir Transação

• Edição mantém histórico updatedAt; exclusão com confirmação.

RF004 — Gerir Categorias

• Criar/renomear/excluir categoria.

• Ao excluir, oferecer: (A) apagar lançamentos junto; (B) reatribuir lançamentos.

RF005 — Configurações

• Salário fixo mensal; dia de recebimento; tema (claro/escuro); import/export JSON; reset do app.

RF006 — Relatório Mensal (Dashboard)

• Seletor de mês/ano. KPIs: Receitas, Despesas, Saldo.

• Gráfico por categoria (despesas); tabela com filtros rápidos (tipo, categoria, texto).

RF007 — Relatório Anual

• Tabela com 12 linhas (1..12) somando receitas, despesas e saldo; gráfico de linhas.

• Indicador de % poupança por mês.

RF008 — Modelos de Recorrência (MVP manual)

• Cadastro de modelos (ex.: “Internet R$120/mês, categoria Contas, forma Pix”).

• Ação “Gerar lançamentos do mês” cria itens no período atual; usuário pode editar antes de salvar.

RF009 — Cartões de Crédito

• CRUD de cartões: nome, final, (futuro: fechamento/vencimento).

• Lançamento vincula a um cartão. Totais mensais incluem compras no cartão pela data do lançamento.

• Filtros por cartão nos relatórios.

5. Requisitos Não Funcionais (UX/UI & Arquitetura)

UX/UI

• Interface responsiva (mobile-first), tema claro/escuro, contrastes adequados (WCAG AA básico).

• Fluxos curtos: botão “+ Novo” sempre visível; atalhos de datas e categorias recentes.

• Feedbacks: toasts para salvar/editar/excluir; empty states didáticos.

• Tabelas com busca e filtros; paginação ou virtual scroll quando necessário.

• Campos com máscaras e formatações pt-BR (moeda, data).

Arquitetura (frontend)

• Framework: Angular 17+ (standalone components) com Angular Material (tema custom).

• Camadas:

– domain/: entidades, validadores, services de cálculo (ex.: MonthlySummaryService).

– ports/: contratos (StoragePort, AuthPort, TransactionRepository, CategoryRepository, CardRepository).

– adapters/:

* local/: LocalStorageAdapter/IndexedDBAdapter implementando StoragePort/AuthPort.

* http/ (futuro): HttpApiAdapter usando os mesmos ports.

– features/: dashboard, transactions, categories, cards, settings, auth.

– shared/: pipes, directives, componentes UI atômicos.

– core/: ThemeService, ErrorHandler, providers.

• Estado: services com Signals/RxJS; evitar over-engineering no MVP.

• Internacionalização: locale pt-BR para CurrencyPipe/DatePipe.

• Qualidade: TypeScript strict, ESLint+Prettier, testes unitários nos services de domínio.

6. Modelo de Dados (MVP)

User (local)

• id, name, email?, passwordHash, createdAt, updatedAt, preferences (theme, firstDayOfMonth).

Settings

• salaryFixedMonthly?: number; firstDayOfMonth: 1|…|31; theme: 'light'|'dark'|'system'; schemaVersion: number.

Category

• id, name, kind?: 'INCOME'|'EXPENSE'|'NEUTRAL', color?, createdAt, updatedAt.

Card

• id, label, last4?, createdAt, updatedAt. (Futuro: billingDay, dueDay)

Transaction

• id, userId, type: 'INCOME'|'EXPENSE', date (ISO YYYY-MM-DD), amount (centavos ou decimal padronizado), categoryId, paymentMethod?: 'PIX'|'CREDITO'|'DEBITO'|'DINHEIRO'|'OUTRO', cardId?, note?, createdAt, updatedAt.

RecurringModel (MVP manual)

• id, label, type, amount, categoryId, paymentMethod?, cardId?, frequency: 'MONTHLY'|'WEEKLY'|'YEARLY', day?: number, active: boolean, createdAt, updatedAt.

• Ação gera instâncias de Transaction no mês selecionado.

7. Autenticação & Conta (Login/Registro)

• Registro local com nome, (e-mail opcional) e senha. Armazenar passwordHash em storage local (com sal e hash; ex.: scrypt via Web Crypto).

• Login local valida contra o hash e guarda um token de sessão local.

• Futuro backend: AuthPort passa a delegar para API (JWT/OAuth2), preservando a mesma interface na UI.

8. Armazenamento & Sincronização

• StoragePort: load/save/remove por chave; implementação inicial LocalStorage (ou IndexedDB para maior volume).

• Chaves nomeadas por usuário: pf.<userId>.transactions.v1, pf.<userId>.categories.v1, etc.

• Export/Import JSON com `schemaVersion` para migrações.

• Migração futura: trocar adapter para HttpApiAdapter que fala com /auth, /transactions, /categories, /cards, /settings.

9. Relatórios (UX/BI)

• Dashboard Mensal: KPIs (Receitas, Despesas, Saldo), gráfico por categoria (despesas), tabela de lançamentos com filtros (tipo, categoria, cartão, texto).

• Página de Relatórios: comparativos com meses anteriores, Top categorias (despesa), visão anual (12 meses) com gráficos.

• Acessos rápidos: alterar mês/ano em um seletor global.

10. Configurações

• Salário fixo mensal (R$).

• Dia do mês que recebe (impacta destaques/atalhos).

• Tema (claro/escuro/sistema).

• Modelos de recorrência (cadastro/editar/ativar/desativar; geração manual).

• Importar/Exportar JSON; Reset do app (confirmação dupla).

11. Fluxos de Usuário (User Stories)

• Como usuário, quero registrar uma despesa rápida informando valor, categoria e data para controlar meus gastos.

• Como usuário, quero cadastrar meu salário fixo para ver automaticamente o saldo do mês.

• Como usuário, quero criar modelos de lançamentos recorrentes e gerar os do mês com um clique.

• Como usuário, quero marcar um gasto como “Cartão de crédito X” para filtrar e somar junto com os gastos normais.

• Como usuário, quero ver relatórios mensais e anuais com gráficos claros e filtros simples.

• Como usuário, quero exportar meus dados em JSON para backup.

12. Critérios de Aceite (exemplos)

Cadastro simples

Dado que estou na tela de “Novo lançamento”

Quando seleciono Despesa, data válida, categoria “Alimentação”, valor R$ 35,00

Então o lançamento aparece na lista do mês e “Despesas” aumenta em R$ 35,00.

Salário fixo + alerta

Dado que configurei salário fixo de R$ 3.000

E criei uma receita “Salário” no mês corrente

Então o sistema sinaliza possível contagem dupla no resumo mensal.

Excluir categoria com uso

Dado que a categoria “Transporte” possui lançamentos

Quando tento excluí-la

Então devo escolher entre apagar também os lançamentos ou reatribuir para outra categoria antes de confirmar.

13. Arquitetura Frontend (Angular)

Padrões

• Angular standalone components, Angular Material, tema custom (light/dark).

• Services de domínio com Signals/RxJS; pipes pt-BR para moeda/data.

Ports (interfaces)

• StoragePort { load<T>(key): Promise<T|null>; save<T>(key,data): Promise<void>; remove(key): Promise<void> }

• AuthPort { register(...): Promise<User>; login(...): Promise<Session>; logout(): Promise<void> }

• TransactionRepository, CategoryRepository, CardRepository (CRUD + queries por mês/ano).

Adapters

• LocalStorageAdapter (MVP): serializa JSON com schemaVersion; chaves por userId.

• HttpApiAdapter (futuro): usa fetch/HttpClient mantendo as mesmas interfaces.

Estrutura de pastas (sugestão)

app/

core/

domain/

ports/

adapters/

local/

http/

features/

auth/

dashboard/

transactions/

categories/

cards/

settings/

shared/

Build/Qualidade

• TypeScript strict; ESLint+Prettier; testes unitários dos services de cálculo.

14. Contratos de API (Futuro, referência)

Auth

• POST /auth/register {name,email?,password} → {user,jwt}

• POST /auth/login {email,password} → {user,jwt}

Transactions

• GET /transactions?month=&year=&type=&categoryId=&cardId=

• POST /transactions

• PUT /transactions/:id

• DELETE /transactions/:id

Categories, Cards, Settings: CRUD equivalentes.

Observação: o frontend falará com a API por meio de ports; a UI não muda ao trocar os adapters.

15. Roadmap de Evolução

Fase 1 (MVP): Auth local, CRUD transações/categorias/cartões, salário fixo, modelos de recorrência (geração manual), relatórios mensal/anual, export/import JSON, tema dark.

Fase 2: IndexedDB, fechamento mensal, orçamentos por categoria, filtros avançados, acessibilidade reforçada.

Fase 3: Backend (login real, sync), recorrências automáticas, consolidação por fatura (cartão), anexos, multi-dispositivo.

16. Anexos — Convenções

• Datas em ISO (YYYY-MM-DD) internamente.

• Valores tratados de forma consistente (decidir centavos inteiros ou decimal e padronizar).

• IDs como UUID v4.

• Mensagens de erro amigáveis ao usuário (sem detalhes técnicos).