Documento de Requisitos Funcionais — MVP

Índice

Projeto: Aplicativo de Gestão Financeira Pessoal (YNAB Clone)

---

1. Introdução

Este documento descreve os requisitos funcionais mínimos (MVP) para o desenvolvimento de um aplicativo de controle financeiro pessoal, com foco em usabilidade offline, controle de orçamento mensal e gestão de contas e lançamentos.

---

2. Objetivo do Produto

Desenvolver uma aplicação desktop que permita ao usuário:

- Gerenciar contas bancárias e lançamentos;
- Categorizar receitas, despesas e transferências;
- Definir e acompanhar limites de orçamento mensal;
- Operar sem conexão com a internet (modo offline).

---

3. Escopo do MVP

O MVP incluirá funcionalidades essenciais para permitir ao usuário controlar seu orçamento mensal. Funcionalidades avançadas como relatórios detalhados e aplicativos móveis serão planejadas para etapas futuras.

---

4. Arquitetura Técnica

4.1. Backend

- Framework: Spring Boot
- Banco de dados: SQLite
- API RESTful com endpoints para todas as entidades descritas abaixo
- Suporte a operação local e offline

4.2. Frontend

- Framework: Angular
- Empacotado com Electron para aplicação desktop
- Telas previstas:
    - Login;
    - Gestão de contas;
    - Gestão de categorias/subcategorias;
    - Lançamento de transações;
    - Visualização do orçamento mensal.

Observação: A interface do usuário será em português, mas todos os nomes internos, endpoints e modelos de dados estarão em inglês.

---

5. Requisitos Funcionais

5.1. Gestão de Contas

Descrição: Permitir o gerenciamento de contas financeiras.

Funcionalidades:

- CRUD completo para contas;
- Campos obrigatórios:

    - `name`
    - `type` (CHECKING, SAVINGS, INVESTMENT, CASH, CREDIT)
    - `balance`
    - `isBudgetIncluded` (boolean)
    - `balanceDate`
- Cálculo automático:
    - **Current Balance** baseado em transações finalizadas;
    - **Projected Balance** baseado em todas as transações, incluindo futuras;
- Listagem de contas (card): `name`, saldos
- Listagem de contas (tabela de edição): `id`, `name`, saldos, `type`, `isBudgetIncluded`, `createdAt`, `updatedAt`
- Não permitir saldo negativo em contas do tipo `CASH` (regra de negócio).
- Informar mensagem de erro ao registrar no caso de um lançamento que torne a conta do tipo `CASH` negativa

---

5.2. Gestão de Categorias e Subcategorias

Descrição: Permitir a organização de despesas e receitas por categorias hierárquicas.

Funcionalidades:

- CRUD completo para categorias e subcategorias;
- Categoria:
    - Campo `name` (nome da categoria)
- Subcategoria vinculada à categoria, com:
    - Campo `name` (nome da subcategoria)
- Listagem de categorias e subcategorias para seleção em lançamentos.
- Não permitir exclusão de categorias/subcategorias que possuam lançamentos ou orçamentos vinculados (bloqueio ou soft delete).

5.3. Gestão de Lançamentos

Descrição: Registrar movimentações financeiras.

Funcionalidades:

- CRUD para lançamentos dos tipos:
    - Receita (`INCOME`)
    - Despesa (`EXPENSE`)
    - Transferência (`TRANSFER`)
- Campos obrigatórios:
    - `date`
    - `amount`
    - `transactionType`
    - `fromAccount` (conta origem)
    - `toAccount` (conta destino, para transferências)
    - `payee`

    - `subcategory`
    - `description`
    - `status` (enum: PENDING, CLEARED)
    - `recurrenceType`
    - `recurrenceFrequency`
    - `installmentCount` (se parcelado)
- Permitir lançamentos futuros (agendamento)
- Atualização automática dos saldos de contas e orçamento ao finalizar lançamentos
- Validação de consistência para transferências (afetar ambas as contas corretamente)
- Transferências devem sempre criar dois lançamentos vinculados pelo `transferGroupId`
- Não permitir orçamento negativo

5.3.1 Lançamentos Recorrentes

Descrição

Permitir o agendamento automático de lançamentos recorrentes (como assinaturas e pagamentos periódicos), com frequência definida pelo usuário. O sistema deve permitir editar ou excluir lançamentos individuais de uma sequência, bem como cancelar ou modificar toda a sequência.

Funcionalidades

- Marcar um lançamento como recorrente na criação, com tipos:
    - `FIXED`: repetido indefinidamente conforme frequência
    - `INSTALLMENT`: repetido por um número definido de parcelas
    - `NONE`: lançamento único
- Parâmetros:
    - `recurrenceFrequency`: DAILY, WEEKLY, BIWEEKLY, MONTHLY, BIMONTHLY, TRIMONTHLY, SIXMONTHLY, YEARLY
    - `installmentCount`: obrigatório apenas para parcelados
- Geração automática de lançamentos futuros conforme configuração:
    - Para `FIXED`: até 12 meses à frente (configurável até 5 anos)
    - Para `INSTALLMENT`: todas as parcelas geradas imediatamente
    - Lançamentos criados com:
        - Mesmo conteúdo do original (exceto data)
        - `status = PENDING`
        - `groupId` compartilhado
        - `manualOverride = false`
- Lançamentos recorrentes:
    - Listados junto aos demais
    - Mantêm vínculo via `groupId`
    - Permitem atualização individual (`manualOverride = true`) ou em grupo

5.3.1.1 Controle de Edição de Lançamentos Recorrentes

Escopo de Edição

Ao editar um lançamento de uma sequência recorrente, o sistema deve perguntar:

- "Aplicar esta alteração em:"
  - Apenas esta ocorrência
  - Esta e as futuras

Comportamento:

- Se "apenas esta": define `manualOverride = true`, impedindo alterações futuras em grupo
- Se "esta e futuras":
  - Atualiza lançamentos futuros (baseado na data)
  - Remove `manualOverride` de todos os futuros do grupo e atualiza

Campo `manualOverride`

- Indica que o lançamento foi editado manualmente e não deve ser alterado automaticamente
- Padrão: `false`
- Pode ser revertido manualmente ao editar uma ocorrência do grupo e escolher "esta e futuras"

Atualização de recorrência

- Executada localmente (sem dependência de rede)
- Pode ser feita:
  - Automaticamente ao abrir o app
  - Manualmente, via botão "Atualizar lançamentos recorrentes"

Validações

- Transferências geram dois lançamentos com `transferGroupId` compartilhado
- Apenas lançamentos com `status = PENDING` podem ser alterados por agendamento
- Lançamentos com `manualOverride = true` são protegidos de alterações em grupo, salvo confirmação do usuário

---

5.4. Orçamento Mensal por Categoria

Descrição: Acompanhar e controlar gastos mensais por categoria.

Funcionalidades:

- CRUD de limites de orçamento mensal por subcategoria cadastrada
- Visualização do orçamento por mês e agrupado por categoria/subcategoria
- Cálculo do saldo consumido e disponível (total mensal) com base em todos os lançamentos planejados (finalizados ou não)
- Atualização em tempo real conforme lançamentos são inseridos
- Alertas visuais para categorias com orçamento excedido

---

6. Requisitos Não Funcionais

- Interface responsiva e amigável
- Operação offline (sem necessidade de rede)
- Desempenho aceitável para operações CRUD em tempo real

- Persistência local via SQLite

---

7. Exclusões do MVP

- Relatórios detalhados (gráficos, comparativos de período)
- Aplicativo móvel
- Sincronização em nuvem
- Múltiplos usuários ou contas compartilhadas

---

8. Modelo de Dados

8.1. Account

- id: Long
- name: String
- type: Enum (CHECKING, SAVINGS, INVESTMENT, CASH, CREDIT)
- balance: BigDecimal // Atualizado dinamicamente com base em lançamentos finalizados
- isBudgetIncluded: Boolean // Indica se entra no orçamento
- balanceDate: LocalDateTime // Data do balanço
- createdAt: LocalDateTime
- updatedAt: LocalDateTime

8.2. Category

- id: Long
- name: String // Nome da categoria
- createdAt: LocalDateTime
- updatedAt: LocalDateTime

8.3. Subcategory

- id: Long
- name: String
- category: FK para Category
- createdAt: LocalDateTime
- updatedAt: LocalDateTime

8.4. Transaction

- id: Long
- fromAccount: FK para Account
- toAccount: FK para Account (destino, para transferências) // nullable se não for transferência
- subcategory: FK para Subcategory // nullable para transferências
- payee: String
- description: String
- amount: BigDecimal
- date: LocalDateTime

- transactionType: Enum (INCOME, EXPENSE, TRANSFER)
- transferGroupId: UUID (nullable) // Identificador compartilhado para transferências
- recurrenceType: Enum (FIXED, INSTALLMENT, NONE)
- recurrenceFrequency: Enum (DAILY, WEEKLY, BIWEEKLY, MONTHLY, BIMONTHLY, TRIMONTHLY, SIXMONTHLY, YEARLY) (nullable)
- installmentCount: Integer (nullable)
- groupId: UUID (nullable) // Identificador compartilhado para recorrências
- manualOverride: Boolean // padrão false, impede alterações em grupo
- status: Enum (PENDING, CLEARED) // Indica o status do lançamento
- createdAt: LocalDateTime
- updatedAt: LocalDateTime

8.5. Budget

- id: Long
- subcategory: FK para Subcategory
- month: Integer (1-12)
- year: Integer
- plannedAmount: BigDecimal // Valor orçado
- createdAt: LocalDateTime
- updatedAt: LocalDateTime

8.6. Notas Técnicas

- Autenticação: Spring Security + JWT
- Criptografia de senha: BCrypt
- Middleware no front-end (Angular) para checar token e redirecionar para login
- O campo `balance` pode ser calculado ou armazenado com triggers
- Modelos `Category` e `Subcategory` têm relação 1:N (uma categoria para várias subcategorias)
- Subcategorias são os nós finais da hierarquia — lançamentos e orçamentos usam apenas elas
- Transferências são controladas por `transferGroupId`
- O campo `payee` é uma string simples, sem FK para outra tabela por enquanto
- Soft delete: adicionar campo `isDeleted` (Boolean) em Category e Subcategory para evitar perda de dados acidental
- Todos os modelos possuem campos `createdAt` e `updatedAt`
- Preparar FK para usuário nas entidades principais para facilitar multiusuário no futuro

9. Especificação da API REST (Endpoints)

Todos os endpoints e modelos de dados usam nomes em inglês. A interface do usuário será em português.

9.1 Accounts

- `GET /accounts`
  Lista todas as contas do usuário.

- `POST /accounts`
  Cria uma nova conta.
- `GET /accounts/{id}`
  Detalha uma conta específica.
- `PUT /accounts/{id}`
  Atualiza uma conta existente.
- `DELETE /accounts/{id}`
  Remove uma conta.

9.2 Categories

- `GET /categories`
  Lista todas as categorias.
- `POST /categories`
  Cria uma nova categoria.
- `GET /categories/{id}`
  Detalha uma categoria específica.
- `PUT /categories/{id}`
  Atualiza uma categoria existente.
- `DELETE /categories/{id}`
  Remove uma categoria (bloqueia se houver subcategorias ou lançamentos vinculados).

9.3 Subcategories

- `GET /subcategories`
  Lista todas as subcategorias.
- `POST /subcategories`
  Cria uma nova subcategoria.
- `GET /subcategories/{id}`
  Detalha uma subcategoria específica.
- `PUT /subcategories/{id}`
  Atualiza uma subcategoria existente.
- `DELETE /subcategories/{id}`
  Remove uma subcategoria (bloqueia se houver lançamentos ou orçamentos vinculados).

9.4 Transactions

- `GET /transactions`
  Lista todos os lançamentos (com filtros opcionais: data, conta, categoria, status, etc.).
- `POST /transactions`
  Cria um novo lançamento.
- `GET /transactions/{id}`
  Detalha um lançamento específico.
- `PUT /transactions/{id}`
  Atualiza um lançamento existente.
- `DELETE /transactions/{id}`
  Remove um lançamento.

9.5 Budgets

- `GET /budgets`
  Lista todos os orçamentos mensais.
- `POST /budgets`
  Cria um novo orçamento mensal.
- `GET /budgets/{id}`
  Detalha um orçamento mensal específico.
- `PUT /budgets/{id}`
  Atualiza um orçamento mensal existente.
- `DELETE /budgets/{id}`
  Remove um orçamento mensal.
- `GET /reports/monthly-summary`
  Obtém resumo mensal de receitas e despesas.
- `GET /reports/balance`
  Obtém saldo atual por conta ou total consolidado.

Notas:

- Recursos são organizados por usuário autenticado.
- Campos obrigatórios e estrutura JSON serão definidos na documentação detalhada da API.
- Especificar respostas de erro para validação, não encontrado e conflito.

---

10. Considerações Finais

Este documento serve como base para o desenvolvimento e validação do MVP. Mudanças e ajustes podem ser aplicados conforme o projeto evolui e com base no uso real.

Referências e Materiais de Apoio

Referência

- Exemplo de layout da tela de orçamentos
- Exemplo de layout da tela de transações

Materiais de apoio

- Protótipo Figma
- Diagrama de classes (PlantUML)
- Diagrama do banco (Mermaid)
- Arquivo SQL de criação do banco