Detalhes do Módulo Financeiro do Sistema de Gestão de Lojas

Este documento fornece informações detalhadas sobre a próxima fase de desenvolvimento: o Módulo Financeiro. Ele é destinado a complementar a visão geral do projeto e auxiliar outras IAs na compreensão e continuidade do trabalho.

1. Objetivo do Módulo Financeiro

O principal objetivo do Módulo Financeiro é fornecer uma visão clara e organizada das entradas e saídas de dinheiro da loja, permitindo o registro de transações, o acompanhamento de contas bancárias e a geração de relatórios financeiros básicos para auxiliar na gestão.

2. Funcionalidades Principais a Serem Implementadas

O módulo financeiro incluirá as seguintes funcionalidades essenciais:

a) Gestão de Contas Bancárias/Caixas Virtuais:

- Cadastro de Contas: Registrar diversas contas bancárias ou "caixas virtuais" (ex: conta corrente, poupança, conta PIX principal, investimentos).
- Visualização de Saldo: Exibir o saldo atual de cada conta.
- Edição/Exclusão: Manter os detalhes das contas atualizados.

b) Registro de Transações Financeiras:

- Entradas (Receitas): Registrar dinheiro recebido que não seja diretamente de vendas (ex: reembolso de fornecedor, empréstimos, aporte de capital).
- Saídas (Despesas): Registrar pagamentos de contas, aluguel, salários, despesas operacionais diversas.
- Transferências: Registrar movimentos de dinheiro entre as contas cadastradas (ex: da conta corrente para investimentos).
- Categorização de Transações: Atribuir categorias (ex: "Aluguel", "Salários", "Marketing", "Serviços") para uma melhor análise.

c) Relatórios Financeiros Básicos:

- Fluxo de Caixa Simplificado: Apresentar um resumo das entradas e saídas por período, consolidando dados de vendas, caixa e transações.
- Análise de Despesas por Categoria: Visualizar onde o dinheiro está sendo gasto.
- Resumo de Receitas: Detalhes sobre as fontes de receita (vendas, outras entradas).

3. Integração com Módulos Existentes

O módulo financeiro será altamente integrado com as funcionalidades de Vendas e Caixa já existentes para consolidar os dados financeiros da loja:

- Vendas (artifacts/{appId}/stores/{storeId}/sales):
    - As vendas finalizadas fornecerão os dados de receita (total, dinheiro, cartão, PIX) para os relatórios financeiros. O totalAmount e paymentMethod de cada venda serão utilizados.
- Gestão de Caixa (stores/{storeId}/cash_register_sessions e artifacts/{appId}/stores/{storeId}/cash_register_history):
    - Os supplies (suprimentos) e outflows (sangrias) registrados nas sessões de caixa serão considerados como movimentos financeiros de entrada/saída de caixa físico.
    - O dailySalesSummary (resumo diário de vendas) dentro do histórico de caixa será fundamental para compor o fluxo de caixa total.
- Pedidos de Compra (stores/{storeId}/purchase_orders) (Futuro):
    - Em uma fase futura, quando os pedidos de compra forem pagos, eles poderão gerar transações de despesa no módulo financeiro, relacionando o pagamento com o fornecedor e o pedido.

4. Estrutura de Dados no Firestore (Módulo Financeiro)

As novas coleções e a forma como os dados serão armazenados no Firestore são cruciais para o módulo financeiro.

a) bankAccounts/{bankAccountId} **(Coleção de nível raiz)**

- Propósito: Armazenar informações de contas bancárias ou caixas da empresa.
- documentId: ID gerado automaticamente pelo Firestore.
- Campos:
    - name: string (Ex: "Conta Corrente Banco X", "Caixa Loja Principal")
    - type: string (Ex: "bank_account", "cash_box", "investment_account")
    - bankName: string (Nome do banco, se for conta bancária)
    - accountNumber: string (Número da conta/identificador)
    - currentBalance: number (Saldo atual da conta. Será atualizado a cada transação relevante.)

- currency: string (Ex: "BRL", "USD")
- storeId: string (ID da loja à qual a conta pertence, se for específica de uma loja. Para contas corporativas, pode ser global.)
- createdAt: Timestamp
- updatedAt: Timestamp

b) transactions/{transactionId} **(Coleção de nível raiz)**

- Propósito: Registrar todas as transações financeiras (receitas, despesas, transferências) que não são vendas diretas ou suprimentos/sangrias de caixa (que já estão em cash_register_sessions).
- documentId: ID gerado automaticamente pelo Firestore.
- Campos:
  - timestamp: Timestamp (Data e hora da transação).
  - type: string (Ex: "revenue", "expense", "transfer").
  - amount: number (Valor da transação. Positivo para receita/entrada, negativo para despesa/saída).
  - description: string (Descrição breve da transação).
  - category: string (Ex: "Aluguel", "Salários", "Material de Escritório", "Reembolso", "Aporte de Capital").
  - sourceAccountId: string (ID da bankAccount de origem, para transferências e algumas despesas/receitas).
  - destinationAccountId: string (ID da bankAccount de destino, para transferências e algumas receitas/despesas).
  - relatedEntity: string (Opcional, nome do fornecedor/cliente/outra parte envolvida).
  - recordedBy: string (Email do usuário que registrou a transação).
  - storeId: string (ID da loja à qual a transação pertence. Essencial para filtragem por loja.)
  - isReconciled: boolean (Futuro: para conciliação bancária. Default: false).

5. Regras de Segurança do Firebase Firestore para o Módulo Financeiro

As regras de segurança serão ajustadas para dar acesso granular ao papel finance, além de admin.

- bankAccounts:
    - read e write: Permitido para usuários com role: 'admin' ou role: 'finance'.
    - Considerar storeId na regra se as contas forem exclusivas por loja.
- transactions:
    - read e create: Permitido para usuários com role: 'admin' ou role: 'finance'.
    - update e delete: Pode ser mais restrito (apenas admin ou limitado a um curto período após a criação) para manter a integridade dos registros financeiros.
    - Também será crucial filtrar por storeId para garantir que o usuário financeiro só veja as transações da sua loja.

Exemplo de como as regras podem ser expandidas (assumindo que bankAccounts e transactions são coleções de nível raiz, mas filtradas por storeId se storeId for definido no documento):

```
// Dentro de match /databases/{database}/documents { ... }


// Regras para 'bankAccounts'
match /bankAccounts/{bankAccountId} {
  allow read, write: if request.auth != null
    && (getUserData(request.auth.uid).role == 'admin' ||
getUserData(request.auth.uid).role == 'finance')
    // Adicionar esta condição se a conta for específica de uma loja
    && (resource.data.storeId == getUserData(request.auth.uid).storeId ||
resource.data.storeId == null);
}


// Regras para 'transactions'
match /transactions/{transactionId} {
```

```
    allow read: if request.auth != null
      && (getUserData(request.auth.uid).role == 'admin' ||
getUserData(request.auth.uid).role == 'finance')
      // A transação deve pertencer à loja do usuário
      && resource.data.storeId == getUserData(request.auth.uid).storeId;
    allow create: if request.auth != null
      && (getUserData(request.auth.uid).role == 'admin' ||
getUserData(request.auth.uid).role == 'finance')
      // A transação que está sendo criada deve ser para a loja do usuário
      && request.resource.data.storeId == getUserData(request.auth.uid).storeId;
    // Pode-se restringir update/delete para manter a integridade
    allow update, delete: if request.auth != null &&
getUserData(request.auth.uid).role == 'admin';
}
```

6. Desafios e Considerações

- Consistência de Saldo: A atualização de currentBalance nas bankAccounts deve ser tratada com cuidado, possivelmente usando transações atômicas para evitar condições de corrida em um ambiente multiusuário.
- Agregação de Dados: A geração de relatórios de fluxo de caixa e despesas exigirá consultas eficientes e possivelmente agregação de dados de múltiplas coleções (vendas, caixa, transações).
- Localização: Valores monetários e datas devem ser formatados corretamente para o padrão brasileiro.

Com estas informações, a outra IA deve ter uma base sólida para começar a trabalhar no módulo financeiro.