Planejamento do projeto

Resumo:

Projeto backend em desenvolvimento com foco em Java e banco de dados relacional, simulando um sistema de livro-caixa financeiro. O projeto segue um fluxo próximo ao utilizado em ambientes corporativos, incluindo levantamento de requisitos, modelagem de banco de dados (conceitual, lógico e físico), definição de regras de negócio e planejamento de uma API REST em arquitetura monolítica.

Estrutura do documento:

- Levantamento de requisitos (definir o que o sistema tem que fazer);
- Modelagem do banco de dados:
    - Criar o modelo conceitual do banco de dados;
    - Criar o modelo lógico;
    - Criar o modelo físico.
- criar a lógica de negócio;
- expor essa lógica através de uma API (criação de endpoints -rotas- que vão receber requisições e responder com os dados ou ações);
- estrutura de monolito.

Requisitos do sistema (o que deve fazer):

Requisitos do Sistema (em ordem de implementação):

1. Autenticação e Autorização
2. Registro de Receitas
3. Registro de Despesas
4. Categorias e Tags
5. Cálculo do Saldo
6. Consulta de Histórico
7. Relatórios e Resumos
8. Dashboard Visual
9. Importação e Exportação de Dados
10. Notificações
11. Backup e Recuperação

Ordem Lógica dos Requisitos do Sistema

1. Autenticação e Autorização

    Implementar um sistema de login, para que cada usuário tenha seu próprio controle financeiro.

- Cadastro e login de usuários.
- Garantir que cada usuário veja apenas seus próprios dados.

2. Registro de Receitas

    Registro de entradas, com detalhes como data, descrição e valor.

- Permitir registrar entradas de dinheiro.
- Campos básicos: valor, data, descrição, categoria.

3. Registro de Despesas

    Registro das saídas, também com data, descrição e valor.

- Registro de saídas de dinheiro.
- Mesma lógica das receitas, mas com impacto negativo no saldo.

4. Categorias e Tags

    Permitir que o usuário categorize as transações, como alimentação, transporte, lazer, etc.

- Classificar receitas e despesas (ex.: alimentação, aluguel, lazer).
- Essencial para relatórios e análises futuras.

5. Cálculo do Saldo

    Resultado direto das operações anteriores.

- Cálculo automático: receitas − despesas.
- Pode ser feito dinamicamente ou armazenado.

6. Consulta de Histórico

    Permitir que o usuário visualize o histórico de transações, com filtros por data, tipo ou categoria.

- Listagem de transações.
- Filtros por data, tipo, categoria.

7. Relatórios e Resumos

Gerar relatórios ou resumos mensais, mostrando as entradas, saídas e o saldo final.

- Relatórios mensais/anuais.
- Total de receitas, despesas e saldo.

8. Dashboard Visual

Incluir gráficos e visualizações para que o usuário possa ver de forma clara o fluxo de caixa, tendências de despesas e receitas.

- Gráficos e indicadores.
- Consumo da API pelo frontend.

9. Importação e Exportação de Dados

Permitir que o usuário importe ou exporte os dados, como um arquivo CSV, para facilitar backups ou análises externas.

- Exportar dados (CSV, por exemplo).
- Importar lançamentos.

10. Notificações

Enviar alertas ou lembretes para o usuário, como notificações de saldo baixo ou quando uma despesa específica for registrada.

- Avisos de saldo baixo.
- Alertas após lançamentos.
- Pode ser simples (log ou e-mail mock).

11. Backup e Recuperação

Garantir que os dados do usuário sejam salvos de forma segura e que possam ser recuperados em caso de necessidade.

- Estratégia de backup do banco.
- Recuperação de dados.
- Alinhamento com boas práticas e DevOps.

Modelagem do banco de dados (entidades e relacionamentos):

Modelo conceitual:

1. definir entidades
    - Usuário (User)
    - Transação (Transaction)
    - Categoria (Category)
    - Tag
    - Notificação (Notification)
    - *Backup (pode ser conceitual, não física)*

2. Definir atributos

- Usuário
    - id_usuario (PK)
    - nome
    - email
    - senha
    - data_criacao

- Transação
    - id_transacao (PK)
    - tipo (entrada ou saída)
    - valor
    - descricao
    - data_transacao
    - id_usuario (FK)
    - id_categoria (FK)

- Categoria
    - id_categoria (PK)
    - nome
    - descricao
    - id_usuario (FK)

  Observação importante: categoria ligada ao usuário permite categorias personalizadas.

- Tag
    - id_tag (PK)
    - nome
    - id_usuario (FK)

- Notificação
    - id_notificacao (PK)
    - mensagem
    - data_envio
    - lida (boolean)
    - id_usuario (FK)

3. Definir relacionamentos

User - Transaction

- Um Usuário pode ter várias Transações
- Uma Transação pertence a um único Usuário
- Cardinalidade: 1 : N

User - Category

- Um Usuário pode criar várias Categorias
- Uma Categoria pertence a um Usuário
- Cardinalidade: 1 : N

User - Notification

- Um Usuário pode receber várias Notificações
- Cardinalidade: 1 : N

Category - Transaction

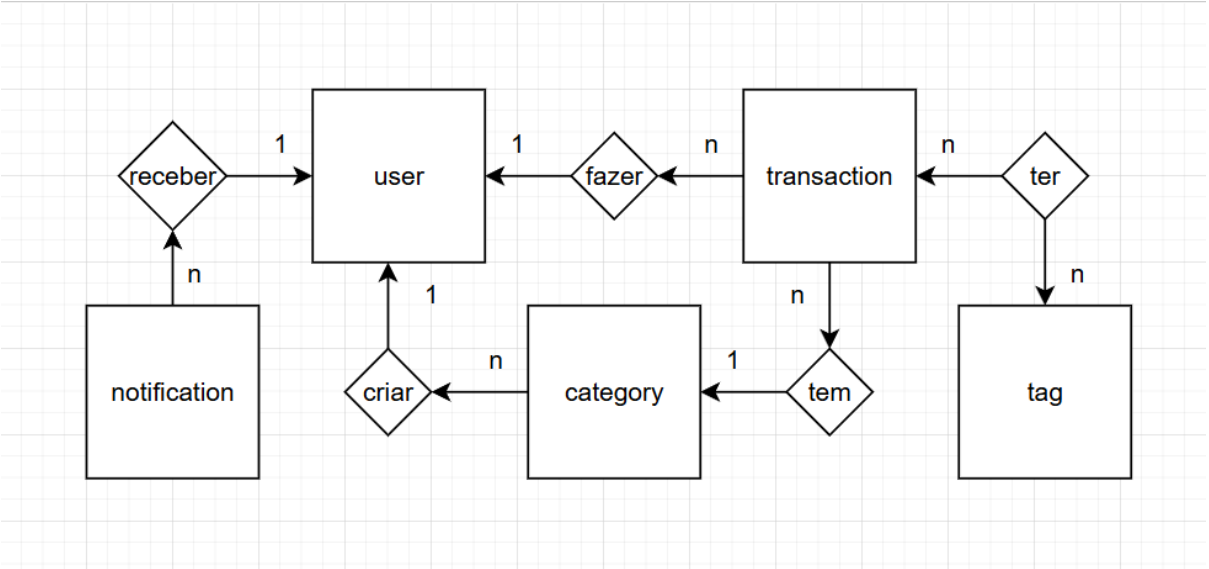
- Uma Categoria pode ter várias Transações
- Uma Transação pertence a uma Categoria
- Cardinalidade: 1 : N

Transaction - Tag

- Uma Transação pode ter várias Tags
- Uma Tag pode estar em várias Transações
- Cardinalidade: N : N

  Esse relacionamento N:N será resolvido depois com uma tabela associativa (no modelo lógico).

4. Construir modelo conceitual final (DER - diagrama de entidade relacionamento)



Modelo lógico:

Modelo físico:

Tecnologias utilizadas:

- java 21,
- Spring Boot
- MySQL.
- Git/GitHub

Status do projeto:

Em andamento.