Ideação: Um software que gerencie a partir de uma rede neural, a organização do dinheiro informado, bem com o que o dinheiro deverá ser repartido, e estratégias para economizar.

Requisitos Funcionais (RF)

RF1 - Gestão de Dados Financeiros:
O sistema deve permitir o gerenciamento dos dados financeiros do usuário (fontes de receita e despesas).

RF2 - Análise e Organização Financeira:
O sistema deve organizar as finanças do usuário e sugerir uma divisão percentual do dinheiro com base nas preferências do usuário.

RF3 - Recomendação de Estratégias de Economia:
O sistema deve gerar sugestões de estratégias para o usuário economizar dinheiro.

RF4 - Exportação de planilhas:
O sistema deve fornecer uma planilha excel com todas as informações cadastradas quando solicitado.

RF5 - Monitoramento de Desempenho Financeiro:
O sistema deve permitir o acompanhamento financeiro com base nas informações recebidas pelo usuário.

RF6 - Autenticação:
O sistema deve permitir a autenticação de usuários.

RF7 - Personalização de metas:
O sistema deve permitir que o usuário defina metas financeiras personalizadas (como poupar para uma viagem, quitar dívidas ou investir) e acompanhar o progresso em relação a essas metas.

RF8 - Histórico financeiro:
O sistema deve manter um histórico dos planejamentos financeiros realizados, permitindo ao usuário visualizar versões anteriores e comparar a evolução das estratégias.

RF9 - Simulação de cenários:
O sistema deve permitir que o usuário simule diferentes cenários financeiros (como aumento de receita ou corte de despesas) e visualizar os impactos dessas mudanças na organização do dinheiro.

RF10 - Classificação Inteligente de despesas:
O sistema deve utilizar a rede neural para classificar automaticamente as despesas informadas pelo usuário em categorias (como essenciais, supérfluas, recorrentes), facilitando a análise e o planejamento.

Requisitos Não Funcionais (RNF)

RNF1 - Usabilidade:
A interface do usuário deve ser intuitiva e fácil de usar.
RNF2 - Performance:
O sistema deve processar e analisar dados financeiros de forma eficiente, com mínima latência.
RNF3 - Escalabilidade:
O sistema deve ser escalável, permitindo adicionar novas funcionalidades sem impacto significativo.
RNF4 - Segurança:
O sistema deve garantir a segurança dos dados financeiros dos usuários (por meio de criptografia e autenticação robusta).
RNF5 - Confiabilidade:
O sistema deve ser confiável, com baixo índice de falhas, especialmente nas funções críticas.
RNF6 - Privacidade:
O sistema deve aderir às leis de proteção de dados e garantir controle do usuário sobre seus dados.
RNF7 - Compatibilidade:
O sistema deve ser compatível com diferentes navegadores.
RNF8 - Manutenibilidade:
O código do sistema deve ser bem estruturado, modular e documentado.

REGRAS DO NEGÓCIO:

RN - Versão gratuita deverá ter uma quantidade limitada de adição / alteração / exclusão de informações e funções, tendo um limite de 20 operações a cada 12 horas

📋 Etapas de Desenvolvimento

🧱 Etapa 1 — Estrutura e ambiente (configuração inicial) ✅ *(você vai começar aqui agora)*

- Criar o projeto Java no IntelliJ.
- Configurar dependências e banco SQLite.
- Testar o servidor local (porta 8080).
- Criar a estrutura de diretórios.
- Garantir que o ambiente está 100% funcionando.

⚙️ Etapa 2 — Backend (Spring Boot + SQL)

- Implementar CRUD para:
    - Usuário
    - Transações (receitas/despesas)
    - Metas financeiras
- Todas as informações são **manuais e textuais** (sem APIs externas).
- Adicionar regras de negócio (limite de 20 operações/12h).
- Testar endpoints com Postman.

🧠 Etapa 3 — Integração com Gemini (IA)

- Implementar um serviço que envia dados financeiros textuais ao Gemini.
- Retornar recomendações e estratégias de economia.
- Garantir comunicação segura via chave de API (armazenada em variável local).

🧩 Etapa 4 — Interface (Streamlit)

- Criar frontend em Python.
- Formulários para entrada de dados manuais (descrição, valor, categoria etc.).

- Dashboard para visualizar saldo, despesas, metas e respostas do Gemini.

💾 Etapa 5 — Exportação e histórico

- Exportar informações para planilha Excel (`.xlsx`).
- Salvar histórico local no banco SQLite.
- Permitir visualizar versões anteriores de planejamentos.

🔐 Etapa 6 — Segurança e privacidade

- Senhas criptografadas com BCrypt.
- Token local (sem login via rede).
- Função “Apagar meus dados” para manter ética de uso.
- Aviso de uso acadêmico e dados simulados.

☁️ Etapa 7 — Deploy e entrega

- Deploy gratuito do backend no Render.com.
- Deploy do frontend no Streamlit Cloud.
- Publicação no GitHub com README completo e documentação técnica.