README

1 - Contexto Geral do Problema

O objetivo é desenvolver uma API para gerenciar `Propostas` de crédito consignado para aposentados (`Proponentes`).

A aplicação deve permitir a inclusão de propostas validação de regras de negócio, simulação de valores e processamento das propostas em várias etapas.

Com isso tomei a liberdade de ressaltar alguns pontos chave:

- A aplicação desenvolvida possui apenas o backend (WebAPI).
- A mesma possui apenas endpoints voltados ao processamento, simulação ou validação de `Propostas` e suas demais etapas. Desta forma, não foram elaborados endpoints de "CRUDs" para entidades de apoio (`Loja`, `Agente`, etc.), apesar de considerar a existência dessas entidades no sistema.
- O processamento de propostas envolve várias etapas, desta forma a aplicação precisa ser resiliente (para, por exemplo tentar constantemente consumir/atualizar informações de outras API do SPC/Serasa)

2 - Requisitos funcionais

3 - Requisitos não funcionais

Visando os requisitos não funcionais, foiu concebido uma arquitetura simples, porém robusta que divide a API em 4 camadas:

API Layer (Controllers) | Application Layer (Handlers) | Domain Layer (Entities) | Infrastructure Layer (Repositories, External Services)

Desta forma, é possível segregar a aplicação em modulos menores e testa-los independentemente. Desta forma pensei em fazer dois tipos de testes:

Unidade (como validação de propostas, cálculo de simulação e clientes de APIs externas) Integração (via `WebApplicationFactory`, para testar todos os endpoints da API)

Cogitei realizar testes E2E com o fluxo completo da aplicação, desde a inclusão da proposta até o processamento, mas tive limitações de tempo.

Também foi levado em consideração o o versionamento e distribuição desta API. Com isso, a mesma está disponibilizada no github aqui.

4 ??

5 - Itens incluidos mas que estavam fora do escopo