Teste Prático de Engenharia Backend - Simulador de Crédito

Objetivo:

Avaliar as habilidades de desenvolvimento backend do candidato, incluindo conhecimentos em arquitetura de software, APIs RESTful, cálculos financeiros, otimização de desempenho, testes automatizados e documentação.

Instruções:

- O teste deve ser concluído em até 4 dias.
- Submeta o código em um repositório Git público ou privado (neste caso, forneça acesso ao avaliador).
- Inclua um arquivo README.md com instruções sobre como configurar e rodar o projeto.
- O código será revisado durante a entrevista técnica.

Descrição do Projeto:

Você deverá criar uma aplicação backend para um simulador de crédito que permita aos usuários simular empréstimos, visualizando as condições de pagamento baseadas no valor solicitado, taxa de juros e prazo de pagamento.

Requisitos:

1. Setup do Projeto

- Utilize uma linguagem de programação backend de sua preferência: Nodejs, kotlin, ruby, python ou a que se sentir mais à vontade.
- Caso queira, fique a vontade para escolher o framework web que mais se adeque com a linguagem.

2. Simulação de Crédito

- Crie endpoints para:
    - Simular um empréstimo com:
        - Valor do empréstimo
        - Data de nascimento do cliente
        - Prazo de pagamento em meses
- A simulação de empréstimo deve considerar:
    - Valor do empréstimo.
    - Taxa de juros anual.
    - Prazo de pagamento em meses.

    - Data de nascimento do cliente.
- A taxa de juros deve ser calculada com base na faixa etária do cliente:
    - Até 25 anos: 5% ao ano
    - De 26 a 40 anos: 3% ao ano
    - De 41 a 60 anos: 2% ao ano
    - Acima de 60 anos: 4% ao ano
- O resultado da simulação deve incluir:
    - Valor total a ser pago.
    - Valor das parcelas mensais.
    - Total de juros pagos.

3. Cálculos de Simulação

- Utilize a fórmula de cálculo de parcelas fixas:

$$PMT = \frac{PV \cdot r}{1 - (1 + r)^{-n}}$$

  Onde:

    - PMT = Pagamento mensal
    - PV = Valor presente (empréstimo)
    - r = Taxa de juros mensal (taxa anual / 12)
    - n = Número total de pagamentos (meses)

4. Testes Automatizados

- Escreva testes unitários e de integração para os principais componentes da aplicação.
- Inclua testes de desempenho para garantir que a aplicação lide bem com alta volumetria de cálculos.
- Utilize ferramentas de teste adequadas (Jest, PyTest, JUnit, etc.).

5. Documentação

- Inclua no README.md:
    - Instruções de setup.
    - Exemplos de requisições para os endpoints.
    - Explicação sobre a estrutura do projeto e decisões de arquitetura.

6. Boas Práticas e Qualidade de Código

- Siga boas práticas de codificação e arquitetura
- Utilize um linter para manter a consistência do código
- Inclua comentários e documentação no código quando necessário

Requisitos Adicionais para Desenvolvedores Seniores (Bônus):

1. Alta Volumetria

- Implemente um endpoint que aceite múltiplas simulações de crédito em uma única requisição (ex: 10.000 simulações).
- Considere utilizar técnicas de paralelismo e/ou processamento assíncrono para melhorar a performance.
- **Nota:** O candidato pode optar por lidar com a alta volumetria de forma assíncrona ou síncrona, conforme achar mais adequado.
- Abstraia no código a utilização de serviços de mensageria (como SQS, Kafka, RabbitMQ), sem a necessidade de implementação completa. Descreva como seriam utilizados em um cenário real.

2. Documentação

- Documente os endpoints da API utilizando uma ferramenta como Swagger ou API Blueprint

Critérios de Avaliação:

- **Funcionalidade**: A aplicação atende a todos os requisitos funcionais?
- **Qualidade do Código**: O código é limpo, bem organizado e segue boas práticas?
- **Testes**: Existem testes automatizados? Eles cobrem casos principais?
- **Documentação**: A documentação está clara e completa?
- **Desempenho**: A aplicação lida eficientemente com alta volumetria de cálculos?

Sugestão de Bônus:

- Implementar notificação por email com os resultados da simulação.
- Adicionar suporte para diferentes cenários de taxa de juros (fixa e variável).
- Criar um Dockerfile e docker-compose para facilitar o setup da aplicação.
- Adicionar suporte para diferentes moedas e conversão de taxas.