TESTE DE PRÁTICO LUMI – DESENVOLVEDOR(A) FULL STACK PLENO(A)

Olá, candidato(a) a Desenvolvedor Full Stack Pleno, estamos empolgados em ter você no processo de seleção da Lumi. Como parte crucial deste processo, apresentamos um desafio técnico que nos permitirá avaliar suas habilidades e competência em desenvolvimento Full Stack.

Nota Importante: O trabalho submetido para este teste não será utilizado para fins comerciais ou integrado em nossos produtos. O objetivo é puramente avaliativo.

DESCRIÇÃO DO DESAFIO

Nós iremos fornecer algumas faturas de energia elétrica. Seu objetivo será desenvolver um código que seja capaz de:

- Extrair os dados relevantes dessas faturas.
- Organizar esses dados de maneira estruturada em um banco de dados PostgreSQL.
- Apresentar esses dados em uma aplicação web, por meio de uma API.

Tecnologias a serem utilizadas: Typescript, Node.js (NestJS/Express) e React.

DETALHAMENTO DO DESAFIO

1. **Extração de Dados:** Você deve desenvolver um extrator de dados para capturar os dados das faturas de energia elétrica fornecidas (anexo FATURAS) e extrair as informações relevantes. Estas incluem, mas não estão limitadas a:

- "Nº DO CLIENTE", por exemplo:

- "Mês de referência", por exemplo:

- 'Energia Elétrica'– Quantidade (kWh) e Valor (R$),
- 'Energia SCEEE s/ICMS' – Quantidade (kWh) e Valor (R$),
- 'Energia Compensada GD I' – Quantidade (kWh) e Valor (R$), por exemplo:

			Valores Faturados	
Itens da Fatura	**Unid.**	**Quant.**	**Preço Unit**	**Valor (R$)**
Energia Elétrica	kWh	100	1,04841351	104,81
Energia SCEE s/ ICMS	kWh	1.860	0,58125187	1.081,12
Energia compensada GD I	kWh	1.860	0,56148931	-1.044,37

- 'Contrib Ilum Publica Municipal' – Valor (R$), por exemplo:

Contrib Ilum Publica Municipal	47,57

Observação: Pode ser útil utilizar uma biblioteca como `pdf-lib` ou `pdf-parse` no Node.js para a extração eficiente dos dados dos PDFs fornecidos.

1.1. **Variáveis de Interesse:** abaixo estão elencadas as variáveis de interesse, algumas delas calculadas:

- **Consumo de Energia Elétrica (KWh)**: corresponde ao somatório das variáveis 'Energia Elétrica kWh' **+** 'Energia SCEEE s/ICMS kWh', por exemplo:
  - Exemplo (Abril/24): Consumo de Energia Elétrica = 50 kWh + 476 kWh = 526 kWh;
- **Energia Compensada (kWh)**: corresponde à variável 'Energia Compensada GD I (kWh)';
- **Valor Total sem GD (R$)**: somatório dos valores faturados de 'Energia Elétrica (R$)' **+** 'Energia SCEE s/ ICMS (R$)' **+** 'Contrib Ilum Publica Municipal (R$)';
- **Economia GD (R$)**: corresponde à 'Energia compensada GD I (R$)'.

2. **Banco de Dados:** Implementar um banco de dados relacional (preferencialmente PostgreSQL) e utilizar ORMs (Prisma/TypeORM/Sequelize) para a interação com o mesmo.

3. **Aplicação:** Implementar uma aplicação web para acessar e visualizar os dados armazenados no banco de dados. O front-end será desenvolvido em React e o back-end em Node.js, utilizando NestJS ou Express para a estrutura da aplicação.

3.1. **Estrutura da Aplicação (FrontEnd):** O front end será organizado em duas páginas principais, sendo elas**:**

3.1.1. **Dashboard:** esta página tratá gráficos e cards representando os totais das variáveis, os gráficos de interesse são:

- Resultados de Energia(kWh) = Consumo de Energia Elétrica KWh vs Energía Compensada kWh;
- Resultados Financeiros (R$) = Valor Total sem GDR$ vs Economia GDR$.

3.1.2. **Biblioteca de Faturas:** Página dedicada à disponibilização das faturas ao usuário final, permitindo que o usuário selecione um determinado **Nº DO CLIENTE** e realize o download de sua fatura de energia elétrica em um mês específico. Ex:

3.2. **Detalhes adicionais da aplicação:**

3.2.1. **Filtros e Visualização**: A especificação exige filtros por número do cliente e período de análise. Garanta que o front-end com React tenha uma interface intuitiva para essa navegação.

3.2.2. **Gráficos**: Use bibliotecas como Chart.js ou Recharts para plotar gráficos claros e personalizáveis.

3.2.3. **Cards Resumo**: Crie cards no dashboard para exibir os totais acumulados por

variável (Ex: total de energia consumida, valor compensado, etc.).

4. **Testes Automatizados**

4.1. **Testes unitários podem incluir validação de:**

- Parsing correto dos PDFs.
- Inserção no banco e retorno pela API.
- Cálculo correto dos valores agregados.

5. **Envio e Hospedagem:** A hospedagem da aplicação é opcional, mas pode ser um diferencial positivo. Você pode usar plataformas gratuitas como Vercel e Render para essa finalidade.

O TESTE AVALIARÁ

Este teste tem como objetivo avaliar suas habilidades técnicas e práticas no desenvolvimento de uma aplicação web completa. Serão considerados os seguintes aspectos:

1. **Extração e Processamento de Dados:**
    a. Precisão e completude na extração dos dados das faturas.
    b. Eficiência e robustez da lógica de processamento e cálculo dos dados.
2. **Modelagem e Persistência de Dados:**
    a. Adequação do esquema do banco de dados (PostgreSQL) para armazenar os dados das faturas e seus relacionamentos.
    b. Eficiência e desempenho das queries para inserção e recuperação de dados.
    c. Utilização e eficácia do ORM (Prisma/TypeORM/Sequelize) na interação com o banco de dados.
3. **Desenvolvimento Backend (Node.js - NestJS/Express):**
    a. Estrutura e organização do código.
    b. Implementação correta da API RESTful para acesso aos dados.
    c. Tratamento de erros e exceções.
4. **Desenvolvimento Frontend (React):**
    a. Estrutura e organização do código.
    b. Design e usabilidade da interface, incluindo filtros, visualizações e interação com o usuário.
    c. Responsividade e acessibilidade da aplicação.
5. **Qualidade do Código:**
    a. Legibilidade, clareza e organização do código.
    b. Boas práticas de programação e princípios de design de software.
    c. Modularização e reutilização de código.

ENTREGA

- **Código**: Disponibilizar o código em um repositório público no GitHub.
- **Documentação**: Fornecer um README detalhado com instruções de configuração, instalação, execução e uso da aplicação.
- **Demonstração**: Opcionalmente, hóspede a aplicação em um servidor de sua escolha e nos envie o link para visualizarmos a aplicação web.

Envie o teste para os emails: gabriel@labs-lumi.com.br, eliane@labs-lumi.com.br, marcelo@labs-lumi.com.br, nicolas@labs-lumi.com.br, robert@labs-lumi.com.br contendo o código fonte em repositório Git (compartilhar o acesso com estes mesmos e-mails), juntamente com instruções no README do projeto sobre como configurar e executar a aplicação. Se possível, hospede a aplicação em um servidor de sua escolha e nos envie o link para visualizarmos a aplicação web, conforme explicitado no item **5. Envio e Hospedagem**.

Este desafio é projetado **exclusivamente** para avaliar sua experiência e competência com as tecnologias listadas, bem como suas habilidades de resolução de problemas, organização e atenção aos detalhes.

Data limite para envio do teste prático: 06/04/2025

Boa sorte! Estamos ansiosos para ver o que você irá criar!