Teste Técnico – Desenvolvimento Web

Olá, candidato(a)!

Primeiro gostaríamos de agradecer por seu interesse em fazer parte do time de desenvolvimento da Shopper.com.br.

Estamos construindo o melhor sistema de abastecimento do Brasil e para isso estamos procurando pessoas apaixonadas por usar a tecnologia para criar soluções inovadoras. Esperamos que seja você!

Explicando um pouco do processo seletivo:

Etapa 1 – Back-end

Nessa primeira fase, você irá construir o back-end de um serviço de leitura de imagens. Serão 3 três endpoints e uma integração com a API do Google Gemini.

Etapa 2 – Front-end

As melhores entregas do back-end serão convidadas a desenvolver um front-end simples e funcional para complementar o projeto. Selecionaremos as melhores para uma discussão mais aprofundada com o time técnico da Shopper.

Etapa 3 - Cultural

Finalizada a validação das suas habilidades técnicas, agora queremos te conhecer melhor como pessoa e profissional. Nesta próxima etapa, você terá a oportunidade de conversar com nosso time, compartilhar seus objetivos de carreira e entender melhor nossa cultura e forma de trabalho. Será um espaço aberto para trocar ideias e esclarecer qualquer dúvida que você possa ter sobre nós.

Além disso, preparamos um bate-papo descontraído para que possamos conhecer seus valores, suas motivações e como você se encaixa em nossa equipe. Queremos ter certeza de que essa oportunidade é ideal para você, tanto quanto você é para nós.

ETAPA 1 – Back-end

O que você precisará saber:

- Ler especificações técnicas em inglês e entender requisitos de negócios
- Desenvolver uma API REST em Node.js com Type Script
- Noção básica de modelagem de bancos de dados
- Criar uma imagem e subir um container utilizando Docker
- O básico do versionamento em um repositório usando Git

No que você será avaliado:

Sua aplicação será submetida a uma bateria de testes que irão verificar cada um dos critérios de aceite. Por isso é importante que você **leia atentamente e siga rigorosamente** todas as instruções. Sua aplicação deve cumprir **integralmente** os requisitos.

Pontos desejáveis, mas que não são eliminatórios:

- Uma arquitetura limpa (clean code)
- Testes unitários

Como entregar seu projeto:

- Preencha esse formulário (https://bit.ly/testedevshopper)
- A aplicação deve ser completamente dockerizada.
- Deve conter um arquivo docker-compose.yml na raiz do seu repositório.
  Nosso script de teste irá criar um arquivo.env na raiz do repositório no seguinte formato. Sua aplicação deve receber essa variável de ambiente para a execução.

  GEMINI_API_KEY=<chave da API>

  ATENÇÃO: NÃO compartilhe sua chave pessoal conosco.

- O *docker-compose* deve ser capaz de subir a aplicação e todos os serviços necessários com um único comando.

Como você deve usar LLMs (ChatGPT, Gemini, Llama, etc..),

Gostamos e incentivamos quem busca a inovação para se tornar mais produtivo, porém queremos avaliar você! Utilize a LLM como ferramenta e não como a criadora do seu código.

Você **NÃO** deve fazer:

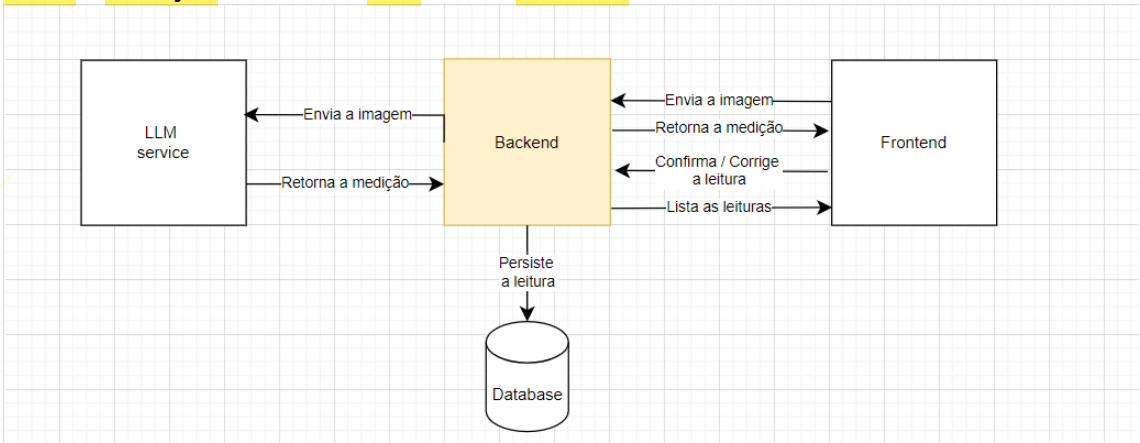
- Copiar esse teste, colar no GPT e apenas copiar o resultado. LLMs geram códigos ruins.

Você pode fazer:

- Usar o GPT para melhorar o código que você criou ou estudar melhores práticas

CENÁRIO

Vamos desenvolver o back-end de um serviço que gerencia a leitura individualizada de consumo de água e gás. Para facilitar a coleta da informação, o serviço utilizará IA para obter a medição através da foto de um medidor.



O back-end deverá conter os seguintes endpoints:

POST /upload
Responsável por receber uma imagem em base 64, consultar o Gemini e retornar a medida lida pela API

Esse endpoint deve:

- Validar o tipo de dados dos parâmetros enviados (inclusive o base64)
- Verificar se já existe uma leitura no mês naquele tipo de leitura.
- Integrar com uma API de LLM para extrair o valor da imagem

Ela irá retornar:

- Um link temporário para a imagem
- Um GUID
- O valor numérico reconhecido pela LLM

Request Body

```
{
  "image": "base64",
  "customer_code": "string",
  "measure_datetime": "datetime",
  "measure_type": "WATER" ou "GAS"
}
```

Response Body:

Status Code	Descrição	Resposta
200	Operação realizada com sucesso	{ "image_url": string, "measure_value":integer, "measure_uuid": string }
400	Os dados fornecidos no corpo da requisição são inválidos	{ "error_code": "INVALID_DATA", "error_description": <descrição do erro> }
409	Já existe uma leitura para este tipo no mês atual	{ "error_code": "DOUBLE_REPORT", "error_description": "Leitura do mês já realizada" }`

Documentação técnica do Google Gemini (LLM):

https://ai.google.dev/gemini-api/docs/api-key
https://ai.google.dev/gemini-api/docs/vision

ATENÇÃO: Você precisará obter uma chave de acesso para usar a funcionalidade. Ela é gratuita. Não realize despesas financeiras para realizar esse teste.

PATCH /confirm
Responsável por confirmar ou corrigir o valor lido pelo LLM,

Esse endpoint deve:

- Validar o tipo de dados dos parâmetros enviados
- Verificar se o código de leitura informado existe
- Verificar se o código de leitura já foi confirmado
- Salvar no banco de dados o novo valor informado

Ele **NÃO** deve fazer:

- Fazer novas consultas ao LLM para validar o novo resultado recebido

Ela irá retornar:

- Resposta de OK ou ERRO dependendo do valor informado.

Request Body

```
{
  "measure_uuid": "string",
  "confirmed_value": integer
}
```

Response Body**:**

Status Code	**Descrição**	**Resposta**

200	Operação realizada com sucesso	{
		“success”: true
		}

400	Os dados fornecidos no corpo da requisição são inválidos	{
		"error_code": "INVALID_DATA",
		"error_description": <descrição do erro>
		}

404	Leitura não encontrada	{ "error_code": "MEASURE_NOT_FOUND", "error_description": "Leitura do mês já realizada" }`
409	Leitura já confirmada	{ "error_code": "CONFIRMATION_DUPLICATE", "error_description": "Leitura do mês já realizada" }`

GET /<customer code>/list
Responsável por listar as medidas realizadas por um determinado cliente

Esse endpoint deve:

- Receber o código do cliente e filtrar as medidas realizadas por ele
- Ele opcionalmente pode receber um query parameter “measure_type”, que deve ser “WATER” ou “GAS”
    - A validação deve ser CASE INSENSITIVE
    - Se o parâmetro for informado, filtrar apenas os valores do tipo especificado. Senão, retornar todos os tipos.

Ex. {base url}/<customer code>/list?measure_type=WATER

Ela irá retornar:

- Uma lista com todas as leituras realizadas.

Response Body**:**

Status Code	**Descrição**	**Resposta**

200	Operação realizada com sucesso	{
		“customer_code”: string,
		“measures”: [
		{
		“measure_uuid”: string,
		“measure_datetime”: datetime,
		“measure_type”: string,
		“has_confirmed”:boolean,
		“image_url”: string
		},
		{
		“measure_uuid”: string,
		“measure_datetime”: datetime,
		“measure_type”: string,
		“has_confirmed”:boolean,
		“image_url”: string
		}
		]
		}

400	Parâmetro measure type diferente de WATER ou GAS	{
		"error_code": "INVALID_TYPE",
		"error_description": “Tipo de medição não permitida”
		}

404	Nenhum registro encontrado	{ "error_code": "MEASURES_NOT_FOUND", "error_description": "Nenhuma leitura encontrada" }`