Desafio Técnico – Vaga AI Engineer Júnior

🎯 Objetivo

Queremos entender como você pensa e estrutura soluções envolvendo **Modelos de Linguagem (LLMs)** e funcionalidades básicas de integração com ferramentas externas.

Você não precisa construir nada super complexo. Estamos avaliando **sua lógica, clareza e iniciativa**. Pode usar o que você souber — e se não souber, pesquise e tente aprender no caminho. Estamos mais interessados na forma como você resolve o problema do que em perfeição.

O Caso

Imagine que você está construindo um pequeno **assistente de IA** que responde perguntas dos usuários. Mas queremos que ele tenha um **"superpoder": saber quando deve responder sozinho e quando deve acionar uma ferramenta externa**.

Essa ferramenta será uma **calculadora**. Ou seja:

- Se o usuário perguntar algo como "Quem foi Albert Einstein?", o modelo deve responder com base em seu conhecimento.
- Mas se perguntar "Quanto é 128 vezes 46?", ele deve usar a calculadora para obter a resposta exata antes de responder.

🛠️ O que você precisa fazer

1. **Criar um assistente simples de IA** que:
    - Receba uma pergunta do usuário;
    - Identifique se é uma pergunta matemática ou não;
    - Caso seja, utilize uma função/calculadora para fazer a conta;
    - Caso não seja, responda diretamente com o modelo.
2. Você pode fazer isso de duas formas (escolha a que preferir):

    - **Em Python**: usando LangChain, OpenAI API, Hugging Face Transformers, ou mesmo lógica própria.
    - **Em JavaScript/TypeScript**: usando o Vercel AI SDK, Hugging Face Inference API, ou outra abordagem.
3. **Você pode usar qualquer LLM** com que se sinta confortável:
    - Modelos da **OpenAI (GPT-3.5 ou GPT-4)**,
    - Modelos **gratuitos** via Hugging Face (ex: Mistral, Zephyr, Cohere, etc.),
    - Modelos locais como **llama.cpp**, **Ollama**, etc.
    - Use o que for mais acessível para você — o importante é integrar um modelo de linguagem funcional no fluxo.

✅ O que avaliamos

- Clareza na lógica de decisão (quando usar a ferramenta ou não);
- Organização do código e explicações no README;
- Capacidade de integrar um LLM (qualquer um!);
- Capacidade de implementar uma ferramenta externa (a calculadora pode ser local mesmo!);
- Criatividade/opcional: se quiser, adicione outro “superpoder” ao assistente (ex: responder previsão do tempo, consultar uma API pública, etc.).

📦 Entrega

Crie um repositório (público ou privado) com (NAO COMPARTILHE SUA CHAVE DE API, deixa ela no .env e nos der as instruções no readme.md do modelo que voce usou e como configurar:

- O código funcional (mínimo viável);
- Um arquivo README.md explicando:

- Como executar o código;
- Qual foi sua lógica de implementação;
- O que você aprendeu e o que faria diferente com mais tempo.

💡 Dicas

- Se não tiver chave de API paga, use modelos **gratuitos** via Hugging Face Inference API ou rode modelos localmente com **Ollama**.
- No Python, frameworks como **LangChain** têm exemplos prontos de agentes com ferramentas https://python.langchain.com/docs/integrations/tools/
- No JS, o **Vercel AI SDK** facilita o streaming de respostas se quiser fazer algo interativo com frontend.