Desafio Técnico – Vaga de Desenvolvedor Python | C2S

Objetivo:

Avaliar sua capacidade de aprender coisas novas, com foco em tecnologias que talvez você ainda não tenha usado, e também seu domínio de Python.

Antes de começar, leia com atenção:

- A gente valoriza soluções bem pensadas, com base teórica forte, uso de boas libs e código que faça sentido.
- Provavelmente você vai precisar explorar algum LLM e pesquisar caminhos que ainda não conhece. Faz parte do desafio.
- A entrega precisa incluir um vídeo mostrando a demo funcionando.
- Usar IA para te apoiar é ok, mas se o código inteiro for gerado por ela, não dá pra entender o que de fato é seu. Eu também uso IA no dia a dia, mas não procuramos avaliar no teste o código feito por uma IA… GPT, Claude ou similares. (Os avaliadores trabalham a muito tempo com programação, dá para saber com facilidade o que é IA e o que é pessoa fazendo 😅)
- Coloque no código/estrutura do projeto coisas que mostrem a sua bagagem como sênior
- Esperamos alguma forma de testes automatizados das respostas de seu agente. Não esperamos uma abordagem de cobertura de testes, mas é crucial conseguir *algum* grau de “testabilidade”, mesmo que seja impreciso.

O que você vai fazer:

1. Modelagem de Dados

Crie um esquema para representar automóveis, com pelo menos 10 atributos relevantes. Pode incluir:

Marca, modelo, ano, motorização, tipo de combustível, cor, quilometragem, número de portas, transmissão… o que fizer sentido.

2. Popular o Banco com Dados Fictícios

Desenvolva um script que insira no mínimo 100 veículos no banco, com dados simulados (fakes).

Pode usar libs como Faker, SQLAlchemy, Pandas — ou outras que preferir pra facilitar o processo.

*Obs: se você só pedir para uma IA gerar isso, todos os desafios vão ficar muito parecidos, dificultando a avaliação para saber se você manda bem ou não 😄

3. Comunicação Cliente-Servidor via Protocolo MCP

Implemente a comunicação entre cliente e servidor usando o protocolo MCP (Model Context Protocol):

- O cliente envia filtros como marca, ano ou tipo de combustível.
- O servidor interpreta, consulta o banco e devolve os resultados.
- Nada de consultas diretas, o fluxo precisa respeitar o modelo Client > Server > Banco.

4. Aplicação no Terminal com Agente Virtual

Monte uma aplicação que roda direto no terminal, com interação via agente virtual:

- Ao iniciar, o agente conversa com o usuário (e não, não vale um “menuzão” de IF e ELSE).
- Ele entende o que o usuário está buscando, faz perguntas (marca, modelo, ano, combustível, faixa de preço…)
- As perguntas não precisam seguir um padrão engessado tipo formulário. Pode ser algo mais solto.
- Depois de coletar os dados, o cliente MCP envia pro servidor, que busca no banco e retorna os veículos compatíveis.
- O agente exibe uma resposta amigável com os resultados, incluindo: marca, modelo, ano, cor, quilometragem e preço.

Recomendações finais:

- Use bibliotecas que te ajudem de verdade, sem complicar o código.(Pesquise, aprenda, aplique)
- Organização, clareza e legibilidade contam muito.
- Envie um repositório com instruções claras pra rodar o projeto.
- Grave um vídeo com a aplicação funcionando, mostrando alguns casos diferentes.
- E se divirta no processo. Mesmo que você não siga com a gente, esse desafio pode te ensinar coisas valiosas.

Entrega

Prazo: 5 dias corridos a partir do envio

Forma: Enviar o link do repositório (GitHub ou similar) + vídeo da demo por e-mail.