Desafio Técnico – Engenheiro de IA Sênior

🎯 Contexto do Problema (Realista)

Uma empresa possui **milhares de documentos internos** (políticas, processos, manuais, incidentes e FAQs) e quer criar um **Assistente de IA Corporativo** para apoiar **times técnicos e de negócio**.

Problemas atuais:

- Documentos desatualizados
- Informações conflitantes
- Perguntas ambíguas
- Alto risco de resposta errada
- Custo alto de uso de IA

Responder errado pode gerar prejuízo financeiro.

🧩 O Desafio

Criar um **serviço de IA** que:

- Responda perguntas em linguagem natural
- Busque informações em documentos
- Identifique **quando NÃO sabe responder**
- Explique **de onde veio a resposta**
- Seja resiliente a erros, custo e abuso

Escopo Técnico

API Backend

Criar uma API com os seguintes endpoints:

🔷 `POST /ask`

Entrada:

```
{
  "question": "Qual é o prazo para reembolso de despesas
internacionais?"
}
```

Saída esperada:

```
{
  "answer": "O prazo é de até 100 dias, conforme política X.",
  "confidence": 0.82,
  "sources": [
    {
      "document": "politica_reembolso_v3",
      "excerpt": "O prazo para reembolso é de até 100 dias..."
    }
  ]
}
```

Dados (Parte crítica)

- Utilizar base enviada via e-mail: com +**10 documento**s
- Alguns documentos podem estar:
  - Estar **desatualizados**
  - Ter **informações conflitantes**
  - Ser **irrelevantes**
- O sistema deve **priorizar documentos mais confiáveis**

Comportamento Inteligente (Obrigatório)

A IA deve:

- **Recusar responder** se não houver informação suficiente
- Explicar que não encontrou base confiável
- Nunca “inventar”

Exemplo esperado:

```
{
  "answer": "Não encontrei informações confiáveis para responder essa
pergunta.",
  "confidence": 0.2
}
```

Controle de Qualidade

Implementar ao menos **2 mecanismos**, por exemplo:

- Score mínimo de confiança
- Validação cruzada de múltiplos documentos
- Prompt com regras rígidas
- Resposta estruturada

Custo, Performance e Resiliência

Demonstrar no **código e README**:

- Onde e como aplicar **cache**
- Como reduzir chamadas desnecessárias ao modelo
- Estratégias para **controle de custo**
- Tratamento de falhas (timeout, erro do modelo, falta de contexto)
- Estratégia de **monitoramento**

Segurança (Obrigatório)

O sistema deve:

- Bloquear perguntas sobre dados sensíveis
- Prevenir prompt injection básico
- Validar entrada do usuário

README.md

Deve explicar claramente:

1. Arquitetura
2. Como a IA decide responder ou não
3. Como trata documentos conflitantes
4. Trade-offs feitos
5. Limitações da solução

Tecnologias

✓ Recomendadas

- Node.js **ou** Python
- Uso de LLM (OpenAI, Azure, Anthropic, etc)
- Embeddings / busca semântica (se aplicável)

❌ Não recomendadas

- Workflows visuais (ex: ferramentas low-code de IA)

Entregáveis

O candidato deve entregar:

- **Link do repositório Git** (público ou privado) com o código da API
- **API publicada em cloud** (AWS, GCP, Azure, Vercel, etc), com endpoint `/ask` acessível
- **README.md completo**, explicando arquitetura, decisões técnicas, controles de qualidade, custo, segurança e limitações
- **Documentação de uso da API**, contendo:
  - Endpoint(s) e payloads
  - Exemplos de request/response
  - **Ao menos um exemplo de chamada via cURL**

Prazo

- **1 semana** para entrega do desafio
- O prazo começa a contar **a partir do envio do teste ao candidato**