Fabio é gerente de Talent Acquisition na TechMatch, uma startup que automatiza processos de contratação e busca otimizar o tempo de sua equipe rumo à contratação ágil de talentos.
Todos os dias, ele recebe diversos currículos em PDF ou imagens escaneadas e luta para analisá-los manualmente em busca dos candidatos que mais se adequam para cada vaga, desperdiçando horas em tarefas repetitivas.
Ele gastava horas e horas, extraindo, sintetizando e comparando informações, fazendo com que ele se atrasasse em outros problemas como decisões de contratação e serviços de gerência geral do seu cargo.

Para agilizar esse processo, Fabio sonha com uma ferramenta inteligente que fosse capaz de receber múltiplos documentos e forneça sumários claros, além de respostas precisas a perguntas de recrutamento — uma verdadeira ponte entre OCR e LLM.
A aplicação ideal para Fabio, deveria permitir que ele pergunte coisas como “Qual desses currículos se enquadra melhor para a vaga de Engenheiro de Software com esses requisitos: {...} ?”, e ter como resposta os currículos que mais se enquadram com a query passada acompanhados de uma justificativa para tal ou até mesmo um resumo de cada um dos currículos.

Além disso, para **auditoria interna** e rastreamento de uso, Fabio precisa registrar cada execução da ferramenta — quem a solicitou, qual query foi enviada, o resultado retornado e quando ocorreu — mas sem sobrecarregar o sistema com o armazenamento dos próprios documentos.

Seu sonho é uma aplicação inteligente que:

1. **Receba** múltiplos PDFs ou imagens (JPEG/PNG).
2. **Extraia** texto via OCR e gere sumários claros de cada currículo.
3. **Responda** a perguntas do tipo “Qual desses currículos se enquadra melhor para a vaga de Engenheiro de Software com requisitos {...}?” com justificativas baseadas no conteúdo.
4. **Registre** em um banco **não relacional** um log contendo request_id, user_id, timestamp, query e resultado — sem armazenar o documento completo — para que Fabio possa auditar e analisar o uso da ferramenta

Com isso, Fabio poderá dedicar seu tempo a entrevistas e estratégia, em vez de tarefas repetitivas de extração manual.

Requisitos Técnicos da Solução

- **Linguagem:** Deve ser implementada em **Python** aproveitando bibliotecas open-source de OCR (Tesseract, EasyOCR, PaddleOCR) e LLMs (Hugging Face, LocalAI).
- **API**:
    - Expor endpoint via **Swagger/OpenAPI** seguindo boas práticas de design (paths claros, exemplos de payload e responses).
    - Aceitar no body:
        - Lista de arquivos (PDF, JPG/PNG)
        - Campo query (string)
        - Campo request_id (UUID ou similar)
        - Campo user_id (identificador do solicitante)
    - Se query não for informado, retornar um JSON com **sumário individual** de cada currículo.
- **Persistência**:
    - Banco **não relacional** (ex.: MongoDB, DynamoDB, Cassandra).
    - Armazenar **logs de uso**:
        - request_id
        - user_id
        - timestamp
        - query
        - resultado
    - **Não** salvar o conteúdo completo dos arquivos para evitar custos de armazenamento desnecessários.
- **Infraestrutura**:
    - Empacotar tudo em **Docker.**
- **Documentação**:
    - Swagger interativo com exemplos de uso e **exemplos de responses** para casos de query e de sumário automático.
    - README com instruções de build e execução.

Importante: facilite a vida do Fabio! Sua solução deve ser intuitiva, confiável e bem documentada, proporcionando a maior produtividade possível para ele!