Roadmap de Desenvolvimento: App de Filtragem e Ranking de Currículos

Este roadmap está estruturado em fases, do conceito à distribuição. Cada fase tem um objetivo claro, tarefas específicas e um critério de sucesso para que você saiba quando pode avançar.

Fase 0: Fundação e Preparação (Tempo Estimado: 1-2 dias)

O objetivo aqui é preparar seu ambiente de desenvolvimento. Não pule esta fase.

Objetivo: Ter todas as ferramentas instaladas e configuradas.

Tarefas:

1. **Ambiente Java:** Instale um JDK (Java Development Kit) versão 17 ou superior. Escolha e configure uma IDE (IntelliJ IDEA Community ou Eclipse são excelentes).
2. **Ambiente Python:** Instale o Python (versão 3.9+). Configure uma IDE (VS Code com a extensão Python é perfeito para este projeto).
3. **Ambiente de IA Local:** Instale o Ollama. É a maneira mais simples de gerenciar e servir os LLMs localmente.

Download dos Modelos: Pelo terminal, baixe os modelos que você irá testar:
ollama pull phi3:mini
ollama pull qwen2:0.5b

4.
5. **Controle de Versão:** Inicie um repositório Git. `git init`. Faça o commit inicial. Use Git desde o primeiro dia.

Critério de Sucesso: Você consegue executar um programa "Hello World" em Java e um script "Hello World" em Python. O comando `ollama list` no terminal mostra os modelos baixados.

Fase 1: Prova de Conceito (PoC) do Backend de IA (Tempo Estimado: 1 semana)

Foco total na parte mais crítica e incerta: a extração de dados com o LLM. Esqueça a interface Java por enquanto.

Objetivo: Criar um script Python que recebe o texto de um currículo e retorna um JSON estruturado com as competências e níveis avaliados.

Tarefas:

1. **Crie o "Prompt Mestre":** Em um editor de texto, escreva e refine o prompt que instrui o LLM a fazer a extração, seguindo o formato JSON que definimos (com competência, nível e justificativa).
2. **Crie o Servidor de API:** Usando Flask ou FastAPI em Python, crie um servidor web mínimo que:
    - Tenha um único endpoint (ex: `/analisar`).
    - Receba um texto via requisição POST.
    - Insira esse texto no seu "Prompt Mestre".
    - Envie o prompt para a API do Ollama (que roda em `http://localhost:11434`).
    - Receba a resposta do Ollama e a retorne como resposta da sua API.
3. **Teste com Ferramentas de API:** Use o Postman, Insomnia ou uma ferramenta similar para enviar textos de currículos para a sua API Python e verificar se o JSON retornado está correto e consistente. Teste com 3 a 5 currículos diferentes.

Critério de Sucesso: Você tem um servidor Python rodando que, ao receber um texto, retorna um JSON válido e com a avaliação de competências, conforme esperado.

Fase 2: Desenvolvimento do Esqueleto da GUI e Integração (Tempo Estimado: 1 semana)

Agora vamos construir a interface e fazê-la "conversar" com o backend.

Objetivo: Ter uma interface Java básica que possa selecionar um arquivo, extrair o texto, enviar para o backend Python e exibir o resultado bruto.

Tarefas:

1. **Setup do Projeto Java:** Crie um novo projeto JavaFX (recomendado) ou Swing.
2. **Desenho da Interface Mínima:** Crie uma janela com:
    - Um botão "Selecionar Currículo...".
    - Uma área de texto (TextArea) para exibir o resultado.
    - Um botão "Analisar".
3. **Extração de Texto:** Adicione uma biblioteca como o Apache Tika ou PDFBox ao seu projeto Java para extrair o texto de arquivos PDF e DOCX.
4. **Comunicação HTTP:** Implemente a lógica do botão "Analisar":
    - Ao clicar, ele pega o texto extraído do arquivo.
    - Usa o cliente HTTP nativo do Java para fazer uma chamada POST para a sua API Python local (`http://localhost:5000/analisar`).
    - Recebe a resposta JSON e a exibe na área de texto.

Critério de Sucesso: Você consegue abrir seu app Java, selecionar um arquivo `.pdf`, clicar em "Analisar" e ver o JSON bruto retornado pelo Python aparecer na tela.

Fase 3: Construção do MVP - Minimum Viable Product (Tempo Estimado: 2 semanas)

É hora de juntar tudo em uma ferramenta funcional de ponta a ponta.

Objetivo: Criar uma versão 100% funcional, mesmo que com design simples, que cumpra o fluxo principal.

Tarefas:

1. **Parse do JSON em Java:** Crie classes Java (POJOs) que espelhem a estrutura do seu JSON. Use uma biblioteca como Gson ou Jackson para converter a string JSON em objetos Java.
2. **Interface de Requisitos:** Adicione campos na GUI para que o usuário possa digitar as competências desejadas e definir um peso/importância para cada uma.
3. **Lógica de Ranking:** Implemente o algoritmo de pontuação em Java. Ele deve:
    - Mapear os níveis ("Iniciante", "Avançado") para pontos.
    - Calcular a pontuação final de cada currículo com base nos pesos definidos pelo usuário.
4. **Exibição dos Resultados:** Substitua a área de texto bruta por uma tabela (TableView) que mostre os currículos analisados, ordenados pela pontuação final. Exiba o nome do arquivo, a pontuação e talvez as principais competências encontradas.
5. **Processamento em Lote:** Permita que o usuário selecione uma pasta inteira de currículos para serem analisados de uma vez. Mostre uma barra de progresso.

Critério de Sucesso: Um usuário pode abrir o app, definir os requisitos de uma vaga, selecionar uma pasta de currículos e obter uma lista ranqueada dos melhores candidatos para aquela vaga.

Fase 4: Empacotamento e Distribuição (Tempo Estimado: 1 semana)

Transforme seu projeto de desenvolvimento em um aplicativo real que qualquer pessoa possa instalar.

Objetivo: Criar instaladores `.exe` (Windows) e `.dmg` (macOS).

Tarefas:

1. **Empacotar o Backend Python:** Use o PyInstaller para converter seu servidor Flask/FastAPI em um único executável (`.exe` / executável Unix).
2. **Empacotar o Frontend Java:** Use o jpackage para criar uma imagem de aplicação autocontida, que inclui o Java Runtime.
3. **Criar o Instalador Final:**

    - **Windows:** Use o Inno Setup para criar um script que instala a aplicação Java e o backend Python, e cria um atalho que executa ambos.
    - **macOS:** Modifique o pacote `.app` gerado pelo jpackage para incluir o executável Python dentro dele e ajuste o código Java para encontrá-lo. Depois, crie o `.dmg`.
4. **Teste de Instalação:** Teste o instalador em uma máquina limpa (uma máquina virtual é ideal) para garantir que ele funciona sem a necessidade de instalar Java ou Python manualmente.

Critério de Sucesso: Você tem um arquivo `.exe` e um `.dmg` que instalam e executam a aplicação com sucesso em máquinas que não possuem o ambiente de desenvolvimento.

Fase 5: Refinamento, Testes e Feedback (Contínuo)

Objetivo: Melhorar a usabilidade, corrigir bugs e adicionar funcionalidades com base no uso real.

Tarefas:

1. **UI/UX:** Adicione mensagens de erro claras (ex: "Ollama não está em execução!"), tooltips de ajuda, e melhore o layout.
2. **Feedback:** Entregue a aplicação para alguns usuários-alvo (colegas, recrutadores) e colete feedback. O que é confuso? O que está faltando?
3. **Iteração:** Use o feedback para planejar as próximas melhorias.

Critério de Sucesso: A aplicação é estável, intuitiva e realmente útil para o propósito a que se destina.