**Documentação da Fase 1: Configuração do Ambiente Essencial**

**Visão Geral da Fase 1**

A Fase 1 do projeto focou em estabelecer a infraestrutura fundamental necessária para rodar o Ollama (o Large Language Model local) e o OpenWebUI (a interface de usuário), além de preparar o ambiente de desenvolvimento Python.

**As principais etapas executadas foram:**

1. **Configuração do Ambiente Python:** Instalação e virtualização do Python para gerenciamento isolado de dependências.
2. **Instalação do Ollama:** Instalação do servidor Ollama e download de modelos de LLM (ex: Mistral) e de Embedding (ex: nomic-embed-text).
3. **Configuração do OpenWebUI:** Implantação da interface gráfica para interação com o Ollama, utilizando Docker.

**Desafios Enfrentados e Soluções**

Durante a configuração da Fase 1, diversos desafios foram encontrados, principalmente relacionados à compatibilidade, orquestração de serviços e uso de ferramentas.

**Desafio 1: Virtualização do Ambiente Python**

* **Sintoma:** Dúvidas sobre como criar um ambiente Python isolado para o projeto.
* **Causa Raiz:** Falta de familiaridade com as práticas recomendadas de gerenciamento de ambientes Python.
* **Solução Implementada:** Orientação para o uso do módulo venv do Python para criar e ativar um ambiente virtual (python -m venv venv e .\venv\Scripts\activate ou source venv/bin/activate).

**Desafio 2: Conflito de Nomes de Módulo Python**

* **Sintoma:** Erro module 'ollama' has no attribute 'chat' ao tentar executar um script Python de teste (meu\_ollama.py).
* **Causa Raiz:** O script de teste do usuário foi nomeado ollama.py, o que gerou um conflito com o nome da biblioteca oficial ollama instalada via pip. O Python priorizou a importação do arquivo local.
* **Solução Implementada:** Renomear o arquivo do script do usuário para um nome distinto (ex: meu\_ollama.py) para evitar a colisão de nomes.

**Desafio 3: Instalação do OpenWebUI via pip Falhou**

* **Sintoma:** Tentativa de instalar open-webui via pip install open-webui resultou em erros de ERROR: No matching distribution found for open-webui devido a requisitos de versão do Python incompatíveis.
* **Causa Raiz:** O OpenWebUI é uma aplicação web que não é destinada à instalação direta via pip como uma biblioteca Python para a interface. Ele é projetado para ser executado em um ambiente isolado, como um contêiner Docker.
* **Solução Implementada:** Reorientação para a instalação do **Docker Desktop** e subsequente execução do OpenWebUI através de um comando docker run, que é a forma recomendada e mais robusta.

**Desafio 4: Falhas na Execução Inicial do Docker para OpenWebUI**

* **Sintoma A (Imagem Não Encontrada/Acesso Negado):** O comando docker run open-webui falhava com Unable to find image 'open-webui:latest' locally ou pull access denied.
  + **Causa Raiz:** O nome da imagem fornecido estava incompleto. O Docker precisa do caminho completo para o registro da imagem.
  + **Solução:** Fornecer o caminho completo e correto da imagem Docker do OpenWebUI (ghcr.io/open-webui/open-webui:main) no comando docker run.
* **Sintoma B (Nome de Contêiner em Uso):** Erro Conflict. The container name "/open-webui" is already in use by container....
  + **Causa Raiz:** Uma instância anterior (possivelmente falha ou não removida) do contêiner OpenWebUI já existia com o mesmo nome.
  + **Solução:** Instruções para parar (docker stop open-webui) e remover (docker rm open-webui) o contêiner existente antes de tentar iniciar um novo.

**Desafio 5: OpenWebUI Não Carrega no Navegador Após Iniciar o Contêiner**

* **Sintoma:** O contêiner Docker do OpenWebUI mostrava-se como Up (healthy) via docker ps -a, mas a interface não carregava em http://localhost:8080/.
* **Causa Raiz:** Embora o ConnectionRefusedError inicial (causado pelo Ollama não estar ativo) tenha sido superado, havia um problema com a configuração de rede Docker. O uso de --network=host pode ser inconsistente e potencialmente causar conflitos de porta na máquina host.
* **Solução Implementada:** Recomendar a configuração mais robusta para o docker run, utilizando o mapeamento de portas explícito (-p 8080:8080) e o parâmetro --add-host=host.docker.internal:host-gateway. Isso garante que o contêiner tenha uma rede isolada, mas ainda possa se comunicar com o Ollama na máquina host de forma confiável.

A superação desses desafios estabeleceu uma base funcional e sólida para as fases subsequentes do seu projeto, demonstrando a capacidade de integrar e depurar múltiplos serviços e tecnologias.