O desenvolvimento do Assistente Acadêmico UFBA foi repleto de desafios técnicos que exigiram soluções diretas na arquitetura e no ambiente de execução do software. Logo no início, o projeto enfrentou um problema de compatibilidade com o Windows. Algumas partes do LangChain usam o módulo `pwd`, que não existe nesse sistema, sendo exclusivo do Unix/Linux. Isso impedia o projeto de iniciar corretamente, gerando erros frequentes. Para resolver isso sem prejudicar a portabilidade, foi preciso reconfigurar o ambiente virtual e ajustar as bibliotecas.

Inicialmente, criamos uma simulação do módulo ausente, permitindo que o Python continuasse carregando as dependências. Depois, reconstruímos o projeto com bibliotecas mais estáveis, priorizando a compatibilidade e reduzindo dependências problemáticas. Isso mostrou a importância de escolher bem as bibliotecas em projetos de Inteligência Artificial, principalmente em frameworks em constante mudança. Depois de resolver os problemas de inicialização, surgiram desafios com a lógica interna do sistema de RAG (Retrieval-Augmented Generation). Ao implementar o processamento de perguntas em lote a partir de um JSON, o terminal mostrava repetidamente o erro `Missing some input keys: {'query'}`. Esse erro indicava que a cadeia de recuperação do LangChain esperava receber as perguntas de uma forma diferente da que o script principal estava enviando.

O problema estava na engenharia de prompt e no mapeamento das variáveis internas da cadeia `RetrievalQA`. O LangChain exige que a chave `query` seja explicitamente fornecida e esteja alinhada com o `PromptTemplate` definido. No começo, o uso de métodos simplificados de chamada (`run` ou `__call__`) escondia essa exigência, dificultando a identificação do erro. A solução foi abandonar essas chamadas simplificadas e usar o padrão de invocação explícita, garantindo que cada pergunta fosse corretamente colocada na chave esperada pela cadeia.

Ajustar isso foi crucial para que a busca vetorial e o modelo de linguagem trabalhassem juntos sem problemas. Além de erros no código, enfrentamos dificuldades por conta do hardware limitado e da falta de recursos para computar. O projeto rodou todo no meu computador, só com a CPU, sem poder usar uma GPU ou ferramentas como o Google Colab. Por isso, escolhemos o modelo `google/flan-t5-base`, que é bom tanto em desempenho quanto em usar poucos recursos. Só que esse modelo tem um limite chato de 512 tokens por vez, o que dava alertas e travava quando juntávamos muitos pedaços de texto na hora de buscar informações. Para resolver isso, tivemos que mudar a estratégia de como pegávamos os documentos da UFBA. Diminuímos a quantidade de documentos que a busca vetorial trazia e ajustamos o tamanho dos textos que tirávamos dos PDFs. Isso ajudou a respeitar o limite do modelo sem estragar a qualidade das respostas. Isso mostrou bem a discussão que tivemos em aula sobre como usar os recursos do computador de forma inteligente e equilibrar informação, processamento e qualidade da resposta.

Outra coisa importante foi cuidar das bibliotecas e das versões delas. Durante o projeto, as versões do `langchain`, `langchain-community`, `langchain-core` e outras bibliotecas não se davam bem, e isso causava erros na hora de importar e avisos de que as coisas estavam ficando velhas. Às vezes, errar na hora de digitar os comandos para instalar só piorava a situação. Para resolver, criamos um arquivo `requirements.txt` com tudo certinho, o que ajudou a manter o ambiente estável e facilitou para quem quisesse usar o projeto de novo.

No final, todo esse trabalho de arrumar os erros, mudar a estrutura e tomar decisões técnicas não só deixou o sistema funcionando melhor, como também mostrou como é complicado criar um sistema RAG que realmente funciona. O projeto provou que criar uma solução com modelos de linguagem é mais do que só usar uma API ou um modelo pronto, precisando entender de programação, usar os recursos com cuidado e verificar se os resultados estão bons. Essa experiência técnica foi fundamental para consolidar o aprendizado, permitindo vivenciar desafios reais enfrentados no desenvolvimento de sistemas de Inteligência Artificial aplicados a contextos institucionais, respeitando limitações técnicas e mantendo a fidelidade aos documentos oficiais analisados.