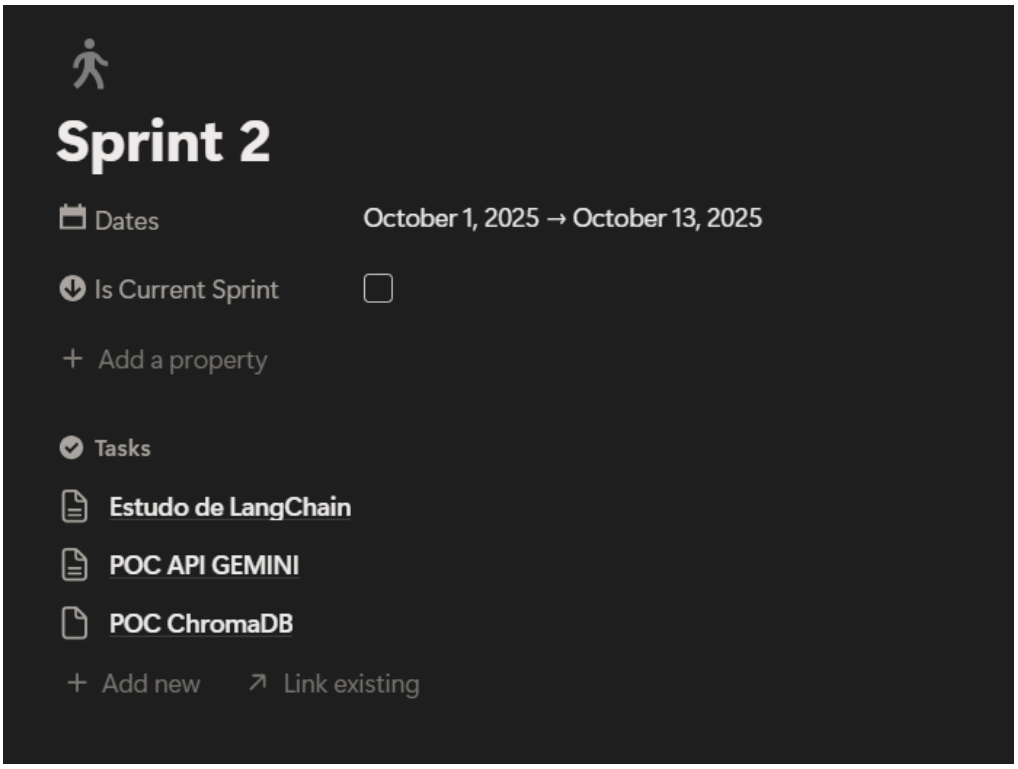
Report

Created by	Ⓢ Silvio Gonçalves
Created time	@October 17, 2025 3:44 PM
Category	Sprint 2
Last edited by	Ⓢ Silvio Gonçalves
Last updated time	@December 4, 2025 10:38 PM



Esta sprint proporcionou um conjunto significativo de aprendizados técnicos. Dividimos o trabalho em três frentes principais: estudo do **LangChain**, desenvolvimento de uma POC com a **API Gemini**, e construção de uma POC utilizando o **ChromaDB**.

1. LangChain

- Durante a análise, constatamos que o LangChain é altamente adequado ao projeto, sobretudo pela simplicidade de implementação, facilidade de configuração e amplo conjunto de ferramentas complementares.
- Desenvolvemos um modelo de estudo aplicando a API do Gemini a um tradutor, que funcionou corretamente.
- *Observação:* inicialmente utilizamos o modelo *1.5-flash*, que operava normalmente. No entanto, alguns dias depois, o modelo passou a retornar erros por ter sido descontinuado, sendo necessária a migração para a versão *2.5*.
- Identificamos que o LangChain oferece uma GUI para monitoramento de tokens e requisições, o **LangSmith**, que se mostrou extremamente útil para análise de uso dos modelos. Além disso, a ferramenta fornece um ambiente simples de teste via *uvicorn*, permitindo validações locais rápidas.
- A estruturação modular do LangChain facilitou o reaproveitamento do pipeline em outras aplicações, já que ele próprio gera uma API de acesso simplificado.
- O modelo de arquitetura foi facilmente organizado, abrangendo desde o carregamento e pré-processamento dos textos até a recuperação dos dados e geração de respostas resumidas com base em trechos específicos das bulas.
- Notou-se que o desempenho e a precisão do sistema dependem fortemente da forma como os **chunks** são gerados — um pré-processamento mais refinado é crucial para garantir que o modelo identifique corretamente medicamentos e suas respectivas informações.
- Um ponto positivo adicional foi a rápida integração com o ChromaDB e o LLM, tornando a etapa de recuperação vetorial notavelmente eficiente.

---

2. API do Gemini para Embeddings

- Desenvolvemos uma POC utilizando a API do Gemini para realizar a chunkerização de PDFs e convertê-los em vetores, posteriormente armazenados no ChromaDB para acesso rápido e de baixo custo computacional.

- Encontramos dificuldades com a documentação, que apresentava informações inconsistentes quanto ao modelo correto a ser utilizado para embeddings. Após discussão em um fórum oficial do Google no Discord, identificamos que a versão adequada era **"models/gemini-embedding-001"**.
- Com esse modelo, conseguimos vetorizar um PDF de 10 páginas, armazenar os vetores no ChromaDB e utilizá-los em consultas semânticas.
- Entretanto, ao testar um PDF maior (220 páginas), excedemos rapidamente o limite de requisições do *free tier*, impedindo a criação completa dos chunks.
- Como próximo passo, planejamos testar arquivos menores ou realizar estratégias de divisão para contornar o limite de requisições.

---

3. ChromaDB

- O ChromaDB demonstrou ser extremamente rápido e adequado para prototipagem e POCs, alinhando-se bem ao estágio atual do projeto.
- A integração nativa com o LangChain e o suporte a metadados simplificaram a criação e consulta ao banco vetorial, proporcionando respostas consideravelmente mais rápidas do que inferências diretas via LLM (15,78s com RAG contra 26s sem RAG).
- Contudo, identificamos que a qualidade da recuperação semântica está diretamente relacionada à clareza e consistência dos chunks. Em casos onde o nome do medicamento não estava adequadamente associado ao trecho, o modelo não conseguia localizar a informação mesmo ela existindo na base.

Conclusão Geral

O principal aprendizado desta sprint foi compreender que a integração entre **LangChain**, **ChromaDB** e **Gemini AI** oferece uma base sólida e eficiente para a construção de um sistema RAG. No entanto, o desempenho depende fortemente da qualidade do pré-processamento, especialmente da estratégia de chunkerização e vetorização dos dados.