# RELATÓRIO DE DESENVOLVIMENTO DO PROJETO SYNAPSE

**Disciplina:** Extensão III

**Professor:** Paulo Henrique Maia

### Equipe:

- **Bruno Sousa** – Backend
- **Ian Soares** – Backend e Relatório
- **Vanessa Andrade** – Frontend, Slides
- **Nicollas Ney** – Frontend, Documentação

# 1. Introdução

O **Synapse** surgiu a partir de uma necessidade real identificada por um dos integrantes da equipe: a dificuldade em coletar e estruturar requisitos durante o desenvolvimento de software em disciplinas como Análise e Projeto de Software (APS). Nessas atividades, é comum depender de clientes fictícios ou entrevistas longas para levantar requisitos, criando inconsistências e retrabalho.

Dessa dor nasceu a ideia de criar uma ferramenta capaz de receber uma descrição, mesmo que informal, e gerar automaticamente requisitos, histórias de usuário e critérios de aceitação, garantindo padronização e aceleração do processo de engenharia de requisitos.

O Synapse foi desenvolvido utilizando:

- **Backend:** Python e FastAPI
- **Frontend:** React, Tailwind
- **LLM:** Gemini
- **Transcrição de áudio:** Whisper
- **Arquitetura:** RAG baseada em arquivos .txt

Ao longo das três entregas do projeto, a equipe implementou, validou e evoluiu funcionalidades que transformaram a ideia inicial em um protótipo funcional.

# 2. Entrega 1

## 2.1 Objetivo da entrega

Construir a primeira interface para entrada de texto e estabelecer comunicação entre o frontend e o backend, permitindo o envio de descrições do usuário ao servidor.

## 2.2 Implementação

A interface consistia em um campo simples onde o usuário podia inserir uma descrição textual do sistema desejado. O backend recebia o texto e retornava a resposta da LLM.

### 2.3 Desafios enfrentados

- Problemas com CORS dificultou a comunicação entre front e back (já que estavam rodando em portas diferentes).
- Dificuldade com as dependências do projeto.
- Dificuldade em configurar e rodar o projeto com facilidade em mais de uma máquina.

### 2.4 Soluções adotadas

- Criação do “requirements.txt” para concentrar todas as depêndencias necessárias para executar o projeto.

### 2.5 Lições aprendidas

- A importância de organizar bem tudo que for necessário para o projeto rodar facilmente em qualquer máquina.
- Familiarização com CORS.

## 3. Entrega 2

### 3.1 Objetivo da entrega

Permitir que o sistema, a partir da descrição inserida pelo usuário, gerasse automaticamente:

- Requisitos funcionais
- Histórias de usuário
- Critérios de aceitação
- Resumo em PDF

### 3.2 Implementação

Utilizando a API do Gemini, criamos o processo responsável por processar o texto do usuário e gerar os artefatos a partir de templates fornecidos pelo RAG.

O time trabalhou na padronização dos formatos, como:

- Padrão das respostas geradas pela LLM, incluindo requisitos, histórias de usuário e critérios de aceitação

### 3.3 Desafios enfrentados

- A resposta gerada pelo Gemini era mal formatada.
- Implementação da memória da LLM.

### 3.4 Soluções adotadas

- Alocação da melhoria da formatação da resposta do Gemini para a última sprint.
- Leitura da documentação do LangChain para conseguir implementar a memória da LLM.

### 3.5 Lições aprendidas

- Ajuste do prompt para que a resposta saia mais próxima do esperado possível.
- Importância da memória para o funcionamento de uma aplicação como a nossa, que exige o contexto da conversa como um todo para que as respostas da LLM façam sentido.

# 4. Entrega 3

## 4.1 Objetivo da entrega

Integrar planejamento de sprints ao Synapse e melhorar a formatação da saída da LLM.

## 4.2 Implementação

Implementamos a geração de planejamento de sprints e aprimoramos a formatação das respostas da LLM. Também fizemos pequenas atualizações no frontend.

## 4.3 Desafios e problemas encontrados

- Necessidade de reorganizar o código devido à adição de novas funcionalidades.
- Implementação do frontend foi um desafio devido ao pouco tempo que tivemos.

## 4.4 Como foram resolvidos

- Refatoração do código
- Trabalho de toda equipe (inclusive desenvolvedores backend) para conseguir entregar tudo à tempo.

## 4.5 Aprendizados

- Importância da organização do código e do trabalho em conjunto das equipes de frontend e backend.

# 5. Desafios gerais do projeto

## 5.1 Desafios técnicos

- Integração completa do backend e frontend.
- Acerto do prompt para gerar bons requisitos.
- Consistência da formatação das respostas geradas pela LLM.
- Entendimento de como criar e como alimentar o RAG.

## 5.2 Desafios organizacionais

- Manter o mesmo ritmo de desenvolvimento entre frontend e backend.
- Divisão das responsabilidades.
- Comunicação
- Gerenciamento de tempo.

# 6. Lições aprendidas

### 6.1 Lições técnicas

- O prompt é uma das partes mais importantes para a aplicação que depende de LLM.
- O RAG pode melhorar a qualidade das respostas geradas.

### 6.2 Lições organizacionais

- Importância da comunicação entre desenvolvedores do frontend e backend.
- Importância de documentar o projeto desde o início.
- Uma divisão clara de tasks melhora a produtividade.

# 7. Conclusão

O Synapse evoluiu de uma necessidade acadêmica para um protótipo funcional com potencial de ser usado por estudantes e por equipes de desenvolvimento. Ao longo das entregas, acredito que a nossa equipe adquiriu novos conhecimentos técnicos, conseguiu estruturar um projeto o qual nenhum dos membros havia trabalhado antes com LLMs e RAG, e adquiriu mais experiência de desenvolvimento, como também mais experiência de como se organizar e trabalhar em equipe.