PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

Assistente inteligente para análise e decisão no mercado financeiro

Projeto Final II

Caio Valle de Vasconcellos das Neves Moraes

PROJETO FINAL DE GRADUAÇÃO

CENTRO TÉCNICO CIENTÍFICO - CTC

DEPARTAMENTO DE INFORMÁTICA

Rio de Janeiro, novembro de 2025

Caio Valle de Vasconcellos das Neves Moraes

Assistente inteligente para análise e decisão no mercado financeiro

Projeto Final II

Relatório de Projeto Final, apresentado ao programa de Graduação do Departamento de Informática da PUC-Rio como requisito parcial para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Augusto Cesar Espíndola Baffa

Rio de Janeiro
novembro de 2025

“Alis grave nil”
Lema da PUC-Rio

Resumo

Moraes, Caio Valle de Vasconcellos das Neves. Baffa, Augusto Cesar Espíndola. Assistente inteligente para análise e decisão no mercado financeiro, Projeto Final II. Rio de Janeiro, 2025, 38 páginas. Relatório de Projeto Final II – Departamento de Informática. Pontifícia Universidade Católica do Rio de Janeiro.

Este projeto tem a proposta de desenvolver um assistente inteligente voltado para a análise e apoio à tomada de decisão no mercado financeiro, um setor caracterizado por um volume massivo e altamente complexo de dados. O método empregado para solucionar este problema é a implementação de um modelo RAG, Retrieval-Augmented Generation. Esta arquitetura utiliza uma base de dados vetorial para permitir que um Modelo de Linguagem acesse informações em tempo real, superando seu conhecimento estático. A base vetorial é alimentada continuamente por múltiplas fontes de dados, integrando dados públicos (notícias recentes via API, posts de redes sociais via web scraping) com dados privados do usuário como os ativos que compõem sua carteira. Os resultados são demonstrados através de um protótipo funcional que permite ao usuário não apenas gerenciar seu portfólio, mas interagir com o assistente. O sistema é capaz de interpretar consultas em linguagem natural e utilizar ferramentas de software para buscar o contexto relevante, seja na base vetorial ou em APIs, antes de formular uma resposta. Conclui-se que, no contexto deste estudo, o uso de arquiteturas RAG aplicadas às finanças é uma abordagem viável e eficaz. O sistema demonstrou a capacidade de integrar informações dispersas e oferecer suporte analítico personalizado, ampliando o acesso à informação e aprimorando a compreensão de cenários complexos do mercado.

Palavras-chave

modelos de linguagem; inteligência artificial; mercado financeiro; análise de ativos; assistente virtual; projeto final II.

Abstract

Moraes, Caio Valle de Vasconcellos das Neves. Baffa, Augusto Cesar Espíndola. Intelligent Assistant for Financial Market Analysis and Decision Support, Undergraduate Thesis II. Rio de Janeiro, 2025, 38 pages. Undergraduate Thesis II – Departamento de Informática. Pontifícia Universidade Católica do Rio de Janeiro.

This undergraduate thesis presents the objective of developing an intelligent assistant aimed at analysis and decision support in the financial market, a sector characterized by a massive volume of data. The method employed to solve this problem is the implementation of a RAG, Retrieval-Augmented Generation architecture. This architecture utilizes a vector database to allow a Large Language Model to access real-time information, overcoming its static knowledge. The vector database is continuously fed by multiple data sources, integrating public data (recent news via API, social media posts via *web scraping*) with the user's private data (uploaded PDF reports and the assets composing their portfolio). The results are demonstrated through a functional prototype that allows the user to not only manage their portfolio but also interact with the assistant. The system is capable of interpreting natural language queries and using software "tools" to retrieve relevant context (whether from the vector database or APIs) before formulating a response. It is concluded that the use of RAG architectures applied to finance is a viable and effective approach. The system demonstrated the ability to integrate disperse information and offer personalized analytical support, broadening access to information and enhancing the understanding of complex market scenarios.

Keywords

language models; artificial intelligence; financial market; asset analysis; virtual assistant; Undergraduate Thesis II.

Lista de Figuras

- Figura 1 - Diagrama do sistema
- Figura 2 - Fluxo de dados para a base vetorial
- Figura 3 - Diagrama de classe

Lista de Tabelas

- Tabela 1 - Testes e resultados

Sumário

1. Introdução 9
2. Situação atual 12
    2.1 Conceitos fundamentais 12
    2.2 Análise de trabalhos existentes 14
3. Objetivo 15
4. Atividades Realizadas 17
    4.1 Estudos preliminares 17
    4.2 Estudos conceituais e de tecnologia 17
    4.3 Testes e protótipos para aprendizado e demonstração 18
    4.4 Método 19
5. Projeto e especificação do sistema 20
    5.1 Funcionalidade e estrutura 20
    5.2 Características marcantes e contribuições 29
    5.3 Descrição de uso 29
    5.4 Engenharia de requisitos e modelagem 30
6. Implementação e avaliação 31
    6.1 Planejamento e Métricas de Validação 31
    6.2 Resultado dos testes funcionais 31
    6.3 Resultados Obtidos 33
    6.4 Comentários sobre a implementação 34
7. Considerações finais 34
    7.1 Contribuições do trabalho 35
    7.2 Aprendizado Obtido 35
    7.3 Limitações e Recomendações 36
    7.4 Oportunidades para trabalhos futuros 36
8. Referências 37

1. Introdução

Apresentação

O avanço das tecnologias de inteligência artificial, aliado à crescente disponibilidade de dados e à facilidade de acesso a essas informações, tem transformado profundamente a forma como as pessoas interpretam, analisam e utilizam o conhecimento no ambiente digital. Hoje, modelos de IA conseguem identificar padrões complexos em grandes volumes de dados com uma precisão impossível de ser alcançada manualmente, o que redefine processos de tomada de decisão em diversos contextos.(Nogueira, 2024)

Por outro lado, a integração entre IA e análise de dados se revela não apenas uma evolução tecnológica, mas também uma poderosa ferramenta estratégica para múltiplos setores, especialmente o financeiro. Nesse campo, onde cada decisão pode representar ganhos ou perdas significativas, a capacidade de prever tendências, automatizar processos e reduzir riscos torna-se um diferencial indispensável. A IA Generativa, em particular, está superando uma grande barreira, tornando-se um diferencial competitivo ao transformar grandes volumes de dados não estruturados em informação útil e acessível em linguagem natural (NÚCLEA, 2025). A união entre algoritmos inteligentes e informações estruturadas permite desde avaliações mais robustas de crédito até a criação de sistemas de investimento automatizados, proporcionando eficiência, segurança e competitividade.

Esse avanço tecnológico da inteligência artificial é ainda mais relevante quando se considera o cenário de pesquisa do sistema financeiro no Brasil. Segundo a 17ª edição do Observatório Febraban (FEBRABAN, 2025), a maioria dos brasileiros (55%) admite entender pouco ou nada sobre educação financeira. A pesquisa revela ainda uma visão restrita sobre o tema: para 47% dos entrevistados, educação financeira resume-se apenas ao controle de gastos básicos, ignorando estratégias de investimento e formação de patrimônio. Assim, a aplicação de IA e análise de dados não só otimiza processos, mas atua como um equalizador, traduzindo conceitos complexos e democratizando o acesso a estratégias de investimento que, de outra forma, estariam fora do alcance dessa parcela da população.

Nesse cenário, sistemas inteligentes baseados em modelos de linguagem natural emergem como soluções promissoras para apoiar o processo de análise e tomada de decisão no mercado financeiro.

Este trabalho apresenta o desenvolvimento de um assistente inteligente voltado para a análise e apoio à decisão no mercado financeiro, cuja principal finalidade é integrar múltiplas fontes de informação e disponibilizá-las de forma clara e interativa para o usuário.

A integração desses elementos permite que o sistema não apenas interprete consultas complexas em linguagem natural, mas também recupere, sintetize e apresente informações relevantes sobre ativos, tendências e indicadores de mercado. Isso torna o processo de análise mais fluido e intuitivo, mesmo para usuários com experiência limitada no setor financeiro. Além disso, ao incorporar análises, visualizações dinâmicas e métricas atualizadas, o assistente oferece uma camada adicional de compreensão, facilitando a identificação de padrões e a avaliação comparativa entre diferentes oportunidades de investimento.

Com essa abordagem, o projeto final realizado demonstra que os modelos de linguagem não se limitam à geração de texto, mas podem funcionar como ferramentas inteligentes de análise, capazes de integrar dados de diferentes fontes, contextualizar informações em tempo real e apoiar decisões baseadas em evidências. Assim, a plataforma proposta reforça o potencial dos LLMs como tecnologias transformadoras no ambiente financeiro, ampliando o acesso à informação qualificada e facilitando o entendimento de um mercado tradicionalmente complexo.

Além disso, o avanço acelerado das tecnologias de IA generativa e sua crescente adoção em processos de apoio à decisão tornam o novo tema ainda mais pertinente, tanto do ponto de vista acadêmico quanto profissional. A oportunidade de trabalhar com sistemas baseados em LLMs representa uma chance de aprofundar habilidades altamente demandadas no mercado de tecnologia e, ao mesmo tempo, contribuir para a compreensão de como essas ferramentas podem ser aplicadas em contextos críticos, como o mercado financeiro. Assim, a alteração do tema se justifica não apenas como uma adequação às minhas preferências, mas como uma decisão estratégica que potencializa a relevância e o impacto do projeto.

Sobre o ambiente computacional

O desenvolvimento do sistema foi realizado em um ambiente operacional Windows. A arquitetura full-stack do projeto utilizou **Python** para o backend e **TypeScript** para o frontend.

O backend foi implementado como uma API RESTful utilizando o *framework* **FastAPI** (FASTAPI, 2025), escolhido por sua alta performance.

O **SQLAlchemy** foi usado como ORM para a modelagem do banco de dados relacional, enquanto o **Pydantic** foi empregado para a validação dos *schemas* de dados da API.

O frontend foi construído com a biblioteca **React** , permitindo a criação de uma interface de usuário reativa para o *chat* e o gerenciamento da carteira.

O núcleo de IA foi orquestrado pela biblioteca **LangChain**, usada para implementar a arquitetura RAG e gerenciar as "Tools". O modelo de linguagem utilizado foi o **Gemini** da Google.

Para a coleta de dados, um conjunto de bibliotecas foi empregado: **yfinance** (YFINANCE, 2025) para cotações de mercado; **newsapi** e **newspaper** para a busca e extração de notícias ; e **Selenium** (SELENIUM, 2025) para o *web scraping* do Twitter (X).

Adequação do trabalho

Este trabalho se alinha aos requisitos de um Projeto Final , pois representa o ciclo completo de concepção, planejamento e implementação de um sistema de software complexo.

O projeto não se limitou ao desenvolvimento de uma aplicação simples, mas envolveu a integração de múltiplas tecnologias de ponta para resolver um problema relevante: a utilização de dados para análise no mercado financeiro.

A implementação de uma arquitetura sofisticada, que combina coleta de dados em tempo real originadas de *web scraping* e APIs, processamento de dados não estruturados e inteligência artificial generativa, demonstra a complexidade esperadas de um trabalho de conclusão de curso.

Durante o desenvolvimento do projeto, pude colocar em prática diversos conceitos que aprendi em disciplinas de **programação, programação modular, programação orientada a objetos**, **modelagem de dados**, **banco de dados**, **inteligência artificial**, **engenharia de requisitos**, **processos** e **engenharia de software**.

- Conceitos de **Engenharia de Software** e **Programação** foram a base para a definição de uma arquitetura full-stack robusta. Isso incluiu a implementação de uma API RESTful em Python (FastAPI), o desenvolvimento de uma interface reativa no *frontend* e a aplicação de padrões de projeto para garantir uma clara separação de responsabilidades e manutenibilidade do código.
- A **Programação Orientada a Objetos (POO)** foi fundamental na estruturação do código Python no backend, onde classes foram utilizadas para modelar entidades do sistema e para criar serviços que encapsulam a lógica de todo o sistema, como o cálculo do preço médio. Já a **Programação Modular** garantiu uma clara separação de responsabilidades entre os componentes como os serviços que lidam com a lógica do sistema.
- As disciplinas de **Modelagem de Dados** e **Banco de Dados** foram fundamentais em dois níveis distintos e complementares. Primeiro, na modelagem relacional tradicional para o gerenciamento das carteiras dos usuários e suas transações (tabelas Assets e Position). Segundo, na

aplicação de bancos de dados vetoriais, uma tecnologia não-convencional essencial para armazenar os *embeddings* de texto e suportar o pipeline de RAG.

- Em **Inteligência Artificial**. Apesar de no periodo cursado não ter aprofundado em modelos de linguagem, a disciplina foi essencial para poder entender a essência das bases conceituais que sustentam o funcionamento das LLMs.
- Por fim, conceitos de **Engenharia de Requisitos** e **Processos** foram essenciais na prática.

2. Situação atual

A análise de mercado financeiro evoluiu significativamente nas últimas décadas. Ela foi impulsionada pelo crescimento exponencial do volume de dados e pelo avanço de técnicas de processamento de linguagem natural e aprendizado de máquina. O domínio, que antes dependia da análise manual de relatórios e jornais, hoje é caracterizado por um fluxo contínuo de informações em alta frequência.

Atualmente, essas informações provêm tanto de fontes estruturadas — como cotações da bolsa e dados fundamentalistas — quanto de fontes não estruturadas. Este último grupo apresenta a maior complexidade, incluindo o fluxo constante de notícias e o sentimento de mercado expresso em redes sociais. A literatura recente corrobora a relevância dessa análise híbrida. Liu et al. (2024) destacam que a interpretação de nuances em textos não estruturados por Modelos de Linguagem (LLMs) supera as abordagens tradicionais, sendo crítica para entender a dinâmica de ativos voláteis . Complementarmente, Qi (2025) demonstra que a incorporação de variáveis de sentimento extraídas por LLMs em modelos de previsão de mercado aumenta a acurácia das projeções quando comparada a modelos que usam apenas dados numéricos.

O principal desafio, portanto, reside na variedade e na velocidade dessas informações. O difícil não é mais acessar os dados, mas sim conseguir juntar e entender, de forma rápida e precisa, fontes muito diferentes: um balanço em PDF, uma cotação em tempo real e o sentimento de uma rede social. É essa necessidade de uma análise unificada que motiva a criação de soluções como a deste trabalho.

2.1 Conceitos fundamentais

A solução proposta neste trabalho fundamenta-se na utilização de três tecnologias que têm ganhado bastante destaque nos últimos anos: Modelos de Linguagem, a arquitetura RAG e bases de dados vetoriais.

Modelos de Linguagem (LLMs) e Arquitetura Transformer: Os LLMs utilizados, como o Gemini, baseiam-se na arquitetura de redes neurais Transformer. Diferente de redes recorrentes anteriores, o Transformer utiliza o mecanismo de Autoatenção (Self-Attention), que permite ao modelo ponderar a importância de cada palavra em relação a todas as outras na sequência de entrada, capturando dependências de longo prazo. Matematicamente, a atenção é calculada pela fórmula:

$$\mathrm{Attention}(Q,K,V)=\mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

em que Q (Query), K (Key) e V (Value) representam matrizes de vetores, d_k corresponde à dimensão dos vetores de keys e K^T indica a transposta da matriz K. A função softmax é utilizada para normalizar os valores de similaridade obtidos pelo produto escalar entre as matrizes de queries e keys, convertendo-os em pesos de atenção que somam 1 e representam a importância relativa de cada elemento da sequência.

Essa capacidade de processamento paralelo permite que o modelo "compreenda" e gere texto com alta coerência. No contexto financeiro, isso traduz-se na capacidade de realizar inferências complexas sobre dados não estruturados (LIU et al., 2024).

RAG (Retrieval-Augmented Generation): Um dos desafios dos LLMs é o seu "conhecimento estático" e a possibilidade de "alucinações". O padrão RAG endereça essa limitação combinando o poder generativo do LLM com um sistema de busca em uma base externa e atualizada.

O processo é definido em duas etapas:

1. **Recuperação ($p_\eta(z|x)$):** Dada uma consulta de entrada x, o sistema recupera os k documentos z mais relevantes.
2. **Geração ($p_\theta(y|x,z)$):** O modelo gerador condiciona a probabilidade da sequência de saída y tanto na entrada x quanto nos documentos recuperados z.

Segundo um estudo feito por Zhang et al. (2023) identificou que notícias financeiras são frequentemente sucintas e carecem de contexto e ao utilizar RAG para "ancorar" a resposta do modelo em dados externos recuperados, observou-se um ganho de precisão entre 15% e 48% em comparação a modelos sem recuperação . Para este projeto, essa base conterá as notícias, relatórios e postagens coletadas, fornecendo o contexto factual necessário.

Bases de Dados Vetoriais: Para que a contextualização do modelo funcione eficientemente, os dados externos precisam ser armazenados de forma que permitam uma busca semântica. As bases vetoriais, como o supabase, realizam isso ao armazenar os dados como embeddings. Quando um usuário faz uma pergunta, ela também é convertida em um vetor, e o sistema busca os vetores de dados mais "próximos" semanticamente para usar como contexto na resposta

do LLM. Para recuperar o contexto mais relevante para a pergunta do usuário, o sistema utiliza o cálculo da **Similaridade de Cosseno**. Dado um vetor de consulta A e um vetor de documento B, a similaridade é dada por:

$$\text{Sim}(A,B) = \cos(\theta) = \frac{A \cdot B}{|A||B|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Este cálculo permite que o sistema identifique e recupere os segmentos de notícias ou relatórios (B) que estão matematicamente mais alinhados com a intenção da pergunta do usuário (A), superando as limitações da busca por palavras-chave exatas (ZHANG et al., 2023) .

Métricas de Avaliação: Para validar a eficácia do sistema RAG, utilizam-se métricas focadas nas duas etapas do processo: recuperação e geração.

- **Relevância de Recuperação:** Conforme definido na equação anterior, esta métrica quantifica o quão próximo o vetor do documento recuperado está da pergunta do usuário. Um *score* alto indica que o sistema encontrou contexto semanticamente relevante.
- **Acurácia da Resposta (Validação Funcional):** Medida binária (Sucesso/Falha) determinada pela verificação se a resposta gerada pelo LLM está factualmente alinhada com as informações recuperadas e se atende à intenção do *prompt* do usuário (ZHANG et al., 2023).

2.2 Análise de trabalhos existentes

A aplicação de LLMs na tomada de decisão financeira tem sido objeto de estudos recentes que validam os componentes da arquitetura proposta.

Previsão de Mercado Baseada em Sentimento: A integração de sinais textuais com dados de mercado é um campo ativo de pesquisa. Qi (2025) propôs um framework metodológico combinando classificação de sentimento e previsão de séries temporais. Seus experimentos demonstram que LLMs, ao capturarem nuances de sentimento financeiro, superam abordagens tradicionais e que a inclusão dessas variáveis de sentimento melhora métricas de previsão. Isso fundamenta a estratégia deste projeto de utilizar o LLM não apenas para conversar, mas para analisar tendências.

Aprimoramento da Análise de Sentimento via RAG: A eficácia da arquitetura proposta neste trabalho encontra respaldo direto no estudo de Zhang et al. (2023). Os autores abordaram o problema da "escassez de contexto" em textos financeiros curtos (como manchetes e *tweets*), que frequentemente levam modelos de linguagem a interpretações errôneas. Ao implementar um *framework* que utiliza RAG para recuperar contexto externo de notícias e redes sociais antes da análise, o estudo demonstrou um aumento de precisão (*accuracy*) entre **15% e 48%** em comparação com LLMs que operam isoladamente. Esse

resultado valida a hipótese central deste projeto de que a contextualização via base vetorial é indispensável para a confiabilidade de assistentes financeiros.

Gestão de Risco e Assimetria de Sentimento: A capacidade de interpretar sentimentos negativos é crítica para a segurança do investidor. Wu (2024) aplicou LLMs para pontuar o impacto de notícias em portfólios e identificou uma assimetria na reação do mercado: os preços dos ativos demonstraram ser significativamente mais sensíveis a notícias negativas. Esse achado acadêmico fundamenta a necessidade de módulos de "Alertas de Risco" em assistentes inteligentes.

Hedging Dinâmico: Avançando para a execução, Yang et al. (2025) propuseram um framework onde LLMs ajustam posições de proteção (hedging) em tempo real baseados em análise de notícias. O estudo comprovou que essa abordagem dinâmica supera métodos estáticos, elevando o retorno ajustado ao risco e reduzindo o drawdown máximo em cenários de volatilidade.

3. Objetivo

O objetivo principal deste Projeto Final é o desenvolvimento de um assistente inteligente especializado no mercado financeiro. O sistema é fundamentado em modelos de linguagem natural, utilizando a ideia de “Retrieval-Augmented Generation” sobre uma base de dados vetorial.

O sistema permite a criação e o gerenciamento de carteiras personalizadas de ativos, além de possibilitar a interação direta com um assistente financeiro virtual capaz de compreender e responder em linguagem natural. As informações processadas incluem notícias diárias, postagens em redes sociais, indicadores e relatórios sobre os principais ativos, de modo a oferecer uma visão ampla e atualizada do mercado.

A base técnica do assistente é o modelo de linguagem Gemini, desenvolvido pela Google, classificado como um Large Language Model (LLM). Esse modelo é projetado para compreender contextos complexos, com alto números de tokens, gerar respostas textuais com elevado grau de coerência e adaptar-se a diferentes tipos de consulta. O sistema proposto utiliza o Gemini para interpretar as solicitações dos usuários, consolidar informações financeiras dispersas e fornecer respostas claras e contextualizadas.

O sistema emprega também a técnica de RAG,Retrieval-Augmented Generation, que combina o poder de geração de linguagem do modelo com um processo de busca em base vetorial contendo os dados coletados como textos, notícias e relatórios financeiros. Adicionalmente, o desenvolvimento de um módulo de análise de ativos permite a visualização de indicadores e tendências de mercado, utilizando dados relevantes e atualizados.

Diante do crescimento exponencial do volume de dados financeiros e da necessidade de interpretações rápidas e precisas, o uso de modelos de linguagem em sistemas interativos representa um avanço significativo na forma como os usuários podem explorar e compreender informações de alta complexidade.

Dessa forma, o presente projeto visa evidenciar a capacidade dos LLMs de auxiliar na análise financeira, demonstrando como esses modelos podem atuar como componentes centrais em sistemas de apoio à decisão. Ao combinar técnicas avançadas de inteligência artificial, processamento de linguagem natural e visualização de dados em uma plataforma unificada e de fácil acesso, o projeto busca mostrar que é possível transformar grandes volumes de informações fragmentadas em análises coerentes, contextualizadas e diretamente aplicáveis ao processo decisório do usuário.

A finalidade do assistente é integrar múltiplas fontes de informação como notícias, redes sociais e relatórios enviados pelo usuário para fornecer análises e recomendações contextualizadas.

O escopo do projeto foi definido para criar uma ferramenta de alto valor agregado, com as seguintes características:

- **Usuário:** O sistema foi projetado para apoiar qualquer usuário, seja ele uma pessoa experiente ou não a respeito do mercado financeiro.
- **Escopo de Dados:** O projeto busca criar uma ferramenta que combina, de forma acessível, **dados públicos** (notícias e redes sociais) com **dados privados do usuário**. Este é um diferencial chave, pois o contexto do assistente inclui não apenas os relatórios em PDF que o usuário envia, mas também os ativos e o preço médio da sua carteira de investimentos.
- **Escopo Técnico:** O sistema RAG é utilizada para garantir que as análises sejam contextualizadas e baseadas em dados factuais recentes, superando o conhecimento estático e as limitações de "alucinação" dos LLMs genéricos.

Para alcançar o objetivo principal, os seguintes objetivos específicos (entregáveis) foram definidos e concluídos:

- O desenvolvimento de um protótipo funcional do assistente. Isso inclui a interface de usuário, o frontend em react, e o servidor backend, FastAPI, que orquestra a lógica de negócios e as chamadas ao LLM.
- A implementação de uma **base vetorial funcional**, utilizando o banco de dados supabase, contendo todo o pipeline de ingestão para processar, vetorizar e armazenar os dados coletados de notícias, redes sociais e relatórios.
- A elaboração do relatório técnico completo e documentado (este documento), detalhando todas as fases do projeto, desde a concepção até a avaliação .
- A preparação da apresentação de defesa perante a banca examinadora.

4. Atividades Realizadas

4.1 Estudos preliminares

Ao iniciar o Projeto Final II, os conhecimentos prévios se concentravam em desenvolvimento Web com Python e em fundamentos práticos e teóricos de Modelos de Linguagem e arquitetura RAG. Em contrapartida, o tema de bases vetoriais era totalmente novo.
Paralelamente, havia um entendimento sólido e prático dos fundamentos teóricos e práticos essenciais para o trabalho com Modelos de Linguagem de Grande Escala e, crucialmente, com a arquitetura Retrieval-Augmented Generation. Essa familiaridade com o RAG significava que já havia uma compreensão para integração de mecanismos de recuperação de informação externa para aprimorar a precisão e a relevância das respostas geradas pelos modelos.

No entanto, o projeto se apresentava como uma oportunidade de expansão significativa desses limites. O tema das bases de dados vetoriais, um componente fundamental para a eficácia do sistema de geração aumentada por recuperação e para a indexação eficiente de dados, era um campo novo ao iniciar o projeto.

Essa lacuna de conhecimento representou um desafio e, ao mesmo tempo, o principal foco de aprendizado prático e teórico do projeto.

4.2 Estudos conceituais e de tecnologia

Para o desenvolvimento deste trabalho, foi necessário realizar um aprofundado estudo conceitual e tecnológico, focado em superar as limitações centrais dos LLMs (conhecimento estático e "alucinações") e adaptá-los ao domínio volátil do mercado financeiro. O foco principal foi a busca por técnicas eficazes para contextualizar o modelo, visando aprimorar significativamente a confiabilidade das respostas geradas.

Este estudo se concentrou na orquestração da arquitetura RAG. Isso exigiu um aprofundamento em *frameworks* de LLM, com foco principal no LangChain (LANGCHAIN, 2025). O aprendizado não se limitou a simples chamadas de API, mas sim à compreensão de seus componentes avançados, como os Agentes e o sistema de Ferramentas. O domínio das "Tools" foi crucial, pois é o mecanismo que permite ao LLM raciocinar e decidir dinamicamente qual fonte de dados (base vetorial, API de cotações, etc.) consultar para responder a uma pergunta.

Paralelamente, foi necessário um estudo completo sobre a aplicação e uso de bases vetoriais, um tema que era novo no início do projeto. Este estudo envolveu três conceitos-chave: a compreensão de como modelos de linguagem transformam texto em representações numéricas (Embeddings); o aprendizado de técnicas para dividir documentos longos em "pedaços" (Chunking); e a aplicação de buscas por similaridade para recuperar os *chunks* de informação mais relevantes.

Adicionalmente, para garantir a alimentação de dados do assistente, o segundo pilar de estudo foi a implementação de um pipeline robusto de coleta de dados de diversas fontes. A pesquisa inicial mostrou que a newsapi era eficaz para a *descoberta* de manchetes, mas limitada, pois não retornava o conteúdo completo. Foi necessário um estudo complementar para integrar a biblioteca newspaper, que é especializada em extrair o texto completo de artigos. Também foi estudada e integrada a biblioteca yfinance para acessar dados de mercado essenciais, como cotações em tempo real. Por fim, para capturar a percepção pública no X (Twitter), foi necessário empregar técnicas avançadas de *web scraping* com Selenium, incluindo o estudo de como contornar o carregamento de conteúdo dinâmico.

4.3 Testes e protótipos para aprendizado e demonstração

Para consolidar os estudos teóricos e demonstrar a viabilidade técnica da proposta, foram desenvolvidos protótipos focados nos componentes críticos do sistema.

Nos testes iniciais, a validação concentrou-se na integração de dados quantitativos. Foram elaborados scripts para consulta de dados de mercado via API do yfinance, com o objetivo de alimentar o modelo de linguagem e verificar sua aptidão para interpretar séries numéricas em tempo real. Com o sucesso desses testes, a complexidade foi elevada para a criação de um sistema de simulação de carteira de investimentos, onde as posições eram atualizadas automaticamente e submetidas ao modelo para a geração de insights de desempenho.

Quanto à contextualização com dados qualitativos, a abordagem seguiu uma evolução incremental. O processo iniciou-se com a inserção manual de notícias no prompt para aferir a capacidade do modelo de interpretar o contexto e gerar respostas coerentes. Subsequentemente, o fluxo foi automatizado através da integração com a newsapi. A etapa final consistiu na validação de agentes autônomos utilizando o LangChain, testando a capacidade do modelo de decidir, através do uso de Tools, o momento exato de acionar a busca externa por notícias recentes diante de uma consulta do usuário.

Foi desenvolvido um protótipo do pipeline RAG para validar a ingestão de dados na base vetorial. Este teste foi implementado através de um script dedicado ao processamento de textos, responsável pela segmentação dos documentos em chunks, sua vetorização e armazenamento, garantindo a viabilidade da recuperação semântica para o sistema final.

4.4 Método

O processo de projeto e desenvolvimento seguiu um método iterativo e incremental, focado na construção e validação de funcionalidades em fases. As atividades desenvolvidas seguiram a seguinte ordem:

1. Estudo sobre APIs de notícias, funcionamento de frameworks para LLM, APIs para mercado financeiro, conceitos de LLM.
2. O projeto foi iniciado com os testes de prototipagem para entender o funcionamento das tecnologias e como elas poderiam ser usadas no desenvolvimento.
3. Após os testes, foi criada a estrutura base da aplicação, que consistia na lógica em Python para o gerenciamento de ativos, permitindo ao usuário simular a compra e venda.
4. Com essa estrutura pronta, foi realizada a implementação do atualizador de preços dos ativos (API do yfinance) e a primeira integração com a LLM para interação com os ativos da carteira.
5. Foi desenvolvida a interface inicial do usuário (frontend) para permitir a interação e o gerenciamento dos ativos.
6. Implementou-se a base vetorial como um serviço, responsável por processar as informações das APIs (devidamente "tagueadas") para serem consumidas pela LLM.
7. Foi criado um servico que faz web scraping no X, Twitter, para coleta de dados para a base de dados.
8. Foram criadas "Tools" para o LangChain, permitindo ao modelo de linguagem utilizar essas ferramentas para realizar consultas parametrizadas na base vetorial.
9. Criou-se um fluxo de análise detalhada, permitindo ao modelo gerar relatórios complexos baseados em informações gerais.
10. Por fim, foi implementada a estrutura no backend e frontend para o *upload* de relatórios (ex: PDFs), permitindo que fossem salvos e vetorizados no banco de dados.
11. Avaliação das respostas e ajustes nos prompts.
12. Preparação do roteiro.

O cronograma de desenvolvimento deste projeto difere integralmente daquele apresentado no relatório do Projeto Final I. Conforme justificado na Introdução, o tema do projeto foi alterado, tornando o cronograma original obsoleto.

O novo cronograma foi elaborado para refletir as etapas de desenvolvimento do assistente financeiro, seguindo as fases de implementação descritas acima, e foi executado ao longo do semestre.

5. Projeto e especificação do sistema

5.1 Funcionalidade e estrutura

Para desenvolver esse projeto tem a estrutura do backend e frontend.

- Frontend: A interface do usuário foi desenvolvida em React. É responsável por renderizar a aplicação no navegador, permitir a interação do usuário e consumir os *endpoints* da API do backend. Ele possui duas telas, uma que é gerenciador de ativo com o chat ao lado e outra com upload de relatorio.
- Backend: O *backend* é o componente central do sistema, responsável por toda a lógica de negócios, gerenciamento de dados e orquestração do assistente de IA. Foi desenvolvido em Python utilizando o framework FastAPI para a criação da API.

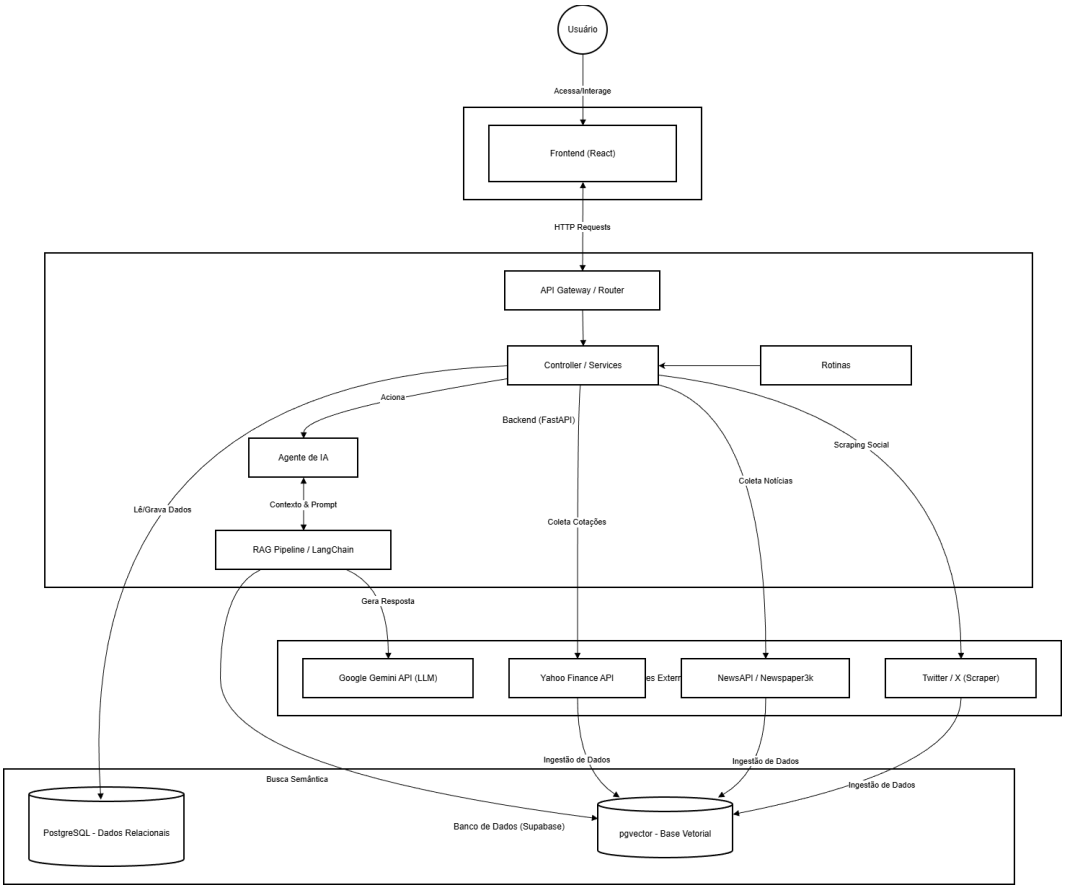

Figura 1 - Diagrama de sistemas

O backend é organizado em módulos funcionais que lidam com responsabilidades específicas:

Gerenciamento de Portfólios: O sistema utiliza um banco de dados para gerenciar os dados dos usuários e suas carteiras. Para o gerenciamento de ativos, foram criados dois modelos de dados principais:

- **Assets:** Armazena todos os ativos que podem ser comprados e vendidos na plataforma.
- **Position:** Armazena a posição de um usuário em um determinado ativo (quantidade e preço médio). Os *endpoints* e *schemas* (formatos de dados) permitem que o *frontend* realize as operações para que um usuário possa "comprar" e "vender" ativos em sua carteira simulada.

Coleta e Processamento de Dados (Serviços): Um conjunto de serviços é responsável por alimentar o sistema com dados externos:

- **price_updater:** Utiliza a API do **yfinance** para atualizar os preços dos ativos cadastrados no banco de dados.
- **get_news:** Interage com as APIs **newsapi** e **newspaper** para buscar as notícias mais recentes, retornando uma resposta estruturada (título, conteúdo, fonte, data).
- **twitter_service:** Realiza *web scraping* de posts do X (Twitter) para tópicos específicos usando Selenium, capturando os últimos posts.
- **reports_service:** Processa relatórios em PDF enviados pelo usuário, extraindo o conteúdo de texto para posterior vetorização.

Módulo de Inteligência Artificial: Este é o núcleo do assistente, responsável pela lógica do LLM, prompts, base vetorial e ferramentas (RAG).

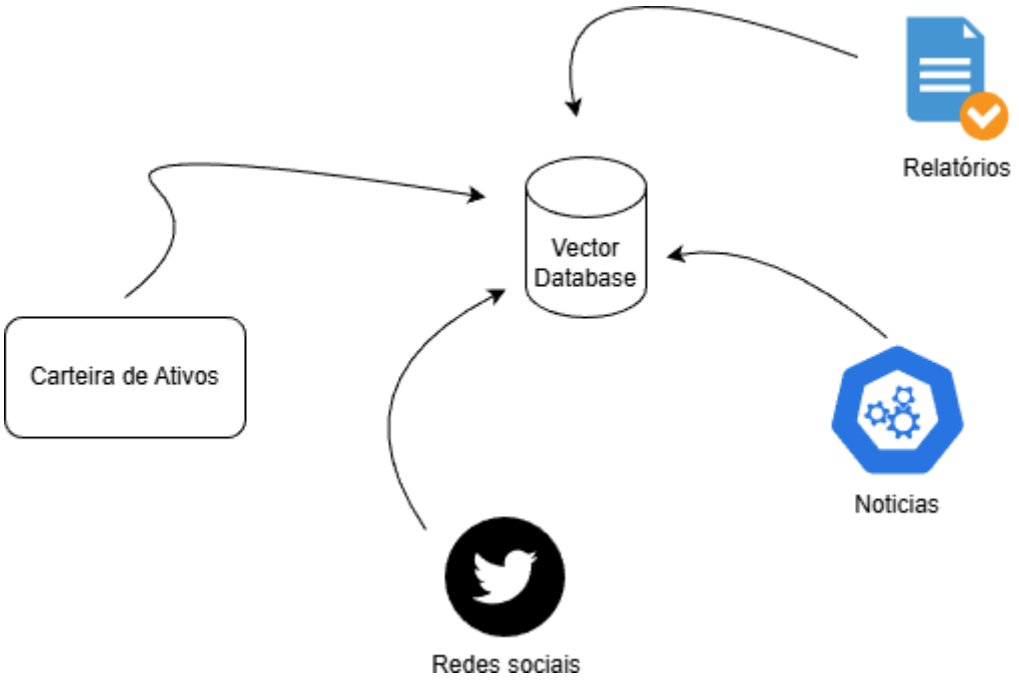

Figura 2 - Fluxo de dados para a base vetorial

- **Base Vetorial:** Os dados coletados (notícias, posts, relatórios) são processados, "quebrados" em *chunks* e armazenados na base vetorial.

  O sistema implementa um *pipeline* de ingestão onde dados não estruturados (notícias, relatórios) são segmentados em *chunks* para granularidade fina. Cada segmento é convertido em um vetor (*embedding*) e indexado, permitindo que o mecanismo de RAG localize trechos semanticamente relevantes com alta precisão através de cálculos de similaridade vetorial.

```
Pipeline de Ingestão de Dados (RAG)
Entrada: Conjunto de Dados Brutos D (Notícias, Posts,
Relatórios PDF)
Saída: Base de Dados Vetorial V atualizada

Para cada documento d em D faça:
    texto_limpo <- LimparTexto(d.conteudo)
    metadados <- ExtrairMetadados(d) // ex: data, fonte,
ticker

    # Divide o texto em segmentos menores de tamanho fixo
    lista_chunks <- DividirTexto(texto_limpo,
tamanho_chunk=1000, sobreposicao=200)

    Para cada chunk c em lista_chunks faça:
        # Utiliza o modelo de embedding
        vetor <- GerarEmbedding(c)
```

```
        # Salva o vetor, o conteúdo original e os metadados
na base vetorial
        V.inserir({
            "id": GerarUUID(),
            "vector": vetor,
            "content": c,
            "metadata": metadados
        })
    Fim Para
Fim Para
```

- **Fluxo de Chat:** O *endpoint* /chat orquestra a interação. Ao receber uma mensagem, o fluxo é acionado, passando o histórico da conversa e os ativos da carteira do usuário como contexto, além do uso das tools para a busca no banco vetorial.
- **Ferramentas (Tools):** O modelo de linguagem utiliza um conjunto de ferramentas RAG para realizar buscas contextualizadas na base vetorial, cada uma com uma função:
    - **vector_search_daily_tool:** Busca notícias dos dois últimos dias.

```
Algoritmo: Busca de Notícias Recentes (Daily Tool)
Entrada: Consulta do Usuário (Q)
Saída: Contexto de Notícias (C)

Início
    DataLimite <- DataAtual() - 2 dias

    VetorConsulta <- ModeloEmbedding.gerar(Q)

    Resultados <- BancoVetorial.buscar(
        vetor_similaridade: VetorConsulta,
        filtro_metadata: {
            "type": "news",
            "date": { "$gte": DataLimite } # Maior ou
igual a 2 dias atrás
        },
        limite_top_k: 20
    )

    C <- ""
    Para cada doc em Resultados faça:
        C <- C + "Título: " + doc.titulo +
"\nConteúdo: " + doc.conteudo + "\n"

    Retornar C
Fim
```

- **vector_search_social_tool:** Busca informações de redes sociais dos últimos dias.

```
Algoritmo: Busca de Sentimento Social (Social Tool)
Entrada: Consulta do Usuário (Query)
Saída: Lista de Postagens Sociais

Início
    DataHoje <- DataAtual(UTC)
    DataOntem <- DataHoje - 1 dia
    DocumentosEncontrados <- ListaVazia()

    # Busca focada na coleção de redes sociais nas
últimas 48h
    Para cada data em [DataHoje, DataOntem] faça:
        DocsDia <- BancoVetorial.buscar_similaridade(
            consulta: Query,
            k: 10,
            filtro: { "created_at": data, "type":
"social" }
        )
        Adicionar DocsDia em DocumentosEncontrados
    Fim Para

    # Prioriza posts mais recentes para análise de
sentimento
    Ordenar DocumentosEncontrados por 'created_at'
(Decrescente)

    Retornar Formatar(DocumentosEncontrados)
Fim
```

- **vector_search_report_tool:** Busca informações nos relatórios financeiros enviados pelo usuário.

```
Algoritmo: Busca em Relatórios Financeiros
Entrada: Consulta (Query), Ticker (Opcional)
Saída: Trechos de Relatórios PDF

Início
    Filtro <- { "type": "report" }

    Se Ticker não for Nulo então:

        Adicionar { "query_source": Ticker } ao
Filtro
    Fim Se

    Documentos <- BancoVetorial.buscar_similaridade(
        consulta: Query,
        k: 10,
        filtro: Filtro
    )
```

```
    Ordenar Documentos por data de criação

    Retornar Formatar
Fim
```

- **vector_search_general_tool:** Realiza uma busca geral na base, com parâmetros de *ticker* e *query*.

```
Algoritmo: Busca Vetorial Geral
Entrada: Consulta (Query), Ticker (Opcional)
Saída: Lista de Resultados Formatada

Início
    Filtro <- {} # Inicia vazio

    Se Ticker for fornecido então:
        Filtro["query_source"] <- Ticker
    Fim Se

    Tente:
        # Recupera os 50 documentos mais similares
semanticamente
        DocumentosBrutos <-
BancoVetorial.buscar_similaridade(
            consulta: Query,
            k: 50,
            filtro: Filtro
        )

        # Garante que, dentre os relevantes, os mais
novos apareçam primeiro
        Ordenar DocumentosBrutos por data de criação
('created_at') Decrescente

        Resultado <- ListaVazia()
        Para cada doc em DocumentosBrutos faça:
            Item <- {
                "Conteúdo": doc.page_content,
                "Título": doc.metadata.get('title',
'N/A'),
                "Fonte/URL":
doc.metadata.get('source', 'N/A')
            }
            Adicionar Item a Resultado
        Fim Para

        Retornar Resultado

    Capturar Erro (e):
        Registrar Log de Erro (e)
        Retornar ListaVazia()
Fim
```

- **get_ticker_info_tool:** Busca informações adicionais de um ativo na API do yfinance.

```
Algoritmo: Consulta de Dados de Mercado
Entrada: Símbolo do Ativo (Ticker)
Saída: Perfil Financeiro e Cotação

Início
    TickerNormalizado <- Ticker.maiusculo()

    Info <- YFinance.obter_info(TickerNormalizado)

    Se (Info estiver vazio) OU (Info.quoteType ==
'NONE') então:
        TickerBR <- TickerNormalizado + ".SA"
        Info <- YFinance.obter_info(TickerBR)

        Se (InfoBR estiver vazio) então:
            Retornar "Erro: Informação não encontrada
para " + Ticker
        Fim Se
    Fim Se

    Moeda <- (Se Info.currency == 'BRL' ou '.SA' no
ticker) ? "R$" : "$"

    Perfil <- ConstruirString({
        "Preço Atual": Info.regularMarketPrice,
        "Min/Max 52 Semanas": Info.fiftyTwoWeekRange,
        "Valor de Mercado": Info.marketCap,
        "P/L (Preço/Lucro)": Info.trailingPE,
        "P/VP (Preço/Valor Patrimonial)":
Info.priceToBook,
        "ROE (Retorno sobre Patrimônio)":
Info.returnOnEquity,
        "Dividend Yield": Info.dividendYield,
        "Beta (Volatilidade)": Info.beta,
        "Resumo do Negócio": Info.longBusinessSummary
    })

    Retornar Perfil
Fim
```

Modelo de dados: Para suportar as funcionalidades do assistente, um modelo de dados relacional foi implementado. O modelo é dividido em duas partes principais: banco de dados relacional e vetorial.

Modelos de Gerenciamento de Portfólio:

Estes modelos definem a relação entre os usuários e os ativos que eles possuem.

Tabela `users` (Usuários) Responsável pelo armazenamento das credenciais e identificação dos usuários do sistema.

- **`id` (Integer, PK):** Identificador único do usuário e chave primária da tabela.
- **`email` (String, Unique):** Endereço de e-mail do usuário, que deve ser único no sistema e é utilizado para o login.
- **`password` (String):** Armazena o *hash* da senha do usuário, garantindo a segurança das credenciais (a senha real não é salva).
- **`positions` (Relationship):** Campo de relacionamento virtual (ORM) que conecta o usuário à lista de todas as posições de ativos que ele possui.

Tabela `assets` (Ativos) Armazena o catálogo de ativos financeiros disponíveis para negociação e análise no sistema.

- **`id` (Integer, PK):** Identificador único do ativo e chave primária.
- **`ticker` (String, Unique):** O código de negociação do ativo (ex: "PETR4", "AAPL"), que serve como índice único para buscas.
- **`name` (String):** O nome completo da empresa ou fundo (ex: "Petróleo Brasileiro S.A.").
- **`current_market_price` (Decimal 10,2):** O preço de mercado mais recente do ativo, que é atualizado periodicamente pelo serviço `price_updater`.
- **`market` (String):** Indica o mercado de origem do ativo, como "B3" (Brasil) ou "NASDAQ" (EUA).

Tabela `positions` (Posições) Esta é a tabela associativa que materializa a carteira de investimentos, vinculando usuários aos ativos que eles compraram.

- **`id` (Integer, PK):** Identificador único do registro da posição.
- **`quantity` (Integer):** O número total de ações ou cotas que o usuário detém daquele ativo.
- **`average_price` (Decimal 10,2):** O preço médio de aquisição do ativo. Este valor é recalculado automaticamente pelo sistema a cada nova operação de compra realizada pelo usuário.
- **`owner_id` (Integer, FK):** Chave estrangeira que referencia o `User` proprietário da posição.
- **`asset_id` (Integer, FK):** Chave estrangeira que referencia o `Asset` correspondente.

Tabela `reports` (Relatórios) Gerencia os metadados dos documentos PDF carregados pelos usuários para a análise via RAG.

- **`id` (String, UUID):** Identificador único universal do relatório.
- **`ticker` (String):** O código do ativo ao qual o relatório se refere (ex: "VALE3"), utilizado para filtrar o contexto durante a geração de respostas.
- **`filename` (String):** O nome original do arquivo enviado (ex: "Balanço_Q3_2025.pdf").

- **`file_date` (Date):** A data de referência ou publicação do relatório, informada pelo usuário no momento do upload.
- **`upload_date` (DateTime):** O carimbo de data e hora (*timestamp*) exato de quando o arquivo foi processado pelo sistema.
- **`chunks_count` (Integer):** Indica em quantos segmentos (*chunks*) o texto do documento foi dividido para ser armazenado na base vetorial.

Além disso, também tem a tabela de “documents" que armazena os vetores dos dados e o conteúdo.

Para suportar as funcionalidades do assistente, um modelo de dados relacional foi implementado. A plataforma escolhida para hospedar a infraestrutura de dados foi o Supabase, que fornece de forma unificada tanto o banco de dados PostgreSQL (para os dados relacionais) quanto a funcionalidade de banco vetorial (pgvector), essencial para armazenar os *embeddings* e viabilizar o RAG.

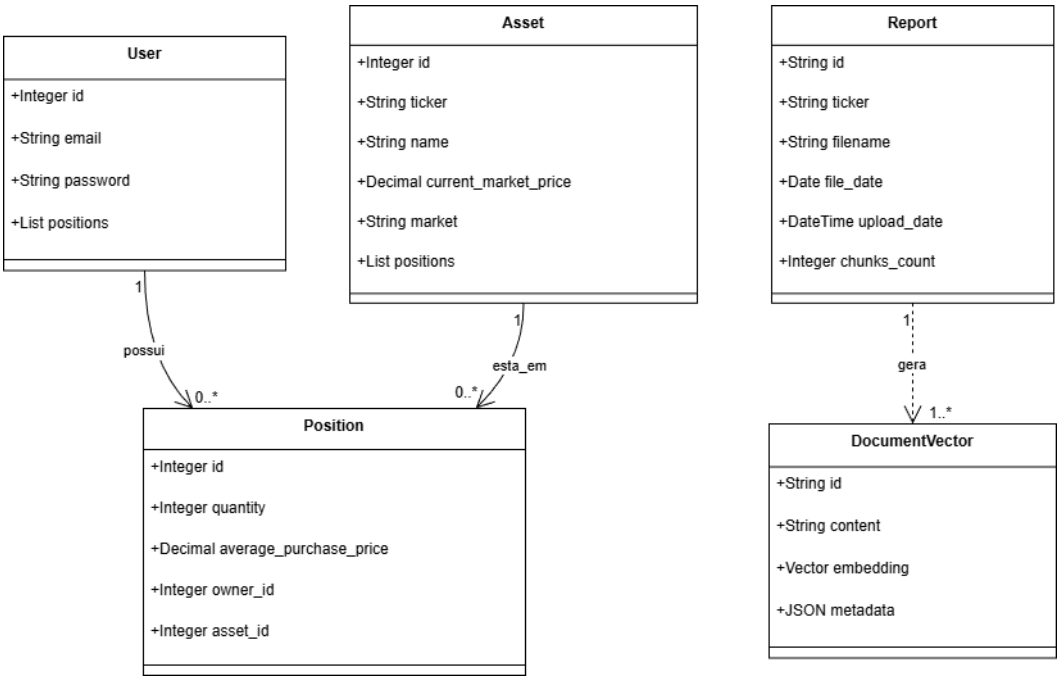

Figura 3 - Diagrama de Classe

5.2 Características marcantes e contribuições

A principal contribuição do trabalho é a implementação de uma arquitetura de contextualização híbrida. A maioria dos assistentes genéricos opera apenas com dados públicos.

A solução desenvolvida, no entanto, é capaz de combinar, em um único prompt, dados de fontes radicalmente diferentes: **Dados Privados do Usuário**, como a carteira de ativos; **Dados de Mercado em Tempo Real**, como cotações consultadas via API; e **Dados Públicos**, como notícias recentes, relatorios e o sentimento de redes sociais.

Essa capacidade de fundir dados privados, em tempo real e públicos permite que o assistente forneça análises hiper-personalizadas, que vão muito além do que um LLM genérico poderia oferecer.

A segunda característica marcante é a **autonomia do assistente através de "Ferramentas"**. O sistema não utiliza apenas uma busca vetorial genérica. O uso de um LLM orquestrado com múltiplas "ferramentas" (como vector_search_daily_tool, vector_search_report_tool, get_ticker_info_tool, etc.) permite que o modelo decida ativamente quais informações buscar e quais fontes consultar com base na pergunta do usuário. Isso torna a análise mais dinâmica e inteligente, pois o LLM passa a "raciocinar" sobre *como* encontrar a resposta, em vez de apenas responder com base em um contexto pré-definido.

5.3 Descrição de uso

Controle de Portfólio O módulo de gestão de carteira permite que o usuário registre suas operações de compra e venda de ativos. A interface apresenta uma visão consolidada das posições atuais, atualizando automaticamente os preços de mercado e calculando a rentabilidade em tempo real. Esta funcionalidade serve como a base de dados "privada" que o assistente utilizará para personalizar as análises.

Chat Interativo (Análise Sob Demanda) A principal forma de interação é o *chat*, onde o usuário pode realizar consultas em linguagem natural, sem a necessidade de comandos rígidos. O usuário pode solicitar, por exemplo, a correlação entre uma notícia recente e um ativo de sua carteira, ou pedir um resumo de um relatório PDF que acabou de enviar. O assistente processa a solicitação e retorna uma resposta textual fundamentada, citando as fontes encontradas (notícias ou documentos) para garantir a transparência da informação.

Análise de Portfólio Para uma visão gerencial rápida, o sistema oferece o modo "Análise de Portfólio". Diferente do chat, esta funcionalidade não exige uma pergunta específica. Ao ser acionada, ela gera automaticamente um *dashboard* visual organizado em *cards*. O usuário visualiza imediatamente os KPIs financeiros (como Lucro Total), alertas de risco (como alta concentração em um setor) e um resumo do sentimento das redes sociais sobre seus ativos, permitindo uma leitura dinâmica da saúde da carteira em uma única tela.

5.4 Engenharia de requisitos e modelagem

Para formalizar o desenvolvimento do sistema, foi realizada a modelagem do domínio e o levantamento de requisitos, conforme descrito a seguir.

Cenário: O ambiente de investimento é caracterizado por um investidor individual que deseja centralizar a gestão de sua carteira e obter análises rápidas. Este investidor possui ativos em custódia (ações) e recebe periodicamente relatórios financeiros em PDF. Ele necessita de um sistema onde possa registrar suas posições, fazer upload desses relatórios e consultar um assistente inteligente que tenha acesso tanto aos dados de mercado em tempo real (notícias/cotações) quanto aos seus dados privados, para obter *insights* personalizados que um LLM genérico não poderia fornecer.

Atores do Sistema:

- **Investidor (Usuário Final):** Responsável por gerenciar a carteira, enviar documentos e interagir com o chat.
- **Agente de IA (Sistema):** Ator interno responsável por orquestrar as ferramentas de busca e gerar as respostas.

Requisitos Funcionais (RF):

- **[RF01] Gerenciamento de Portfólio:** O sistema deve permitir ao usuário adicionar, remover e visualizar ativos, calculando automaticamente o preço médio e o valor total.
- **[RF02] Ingestão de Documentos:** O sistema deve processar arquivos PDF, extrair texto, vetorizar o conteúdo e armazená-lo para busca semântica.
- **[RF03] Coleta de Dados Externos:** O sistema deve buscar autonomamente cotações (via *yfinance*) e notícias recentes (via APIs) para manter a base de conhecimento atualizada.
- **[RF04] Chat Contextual (RAG):** O sistema deve fornecer uma interface de *chat* capaz de responder perguntas utilizando o contexto (Carteira + Notícias + Relatórios).

Diagramas de Modelagem: A estrutura de classes e o relacionamento entre as entidades (Usuário, Ativo, Posição, Relatório) foram formalizados no Diagrama de Classes apresentado na **Figura 3**.

6. Implementação e avaliação

Esta etapa visa validar se o sistema desenvolvido atende aos requisitos funcionais estabelecidos na Seção 5, bem como avaliar a eficácia da arquitetura RAG utilizando as métricas de avaliação fundamentadas na Seção 2.1.

6.1 Planejamento e Métricas de Validação

A estratégia de validação adotou uma abordagem de testes de caixa-preta baseados em cenários de uso reais. Para quantificar o sucesso dos testes, foram aplicados os critérios:

- **Conformidade Funcional:** Aplicada à lógica determinística (gestão de portfólio), onde o resultado obtido deve ser matematicamente idêntico ao esperado.
- **Relevância de Recuperação e Acurácia:** Aplicadas ao componente de IA. Avalia-se se o sistema recuperou vetores com alta similaridade semântica e se a resposta gerada pelo LLM atendeu à intenção do *prompt* sem alucinações (acurácia binária)

6.2 Resultado dos testes funcionais

Os testes foram executados manualmente e os resultados consolidados na tabela, que relaciona o cenário, o critério de aceitação e o *status* final.

ID	Módulo	Cenário de Teste	Critério de Aceite (Resultado Esperado)	Resultado Obtido (Validação)	Status
CT 01	**Gestão (RF01)**	**Cálculo de Preço Médio:** Compra de 100 ações a R$ 40,00 seguida de compra de 100 a R$ 42,00.	O sistema deve recalcular o PM para exatamente **R$ 41,00**.	O banco de dados foi atualizado com PM = 41.00. Conformidade funcional validada.	✅ Aprovado
CT 02	**Ingestão (RF02)**	**Processamento de PDF:** Upload de relatório financeir	O texto deve ser vetorizado e indexado com a tag `type: report`.	Documento armazenado na base vetorial e recuperável via busca de	✅ Aprovado

		o complexo.		similaridade.	
CT 03	**IA / RAG (RF04)**	**Roteamento de Ferramentas:** Pergunta: *"Qual o sentimento sobre o Bitcoin no Twitter?"*	O agente deve acionar *apenas* a `vector_search_social_tool`.	O *log* confirmou o acionamento exclusivo da ferramenta social.	✅ Aprovado
CT 04	**IA / RAG (RF04)**	**Contextualização Híbrida:** Pergunta: *"Com base no relatório da PETR4 e nas notícias..."*	O agente deve acionar `report_tool` E `daily_tool` e sintetizar ambas.	Ambas as ferramentas foram chamadas. A resposta citou dados do PDF e da notícia recente.	✅ Aprovado
CT 05	**IA / RAG (RF04)**	**Geração Estruturada:** Solicitação de "Análise de Portfólio".	O retorno deve ser um JSON estrito para renderização de *dashboard*.	O modelo gerou um JSON válido, permitindo a visualização dos *cards*.	✅ Aprovado

CT 06	**Gestão (RF03)**	**Coleta de Dados Externos:** Execução da rotina `price_updater`.	Os preços na tabela `Assets` devem ser atualizados.	Preços atualizados conforme cotação da B3 em tempo real.	✅ Aprovado

Tabela 1 - Testes e Resultados

6.3 Resultados Obtidos

Os testes funcionais validaram que todos os componentes principais do sistema (gestão de portfólio, *pipeline* de ingestão de dados e orquestração do chat) estão operando em conformidade com as especificações. A análise dos cenários de RAG (CT03 a CT05) confirmou a Acurácia da Resposta, validando a capacidade do agente de interpretar a intenção do usuário e realizar o "roteamento semântico" adequado, selecionando dinamicamente as ferramentas de busca corretas (dados de mercado, notícias ou documentos privados).

Notavelmente, a validação do cenário de contextualização confirmou a hipótese de que a arquitetura RAG permite a um LLM "raciocinar" sobre dados heterogêneos, recuperando o contexto apropriado para formular respostas fundamentadas.

No entanto, foi identificada uma limitação na métrica de Relevância de Recuperação em casos de ambiguidade. Observou-se que perguntas excessivamente genéricas (ex: *"Como está o mercado hoje?"*) tendem a enviesar o modelo para a busca de notícias (vector_search_daily_tool), falhando ocasionalmente em acionar ferramentas complementares, como a get_ticker_info_tool, para cruzar esses dados com os ativos específicos da carteira do usuário. Embora as respostas geradas sejam factualmente corretas, elas apresentam menor grau de personalização nesses casos específicos.

6.4 Comentários sobre a implementação

Durante o desenvolvimento do projeto, desafios técnicos relacionados à integração de dados e à confiabilidade da IA foram identificados e mitigados através de ajustes arquiteturais.

No contexto da **integração de dados**, o limite de requisições das APIs de notícias (newsapi) e dados financeiros (yfinance) impôs restrições de latência. A solução técnica adotada foi a implementação de uma arquitetura de rotinas nos serviços price_updater e get_news, permitindo a operação em intervalos controlados em vez de tempo real absoluto. Adicionalmente, para contornar a limitação de conteúdo da newsapi (que fornece apenas fragmentos), integrou-se a biblioteca newspaper, permitindo a extração do texto completo das URLs de origem.

Para a coleta de dados sociais, o escopo foi definido tecnicamente focando na plataforma X (Twitter). A alta complexidade de manutenção de *scripts* de *web scraping* em múltiplas plataformas, devido a medidas agressivas de *anti-bot* e renderização dinâmica, levou à decisão de concentrar a análise em uma única fonte robusta e representativa. Esta abordagem garantiu a estabilidade operacional do sistema sem comprometer a validade da análise de sentimento.

Quanto à **confiabilidade da IA**, o principal desafio foi a ambiguidade na seleção de contexto pelo LLM. A natureza probabilística do modelo exigiu a implementação de *prompts* com instruções estritas para garantir a formatação de saída. Embora o uso de ferramentas especializadas (*Tools*) tenha garantido o roteamento correto na maioria dos casos, identificou-se que a adoção futura de técnicas de *Chain-of-Thought* poderia refinar ainda mais a interpretação de consultas complexas.

7. Considerações finais

O ponto de partida deste trabalho foi o desafio imposto pelo volume massivo e pela complexidade de dados no mercado financeiro, que frequentemente dificulta a tomada de decisão ágil por investidores individuais. Embora os Modelos de Linguagem apresentem capacidades avançadas de interpretação, sua eficácia no domínio financeiro mostrou-se restringida por limitações intrínsecas, notadamente a geração de informações incorretas (alucinações) e a ausência de conhecimento atualizado. Identificou-se, portanto, a oportunidade estratégica de aplicar a arquitetura RAG como solução para unir a fluência dos modelos generativos com a precisão de dados externos.

Nesse contexto, este projeto alcançou êxito em atingir seus objetivos, culminando no desenvolvimento de um protótipo funcional de assistente financeiro. O trabalho validou a aplicação do modelo "Retrieval-Augmented Generation" para um caso de uso notavelmente complexo, que exigiu a integração de fontes de dados privados do usuário com dados de mercado em tempo real e dados públicos (redes sociais, relatórios e notícias), preenchendo a lacuna identificada inicialmente.

Esta seção final consolida a análise do projeto, abordando as contribuições técnicas e de usuário, os aprendizados obtidos durante o desenvolvimento, as limitações identificadas no protótipo e as oportunidades para trabalhos futuros.

7.1 Contribuições do trabalho

A principal contribuição deste projeto para os usuários é um protótipo funcional que permite a qualquer pessoa o acesso a análises financeiras complexas, seja ela uma pessoa experiente ou não no mercado financeiro.

O sistema permite que um investidor individual, mesmo sem experiência técnica avançada, combine e sintetize dados de mercado em tempo real, notícias, sentimento de redes sociais e relatórios privados em uma única interface conversacional.

Para a comunidade técnica e acadêmica, o trabalho demonstra uma implementação prática e viável de uma arquitetura RAG.

Esta arquitetura prova a eficácia de consumir dados de fontes (APIs, *web scraping* e uploads de documentos) para contextualizar um LLM, aumentando a relevância e a confiabilidade de suas respostas no domínio financeiro.

7.2 Aprendizado Obtido

O desenvolvimento deste trabalho proporcionou um aprendizado prático significativo em tecnologias de IA de ponta. Foi possível aprofundar os conhecimentos na aplicação da arquitetura RAG, no uso e gerenciamento de bases vetoriais e na orquestração de LLMs com ferramentas, especificamente utilizando LangChain e a API do Gemini.

Além disso, o projeto permitiu vivenciar os desafios práticos da engenharia de dados. As dificuldades enfrentadas, como a instabilidade da coleta de dados via web scraping e a necessidade de gerenciar os limites e as inconsistências no consumo de múltiplas APIs externas, foram fundamentais para o aprendizado de como construir sistemas resilientes.

7.3 Limitações e Recomendações

A principal limitação identificada no sistema desenvolvido é a dependência da clareza da pergunta do usuário, como visto nos testes. Respostas ambíguas podem levar o assistente a não acionar as ferramentas corretas, gerando respostas menos completas.

Refletindo sobre o processo, caso o projeto fosse iniciado agora, seria dedicado um tempo significativamente maior ao aprimoramento das técnicas de engenharia de prompt (PROMPTING GUIDE, 2025).

Um “prompt engineering” mais complexo e robusto é fundamental para garantir que o modelo de linguagem interprete as intenções do usuário de forma mais assertiva e funcione de maneira apropriada, mesmo com perguntas menos estruturadas.

7.4 Oportunidades para trabalhos futuros

O protótipo desenvolvido estabelece uma base sólida, mas abre diversas oportunidades para investigações e aprimoramentos futuros. Recomenda-se a exploração das seguintes frentes de expansão:

- **Expansão e Diversificação das Fontes de Dados:** O projeto pode evoluir para integrar um ecossistema de dados mais vasto. Sugere-se a conexão com **fóruns especializados** (como Reddit e comunidades de investidores) para uma análise de sentimento mais granular. Além disso, a integração direta com fontes de dados regulatórios oficiais permitiria o acesso a demonstrações financeiras padronizadas e fatos relevantes em sua fonte primária, reduzindo a dependência de notícias de terceiros.
- **Engenharia de Prompt Avançada e Agentes Autônomos:** Conforme identificado nas limitações, há espaço para a implementação de técnicas avançadas de *Prompt Engineering*. Um trabalho futuro poderia focar na evolução do assistente para uma arquitetura de **Agentes Autônomos**, onde o modelo não apenas seleciona uma ferramenta, mas planeja e executa uma sequência de passos lógicos para resolver problemas complexos de análise fundamentalista, decompondo uma pergunta em múltiplas sub-tarefas de pesquisa.
- **Comparação com Fine-Tuning e Abordagens Híbridas:** Uma evolução natural da pesquisa seria a aplicação de técnicas de *fine-tuning* (ajuste fino) em modelos menores e abertos (como Llama ou Mistral) utilizando *datasets* específicos do mercado financeiro. O objetivo seria comparar o desempenho, a latência e o custo operacional dessa abordagem frente à arquitetura RAG implementada. Adicionalmente, poderia ser investigada uma abordagem híbrida, onde um modelo afinado é utilizado em conjunto com o RAG (que fornece os dados atualizados), buscando propor uma junção de ideias para melhorar o desempenho.

8. Referências

FASTAPI. **FastAPI Documentation**. Disponível em: https://fastapi.tiangolo.com/. Acesso em: 05 set. 2025.

FEBRABAN. **Maioria dos brasileiros admite entender pouco de educação financeira, mostra pesquisa**. Publicado em: Febraban Notícias. 21 jul. 2025. Disponível em: https://portal.febraban.org.br/noticia/4324/pt-br/. Acesso em: 02 nov. 2025.

LANGCHAIN. **LangChain Documentation**. Disponível em: https://docs.langchain.com/. Acesso em: 04 set. 2025.

LIU, Chenghao et al. Large Language Models and Sentiment Analysis in Financial Markets: A Review, Datasets, and Case Study. **IEEE Access**, v. 12, p. 134041-134056, 2024. Disponível em: https://ieeexplore.ieee.org/document/10638546. Acesso em: 02 set. 2025.

NOGUEIRA, Bruna. **O Impacto da Inteligência Artificial na Análise de Dados**. Publicado em: Clarify. 4 jun. 2024. Disponível em: https://clarify.com.br/blog/o-impacto-da-inteligencia-artificial-na-analise-de-dados/ . Acesso em: 02 nov. 2025.

NÚCLEA. **Mais do que hype: aplicações reais de IA Generativa no mercado financeiro**. Publicado em: Blog Núclea. 6 jun. 2025. Disponível em: https://www.nuclea.com.br/ia-generativa-no-mercado-financeiro/. Acesso em: 02 nov. 2025.

PROMPTING Guide. Disponível em: https://www.promptingguide.ai/pt. Acesso em: 20 set. 2025.

QI, Ruoyu. Financial News Sentiment Analysis and Market Sentiment Prediction Based on Large Language Models. **Journal of Computer, Signal, and System Research**, v. 2, n. 6, p. 23-29, 2025. Disponível em: https://www.gbspress.com/index.php/JCSSR/article/view/473. Acesso em: 02 set. 2025.

SELENIUM. **Selenium with Python Documentation**. Disponível em: https://selenium-python.readthedocs.io/. Acesso em: 04 set. 2025.

WU, Ruoxu. Portfolio Performance Based on LLM News Scores and Related Economical Analysis. **SSRN Electronic Journal**, 2024. Disponível em: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4709617. Acesso em: 02 set. 2025.

YANG, Jie et al. Dynamic Hedging Strategies in Derivatives Markets with LLM-Driven Sentiment and News Analytics. **arXiv preprint arXiv:2504.04295**, 2025. Disponível em: https://arxiv.org/abs/2504.04295. Acesso em: 02 set. 2025.

YFINANCE: 10 ways to get stock data with python. Publicado em: Medium. Disponível em: https://medium.com/@kasperjuunge/yfinance-10-ways-to-get-stock-data-with-python-6677f49e8282. Acesso em: 04 set. 2025.

ZHANG, Boyu et al. Enhancing Financial Sentiment Analysis via Retrieval Augmented Large Language Models. **arXiv preprint arXiv:2310.04027**, 2023. Disponível em: https://arxiv.org/abs/2310.04027. Acesso em: 02 set. 2025.