UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO
MESTRADO ACADÊMICO EM COMPUTAÇÃO

PEDRO PAULO DE SALES FARIAS

EXTRAÇÃO DE REQUISITOS DE SOFTWARE A PARTIR DE COMENTÁRIOS DE APLICATIVOS POR GRANDES MODELOS DE LINGUAGEM E ORQUESTRAÇÃO DE AGENTES

QUIXADÁ
2025

PEDRO PAULO DE SALES FARIAS

EXTRAÇÃO DE REQUISITOS DE SOFTWARE A PARTIR DE COMENTÁRIOS DE APLICATIVOS POR GRANDES MODELOS DE LINGUAGEM E ORQUESTRAÇÃO DE AGENTES

Dissertação apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal do Ceará, como requisito parcial à obtenção do título de Mestre em Computação. Área de concentração: Ciência da Computação.

Orientador: Prof. Dr. Marcos Antonio De Oliveira.

QUIXADÁ

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

F238e Farias, Pedro Paulo de Sales.
Extração de requisitos de software a partir de comentários de aplicativos por grandes modelos de linguagem e orquestração de agentes / Pedro Paulo de Sales Farias. – 2025.
68 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Campus de Quixadá, Programa de Pós-Graduação em Computação, Quixadá, 2025.
Orientação: Prof. Dr. Marcos Antonio De Oliveira.

1. Engenharia de requisitos. 2. Comentários de usuários. 3. Agentes de software. 4. Orquestração de agentes. 5. Extração de requisitos. I. Título.

CDD 005

PEDRO PAULO DE SALES FARIAS

EXTRAÇÃO DE REQUISITOS DE SOFTWARE A PARTIR DE COMENTÁRIOS DE APLICATIVOS POR GRANDES MODELOS DE LINGUAGEM E ORQUESTRAÇÃO DE AGENTES

Dissertação apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de concentração: Ciência da Computação.

Aprovada em: 27/11/2025.

BANCA EXAMINADORA

_____

Prof. Dr. Marcos Antonio De Oliveira (Orientador)
Universidade Federal do Ceará (UFC)

_____

Prof. Dr. Enyo José Tavares Gonçalves
Universidade Federal do Ceará (UFC)

_____

Prof. Dr. Gustavo Augusto Lima De Campos
Universidade Estadual do Ceará (UECE)

A Deus.

Aos meus pais, Isaura Sales e Antônio Farias.

AGRADECIMENTOS

Ao Prof. Dr. Marcos Antonio de Oliveira, pela excelente orientação.

Aos professores participantes da banca examinadora, Prof. Dr. Enyo José Tavares Gonçalves e Prof. Dr. Gustavo Augusto Lima de Campos, pelo tempo, pelas valiosas colaborações e sugestões.

Aos colegas da turma de mestrado, pelas reflexões, críticas e sugestões recebidas.

“A integração entre LLMs e sistemas multiagente inaugura uma nova geração de ferramentas para Engenharia de Requisitos, ampliando a capacidade de análise de grandes corpus textuais e reduzindo significativamente o esforço manual.”

(He et al., 2023).

RESUMO

O trabalho investiga a extração de requisitos de software com recursos de modelos LLM a partir de comentários de usuários publicados em lojas de aplicativos, considerando os desafios clássicos da Engenharia de Requisitos, como a ambiguidade, duplicidades de texto e variações linguísticas. É sabido que os sistemas computacionais modernos têm alcance global, o que amplia as diferenças culturais e semânticas que comprometem a qualidade dos requisitos e elevam custos, retrabalho e riscos ao projeto. Nesse contexto, o estudo apresenta quatro questionamentos principais: a possibilidade de extrair requisitos diretamente de comentários de usuários; a viabilidade de automatizar esse processo por meio de agentes de software; a construção de um workflow de agentes para apoiar a Engenharia de Requisitos; e a qualidade dos requisitos extraídos. Para responder a esses questionamentos, nosso trabalho propõe o uso integrado de modelos de Processamento de Linguagem Natural, Grandes Modelos de Linguagem (LLMs) e técnicas de orquestração de agentes (CrewAI), avaliando sua capacidade de identificar, estruturar e validar requisitos funcionais e não funcionais. Finalmente, destacamos que os avanços recentes em IA abrem caminhos promissores para reduzir esforços manuais, mitigar ambiguidades e aprimorar a sistematização da Engenharia de Requisitos.

Palavras-chave: engenharia de requisitos; comentários de usuários; agentes de software; orquestração de agentes; extração de requisitos.

ABSTRACT

The study investigates the extraction of software requirements using LLM-based resources from user comments published in application store platforms, considering the classical challenges of Requirements Engineering, such as ambiguity, duplicated text and linguistic variation. It is well known that modern computational systems have a global reach, which magnifies cultural and semantic differences, compromising requirement quality and increasing costs, rework, and project risks. In this context, the study addresses four main questions: the feasibility of extracting requirements directly from user comments; the possibility of automating this process through software agents; the construction of an agent workflow to support Requirements Engineering; and the quality of the extracted requirements. To address these questions, our work proposes the integrated use of Natural Language Processing models, Large Language Models (LLMs), and CrewAI agent-orchestration techniques, assessing their ability to identify, structure, and validate functional and non-functional requirements. Finally, we highlight that recent advances in AI offer promising avenues for reducing manual effort, mitigating ambiguities, and enhancing the systematisation of Requirements Engineering.

Keywords: requirements engineering; user comments; software agents; agent orchestration; requirements extraction.

LISTA DE QUADROS

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AM (ML)	Aprendizado de Máquina
APPs	Programa de software para dispositivos móveis ou computadores.
CP	Caixa Preta
CSV	Comma-Separated Values ou Valores Separados por Vírgulas
DSR	Design Science Research
ER	Engenharia de Requisitos
ES	Engenharia de Software
FAISS	Facebook AI Similarity Search
GDPR	Regulamento Geral de Proteção de Dados
IA	Inteligência Artificial
LDA	Latent Dirichlet Allocation
LLMs	Grandes Modelos de Linguagens
MAS	Sistema multiagente
ML	Aprendizado de máquina
PLN	Processamento de Linguagem Natural
PO	Product Owner
QA	Usado para tarefas de Perguntas e Respostas (Question Answering - QA)
RAG	Geração Aumentada de Recuperação
RF	Requisitos Funcionais
RNF	Requisitos não Funcionais
RS	Requisitos de Software
SRS	Especificação de Requisitos de Software (*Software Requirement Specification*).
SVM	Support Vector Machine
TF-IDF	Term Frequency - Inverse Document Frequency
ZS	Zero-Shot
ZSL	Zero-Shot Learning

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização

O desenvolvimento de sistemas computacionais permanece uma atividade complexa, mesmo diante do avanço de tecnologias como o Aprendizado de Máquina (AM) e a Inteligência Artificial (IA). Em muitas organizações, a etapa de Engenharia de Requisitos (ER) continua sendo um gargalo crítico, marcado por falhas de comunicação, ambiguidades de linguagem, incompletude de requisitos e documentação inconsistente. Esses problemas afetam diretamente o custo, o prazo e a qualidade dos produtos de software.

Pressman (2011, p. 126) destaca que grande parte dos insucessos em projetos de software decorre de requisitos mal entendidos ou mal documentados, isto é, de uma compreensão equivocada entre o que o cliente deseja e o que é definitivamente especificado pela equipe técnica. Em projetos modernos, essa dificuldade é agregada pelo contexto globalizado de uso dos sistemas: uma mesma aplicação pode atender usuários em diferentes países, culturas e níveis de letramento digital, o que amplia a diversidade linguística e semântica presente nos artefatos textuais.

Nos ambientes digitais contemporâneos, os usuários manifestam suas percepções por meio de comentários em lojas de aplicativos, fóruns on-line, redes sociais e canais de suporte. Esses textos incluem elogios, críticas, relatos de falhas, sugestões de melhorias e queixas relativas ao desempenho, à usabilidade, à segurança e a outros aspectos de qualidade. Do ponto de vista da ER, tais comentários representam uma fonte rica e dinâmica de dados, frequentemente atualizada e, em muitos casos, mais espontânea do que os relatos coletados em entrevistas formais.

Entretanto, a exploração sistemática desse tipo de dado ainda é limitada na prática. Em geral, equipes de desenvolvimento não dispõem de processos consolidados para transformar grandes volumes de comentários não estruturados em requisitos de software claros, verificáveis e rastreáveis. Além disso, tais textos costumam apresentar características desafiadoras para análise automática: linguagem informal, uso intensivo de gírias e abreviações, erros ortográficos, frases incompletas, emojis, ironias e ambiguidades, entre outros (Bhatia; Singh; Sharma, 2023).

Paralelamente, recentes avanços em Processamento de Linguagem Natural (PLN) e Grandes Modelos de Linguagem (LLMs) têm ampliado significativamente a capacidade de análise automatizada de textos em larga escala. Modelos como GPT, Gemini e outros LLMs baseados em arquiteturas de transformadores conseguem interpretar, resumir, traduzir e extrair textos com alto grau de coerência, tornando-se ferramentas promissoras para apoiar atividades tradicionalmente manuais da Engenharia de Requisitos (Karanikolas et al. 2024; Naveed et al. 2024).

Mais recentemente, observa-se também o surgimento de sistemas multiagente baseados em LLMs, nos quais múltiplos agentes especializados, coordenados por frameworks como o CrewAI, colaboram para executar fluxos de trabalho complexos de forma autônoma ou semi-autônoma. Trabalhos como LLMBA4RS demonstram que agentes orquestrados podem apoiar a especificação de requisitos, a elaboração de histórias de usuários e a geração de cenários de uso, reduzindo ambiguidades e aumentando a padronização dos artefatos (Zhao, 2023).

Nesses cenários, emerge a oportunidade de integrar PLN, LLM, Técnicas de Recuperação de Informação (RAG) e orquestração de agentes e multiagente para extrair, classificar e consolidar requisitos de software diretamente a partir de comentários de usuários retirados de lojas de aplicativos. Essa integração, se bem estruturada e acompanhada de mecanismos de controle (guardrails), pode contribuir para reduzir o esforço manual, ampliar a cobertura de requisitos e proporcionar maior transparência no nosso processo de elicitação.

1.2 O problema da pesquisa

Apesar do potencial descrito, a aplicação prática desses recursos ainda enfrenta lacunas importantes. Dentre elas:

- A ausência de processos sistemáticos que transformem comentários de usuários em requisitos formais alinhados a normas e boas práticas da ER;
- A dificuldade de lidar com textos ruidosos, ambíguos e redundantes, que podem induzir a extração de requisitos inconsistentes ou irrelevantes;
- A escassez de estudos empíricos que combine, em único pipeline, diferentes LLMs e um orquestrador de agentes para realizar a extração e a consolidação de requisitos;
- Os riscos de alucinações e anomalias geradas por LLMs, o que exige a definição de *guardrails* e mecanismos de validação complementar (Ayyamperumal; Ge, 2024;

Dong et al., 2024).

Do ponto de vista da prática, muitas equipes de ER ainda realizam a leitura manual de comentários, selecionando trechos relevantes e, então, reescrevendo-os em requisitos. Esse processo é oneroso, pouco escalável e sujeito a vieses pessoais. Do ponto de vista comercial, industrial e científico, falta compreender melhor:

1 - Em que ponto LLMs e agentes são capazes de identificar, estruturar e classificar requisitos a partir de comentários reais de usuários;

2 - Como combinar modelos de LLMs e um pipeline RAG com LangChain para cobrir diferentes perspectivas de análise;

3 - Como consolidar os resultados desses modelos em um artefato unificado, alinhado ao papel de um Product Owner (PO) em contextos ágeis;

4 - Quais tipos de guardrails são mais adequados para reduzir redundâncias, incoerências e alucinações na extração automática de requisitos.

Assim, o problema de pesquisa que orienta esta dissertação pode ser formulado nos termos que seguem:

Como extrair e consolidar Requisitos de Software (RS), Requisitos Funcionais (RF) e Requisitos não Funcionais (RNF), a partir de comentários de usuários em lojas de aplicativos, utilizando LLMs e orquestração de agentes, de forma a mitigar problemas clássicos da ER, como ambiguidade, redundância e inconsistência?

1.3 As questões da pesquisa

A partir do problema central, este estudo busca responder às seguintes questões de pesquisa:

- QP1 - Quanto à extração: É possível extrair requisitos de software diretamente de comentários de usuários publicados em lojas de aplicativos, utilizando LLMs e técnicas de PLN?
- QP2 - Quanto à automação: Em que medida a orquestração de agentes, por meio de frameworks como o CrewAI, permite automatizar, parcialmente ou integralmente, o processo de elicitação e classificação de requisitos?
- QP3 - Sobre a orquestração de agentes: Como estruturar um workflow multiagente que integre diferentes modelos (Gemini, RAG+LangChain e GPT) para apoiar a ER a partir de dados textuais não estruturados?
- QP4 - Sobre a qualidade dos RS: Qual é a qualidade dos requisitos extraídos, em termos de clareza, completude, ausência de duplicidade e aderência às normas de

requisitos, quando comparados aos critérios de referência da literatura de ER?

- QP5 - Sobre os guardrails: Quais mecanismos de guardrails e de detecção de anomalias são necessários para reduzir alucinações e filtrar requisitos incoerentes ou desalinhados com os comentários originais?

1.4 O objetivo geral

Desenvolver e avaliar uma abordagem com base em PNL, LLM e orquestração de agentes, para extrair, classificar e consolidar RS a partir de comentários de usuários retirados de lojas de aplicativos, mitigando problemas de ambiguidade, redundância e inconsistência na ER.

1.5 Dos objetivos específicos

Para alcançar o objetivo geral, estabelecemos as etapas a seguir:

- Coleta e preparação dos dados
    - Identificar e coletar comentários de usuários em uma loja de APPs.
    - Realizar o pré-processamento textual (limpeza, normalização e remoções de duplicidades), estruturando os dados em arquivo .CSV,
- Estruturação preliminar com PLN e aprendizado em contexto
    - Aplicar técnicas de representação vetorial, por exemplo, Term Frequency Inverse Document Frequency (TF-IDF), para identificar termos de maior relevância semântica no corpus.
    - Utilizar Zero-Shot Learning (ZSL) para classificar os comentários em tópicos preditos. Formando uma base de dados rotulada para os modelos subsequentes.
- Estruturação de requisitos com modelos de LLMs e CrewAI
    - Projetar e implementar um modelo Gemini e CrewAI, no qual um agente gera requisitos a partir de comentários classificados.
    - Projetar e implementar um modelo RAG, LangChain e CrewAI, integrando técnicas de recuperação de contexto à extração de requisitos a partir dos comentários classificados.
    - Projetar e implementar um modelo GPT e CrewAI, no qual um agente gera requisitos a partir dos comentários classificados.
- Fusão dos resultados via PO
    - Definir e implementar um modelo de Product Owner com base em LLM

(Gemini) e em orquestrador de agentes (CrewAI) para unificar os requisitos extraídos pelos três modelos anteriores, removendo duplicidades, refinando as relações e organizando-os por tópicos.
    - Gerar arquivos consolidados (.CSV e .PDF) contendo os comentários originais, respectivos requisitos e tópicos.
- Definição e aplicação de guardrails
    - Especificar e implementar mecanismos de guardrails e deteção de anomalias para reduzir alucinações, requisitos incoerentes e inconsistentes com o corpus original.
    - Avaliar o impacto desses *guardrails* na qualidade e na confiabilidade dos requisitos extraídos.
- Avaliação dos resultados
    - Analisar quantitativamente os requisitos produzidos no que se refere às quantidades geradas por cada modelo aplicado.
    - Realizar avaliações qualitativas quanto à análise de sentimentos dos comentários em relação aos requisitos.
    - Realizar avaliação quanto à clareza, à completude e à consistência.

1.6 Justificativa

A pesquisa justifica-se em três pontos principais: científico, tecnológico e prático. No que se refere ao primeiro ponto, este trabalho contribui para a interseção entre ER, PLN e LLMs, ao propor e avaliar um pipeline multiagente para a extração de requisitos a partir de comentários reais de usuários. Embora existam estudos que exploram LLMs para a classificação de requisitos (Alhoshan; Ferrari; Zhao, 2023; Binkhonain; Alfayez, 2025), ainda são raras as propostas que combinam múltiplos LLMs, técnicas de RAG e um orquestrador de agentes para consolidar os resultados em um modelo de PO.

Com relação ao segundo ponto, a integração de frameworks como LangChain e CrewAI com LLMs de última geração (GPT, Gemini) representa uma oportunidade de explorar arquiteturas modernas de sistemas baseadas em agentes autônomos e em workflows orquestrados. Essa combinação permite testar, em ambiente controlado, diferentes estratégias e a mesma implementação de agentes, para recuperação de conteúdos e aplicação de guardrails (Winland; Syed; Gutowska, 2025; Mavroudis, 2022).

Em relação ao último ponto, as empresas que mantêm aplicações amplamente utilizadas, em especial no setor bancário e de serviços digitais, dispõem de grandes volumes de feedback textual de usuários, muitas vezes subutilizados. Automatizar, mesmo que parcialmente, a transformação desses comentários em requisitos estruturados possibilita:

- Reduzir o esforço manual de análises de requisitos e PO;
- Aumentar a cobertura de requisitos capturados a partir de voz do usuário;
- Antecipar a identificação de problemas recorrentes de usabilidade, desempenho ou confiabilidade;
- Apoiar decisões de priorização de backlogs em contextos ágeis.

Além disso, ao explorar dados públicos de uma loja de aplicativos, este trabalho favorece a reprodutibilidade e a comparação com estudos futuros, contribuindo para a consolidação de um corpo de evidências empíricas sobre o uso de LLMs e de agentes na ER.

1.7 Estrutura da dissertação

Para facilitar a compreensão da abordagem proposta, esta dissertação está organizada da seguinte forma:

Capítulo 1 - Introdução: apresenta a contextualização do tema, o problema de pesquisa, as questões investigadas, os objetivos gerais e específicos, a justificativa e a organização do trabalho.

Capítulo 2 - Fundamentação teórica: discute os conceitos essenciais para o estudo, incluindo ER, PLN, LLMs, ZSL, Sistemas multiagente, CrewAI, técnicas de RAG, *guardrails* e trabalhos relacionados.

Capítulo 3 - Metodologia: descreve em detalhes o desenho metodológico adotado, os critérios de seleção do aplicativo e dos comentários, o pré-processamento e os cinco modelos (Zero-Shot, Gemini+CrewAI, RAG+LangChain+CrewAI, GPT+CrewAI, aplicação dos *guardrails* e PO), ver Figura 1.

Capítulo 4 - Resultados: apresenta e discute os resultados por modelo, distribuição por tópicos, comparação entre LLMs e análises qualitativas (qualidade dos requisitos, exemplos representativos, efeitos dos *guardrails*).

Capítulo 5 - Conclusões e trabalhos futuros: sintetiza as principais contribuições da pesquisa, responde aos questionamentos da pesquisa, discute limitações e aponta direções para trabalhos futuros e para a integração da abordagem a pipelines de desenvolvimento ágil.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Sobre a Engenharia de Requisitos

A Engenharia de Requisitos (ER) é uma área estruturante dentro da Engenharia de Software (ES), responsável por identificar, analisar, especificar, validar e gerenciar as necessidades dos stakeholders para a construção de sistemas (Pressman, 2011). Requisitos representam “as condições ou capacidades que um sistema deve possuir para atender a um propósito” (IEEE, 2024).

A literatura enfatiza que as falhas na ER são uma das principais causas de fracasso em projetos. A comunicação imprecisa entre usuários e desenvolvedores frequentemente resulta em requisitos mal definidos, ambíguos, contraditórios ou incompletos. Em ambientes de grande escala, com alto volume de feedbacks, essa dificuldade se multiplica.

Tradicionalmente, as técnicas de elicitação incluem entrevistas, questionários, reuniões de grupo e observação direta no ambiente ao qual o sistema pertencerá. Entretanto, tais técnicas demandam grande esforço manual, dependem da disponibilidade de stakeholders e, em muitos casos, deixam de capturar problemas reais no que se refere aos domínios do sistema.

Dessa forma, comentários em lojas de APPs emergem como fontes relevantes de informação para apoiar a ER, pois são espontâneos, atualizados continuamente e refletem diretamente a percepção do usuário, incluindo problemas de desempenho, falhas recorrentes e sugestões de melhorias.

2.2 O Processamento de Linguagem Natural (PLN)

O PLN permite que máquinas que “pensam” compreendam e processem linguagem humana de forma significativa. No contexto de ER, o PLN é essencial, pois os requisitos são quase sempre expressos em linguagem natural.

As tarefas importantes incluem:

- Tokenização, lematização e normalização são necessárias para preparar textos ruidosos como os comentários dos usuários;
- Classificação de textos, útil para diferenciar comentários relevantes dos não relevantes;

- Possíveis identificações de novas entidades e eventos, fundamentais para a identificação de funcionalidades e problemas;
- Análise semântica e transformação textual, úteis para reescrever comentários em requisitos formais.

Com o avanço de arquiteturas modernas, PLN tradicional, como TF-IDF, Máquina de Suporte de Vetores (SVM) e Análise Discriminante Linear (LDA), coexiste com abordagens mais recentes baseadas em redes neurais e modelos pré-treinados em larga escala.

2.3 Os Grandes Modelos de Linguagem (LLMs)

Os LLMs são modelos baseados em arquiteturas Transformer (Vaswani et al., 2023), capazes de gerar e interpretar textos com alta coerência. Eles utilizam pré-treinamento em grandes volumes de dados e são ajustados ou utilizados por meio de *prompt engineering* para tarefas específicas.

Ferramentas como o GPT-4, GPT-5, Gemini 1.5 Flash/Pro e Claude 3 redefinem o campo ao permitir:

- Inferência sem exemplos (ZSL);
- Sumarização de grandes volumes textuais;
- Extração de requisitos bem estruturados;
- Explicação contextualizada;
- Raciocínio sobre artefatos complexos.

Segundo Naveed et al. (2024), LLMs podem apoiar sistematicamente a ER, especialmente em tarefas de análise de comentários, classificação de requisitos, entre outras atribuições.

Mas, apresentam desafios:

- Alucinações - respostas coerentes, porém falsas;
- Inconsistência semântica;
- Fragilidade ao ruído linguístico;
- Necessidade de guardrails (Ayyamperumal; Ge, 2024).

Por isso, a integração de LLMs com framework de controle, RAG e sistemas multiagente torna-se fundamental, conforme será tratado nas próximas seções.

2.4 Zero-Shot Learning (ZSL) aplicado à classificação de comentários

Zero-Shot Learning permite que um modelo execute uma classificação sem exemplos previamente rotulados no dataset. Em vez disso, utiliza descrições de classes em linguagem natural e representa, de forma semântica, o texto e os rótulos no mesmo espaço vetorial.
Segundo Bergmann (2024):

> A IA Generativa oferece uma solução alternativa para o problema de aprendizado zero-shot: usar informações auxiliares para gerar dados de amostra.

As aplicações deste modelo consiste em:

- Classificar comentários segundo tópicos dominantes (sementes);
- Formar um conjunto inicial de rótulos para orientar os agentes subsequentes nos modelos (Gemini, RAG+LangChain, GPT).

2.5 Os Sistemas Multiagente (MAS)

Sistemas multiagente envolvem múltiplos agentes autônomos que interagem para cumprir objetivos comuns. Um agente é definido como uma entidade que percebe o ambiente, raciocina e age de forma autônoma (Wooldridge, 2009).
Em associações recentes com LLMs, os agentes:

- Realizam tarefas especializadas;
- Cooperam em pipeline;
- Supervisionam e validam resultados;
- Trocam mensagens com raciocínio contextual.

A abordagem multiagente tem sido explorada para tarefas de:

- Curadoria de dados;
- Redação técnica;
- Validação cruzada;
- Detecção de anomalias (Winland; Syed; Gutowska, 2023).

No contexto desta dissertação, será útil para estruturar:

- Agentes extratores de requisitos;
- Agentes de fusão PO;
- Agentes analisadores de requisitos.

2.6 Orquestração de agentes com CrewAI

CrewAI é um framework moderno de coordenação de agentes LLM. Ele permite:

- Agentes especializados com papéis e competências distintas;
- Tarefas encadeadas (Tasks);
- Workflows personalizáveis;
- Memória compartilhada;
- Gestor central (Manager) responsável pela coordenação.

Trabalhos escritos, por exemplo (Mavroudis, 2022; Zhao, 2024), mostram que CrewAI tem sido utilizado para: projetos de escrita técnica; geração de relatórios; automação (para tarefas de Question Answering (QA) e workflows de (ES).

A orquestração garante que cada agente receba apenas o contexto necessário, evitando sobrecarga e reduzindo alucinações.

2.7 Recuperação apoiada por geração (RAG)

RAG (Retrieval-Augmented Generation) é uma técnica que combina:

- Recuperação de informações (retrievers: vetores, BM25, FAISS);
- Geração textual (LLM).

Os retrievers podem ser do tipo:

- Vetoriais (embeddings);
- Híbridos;
- Semântico (dense retrievers);
- Recuperadores neurais, que utilizam redes neurais profundas para buscar textos semanticamente relevantes.

No contexto de requisitos de software:

- LangChain atua como integrador do pipeline;
- O retriever encontra trechos relevantes dos comentários originais;
- O LLM reduz alucinações, pois é forçado a responder apenas com base no contexto recuperado.

Segundo Lewis et al. (2020), o RAG melhora a factualidade, a coerência contextual e a rastreabilidade.

2.8 Guardrails e detecção de anomalias

Os guardrails são mecanismos de controle para limitar comportamentos inaceitáveis de LLMs. Podem prevenir:

- Respostas incoerentes;
- Conteúdo fora do escopo;
- Requisitos que não correspondem aos comentários.

Aplicamos guardrails nos modelos:

1 - Gemini e CrewAI, (na filtragem semântica);

2 - RAG+LangChain e CrewAI, na validação baseada em contexto recuperado;

3 - GPT e CrewAI (na filtragem semântica);

4 - Na fusão dos três modelos (itens 1, 2, 3), na detecção de duplicidade, na consistência entre RF e RNF e na conformidade com IEEE 830 e 29148.

Ayyamperumal e Ge (2024) defendem que os *guardrails* são essenciais para LLMs em domínios críticos, reduzindo significativamente o risco de alucinações.

3 TRABALHOS RELACIONADOS

Os avanços recentes em PLN, LLMs e arquitetura multiagente têm impulsionado novas abordagens para a automação de tarefas de ER. Tais abordagens se mostram particularmente relevantes em contextos com grande volume de feedback textual, como lojas de aplicativos, onde a extração manual de requisitos é inviável, sujeita a inconsistências e dependente de interpretação humana.

De acordo com Naveed et al. (2024), os LLMs apresentam desempenho superior ao de modelos tradicionais ajustados, sobretudo quando orientados por estratégias de *prompting* adequados. Nesse mesmo sentido, Alhoshan, Ferrari e Zhao (2023) afirmam que LLMs podem reduzir ambiguidades e aprimorar a clareza dos requisitos. Esses achados dialogam diretamente com os objetivos desta pesquisa, que explora técnicas como ZSL, few-shot *prompting*, RAG e orquestração multiagente com CrewAI para extração automatizada de requisitos funcionais e não funcionais a partir de comentários de usuários.

A seguir, apresentam-se estudos alinhados aos pilares conceituais desta dissertação: PLN, LLMs, multiagente, ER para sistemas complexos e práticas profissionais em ML e Low-Code.

3.1 - “A design science research approach to Large Language Model-Based Agents for Requirements Specification (LLMBA4RS) in low-code applications”

Rotar et al. (2025) propõem o método LLMBA4RS, fundamentado no Design Science Research (DSR), para apoiar a especificação automatizada de requisitos em ambientes *Low-Code*. O método utiliza CrewAI como orquestrador de múltiplos agentes especializados e integra o RAG para fornecer contexto adicional aos LLMs durante a geração de histórias de usuário.

O estudo destaca que a ausência de padronização em sistemas Low-Code aumenta a ambiguidade dos requisitos, um problema recorrente também nos comentários de usuários analisados nesta dissertação. Assim como no presente trabalho, o método emprega agentes com papéis definidos, capazes de colaborar na extração e validação de requisitos.

Os resultados evidenciam:

- A redução de ambiguidades;
- O aumento da completude;

- Maior consistência na estruturação das histórias de usuários;
- A identificação de funcionalidades não mencionadas explicitamente.

Ainda assim, limitações relacionadas à dependência de prompts bem elaborados, à necessidade de curadoria humana e ao excesso de critérios de aceitação indicam espaço para aprimoramento, desafios que esta dissertação também busca mitigar por meio de *guardrails*, análise semântica e validações automáticas.

3.2 - "A Comprehensive Overview of Large Language Models"

Naveed et al. (2024) realizam um estudo sistemático comparando os modelos: Claude. DeepSeek, Gemini, GPT-4 e LLaMA em tarefas de classificação de requisitos. São avaliados três problemas centrais:

1 - A distinção entre RF e RNF;

2 - A categorização multicategórica de RNFs;

3 - A diferenciação entre requisitos de segurança e não segurança.

Técnicas de prompting, como ZS, few-shot, persona prompting e chain-of-thought, foram analisadas. Os autores concluíram que Gemine e DeepSeek apresentaram altíssimo desempenho, superando inclusive modelos *de transformer* ajustados. Essa conclusão sustenta a adoção de técnicas como ZSL e Few-Show, utilizadas nesta dissertação para classificação preliminar e posterior refinamento realizado por agentes especialistas.

A aplicação do macro-F1 como métrica reforça a importância de lidar com datasets desbalanceados, um desafio presente também nos comentários da Google Play utilizados nesta pesquisa.

3.3 - "Industrial Practices of Requirements Engineering for ML-Enabled Systems in Brazil"

Alves et al. (2023) examinam como profissionais brasileiros conduzem atividades de ER em sistemas orientados por ML. A pesquisa mostra que 40% das empresas utilizam ML e enfrentam desafios, eis alguns:

- Expectativas irreais de stakeholders;
- Indefinição adequada do problema;
- Falta de documentação formal;
- Descompasso entre requisitos e dados indisponíveis.

Os autores destacam que cientistas de dados frequentemente assumem tarefas de ER, o que pode levar à ausência de padronização e de documentação consistente, um cenário semelhante à problemática de comentários informais e ruidosos analisados nesta dissertação.

Além disso, requisitos não funcionais como qualidade dos dados, confiabilidade e explicabilidade emergem como prioritários, reforçando a necessidade de LLMs e agentes criteriosos, além da mera classificação textual, alinhando-se às demandas de sistemas inteligentes e modernos.

3.4 – ”Expel: LLM Agents Are Experimental Learners”

O Expel é um arcabouço, descrito por Dong et al. (2024), introduz um mecanismo de aprendizagem experimental em agentes LLM, baseado em três componentes:

1 Memória de experiências;

2 Autoavaliação do desempenho;

3 Reforço de comportamentos últimos.

A aplicação dessa abordagem resultou em melhorias expressivas nas tarefas de raciocínio de longo prazo, planejamento e tomada de decisão. Os agentes passaram a apresentar comportamentos semelhantes aos de aprendizes humanos, ajustando suas respostas de acordo com falhas ocorridas anteriormente.

Esse conceito se aproxima dos mecanismos de ajuste interativo, das validações semânticas e da detecção de anomalias utilizados nesta dissertação, como os *guardrails*. A aprendizagem experimental contribui para a visão de agentes autônomos capazes de evoluir continuamente, característica essencial para aplicações futuras em ER automatizada.

3.5 – *“Requirements Engineering for Machine Learning: Perspectives from Data Scientists”.*

O paradigma de desenvolvimento orientado a treinamento, característico de sistemas baseados em ML, exige novas competências e adaptações na ER tradicional. Discutido por Karanikolas et al. (2024), aborda os requisitos específicos que são:

- As métricas quantitativas (acurácia, precisão, recall);
- A explicabilidade e imparcialidade;
- A conformidade legal (como GDPR);
- Os requisitos referentes aos próprios dados de treinamento.

Um dos maiores desafios identificados é a tradução dessas métricas, técnicas em requisitos, em requisitos compreensíveis para stakeholders, problema semelhante à necessidade de traduzir comentários informais e ambíguos em requisitos estruturados e validados, como realizado nos modelos desta dissertação.

Os autores propõem um processo em quatro etapas (elicitação, análise, especificação e verificação e validação), que se alinha ao fluxo metodológico adotado neste trabalho, que envolve análise, extração de requisitos, refinamento e validação por agentes.

3.6 - Conclusões dos Trabalhos Relacionados

De forma abrangente, os estudos revisados convergem para a conclusão de que a integração com PLN, LLMs RAG, técnicas de *prompting* e orquestração multiagente representam a direção mais promissora para a automação da ER. Todavia, persistem desafios significativos, especialmente no que se refere à padronização dos artefatos extraídos, à explicabilidade dos modelos, à validação sistemática e à adaptação às especificidades de diferentes domínios.

Então. podemos concluir que:

Os avanços observados na literatura justificam e reforçam a relevância desta pesquisa, que se insere em uma lacuna ainda pouco explorada: a integração de LLMs, técnicas de *prompting*, RAG, multiagente e mecanismos de validação *(guardrails*) para extração e refinamento automatizado de requisitos a partir de comentários de usuários. A proposta apresentada nesta dissertação alinha-se às tendências contemporâneas e oferece contribuições concretas para o avanço da ER orientada por dados e suportada pela Inteligência Artificial (IA).

Veremos no quadro 1 um comparativo dos trabalhos explorados neste tópico.

O diferencial dos trabalhos explorados e do nosso trabalho é a utilização combinada de múltiplos modelos de LLMs e de um orquestrador de agentes para consolidar os resultados em um modelo de Product Owner, buscando o aprimoramento da qualidade dos requisitos extraídos.

O quadro 1 - demonstra a delineação dos cinco artigos cinco eixos conforme exposto.

Artigo	Tipo de estudo	Foco principal	Método/Modelo proposto	Resultados e contribuições	Desafios
1	Design Science Research	Automação da ER em ambientes	CrewAI, RAG, Agentes multiatores	Redução de ambiguidade; padronização de cenários de usuários; rastreabilidade	Dependência de *prompts*, integração ética

		Low-Code		aprimorada.	e de segurança.
2	Estudo comparativo experimental	Classificação automática de requisitos.	Avaliações de modelos de LLMs com técnicas de *prompts*.	Gemini e DeepSeeker superam modelos *transformers* ajustados, eficácia do few-shot.	Padronização de *prompts*, custo computacional.
3	Estudo empírico (misto)	Práticas de ER na indústria de ML.	Questionários e análises qualitativas.	Identificação de lacunas na formalização e maturidade da ER.	Falta de documentação e engajamento de especialistas.
4	Estudo experimental	Aprendizado experimental em agentes LLM.	Modelo Expel com memória e autoavaliação..	Aumento de eficiência e consistência; agentes reflexivos.	Dependência de modelos preparatórios; escalabilidade.
5	Estudo empírico.	Aplicação da ER no contexto de ML.	Entrevista com cientistas de dados.	Preparação de processo adaptado a ER em ML.	Falta de métricas claras e aplicabilidade.
6	Estudo experimental	Elicitação experimental com modelos de LLMs e agentes.	Cinco modelos LLMs integrados a um orquestrador de agentes.	Relatórios com requisitos de software extraídos de comentários. Mais agilidade na elicitação de requisitos.	Elaboração de prompts e requisitos mais enxutos.

Fonte: elaborado pelo autor.

4 METODOLOGIA

4.1 - Contextualização da metodologia

A metodologia adotada nesta pesquisa integra fundamentos de PLN, ER, LLMs, *guardrails* e Sistemas multiagente (Ghosh; Gunning, 2019; Wooldridge; Jennings, 1995; Wooldridge, 2009). O propósito é construir um pipeline híbrido e escalável capaz de transformar comentários de usuários, marcados por informalidades, ambiguidades e ruídos, em RF e RNF estruturados, rastreáveis e consolidados por um PO, alinhando-se à tendência de uso de técnicas de ML e LLM em ER (Cruz, 2025; Vogelsang; Bong, 2019; Alves et al., 2023).

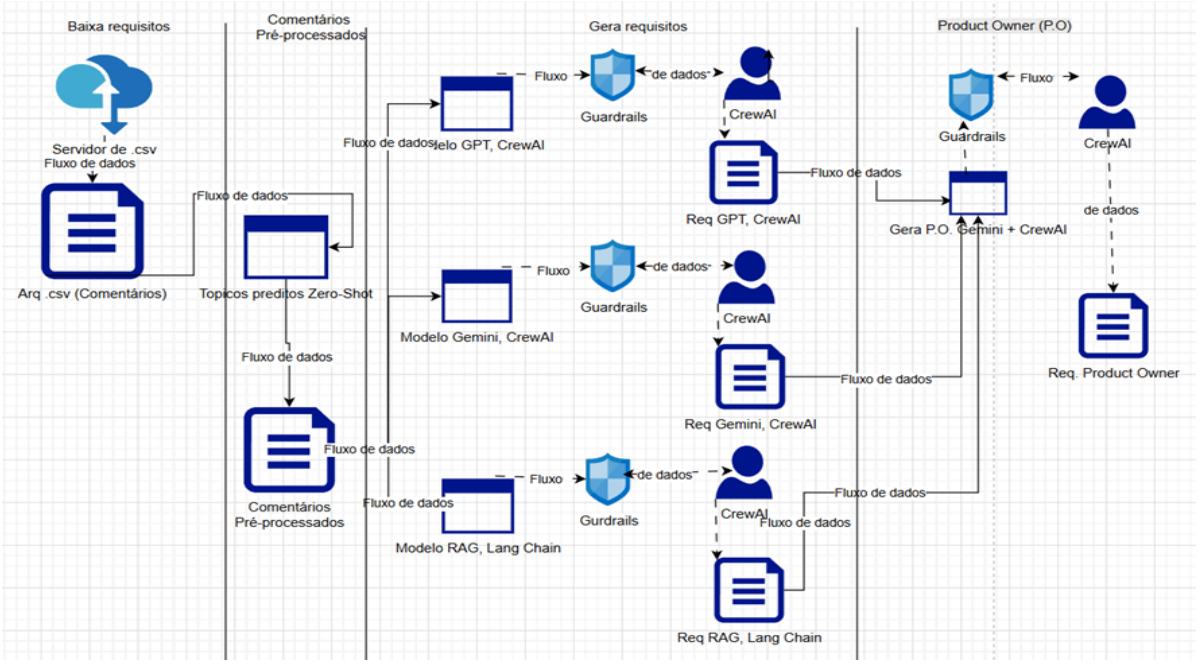
A abordagem metodológica foi organizada de forma modular, refletindo a evolução progressiva de complexidade das técnicas aplicadas. Assim definiram-se cinco modelos:

- Primeiro Modelo
  Para pré-processamento e estruturação da base de dados (um dataset);
- Segundo Modelo
  Para extração de requisitos com Gemini e CrewAI;
- Terceiro Modelo
  Para extração de requisitos com RAG, LangChain e CrewAI;
- Quarto Modelo
  Para extração de requisitos com GPT e CrewAI;
- Quinto Modelo
  Para Fusão e Consolidação via PO (Gemini e CrewAI)

Essa arquitetura em camadas, Figura 1, permite comparar distintos paradigmas de extração (LLMs, RAG, multiagente) e construir mecanismos de validação cruzada entre modelos, reduzindo a alucinação, a duplicidade de requisitos e as contradições, elevando o nível de confiabilidade dos requisitos obtidos (Lewis et al., 2020; Zheng et al., 2023; Jin et al., 2024).

Ainda na Figura 1 e subseção 3.8. É possível observar a arquitetura dos modelos deste experimento, a relação entre os *guardrails* nos modelos integrados aos agentes e fluxos dos dados indicados pelas setas.

Figura 1 - Arquitetura para extração de requisitos com modelos de LLMs inclusive com ação de *guardrails* e fusão com PO



Fonte: Elaborado pelo autor

4.2 - Prepara o arquivo .CSV

A primeira etapa consiste na preparação dos dados, compostos por comentários coletados em lojas de aplicativos. Os textos apresentam muitas variabilidades linguísticas possíveis, dentre elas:

- Informalidades e erros ortográficos;
- Abreviações e gírias;
- Emojis e símbolos;
- Frases incompletas;
- Metáforas ou ambiguidades semânticas;
- Descrições subjetivas ou pouco técnicas.

Esse tipo de dado textual tem sido explorado em trabalhos de elicitação de requisitos a partir de reviews, que destacam potenciais quanto aos desafios no trato de linguagem ruidosa (Bhatia; Singh; Sharma, 2023; Bhatia; Kaluza, 2018). Essas variações heterogêneas torna inviável a extração manual em larga escala e reforça a necessidade de um pipeline automatizado, apoiado em técnicas de PLN e de mineração de texto (Ghosh; Gunning, 2019; Lamba; Madhusudhan, 2022).

4.2.1 - Etapa de limpeza e normalização dos textos

Para mitigar o ruído linguístico, aplicaram-se técnicas de pré-processamento:

- Remoção de emojis, metadados e símbolos não textuais;
- Normalização dos textos;
- Eliminação de comentários duplicados;

Essas ações são recomendadas em pipelines de PLN e de mineração de texto (Ghosh; Gunning, 2019; Lamba; Madhusudhan, 2022). Essa etapa permite que os algoritmos subsequentes trabalhem com dados consistentes e comparáveis.

4.2.2 - TF-IDF e identificação de palavras-chave dominantes

O método (*Term Frequency - Inverse Document Frequency)* foi aplicado para realizar três finalidades principais:

- Identificar termos representativos do corpus;
- Fornecer insumos para análises exploratórias (por exemplo: nuvem de palavras, agrupamento semântico);
- Enriquecer o contexto fornecido aos agentes e aos LLMs durante a extração.

O uso do TF-IDF é consolidado em Recuperação de Informação e mineração de texto como estratégia para ponderar a relevância de termos em conjuntos de documentos (Ghosh; Gunning, 2019; Lamba; Madhusudhan, 2022). As palavras de maior peso TF-IDF serviram como *features* semânticas de apoio às etapas posteriores e dialogam com abordagens de extração de características a partir de reviews de aplicativos, utilizadas em ER (Bhatia; Singh; Sharma, 2023).

4.3 - Modelo de Classificação Zero-Shot Learning (Gemini+CrewAI)

O ZSL foi aplicado com o objetivo de fornecer uma classificação preliminar, auxiliando na triagem e classificando os comentários.

Esta classificação foi necessária para que houvesse agrupamentos por tópicos, pois, nos textos na sua origem, não há essa classificação; então, realizamos a identificação das possíveis categorias dos comentários dos usuários.

A utilização do ZSL para agrupamentos de requisitos é explorada na literatura; dentre elas, destacam-se: (Alhoshan; Ferrari; Zhao et al., 2023; Binkhonain; Alfayez, 2025). Apoiando-se em conceitos de aprendizado zero-shot e prompting (Bergmann, 2024; Syed; Gadesha, 2025; Kojima et al., 2022). Essa classificação inicial serve como subsídio para as

etapas subsequentes que são as extrações de requisitos.

Após a aplicação para a predição de tópicos usamos um algoritmo para a verificação da existência de ruídos, essa verificação foi aplicada com foco em:

- Detecção de emojis;
- Caracteres especiais;
- Detecção de valores numéricos com repetições;
- Textos com repetições;
- Vocabulários repetidos.

Para avaliar os possíveis ruídos, foi necessária a aplicação de uma taxa de ruído (0-1), pois a presença deles pode prejudicar o desempenho de técnicas de PLN e LLMs; assim, é necessária a identificação e mitigação nas etapas iniciais do processo (Kumar; Makhija; Gupta, 2020).

$$\text{Taxa de ruído} = \frac{\text{componentes ruidosos}}{\text{total de componentes analisados}}$$

O valor da métrica é normalizado no intervalo (0 - 1) , em que quanto mais próximos de 0 estivermos, menos ruído teremos.

A identificação dos componentes ruidosos baseia-se em critérios heurísticos amplamente discutidos na literatura (Al Sharou; Li; Specia, 2021; Liu et al., 2022).

A métrica aplicada nesta etapa serve para avaliar os procedimentos preventivos iniciados em 4.2.

4.4 - Modelo de Extração de Requisitos com Gemini e CrewAI

Este modelo explora a arquitetura Gemini 1.5 Flash/Pro integrada a agentes especialistas da CrewAI, em uma arquitetura multiagente orientada a tarefas (Caballar; Stryker, 2025; Lorenze, 2024; Winland; Syed; Gutowska, 2025). O objetivo é extrair requisitos estruturados a partir do texto original, mantendo clareza, consistência e aderência às boas práticas de especificação de requisitos de software (Pressman, 2011; Gonca; Gonca, 2023; He et al., 2023).

4.4.1 - Agente utilizado e ações

- Um agente Extrator de Requisitos;

- Que classifica cada requisito como RF ou RNF, com justificativa, em linha com estudos que exploram LLMs para a classificação de requisitos (Alhoshan; Ferrari; Zhao et al., 2023; Binkhonain; Alfayez, 2025).

4.5 - Modelo de Extração de requisitos com RAG, LangChain e CrewAI

Este utiliza de pipeline RAG para minimizar as alucinações e garantir que cada requisito esteja totalmente ancorado em partes do texto do dataset gerado no tópico 4.3 (Lewis et al., 2020). Essa abordagem combina recuperação neural com extração condicionada, sendo particularmente adequada para tarefas intensivas em conhecimento.

4.5.1 Componentes principais do RAG

- Um processo que cria vetores usando aprendizado profundo - Embedding Model (sentence-transformers): utilizado para gerar representações vetoriais de alta densidade semântica, alinhadas ao uso de embedding em tarefas de recuperação densa;
- Uma biblioteca de código aberto do Facebook AI — FAISS Retriever — para recuperar comentários mais semelhantes ao texto analisado.

Além disso:

- Janela de Contexto Dinâmica: adapta a quantidade de texto entregue ao LLM, evitando sobrecarga cognitiva;
- LLM Generator: para gerar requisitos com base exclusivamente no contexto recuperado, em conformidade com o paradigma RAG (Lewis et al., 2020).

A implementação do fluxo foi apoiada em LangChain como *framework* de orquestração de cadeias de recuperação e extração (Mavroudis, 2022).

4.5.2 Estrutura de agente CrewAI

- Um agente gerador de RF e RNF - para elaborar os requisitos com base no contexto recuperado.
- São realizadas a padronização e validação técnica, em linha com boas práticas de documentação de requisitos (Pressman, 2011; Gonca; Gonca, 2023; He et al., 2023).

4.5.3 Vantagens técnicas estimadas

- Rastreabilidade explicita: cada requisito aponta para trechos recuperados;
- Redução significativa de alucinações e aumento da factualidade em comparação a modelos puramente paramétricos (Lewis et al., 2020; Ayyamperumal; Ge., 2024);
- Maior confiabilidade no vínculo entre texto original e requisito gerado;
- Aumento da consistência semântica entre requisitos, favorecendo a evolução de requisitos guiados por dados (Vogelsang; Bong., 2019; Alves et al., 2023).

4.6 Modelo de Extração de Requisitos com GPT e CrewAI

Para este modelo, utilizamos o GPT-4 integrado a uma arquitetura multiagente similar ao modelo no tópico 4.4; integrado ao GPT-4 usamos o CrewAI (Lorenze, 2024; Winland; Syed; Gutowska, 2025; Naveed et al., 2024; Jin et al., 2024). Na orquestração de agentes também utilizamos um para extração de requisitos:

4.6.1 Agente utilizado e ações

- Um agente Extrator de Requisitos;
- Ação - Produz um conjunto de RF e RNF a partir dos comentários inicialmente tratados em 4.3;
- Consequência - Amplia a cobertura semântica ao explorar este recurso de LLM (Naveed et al., 2024; Zheng et al., 2023);
- Esperado - Contribui para a robustez do processo, uma vez que os requisitos tendem a ser mais confiáveis; todavia, as discrepâncias podem ser tratadas como insumos para análise crítica.

O agente aplicado responsável pela classificação e escrita dos requisitos segue os princípios de clareza, completude e boa formação de requisitos, também as diretrizes textuais inspiradas em SRS (Gonca; Gonca, 2023; He et al., 2023) e em estudos que exploram LLMs para classificação e tratamento automáticos de requisitos (Alhoshan; Ferrari; Zhao et al., 2023; Binkhonain; Alfayez, 2025).

4.7 Product Owner (PO)

O Product Owner (PO) integra, consolida e refina os requisitos provenientes dos três modelos anteriores. É etapa central no pipeline, pois estabelece um mecanismo de reconciliação entre as três visões independentes de extração de requisitos, cada uma de uma arquitetura distinta.

Ao receber essas três perspectivas complementares, o PO atua como um mecanismo de síntese, inspirado em processos humanos de ER nos quais múltiplas fontes de evidência são analisadas para construir um documento consistente e livre de ambiguidades (Pressman, 2011; Gonca; Gonca, 2023; He et al., 2023).

Funções principais do PO:

- Unificação semântica das três visões independentes;
  Agrupa os requisitos provenientes do Gemini, do RAG e do GPT, tratando sobreposições, divergências e complementaridades.
- Eliminação de duplicidades e redundâncias;
  Requisitos semelhantes ou sem diferença funcional relevante são consolidados em uma única versão mais clara e completa.
- Detecção e resolução de inconsistências;
  Diferentes modelos podem propor requisitos contraditórios ou mutuamente excludentes.
  O PO identifica esses casos e seleciona a formulação mais adequada e mais próxima do texto original.
- Classificação final (RF e RNF);
  Embora cada modelo já produza rótulos preliminares, a decisão final é tomada neste módulo, garantindo a coerência entre todos os requisitos.
- Padronização textual com base em boas práticas de SRS;
  O PO, reestrutura o texto para aderir a padrões de clareza, precisão e testabilidade, amparando-se em estudos sobre *boilerplates* e qualidade de requisitos.
- Remoção de vieses e inferências indevidas;
  Uma vez que diferentes LLMs podem introduzir inferências não fundamentadas, essa etapa assegura que cada requisito esteja alinhado ao comentário original ou ao contexto recuperado.

Os requisitos são categorizados segundo os temas identificados no modelo referente ao subitem 4.3, permitindo a análise por área.

4.8 Guardrails e Anomalias

Os *guardrails* são mecanismos de proteção e de detecção de anomalias. Foram utilizados, inclusive, nos modelos de extração de requisitos, para mitigar efeitos indesejados de alucinações, inconsistências e redundâncias em saídas de LLMs, em consonância com a literatura recente sobre riscos e guardrails para LLMs (Ayyamperumal; Ge, 2024; Dong et al., 2024).

Os *guardrails* foram aplicados com os seguintes propósitos:

- Na filtragem semântica

  Para eliminam os requisitos que:

  Não possuem correspondência real com o comentário;

  Adicionam funcionalidades inexistentes;

  Apresentam inferências factualmente incorretas.

Esse tipo de filtro está alinhado com propostas que defendem camadas de proteção para limitar o espaço de respostas dos modelos e reduzir riscos de alucinação (Ayyamperumal; Ge. 2024; Dong et al., 2024).

- Na checagem de inconsistências

  Validação de requisitos:

  Se RF descreve uma função concreta;

  Se RNF descreve a qualidade ou atributo do sistema;

  Tem a ação em conformidade com padrões de clareza e precisão de requisitos (Pressman, 2011; Gonca; Gonca, 2023; He et al., 2023);

  Se há contradições nos requisitos.

- Detecção de anomalias

  Avalia:

  A duplicação excessiva de requisitos;

  Fraude de contexto (uso incorreto de frases não relacionada);

  Requisitos inventados ou redundantes;

Além disso, incorpora a discussão sobre agentes LLM que aprendem com experiência e interagem com ambientes complexos (Zhao et al., 2023; Jin et al., 2024), reforçando a importância de monitoramento e mitigação contínua de comportamentos indesejados (Ayyamperumal; Ge, 2024; Dong et al., 2024).

4.9 Algumas considerações sobre Interpretabilidade e o Problema da Caixa-Preta (CP) no PLN

As técnicas de PNL empregadas neste trabalho baseiam-se em modelos de aprendizado de máquina, LLMs e orquestração de agentes com CrewAI. Essas abordagens apresentam elevado desempenho em tarefas de análise e geração de textos; a literatura observa que algumas etapas dessas técnicas são realizadas em sistemas de caixa-preta; portanto, não são facilmente interpretáveis (Guidotti et al., 2018).

Em modelos neurais profundos, especialmente aqueles baseados em arquiteturas Transformer, a relação entre entradas textuais e saídas geradas é mediada por uma grande quantidade de parâmetros ajustados durante o treinamento, o que dificulta a explicação clara de como determinadas decisões são tomadas (Jurafsky,; Martins, 2023). Essa característica representa um desafio relevante em contextos em que a transparência, a rastreabilidade e a confiabilidade das decisões são desejáveis, como na ES e, particularmente, na ER.

Quando nos referimos à extração automática de requisitos, a natureza da CP nos modelos de PLN pode resultar na geração de respostas inesperadas e complexas para esclarecer os resultados. Conforme discutido por Ribeiro; Singh; Guestrin et al. (2016), a ausência de mecanismos de explicação compromete a confiança nos resultados.

Diante desse cenário, a metodologia proposta neste trabalho adota estratégias para mitigar problemas da CP em PLN, sem desconsiderar seus benefícios. Em vez de depender exclusivamente das decisões implícitas dos modelos de linguagem, a abordagem incorpora etapas explícitas de pré-processamento, métricas quantitativas e validações intermediárias, como a aplicação de métricas de ruído textual e o uso de *guardrails* ao longo do *pipeline*. Observe as implementações nas Figuras 2, 3, 5, 12, 20 e 21.

Além disso, a utilização de uma arquitetura baseada em agentes especializados e a introdução do PO com o propósito de aumentar o controle sobre o processo de qualificação dos requisitos. Esses elementos funcionam como camadas adicionais de verificação e refinamento, visando reduzir a opacidade das decisões e promover maior previsibilidade e alinhamento dos resultados com os objetivos da ER.

Dessa forma, embora o PLN e os LLMs utilizados mantenham características inerentes de CP, a metodologia proposta busca minimizar seus impactos por meio de uma estrutura modular, orientada por métricas e regras explícitas, aumentando a confiabilidade e a qualidade dos requisitos de software extraídos automaticamente.

5 DOS RESULTADOS OBTIDOS

Neste capítulo, apresentamos os resultados obtidos a partir da execução dos cinco modelos da metodologia descrita no capítulo três. São analisados os aspecto quantitativos, por exemplo: (o volume de requisitos extraídos, a distribuição por tópicos, a frequência de RF e RNF e as diferenças entre os modelos), no aspecto qualitativo temos (a clareza, a completude. a redundância, a coerência, e a aderência às normas IEEE 29148).

O capítulo também discute:

- As contribuições na geração de nuvem de palavras;
- A eficácia dos guardrails;
- A avaliação do Product Owner;
- As limitações dos artefatos utilizados;
- As relações entre os resultados e as questões da pesquisa.

5.1 A Estrutura dos Dados e Estatísticas Iniciais

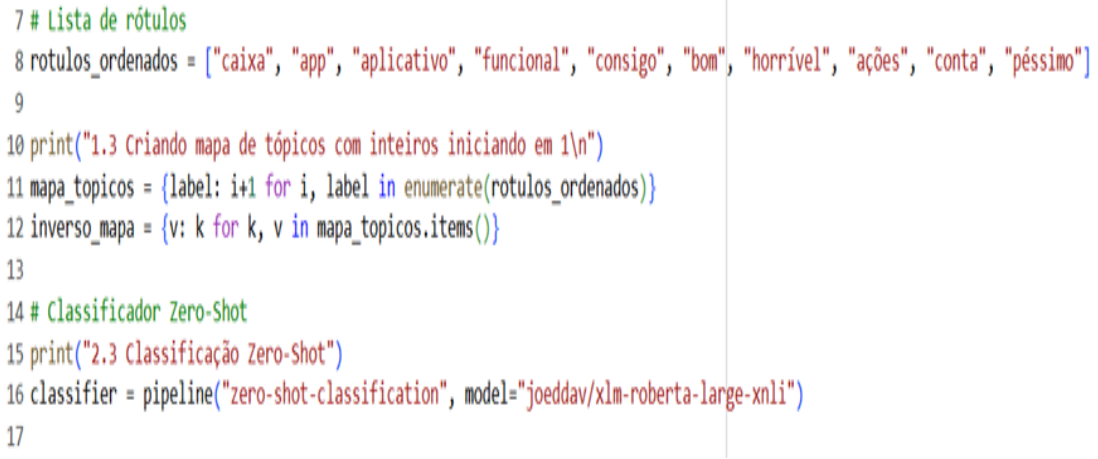
5.1.1 Das criações de tópicos dos comentários.

Após o pré-processamento no subtópico 4.3, obteve-se:

- Número total de comentários = 229.
- Quantidade de tópicos gerados = 10.

Trecho de código do modelo, Figura 2 (Gemini e Zero-Shot) para geração de tópicos, classificação e agrupamento de comentários por tópicos a partir das sementes.

Figura 2 - Sementes para geração de grupos de comentários

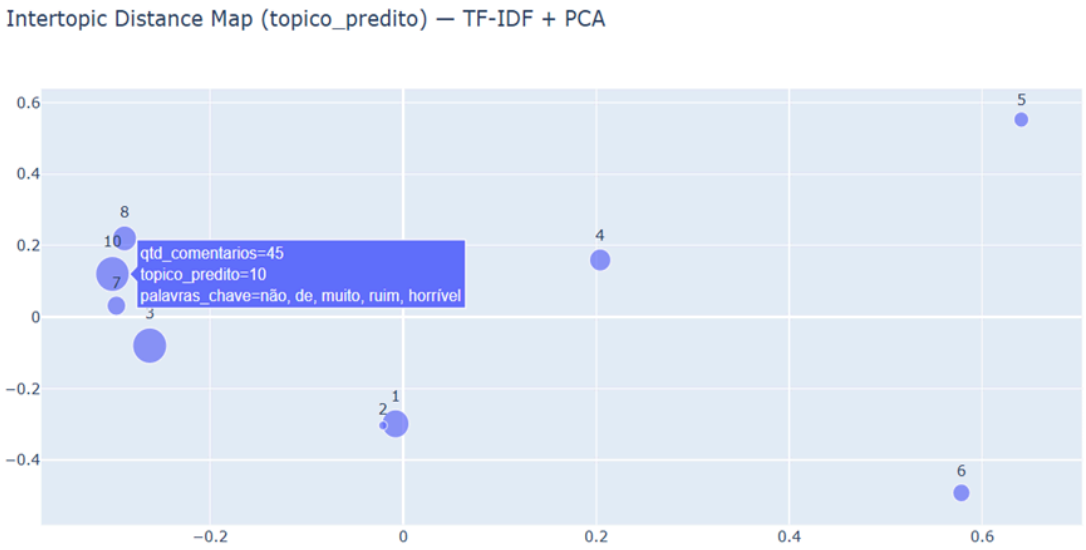


```
# Lista de rótulos
rotulos_ordenados = ["caixa", "app", "aplicativo", "funcional", "consigo", "bom", "horrível", "ações", "conta", "péssimo"]

print("1.3 Criando mapa de tópicos com inteiros iniciando em 1\n")
mapa_topicos = {label: i+1 for i, label in enumerate(rotulos_ordenados)}
inverso_mapa = {v: k for k, v in mapa_topicos.items()}

# Classificador Zero-Shot
print("2.3 Classificação Zero-Shot")
classifier = pipeline("zero-shot-classification", model="joeddav/xlm-roberta-large-xnli")

```

Fonte: Elaborado pelo autor

- Análise sobre os grupos criados

Gráfico 1 - Intertopic Distance Map, distribuição dos comentários por quadrante.



Fonte: Elaborado pelo autor

Exemplos de textos **COMO ESTÃO PUBLICADOS** pelos usuários do APP e classificados de acordo com seus tópicos após aplicação do ZSL:

Quadrante I - Tópico 4

“Já usava o home broker no internetbank. Funciona normal. Às vezes o sistema tá em manutenção, no mais, funciona bem. Sugestão: acrescentar um "dark mode" ajudaria muito minha visão. Outra, se fazer esquema para gerenciar os investimentos, ver ganhos, alocação seria uma boa. Evitaria o trabalho de tá quebrando a cabeça com planilhas. Outra boa idéia seria aqueles resuminhos para declarar IR dia ativos. E, se não for demais diminuir a taxa de corretagem.? ”

Quadrante I - Tópico 5

“Quero consultar meu estrato do mês de Dezembro e salário do mês de Dezembro”

Quadrante II - Tópico 7

“Absurdo! Ao instalar e se logar pela primeira vez, ele DESINSTALA a sua conta do Internetbanking. Você é obrigado a ir na agência resolver o problema. “

Quadrante II - Tópico 8

“Não aparece nada na tela de COMPRAR E VENDER ACOES. Faz 5 dias que não consigo. consiltar minhas ações. “

Quadrante II - Tópico 9

“Não consigo validar o dispositivo. Sou correntista, tentei usar o Internet Banking mas pede a leitura do QR code que não aparece no app Caixa Ações On-line. Tentei o caixa

eletrônico na agência. Porém, no final do processo, após a leitura dos QR codes, aparece uma mensagem dizendo que a criptografia não é reconhecida".

Quadrante II - Tópico 10.”

“Dando uma estrela para publicar a minha indignação, péssimos todos os aplicativos da caixa nenhum presta todos são uma vergonha decepção total por favor os desenvolvedores destes apps devem sentir-se envergonhados ou eles se divertem com os usuários. Qual é mesmo a utilidade destes apps. Você desenvolvedor sente-se lisonjeado?”

Quadrante III - Tópico 1

“pede uma validação que não consigo fazer. já fui ao banco e da erro ao ler o qrcode do caixa”.

Quadrante III - Tópico 2

“App que que rouba informações de localização e contatos da agenda. Pois e desnecessario obter essas informações para app funvionar, mas ficamos reféns disto porque se não autorizar, então o app não permite acessar a conta. “

Quadrante III - Tópico 3

“Esse APP poderia ser mais prático. Algumas transações poderiam ser mais simplificadas. Porém, em tempos como esse um app assim acaba sendo importante pela necessidade.”

Quadrante IV - Tópico 6

“Absurdo! Ao instalar e se logar pela primeira vez, ele DESINSTALA a sua conta do Internetbanking. Você é obrigado a ir na agência resolver o problema”

Até o momento, sobre o conjunto de dados analisado, especificamente as colunas comentários e topico_predito. Podemos concluir que:

- O modelo realizou a segmentação dos comentários, em que cada tópico representa um conjunto de textos semanticamente relacionados.
- Determinados temas concentram maior volume de ocorrências, indicando maior recorrência na percepção dos usuários.
- Os tópicos associados ao uso do aplicativo, ao funcionamento do sistema e às avaliações gerais (positivas e negativas) apresentam maior frequência.
- Os usuários concentraram mais feedbacks nas funcionalidades e nas qualidades de uso.
- O gráfico Intertopic Distance Map, evidencia que, os tópicos semelhantes, portanto semanticamente mais próximos, como os relacionados ao uso do aplicativo e

as funcionalidades do sistema.

- Os agrupamentos mais próximos podem indicar a existência de subtemas em um mesmo domínio funcional.
- Os agrupamentos mais distantes podem indicar diferenças semânticas, como avaliações positivas e negativas, e distinguir polaridades existentes nos comentários.

De modo geral, os resultados demonstram uma abordagem eficaz para transformar comentários não estruturados em grupos semânticos coerentes. A combinação da classificação automática com a análise visual, por meio do Intertopic Distance Map, contribui para uma interpretação mais clara dos temas dominantes.

5.1.2 Dos ruídos investigados e caracteres especiais

Dados quanto aos ruídos e outros obstáculos:

- Taxa de ruído nos comentários, aproximadamente = 0.2.
- Taxa para caracteres especiais = 0.0644.

Figura 3 - Trecho de código para identificação dos ruídos.

```
30 def noise_score(text):
31     if not isinstance(text, str) or len(text.strip()) < 3:
32         return 1.0
33     score = 0
34     if has_emoji(text):
35         score += 0.15
36     if special_char_ratio(text) > 0.2:
37         score += 0.2
38     if numeric_ratio(text) > 0.3:
39         score += 0.15
40     if repetition_ratio(text) > 0.1:
41         score += 0.15
42     if out_of_vocab_ratio(text) > 0.4:
43         score += 0.2
44     if len(text.split()) < 3:
45         score += 0.15
46     return min(score, 1.0)
```

```
nltk_data] Downloading package words to /root/nltk_data...
nltk_data]   Unzipping corpora/words.zip.
```

```
1 df = dfComentariosClassificados
2 df["taxa_ruido"] = df["comentarios"].apply(noise_score)
```

Fonte: Elaborado pelo autor

Sobre os ruídos inicialmente investigados é possível considerar que, o percentual para o tamanho e quantidade de registros consideramos um valor pequeno. Este resultado possivelmente é devido à limpeza realizada no início da aplicação deste modelo; veja o tópico 4.3.

Sobre a quantidade de caracteres especiais, possivelmente devido à presença de um fragmento isolado e não identificado pelo algoritmo aplicado.

5.2 Resultados com Gemini e CrewAI

5.2.1 Das gerações de requisitos

O modelo em questão gerou:

- Total de requisitos extraídos = 3882.
- Exemplos de requisitos extraídos no apêndice A.

Trechos de código para extração de requisitos a partir dos comentários em 4.1.

Figura 4 - Agente para extração de requisitos

```
#Agente gerador de requisitos
req_generator = Agent(
    role = "Requiriments Generator",
    goal = "Generator clear and concise software requiriments based on the text analysis, "
           "categorizing then into functional and non-functional requirements. ",
    backstory= "You are skilled software enginer with expertise in writing high-quality "
               "requiriments, specifications. You can translate text insights into actinable "
               "software requiriments, disringguisshing between funcional and non-funcional aspects.",
    verbose=True,
    llm=openai_llm # Pass the instantiated LLM here
)
```

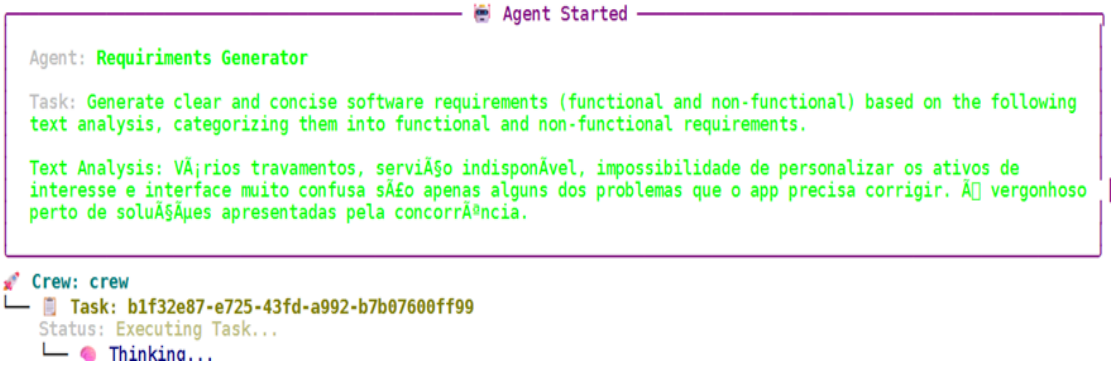
Fonte: Elaborado pelo autor.

Figura 5 - Tarefa para extração de requisitos

```
# Definição da tarefa
tarefa_gerar_req = Task(
    description=(
        "Generate clear and concise software requirements (functional and non-functional) based on the following text
         "analysis, "
        "categorizing them into functional and non-functional requirements.\n\n"
        "Text Analysis: {text_data}"
    ),
    expected_output="A list of categorized software requirements based on the provided text analysis. Format the output
                    "clearly with Functional and Non-Functional sections.",
    agent=req_generator
)

# Criação da crew
crew_per_insight = Crew(
    agents=[req_generator],
    tasks=[tarefa_gerar_req],
    verbose=True,
    process='sequential'
)
```

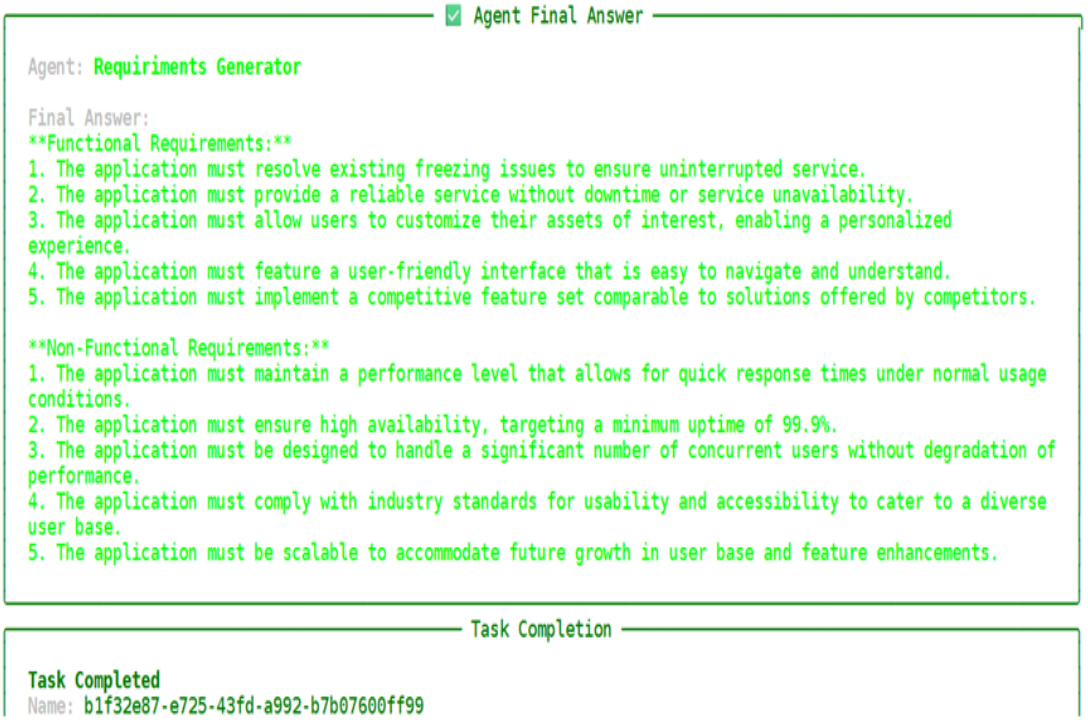
Fonte: Elaborado pelo autor.

Figura 6 - Tarefa para extração de requisitos



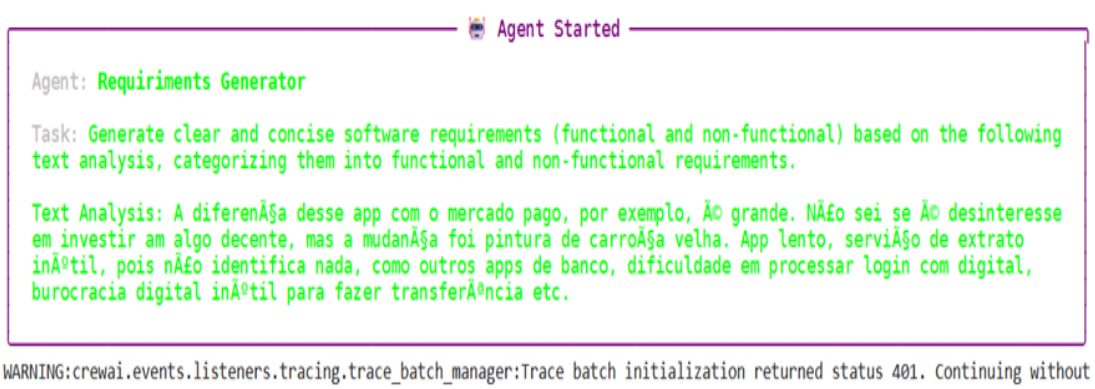
Fonte: Elaborado pelo autor

Figura 7 - Exemplos de requisitos



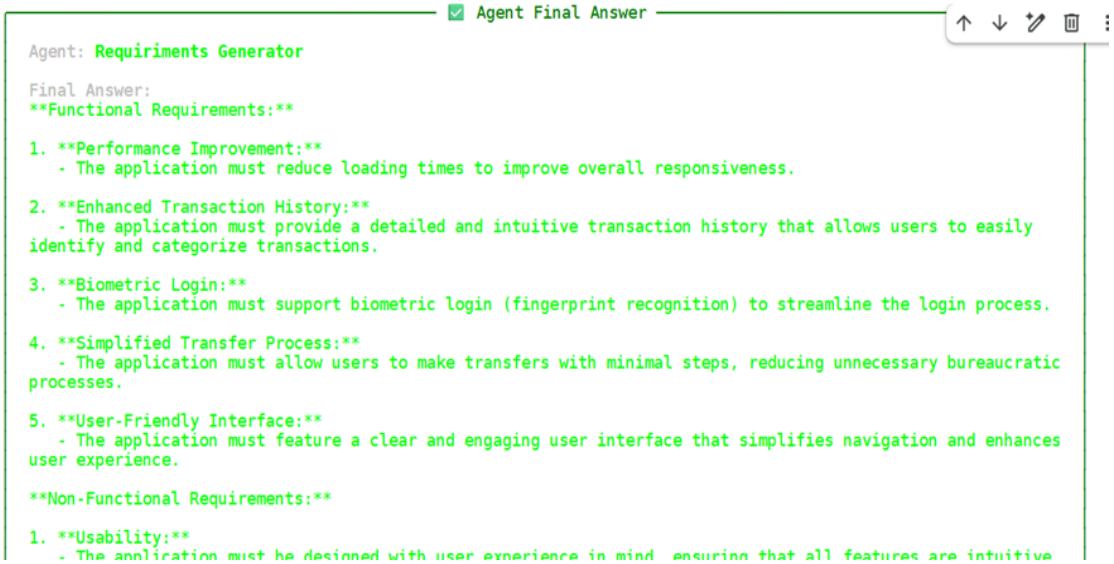
Fonte: Elaborado pelo autor

Figura 8 - Alguns requisitos extraídos antes de serem submetidos ao PO



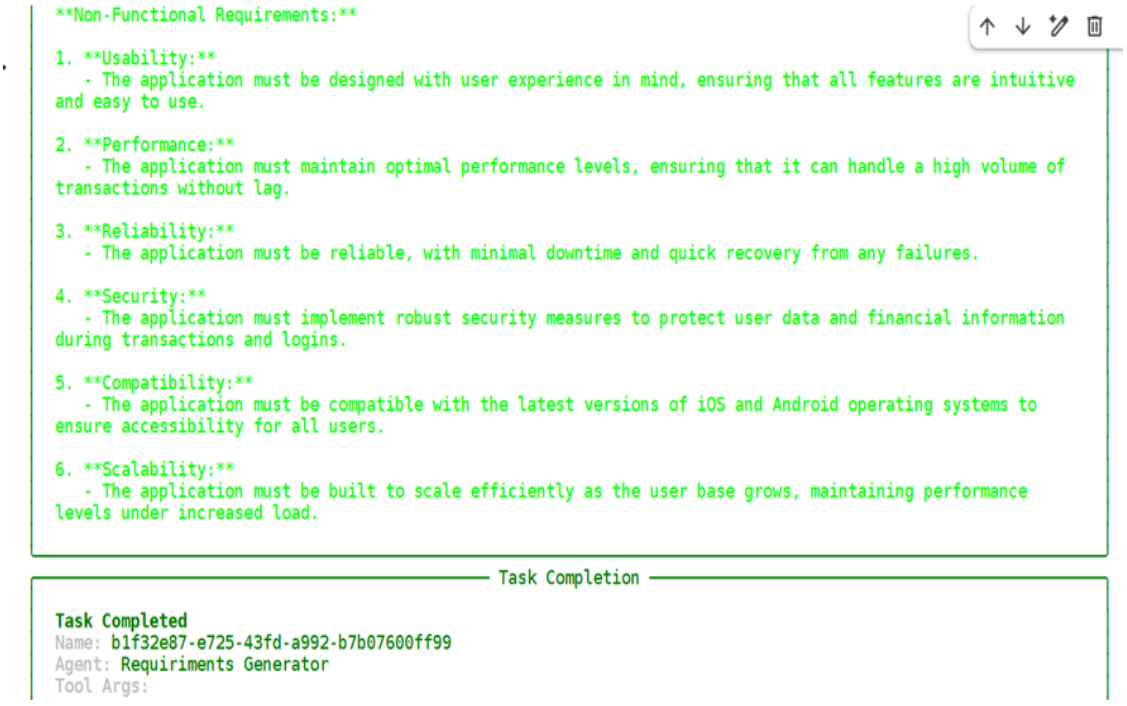
Fonte: elaborado pelo autor

Figura 9 - Alguns requisitos extraídos antes de serem submetidos ao PO



Fonte: Elaborada pelo autor

Figura 10 - Alguns requisitos extraídos antes de serem submetidos ao PO



Fonte: Elaborada pelo autor

Observando esses resultados e outros requisitos extraídos constantes no apêndice A, é possível perceber que, a partir de alguns comentários, foram extraídos mais de um requisito e, a partir deles, outros subgrupos de requisitos.

- Alguns requisitos condizem com seus comentários.

- A quantidade de requisitos extraídos para a mostra de comentários foi bem além da quantidade dos modelos seguintes. Pode ter ocorrido alguma anomalia nas gerações de requisitos neste modelo de LLM e no CrewAI.

5.3 Resultados com RAG, LangChain e CrewAI

5.3.1 Dos requisitos extraídos

O modelo em questão gerou:

- Total de requisitos extraídos = 229.
- Exemplos de requisitos extraídos no apêndice A.

Figura 11 - Trechos de códigos para extração de requisitos a partir dos comentários em 4.1.

```
req_gererator = Agent(
    role="Requirements Engineer",
    goal = "Generator clear and concise software requiriments based on the text analysis, "
           "categorizing then into functional and non-functional requiriments. ",
    backstory = "You are a skilled software enginnner with expertise in writing high-quality "
                "requiriments, specifications. You can translate text insights into actionable "
                "software requiriments, distinguisshing between funcional and non-funcional aspects.",
    verbose=False,
    llm=llm # Explicitly pass the llm to the agent
)
```

Fonte: Elaborado pelo autor

Figura 12 - Tarefa para extração de requisitos

```
# Cria a tarefa
tarefa_gerar_requisitos = Task(
    name='generate_reqs',
    description='Generate clear and concise software requirements based on the text analysis, '
                'categorizing them into functional and non-functional requirements.',
    expected_output='A detailed list of functional and non-functional software requirements derived from the text analysis.',
    agent=req_gererator
)

# Cria a Crew
crew = Crew(
    agents=[req_gererator],
    tasks=[tarefa_gerar_requisitos],
    verbose=False
)
```

Fonte: Elaborada pelo autor

Figura 13 - Alguns requisitos extraídos antes de serem submetidos ao PO.

```
Functional Requirements:
1. The software must have a user authentication feature to ensure secure access to the system.
2. The system should provide users with the ability to create, edit, and delete their profiles.
3. Users should be able to search for and connect with other users based on specific criteria.
4. The software needs to support real-time messaging functionality for communication between users.
5. The system must allow users to upload, share, and manage multimedia content such as images and videos.
6. Users should have the option to customize their privacy settings to control who can view their profile and
```

Fonte:- elaborada pelo autor.

Figura 14 - Alguns requisitos extraídos antes de serem submetidos ao PO.

```
Non-Functional Requirements:
1. The system should have a response time of under 2 seconds for user actions such as profile loading and messag
2. The software must be scalable to accommodate a growing user base without compromising performance.
3. The system should adhere to data protection regulations such as GDPR to ensure user privacy and security.
4. The software must be reliable, with an uptime of at least 99.9% to minimize service disruptions.
5. The system should be user-friendly, with an intuitive interface and clear navigation for users of all technic
6. The software must be developed using secure coding practices to prevent vulnerabilities and protect user data
```

Fonte: Elaborada pelo autor.

Para esse modelo, tivemos um requisito por comentário; sabemos, entretanto, que, a partir de alguns comentários, pode haver necessidade de mais de um requisito.

5.4 Resultados com GPT e CrewAI

5.4.1 Dos requisitos extraídos

O modelo em questão gerou:

- Total de requisitos extraídos = 34
- Exemplos de requisitos extraídos no apêndice A.

Figura 15 - Trecho de código para extrair requisitos

```
req_generator = Agent( # renamed from agente_gerador_requisitos for consistency with cell 3.2
    role = "Requirimens Generator",
    goal = "Generator clear and concise software requiriments based on the text analysis, "
           " categorizing then into functional and non-functional requirements. ",
    backstory= "You are a skilled software enginner with expertise in writing high-quality "
               " requiriments, specifications. You can translate text insights into actionable "
               " software requirements, disringuisshing between funcional and non-funciotional aspects.",
    verbose=True
```

Fonte: Elaborada pelo autor

Figura 16 - Tarefa para extração de requisitos

```
generate_reqs = Task(
    name='generate_reqs',
    description="Generate software requiriments based on the text analysis results.",
    expected_output="A compreesive list of functional and non-functional software requiriments derived from
    output=TaskOutput(
        description="A compreesive list of functional and non-functional software requiriments derived from
        agent=req_generator.role,
        functional_requirements=[],
        non_functional_requirements=[],
        exemples=[
        {
            "input":"O sistema deve ser rápido.",
            "output": "Requisito muito vago. 'Rápido', é subjetivo. Recomenda-se incluir métricas de desempe
        },
        {
            "input": "Não consigo encontrar onde alterar minha senha.",
            "output": "O sistema deve oferecer uma funcionalidade de alteração de senha acessível e visível.
        },
        {
```

Fonte: Elaborada pelo autor

Figura 17 - Alguns requisitos extraídos

```
Agent: Requirimens Generator

Final Answer:
**Functional Requirements:**

1. **User Authentication:**
   - The system shall allow users to create accounts using an email and password.
   - The system shall implement multi-factor authentication for enhanced security.
   - The system shall enable users to reset their passwords via a secure email link.

2. **User Roles and Permissions:**
   - The system shall support multiple user roles (e.g., Admin, User, Guest) and assign permissions
accordingly.
   - The system shall allow administrators to manage user roles and permissions dynamically.

3. **Data Input and Management:**
   - The system shall allow users to input data through forms with validations.
   - The system shall enable users to edit and delete their data entries.
   - The system shall allow bulk data uploads via CSV files.
```

Fonte: Elaborada pelo autor

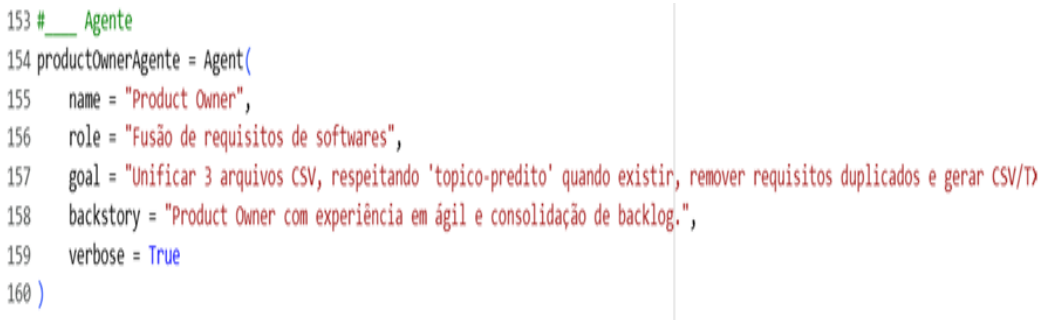
5.5 Resultados com realização da fusão dos requisitos extraídos nos três modelos anteriores

5.5.1 Fusão dos requisitos extraídos

O propósito deste subitem é apresentar os resultados iniciais da fusão, com os mesmos recursos preventivos aplicados nos modelos anteriormente implementados, no início todos os requisitos foram mantidos sem alterações, inclusive as quantidades geradas em seus modelos individualmente e informadas em seus subtópicos, somando todos os requisitos temos o total de 4129.

Aplicamos também o conceito de orquestração de agentes; veja as figuras 18 e 19.

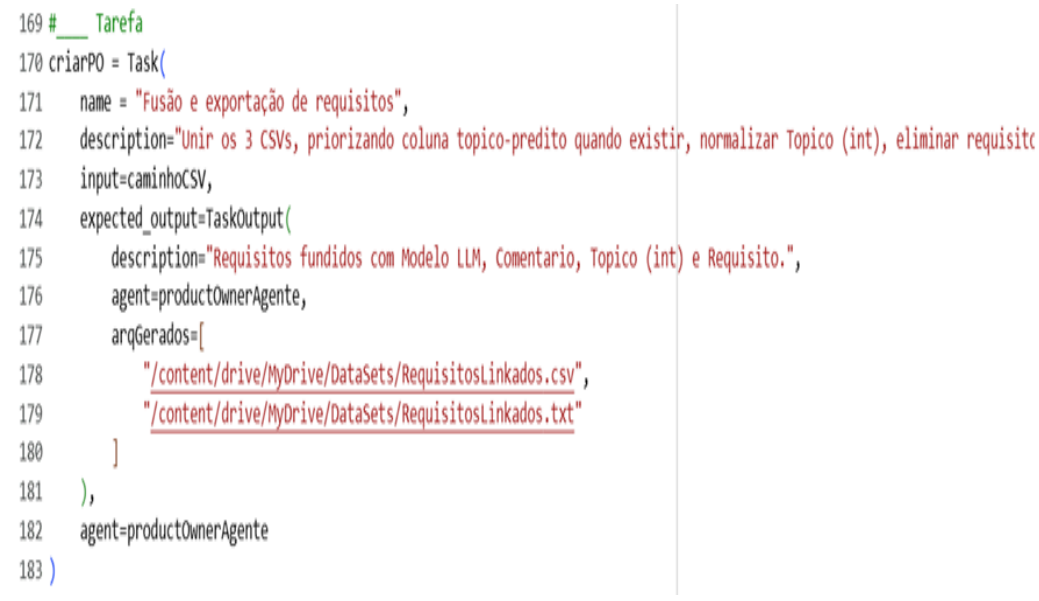
Figura 18 - Implementação do agente para realizar a fusão dos requisitos



```
153 #___ Agente
154 productOwnerAgente = Agent(
155     name = "Product Owner",
156     role = "Fusão de requisitos de softwares",
157     goal = "Unificar 3 arquivos CSV, respeitando 'topico-predito' quando existir, remover requisitos duplicados e gerar CSV/T
158     backstory = "Product Owner com experiência em ágil e consolidação de backlog.",
159     verbose = True
160 )
```

Fonte: Elaborada pelo autor

Figura 19 - Chamada para execução do agente



```
169 #___ Tarefa
170 criarPO = Task(
171     name = "Fusão e exportação de requisitos",
172     description="Unir os 3 CSVs, priorizando coluna topico-predito quando existir, normalizar Topico (int), eliminar requisitc
173     input=caminhoCSV,
174     expected_output=TaskOutput(
175         description="Requisitos fundidos com Modelo LLM, Comentario, Topico (int) e Requisito.",
176         agent=productOwnerAgente,
177         arqGerados=[
178             "/content/drive/MyDrive/DataSets/RequisitosLinkados.csv",
179             "/content/drive/MyDrive/DataSets/RequisitosLinkados.txt"
180         ]
181     ),
182     agent=productOwnerAgente
183 )
```

Fonte: Elaborada pelo autor.

Como havia alguns requisitos duplicados, aplicamos um algoritmo para lapidar melhor o conjunto total de requisitos. Após a aplicação do algoritmo, logo em seguida, obtivemos 2796 requisitos agrupados em 10 tópicos, a mesma quantidade de tópicos quando os comentários foram submetidos aos algoritmos com ZS. Observe figuras 20

e 21.

Figura 20 - Agente para remoção de ruídos, …

```
# Agente CrewAI
dedupe_agent = Agent(
    role="Analista de Qualidade de Requisitos (Deduplicação)",
    goal="Remover requisitos duplicados (exatos e quase duplicados) preservando comentários e tópicos no CSV final e gerar rel
    backstory=(
        "Você é responsável por realizar a limpeza de requisitos em um dataset gerado a partir de comentários, "
        "reduzindo redundância e mantendo rastreabilidade (comentários e tópicos)."
    ),
    verbose=True,
    llm=openai_llm
)

```

Fonte: Elaborada pelo autor.

Figura 21 - Remoção de requisitos duplicados

```
dedupe_task = Task(
    description=(
        "Leia o CSV de entrada, remova requisitos duplicados (exatos e quase duplicados por similaridade), "
        "agrupando comentários e tópicos. Gere CSV final e um PDF resumindo o processo e exibindo uma amostra."
    ),
    expected_output=f"Arquivos '{OUTPUT_CSV}' e '{OUTPUT_PDF}'.",
    agent=dedupe_agent,
    callback=run_pipeline
)
```

Fonte: Elaborada pelo autor.

Em média, para cada comentário, foram extraídos pelo menos 2 tipos principais de requisitos, sejam RF e RNF, e para alguns, subdivisões desses requisitos. Observou-se também que, para o mesmo comentário, foram extraídos outros subtipos de requisitos. Observe as Figuras 22 e 23.

1. Figura 22 - Alguns requisitos, com os devidos comentários e tópicos. Na sequência, por colunas, temos: Índice, Requisito, Comentário, Tópico e quantidade de registros por clúster.

index	requisito	comentarios	topicos	qtdRegistrosCluster
0	1. Access to Internet Banking	Problema que está acontecendo todos os dias após esta última atualização...Ao entrar pelo Link do Internet Bank o acesso é normal + aí vem os problemas, porque na compra de um Ativo tudo ok e quando vai vender ...mensagens de erro que está sem assinatura eletrônica se está salva e não autoriza vc vender já tive que desconectar dvs vezes para consegui agendar a venda e se precisa abrir outra tela no app celular cai a conexão, caso continue assim irei tirar a minha conta da cx	5	1
1	1. Accessibility Improvement The system must allow users to access their account without failures, ensuring that access is consistently available.	Tem que melhorar muito pra ficar razoável. Estou desde sexta tentando acessar sem sucesso. Quando consigo acessar, as telas aparecem todas em branco. Mais um final de semana sem acessar minha carteira. CAIXA, vamos melhorar isso, senão, os clientes sairão todos.	2	1
2	1. Account Balance Inquiry	Caixa econômica Federal facilitadora de saldos consultar conta poupança . Entre todos os que já testei, é o da caixa é o melhor.o layout é limpo e claro é o aplicativo que mais proveTudo posso na queli q , mim fortalecer	1	1
3	1. Account Creation	Conta poupanca	2	1
4	1. Account Creation Workflow	Estou tentando abrir uma conta (13) poupança na Caixa Econômica Federal agência 2869 Mogi das Cruzes,pelo app. mas estou tendo dificuldades, adicionei meu nome,CPF e data do nascimento, tipo da conta e agencia, mas no final me pede o número da conta que está no cartão, sendo que ainda nem abri a conta, como devo proceder??. Obrigada	1	1
5	1. Account Verification	Me decepcionei, só pode investir se for na conta corrente 001 e eu não tenho. ?	1	1
6	1. Alert System Functionality	A opção Alertas não funciona e parou de enviar e-mail, me obriga a acompanhar as posições todo santo dia! Não encontrei notas de negociação, embora indicada nos termos de uso disponíveis no "aplicativo minha conta", nem qualquer ferramenta com histórico. Tutorial com vídeos privado. Informação para declarar IR pouca e dispersa. Home broker parou de conectar no meu PC qualquer que seja o navegador utilizado no passado, acesso pelo app que é razoável, pode melhorar a conectividade.	1	1

Fonte: Elaborado pelo autor.

Figura 23 - Alguns requisitos, com os devidos comentários e tópicos. Na sequência, por colunas, temos: Índice, Requisito, Comentário, Tópico e quantidade de reg. por clúster.

38	1. Performance The application must load within 2 seconds and support at least 1000 concurrent users without performance degradation.	Esperei muito pelo app home broker Caixa. Espero que ele continue evoluindo e que fique tão bom que eu não precise mais do computador.	1	1
39	1. Performance The system should process user feedback and complaints within two seconds to ensure a responsive user experience.	Sou cliente, tem que melhorar e muito!!! Passei raiva.	7	1
40	1. Performance The system shall load within 3 seconds to ensure a smooth user experience.	Pior site, esta sempre com problemas, não consegui acessar p comprar ações	2	1
41	1. Performance The system must respond to user login attempts within 2 seconds to ensure a	Não consigo acessar está sistema, ele e muito ruim.	3	1

Fonte: Elaborado pelo autor.

As duas figuras anteriormente, tem o propósito de previamente facilitar o entendimento do resultado do que foi proposto em todo o projeto, mais resultados, observe a apêndice A. Ainda sobre este projeto, temos os seguintes pontos convergentes: a qualidade; a cobertura e natureza dos requisitos extraídos e principalmente identificar padrões recorrentes que evidenciem problemas e expectativas dos usuários em relação ao APP analisado para extrair requisitos.

Cada registro ou linha pode conter:

- Quanto à classificação dos requisitos
  a) RF

     Criações de contas;

     Autenticação por senha e leitura de digitais;

     Compras e vendas;

     Visualização de saldo, extratos, …;

     Leitura de QR e código de barras para validação de documentos e dispositivos;

     Emissões de alertas, notificações, …

  b) RNF

     Quanto ao desempenho - tempo de resposta, travamento, concorrência;

     Quanto à disponibilidade - falhas de acesso, indisponibilidade de alguns

serviços;

Quanto à confiabilidade - erros frequentes, falhas em movimentos financeiros;

Quanto à usabilidade - navegação confusa, interfaces pouco intuitivas;

Quanto à segurança e privacidade — biometria, criptografia e proteção de dados.

Os comentários e requisitos parecem ser mais abundantes nesta classificação de requisitos, por identificar maior quantidade de insatisfação principalmente quando os aspectos de qualidade do APP não são atendidos.

- Quanto às análises dos tópicos

Os agrupamentos, por meio do processo de clusterização, permitiram a criação de grupos de comentários semanticamente semelhantes (gráfico 2), resultando em conjuntos de dados que representam macroproblemas em um sistema. Temos que:

a) Tópicos relacionados à performance e disponibilidade abrangem problemas críticos e recorrentes percebidos pelos usuários;
b) Tópicos com menor quantidade de registros, estão relacionados a sugestões específicas, como melhorias visuais (ex.: “modo escuro”) ou ainda funcionalidades adicionais.
c) Um dos clusters apresenta 48 comentários associados, o que indica um problema crítico e recorrente percebido pelos usuários.

Esses agrupamentos evidenciam que a abordagem proposta é capaz de identificar áreas críticas do sistema, fornecendo subsídios objetivos para priorização de requisitos.

- Quanto à redundância e similaridade semântica

A análise qualitativa dos requisitos revela a existência de possíveis redundâncias semânticas, principalmente entre os RNFs. Alguns requisitos descrevem expectativas similares, variando somente em valores temporais ou forma de relação, exemplos:

- “The application must load within 3 seconds”;
- “The application must have a response time of less than 2 seconds”;
- “The system shall respond within 2 seconds”.

Entretanto, essa redundância indica consistência na percepção dos usuários, ela também evidencia a necessidade de uma etapa posterior de consolidação e

refinamento, típicas de processos clássicos de ER.

- Quanto à qualidade dos requisitos extraídos

  Temos pontos fortes e fracos.

  Pontos fortes

  - Clareza na maioria dos requisitos;
  - Forte alinhamento com problemas reais relatados pelos usuários;
  - Manutenção do contexto original por meio dos comentários associados;
  - Boa cobertura funcional e não funcional.

  Fracos e limitações observadas

  - Ausência de priorização explícita;
  - Mistura ocasional de RF e RNF em um mesmo enunciado;

- Sobre os resultados no aspecto geral

Os resultados demonstram que a abordagem proposta é eficaz na fase de elicitação automática de requisitos, especialmente em cenários com grande volume de feedback textual. A maior quantidade de RNFs reforça a importância de considerar os aspectos de qualidade desde as fases iniciais do processo de desenvolvimento de software.

Os resultados também indicam que a saída gerada deve ser compreendida como um artefato intermediário, que demanda etapas adicionais de análise, consolidação e validação. Essas atividades são geralmente desempenhadas por Engenheiros de Requisitos.

- Em resumo:

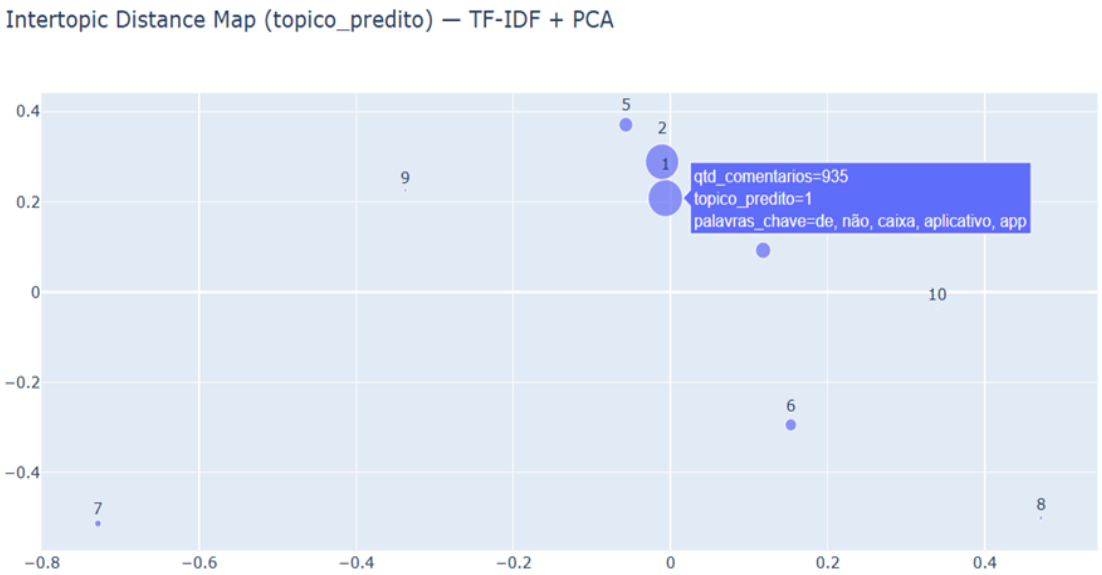
Os resultados evidenciam que: A abordagem consegue capturar, de forma automática, os comentários dos usuários, mesmo com padrões não formais; Os requisitos críticos emergem naturalmente a partir da frequência e do conteúdo dos comentários; A técnica aplicada é eficaz para a identificação de RNFs; A clusterização temática contribui para a organização e a análise dos requisitos.

No gráfico 2, temos uma visão macro das localizações dos requisitos extraídos; as localizações das circunferências indicam previsíveis classificações e a qual grupo pertence.

Na tabela 1 temos um resumo com os dados de entrada, quantidade de registros

e as quantidades de requisitos gerados em cada modelo.

Gráfico 2 - Requisitos extraídos e unificados.



Fonte: Elaborado pelo autor

Tabela 1 - com os quantitativos de dados de entrada, dados extraídos e recursos utilizados.

Modelos	Recursos	% Dados de entrada	Qtd de comentários	Qtd de req. extraídos	Qtd de tópicos
Tópicos preditos	Zero-Shot, LLM	100	229		10
RAG e LangChain	RAG, LangChain e CrewAI	100	229	229	
Gemini	Gemini e CrewAI	100	229	3660	
GPT	GPT e CrewAI	100	229	34	

Fonte: elaborada pelo autor

5.6 Observação sobre os guardrails e suas aplicações nas obtenções de resultados

Como descrito em parágrafos anteriores sobre os usos dos *guardrails* nas prevenções (trechos de implementações) de ocorrências de anomalias e alucinações nos requisitos extraídos. Segundo Ayyamperumal; Ge 2024; uma das barreiras implementadas foi a restrição de algumas palavras e outros caracteres desnecessários ou inúteis ou ainda delimitar filtros na extração de requisitos, por exemplo, a análise semântica.

Temos alguns trechos de códigos aplicados nos modelos anteriormente descritos:

- No modelo de implementação para a predição de tópicos

Figura 24 - Delimitação para a criação de sementes

```
palavras = vectorizer.get_feature_names_out()
pesos = X.mean(axis=0).A1

df_pesos = pd.DataFrame({'palavra': palavras, 'peso_medio': pesos})
df_pesos = df_pesos.sort_values(by='peso_medio', ascending=False).reset_index(drop=True)

print("\nZ.S Palavras de maior peso TF-IDF predição de tópicos:")
print(df_pesos.head(10))

dict_pesos = dict(zip(df_pesos['palavra'], df_pesos['peso_medio']))
```

Fonte: Elaborada pelo autor

Figura 25 - Resultados após a delimitação para a criação de sementes do autor

```
Foram preditos 10 tópicos distintos.
Distribuição por tópico:
  - caixa (id=1.0): 29 comentários
  - app (id=2.0): 3 comentários
  - aplicativo (id=3.0): 46 comentários
  - funcional (id=4.0): 18 comentários
  - consigo (id=5.0): 9 comentários
  - bom (id=6.0): 12 comentários
  - horrível (id=7.0): 14 comentários
  - ações (id=8.0): 23 comentários
  - conta (id=9.0): 30 comentários
  - péssimo (id=10.0): 45 comentários
```

Fonte: Elaborada pelo autor

- No PO, quando realizada a varredura nos requisitos para detectar termos indesejáveis antes de realizar a fusão.

  Veja a figura 21.

5.7 Comparações entre os modelos

Os requisitos extraídos dos comentários dos usuários apresentaram diferentes níveis de contextualização e quantidades. Alguns requisitos são bem estruturados e coerentes, em conformidade com as expectativas do nosso estudo. Isso pode ser observado no relatório gerado pelo modelo com PO (ver apêndice A), que apresenta todos os requisitos, os devidos comentários e os tópicos.

Os algoritmos e métodos aplicados — ao longo deste trabalho, aliados aos filtros e mecanismos preventivos empregados — contribuíram significativamente para a obtenção de resultados consistentes.

6 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho teve como objetivo investigar a viabilidade da extração automatizada de requisitos de software a partir de comentários de usuários disponibilizados em lojas de APPs, utilizando LLMs, técnicas de PLN, arquitetura baseada em agentes e apoio de mecanismos de *guardrails* para controle de qualidade. As conclusões aqui apresentadas são fundamentadas no Capítulo 4.

Em resposta a QP1, os resultados apresentados na subseção 4.1 demonstraram que é viável extrair requisitos de softwares diretamente de comentários de usuários, mesmo em cenários caracterizados por linguagem informal, ruído textual e ausência de estrutura sintática. Os experimentos evidenciaram que os LLMs avaliados foram capazes de identificar intenções funcionais e não funcionais relevantes, sobretudo quando precedidos por etapas de pré-processamento e normalização de textos.

Quanto a QP2 - conforme discutido nas subseções 4.2, 4.3 e 4.4, observou-se que a orquestração de agentes por meio do framework CrewAI possibilita a automação significativa do processo de elicitação e classificação de requisitos. A definição de papéis especializados e de tarefas encadeadas permitiu automatizar etapas como a extração e a classificação de RF/RNF, reduzindo a dependência de intervenção humana contínua.

Em relação a QP3 - os resultados da seção subseção 4.5 indicam que a estruturação de um workflow multiagente integrando diferentes modelos de LLM (Gemini, GPT e RAG+LangChain) mostrou-se adequada para apoiar a ER a partir de dados textuais não estruturados. A adoção de múltiplas visões independentes, posteriormente consolidadas por um agente do tipo PO e outro para realizar a manutenção, diminuindo a redundância, ambiguidade e duplicidade dos requisitos extraídos, contribuindo assim para a obtenção de maior robustez e diversidade na extração dos requisitos.

No que se refere a QP4, a análise apresentada na subseção 4.4 indica que alguns requisitos extraídos após as etapas de refinamento e validação apresentam níveis satisfatórios de clareza e completude, conforme critérios discutidos na literatura de ER e em normas como a IEEE 29148. Observou-se, ainda, que comentários excessivamente curtos ou ambíguos tendem a prejudicar a qualidade dos requisitos.

Finalmente, QP5 - os resultados das subseções 4.5 e 4.6 - demonstram que a

incorporação de mecanismos de *guardrails* e de detecção de anomalias é essencial para reduzir alucinações e requisitos incoerentes. Estratégias como validação individual e cruzada entre agentes, rastreabilidade do comentário-requisito e filtragem semântica mostram-se eficazes para aumentar a confiabilidade do processo automatizado.

Em geral, conclui-se que a abordagem proposta é tecnicamente viável, alinhada ao estado da arte e aplicável em contextos reais, contribuindo para a automação da ER inclusive para grandes volumes de feedbacks textuais. Entretanto, as limitações, destacam-se a dependência de alguns recursos proprietários e o uso de base de dados pertencentes ao domínio do assunto, sugere-se a ampliação da amostra, a inclusão de avaliações com especialistas humanos e o aprimoramento dos recursos de guardrails.

REFERÊNCIAS

AL SHAROU, Khetam; LI, Zhebhao; SPECIA, Lucia. **Towards a better understanding of noise in natural language processing**. In: INTERNATIONAL CONFERENCE ON RECENT ADVANCES IN NATURAL LANGUAGE PROCESSING (RANLP), 2021. Proceedings[...]. Varna, Bulgaria: INCOMA ltd, 2021. p. 53-62. Disponível em: https://aclanthology.org/2021.ranlp-1.7.pdf. Acesso em 08 dez. 2025.

ALHOSHAN, Waad; FERRARI, Alessio; ZHAO, Liping. **Zero-Shot Learning for Requirements Classification: an Exploratory Study**, Cornell University, 2023. Disponível em: https://arxiv.org/pdf/2302.04723. Acesso em: 12 out. 2025.

ALVES, P S Antonio; KALINOWSKI Marcos; MENDEZ Daniel; VILLAMIZAR Hugo; AZEVEDO Kelly; ESCOVEDO Tatiana, LOPES Helio. **Industrial Practices of Requirements Engineering for ML-Enabled Systems in Brazil**. Cornell University, 2024. Disponível em: https://sol.sbc.org.br/index.php/sbes/article/view/30364. Acesso em 15 out. 2025.

AYYAMPERUMAL, Ganesh, Suriya; GE, Limin. **Current State of LLM Risks and AI Guardrails**. Cornell University 2024. Disponível em: https://arxiv.org/pdf/2406.12934. Acesso em 28 out. 2025.

BERGMANN, Dave. **O que é aprendizado zer-shot?**. IBM, 2024. Disponível em: https://www.ibm.com/br-pt/think/topics/zero-shot-learning. Acesso em 09 set. 2025.

BINKHONAIN, Manal; ALFAYEZ, Reen. **Are prompts all you need? Evaluating prompt-based Large Language Models (LLM)s for software requirements classification**. Cornel University arXiv, 2025 Disponível em: https://arxiv.org/abs/2509.13868. Acesso em 27 out. 2025.

BHATIA, AshishSing; KALUZA, Bostjan. **Machine Learning in Java**, Second Edition, Published by Packt Publishing Ltd, B3 2PB, UK, 2018.

BHATIA, M.; SINGH, R.; SHARMA, S. ***Leveraging Sentiment and Topic Modeling for Requirement Elicitation from App Reviews***. ResearchGate, 2023. Disponível em: https://www.researchgate.net/publication/379531234_Sentiment_Analysis_for_Requirements_Elicitation_from_App_Reviews_A_Systematic_Mapping_Study . Acesso em 12 jul. 2025.

CABALLAR, Rina Diane; STRYKER, Stryker. IBM. **O que é o Google Gemini?**. Disponível em: https://www-ibm-com.translate.goog/think/topics/google-gemini?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt&_x_tr_pto=wa. Acesso em: 05 set. 2025.

CRUZ, Antonio Miguel Rosado da; CRUZ, Estrela Ferreira. **Machine Learning Techniques for Requirements Engineering**: A Comprehensive Literature Review. ResearchGate, 2025. Disponível em: https://www.researchgate.net/publication/393160847_Machine_Learning_Techniques_for_Requirements_Engineering_A_Comprehensive_Literature_Review. Acesso em 14 out. 2025.

DONG, Yi; MU, Ronghui; JIN, Gaojie; QI, Yi, HU, Jinvei; ZHAO; Xingyu; MENG, Jie; RUAN; Wenjie; HUANG; Xiapwei. **Building Guardrails for Large Language Models**. Cornell University, 2024. Disponível em: https://arxiv.org/abs/2402.01822. Acesso em 29 out. 2025.

GONCA, Canan Oztekin; GONCA, Gokce Menekse Dalveren. **STRUCTURE SRS for e-Government Services With Boilerplate Design and Interface**. IEEE XPLORE, 2023. Disponível em: https://ieeexplore.ieee.org/document/10156846. Acesso em: 05 jun. 2025.

GHOSH, Sonho; GUNNING, Dwight. **Natural Language Processing Fundamentals**. ResearchGate, 2019. Disponível em: https://www.researchgate.net/publication/341548483_Natural_Language_Processing_Fundamentals. Acesso em 12 jun 2025.

GUIDOTTI, Riccardo; MONREALE, Anna; RUGGIERI, Salvatore; TURINI, Franco; GIANNOTTI, Fosca; PEDRESCHI, Dino. **A Survey of methods for explaining black box models**. ACM Computing Surveys, w. v 51, n. 5, p. 1-42, 2018. Disponível em: https://dl.acm.org/doi/epdf/10.1145/3236009. Acesso em 11 dez. 2025.

Systems and software engineering — Life cycle processes — Requirements engineering. INTERNATIONAL STANDARD, 2011. Disponível em: https://github.com/Orthant/IEEE/blob/master/29148-2011.pdf. Acesso em 07 set 2025.

HE, Jianying; MOHD, Hafeez Osman; SA'ADAH Hassan; NG Keng Yap. **ENHANCING CLARITY INCHINESE SOFTWARE REQUIREMENTS:** A BOILERPLATE-BASED APPROACH TO IMPROVE REQUIREMENT EXPRESSION. Journal of Theoretical and Applied Information Technology, 2023. Disponível em: https://www.jatit.org/volumes/Vol102No24/5Vol102No24.pdf. Acesso em: 18 mai. 2025.

IEEE Standart for System, Software, and Hardware Verification and Validation. IEEE SA Standards Association, 2024. Disponível em: https://standards.ieee.org/ieee/1012/7324/. Acesso em: 16 mai. 2025.

JIN, Haolin; HUANG, Linghan; CAI, Haipeng; YAN, Jun; LI, Bo; CHEN, Huaming. **From LLMs to LLM-based Agents for Software Engineering:** A Survey of Current, Challenges and Future. Cornel University arXiv, 2024. Disponível em: https://arxiv.org/abs/2408.02479. Acesso em 27 out. 2025.

JURAFSKY, Daniel; MARTINS, H James. **Transformers**. Speech and Language. 3 ed. Stanford: Draft, 2023. Disponível em: https://web.stanford.edu/~jurafsky/slp3/8.pdf. Acesso em: 11 dez. 2025.

KARANIKOLAS, Nikitas; MANGA, Eirini; SAMARIDI, Nikoletta; TOUSIDOU, Eleni; VASSILAKOPOULOS, Michael. ACM Digital Library, 2024. **Large Language Models versus Natural Language Understanding and Generation**. Disponível em: https://dl.acm.org/doi/10.1145/3635059.3635104. Acesso em 10 nov. 2025.

KOJIMA, Takeshi; GU, S Shixiang; REID Machel; MATSOU, Yutaka; IWASAWA. Yusuke. **Large Language Models are Zero-Shot Reasoners**. Cornell University anXiv, 2022. Disponível em: https://arxiv.org/pdf/2205.11916. Acesso em 12 out. 2025.

KUMAR, Anshuman; MAKHIJA, Prakhar; GUPTA, Ankar. **Noisy text data: Achilles'heel ofBERT**. arXiv preprint. arXiv:2023.12932, 2020. Disponível em: https://arxiv.org/abs2003.12932. Acesso em: 08 dez. 2025.

LAMBA, Manika; MADHUSUDHAN, Margam. ResearchGate, 2022. **Sentiment Analysis. In: Text Mining for Information Professionals**: Uncharted Territory. Disponível em: https://www.researchgate.net/publication/360116605_Text_Mining_for_Information_Professionals_An_Uncharted_Territory. Acesso em: 02 ago. 2025.

LEWIS, Patrick; PEREZ, Ethan; PIKTUS, Aleksandra; PETRONI, Fabio; KARPUKHIN, Vladimir; GOYAL, Naman; YIH, Wen-tau; ROCKTÄSCHEL, Tim; RIEDEL, Sebastian; KIELA, Douwe. **Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks**. arXiv preprint arXiv:2005.11401 NeurlsPS Proceedings, 2020. Disponível em: https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html. Acesso em 12 nov. 2025.

LORENZE, Jay. **CrewAI**: open-source multi-agent orchestration framework. GitHub, 2024. Disponível em: https://github.com/joaomdmoura/crewAI. Acesso em: 18 out. 2025.

MAVROUDIS, Vasilios. **LangChain**. Alan Turing Institute, 2022. Disponível em: https://www.researchgate.net/profile/Vasilios-Mavroudis/publication/385681151_LangChain/links/67309cef77b63d1220e9a7a4/LangChain.pdf?origin=publication_detail&_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6InB1YmxpY2F0aW9uIiwicGFnZSI6InB1YmxpY2F0aW9uRG93bmxvYWQiLCJwcmV2aW91c1BhZ2UiOiJwdWJsaWNhdGlvbiJ9fQ&__cf_chl_tk=wgHjGoYBzM.kZ7AEWRHQI0ZnWMgs9kMgSo4G.YhqcoE-1762935369-1.0.1.1-xo.KylI3BXxQrhOYMPWwYk1ir2eK2dzRwMdL5kwQ3QE. Acesso em: 12 nov. 2025.

NAVEED, Humza; KHAN U Asad; QUIF, Shi; SAQIB, Muhammad; ANWAR, Saeed; USMAN, Muhammad; AKHTAR, Naveed; BARNES, Nick; MIAN, Ajmal. **A Comprehensive Overview of Large Language Models**. Cornel University, 2024. Disponível em: https://arxiv.org/pdf/2307.06435. Acesso em 12 out. 2025.

PRESSMAN, Roger S. **Engenharia de Software Uma**: Abordagem Profissional / Roger S. PRESSMAN; tradução de Ariovaldo Griesi, Mario Moro Fecchio; revisão técnica Reginaldo Arakaki, Julio Arakaki, Renato Manzan de Andrade. 7. ed. - Porto Alegre: AMGH, 2011.

RIBEIRO, Túlio Marco; SINGH, Sameer; GUESTRIN, Carlos. **Why should I trust you? Explaining the predictions of any classifier.** Cornell University, 2016. Disponível em: https://arxiv.org/abs/1602.04938. Acesso em: 11 dez. 2025.

ROTAR, Cristian; Zhang Qingyu. **A design science research approach to Large Language Model-Based Agents for Requirements Specification (LLMBA4RS) in low-code applications**. ResearchGate, 2025. Disponível em: https://www.researchgate.net/publication/395466502_A_design_science_research_approach_to_Large_Language_Model-Based_Agents_for_Requirements_Specification_LLMBA4RS_in_low-code_applications. Acesso em 10 out 2025.

STRYKER, Cole; HOLDSWORTH, Jim. **O que é NLP (procesamento de linguagem natural)**. IBM. Disponível em: https://www.ibm.com/br-pt/think/topics/natural-language-processing. Acesso em: 10 out. 2025.

SYED, Meredith; GADESHA, Vrunda. IBM, 2025. **What is a zero-shot prompt?** Disponível em: https://www.ibm.com/think/topics/zero-shot-prompting. Acesso em: 11 jul. 2025.

VOGELSANG, Andreas; BONG, Markus. **Requirements Engineering for Machine Learning**: Perspectives from Data Scientists. IEEExplore, 2019. Disponível em: https://ieeexplore.ieee.org/document/8933800. Acesso em: 27 out. 2025.

VASWANI Ashish; SHAZEER Noam; PARMAR Niki; USZKOREI Jakobt; JONES Llion; GOMEZ N. Aidan; KAISER Łukasz. arXiv:1706.03762v7, 2023. Disponível em: https://arxiv.org/abs/1706.03762. Acesso em 10 jul. 2025.

WINLAND, Vanna; SYED, Meredith; GUTOWSKA, Anna. **What is crewAI?**. IBM, 2025. Disponível em: https://www.ibm.com/think/topics/crew-ai. Acesso em: 05 jun. 2025.

WOOLDRIDGE, Michael J; JENNINGS, Nicolas R. **Intelligent Agents: Theory and Practice. The Knowledge Engineering Review**. CMU School of Computer Science, 1995. Disponível em: https://www.cs.cmu.edu/~motionplanning/papers/sbp_papers/integrated1/woodridge_intelligent_agents.pdf. Acesso em: 17 out. 2025.

WOOLDRIDGE, Michael J. **Large Language Models Miss the Multi-Agent Mark**. Cornell University, 2009. Disponível em: https://arxiv.org/pdf/2505.21298. Acesso em: 16 out. 2025.

ZHAO, Andrew; HUANG, Daniel; XU, Quentin; LIN, Mathieeu; LIU, Yong-Jin; HUANG, Gao. **Expel**: LLM Agents Are Experiential Learners. Cornel University arXiv, 2023. Disponível em: https://arxiv.org/abs/2308.10144. Acesso em: 08 nov. 2025.

ZHENG, Zibin; NING, Kaiwen; ZHONG, Qingyuan; CHEN, Jiachi; CHEN, Wenqing; GUO, Lianghong; WANG, Weicheng; WANG, Yanlin. **Towards an Understanding of Large Language Models in Software Engineering Tasks**. Cornell University arXiv, 2023. Disponível em: https://arxiv.org/abs/2308.11396. Acesso em: 27 out. 2025.

APÊNDICE A - REQUISITOS EXTRAÍDOS

Requisitos no contexto do comentário do usuário

Modelo LLM: dfRequisitosGeradosGeminiLimpo.csv

Comentario: Vários travamentos, serviço indisponível, impossibilidade de personalizar os ativos de interesse e interface muito confusa são apenas alguns dos problemas que o app precisa corrigir. É vergonhoso perto de soluções apresentadas pela concorrência.

Topico: 1

Requisitos Soft: NonFunctional Requirements

Modelo LLM: dfRequisitosGeradosGeminiLimpo.csv

Comentario: Vários travamentos, serviço indisponível, impossibilidade de personalizar os ativos de interesse e interface muito confusa são apenas alguns dos problemas que o app precisa corrigir. É vergonhoso perto de soluções apresentadas pela concorrência.

Topico: 1

Requisitos Soft: 1. The application must maintain a performance level that allows for quick response times under normal usage conditions.

Topico: 1

Requisitos Soft: 2. Enhanced Transaction History

Modelo LLM: dfRequisitosGeradosGeminiLimpo.csv

Comentario: A diferença desse app com o mercado pago, por exemplo, é grande. Não sei se é desinteresse em investir am algo decente, mas a mudança foi pintura de carroça velha. App lento, serviço de extrato inútil, pois não identifica nada, como outros apps de banco, dificuldade em processar login com digital, burocracia digital inútil para fazer transferência etc.

Topico: 1

Requisitos Soft: The application must provide a detailed and intuitive transaction history that allows users to easily identify and categorize transactions.

Modelo LLM: dfRequisitosGeradosGeminiLimpo.csv

Comentario: A diferença desse app com o mercado pago, por exemplo, é grande. Não sei se é desinteresse em investir am algo decente, mas a mudança foi pintura de carroça velha. App lento, serviço de extrato inútil, pois não identifica nada, como outros apps de banco, dificuldade em processar login com digital, burocracia digital inútil para fazer transferência etc.

Topico: 1

Requisitos Soft: 3. Biometric Login

Modelo LLM: dfRequisitosGeradosGeminiLimpo.csv

Comentario: A diferença desse app com o mercado pago, por exemplo, é grande. Não sei se é desinteresse em investir am algo decente, mas a mudança foi pintura de carroça velha. App lento, serviço de extrato inútil, pois não identifica nada, como outros apps de banco, dificuldade em processar login com digital, burocracia digital inútil para fazer transferência etc.

Topico: 1

Requisitos Soft: 2. Enhanced Transaction History

Modelo LLM: dfRequisitosGeradosGeminiLimpo.csv

Comentario: A diferença desse app com o mercado pago, por exemplo, é grande. Não sei se é desinteresse em inve
stir am algo decente, mas a mudança foi pintura de carroça velha. App lento, serviço de extrato inútil, pois n
ão identifica nada, como outros apps de banco, dificuldade em processar login com digital, burocracia digital
inútil para fazer transferência etc.

Topico: 1

Requisitos Soft: The application must provide a detailed and intuitive transaction history that allows users t
o easily identify and categorize transactions.

Modelo LLM: dfRequisitosGeradosGeminiLimpo.csv

Comentario: A diferença desse app com o mercado pago, por exemplo, é grande. Não sei se é desinteresse em inve
stir am algo decente, mas a mudança foi pintura de carroça velha. App lento, serviço de extrato inútil, pois n
ão identifica nada, como outros apps de banco, dificuldade em processar login com digital, burocracia digital
inútil para fazer transferência etc.

Topico: 1

Requisitos Soft: 3. Biometric Login

Modelo LLM: dfRequisitosGeradosGeminiLimpo.csv

Comentario: A diferença desse app com o mercado pago, por exemplo, é grande. Não sei se é desinteresse em inve
stir am algo decente, mas a mudança foi pintura de carroça velha. App lento, serviço de extrato inútil, pois n
ão identifica nada, como outros apps de banco, dificuldade em processar login com digital, burocracia digital
inútil para fazer transferência etc.

Topico: 6

Requisitos Soft: 1. The application must have an intuitive user interface that accommodates users unfamiliar w
ith the software.

Modelo LLM: dfRequisitosGeradosGeminiLimpo.csv

Comentario: Não sei como funciona ainda não usei

Topico: 6

Requisitos Soft: 2. The system should ensure that the onboarding process does not exceed 5 minutes to enhance
user engagement.

Modelo LLM: dfRequisitosGeradosGeminiLimpo.csv

Comentario: Não sei como funciona ainda não usei

Topico: 6

Requisitos Soft: 3. The help section must be easily accessible and load within 2 seconds to maintain user sati
sfaction.

Modelo LLM: dfRequisitosGeradosGeminiLimpo.csv

Comentario: Não sei como funciona ainda não usei

Topico: 6

Requisitos Soft: 4. The software should be compatible with multiple devices desktop and mobile to reach a wide
r audience.

Modelo LLM: dfRequisitosGeradosGeminiLimpo.csv

Comentario: Indisponível frequentemente. Não é possível verificar o movimento financeiro previsto em D0, D1 e D2 apresenta sempre resultado zero. Não é possível identificar os dividendos e jcp creditados de qual papel sã o oriundos. Histórico de ordens não é apresentado. Realmente está bastante insatisfatorio.

Topico: 5

Requisitos Soft: 2. The response time for displaying financial movements and historical orders must be under 2 seconds to ensure a smooth user experience.

Modelo LLM: dfRequisitosGeradosGeminiLimpo.csv

Comentario: Indisponível frequentemente. Não é possível verificar o movimento financeiro previsto em D0, D1 e D2 apresenta sempre resultado zero. Não é possível identificar os dividendos e jcp creditados de qual papel sã o oriundos. Histórico de ordens não é apresentado. Realmente está bastante insatisfatorio.

Topico: 5

Requisitos Soft: 3. The user interface must be intuitive and userfriendly to improve navigation and reduce con fusion related to financial data presentation.

Modelo LLM: dfRequisitosGeradosGeminiLimpo.csv

Comentario: Indisponível frequentemente. Não é possível verificar o movimento financeiro previsto em D0, D1 e D2 apresenta sempre resultado zero. Não é possível identificar os dividendos e jcp creditados de qual papel sã o oriundos. Histórico de ordens não é apresentado. Realmente está bastante insatisfatorio.

Topico: 5

Requisitos Soft: 4. The system must implement robust error handling to ensure users receive informative messag es in case of system failures or data retrieval issues.

Modelo LLM: dfRequisitosGeradosGeminiLimpo.csv

Comentario: Muito travado,sem condições de fazer o que é preciso

Topico: 3

Requisitos Soft: 4. The software shall offer a responsive design that adapts to various devices and screen siz es to ensure usability.

Modelo LLM: dfRequisitosGeradosGeminiLimpo.csv

Comentario: Muito travado,sem condições de fazer o que é preciso

Topico: 3

Requisitos Soft: 5. The software shall provide clear instructions and feedback to users during task execution.

Modelo LLM: dfRequisitosGeradosGeminiLimpo.csv

Comentario: Muito travado,sem condições de fazer o que é preciso

Topico: 3

Requisitos Soft: NonFunctional Requirements

Modelo LLM: dfRequisitosGeradosGeminiLimpo.csv

Comentario: Muito travado,sem condições de fazer o que é preciso

Topico: 3

Requisitos Soft: 1. The software shall maintain a performance level that ensures response times are under 2 se conds for any user action.

Requisitos fora do contexto dos comentários dos usuários

Total de Linhas: 4129

Modelo LLM: dfRequisitosGeradosGeminiLimpo.csv
Comentario: gosteimuitodo
Topico: 2
Requisitos Soft: Functional Requirements

Modelo LLM: dfRequisitosGeradosGeminiLimpo.csv
Comentario: gosteimuitodo
Topico: 2
Requisitos Soft: 1. The system shall provide a user interface for users to interact with the application.

Modelo LLM: dfRequisitosGeradosGeminiLimpo.csv
Comentario: gosteimuitodo
Topico: 2
Requisitos Soft: 2. The application shall allow users to create, read, update, and delete CRUD entries.

Modelo LLM: dfRequisitosGeradosGeminiLimpo.csv
Comentario: A opção Alertas não funciona e parou de enviar e-mail, me obriga a acompanhar as posições todo san to dia! Não encontrei notas de negociação, embora indicada nos termos de uso disponíveis no aplicativo minha c onta, nem qualquer ferramenta com histórico. Tutorial com vídeos privado. Informação para declarar IR pouca e dispersa. Home broker parou de conectar no meu PC qualquer que seja o navegador utilizado no passado, acesso p elo app que é razoável, pode melhorar a conectividade.
Topico: 1
Requisitos Soft: 2. Negotiation Notes Access

Modelo LLM: dfRequisitosGeradosGeminiLimpo.csv
Comentario: A opção Alertas não funciona e parou de enviar e-mail, me obriga a acompanhar as posições todo san to dia! Não encontrei notas de negociação, embora indicada nos termos de uso disponíveis no aplicativo minha c onta, nem qualquer ferramenta com histórico. Tutorial com vídeos privado. Informação para declarar IR pouca e dispersa. Home broker parou de conectar no meu PC qualquer que seja o navegador utilizado no passado, acesso p elo app que é razoável, pode melhorar a conectividade.
Topico: 1
Requisitos Soft: The application must provide users access to negotiation notes as referenced in the terms of use.