UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ANTÔNIO CARLOS MONCKEN GONÇALVES DA SILVA
THALES MONTEIRO PIERINI MACENA

Relato de experiência sobre o uso de LLM no apoio ao Desenvolvimento de Aplicações

RIO DE JANEIRO
2025

ANTÔNIO CARLOS MONCKEN GONÇALVES DA SILVA
THALES MONTEIRO PIERINI MACENA

Relato de experiência sobre o uso de LLM no apoio ao Desenvolvimento de Aplicações

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Rafael Maiani de Mello

RIO DE JANEIRO
2025

Silva, Antônio Carlos Moncken Gonçalves da
S586r Relato de experiência sobre o uso de LLM no apoio ao Desenvolvimento de Aplicações / Antônio Carlos Moncken Gonçalves da Silva e Thales Monteiro Pierini Macena. – Rio de Janeiro, 2025.

116 f.

Orientador: Rafael Maiani de Mello.

Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação)- Universidade Federal do Rio de Janeiro, Instituto de Computação, Bacharel em Ciência da Computação, 2025.

1. Inteligência artificial. 2. Desenvolvimento de software. 3. Engenharia de software. 4. ChatGPT. 5. DeepSeek. I. Macena, Thales Monteiro Pierini. II. Mello, Rafael Maiani de (Orient.). III. Universidade Federal do Rio de Janeiro, Instituto de Computação. IV. Título.

ANTÔNIO CARLOS MONCKEN GONÇALVES DA SILVA
THALES MONTEIRO PIERINI MACENA

Relato de experiência sobre o uso de LLM no apoio ao Desenvolvimento de Aplicações

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 18 de Dezembro de 2025

BANCA EXAMINADORA:

gov.br Documento assinado digitalmente
RAFAEL MAIANI DE MELLO
Data: 20/04/2026 09:44:45-0300
Verifique em https://validar.iti.gov.br

Rafael Maiani de Mello
Doutor (UFRJ)

gov.br Documento assinado digitalmente
CLAUDIO MICELI DE FARIAS
Data: 20/04/2026 11:43:24-0300
Verifique em https://validar.iti.gov.br

Claudio Miceli de Farias
Doutor (UFRJ)

gov.br Documento assinado digitalmente
DANIEL SADOC MENASCHE
Data: 19/04/2026 00:13:16-0300
Verifique em https://validar.iti.gov.br

Daniel Sadoc Menasché
Doutor (UFRJ)

AGRADECIMENTOS

À família, especialmente a Mara Moncken Gonçalves, Aline Monteiro Barone, Christian Pierini Macena e Gabriele Batista Melo Pierini Macena: cujo sacrifício, suporte emocional e incentivo constante foram indispensáveis ao longo de toda a trajetória universitária.

Ao professor Rafael Maiani de Mello, nosso orientador durante a elaboração do trabalho, cuja visão foi indispensável para o resultado que obtivemos e ao João Antonio Recio da Paixão, nosso professor de muitas disciplinas antes deste trabalho, por nos mostrar como a Ciência da Computação pode ser gratificante.

Por fim, manifestamos profundos agradecimentos aos colegas e amigos que, direta ou indiretamente, contribuíram com sugestões, revisões e apoio técnico durante toda a jornada acadêmica, ressaltando especialmente Luís Fernando Garcia Jales e Lucas Rufino Martelotte, cujo conhecimento se provou uma ajuda essencial para a conclusão da graduação.

"Nanos gigantum humeris insidentes."

William of Conches's

RESUMO

Este Trabalho de Conclusão de curso aborda a realização de um experimento de criação de uma aplicação utilizando Large Language Model – um modelo de linguagem treinado com grandes volumes de texto para entender e gerar linguagem naturals (LLMs), como principal ferramenta de trabalho, partindo dos primeiros passos possíveis até a criação de especificações de APIs. A proposta deste trabalho é averiguar a capacidade de um modelo do tipo de avaliar os requisitos, tomar decisões e converter linguagem natural em artefatos técnicos, produzindo o máximo possível de material com ótima qualidade, de forma a simular a maior quantidade possível de trabalho de um engenheiro ou equipe de desenvolvimento. A metodologia consiste em analisar a forma mais adequada de fornecer as informações de entrada ao modelo e inspecionar a qualidade das saídas, avaliando as respostas e quantificando o número de alterações necessário para retificar possíveis respostas incorretas. A conclusão geral do experimento é que tal prática pode ajudar a reduzir grandes quantidades de carga de trabalho repetitiva, mas demanda esforço, atenção e competência de qualquer profissional que faça uso deste método em sua rotina de trabalho para impedir que informações incorretas cheguem ao produto final.

Palavras-chave: inteligência artificial; desenvolvimento de software; engenharia de software; ChatGPT; DeepSeek.

ABSTRACT

This Final Course Project addresses the creation of an application using LLMs as the main work tool, starting from the first possible steps up to the creation of API specifications. The aim of this work is to investigate the ability of a model of this type to assess requirements, make decisions, and convert natural language into technical artifacts, producing as much high-quality material as possible, thus simulating the greatest possible amount of work by an engineer or development team. The methodology involves analyzing the most appropriate way to provide input information to the model and inspecting the quality of the outputs, evaluating the responses and quantifying the number of changes necessary to correct potentially incorrect responses. The overall conclusion of the experiment is that this practice can help reduce large amounts of repetitive workload, but it requires effort, attention, and competence from any professional who uses this method in their routine work to prevent incorrect information from reaching the final product.

Keywords: artificial intelligence; software development; software engineering; ChatGPT; Deepseek.

LISTA DE ILUSTRAÇÕES

LISTA DE CÓDIGOS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

Este trabalho consiste em um relato de experiência de uso de LLMs para a criação de uma aplicação, abrangendo diversas etapas do processo, desde a avaliação de requisitos até as bases que serão utilizadas para a implementação de código. Existem vários tipos de implementações de LLMs, como o uso de Workflow de LLMs e Agente Autônomo para construções de aplicações voltadas para engenharia de código, como completude ou padrões Evaluator e Optimizer. Nesse trabalho focaremos em utilizar aplicações prontas e pequenos experimentos ao invés de desenvolver pipelines complexos.

Há uma vasta gama de estudos que medem a Acurácia – mede a proporção de acertos de um modelo de classificação (ACC) e a Precisão – mede quantas das previsões positivas estavam corretas (Precisão), das respostas geradas por esses modelos no que se refere à implementação de código e à solução de desafios algorítmicos e de estruturas de dados, conhecidos popularmente como Plataforma de desafios algorítmicos usada para treinar lógica, algoritmos e estruturas de dados (LeetCode). Existem alguns artigos que fazem um apanhado geral sobre o estado das LLMs como o (ZHAO et al., 2025), outros são focados em técnicas de engenharia de prompt para melhorar a assertividade como o (CHEN et al., 2024). Por fim, o trabalho mais relacionado com este é o (HOU et al., 2024) que é uma revisão sistemática sobre o emprego de LLMs em diferentes aplicações de Engenharia de software.

Desejamos, com este experimento, ir além da mera implementação de código ou implementações de aplicações específicas de LLM para explorar o uso de LLMs na inspeção de inconsistências em US. Passando pela modelagem de bancos de dados, na especificação de APIs e, por fim, definindo as bases para a geração de código propriamente dita. Também pretendemos avaliar, em cada uma dessas etapas, as métricas obtidas, por meio de inspeção e avaliação manuais, bem como utilizando outra LLM como instrumento de verificação.

Embora o conceito de modelo de linguagem não seja recente, podendo ser implementado de diversas formas, as LLMs surgiram mais recentemente. A arquitetura Transformer, base das LLMs, foi proposta por Vaswani et al. (2017) no artigo da Google *Attention Is All You Need* e implementada pela primeira vez no modelo BERT (DEVLIN et al., 2018).

Evidências iniciais na área sugerem que o uso de LLMs é extremamente promissor em atividades como agilizar o desenvolvimento de software, pois reduz a complexidade sintática e automatiza tarefas repetitivas de baixo nível técnico. Não é incomum encontrar levantamentos que indicam que muitos usuários preferem utilizar LLMs em vez de

provedores de busca tradicionais para a obtenção de informações diversas como demostra (Adobe, 2025).

No cenário atual, circulam diversas opiniões sobre o futuro uso da Inteligência Artificial – ramo da computação que desenvolve sistemas capazes de simular comportamentos inteligentes (AI), que vão desde posturas céticas em relação à sua adoção até afirmações de que muitos empregos serão substituídos pela automação com AI ou Agente Autônomo. A maioria dessas opiniões tem caráter pessoal ou faz parte de campanhas de marketing de empresas que oferecem tais serviços, (WALLACE, 2024) e (DIFELICIANTONIO, 2024).

À medida que os modelos evoluem, revisões sistemáticas de suas métricas rapidamente se tornam desatualizadas, restando apenas os resultados fornecidos pelos desenvolvedores como referência para avaliar até onde as LLMs podem chegar. Nesse sentido, uma das motivações deste trabalho é focar não apenas em métricas, mas em reunir informações e propor experiências que mostrem o poder e as limitações da arquitetura de *transformer*.

1.2 OBJETIVO

A aplicação proposta neste trabalho consiste em avaliar o emprego de LLMs em diferentes etapas do processo de desenvolvimento de software. Neste sentido, utilizamos como objeto de estudo o desenvolvimento de um *marketplace* genérico, utilizado como exemplo de aplicação robusta, com diversos processos e regras de negócio, mas com um número suficiente de referências existentes para servir de apoio à concepção de Oráculos. Desta forma, teremos um projeto suficientemente próximo da realidade e complexo para possibilitar a análise das limitações das LLMs no entendimento de linguagem natural e na conversão para linguagem técnica, além das limitações técnicas de partes fundamentais do funcionamento do modelo, como Context Window e pelo uso de Retrieval-Augmented Generation – técnica que combina busca de informações externas com geração de texto por modelos de linguagem (RAG). Os modelos empregados neste trabalho são LLMs *decoder-only*, com foco principal nos modelos GPT-4o e GPT-4-Turbo e, em caráter secundário, no DeepSeek-V3. Para todos os modelos citados, foram utilizadas suas versões gratuitas.

Para este fim, adotamos e adaptamos métricas para realizar avaliações de desempenho das LLMs nas diferentes etapas do projeto e também destacamos os principais problemas estruturais dos modelos, que vão além do poder de processamento e do tamanho das arquiteturas atuais. Um exemplo digno de nota é o fato de que, uma vez treinado, o modelo não dispõe de memória persistente. Cada nova iteração depende da Context Window, a qual possui um limite que, quando ultrapassado, pode ocasionar inconsistências. Outra característica relevante é que os modelos são treinados para aprender correlações a partir de grandes volumes de dados textuais, mas não possuem capacidade de raciocínio abstrato, senso comum ou intenção. Isso pode levar à geração de respostas incorretas, por exemplo, quando solicitados a contar a quantidade de ocorrências de um termo em um texto ou

realizar operações matemáticas simples. Tais limitações serão elucidadas e abordadas na parte de fundamentação teórica, que pode ser vista no Capítulo 2 e serão demostrados na prática no Capítulo 3.

No Capítulo 3, apresentaremos em detalhe as etapas deste relato de experiência, explorando seu desenvolvimento e classificando os prompts de entrada e suas respectivas respostas. O modelo de classificação dos prompts segue o apresentado em (WHITE et al., 2023a) e pode ser consultado no apêndice, juntamente com as tabelas de avaliação e classificação. Os resultados empíricos, devidamente avaliados e classificados, são discutidos no Capítulo 4.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 LLM

O nome LLM (Large Language Model, "modelo de linguagem ampla"em Português) compreende uma categoria de modelos de linguagem que é definido pela quantidade particularmente grande de dados que são utilizados em seu processo de treinamento.

A ideia básica por trás de qualquer modelo de linguagem é ser capaz de gerar saídas que façam sentido através da probabilidade de uma palavra aparecer em uma determinada sequência de palavras (ZHAO et al., 2025). Expandindo o conceito, a forma como a linguagem natural funciona remete a certos padrões de construção, onde certas palavras ou frases tendem a ser utilizadas, frequentemente, em contextos semelhantes. Desta forma, ao analisar palavras ou pedaços de frases, que são chamados de tokens Tokens, é possível reproduzir um comportamento de criação de frases coerentes e compreensíveis.

O desenvolvimento dos modelos de linguagem progrediu através das últimas décadas, se iniciando com o surgimento dos modelos estatísticos, que utilizavam Cadeias de Markov como ferramenta básica para seus cálculos estatísticos de previsão. Apesar de funcionarem muito bem para tarefas específicas, como geração de horóscopo, estes modelos possuem um problema crítico, pois um grande aumento na quantidade de Tokens de entrada utilizados gera um enorme impacto na complexidade da aplicação. (ZHAO et al., 2025)

Com o passar do tempo e o avanço das pesquisas, a evolução dos modelos de linguagens passou por diversas etapas até tomar a forma que vemos nos dias de hoje. As LLMs são derivadas das Pre-treined Language Model - um modelo de linguagem pré-treinado e refinado para entender e gerar linguagem natural (PLM)s, que são modelos caracterizados por serem pré-treinados baseados na arquitetura Transformer e, depois, submetidos ao processo de Fine-tuning. A principal diferença entre as LLMs e as PLMs se dá na escala, isto é, quantidade de dados utilizados no processo de treino e no tamanho do modelo. As LLMs podem, de certa forma, serem compreendidas como PLMs de enorme escala.

2.1.1 Tipos de LLM

As LLM derivam do modelo Transformer, que segue uma arquitetura composta por uma pilha de camadas de autoatenção e camadas totalmente conectadas (*feed-forward*) para o codificador (encoder) e o decodificador (decoder), conforme proposto por Vaswani et al. (2017). As camadas de encoder são responsáveis por codificar a sentença de entrada, capturando as relações contextuais entre os Tokens. Já as camadas de decoder geram o texto de saída de forma autoregressiva, realizando a predição do próximo Token com base nos anteriores.

De acordo com (PAN et al., 2024), as LLM podem ser classificados, com base na utilização das camadas de encoder e decoder, em três arquiteturas distintas: *encoder-only*, *encoder-decoder* e *decoder-only*. Essa taxonomia organiza os principais modelos de linguagem de acordo com a estrutura do Transformer que empregam.

Modelos do tipo *encoder-only*, como o próprio nome sugere, utilizam exclusivamente as camadas de encoder. Nesse arranjo, o encoder processa toda a sequência de entrada e gera representações contextuais com atenção bidirecional, isto é, analisando os Tokens à esquerda e à direita dentro da Context Window. Devido a essas características, tais modelos são especialmente adequados para tarefas que envolvem compreensão profunda do texto, como classificação, extração de informação, detecção de falhas em código e reconhecimento de padrões incorretos.

Modelos do tipo *decoder-only*, por sua vez, utilizam apenas as camadas de decoder. Essa é atualmente a arquitetura mais popular em número de aplicações e citações, conforme demonstrado em (HOU et al., 2024). Esse tipo de LLM será o foco deste trabalho, uma vez que os modelos gerativos mais avançados, como GPT-4-turbo, GPT-4o e DeepSeek-V3, utilizam exclusivamente a arquitetura decoder-only. Como esses modelos não possuem um módulo separado de codificação, não há pré-processamento explícito da Context Window; em vez disso, a geração começa diretamente com o primeiro Token, interpretado como continuação do contexto fornecido no prompt inicial.

"Essa abordagem depende muito da capacidade do modelo de entender e antecipar a estrutura, a sintaxe e o contexto da linguagem." (HOU et al., 2024, p. 11)

Por empregarem apenas o módulo de decoder, esses modelos se destacam em tarefas generativas, como geração de código, exemplos de uso, casos de teste e interação em linguagem natural por meio de diálogos.

Já os modelos encoder-decoder combinam ambos os módulos. Nesse caso, o encoder transforma a Context Window em uma representação intermediária latente, que é então utilizada pelo decoder para gerar a sequência de saída, token a token. Essa arquitetura foi originalmente proposta por Vaswani et al. (2017) e é amplamente utilizada em tarefas de tradução automática, sumarização e síntese de texto. No domínio da Engenharia de Software, essa estrutura é aplicada, por exemplo, à sumarização de código, tradução entre linguagens de programação e geração automatizada de documentação.

Cada uma dessas arquiteturas possui vantagens e limitações em diferentes contextos. Modelos encoder-only são eficientes em tarefas de classificação e compreensão textual, mas são limitados em geração, pois geralmente produzem apenas rótulos ou trechos condicionais. Modelos encoder-decoder, que oferecem maior equilíbrio entre compreensão e geração, são considerados ideais para tarefas que requerem transformação de entrada em saída contextualizada; no entanto, sua maior complexidade estrutural acarreta maior custo computacional. Por fim, modelos decoder-only são altamente eficazes em geração de texto e se adaptam bem a aplicações interativas, mas são tipicamente unidirecionais

e, por não contarem com um módulo dedicado à codificação da entrada, podem gerar conteúdo impreciso Hallucination quando não são fornecidos prompts bem especificados.

2.1.2 Tokens

Como é de se esperar, uma LLM não possui a capacidade de compreender de forma plena o significado de uma palavra ou frase em linguagem natural. Para que o modelo seja capaz de processar os Prompts, é necessário que a entrada seja subdividida em unidades de pequenos trechos de textos, podendo, estes, isoladamente, compreender palavras que façam sentido ou não. Estas unidades, que são a quantidade mínima de texto que o modelo processa de forma independente, são chamadas de Tokens.

O processo de transformação do texto de entrada em Tokens é chamado de *tokenização*, e pode ser feito através de alguns tipos diferentes de algoritmo. O modelo GPT, que é o principal foco deste experimento, utiliza, desde a verão 2, o *Byte-Pair Encoding* como algoritmo de tokenização. Este algoritmo consiste em partir de um conjunto de símbolos básicos e combinar pares de Tokens consecutivos frequentes como um novo Token. Este processo se repete de forma iterativa até que o tamanho do conjunto alcance um valor pré-determinado. (ZHAO et al., 2025)

Após a *tokenização*, cada Token é convertido em um vetor numérico chamado *embedding*. Esses vetores têm sempre a mesma dimensão, normalmente indicada por d_{model}. Para que o modelo saiba em que posição cada Token aparece, é somado a cada embedding um vetor de posição. Assim, obtemos uma matriz de tamanho $n \times d_{\text{model}}$ - sendo n o número de Tokens na entrada - que é enviada às camadas de atenção da LLM. Nessas camadas, o modelo aplica operações matriciais para identificar como os Tokens se relacionam entre si, permitindo gerar a saída em linguagem natural (VASWANI et al., 2017).

O número de Tokens que podem ser processados de uma vez é limitado pela Context Window do modelo, que define o tamanho máximo da sequência de entrada em termos de tokens. Se a entrada gerar mais Tokenss do que a capacidade da Context Window, o excesso será truncado ou dividido em janelas sobrepostas. Na prática, isso significa que a matriz de embeddings de dimensão $n \times d_{\text{model}}$ só pode ter n até o valor máximo da Context Window.

Os *positional embeddings* usam índices de posição que vão de 1 até esse limite máximo, fornecendo ao modelo informações sobre a ordem e proximidade dos Tokens dentro da janela ativa. Dessa forma, tanto a vetorização dos Tokens (via *embeddings*) quanto a capacidade de manter o contexto dependem diretamente do tamanho da Context Window, pois ela determina quantos Tokens simultaneamente podem influenciar o cálculo das atenções e, consequentemente, a qualidade das inferências e da geração de texto pela LLM.

2.2 RAG

A sigla *RAG* corresponde a *Retrieval-Augmented Generation* (geração aumentada por recuperação), e nomeia a tecnologia responsável por permitir que a LLM acesse informações de fontes externas à sua base de dados de treinamento e à Context Window.

Como já foi definido previamente, o conhecimento de uma LLM é, a princípio, extremamente restrito à sua base de dados de treinamento prévio. Uma consequência deste fato é que, em diversos momentos, é possível a ocorrência do fenômeno Hallucination porque o modelo pode simplesmente desconhecer a informação que está sendo referenciada naquele momento.

Inicialmente, o ideia de *RAG* surgiu como parte do processo de Fine-tuning do modelo, servindo como parte do treinamento. Com o avanço dos estudos sobre LLM, entendeu-se a necessidade de ser capaz de buscar informações de outras fontes, como arquivos abertos pelo usuário no momento do uso e até mesmo a internet como um todo. (GAO et al., 2024)

A ideia principal é ser capaz de recuperar informações de uma fonte externa e adicionar à Context Window. De acordo com (GAO et al., 2024), a evolução dos estudos sobre *RAG* pode ser dividida em três estágios: (*Naive RAG*, *Advanced RAG* e *Modular RAG*), e diversos frameworks dentro das três categoriais são utilizados ao longo das etapas de Fine-tuning, pré-treino e inferência de uma LLM moderna, especialmente o GPT-4, que é o principal foco deste experimento.

2.3 MODEL CONTEXT PROTOCOL (MCP)

Model Context Protocol – protocolo que define como diferentes aplicações, ferramentas e modelos de linguagem podem compartilhar e estruturar informações de contexto de maneira padronizada (MCP), é um protocolo de interoperabilidade que padroniza a conexão entre clientes LLM e fontes externas de dados e ações, por meio de servidores especializados que expõem *recursos*, *ferramentas* e *prompts* ao ambiente de execução do modelo. Enquanto a RAG (Seção 2.2) foca em recuperar e injetar conhecimento na Context Window, o MCP descreve como descobrir, negociar e consumir tais capacidades de maneira estruturada, audível e segura, reduzindo ambiguidade na integração e favorecendo rastreabilidade de decisões em Workflow de LLMs agentivos (Anthropic, PBC, 2024a), (Anthropic, PBC, 2024b), (Anthropic, PBC, 2024c).

Do ponto de vista arquitetural, um cliente (por exemplo, um IDE com assistência de LLM) estabelece sessão com um ou mais servidores MCP. Cada servidor anuncia *capacidades* em três eixos: (i) *recursos* (e.g., arquivos versionados, URLs, coleções de documentos, catálogos de dados), (ii) *ferramentas* (operações invocáveis com contratos de entrada e saída, como busca semântica, execução de consultas, validação de esquemas) e (iii) *prompts* pré-curados que encapsulam práticas recomendadas para tarefas recorrentes.

A negociação é mediada por esquemas de mensagens e metadados de versão e autorização, permitindo que o cliente componha chamadas determinísticas e que a execução seja registrada com granularidade suficiente para auditoria e reprodução. Ao tornar os contratos explícitos e tipados, o MCP ajuda a mitigar Hallucinations associadas a chamadas livres e não validadas, aproximando a geração do modelo a um espaço de soluções mais restrito e verificável.

Por exemplo é possível conectar um MCPs para consultar uma base de dados, ou uma API externa, através do prompt o Agente Autônomo determina se a tarefa exige uma consulta ou ação em um MCP e as executa, integrando sua resposta como parte do Context Window.

Porém um Agente Autônomo não está restrito apenas a RAG ou MCP, o mesmo pode possuir APIs internas para realizar ações concretas através de seu ambiente executor, como por exemplo APIs de **readFile** ou **writeFile**.

2.4 PROMPT ENGINEERING

A Engenharia de Prompt é a área de pesquisa responsável por estudar técnicas de construção de Prompts e seu impacto no resultado final. Com o enorme crescimento na qualidade e na usabilidade das LLMs, que surja o questionamento de utilizá-las de forma efetiva para se obter o resultado desejado. É possível impor regras sobre o padrão de resposta, atribuir exemplos, exigir tipos específicos de comportamento e uma infinidade de outras possibilidades que, quando aplicadas, podem criar alterações drásticas nas respostas fornecidas pelo modelo.

Uma abordagem interessante sobre o assunto é a de (WHITE et al., 2023b), que busca trazer conceitos de padrões de software e adaptar ao contexto das LLMs. Segundo o artigo, pode-se descrever um padrão de software através de um nome, uma classificação, uma intenção (o propósito que o padrão se propõe a alcançar), uma motivação (um problema que o padrão se propõe a resolver), uma estrutura que define classes e objetos e como estes se relacionam para criar uma solução, um código de exemplo e uma lista sucinta de prós e contras que podem surgir em decorrência de uma aplicação prática deste padrão proposto. Desta forma, traduzindo o conceito para o contexto de Prompts para uma LLM, é possível reaproveitar, basicamente, todas as ideias, substituindo apenas, naturalmente, os conceitos diretamente ligados a código propriamente dito por conceitos de construção de pensamentos coerentes. Sendo assim, ao invés de elementos como classes e objetos, temos "ideias chave", ou seja, coisas que possuem um papel de destaque no raciocínio a ser construído, e, ao invés de exemplos de código, entram exemplos de Prompts. Os prompts utilizados durante a metodologia deste trabalho usaram este padrão de descrição e podem ser encontrados no Apêndice B.

De forma mais abrangente, existem diversas técnicas conhecidas de engenharia de

Prompt que são muito utilizadas nos dias de hoje, como *role-prompting*, que consiste em solicitar à LLM que se comporte como se estivesse interpretando um determinado papel, como, por exemplo: *"como um médico pediatra, me explique quais são as contraindicações e os possíveis colaterais sobre o uso de um medicamento X"*. Também são comuns técnicas de Reasoning, que buscam guiar a geração do output, forçando passos e etapas e realimentando a Context Window com elas, como por exemplo o Chain of Thought – técnica de construção de prompts em que o modelo é estimulado a apresentar seu raciocínio passo a passo antes de fornecer uma resposta final, aumentando a precisão em tarefas complexas (CoT). Além de técnicas específicas, também existem certas convenções na forma de escrita, como separar trechos que possuem contextos diferentes utilizando um trio de aspas duplas e procurar escrever os Prompts da forma mais explicada e específica possível. Neste trabalho, usaremos algumas destas técnicas de engenharia de Prompt e as abordaremos de forma mais detalhada mais adiante neste documento.

2.4.1 N-Shots

Vamos utilizar a nomenclatura *N-Shots* para sintetizar três técnicas de engenharia de prompt que possuem uma natureza fundamentalmente semelhante: *zero-shot*, *one-shot* e *few-shots* (CHEN et al., 2024). Dito isto, o *N-Shots* consiste na técnica de fornecer, no próprio prompt, N exemplos do que se espera receber como resposta da LLM.

É comum se utilizar de exemplos como ferramenta para explicar todo o tipo de coisas até mesmo em conversas corriqueiras entre duas ou mais pessoas, e este conceito cai como uma luva quando se envia um Prompt a uma LLM. Sabendo-se que o modelo utilizado é de alta capacidade, ou tendo em mente uma tarefa ou pergunta simples, utilizar um único exemplo de saída pode ser o suficiente para que a resposta obtida seja satisfatória. Este seria o conceito de *one-shot*.

Naturalmente, na ocorrência de uma tarefa ou assunto com maior nível de complexidade ou especificidade, ou até mesmo devido à capacidade do modelo utilizado não ser tão confiável, podemos recordar da ideia citada anteriormente neste trabalho sobre procurar fornecer sempre a maior quantidade possível de detalhes no Prompt. Sendo assim, a ideia do *few-shots* consiste em fornecer uma quantidade arbitrária (maior do que 1) de exemplos, na tentativa de fornecer ao modelo contexto suficiente para uma saída mais precisa. Um ponto negativo de fornecer muitos exemplos é ocupar a Context Window o que pode acarretar no truncamento de tokens importantes do histórico.

Por fim, temos o *zero-shot*, com a proposta de que exemplos, nem sempre, ajudam. O artigo (WHITE et al., 2023b) mostra que fornecer exemplos no Prompt de entrada acabe confundindo a LLM até mais do que ajudando. Ou seja, para alguns cenários, seja mais eficiente apenas deixar que o modelo tente gerar a resposta apenas com base na sua própria capacidade de tomada de decisões ao invés de tentar manipular a saída para que obedeça um determinado padrão ou formato.

2.4.2 Chain-of-Thought

O nome *chain-of-thought* (CoT) traduzido literalmente como *cadeia de pensamento*, é, provavelmente, o mais explicativo possível. O conceito é muito simples: organizar as ideias de tal forma que a LLM siga uma sequência específica e muito bem definida de etapas para construir a resposta, de maneira muito semelhante a seguir uma espécie de algoritmo. Como abordado em (CHEN et al., 2024), atitudes simples como dizer para a LLM "vamos pensar passo a passo"ou montar uma sequência de demonstrações com perguntas e formas específicas de responder pode guiar o modelo através de uma forma estruturada de pensamento para a obtenção de um tipo específico de resposta. Você pode ver exemplos de uso de CoT nos Prompt do Apêndice B.

Aplicações mais complexas de CoT podem ser encontradas em Workflow de LLMs, com o uso de outras técnicas de Reasoning como a *self-reflection*, esses padrões de rácio-cinio podem evoluir com a tomada de decisão oferecida por Workflow de LLMs e Agente Autônomo, porém puramente na LLM esses *ráciocinio* é baseado apenas na inferência e não operações lógicas.

2.5 CONTEXT WINDOW

Context Window (*janela de contexto* em tradução livre) é um conceito fundamental quando precisamos entender o funcionamento de LLMs. Context Window refere-se à quantidade de informação textual (isto é, quantidade de Tokens) que um modelo pode processar de uma só vez. Como uma LLM não possui memória persistente, para reutilizar interações anteriores é necessário reintroduzi-las no input como parte da Context Window. Isto significa que, ao gerar uma resposta, o modelo leva em consideração não apenas o prompt atual, mas também outras informações obtidas por RAG ou por entradas anteriores. Portanto, todo o conjunto de Tokens utilizados faz parte da Context Window.

Como pode ser visto na tabela 1, o modelo GPT-4o (OPENAI, 2025a), por exemplo, consegue processar até 128.000 Tokens na (Context Window) e 16.384 Tokens na saída. Já o GPT-4 Turbo (OPENAI, 2025b) suporta 128.000 Tokens na Context Window e apenas 4.096 Tokens na saída. Tome como exemplo um arquivo extenso do kernel Linux (TORVALDS, 2025), que possui 6.977.299 Tokens, valor, aproximadamente, 54 vezes superior à Context Window dos modelos mencionados. Desta forma, torna-se inviável que o arquivo inteiro seja analisado completamente para a geração de uma resposta.

Essa limitação da Context Window impacta tanto a quantidade de informação útil que pode ser considerada pelo modelo quanto o número de Tokens de saída nos quais a resposta deve caber. Em aplicações complexas com domínios extensos - por exemplo, o sistema operacional citado - ou requisitos que interagem entre si, em um cenário real, pode ser necessária uma Context Window de dezenas a centenas de milhões de Tokens.

Tabela 1 – Context Window de LLMs

MODELO	CONTEXT WINDOW	TOKENS MAX. DE SAÍDA
GPT-4o	128.000	16.384
GPT-4-turbo	128.000	4.096
DeepSeek-v3	128.000	8.192

Fonte: (OPENAI, 2025a), (OPENAI, 2025b), (DEEPSEEK-AI et al., 2025).

Se a quantidade de Tokens for superior àquela suportada, os modelos simplesmente rejeitam o input. Caso nem toda a informação necessária para uma resposta completa e correta seja inserida ou parte dela seja excluída, pode não ser possível obter insights precisos ou realizar alterações que considerem todo o domínio e as interações de uma aplicação, ou mesmo de várias aplicações distribuídas. Assim, a Context Window está diretamente relacionada ao fenômeno de Hallucination.

Se o número de Tokens de saída for ultrapassado, o modelo pode omitir ou resumir trechos da resposta, na expectativa de que iterações subsequentes permitam obter a resposta completa. Nesse caso, a Context Window pode ficar progressivamente ocupada pelos trechos gerados anteriormente.

A Context Window é limitada pela arquitetura de Transformer, que utiliza o mecanismo de Attention – mecanismo essencial em modelos de linguagem que permite ao modelo ponderar a relevância de diferentes partes da entrada ao gerar cada saída, possibilitando a modelagem de dependências contextuais de curto e longo alcance (ATTN) e cresce em tempo de processamento e uso de memória de forma quadrática em relação ao número de Tokens. Em outras palavras, dobrar a quantidade de Tokens de entrada praticamente quadruplica o custo computacional e a memória necessária, conforme demonstrado em (LU et al., 2025).

Janelas de contexto grandes também exigem maior tempo de processamento para inferência. Modelos que utilizam internamente CoT, como o o3 (OPENAI, 2025c), podem levar vários minutos para processar uma única tarefa. Os custos para esses modelos são maiores do que os de modelos com janelas de contexto menores, pois maior uso de recursos implica maiores custos operacionais.

A melhor forma de utilizar a Context Window é ser o mais conciso possível, incluindo apenas as informações necessárias para a execução da tarefa, em vez de inserir todo o contexto. Entretanto, nem sempre é fácil determinar quais dados são essenciais, o que pode aumentar o tempo de preparação do prompt - tempo esse que poderia ser empregado para automatizar ou executar a própria tarefa.

Existem diversas técnicas para ampliar a Context Window, como por exemplo (ZHANG; LI; LIU, 2024), as quais não serão abordadas neste trabalho. Em resumo, o gerenciamento da Context Window acompanhado do fato da sua perda de performance conforme ela au-

menta, constitui um dos principais desafios para aprimorar a inferência e a precisão na utilização de LLMs, um fenômeno apelidado de context-rot (FRAGA, 2024) (HONG; TROYNIKOV; HUBER, 2025).

2.6 HALLUCINATION

Hallucination (*alucinação* em tradução livre) em LLM refere-se à geração de conteúdo que, embora plausível, é factualmente incorreto, não fundamentado em evidências reais, que entra em conflito com a Context Window ou resulta de inferências equivocadas em razão de sua limitação.

Por exemplo, segundo Huang et al. (2025), ao ser questionado sobre as principais contribuições de Thomas Edison para a ciência e a tecnologia, um modelo pode (falsamente) afirmar que ele foi o inventor da lâmpada ou do telefone, apesar de ele apenas ter aprimorado o design da lâmpada, sendo Alexander Graham Bell o inventor do telefone.

Outros exemplos de Hallucination factuais incluem a fabricação de eventos inexistentes, como quando a LLM gera informações falsas para responder a uma solicitação mesmo que o fato nunca tenha ocorrido. Por exemplo, no contexto da engenharia de software, podemos pedir a uma LLM que gere uma lista de inconsistências em USs, e ela pode supor informações incorretas a fim de aumentar ou padronizar o número de itens retornados.

No contexto da engenharia de software, existem também outros tipos de Hallucination que são preocupantes, Huang et al. (2025) as descreve como a inconsistência de contexto e a inconsistência lógica. A primeira ocorre quando alguma informação na Context Window está incorreta ou ausente, e geralmente só pode ser identificada em contextos privados. A segunda ocorre devido à incapacidade das LLMs de realizarem raciocínio abstrato, possuírem senso comum ou captarem intenções.

Enquanto as Hallucination factuais podem ser mitigadas por métodos como RAG, a inconsistência de contexto pode depender de revisão manual, agrupamento e classificação de entradas, além do limite da Context Window.

Já as inconsistências lógicas podem ser frequentes ao solicitar tarefas de análise de contexto, o que não é o ponto forte das LLMs do tipo decoder-only, como, por exemplo, separar e listar quais classes ou USs correspondem a um fluxo específico, ou quando, no momento da geração de código, alguma lógica de negócio fica quebrada ou sem as verificações necessárias, que muitas vezes precisam ser completamente explicitadas e não deduzidas pelo senso comum, o qual seria claro até para desenvolvedores inexperientes.

Como descrito por Huang et al. (2025), algumas Hallucination podem ter origem no próprio treinamento ou na base de dados utilizada. Uma base de dados incorreta pode gerar viés e informações equivocadas, ou possuir limite de conhecimento; já o treinamento por meio de Supervised Fine-Tuning – ajuste supervisionado de um modelo de linguagem utilizando exemplos rotulados para melhorar seu desempenho em tarefas específicas (SFT)

ou Reinforcement Learning from Human Feedback – técnica que utiliza aprendizado por reforço guiado por avaliações humanas para alinhar o comportamento do modelo com expectativas humanas (RLHF) pode acarretar problemas semelhantes, dependendo de como a interação humana influencia e molda o comportamento da LLM.

Em relação à base de dados, até a escolha da linguagem do prompt pode causar distorções e Hallucination, como demonstrado em um estudo que evidencia como prompts no idioma hindi apresentam menor precisão que prompts em inglês, demonstrando que linguagens com menos recursos impactam o treinamento e as respostas de LLMs:

> *Our findings demonstrate that asking the same question in English consistently yields better responses, as evidenced by higher factual accuracy, overall scores, and language consistency metrics. This disparity underscores the advantages of resource-rich languages like English in current LLM training paradigms. (ROHERA et al., 2025, p. 5)*

Existem várias formas de detectar e mitigar Hallucinations. Alguns estudos, como Li et al. (2023), propõem benchmarks para avaliar a capacidade de LLMs em detectar Hallucinations; neste estudo, veremos adiante como uma LLM pode ser usada para avaliar a resposta de outra LLM. Além disso, usando técnicas como RAG e CoT, podemos mitigar a geração de Hallucination ao prover melhores contextos ou ao inserir etapas de validação nos processos de geração.

Hallucinations em LLMs podem ter consequências graves em aplicações de alto risco. Na medicina, informações falsas podem conduzir a diagnósticos ou tratamentos impróprios. No campo jurídico, conselhos baseados em fatos incorretos podem comprometer decisões legais. Na educação, estudantes expostos a conteúdos falsos aprenderiam conceitos incorretos. Mesmo em atendimentos ao cliente, respostas enganosas abalam a confiança dos usuários e podem causar danos comerciais significativos. Na engenharia de software, uma alteração de código não revisada pode gerar impactos como downtime das aplicações ou aumento de custos.

3 METODOLOGIA

Para a condução deste trabalho, foi concebida uma metodologia simplificada, a qual consiste em resumir o processo de construção de uma aplicação em quatro grandes etapas, propondo formas produtivas e convenientes de incorporar o auxílio de LLMs em cada uma delas. A aplicação em questão é um Marketplace capaz de realizar a venda de qualquer tipo de mercadoria, representando um exemplo realista que contempla diferentes perfis de usuários, múltiplas camadas de complexidade e os requisitos típicos de um sistema distribuído.

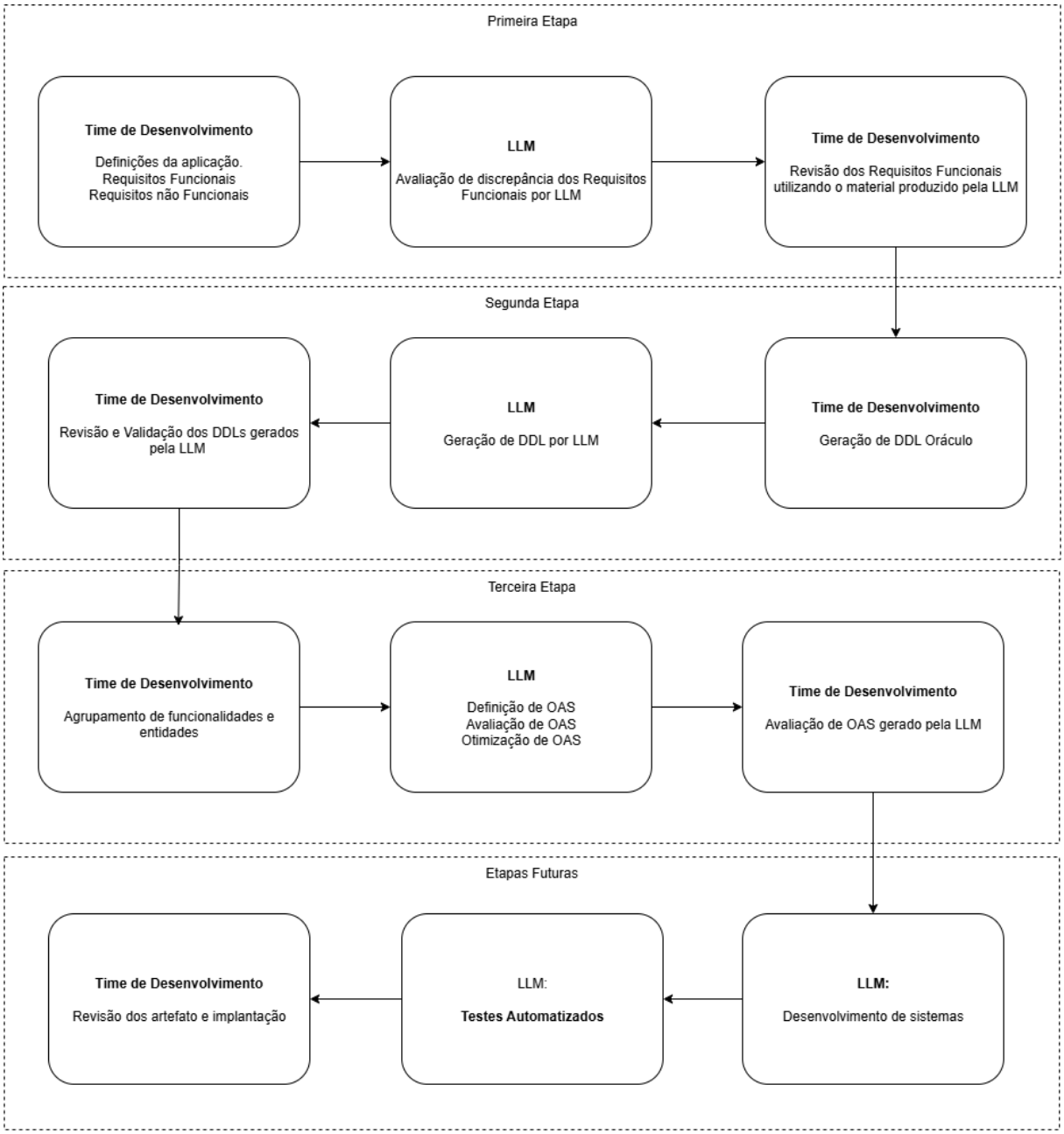
Como pode ser visto na figura 1, as três etapas principais definidas para o experimento foram: (i) levantamento de USs, analisadas posteriormente por uma LLM com o intuito de identificar inconsistências; (ii) modelagem do banco de dados e geração de Data Definition Language – conjunto de comandos SQL utilizados para definir, alterar ou remover estruturas de dados como tabelas, esquemas e índices em um banco de dados (DDL); (iii) definição OpenAPI Specification – especificação padronizada para descrever, documentar e consumir APIs RESTful, permitindo geração automática de documentação e código cliente/servidor (OAS), permitindo o desenvolvimento paralelo e coerente entre *front-end* e *back-end*.

Na *primeira etapa*, foi realizada a coleta dos requisitos do sistema, incluindo requisitos não funcionais, definição de um escopo arquitetural e elaboração preliminar das USs. Essas histórias foram inicialmente criadas sem o apoio de uma LLM. Posteriormente, foi desenvolvido um prompt com o objetivo de submetê-las à análise de uma LLM, permitindo a identificação de discrepâncias e possibilitando a revisão das histórias com base nas sugestões geradas.

A *segunda etapa* consistiu na utilização das USs revisadas para a construção de um Oráculo - um modelo de validação semelhante ao conceito de Minimum Viable Product – versão inicial de um produto com o mínimo de funcionalidades necessárias para validar sua proposta de valor com usuários reais e obter feedback rápido (MVP). A partir disso, elaborou-se um prompt que, ao receber como entrada as USs, era capaz de definir as entidades do sistema e gerar os correspondentes artefatos de definição de dados (DDLs) por meio do uso de uma LLM.

A *terceira etapa* teve como foco a organização das tabelas geradas na etapa anterior, agrupando-as por funcionalidades e vinculando-as às respectivas USs. Em seguida, foi elaborado um prompt responsável por, a partir da estrutura dos dados e do comportamento esperado, gerar uma OAS. Essa especificação, uma vez criada, era submetida à avaliação de outra instância de LLM, com um prompt distinto, que atribuía uma pontuação entre 1 e 100 pontos e fornecia sugestões de aprimoramento. Os resultados da geração e da avaliação eram então submetidos a um terceiro prompt, cujo objetivo era otimizar a espe-

Figura 1 – Fluxo de desenvolvimento da aplicação com uso de LLMs



Fonte: Elaborado pelos autores.

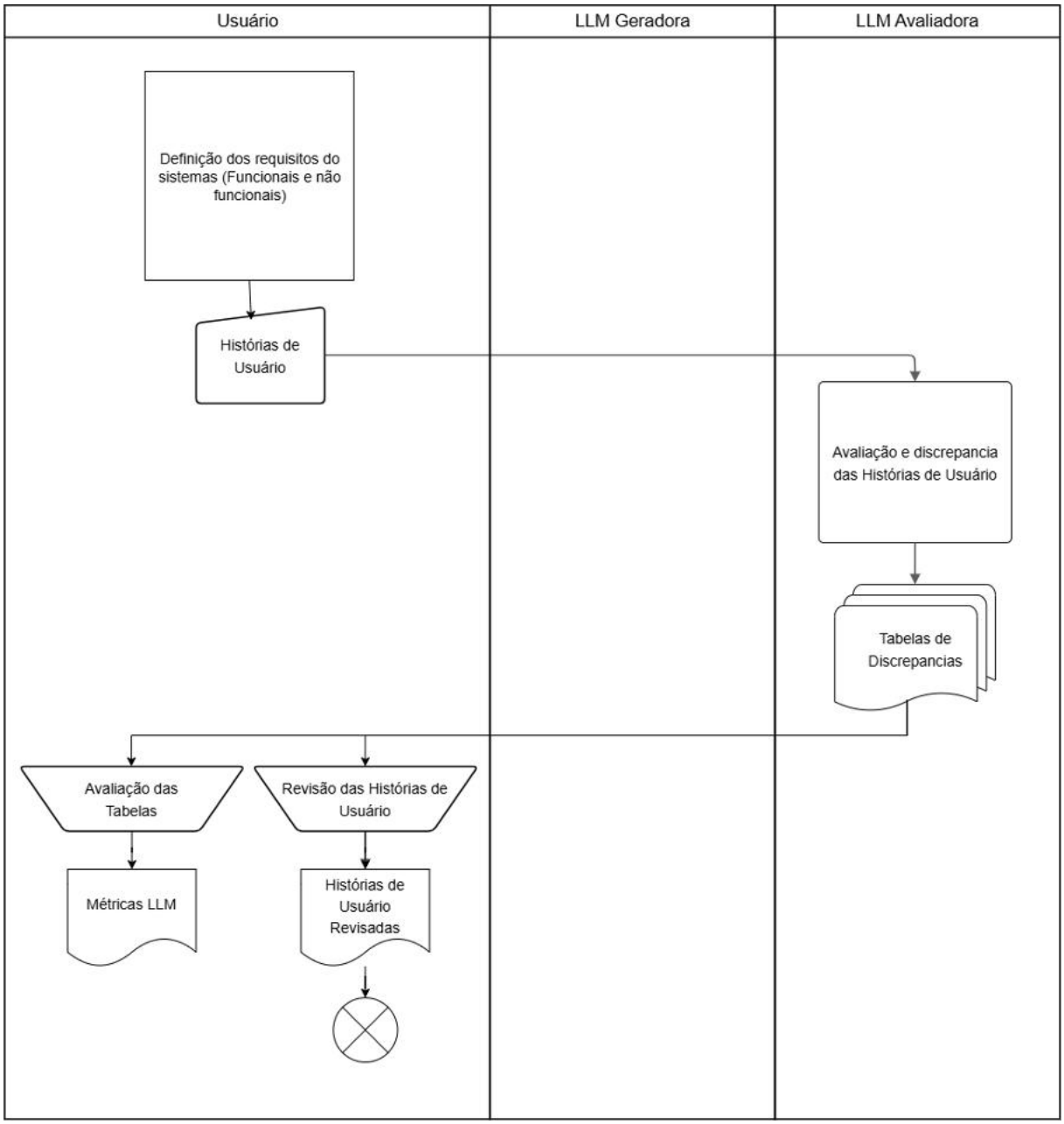
cificação. Esse ciclo se repetiu até que a pontuação atingisse o limite estabelecido como critério de aprovação. Essa abordagem é conhecida como o padrão *Evaluator-Optimizer*.

Por fim, a partir disso, seria possível planejar e executar etapas futuras que não serão abrangidas. Este capítulo detalha o processo correspondente a cada uma dessas etapas.

3.1 AVALIAÇÃO DE DISCREPÂNCIA

O primeiro passo do experimento em que o uso da LLM foi empregado de maneira direta foi a avaliação de discrepâncias. A proposta desta etapa consistia em fornecer ao modelo um conjunto contendo a totalidade das USs levantadas e solicitar uma avaliação individual de cada uma delas, classificando as irregularidades encontradas em 3 grupos separados: ambiguidade, omissão e fato incorreto.

Figura 2 – Fluxo da Primeira Etapa do desenvolvimento da aplicação com uso de LLMs



Fonte: Elaborado pelos autores.

O levantamento dos requisitos do sistema foi realizado através de pesquisas em sites

de marketplace amplamente conhecidos, como Shopee e Mercado Livre, e elaborado de acordo com a necessidade de criar um MVP que pudesse atender às expectativas do experimento, isto é, ser um marketplace genérico e o mais simplificado possível. Com base neste levantamento, foram escritas todas as 42 USs, cada uma com seus respectivos Critérios de Aceite – condições específicas que uma funcionalidade deve atender para ser considerada completa e aceitável pelo usuário ou stakeholders (ACs). Este material pode ser encontrado no Apêndice A

Abaixo você encontra um exemplo de US escrita:

US-4: Sessões simultâneas e OAuth2

- Descrição: Como usuário cadastrado, quero acessar a plataforma em vários dispositivos simultaneamente.
- Critérios de Aceite:
  - US-4-AC-1: O usuário pode logar e manter sessões ativas em múltiplos dispositivos sem interrupções.
  - US-4-AC-2: A autenticação precisa ser no padrão OAuth2.
  - US-4-AC-3: Sempre que um refresh token for utilizado mais de uma vez, todas as sessões do usuário devem ser encerradas.

Cada discrepância apontada pela LLM foi classificada um entre três grupos de acordo com os seguintes critérios:

- *Ambiguidade*, caso a história de usuário fosse escrita de maneira vaga, pouco objetiva, onde caberiam múltiplas interpretações contraditórias;
- *Inconsistência*, caso a história de usuário apresentasse informações que vão de encontro a outras informações dentro do próprio artefato;
- *Omissão*, caso a história de usuário deixasse perceptível a ausência de alguma informação importante para a total compreensão da mesma;
- *Informação estranha*, caso a história de usuário fornecesse informações desnecessárias ou não utilizadas;
- *Fato incorreto*, caso a história de usuário estivesse devidamente detalhada e plenamente compreensível, porém contendo informações incorretas acerca da realidade que está sendo ali representada.

Esta classificação foi baseada na proposta em (MELLO; TRAVASSOS; SILVA, 2011).

A primeira limitação importante da ferramenta se apresentou como um obstáculo já na etapa inicial do projeto. O documento escrito inicial, contendo todas as USs e ACs, possuía um número de caracteres bem acima do limite máximo aceito por entrada pelo ChatGPT no dia em que foi iniciada a avaliação de discrepâncias. A solução mais lógica seria, simplesmente, dividir o texto de entrada em várias partes e fornecê-las à ferramenta de forma gradual. Esta abordagem não se mostrou eficiente, pois o modelo não conseguia criar um senso de continuidade entre as diferentes entradas e, frequentemente, perdia ou repetia informações de uma forma que inviabilizou, num primeiro momento, uma análise completa.

Após um breve debate, foi tomada a decisão de retroceder à etapa anterior e reescrever as USs de forma que as mesmas pudessem ser agrupadas em módulos menores, separando as histórias nas visões dos diferentes tipos de usuário que interagiam no sistema: todos, administradores, compradores e vendedores, viabilizando que estes módulos fossem fornecidos como entrada da LLM de maneira independente. Esta abordagem se provou, na prática, como sendo a solução mais eficaz e, assim, possibilitando avançar à próxima etapa.

Foram criados dois tipos de Prompt, o tipo 0-shot (Apêndice B.1) e o tipo 1-shot (Apêndice B.2), ambos com o objetivo de gerar um arquivo CSV contendo as discrepâncias identificadas.

Prompt de Avaliação de Discrepância 0-shot (Apêndice B.1)

```
Como um engenheiro de software especializado em inspeção de software. Valide todas as seguintes historias
de usuario e seus criterios de aceite (ambos, nao somente um ou outro) e descreva o relatorio de
discrepancias num formato csv onde cada linha deve seguir o formato abaixo. Para entender as falhas e
discrepancia nos requisitos desenvolvidos, voce pode identificar uma discrepancia sobre historias de
usuario em si e/ou discrepancias sobre um ou mais criterios de aceite de historia de usuario e portanto
deve avaliar ambos separadamente. Nao faca suposicoes, caso nao consiga extrair dos requisitos as
informacoes necessarias para a construcao do modelo de projeto, relate essa omissao de informacoes como
discrepancias, ao inves de tentar deduzi-las. Nao tente corrigir eventuais discrepancias no documento de
requisitos, apenas descreva-as de forma a possibilitar a correcao. Lembre-se de pensar etapa por etapa.

O formato de saida deve possuir um "Id" que e o id unico numero incremental da discrepancia, um campo
"item de avaliacao" relacionado ao codigo da historia de usuario ou o codigo do criterio de aceite, um
campo "Tipo de defeito" que e um dos defeitos da lista de defeitos abaixo, um campo "Descricao" que e uma
breve descricao da discrepancia, um campo "Localizacao" que e a raia responsavel pela acao da historia de
usuario ou criterio de aceite. um campo "Locais onde se repete" outras historias de usuario ou criterios
de aceite onde se repete

"""Lista de Defeitos
* Omissao (Informacoes necessarias foram omitidas do artefato)
* Fato incorreto (Algumas informacoes no artefato de software contradizem informacoes presentes no oraculo
ou o
proprio conhecimento geral do dominio)
* Inconsistencia (As informacoes em uma parte do artefato de software estao inconsistentes com outras no
artefato
de software).
* Ambiguidade (As informacoes no artefato de software sao ambiguas, isto e, e possivel ao desenvolvedor
interpretar as informacoes de diferentes maneiras, podendo levar a uma implementacao incorreta).
* Informacao estranha (As informacoes sao fornecidas, mas nao sao necessarias ou mesmo usadas.)
```

```
"""

"""Historias de usuario para avaliacao

"""
```

Para cada combinação entre módulo e tipo de Prompt, foram realizadas três execuções, totalizando seis avaliações por módulo e vinte e quatro no total. Em seguida, cada discrepância detectada pela LLM foi analisada de forma individual quanto à sua corretude (ou seja, se representava de fato uma divergência válida) por cada um de nós de forma separada, tanto no ponto de vista da discrepância em si quanto de sua classificação e verificada quanto à recorrência em execuções anteriores. Foram contabilizadas as discrepâncias válidas não repetidas, bem como aquelas que, apesar de legítimas, apresentaram erro na classificação do tipo de discrepância.

Acima está um exemplo de Prompt 0-shot utilizado neste trabalho.

Por fim, todas as discrepâncias válidas e não repetidas foram utilizadas para a correção e melhoria das USs, resultando na criação de um novo documento contendo as USs revisadas que pode ser encontrado no Apêndice C.

3.2 MODELO DE BANCO DE DADOS

Uma vez revisadas todas as USs e aplicadas as correções pertinentes, já era possível formular quais entidades fazem parte do sistema, dessa forma um modelo de banco dados poderia ser produzido para atender todas as USs. Deste modo foram elaborados um Prompt e um Oráculo.

O Oráculo, desenvolvido pelos autores, consiste em um modelo relacional convertido em um script DDL, o qual cobre todas as USs com o objetivo de facilitar a validação dos DDLs gerados pela LLM. Esse oráculo foi projetado com o mínimo necessário para atender adequadamente a todos os ACs, sem a inclusão de índices ou otimizações específicas; portanto, o Oráculo é o mínimo para ser considerado correto.

Esse oráculo pode ser encontrado no Apêndice D e abaixo está um exemplo de tabela pertencente a ele:

DDL Criação de Categoria de Produto (Apêndice D.1)

```
CREATE TABLE product_category (
    id           BIGINT      NOT NULL PRIMARY KEY,
    name         VARCHAR     NOT NULL,
    created_date TIMESTAMP   NOT NULL,
    updated_date TIMESTAMP
);
```

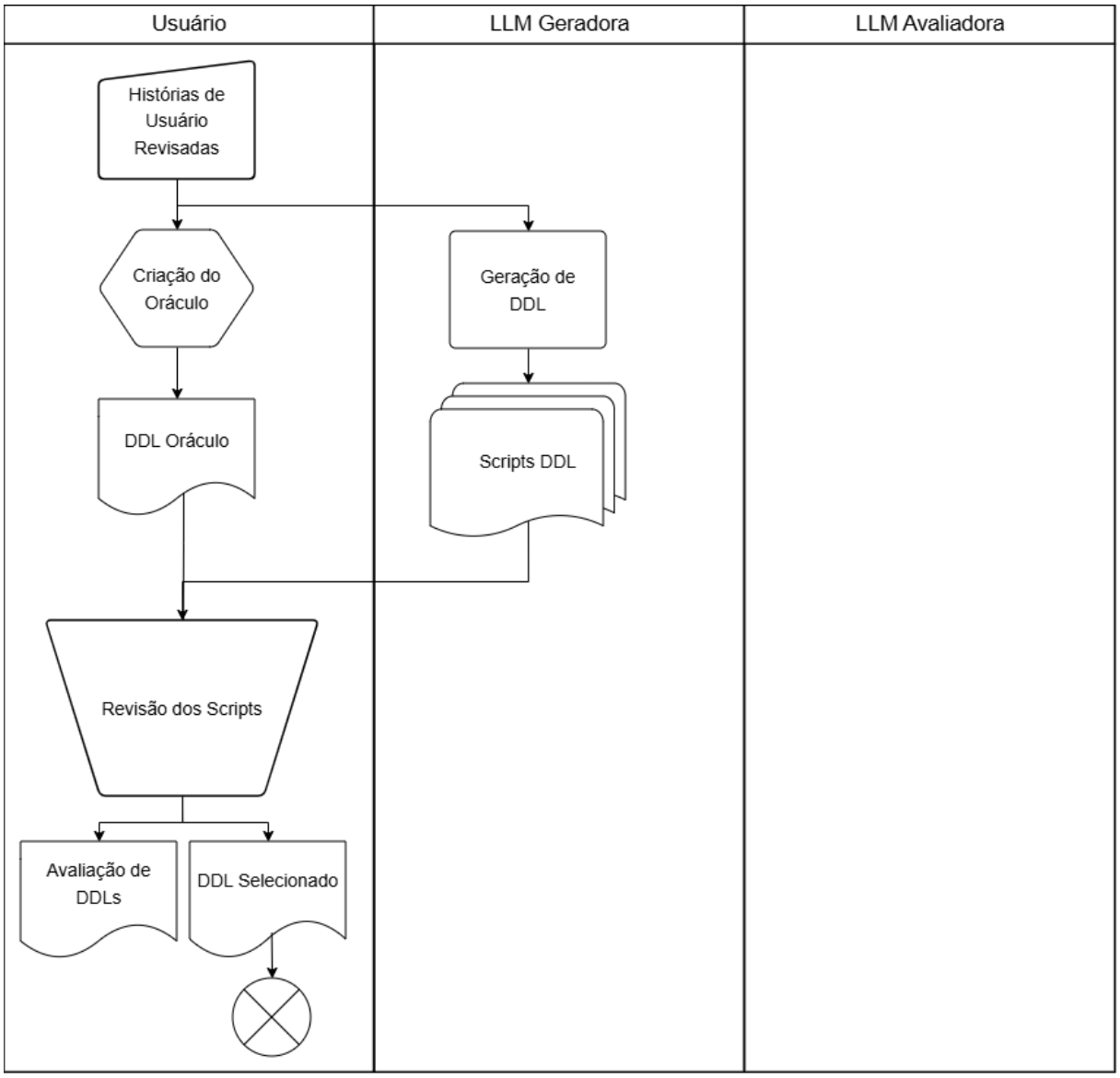


Figura 3 – Fluxo da segunda etapa do desenvolvimento com uso de LLMs

Fonte: Elaborado pelos autores.

Durante a criação de um protótipo de script para identificar eventuais problemas no experimento, observou-se que o ChatGPT impunha um limite de entrada entre 23.000 e 25.000 caracteres, enquanto o Microsoft Copilot apresentava uma limitação mais restrita, de aproximadamente 4.000 caracteres. Como o documento de requisitos possuía 34.061 caracteres e cerca de 5.300 palavras, foi necessário estabelecer uma estratégia para viabilizar o fornecimento desse material como entrada. Optou-se pelo ChatGPT, já utilizado anteriormente e com maior capacidade de entrada. Após breves testes e discussões, definiu-se uma abordagem incremental: o primeiro Prompt incluiu um modulo de requisitos, e sua resposta foi utilizada como parte do contexto para um segundo Prompt contendo um novo mudulo, até que todas fossem adicionados. Embora a primeira versão completa do banco de dados gerada apresentasse erros, ela demonstrou a viabilidade da estratégia.

O Prompt final foi, então, concebido para gerar um script DDL com base nas USs revisadas, identificando os campos necessários e os relacionamentos entre as entidades do sistema. Adotou-se o banco de dados PostgreSQL, amplamente utilizado no mercado, e uma abordagem *zero-shot*, dado o conhecimento difundido da sintaxe DDL do banco em questão. Instruções detalhadas foram evitadas, tanto por serem desnecessárias quanto para não ultrapassar o limite de Context Window.

Prompt de Geraçao de DDL (Apêndice B.3)

```
/* Prompt Inicial */

Como um gerente de banco de dados especializado em bancos postgres utilizados por microsserviço,.leia as
seguintes histórias de usuário e seus crité rios de aceite e gere scripts DDL para a criação dessas
tabelas, constraints, relacionamentos e indexes. Lembre-se de pensar etapa por etapa para atender todas as
histórias e seus crité rios de aceite.
"""Histórias de usuário

"""

/* Prompt Incremental */

Como um gerente de banco de dados especializado em bancos postgres utilizados por microsserviços, leia as
seguintes histórias de usuário e seus crité rios de aceite e baseados no banco de dados abaixo gere
scripts DDL para a criação dessas tabelas, constraints, relacionamentos e indexes. Lembre-se de pensar
etapa por etapa para atender todas as histórias e seus crité rios de aceite. Se necessário alterar alguma
tabela existente, gere o script de criação com a tabela alterada ao invé s de gerar um script de
atualização. Sua resposta deve conter todas as tabelas, incluindo as do input, e não somente as alteradas
ou criadas como resposta a esse prompt..

"""Banco Atual

"""

"""Histórias de usuário

"""
```

Inicialmente, o Prompt foi alimentado apenas com as USs do Módulo 1 (Apêndice A.2), sem qualquer DDL prévio. Em seguida, ao processar o Módulo 2, as histórias do Módulo 1 foram substituídas, e o DDL gerado anteriormente foi reaproveitado para compor um novo script, agora contendo as entidades e relacionamentos de ambos os módulos. Esse processo foi repetido até a inclusão dos quatro módulos, resultando em um Prompt final mais abrangente.

Contudo, verificou-se que essa abordagem sequencial gerava inconsistências devido à ausência de uma visão integrada das interações entre os módulos. Considerando que o limite de Context Window permitia a inclusão simultânea de dois módulos, optou-se por combinar os Módulos 1 e 2 na primeira execução, e os Módulos 3 e 4 na segunda.

Para contonar essa limitação, foi solicitado explicitamente na versão final do prompt que a resposta final integrasse as respostas das etapas anteriores corrigidas, incorporando eventuais novas tabelas e modificações diretamente no script consolidado, em vez de gerar um script complementar com alterações às estruturas existentes. No entanto, a ferramenta seguiu a segunda abordagem, que se pretendia evitar. Acredita-se que essa escolha decorra da tendência das LLMs de minimizar o número de Tokens de saída gerados. Nessas situações, descartou-se a resposta gerada até que fosse obtida uma versão que atendesse plenamente ao objetivo do Prompt.

No total, foram realizadas três execuções independentes do fluxo de trabalho, resultando em três versões distintas do script DDL. Cada uma foi avaliada manualmente, adotando critérios lenientes, com o objetivo de verificar se todas as USs e seus ACs haviam sido contemplados, ainda que algumas responsabilidades fossem delegadas à camada de aplicação. Também seria avaliado se a LLM tinha a capacidade de pensar em possíveis otimizações, como a criação de views, procedures ou indexes.

Para continuidade do experimento, considerando os resultados obtidos como definido no Capitulo 4 optou-se por utilizar o Oráculo como referência principal, em vez de adotar qualquer um dos DDLs gerados pelas ferramentas.

3.3 DEFINIÇÃO DE APIS

Com o DDL definido, o utilizamos juntamente com as USs e os requisitos não funcionais para elaborarmos uma especificação de API REST em formato OAS. Esse formato padronizado estabelece um contrato transparente entre as equipes (front-end, back-end) e os stakeholders, promovendo alinhamento desde o início e permitindo o desenvolvimento paralelo sem bloqueios.

Além disso, essa abordagem favorece a *reusabilidade e escalabilidade*. Com contratos de API bem estruturados desde o início, componentes podem ser reutilizados em diferentes contextos e módulos. Ainda, permite-se incorporar políticas de *segurança e governança* já no design, aplicando padrões de autenticação, autorização e versionamento diretamente na especificação da API.

Adicionalmente, organizamos os módulos por funcionalidades completas, agrupando todas as USs relacionadas (por exemplo, criação, atualização e gerenciamento de sessão) independentemente do tipo de sessão (administrador, comprador ou vendedor). Nesse agrupamento, também incluímos os elementos do sistema gerados pelo DDL, como as tabelas de usuário, sessão e tokens de autenticação, que estão associados àquela funcionalidade.

A Tabela 2 apresenta o mapeamento completo das 42 USs organizadas em 26 funcionalidades distintas, demonstrando como as diferentes operações do sistema foram agrupadas de forma lógica e coesa para facilitar o desenvolvimento da especificação OAS.

Tabela 2 – Agrupamento de USs em Módulos de Funcionalidades Completas

COD	**Funcionalidades**	**USs**
F-1	Criação de Usuário Comprador	US-1
F-2	Gerenciamento dos próprios dados	US-5
F-3	Confirmação de Email	US-2
F-4	Recuperação de Senha	US-3
F-5	Criação, atualização e gerenciamento de sessão	US-4, US-6, US-7, US-8
F-6	Encerramento de Conta	US-11
F-7	Gerenciamento de Loja, Criação de Dono da Loja, Gerenciamento de usuário vendedor	US-37
F-8	Gerenciamento de Produto	US-32, US-33, US-34, US-39
F-9	Gerenciamento de Endereços de Entrega	US-22
F-10	Bloqueio de Produto	US-14
F-11	Avaliação de Produto	US-9, US-27, US-40
F-12	Perguntas e Respostas de Produto	US-16, US-23, US-36
F-13	Abertura de pedido de Devolução	US-29
F-14	Gerenciamento de Pedido de Devolução	US-17
F-15	Gerenciamento de Nivel de Acesso e Gerenciamento de Papeis de Autorização	US-12
F-16	Criação de Administrador e Gerenciamento de Administradores	US-13
F-17	Abertura de Ticket de suporte	US-10
F-18	Gerenciamento de Ticket de suporte	US-18
F-19	Visualização de Estatisticas de Loja	US-19
F-20	Visualização de Estatistica de Produto	US-35
F-21	Visualização de pedidos, Criação de Pedidos, Cancelamento de Pedidos, Atualização de andamento de Pedidos	US-24, US-25, US-26, US-28, US-41, US-42
F-22	Favoritar Produtos	US-30
F-23	Visualização de Produtos	US-20
F-24	Abetura de Ticket de Denuncia de Produtos	US-31
F-25	Gerenciamento de Saldo de Loja	US-38
F-26	Bloqueio de Usuários	US-15

Fonte: Elaborado pelos autores.

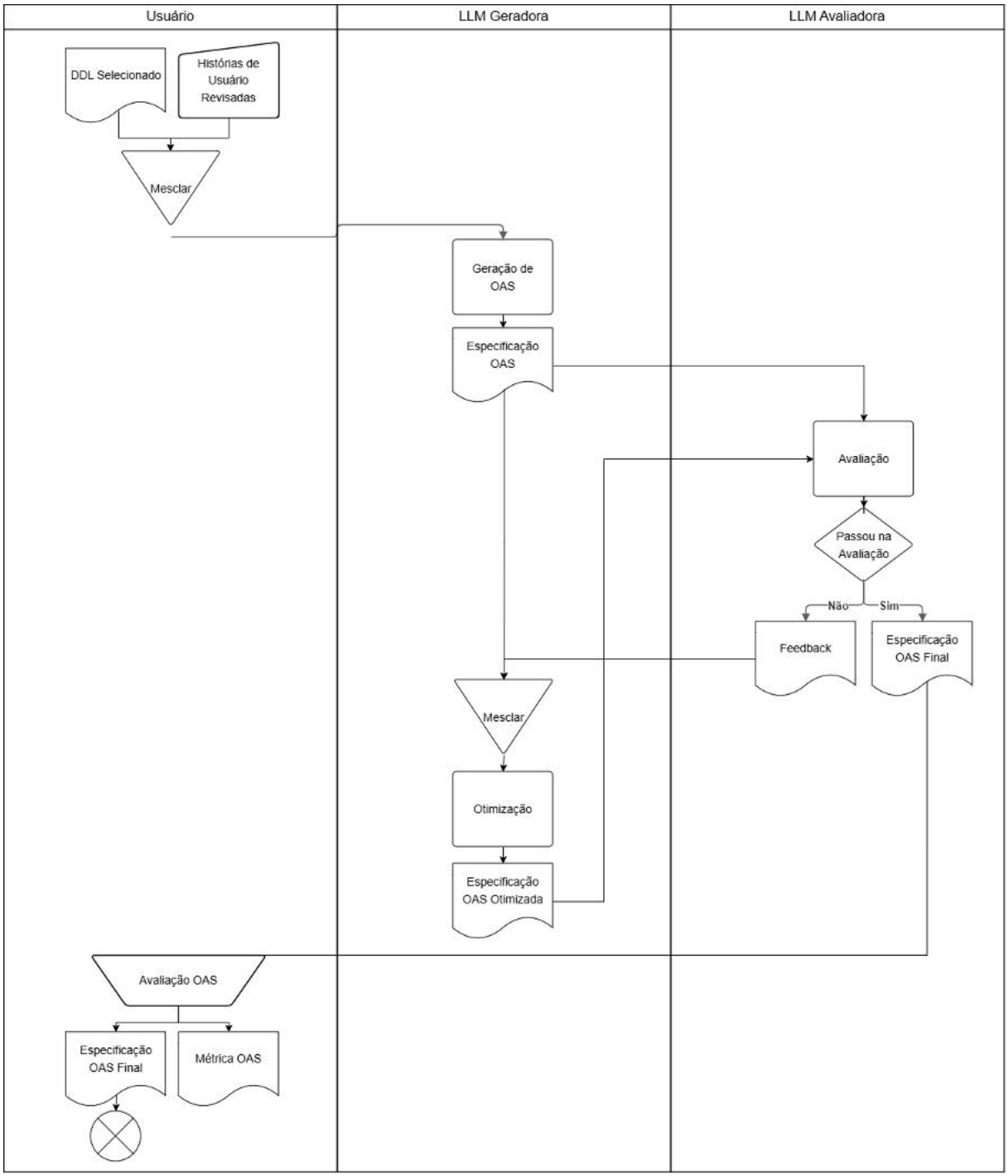


Figura 4 – Fluxo da terceira etapa do desenvolvimento da aplicação com uso de LLMs.

Fonte: Elaborado pelos autores.

Para esta etapa, adotamos o padrão **Evaluator–Optimizer**: uma LLM era responsável por gerar e otimizar a especificação OAS, enquanto outra LLM avaliava essa especificação e fornecia um feedback. Existem referencias para esse pattern de workflow, que pode ser encontrado em (ANTHROPIC, 2024) ou (GURU, 2025).

Foram empregadas duas LLMs distintas no processo. Considerando que o método adotado envolve ambas as etapas de geração e avaliação de OAS, optou-se por utilizar uma para a geração de material e outra para a avaliação deste material gerado, com o objetivo de mitigar vieses e aumentar a confiabilidade dos resultados, uma vez que cada modelo, após o treinamento, desenvolve seus próprios parâmetros e características peculiares.

A LLM geradora utilizada foi o GPT-4o (OPENAI, 2025a), aplicada em dois prompts:

1. **Prompt inicial (0-shot) - B.4**: gera uma especificação OAS (versão 3.0.1) a partir das USs, requisitos não funcionais e entidades do sistema. O objetivo é produzir uma especificação completa, com schemas reutilizáveis e aplicação de segurança (como OAuth2), sem a necessidade de exemplos de sintaxe, por ser um formato amplamente conhecido.

2. **Prompt de otimização (0-shot) - B.6**: corrige e aprimora a especificação gerada, com base nos mesmos insumos iniciais, na versão anterior da especificação e no feedback da LLM avaliadora.

A LLM avaliadora foi o DeepSeek-V3 (DEEPSEEK-AI et al., 2025). Seu prompt (1-shot) foca na geração de feedback estruturado, pontuando a especificação entre 0 e 100 com base em discrepâncias classificadas como ERROR, WARNING ou MINOR. O formato do feedback segue uma estrutura padronizada, permitindo a extração automática por algoritmos. Ele pode ser encontrado no Apêndice B.5.

Para os três prompts foi adotado CoT, guiando as LLMs a identificar corretamente as rotas e entidades necessárias, e a produzir tanto a especificação final (OAS) quanto o feedback estruturado.

O experimento visa repetir o ciclo de geração, avaliação e otimização até que a LLM avaliadora atribua nota superior a um limiar predefinido, sem que essa LLM conheça esse limiar, evitando vieses. A versão da especificação que atingir a nota requerida é então considerada aceitável. O ciclo de avaliação e otimização foi operada manualmente, mas poderia ser incluída em um Workflow de IA.

Prompt de Geração OAS (0-shot) (Apêndice B.4)

```
Você é um modelador de APIs Restful completas, experiente na construção de OpenAPI Specifications, seu
objetivo é ler as entidades do sistema, e criar APIs que cumpram as histórias de usuários respeitando os
requisitos não funcionais. Suas APIs devem ser coesas, respeitando todos os padrões REST como por exemplo
HATEOAS, e devem possibilitar tracking com a adição de headers opcionais. A especificação gerada deve ser
feita utilizando componentes e schemas e incluir título, descrição. As rotas alé m das requisições, devem
possuir tags, sumário, descrição, tipagem, exemplo e descrição de todos os campos. As respostas devem
possui exemplos para todos os status possíveis das famílias 2xx, 4xx e 5xx. Não suponha nenhuma
funcionalidade, apenas implemente as histórias de usuário fornecidas.

Para isso você vai seguir as seguintes etapas:
1 - Entender as funcionalidades e identificar quais rotas devem ser geradas
2 - Identificar quais campos devem ser utilizados nas requisições, separando os da requisição e aqueles
que serão gerados no processamento da API, podendo criar novos dados se necessário.
3 - Escrever a especificação no padrão OAS 3.0.1 no formato yml

"""Requisitos não funcionais

"""

"""Histórias de usuário

"""

"""Entidades do Sistema

"""
```

Prompt de Otimização de OAS (0-shot) (Apêndice B.6)

```
Voce e um modelador de APIs Restful completas, experiente na construcao e correcao de OpenAPI
Specifications. Seu objetivo e ler todo o contexto fornecido, revisar um feedback e entender seus pontos
de critica para corrigir uma especificação OAS fornecida. Você deve gerar uma nova especificacao completa
que contenha todo o feedback informado e corrigido, não apenas salientar quais locais devem ser alterados.
Nao suponha nenhuma correcao alem das informadas no feedback.

Para isso voce vai seguir as seguintes etapas:
1 - Entender as solicitacoes do feedback e vincular com o contexto e OAS informada.
2 - Baseado nessa compreensao, identificar quais correcoes devem ser feitas.
3 - Reescrever a especificacao OAS 3.0.1 completa, incluindo as correcoes

"""Contexto

"""

"""Especificacao OAS 3.0.1

"""

"""Feedback

"""
```

Por fim, esta especificação OAS validada é revisada, possibilitando a extração de métricas e servindo como base para etapas futuras do desenvolvimento.

3.4 PRÓXIMAS ETAPAS

Escrever código é uma tarefa intrinsecamente complexa, o que torna ineficiente o uso direto e não estruturado de LLMs. A simples colagem segmentada de artefatos no contexto (Context Window) ou a aplicação de RAG resulta em um processo árduo e pouco produtivo, especialmente devido à dificuldade de organizar e inserir esses artefatos de forma controlada. Nesse cenário, a prática recomendada é utilizar as LLMs a partir de um nodo Agente Autônomo, integrado a um editor de código. Mesmo com editores modernos que incorporam LLMs agênticos, é fundamental tratar os artefatos como fontes de verdade e garantir sua inserção controlada no contexto. Tais editores geralmente contam com integração a modelos com Reasoning, capazes de interpretar, decidir e agir diretamente sobre uma base de código. Além disso, oferecem suporte a MCPs e a APIs internas, o que possibilita operações como leitura e edição de arquivos, execução de comandos, acesso a repositórios, consulta ao histórico de *pull requests*, abertura de *tickets* e tarefas, entre outras funcionalidades.

Nesse contexto, a geração e aplicação de *edits* ocorre, em termos práticos, por meio de um ciclo no qual o modelo propõe *diffs* e o editor aplica as alterações no sistema de arquivos do projeto, mediante aprovação do usuário. O Agente Autônomo utiliza o contexto do código para propor modificações consistentes com os artefatos de referência (como OAS e DDL), apresentando visualizações de diferenças, escopos de arquivos afetados e, geralmente, mantendo um histórico das alterações realizadas.

Quando estão disponíveis requisitos não funcionais, USs com ACs, DDL e OAS, é possível estabelecer um **contrato de referência**, que atua como Oráculo de consistência técnica. Os requisitos não funcionais delimitam atributos de qualidade, como desempenho, segurança, observabilidade e integração com ferramentas externas, orientando decisões arquiteturais e trade-offs. As USs, acompanhadas de seus ACs, definem comportamentos observáveis e regras de negócio, além de servirem de base para testes de aceitação. O DDL fornece o esquema canônico de dados, utilizado para mapear entidades de domínio e camadas de persistência. Já a OAS define o contrato externo da API, a partir do qual derivam-se modelos, validações e roteamento.

A seguir, propomos um procedimento para utilização das LLMs nesse contexto. Ressaltamos, entretanto, que não discutiremos a iteração sobre esse procedimento nem a coleta de métricas a seu respeito.

O procedimento organiza-se em quatro microetapas iterativas: planejamento, geração, verificação e iteração. No *planejamento*, consolida-se um guia de tarefa que alinha requisitos não funcionais às USs, derivando restrições arquiteturais explícitas e comporta-

mentos observáveis. Nessa etapa, solicita-se ao modelo a criação de esqueletos de código orientados por contrato, nos quais o DDL é traduzido em modelos, entidades e migrações, enquanto a OAS é convertida em controladores, DTOs, clientes HTTP e validações. Na etapa de *geração*, conduzida pelo Agente Autônomo, aplicam-se técnicas de Reasoning como CoT, a fim de tornar explícito o raciocínio do modelo e reduzir erros de integração. Em paralelo, o agente executa *linters*, testes unitários e de aceitação derivados dos ACs, além de verificações estáticas de conformidade ao contrato definido pelo DDL e pela OAS, utilizando também ferramentas de *build*. O resultado, ao final dessa etapa, deve ser estaticamente consistente. Na *validação*, o usuário analisa as alterações propostas, verificando sua adequação em termos de desempenho e conformidade aos requisitos, aprovando ou rejeitando mudanças. Por fim, a etapa de *iteração* permite que o desenvolvedor refine e ajuste o código gerado, formulando novas exigências com base na versão atualizada do projeto e no contexto consolidado das etapas anteriores.

A operacionalização desse procedimento apoia-se na curadoria de contexto e na execução de ciclos curtos. Inicialmente, insere-se no contexto um resumo estruturado dos requisitos não funcionais, seguido das USs e ACs, além de trechos relevantes do DDL e da OAS. Dada a limitação da Context Window, recomenda-se segmentar os artefatos por domínio funcional, preservando referências cruzadas. Em seguida, solicita-se a geração dos módulos mínimos para um MVP orientado por contrato: modelos de domínio alinhados ao DDL, adaptadores de persistência, clientes HTTP e controladores baseados na OAS. O Agente Autônomo deve ser instruído a não extrapolar o contrato, a justificar decisões que impactem requisitos não funcionais e a produzir *stubs* de testes de aceitação parametrizados pelos ACs. Em edições subsequentes, amplia-se a cobertura de testes e a observabilidade, incorporando instrumentação e métricas de latência consistentes com os requisitos. A cada ciclo, verifica-se a compatibilidade entre estrutura de dados e endpoints, por meio de geração ou validação automática de clientes/servidores a partir da OAS, além da aplicação de migrações consistentes com o DDL.

A derivação de código a partir dos artefatos citados deve seguir princípios de rastreabilidade. Do DDL, extraem-se tipos, chaves e relacionamentos que subsidiam entidades, repositórios e migrações, incluindo restrições e índices orientados a requisitos de desempenho. Da OAS, derivam-se contratos de requisição e resposta, códigos de erro e esquemas de autenticação, que parametrizam validações e serialização. Das USs e respectivos ACs, obtêm-se casos de teste e cenários de comportamento, com dados de exemplo úteis tanto para testes quanto para *seeding* de desenvolvimento. Por fim, dos requisitos não funcionais, derivam-se *budgets* de latência, níveis de log e políticas de *retry*/*circuit breaking*. Esse processo de derivação deve ser incorporado ao Prompt do Agente Autônomo, deixando explícito que qualquer código gerado é inválido caso não satisfaça os contratos definidos pelos artefatos e que ambiguidades devem ser reportadas com hipóteses mínimas e testáveis.

4 ANÁLISE E DISCUSSÃO DOS RESULTADOS

4.1 AVALIAÇÃO DE DISCREPÂNCIA

Tabela 3 – Apuração de resultados da análise de discrepâncias

Seção	**# discrepâncias relatadas**	**# defeitos confirmados**	**precisão**
M1-0S-T1	13	8	61,51%
M1-0S-T2	11	10	90,91%
M1-0S-T3	10	7	70,00%
M1-1S-T1	23	10	43,48%
M1-1S-T2	21	11	52,32%
M1-1S-T3	12	7	58,33%
M2-0S-T1	15	6	40,00%
M2-0S-T2	12	7	58,33%
M2-0S-T3	18	10	55,56%
M2-1S-T1	20	11	55,00%
M2-1S-T2	10	6	60,00%
M2-1S-T3	18	11	61,11%
M3-0S-T1	10	6	60,00%
M3-0S-T2	12	4	33,33%
M3-0S-T3	11	5	45,45%
M3-1S-T1	12	5	41,67%
M3-1S-T2	16	7	43,75%
M3-1S-T3	11	4	36,36%
M4-0S-T1	13	5	38,46%
M4-0S-T2	11	4	50,00%
M4-0S-T3	11	4	53,85%
M4-1S-T1	14	8	57,14%
M4-1S-T2	24	9	37,50%
M4-1S-T3	22	10	45,45%

Fonte: Elaborado pelos autores.

Conforme abordado no capítulo 3.1, a proposta deste passo do experimento consistia em montar uma tabela com todas as discrepâncias que a LLM fosse capaz de encontrar nas USs, que foram particionadas de acordo com a metodologia que foi utilizada para contornar as limitações de entrada. Todas as sugestões de correção geradas pela LLM foram verificadas individualmente para que fosse decidido se era cabível empregá-las na correção das USs originais, e o número de intervenções consideradas positivas em relação ao total foi contabilizado. A tabela 3 sintetiza os resultados, enquanto a tabela 4 explica de maneira mais clara a nomenclatura utilizada para definir cada seção.

Tabela 4 – Legenda da terminologia usada na avaliação de discrepâncias

Termo	**Legenda**
M1	Módulo 1 - Usuário Geral
M2	Módulo 2 - Usuário Administrador
M3	Módulo 3 - Usuário Comprador
M4	Módulo 4 - Usuário Vendedor
0S	Zero Shot - Prompt sem exemplo de resposta
1S	One Shot - Prompt com um único exemplo de resposta
T#	Tentativa; # representa o número da tentativa

Fonte: Elaborado pelos autores.

Seguindo a explicação da tabela, a linha M1-0S-T1, por exemplo, corresponde à seção do Módulo 1 - Usuário Geral, utilizando a técnica de prompt zero-shot, primeira tentativa.

Em todas as avaliações, a Precisão foi calculada usando a equação 4.1, onde o Total é a quantidade de itens que foram geradas pela LLM enquanto os itens avaliados corretamente eram aqueles que atendiam os critérios como USs e ACs ou que estavam corretos via validação humana, no caso da Avaliação de Discrepâncias ela foi calcula usando a equação 4.2.

$$\text{Precisão} = \frac{\text{Itens avaliados corretamente}}{\text{Total de itens}} \tag{4.1}$$

$$\text{Precisão} = \frac{\text{Discrepâncias corretas}}{\text{Total de Discrepâncias geradas}} \tag{4.2}$$

O modelo apresentou dificuldades em entender detalhes sobre a formatação específica pedida, no caso, de um arquivo CSV, colocando vírgulas de maneira inadequada em algumas das linhas da resposta. Segue, abaixo, uma demonstração do problema:

> 10,US-9-AC-2,Omissão,Nenhuma especificação sobre ordenação ou critérios de exibição das avaliações (ex.: por data, por relevância),Usuário Geral,nenhum

É possível observar a vírgula empregada de maneira errônea, semelhante à linguagem natural, ignorando o fato de que a mesma cumpre a função de separador de colunas em arquivos CSV.

Após revisar planilhas de discrepância, foi possível notar que a LLM, ao longo das diferentes execuções, relatou problemas iguais ou, ao menos, semelhantes. Por outro lado, a forma como ela descrevia tais problemas não seguia uma linha coerente de raciocínio, ocasionando em uma frequente divergência de compreensão, fazendo com que, em diversos casos, uma nova ocorrência do mesmo problema fosse descrita de maneira que parecesse um outro problema diferente. Além disso, apesar de ser capaz de identificar os problemas

consistentemente, nem sempre o modelo foi capaz de descrevê-lo de forma clara. Confundir omissões com ambiguidades foi mais um comportamento negativo recorrente.

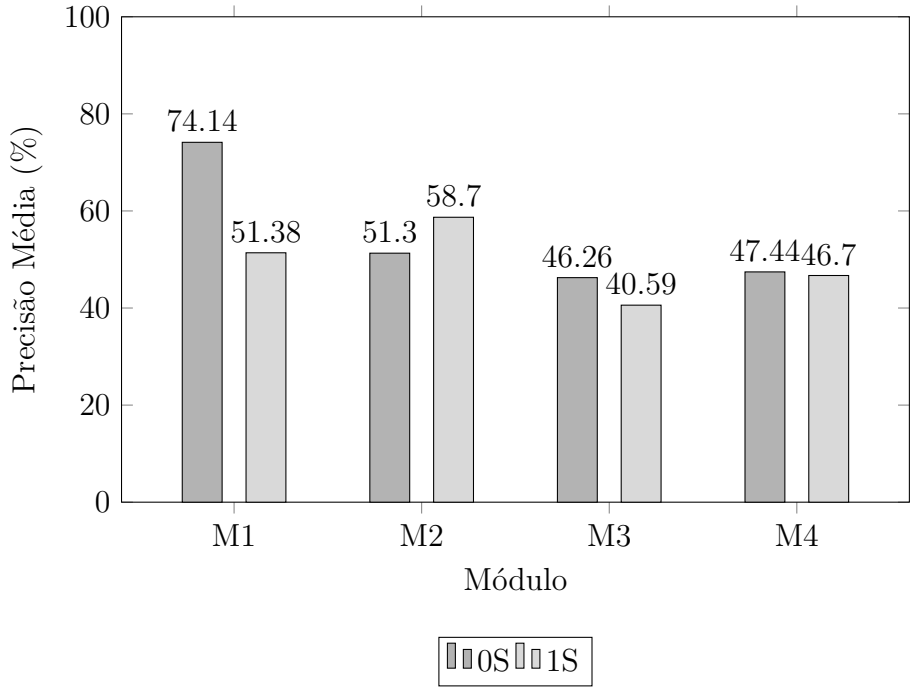
Por fim, a tabela 5 reúne as principais métricas utilizadas para avaliar a eficiência do modelo ao longo de toda a etapa de avaliação de discrepâncias. Com a quantidade de discrepâncias encontradas e aprovadas, a quantidade referente ao experimento zero-shot e one-shot, a quantidade dos itens aprovados que não são duplicados, e a quantidade dos itens não duplicados que possuiam tipo de defeito correto.

Tabela 5 – Análise de eficácia da avaliação de discrepâncias

Métrica	**Total**	**Aprovados**	**Porcentagem**
Discrepâncias encontradas	353	180	50,99%
Obtidos com zero-shot	150	81	54%
Obtidos com one-shot	203	99	48,77%
Aprovados não-duplicados	180	92	51,11%
Aprovados com tipo correto	92	75	81,52%

Fonte: Elaborado pelos autores.

Figura 5 – Média de Precisão por módulo (M) e estratégia (0S e 1S)



Fonte: Elaborado pelos autores.

Ao todo, tivemos 353 discrepâncias encontradas pela LLM, das quais 180 foram aprovadas. Este número representa quase metade (50,99%) do material gerado. O fato de este número significar que a maior parte dos frutos do modelo foi aproveitada sugere que a ferramenta possui boas capacidades e indica uma expectativa positiva sobre a evolução

dos modelos, mas evidencia que, até o dia da realização do experimento, a realidade exige uma curadoria rígida e criteriosa para complementar o trabalho do modelo, afinal, quase metade dos dados de saída não foi aproveitada ao fim do processo.

Outro dado que chama bastante atenção são as diferenças entre a discrepâncias obtidas e taxas de aprovação utilizando os métodos zero-shot e one-shot, que foram, respectivamente, de (150, 54%) e (203, 48,77%). A 5 mostra, em cada módulo, a porcentagem de precisão para zero-shot e one-shot. no M1 e M3, a abordagem com zero-shot foi melhor, enquanto no M2 a one-shot se saiu melhor, e ambas possuindo uma precisão muito parecida no M4. Apesar do zero-shot ter apresentado um melhor aproveitamento geral, a diferença é modesta, sugerindo que intercalar as técnicas utilizadas não aparentou gerar grandes impactos na qualidade das respostas. Por outro lado, o one-shot apresentou mais exemplos de saída, produzindo uma quantidade maior de Tokens de saída, o que pode ter sido a causa da menor Precisão geral. A complexidade dos módulos também é diferente, enquanto o M1 é o mais simples pois possui funcionidades para todos os usuários, seguido do M2 que foca em autenticação e usuários administrativos com boa parte das permissões do sistema, os módulos M3 e M4 são os mais complexos, pois envolvem visões de usuários com permissões especificas e a relação de compra e faturamento que é a parte mais crítica do sistema. Portanto, essa complexidade também afetou a precisão; dessa forma, neste experimento, conforme a complexidade das regras de negócio aumenta, a precisão tende a diminuir.

Tabela 6 – Análise de aproveitamento das discrepâncias encontradas

Etapa de Aprovação	**Quantidade**	**Porcentagem**
Discrepâncias Geradas	353	100%
Discrepâncias Aprovadas	180	51,00%
Aprovadas Únicas (não-duplicadas)	92	51,11%
Categoria Correta	75	81,52%

Fonte: Elaborado pelos autores.

A figura 6 apresenta uma visualização do aproveitamento das discrepâncias. Dos 353 itens gerados, 180 foram aprovados. Dos 180 aprovados, que foram filtrados por semelhança, descartando discrepâncias iguais subsequentes, restando 92 discrepâncias aprovadas e únicas, indicando um aproveitamento de 26% de todas as discrepancias geradas.

Principais Descobertas: Avaliação de Discrepâncias

a) Aproximadamente 51% das discrepâncias geradas pela LLM foram aprovadas após validação humana, totalizando 180 de 353 discrepâncias encontradas.

b) O método One-Shot gerou cerca de um terço a mais de discrepâncias em comparação ao Zero-Shot, mas apresentou uma precisão aproximadamente 6% inferior.

c) Aproximadamente metade das discrepâncias aprovadas eram duplicadas. A duplicidade ocorria em execuções do prompt diferentes.

d) A precisão diminui conforme aumenta a complexidade dos módulos, sendo M1 (mais simples) o mais preciso e M3/M4 (mais complexos) os menos precisos.

e) O modelo apresentou dificuldades com sintaxe e formatação CSV, mesmo com exemplo de saída (one-shot)

f) O modelo apresentou inconsistência na descrição de problemas similares, frequentemente confundindo omissões com ambiguidades.

4.2 MODELO DE BANCO DE DADOS

Como detalhado no capítulo 3.2, a modelagem do banco de dados foi conduzida de forma incremental, com a divisão do problema em conjuntos menores de entradas. Conforme pedido no prompt B.3, em cada iteração, buscou-se consolidar num único DDL a resposta já corrigida das etapas anteriores, incorporando novas tabelas e ajustes diretamente no script final. Todavia, observou-se que a LLM priorizou a geração de scripts complementares (p.ex., *ALTER TABLE*) em vez de um DDL unificando-o oposto do que foi solicitado, possivelmente refletindo uma preferência pela emissão de menos tokens e, consequentemente, pela minimização do esforço de reescrita do esquema. A preferência por gerar um único DDL sem scripts complementares se deu pelo fato de que tais scripts complementares dependeriam do original para funcionar, o que tende a acumular Hallucination conforme as iterações e, portanto, todos os scripts gerados desta forma foram descartados, enquanto a iteração com a LLM foi repetida até obter uma resposta que atendesse o prompt.

A técnica descrita foi aplicada utilizando o prompt 0-shot até a obtenção uma resposta considerada satisfatória e repetida três vezes, cobrindo o processo inteiro do início ao fim. Estas saídas selecionadas foram nomeadas como 0ST1, 0ST2 e 0ST3.

A Tabela 7 sintetiza o universo de itens avaliados (USs e ACs) quanto à sua rele-

vância para a geração de DDL. É importante especificar que a relevância dos itens está diretamente relacionada ao conjunto de histórias de usuário utilizado como base para do experimento. Como várias destas histórias cobriam questões que foram desprezadas, foram considerados irrelevantes itens que atendiam requisitos que acabaram não sendo levados em consideração. Nota-se que 59,52% dos 210 itens foram considerados relevantes, estabelecendo uma base suficientemente robusta para os experimentos de *Zero-Shot* e para a avaliação da capacidade de generalização do modelo.

Tabela 7 – Distribuição dos itens avaliados quanto à relevância para geração de DDL

	# total	**# relevantes**	**# irrelevantes**
Quantidade	210	125	85
Percentual	100%	59,52%	40,48%

Fonte: Elaborado pelos autores.

Nos testes *Zero-Shot* (0ST1, 0ST2 e 0ST3), avaliamos Precisão, presença de incrementos de performance (índices e views) e viabilidade funcional do DDL. Os resultados, apresentados na Tabela 8, mostram que 0ST1 obteve a maior Precisão (62,70%) com DDL funcional, enquanto 0ST2 manteve o DDL funcional, porém com Precisão inferior (47,62%). Em 0ST3, a Precisão foi idêntica a 0ST2 (47,62%), mas o DDL não foi funcional, indicando degradação qualitativa sob repetições do mesmo pedido.

Tabela 8 – Resumo dos resultados dos experimentos Zero-Shot de Geração de DDL

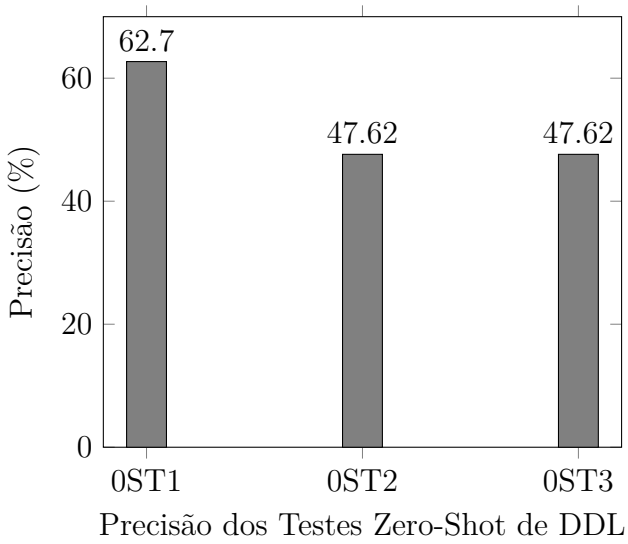
Descrição	**0ST1**	**0ST2**	**0ST3**
Incrementos para performance	16	17	0
Histórias Atendidas	79	60	60
Histórias Não Atendidas	47	66	66
DDL Funcional	Sim	Sim	Não
Precisão (%)	62,70	47,62	47,62

Fonte: Elaborado pelos autores.

A Figura 6 reforça visualmente a queda de desempenho após a primeira tentativa: a Precisão cai de 62,7% (0ST1) para 47,62% (0ST2 e 0ST3). Esse resultado é consistente com a degradação observada qualitativamente em iterações subsequentes, com aumento de erros de sintaxe, omissões de entidades e, em alguns casos, comandos inválidos, sugerindo acúmulo de ruído no contexto e viés de repetição.

No tocante à aderência a requisitos não funcionais, registrou-se um desvio relevante: ao solicitar explicitamente DDL para PostgreSQL, o modelo retornou, inicialmente, um script em MySQL, inválido para o alvo proposto. Somente após instrução adicional para conversão o script passou a ser compatível, o que evidencia capacidade de tradução entre dialetos SQL, porém com tendência a não priorizar rigidamente a especificação do prompt

Figura 6 – Resumo dos resultados dos experimentos Zero-Shot de Geração de DDL



Fonte: Elaborado pelos autores.

na resposta inicial. Conforme a Context Window aumenta, detalhes do Prompt se tornam menos relevantes, como por exemplo a frase que determina o dialeto SQL do resultado.

Outro notável ponto de falha foi a incapacidade de generalizar algumas funcionalidades, por exemplo uma generalização poderia ter sido usada para atender à US-2, US-3 e US-37, renomeando uma tabela e deixando mais claro o seu uso, de modo a abarcar as funcionalidades distintas. Apesar disso, foi aprovado sob leitura leniente, embora comprometesse a clareza semântica do esquema.

Do ponto de vista de requisitos funcionais, a avaliação das histórias destacou um padrão de fragilidade: a criação de tabelas para dados que deveriam residir no *client-side*, contrariando explicitamente a história que proibia sua persistência em banco. Isso ilustra a tendência da LLM em privilegiar estruturas relacionais convencionais mesmo quando o contexto demanda alternativas arquiteturais.

Principais Descobertas: Modelo de Banco de Dados

a) A primeira geração do DDL em um contexto a partir do prompt tende a ser a mais fiel e funcional.

b) Repetições subsequentes do mesmo prompt, no mesmo contexto, podem induzir degradação na qualidade das respostas.

c) Requisitos de dialeto SQL e restrições arquiteturais devem ser enfatizados e reforçados de forma recorrente no prompt.

d) Notou-se uma demanda por ferramentas de validação automática para corrigir as inconsistências mencionadas no item anterior devido ao alto volume.

e) Caso o objetivo final seja a consolidação incremental do DDL, é necessário fornecer instruções e validações adicionais para evitar a produção indesejada de scripts complementares em cascata.

4.3 DEFINIÇÃO DE APIS

A análise dos resultados obtidos através da aplicação da técnica **Evaluator-Optimizer** detalhado no capitulo 3.3, revelou padrões que indicam limitações significativas na consistência e confiabilidade do modelo avaliador. Nas primeiras análises, independente do conteúdo avaliado, o sistema tendia a atribuir sistematicamente a pontuação de 85 pontos, com criticas repetitivas sobre erros, warnings e pontos menores, sempre por volta de uma quantidade parecida distribuída entre essas categorias, enquanto que a segunda avaliação invariavelmente sua para 92 pontos, sugerindo um viés de pontuação pré-definido em vez de uma análise contextualizada. Esse comportamento se agravava ao identificarmos que, ao limpar o contexto da primeira avaliação pontuada em 85, a segunda avaliação invariavelmente era pontuada novamente com 85 pontos, apesar de terem seus erros corrigidos. A pontuação ideal determinada para encerrar o fluxo era 90 pontos.

Um outro detalhe importante a se destacar é o fato das LLMs demostrar tendência a extrapolar além do escopo definido nas historias de usuário do Context Window atual. Apesar de não mencionado explicitamente nas USs do Context Window, mas presentes no DDL requerido para atende-las, como por exemplo uma coluna de produto na API de perguntas de produto, no momento de gerar a especificação a LLM tomou essas tabelas e colunas como fazendo parte dos requisitos e criou rotas de CRUD para elas, mesmo que essas rotas não sejam contempladas diretamente pelas USs da Context Window. Essas rotas extras devem ser descartadas, pois apesar de bem intencionadas, foram feitas sem

contexto ou regra de negócio para sua definição e iterações com outros recursos do sistema, principalmente pelo fato de que essas funcionalidades já estavam descritas em outras USs que seriam inseridas na Context Window em outro shot.

A Tabela 9 apresenta uma análise comparativa entre o número de execuções de otimização necessárias e os valores de pontuação obtidos nas execuções de avaliação para cada funcionalidade. Esta tabela permite identificar o viés de pontuação descrito anteriormente.

Tabela 9 – Quantidade de otimizações e pontuações de avaliação por funcionalidade

COD	**AV 1**	**AV 2**	**AV 3**	**AV 4**	**Otimizações**
F-1	85	92	-	-	1
F-2	85	92	-	-	1
F-3	85	92	-	-	1
F-4	85	92	-	-	1
F-5	85	92	-	-	1
F-6	85	92	-	-	1
F-7	85	92	-	-	1
F-8	75	88	95	-	2
F-9	85	95	-	-	1
F-10	85	95	-	-	1
F-11	85	95	-	-	1
F-12	85	92	-	-	1
F-13	85	92	-	-	1
F-14	85	75	92	-	2
F-15	85	92	-	-	1
F-16	85	92	-	-	1
F-17	85	95	-	-	1
F-18	85	95	-	-	1
F-19	85	92	-	-	1
F-20	85	95	-	-	1
F-21	75	68	85	92	3
F-22	85	95	-	-	1
F-23	85	94	-	-	1
F-24	65	88	95	-	2
F-25	85	92	-	-	1
F-26	85	92	-	-	1

Fonte: Elaborado pelos autores.

Além de limpeza do contexto indicar o viés, a manutenção do histórico de conversa na Context Window mostrou-se problemática, aumentando significativamente as Hallucinations do modelo. Em alguns casos, foram necessárias múltiplas execuções para obter respostas satisfatórias que atendessem o prompt proposto, enquanto em outros a ferramenta alternava entre comportamentos disfuncionais como ativar indevidamente funcionalidades de lousa com executor Python, gerando arquivos YAML inválidos, fragmentar a especi-

ficação em blocos desconexos que mesmo reunidos continham erros estruturais graves, ou fornecer links quebrados ou redirecionar para plataformas externas inexistentes onde supostamente estariam dispostos um repositório com a resposta completa. Esses problemas sistemáticos comprometem a utilidade prática da ferramenta para geração confiável de especificações OAS, exigindo verificação manual extensiva e retrabalho constante por parte do usuário.

A Tabela 10 apresenta uma análise detalhada dos problemas encontrados durante as execuções de avaliação para cada funcionalidade, categorizados por tipo de problema (ERROR, WARNING e MINOR). Esta tabela permite identificar uma proporção de avaliação aonde erros possuíam menos frequência, seguidos de warning e minor com maior frequência.

Para algumas APIs que necessitam de segurança, algumas rotas não possuíam o campo security para determinar que elas deveriam ser protegidas com OAuth, o que constitui uma falha grave, e, portanto, invalida as USs onde ocorre. A LLM não conseguiu se adaptar a um formato de API externa, como por exemplo a API de webhooks do Stripe, e portanto falhou em identificar que essas APIs já possuem um contrato que deve ser seguido e tem seu próprio tipo de autenticação, seja uma API-KEY, por assinatura e encripatação de payload, ou outro tipo de autenticação, forçando sempre o Oauth como padrão das rotas protegidas. O caso mais grave ocorreu na F-16 (Criação de Administrador e Gerenciamento de Administradores) aonde as LLMs não foram capazes de entender que administradores possuem permissões como usuários comuns.

Tabela 10 – Quantidade de problemas relatados por tipo: ERROR (E), WARNING (W) e MINOR (M) em cada avaliação por funcionalidade

	AV 1			**AV 2**			**AV 3**			**AV 4**		
API	**#E**	**#W**	**#M**	**#E**	**#W**	**#M**	**#E**	**#W**	**#M**	**#E**	**#W**	**#M**
F-1	1	2	4	1	3	3	-	-	-	-	-	-
F-2	2	3	4	1	2	3	-	-	-	-	-	-
F-3	3	3	3	1	2	3	-	-	-	-	-	-
F-4	3	3	4	1	2	3	-	-	-	-	-	-
F-5	3	3	4	2	3	4	-	-	-	-	-	-
F-6	2	2	3	1	2	3	-	-	-	-	-	-
F-7	3	3	3	2	2	3	-	-	-	-	-	-
F-8	1	5	4	2	2	3	3	1	3	-	-	-
F-9	2	3	3	1	2	3	-	-	-	-	-	-
F-10	2	2	3	0	2	3	-	-	-	-	-	-
F-11	2	3	3	1	1	3	-	-	-	-	-	-
F-12	2	3	3	2	2	3	-	-	-	-	-	-
F-13	2	2	3	2	2	3	-	-	-	-	-	-
F-14	2	3	3	2	2	4	0	2	2	-	-	-
F-15	3	3	3	2	2	3	-	-	-	-	-	-
F-16	2	3	3	1	2	3	-	-	-	-	-	-
F-17	3	3	4	1	2	3	-	-	-	-	-	-
F-18	2	3	3	1	1	3	-	-	-	-	-	-
F-19	2	2	3	1	1	3	-	-	-	-	-	-
F-20	2	2	4	1	2	3	-	-	-	-	-	-
F-21	1	4	4	2	6	5	3	2	4	4	1	3
F-22	3	3	3	1	1	2	-	-	-	-	-	-
F-23	2	3	4	1	1	3	-	-	-	-	-	-
F-24	2	2	3	0	2	3	0	1	3	-	-	-
F-25	2	3	3	2	2	3	-	-	-	-	-	-
F-26	2	3	3	1	0	3	-	-	-	-	-	-

Fonte: Elaborado pelos autores.

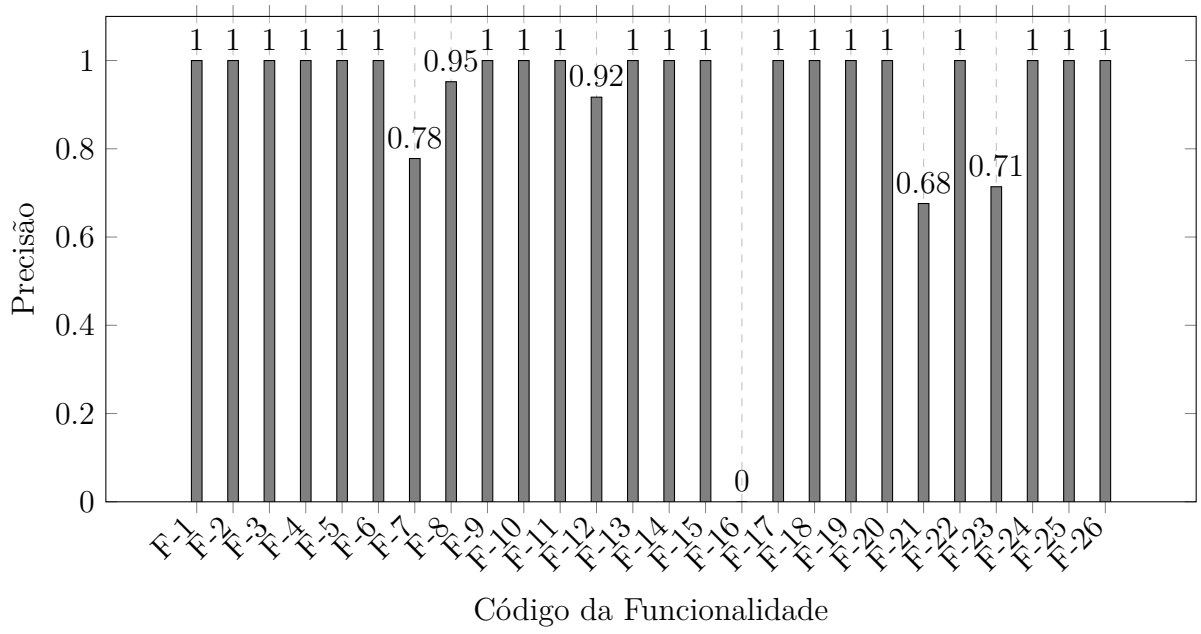
Além disso a validação das especificações geradas revelou inconsistências semânticas significativas. Como os resultados foram criados em módulos independentes, algumas especificações não seguiam o mesmo padrão semântico, enquanto outras que possuíam funcionalidades similares acabavam por criar recursos com redundância. Possivelmente, uma nova etapa seria necessária para unir todos os recursos redundantes e ajustar as especificações geradas, caso o desejo fosse expô-los atrás de um gateway de API ou centralizá-los em um serviço monolítico.

Em vários pontos, foram utilizados caracteres ilegais no formato de especificação, como ":"que quebravam a sintaxe YAML, apesar de serem facilmente identificados e corrigidos com ação humana. Esses erros de sintaxe, embora simples de corrigir, indicam uma

limitação na compreensão do modelo sobre as restrições específicas do formato OAS e a necessidade de validação manual adicional para garantir conformidade com padrões estabelecidos.

Apesar disso, como pode ser observado na Figura 7, que mostra a distribuição da Precisão entre as funcionalidades e as USs e ACs atendidos, pode-se identificar que as funcionalidades mais complexas necessitam de mais esforço de otimização e apresentam mais falhas, porém a maioria das funcionalidades atingiu a Precisão máxima de 100%, indicando que o padrão de **Evaluator-Optimizer** foi bem-sucedido em identificar problemas e corrigi-los mesmo sem a intermediação de um humano e, portanto, a maioria das especificações das funcionalidades atendeu todos os ACs e as USs.

Figura 7 – Precisão das funcionalidades na geração de especificação OAS



Fonte: Elaborado pelos autores.

O cálculo da Precisão foi realizado com base na quantidade de USs e ACs atendidos por uma versão final de especificação (avaliado como acima da pontuação de parada); apesar dos requisitos não funcionais não serem pontuados, o não atendimento deles determina a não aceitação de um critério de aceite ou USs.

O resultado, embora consistente, ainda está aquém do estado da arte e não apresenta refinamento suficiente para ser considerado alinhado às soluções de referência e livre de retrabalho de otimização. Ou seja, as LLMs geraram especificações funcionais que cumpriam os ACs e USs, mas, apesar do desempenho observado ser satisfatório em termos absolutos (chegando a 100% de Precisão em algumas APIs), ele é limitado quando comparado às abordagens de referência. Uma analogia a isso seria o fato de que nem todo código funcional que atende aos requisitos é um "código limpo", de total clareza semântica, fácil manutenção e representa uma versão final na qual nenhum tipo de melhoria seja

necessária. Apesar disso, representou os melhores resultados obtidos neste experimento e que poderiam ser ainda melhores caso existisse um Fine-tuning da LLM.

Principais Descobertas: Definição de APIs

a) O modelo avaliador apresentou viés sistemático de pontuação, atribuindo consistentemente uma pontuação na primeira e segunda avaliação, independentemente do conteúdo avaliado.

b) A limpeza do contexto da primeira avaliação resultava invariavelmente em nova pontuação de 85, mesmo após correção dos erros identificados, confirmando pontuação pré-definida ao invés de análise contextualizada.

c) As LLMs demonstraram tendência a extrapolar além do escopo definido nas histórias de usuário, criando rotas CRUD para tabelas e colunas presentes no DDL mas não contempladas diretamente pelas USs da Context Window, gerando funcionalidades sem contexto ou regra de negócio adequada.

d) Manter o histórico de conversa na Context Window aumentou significativamente as Hallucinations do modelo, gerando comportamentos disfuncionais, como ativação indevida de funcionalidades, arquivos YAML inválidos, caracteres ilegais, fragmentação de especificações e links quebrados.

e) Para APIs que necessitam de segurança, algumas rotas não possuíam o campo security para determinar autenticação OAuth, constituindo falha grave de requisitos não funcionais.

f) O modelo não conseguiu se adaptar a formatos de API externa (como webhooks), falhando em identificar que essas APIs possuem contrato próprio e tipo de autenticação específico, forçando sempre requisitos não funcionais como padrão.

g) A validação das especificações geradas revelou inconsistências semânticas significativas, com especificações não seguindo o mesmo padrão semântico e recursos redundantes, especialmente quando gerados em módulos independentes.

h) A maioria das funcionalidades atingiu a Precisão máxima de 100%, indicando que o padrão **Evaluator-Optimizer** foi bem-sucedido em identificar e corrigir problemas sem intermediação humana.

5 CONCLUSÃO

Este trabalho apresentou um relato de experiência sobre o uso de LLMs no suporte ao desenvolvimento de software, desde os requisitos à definição de contratos de API. A métodologia no capitulo 3 estruturou o fluxo em três macroetapas integradas (Figura 1): (i) avaliação de discrepâncias em USs (Seção 3.1); (ii) modelagem de dados e geração de DDL (Seção 3.2); e (iii) definição de OAS com ciclo de avaliação e otimização (Seção 3.3). Essa organização, apoiada por artefatos de referência e por um Oráculo, permite limitações práticas de LLMs, como dependência da Context Window, ocorrência de Hallucination e sensibilidade às instruções do Prompt.

Na avaliação de discrepâncias, os resultados sintetizados nas Tabelas 3 e 5 e na Figura 6 mostram que 50,99% das discrepâncias geradas foram aprovadas. Fazendo uma curadoria mais rigorosa, 26,06% são aprovadas e únicas (descartando as segundas ocorrências), o que totaliza quase 25% de redundância. Este número elevado é coerente com a natureza de repetitividade do experimento. Apesar da utilidade como triagem, observou-se falhas de aderência a formatos (p.ex., CSV) e variações de clareza, com confusões entre omissão e ambiguidade. Conclui-se que LLMs são valiosas para apontar inconsistências, desde que acompanhadas de curadoria humana criteriosa.

Na modelagem de dados, a Tabela 8 e a Figura 6 indicam desempenho superior na primeira tentativa (Precisão de 62,70%) seguido de degradação em repetições do mesmo pedido. Houve ainda respostas com dialeto SQL incorreto (MySQL em vez de PostgreSQL) e preferência por scripts complementares em detrimento de um DDL consolidado, o que requisita ainda mais atenção e ação humana. Esses achados reforçam a necessidade de enfatizar restrições (dialeto, consolidação) e de empregar validações automáticas para reduzir retrabalho.

Na definição de OAS, o padrão *Evaluator–Optimizer* elevou a Precisão em diversas funcionalidades (Figura 7), mas revelou vieses do avaliador (Tabela 9) e distribuição quase fixa entre severidades (Tabela 10). Foram observadas inconsistências de segurança (rotas sem campo *security*), erros de sintaxe YAML e dificuldade de respeitar contratos de APIs externas. Além disso, foi percebido um aumento muito peculiar de Hallucination, que, muito provavelmente, pode ser atribuído ao acúmulo de contexto na Context Window. Esta suspeita ocorre porque diversas execuções precisaram ser refeitas em um contexto completamente "limpo". Algumas das saída eram de completa Hallucination e inaceitáveis como resposta para o prompt, enquanto o mesmo problema não ocorria após a limpeza de contexto. Ainda assim, a combinação de geração, avaliação e otimização, com validação humana e verificações estáticas, mostrou-se eficaz para alcançar especificações aceitáveis.

Em termos gerais, os resultados sustentam que LLMs possuem um enorme potencial acelerador de atividades de análise e desenho orientado a contratos (OAS, DDL), mas

não substituem governança técnica, validação sistemática e curadoria. Persistem, contudo, limitações intrínsecas: ausência de memória persistente, dependência da Context Window e natureza não determinística das respostas, o que exige ciclos curtos, explicitação de raciocínio (p.ex., CoT) e auditoria contínua do que é aceito como correto.

Em conclusão, este estudo indica que: (a) há ganho real de produtividade quando LLMs operam sob artefatos canônicos e validações; (b) O uso repetido de Context Window entre os shots tendem a gerar menos aderência; (c) a qualidade depende do reforço explícito de restrições (dialeto SQL, segurança, estilo de contrato) mesmo que em redundância e de checagens automáticas; e (d) a curadoria humana permanece essencial para garantir consistência semântica e conformidade com requisitos, sobretudo em domínios extensos e com múltiplas interações entre artefatos.

5.1 ASPECTOS ÉTICOS, LEGAIS E ORGANIZACIONAIS

O uso intensivo de LLMs em ambientes profissionais de engenharia de software extrapola a escolha de ferramentas e toca questões de responsabilidade coletiva. Do ponto de vista ético, a automação parcial de decisões de análise e desenho exige transparência sobre limitações conhecidas como Hallucination, sensibilidade ao Prompt e dependência da Context Window, e clareza sobre quem valida e assume o produto final. Sem políticas explícitas de revisão humana e registro de decisões, o risco é externalizar a responsabilidade para o modelo e obscurecer a autoria e a obrigação de membros do time ou representativo de prestar contas a instâncias controladoras.

No plano legal, o envio de código, requisitos ou dados a provedores de modelos levanta pontos sensíveis: confidencialidade e propriedade intelectual (o que pode ser retido ou usado para treino), conformidade com leis de proteção de dados, contratos de cliente ou emprego, e aderência aos termos de uso dos serviços. Organizações costumam mitigar isso com classificação de informação, ambientes corporativos ou implantação on-premise, e acordos que delimitem o tratamento de artefatos sensíveis.

Em síntese, enriquece o debate técnico deste trabalho reconhecer que o mesmo potencial acelerador observado experimentalmente convive com deveres de transparência, conformidade e organização do trabalho. A LLM não pode ser responsável pelos artefatos gerados nem pelo vazamento de dados confidenciais, cabendo aos desenvolvedores e operadores tomarem a responsabilidade pelo que é, de fato, implantado.

5.2 MELHORIAS NA DEFINIÇÃO DOS PROMPTS

Os prompts foram construídos utilizando diversas técnicas e foram submetidos a provas e testes antes de chegarem às versões finais utilizadas e demonstradas neste trabalho. Apesar disso, algumas recomendações de melhorias podem ser propostas para esses modelos. O modelo Evaluator-Optimizer poderia ser utilizado em todas as etapas, além da

especificação, o que ajudaria a diminuir a quantidade de validação humana e melhoraria a Precisão dos resultados, apesar de aumentar o custo.

Conforme descrito na seção 4.1, muitas das discrepâncias geradas pela LLM foram descartadas após a validação humana. Ademais, houve uma grande diferença entre a quantidade de itens gerados pelos modelos 1-shot e 0-shot, o que pode ter comprometido a Precisão do modelo 1-shot. Além disso, repetições foram encontradas em iterações subsequentes. Dessa forma, uma recomendação seria limitar a quantidade de itens de saída, definindo um máximo. Outra sugestão seria reintroduzir em iterações subsequentes as discrepâncias geradas para validação e impedir duplicatas, porém isso só é recomendado em modelos que possuam uma Context Window maior.

Nos prompts usados para geração de DDLs, que tiveram a pior performance, como mostrado na seção 4.2, ficou claro que o gerenciamento de Context Window é crítico. Uma sugestão seria inserir redundância na construção dos prompts para duplicar exigências sobre a sintaxe do formato de saída. Um ponto de melhoria seria a forma como o contexto de entrada é introduzido: ao invés de apresentar separadamente diferentes visões de usuários no sistema, apresentar todas as visões de cada funcionalidade, pois algumas tabelas são usadas por apenas uma funcionalidade e seriam criadas de uma vez. Outra sugestão seria reforçar a criação de comentários em colunas de tabela; apesar de não serem suportados por todos os bancos de dados, esses comentários trariam contexto para execuções subsequentes do prompt, permitindo que a LLM consiga entender melhor o que cada coluna representa no sistema. Essas alterações gerariam mais tokens de saída e aumentariam o uso da Context Window, mas poderiam ajudar na melhoria da Precisão.

Por fim, a definição de APIs com o modelo Evaluator-Optimizer se mostrou bem consistente, como visto na seção 4.3. As recomendações seriam apenas para melhorar a qualidade dos artefatos gerados e não sua Precisão. Entre elas, temos boas práticas de escrita de OAS, aumentando a descrição do que seriam problemas MINOR e ERROR para aumentar a gravidade entre as categorias, de forma que ERROR seja grave e não permitido acima da pontuação de corte.

5.3 TRABALHOS FUTUROS

Com base nos achados, destacam-se algumas direções objetivas para continuidade. Primeiramente, seria ideal operacionalizar o padrão *Evaluator–Optimizer* com um Agente Autônomo acoplado a servidores MCP, integrando geração incremental, linters, validações estáticas e testes unitários derivados de ACs, com controle de alterações por *edits* auditáveis;

Além disso avanços na prototipação de um avaliador mais determinístico e calibrado, mitigando viés de pontuação e ponderando severidade e cobertura, combinando ACC, Precisão e métricas de conformidade (segurança, esquema e aderência a contratos exter-

nos);

Futuros estudos podem focar na ampliação de verificações automáticas para OAS (*security* consistente, compatibilidade com APIs de terceiros, sintaxe e semântica) e para DDL (dialeto, chaves e restrições), visando reduzir Hallucination e retrabalho, fazendo o uso de MCPs para integrações com recursos externos garantindo a validações dos artefatos resultantes.

Um ponto que precisa de atenção é a investigação de estratégias de gestão de Context Window e RAG segmentadas por domínio funcional, com sumarização e ancoragem em artefatos canônicos, bem como o impacto de CoT na estabilidade e na auditabilidade das respostas.

Por fim, explorar *fine-tuning* dirigido a domínios de OAS e SQL, enfatizando restrições normativas e dialetos provavelmente traria muitos benefícios as técnicas propostas neste trabalho.

Essas frentes visam consolidar um processo reproduzível, auditável e de baixo custo, no qual LLMs atuem como aceleradores sob governança de artefatos de referência e validações automatizadas, mantendo a curadoria humana como salvaguarda de qualidade.

REFERÊNCIAS

Adobe. **ChatGPT as a Search Engine: How It Works and How to Use It**. 2025. Acessado em: 07 ago. 2025. Disponível em: https://www.adobe.com/express/learn/blog/chatgpt-as-a-search-engine?utm_source=openai.

ANTHROPIC. **Building Effective Agents - Evaluator Optimizer**. 2024. Acesso em: 23 jul. 2025. Disponível em: https://www.anthropic.com/engineering/building-effective-agents#workflow-evaluator-optimizer.

Anthropic, PBC. **Introducing the Model Context Protocol**. 2024. https://www.anthropic.com/news/model-context-protocol. Acesso em 3 de agosto de 2025.

Anthropic, PBC. **Model Context Protocol**. 2024. https://modelcontextprotocol.io/docs/getting-started/intro. Acesso em 3 de agosto de 2025.

Anthropic, PBC. **Model Context Protocol Github Project**. 2024. https://github.com/modelcontextprotocol. Acesso em 3 de agosto de 2025.

CHEN, B. et al. **Unleashing the Potential of Prompt Engineering in Large Language Models: A Comprehensive Review**. 2024. Disponível em: https://arxiv.org/pdf/2310.14735v5.

DEEPSEEK-AI et al. **DeepSeek-V3 Technical Report**. 2025. Disponível em: https://arxiv.org/abs/2412.19437.

DEVLIN, J. et al. BERT: pre-training of deep bidirectional transformers for language understanding. **CoRR**, abs/1810.04805, 2018. Disponível em: http://arxiv.org/abs/1810.04805.

DIFELICIANTONIO, C. **If AI Puts Everyone Out of Work, Is UBI the Solution?** 2024. Accessed: 2025-09-30. Disponível em: https://www.governing.com/finance/if-ai-puts-everyone-out-of-work-is-ubi-the-solution.

FRAGA, N. Challenging llms beyond information retrieval: Reasoning degradation with long context windows. **Preprints**, Preprints, August 2024. Disponível em: https://doi.org/10.20944/preprints202408.1527.v1.

GAO, Y. et al. **Retrieval-Augmented Generation for Large Language Models: A Survey**. 2024. Disponível em: https://arxiv.org/pdf/2312.10997v5.

GURU, W. **Common Agentic Workflow Patterns**. 2025. Acesso em: 23 jul. 2025. Disponível em: https://www.workflows.guru/resources/agentic-workflows-patterns#:~:text=(summary)-,Evaluator%2DOptimizer%20Pattern,-The%20Evaluator%2DOptimizer.

HONG, K.; TROYNIKOV, A.; HUBER, J. **Context Rot: How Increasing Input Tokens Impacts LLM Performance**. [S.l.], 2025. Accessed: 2025-09-30. Disponível em: https://research.trychroma.com/context-rot.

HOU, X. et al. **Large Language Models for Software Engineering: A Systematic Literature Review**. 2024. Disponível em: https://arxiv.org/abs/2308.10620.

HUANG, L. et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. **ACM Transactions on Information Systems**, Association for Computing Machinery (ACM), v. 43, n. 2, p. 1–55, jan. 2025. ISSN 1558-2868. Disponível em: http://dx.doi.org/10.1145/3703155.

LI, J. et al. **HaluEval: A Large-Scale Hallucination Evaluation Benchmark for Large Language Models**. 2023. Disponível em: https://arxiv.org/abs/2305.11747.

LU, E. et al. **MoBA: Mixture of Block Attention for Long-Context LLMs**. 2025. Disponível em: https://arxiv.org/abs/2502.13189.

MELLO, R. de; TRAVASSOS, G.; SILVA, J. Técnica de inspeção baseada em checklist para identificação de defeitos em diagramas de atividades. In: . [S.l.: s.n.], 2011.

OPENAI. **Model - OpenAI API**. 2025. Acesso em: 29 abr. 2025. Disponível em: https://platform.openai.com/docs/models/gpt-4o.

OPENAI. **Model - OpenAI API**. 2025. Acesso em: 29 abr. 2025. Disponível em: https://platform.openai.com/docs/models/gpt-4-turbo.

OPENAI. **Model - OpenAI API: o3**. 2025. Acesso em: 3 maio 2025. Disponível em: https://platform.openai.com/docs/models/o3.

PAN, S. et al. Unifying large language models and knowledge graphs: A roadmap. **IEEE Transactions on Knowledge and Data Engineering**, Institute of Electrical and Electronics Engineers (IEEE), v. 36, n. 7, p. 3580–3599, jul. 2024. ISSN 2326-3865. Disponível em: http://dx.doi.org/10.1109/TKDE.2024.3352100.

ROHERA, P. et al. **Better To Ask in English? Evaluating Factual Accuracy of Multilingual LLMs in English and Low-Resource Languages**. 2025. Disponível em: https://arxiv.org/abs/2504.20022.

TORVALDS, L. **dcn__3__2__0__sh__mask.h — Linux Kernel Source**. 2025. Acesso em: 29 abr. 2025. Disponível em: https://raw.githubusercontent.com/torvalds/linux/refs/heads/master/drivers/gpu/drm/amd/include/asic_reg/dcn/dcn_3_2_0_sh_mask.h.

VASWANI, A. et al. Attention is all you need. **CoRR**, abs/1706.03762, 2017. Disponível em: http://arxiv.org/abs/1706.03762.

WALLACE, C. **Jensen Huang advises against learning to code: leave it up to AI**. 2024. Accessed: 2025-09-30. Disponível em: https://www.tomshardware.com/tech-industry/artificial-intelligence/jensen-huang-advises-against-learning-to-code-leave-it-up-to-ai.

WHITE, J. et al. **A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT**. 2023. Disponível em: https://arxiv.org/abs/2302.11382.

WHITE, J. et al. **A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT**. 2023. Disponível em: https://arxiv.org/pdf/2302.11382v1.

ZHANG, Y.; LI, J.; LIU, P. **Extending LLMs' Context Window with 100 Samples**. 2024. Disponível em: https://arxiv.org/abs/2401.07004.

ZHAO, W. X. et al. **A Survey of Large Language Models**. 2025. Disponível em: https://arxiv.org/pdf/2303.18223.

GLOSSÁRIO

AC Critérios de Aceite – condições específicas que uma funcionalidade deve atender para ser considerada completa e aceitável pelo usuário ou stakeholder.

ACC Acurácia – mede a proporção de acertos de um modelo de classificação.

Agente Autônomo Entidade autônoma que toma decisões com base em percepções do ambiente, sem depender de controle central.

AI Inteligência Artificial – ramo da computação que desenvolve sistemas capazes de simular comportamentos inteligentes.

ATTN Attention – mecanismo essencial em modelos de linguagem que permite ao modelo ponderar a relevância de diferentes partes da entrada ao gerar cada saída, possibilitando a modelagem de dependências contextuais de curto e longo alcance.

Context Window Limite de tamanho (em tokens ou palavras) que um modelo de linguagem consegue considerar de uma vez na entrada. Quanto maior, mais informação o modelo pode processar ao mesmo tempo.

CoT Chain of Thought – técnica de construção de prompts em que o modelo é estimulado a apresentar seu raciocínio passo a passo antes de fornecer uma resposta final, aumentando a precisão em tarefas complexas.

DDL Data Definition Language – conjunto de comandos SQL utilizados para definir, alterar ou remover estruturas de dados como tabelas, esquemas e índices em um banco de dados.

Fine-tuning Processo de ajuste fino de modelos de linguagem.

Hallucination Fenômeno em que um modelo de linguagem gera informações incorretas, inventadas ou factualmente infundadas, mesmo que a resposta pareça plausível ou bem formulada.

LeetCode Plataforma de desafios algorítmicos usada para treinar lógica, algoritmos e estruturas de dados.

LLM Large Language Model – um modelo de linguagem treinado com grandes volumes de texto para entender e gerar linguagem natural.

MCP Model Context Protocol – protocolo que define como diferentes aplicações, ferramentas e modelos de linguagem podem compartilhar e estruturar informações de contexto de maneira padronizada.

MVP Minimum Viable Product – versão inicial de um produto com o mínimo de funcionalidades necessárias para validar sua proposta de valor com usuários reais e obter feedback rápido.

OAS OpenAPI Specification – especificação padronizada para descrever, documentar e consumir APIs RESTful, permitindo geração automática de documentação e código cliente/servidor.

Oráculo Entidade hipotética que fornece respostas corretas instantaneamente para decisões ou problemas computacionais, usada como referência teórica em algoritmos e aprendizado supervisionado.

PLM Pre-treined Language Model - um modelo de linguagem pré-treinado e refinado para entender e gerar linguagem natural.

Precisão Precisão – mede quantas das previsões positivas estavam corretas.

Prompt Conjunto de instruções que é fornecido como entrada para a LLM através de um texto escrito em linguagem natural.

RAG Retrieval-Augmented Generation – técnica que combina busca de informações externas com geração de texto por modelos de linguagem.

Reasoning Técnica de engenharia de prompts que busca estruturar uma espécie de raciocínio de resposta da LLM por meio de etapas explícitas, como pensamento passo a passo, cadeias de pensamento ou decomposição de problemas.

RLHF Reinforcement Learning from Human Feedback – técnica que utiliza aprendizado por reforço guiado por avaliações humanas para alinhar o comportamento do modelo com expectativas humanas.

SFT Supervised Fine-Tuning – ajuste supervisionado de um modelo de linguagem utilizando exemplos rotulados para melhorar seu desempenho em tarefas específicas.

Token Menor unidade de texto processada por um determinado modelo. Pode variar de tamanho de acordo com a necessidade.

Transformer Arquitetura de rede neural baseada em mecanismos de atenção, usada principalmente em tarefas de processamento de linguagem natural em LLMs.

US Histórias de Usuário – descrições concisas de funcionalidades de software escritas do ponto de vista do usuário final.

Workflow de LLM Sequência orquestrada de etapas que envolvem a preparação, execução e manipulação de interações com modelos de linguagem, com o objetivo de resolver uma tarefa complexa ou automatizar um processo.

APÊNDICE A – HISTÓRIAS DE USUÁRIOS NÃO REVISADAS

A.1 REQUISITOS NÃO FUNCIONAIS

API Gateway

- **NFR-6: Java (Spring Cloud Gateway)** - Deve ser feito utilizando o framework de Gateways.
- **NFR-7: Redis (Cache)** - Deve possuir cache para agilizar respostas para o usuário.
- **NFR-8: Reativo** - Deve ser não bloqueante.

Backend

- **NFR-9: Java (Spring Boot)** - Deve ser feito utilizando o framework Spring Boot.
- **NFR-10: Postgres** - Deve possuir um banco relacional e armazenar imagens no banco de dados.
- **NFR-11: Redis** - Deve possuir cache para agilizar e diminuir o impacto no banco de dados.

Detalhes Técnicos e Arquiteturais

- **NFR-12: OAuth2** - Autenticação e autorização devem seguir o padrão OAuth2.
- **NFR-13: Envio de Email** - A notificação ao usuário deve ser realizada por email.
- **NFR-14: Método de Pagamento Stripe** - Deve possuir um método de pagamento de mercado como o Stripe.
- **NFR-17: Microsserviços** - A arquitetura de backend deve ser baseada em microsserviços.
- **NFR-18: Consistência de Respostas** - A interface de usuário e o backend devem fornecer respostas consistentes e completas para todas as funcionalidades principais.
- **NFR-19: Otimização de Recursos** - O uso de cache no Redis (NFR-7 e NFR-11) deve reduzir pelo menos 30% das chamadas ao banco de dados.
- **NFR-20: Documentação Técnica Completa** - Todo o código deve estar documentado, incluindo dependências e configurações, para facilitar a manutenção e expansão futura.

- **NFR-21: Testes Unitários** - O sistema deve permitir a criação de testes unitários e de integração, com cobertura mínima de 80%.
- **NFR-22: Tolerância a Falhas** - O sistema deve possuir mecanismos de recuperação para operações críticas, como transações de pagamento.
- **NFR-23: Alta Disponibilidade** - O backend deve ter alta disponibilidade, garantindo 90% de uptime.
- **NFR-24: Tempo de Resposta** - Todas as respostas da API devem ser processadas em um tempo médio inferior a 500ms para melhorar a experiência do usuário.
- **NFR-25: OWASP v4.0.3-1 Nível 1** - O sistema deve atender aos requisitos de segurança de nível 1 da versão v4.0.3-1 da OWASP.

A.2 MÓDULO 1 - USUÁRIO GERAL

US-1: Cadastro de novo usuário

- **Descrição:** Como um novo usuário, eu quero me cadastrar no marketplace para que eu possa começar a comprar e vender produtos.
- **Critérios de Aceite:**
  - US-1-AC-1: O usuário pode se cadastrar preenchendo campos obrigatórios como nome, email e senha.
  - US-1-AC-2: O sistema valida a senha e verifica se o email já está cadastrado.
  - US-1-AC-3: O cadastro é concluído com sucesso e o usuário é redirecionado para a página de confirmação de email.

US-2: Confirmação de email

- **Descrição:** Como um usuário cadastrado, eu quero confirmar meu endereço de email após o cadastro para que eu possa validar minha conta e começar a usá-la.
- **Critérios de Aceite:**
  - US-2-AC-1: Um email de confirmação é enviado automaticamente após o cadastro.
  - US-2-AC-2: O email contém um link de verificação que, ao ser clicado, valida a conta do usuário.
  - US-2-AC-3: O usuário só pode acessar as funcionalidades de compra e venda após confirmar o email.

US-3: Recuperação de senha

- **Descrição:** Como um usuário que perdeu a senha, eu gostaria de conseguir recuperar o meu acesso através do meu email, para que eu possa criar uma nova senha.
- **Critérios de Aceite:**
  - US-3-AC-1: O usuário pode solicitar a recuperação de senha fornecendo seu email.
  - US-3-AC-2: Um link de redefinição de senha é enviado para o email do usuário.
  - US-3-AC-3: O usuário pode redefinir sua senha através do link e fazer login com a nova senha.

US-4: Sessões simultâneas e OAuth2

- **Descrição:** Como usuário cadastrado, quero acessar a plataforma em vários dispositivos simultaneamente.
- **Critérios de Aceite:**
  - US-4-AC-1: O usuário pode logar e manter sessões ativas em múltiplos dispositivos sem interrupções.
  - US-4-AC-2: A autenticação precisa ser no padrão OAuth2.
  - US-4-AC-3: Sempre que um refresh token for utilizado mais de uma vez, todas as sessões do usuário devem ser encerradas.

US-5: Atualização de perfil

- **Descrição:** Como um usuário logado, eu quero atualizar minhas informações de perfil para que meus dados estejam sempre corretos e atualizados.
- **Critérios de Aceite:**
  - US-5-AC-1: O usuário logado pode acessar e editar seu perfil, incluindo nome, email, e endereço.
  - US-5-AC-2: O sistema salva e reflete imediatamente as atualizações.

US-6: Logout seguro

- **Descrição:** Como um usuário logado, eu quero fazer logout da aplicação para que eu possa garantir a segurança da minha conta.
- **Critérios de Aceite:**

  - US-6-AC-1: O usuário pode fazer logout de forma segura, encerrando a sessão atual.
  - US-6-AC-2: Após o logout, o sistema impede o acesso a áreas restritas até que o usuário faça login novamente.

US-7: Notificação de logins

- **Descrição:** Como usuário validado, eu quero receber notificações quando houver um login na minha conta para que eu possa ser alertado sobre acessos não autorizados.
- **Critérios de Aceite:**
  - US-7-AC-1: O usuário recebe notificações via email ou push quando um login é feito em sua conta.
  - US-7-AC-2: As notificações incluem informações sobre o dispositivo e a localização aproximada do login.
  - US-7-AC-3: O email de aviso deve conter um link para bloquear acessos não identificados pelo usuário.

US-8: Gerenciamento de sessões ativas

- **Descrição:** Como usuário logado, eu quero visualizar e gerenciar minhas sessões ativas para que eu possa encerrar sessões em dispositivos que não estou mais usando.
- **Critérios de Aceite:**
  - US-8-AC-1: O usuário pode visualizar todas as sessões ativas.
  - US-8-AC-2: O usuário pode encerrar manualmente sessões indesejadas em dispositivos específicos.

US-9: Visualização de avaliações

- **Descrição:** Como um visitante do site, eu quero ler avaliações de outros usuários para que eu possa tomar decisões de compra mais informadas e obter detalhes sobre produtos.
- **Critérios de Aceite:**
  - US-9-AC-1: O visitante pode acessar a página do produto e visualizar avaliações deixadas por outros compradores.
  - US-9-AC-2: As avaliações são exibidas de forma organizada, com notas e comentários.

US-10: Suporte ao cliente

- **Descrição:** Como usuário logado e validado, eu quero acessar o suporte ao cliente facilmente para que eu possa resolver qualquer problema que encontrar.
- **Critérios de Aceite:**
  - US-10-AC-1: O usuário pode acessar o suporte através de um botão ou link visível na interface.
  - US-10-AC-2: O sistema redireciona para um formulário de contato.

US-11: Exclusão de conta

- **Descrição:** Como usuário logado, eu quero excluir minha conta permanentemente para que meus dados sejam removidos do sistema.
- **Critérios de Aceite:**
  - US-11-AC-1: O usuário pode solicitar a exclusão da conta por meio de uma opção na área de perfil.
  - US-11-AC-2: Após confirmação, todos os dados do usuário são removidos do sistema, e a conta é desativada.
  - US-11-AC-3: Se houverem assuntos pendentes do usuário como compras ou vendas em processamento, a conta não é encerrada.

A.3 MÓDULO 2 - USUÁRIO ADMINISTRADOR

US-12: Gerenciamento de permissões de acesso

- **Descrição:** Como administrador, eu quero gerenciar detalhadamente as permissões de acesso dos usuários, incluindo papeis e níveis de acesso específicos, para que eu possa garantir que cada usuário tenha acesso somente às funcionalidades necessárias para sua função, mantendo a segurança do sistema.
- **Critérios de Aceite:**
  - US-12-AC-1: O administrador pode criar, editar e remover permissões de acesso para diferentes tipos de usuários.
  - US-12-AC-2: As permissões podem ser configuradas por nível de acesso e papéis (administrador, vendedor, comprador).

US-13: Gerenciamento de administradores

- **Descrição:** Como administrador, eu quero poder cadastrar, editar e remover outros administradores, com diferentes níveis de privilégio, para que o sistema possa ser gerenciado de forma segura e eficiente por múltiplos responsáveis.
- **Critérios de Aceite:**
  - US-13-AC-1: O administrador pode adicionar novos administradores, editar suas permissões e removê-los.
  - US-13-AC-2: Diferentes níveis de privilégio são atribuídos aos administradores conforme necessário.

US-14: Moderação de anúncios

- **Descrição:** Como administrador, eu quero bloquear e desbloquear anúncios que não estejam em conformidade com as políticas da plataforma, para manter a qualidade e segurança dos produtos oferecidos no marketplace.
- **Critérios de Aceite:**
  - US-14-AC-1: O administrador pode bloquear ou desbloquear anúncios que violam as políticas da plataforma.
  - US-14-AC-2: Um motivo deve ser fornecido ao bloquear um anúncio, e o vendedor é notificado.

US-15: Moderação de usuários

- **Descrição:** Como administrador, eu quero bloquear e desbloquear diferentes tipos de usuários (vendedores, compradores, lojistas), para que eu possa moderar comportamentos inadequados e garantir a segurança e confiabilidade da plataforma.
- **Critérios de Aceite:**
  - US-15-AC-1: O administrador pode bloquear ou desbloquear contas de usuários que violam as regras da plataforma.
  - US-15-AC-2: O usuário bloqueado é notificado via email sobre a ação tomada.
  - US-15-AC-3: Quando um comprador é bloqueado, ele não pode realizar nenhuma compra ou comentar anúncios.
  - US-15-AC-4: Quando um vendedor é bloqueado, todos os anúncios são pausados, e ele não pode responder nenhuma avaliação.

US-16: Moderação de perguntas e respostas

- **Descrição:** Como administrador, eu quero revisar, editar ou remover perguntas e respostas que sejam inapropriadas, falsas ou que violem as regras da plataforma, para manter um ambiente de compra e venda transparente e confiável.
- **Critérios de Aceite:**
  - US-16-AC-1: O administrador pode revisar e remover perguntas ou respostas que sejam inadequadas.
  - US-16-AC-2: Perguntas e respostas removidas desaparecem imediatamente da página do produto.

US-17: Gerenciamento de pedidos de devolução

- **Descrição:** Como administrador, eu quero autorizar ou rejeitar pedidos de devolução com base em critérios estabelecidos, como conformidade com a política de devoluções e análise do estado do produto, para garantir uma experiência justa tanto para compradores quanto para vendedores.
- **Critérios de Aceite:**
  - US-17-AC-1: O administrador pode revisar pedidos de devolução com base nas políticas de devolução.
  - US-17-AC-2: O administrador pode aceitar ou rejeitar o pedido, e o comprador e vendedor são notificados da decisão.

US-18: Visualização de suporte ao cliente

- **Descrição:** Como administrador, eu quero acessar o suporte ao cliente e visualizar os tickets em andamento, incluindo o progresso e a responsabilidade de cada ticket, para acompanhar e garantir a resolução eficiente dos problemas relatados pelos usuários.
- **Critérios de Aceite:**
  - US-18-AC-1: O administrador pode visualizar e gerenciar tickets de suporte em andamento.
  - US-18-AC-2: O progresso e a responsabilidade de cada ticket são atualizados conforme necessário.

US-19: Acompanhamento de desempenho de lojas

- **Descrição:** Como administrador, eu desejo visualizar estatísticas detalhadas de uma loja, como faturamento total, comissão repassada à plataforma, número de vendas e crescimento ao longo do tempo, para monitorar o desempenho financeiro da plataforma e dos lojistas.

- **Critérios de Aceite:**
  - US-19-AC-1: O administrador pode acessar estatísticas detalhadas sobre o desempenho de lojas e vendedores.
  - US-19-AC-2: Os dados incluem faturamento, número de vendas e crescimento ao longo do tempo.

A.4 MÓDULO 3 - USUÁRIO COMPRADOR

US-20: Pesquisa de produtos

- **Descrição:** Como um comprador, eu quero pesquisar produtos por categoria e preço para que eu possa encontrar o que estou procurando facilmente.
- **Critérios de Aceite:**
  - US-20-AC-1: O comprador pode pesquisar produtos usando filtros de categoria e faixa de preço.
  - US-20-AC-2: Os resultados da pesquisa são exibidos corretamente conforme os critérios aplicados.

US-21: Adicionar produtos ao carrinho

- **Descrição:** Como um comprador, eu quero adicionar produtos ao meu carrinho de compras para que eu possa revisá-los antes de finalizar a compra.
- **Critérios de Aceite:**
  - US-21-AC-1: O comprador pode adicionar produtos ao carrinho, e os produtos aparecem na lista de revisão antes da compra.
  - US-21-AC-2: O carrinho armazena os produtos até que a compra seja finalizada ou o carrinho seja esvaziado.
  - US-21-AC-3: O carrinho é armazenado no dispositivo com a sessão ativa.
  - US-21-AC-4: Se o comprador fechar o site e abrir novamente, as informações do carrinho devem ser recuperadas.

US-22: Gerenciamento de endereços

- **Descrição:** Como um comprador, quero salvar meus endereços de entrega para facilitar futuras compras.
- **Critérios de Aceite:**

  - US-22-AC-1: O comprador pode salvar múltiplos endereços de entrega para futuras compras.
  - US-22-AC-2: Os endereços salvos podem ser editados ou excluídos a qualquer momento.

US-23: Perguntar ao vendedor

- **Descrição:** Como um comprador, eu quero fazer perguntas ao vendedor sobre um produto antes de comprar, para que eu possa esclarecer dúvidas antes de finalizar a compra.
- **Critérios de Aceite:**
  - US-23-AC-1: O comprador pode enviar perguntas diretamente ao vendedor por meio da página do produto.
  - US-23-AC-2: O sistema notifica o comprador quando o vendedor responde.

US-24: Pagamento seguro

- **Descrição:** Como um comprador, eu quero um processo de pagamento seguro e fácil para que eu possa concluir minhas compras com confiança.
- **Critérios de Aceite:**
  - US-24-AC-1: O comprador pode concluir o pagamento usando Stripe, e os dados de pagamento são processados de forma segura.
  - US-24-AC-2: No Stripe devem ser habilitadas as cobranças de Pix, Boleto e Cartão de Crédito.
  - US-24-AC-3: O sistema exibe uma confirmação de compra após o pagamento bem-sucedido.
  - US-24-AC-4: Caso o pagamento não seja bem-sucedido, o usuário deve ser notificado e a compra deve ser cancelada.
  - US-24-AC-5: O processo de pagamento deve ser idempotente, para garantir que o usuário seja tarifado apenas uma vez, e que o callback de pagamento seja válido uma vez, e não altere o status do pagamento.
  - US-24-AC-6: O pagamento deve ser retido e somente liberado após o prazo final de entrega (sem pedidos válidos de reembolso), ou a confirmação do recebimento do produto.

US-25: Cancelamento de pedidos

- **Descrição:** Como um comprador, eu quero cancelar meu pedido enquanto ele estiver sendo processado, para que eu possa mudar de ideia antes do envio.

- **Critérios de Aceite:**
  - US-25-AC-1: O comprador pode cancelar o pedido enquanto ele ainda estiver sendo processado.
  - US-25-AC-2: O sistema notifica o comprador sobre o cancelamento e reembolsa automaticamente.

US-26: Notificações sobre o pedido

- **Descrição:** Como um comprador, eu quero ser notificado em todas as etapas do meu pedido, para que eu possa saber quando o pagamento foi aprovado, quando foi separado, quando o pedido foi enviado e problemas durante esse processo.
- **Critérios de Aceite:**
  - US-26-AC-1: O comprador recebe notificações em tempo real sobre o status do pedido (aprovado, enviado, etc.).
  - US-26-AC-2: O comprador pode visualizar o histórico completo do pedido na sua conta.

US-27: Avaliação de produtos

- **Descrição:** Como um comprador, eu quero avaliar o produto após receber meu produto, para que eu possa compartilhar minha experiência e ajudar outros compradores.
- **Critérios de Aceite:**
  - US-27-AC-1: O comprador pode avaliar o produto e deixar um comentário sobre a sua experiência.
  - US-27-AC-2: A avaliação aparece publicamente na página do produto após a moderação.

US-28: Histórico de pedidos

- **Descrição:** Como um comprador, eu quero um histórico de todos os pedidos que fiz, para que eu possa acompanhar minhas compras e revisar informações sobre produtos adquiridos anteriormente.
- **Critérios de Aceite:**
  - US-28-AC-1: O comprador pode visualizar um histórico detalhado de todas as suas compras anteriores.
  - US-28-AC-2: O histórico inclui informações como data da compra, valor e status do pedido.

US-29: Devolução de produtos

- **Descrição:** Como um comprador, eu quero devolver um produto por problemas técnicos ou exercer meu direito de arrependimento, para que eu possa receber o reembolso após a confirmação da devolução do item pelo vendedor ou comprovação de fraude.
- **Critérios de Aceite:**
  - US-29-AC-1: O comprador pode solicitar a devolução de um produto através da interface de pedidos.
  - US-29-AC-2: O sistema processa a devolução conforme a política de devolução, e o comprador recebe o reembolso após confirmação.

US-30: Lista de favoritos

- **Descrição:** Como comprador, quero adicionar produtos à minha lista de favoritos, para lembrar de itens que eu possa querer comprar no futuro.
- **Critérios de Aceite:**
  - US-30-AC-1: O comprador pode adicionar produtos a uma lista de favoritos, acessível a qualquer momento.
  - US-30-AC-2: Adicionar um produto à lista deve ser idempotente. Ou seja, somente deve se adicionar um produto que não exista, mas o usuário não deve visualizar um erro caso a interface esteja desatualizada.
  - US-30-AC-3: Antes de adicionar um produto à lista de favoritos, deve ser possível verificar se o produto já está presente, trocando a ação de adicionar para remover da lista.
  - US-30-AC-4: A lista pode ser editada para remover ou adicionar produtos.

US-31: Denúncia de conteúdo

- **Descrição:** Como comprador eu quero poder relatar conteúdos inapropriados ou fraudulentos na plataforma.
- **Critérios de Aceite:**
  - US-31-AC-1: O comprador pode relatar conteúdos inadequados através de um botão dedicado.
  - US-31-AC-2: O sistema processa a denúncia e o conteúdo é revisado por um administrador.

A.5 MÓDULO 4 - USUÁRIO VENDEDOR

US-32: Gerenciamento de inventário de produtos

- **Descrição:** Como um vendedor, eu quero gerenciar meu inventário de produtos para que eu possa manter minhas ofertas atualizadas.
- **Critérios de Aceite:**
  - US-32-AC-1: O vendedor pode adicionar, editar e remover produtos de seu inventário.
  - US-32-AC-2: O estoque é atualizado automaticamente com base nas vendas.

US-33: Edição de descrições detalhadas dos produtos

- **Descrição:** Como um vendedor, eu quero criar e editar descrições detalhadas dos meus produtos para que os compradores possam entender melhor o que estão comprando.
- **Critérios de Aceite:**
  - US-33-AC-1: O vendedor pode criar ou editar descrições de produtos de forma fácil.
  - US-33-AC-2: A descrição atualizada não deve impactar um produto que já foi vendido, ou seja, devem ser armazenados os dados do produto no histórico de compra.
  - US-33-AC-3: As descrições são atualizadas em tempo real na página do produto.

US-34: Upload e otimização de fotos dos produtos

- **Descrição:** Como vendedor, eu quero adicionar fotos aos meus produtos para que eles se destaquem visualmente no marketplace.
- **Critérios de Aceite:**
  - US-34-AC-1: O vendedor pode fazer upload de fotos de alta qualidade para seus produtos.
  - US-34-AC-2: O sistema redimensiona as imagens conforme necessário para garantir um carregamento rápido.
  - US-34-AC-3: A otimização da imagem deve ser feita com a funcionalidade do Next de otimização de imagem.

US-35: Visualização de estatísticas de vendas

- **Descrição:** Como vendedor, eu quero visualizar estatísticas de vendas de cada produto (como número de visualizações e conversões) para que eu possa ajustar minhas estratégias de marketing e preços.
- **Critérios de Aceite:**
  - US-35-AC-1: O vendedor pode visualizar o número de visualizações de cada produto na página de gerenciamento de inventário.
  - US-35-AC-2: O vendedor pode ver a taxa de conversão (visualizações para compras) de cada produto.
  - US-35-AC-3: As estatísticas exibem dados em gráficos ou tabelas que mostram informações ao longo do tempo.
  - US-35-AC-4: O vendedor pode filtrar as estatísticas por intervalo de tempo (dia, semana, mês).

US-36: Histórico de interações com clientes

- **Descrição:** Como um vendedor, eu quero ver o histórico de interações com os clientes, como perguntas e respostas, para que eu possa responder rapidamente a novas perguntas e acompanhar o feedback.
- **Critérios de Aceite:**
  - US-36-AC-1: O vendedor pode visualizar uma lista de perguntas recebidas para cada produto em uma página dedicada ou na área de gerenciamento de produtos.
  - US-36-AC-2: O vendedor pode acessar o histórico completo de perguntas e respostas de clientes para cada produto.
  - US-36-AC-3: Novas perguntas são destacadas ou categorizadas para facilitar a identificação.
  - US-36-AC-4: O vendedor é notificado quando há novas perguntas não respondidas.
  - US-36-AC-5: O sistema mantém o histórico de interações ordenado cronologicamente e acessível.

US-37: Convite de usuários para a loja

- **Descrição:** Como vendedor que gerencia uma loja, eu quero convidar outros usuários a criarem contas em minha loja.
- **Critérios de Aceite:**

  - US-37-AC-1: O vendedor pode enviar convites via email ou link gerado para que outros usuários criem contas vinculadas à sua loja.
  - US-37-AC-2: O vendedor pode definir as permissões e o papel (gerente, assistente, etc.) para cada novo usuário convidado.
  - US-37-AC-3: O usuário convidado recebe um link de inscrição e, após a criação da conta, é adicionado automaticamente à equipe da loja.
  - US-37-AC-4: O vendedor pode visualizar e gerenciar todos os usuários vinculados à sua loja, incluindo alterar permissões ou remover usuários.

US-38: Gestão financeira e saldo do vendedor

- **Descrição:** Como um vendedor, eu quero gerenciar meu saldo e ver os pagamentos recebidos, incluindo taxas e deduções da plataforma, para ter controle financeiro total das minhas vendas.
- **Critérios de Aceite:**
  - US-38-AC-1: O vendedor pode visualizar seu saldo disponível e ver os pagamentos recebidos em uma interface clara.
  - US-38-AC-2: O vendedor pode ver o histórico de transações, incluindo o valor bruto das vendas, taxas de transação, e deduções aplicadas pela plataforma.
  - US-38-AC-3: O vendedor pode gerar relatórios financeiros detalhados de períodos específicos.
  - US-38-AC-4: O sistema exibe as taxas e deduções aplicadas de forma transparente para cada venda.
  - US-38-AC-5: O vendedor pode solicitar saques de seu saldo para uma conta bancária vinculada.

US-39: Suspensão ou remoção temporária de produtos

- **Descrição:** Como um vendedor, eu quero poder suspender ou remover produtos temporariamente do marketplace quando houver necessidade de ajustes ou indisponibilidade de estoque.
- **Critérios de Aceite:**
  - US-39-AC-1: O vendedor pode suspender temporariamente a exibição de um produto no marketplace sem removê-lo permanentemente.
  - US-39-AC-2: Produtos suspensos permanecem visíveis no painel de gerenciamento de produtos com um status que indica que estão inativos.

  - US-39-AC-3: O vendedor pode reativar ou remover permanentemente um produto com um simples clique.
  - US-39-AC-4: Produtos que possuem vendas pendentes não podem ser removidos, apenas suspensos.

US-40: Responder avaliações de clientes

- **Descrição:** Como um vendedor, eu quero responder diretamente às avaliações dos clientes sobre meus produtos para melhorar minha reputação e resolver qualquer questão.
- **Critérios de Aceite:**
  - US-40-AC-1: O vendedor pode visualizar todas as avaliações deixadas por clientes em seus produtos em uma página de gerenciamento de avaliações.
  - US-40-AC-2: O vendedor pode responder a cada avaliação diretamente pela interface da plataforma.
  - US-40-AC-3: As respostas do vendedor são exibidas junto com a avaliação do cliente, visíveis para todos os usuários.
  - US-40-AC-4: O sistema notifica o vendedor sempre que uma nova avaliação é feita em um de seus produtos.
  - US-40-AC-5: O vendedor pode editar ou excluir suas respostas, desde que dentro de um período de tempo definido.

US-41: Notificações de venda

- **Descrição:** Como vendedor, eu quero receber notificações quando um produto for vendido para que eu possa preparar o envio rapidamente.
- **Critérios de Aceite:**
  - US-41-AC-1: O vendedor recebe notificações em tempo real via email ou push sempre que um produto é vendido.
  - US-41-AC-2: As notificações contêm detalhes da venda, incluindo o nome do comprador, endereço de envio, e informações do produto vendido.
  - US-41-AC-3: O sistema permite que o vendedor veja um histórico de todas as notificações de vendas recebidas.
  - US-41-AC-4: O vendedor pode configurar preferências para o tipo de notificações (email, SMS, push) e a frequência delas.

US-42: Atualização de status de pedidos

- **Descrição:** Como um vendedor, eu quero atualizar o status do pedido, para manter meu cliente informado.
- **Critérios de Aceite:**
  - US-42-AC-1: O vendedor pode alterar o status do pedido diretamente na plataforma (ex: "Processando", "Enviado", "Entregue", "Cancelado").
  - US-42-AC-2: Cada mudança de status é registrada e visível para o vendedor e o comprador.
  - US-42-AC-3: O sistema notifica automaticamente o comprador por email ou push sempre que o status do pedido é atualizado.
  - US-42-AC-4: O vendedor pode visualizar o histórico completo de status de um pedido em uma página de detalhes do pedido.
  - US-42-AC-5: O vendedor pode adicionar comentários ou notas ao atualizar o status do pedido, que são visíveis para o cliente.
  - US-42-AC-6: A interface de atualização de status é de fácil acesso no painel de controle do vendedor, com uma lista de todos os pedidos pendentes e concluídos.
  - US-42-AC-7: O vendedor é impedido de alterar o status de pedidos que já tenham sido finalizados (ex: "Entregue").

APÊNDICE B – CATALOGAÇÃO DE PROMPTS

B.1 PROMPT PARA AVALIAÇÃO DE DISCREPÂNCIA 0-SHOT

Nome e classificação: Gerador de discrepâncias de inspeção de requisitos. **Identificação de erros, 0 Shot**

Intenção e contexto: Gerar uma ou mais linhas de um documento de relatório de discrepâncias para um contexto de requisitos funcionais de um sistema, como histórias de usuário

Motivação: Inspeção de software detecta e identifica anomalias de software, incluindo erros e desvios de padrões das especificações de software com o objetivo de encontrar defeitos. Gerar um documento de relato de discrepâncias ajuda a identificar discrepâncias no projeto.

Estrutura e ideais chave: Um passo a passo de como gerar o relatório incluindo um exemplo de uma linha de discrepância. Uma classificação de discrepâncias que guiará o LLM a revisar as histórias de usuário. Um contexto de todas as histórias de usuário do projeto.

Pontos fortes: Obter um breve resumo de possíveis discrepâncias nos requisitos do sistema.

Limitações: Não se pode confiar que todas as discrepâncias serão relatadas pelo LLM.

Exemplo de implementação:

Pode ser encontrado no código 1

Código 1 – Avaliação de Discrepâncias 0-Shot

```
Como um engenheiro de software especializado em inspeção de software. Valide todas as seguintes historias
de usuario e seus criterios de aceite (ambos, nao somente um ou outro) e descreva o relatorio de
discrepancias num formato csv onde cada linha deve seguir o formato abaixo. Para entender as falhas e
discrepancia nos requisitos desenvolvidos, voce pode identificar uma discrepancia sobre historias de
usuario em si e/ou discrepancias sobre um ou mais criterios de aceite de historia de usuario e portanto
deve avaliar ambos separadamente. Nao faca suposicoes, caso nao consiga extrair dos requisitos as
informacoes necessarias para a construcao do modelo de projeto, relate essa omissao de informacoes como
discrepancias, ao inves de tentar deduzi-las. Nao tente corrigir eventuais discrepancias no documento de
requisitos, apenas descreva-as de forma a possibilitar a correcao. Lembre-se de pensar etapa por etapa.

O formato de saida deve possuir um "Id" que e o id unico numero incremental da discrepancia, um campo
"item de avaliacao" relacionado ao codigo da historia de usuario ou o codigo do criterio de aceite, um
campo "Tipo de defeito" que e um dos defeitos da lista de defeitos abaixo, um campo "Descricao" que e uma
breve descricao da discrepancia, um campo "Localizacao" que e a raia responsavel pela acao da historia de
usuario ou criterio de aceite. um campo "Locais onde se repete" outras historias de usuario ou criterios
de aceite onde se repete

"""Lista de Defeitos
* Omissao (Informacoes necessarias foram omitidas do artefato)
* Fato incorreto (Algumas informacoes no artefato de software contradizem informacoes presentes no oraculo
ou o
proprio conhecimento geral do dominio)
* Inconsistencia (As informacoes em uma parte do artefato de software estao inconsistentes com outras no
artefato
de software).
* Ambiguidade (As informacoes no artefato de software sao ambiguas, isto e, e possivel ao desenvolvedor
interpretar as informacoes de diferentes maneiras, podendo levar a uma implementacao incorreta).
* Informacao estranha (As informacoes sao fornecidas, mas nao sao necessarias ou mesmo usadas.)
"""

"""Historias de usuario para avaliacao

"""
```

B.2 PROMPT PARA AVALIAÇÃO DE DISCREPÂNCIA 1-SHOT

Nome e classificação: Gerador de discrepâncias de inspeção de requisitos. **Identificação de erros, 1 Shot**

Intenção e contexto: Gerar uma ou mais linhas de um documento de relatório de discrepâncias para um contexto de requisitos funcionais de um sistema, como histórias de usuário

Motivação: Inspeção de software detecta e identifica anomalias de software, incluindo erros e desvios de padrões das especificações de software com o objetivo de encontrar defeitos. Gerar um documento de relato de discrepâncias ajuda a identificar discrepâncias no projeto.

Estrutura e ideais chave: Um passo a passo de como gerar o relatório incluindo um exemplo de uma linha de discrepância. Uma classificação de discrepâncias que guiará o LLM a revisar as histórias de usuário. Um contexto de todas as histórias de usuário do projeto.

Pontos fortes: Obter um breve resumo de possíveis discrepâncias nos requisitos do sistema.

Limitações: Não se pode confiar que todas as discrepâncias serão relatadas pelo LLM.

Exemplo de implementação: Pode ser encontrado no código 2

Código 2 – Avaliação de Discrepâncias 1-Shot

```
Como um engenheiro de software especializado em inspecao de software. Valide todas as seguintes historias
de usuario e seus criterios de aceite (ambos nao somente um ou outro) e descreva o relatorio de
discrepancias num formato csv onde cada linha deve seguir o formato abaixo. Para entender as falhas e
discrepancia nos requisitos desenvolvidos, voce pode identificar uma discrepancia sobre historias de
usuario em si e/ou discrepancias sobre um ou mais criterios de aceite de historia de usuario e portanto
deve avaliar ambos separadamente.

Nao faca suposicoes, caso nao consiga extrair dos requisitos as informacoes necessarias para a construcao
do modelo de projeto, relate essa omissao de informacoes como discrepancias, ao inves de tentar
deduzi-las. Nao tente corrigir eventuais discrepancias no documento de requisitos, apenas descreva-as de
forma a possibilitar a correcao.

Lembre-se de pensar etapa por etapa.

"""Formato de saida
id,item de avaliacao relacionado, Tipo de defeito, Descricao, Localizacao, Locais aonde se repete
"""

Onde "Id" e o id unico numero incremental da discrepancia, "item de avaliacao" relacionado e o codigo da
historia de usuario ou o codigo do criterio de aceite, "Tipo de defeito" e um dos defeitos da lista de
defeitos abaixo, "Descricao" e uma breve descricao da discrepancia, "Localizacao" e a raia responsavel
pela acao da historia de usuario ou criterio de aceite. "Locais onde se repete" outras historias de
usuario ou criterios de aceite onde se repete

"""Lista de Defeitos
* Omissao (Informacoes necessarias foram omitidas do artefato)
* Fato incorreto (Algumas informacoes no artefato de software contradizem informacoes presentes no oraculo
ou o
proprio conhecimento geral do dominio)
* Inconsistencia (As informacoes em uma parte do artefato de software estao inconsistentes com outras no
artefato
de software).
* Ambiguidade (As informacoes no artefato de software sao ambiguas, isto e, e possivel ao desenvolvedor
interpretar as informacoes de diferentes maneiras, podendo levar a uma implementacao incorreta).
* Informacao estranha (As informacoes sao fornecidas, mas nao sao necessarias ou mesmo usadas.)
"""

"""Exemplo de linha no formato de saida
5, us-12, Fato incorreto, somente usuarios administradores podem aprovar reembolsos, usuarios lojistas,
nenhum
5, us-12-AC-1, Fato incorreto, reembolsos devem aguardar uma espera de 30 dias do momento do pedido,
usuarios lojistas, nenhum
"""

"""Historias de usuario para avaliacao


"""
```

B.3 PROMPT PARA GERAÇÃO DE DDL 0-SHOT

Nome e classificação: Gerador de ddl de banco de dados através de história de usuários. **Customização de Output, 0-Shot**

Intenção e contexto: Gerar uma ou mais instruções de DDL para definir um banco de dados que atenda as histórias de usuário informadas.

Motivação: Modelagem das entidades de banco de dados é uma das principais etapas para o desenvolvimento de um software, com um banco bem estruturado, diversas aplicações poderão consultar e persistir informações úteis além de aplicar regras de negócio.Gerar um DDL permite gerar, visualizar, alterar e replicar a estrutura de um banco de dados.

Estrutura e ideais chave: Um prompt inicial onde devem ser introduzidas apenas as histórias de usuário, e um prompt incremental que deve receber o ddl de um banco atual e o restante das histórias de usuário. O retorno do último prompt deve ser o suficiente para gerar o banco de dados, pois deve conter o DDL de todas as tabelas.

Pontos fortes: Agiliza o processo de modelagem de banco de dados, permitindo acessar padrões de criação de banco de dados utilizados como treinamento da LLM

Limitações: Não se pode confiar que o banco de dados gerados atende a todos os requisitos de usuário, ou que atenda de forma performática

Exemplo de implementação: Pode ser encontrado no código 3

Código 3 – Geração de DDL 0-Shot

```
/* Prompt Inicial */

Como um gerente de banco de dados especializado em bancos postgres utilizados por microsserviço,.leia as
seguintes histórias de usuário e seus crité rios de aceite e gere scripts DDL para a criação dessas
tabelas, constraints, relacionamentos e indexes. Lembre-se de pensar etapa por etapa para atender todas as
histórias e seus crité rios de aceite.
"""Histórias de usuário

"""

/* Prompt Incremental */

Como um gerente de banco de dados especializado em bancos postgres utilizados por microsserviços, leia as
seguintes histórias de usuário e seus crité rios de aceite e baseados no banco de dados abaixo gere
scripts DDL para a criação dessas tabelas, constraints, relacionamentos e indexes. Lembre-se de pensar
etapa por etapa para atender todas as histórias e seus crité rios de aceite. Se necessário alterar alguma
tabela existente, gere o script de criação com a tabela alterada ao invé s de gerar um script de
atualização. Sua resposta deve conter todas as tabelas, incluindo as do input, e não somente as alteradas
ou criadas como resposta a esse prompt..

"""Banco Atual

"""

"""Histórias de usuário

"""
```

B.4 PROMPT PARA GERAÇÃO DE OAS 0-SHOT

Nome e classificação: Gerador de OAS através de requisitos funcionais e não funcionais e entidades do sistema. **Customização de Output, 0-Shot**.

Intenção e contexto: Gerar uma uma definição de API no formato OAS 3.0.1

Motivação: A definição de APIs é um ponto de partida para a implementação de um software, em sistemas distribuídos que se conectam via requisições HTTP, definir schemas de API bem estruturados antes da implementação, ajuda a manter o contrato coeso, permitindo o desenvolvimento de back-ends e front-ends simultaneamente. Uma definição OAS por exemplo, pode ser usada para criar serviços de MOCK e ajudar a gerar clientes http que implementem as APIs.

Estrutura e ideais chave: Um prompt inicial onde devem ser introduzidas as entidades do sistema em forma de DDL ou outro formato e os requisitos funcionais e não funcionais relevantes para o seu comportamento, esse prompt deve gerar uma especificação de API OAS 3.0.1 completa, junto com data schemas e exemplos de resposta.

Pontos fortes: Agiliza o processo de criação de uma especificação de API a partir de linguagem natural e entidades do sistemas

Limitações: Não se pode confiar que a API irá respeitar as melhores práticas ou atender todos os requisitos do sistema.

Exemplo de implementação: Pode ser encontrado no código 4

Código 4 – Geração de OAS 0-Shot

```
Você é um modelador de APIs Restful completas, experiente na construção de OpenAPI Specifications, seu
objetivo é ler as entidades do sistema, e criar APIs que cumpram as histórias de usuários respeitando os
requisitos não funcionais. Suas APIs devem ser coesas, respeitando todos os padrões REST como por exemplo
HATEOAS, e devem possibilitar tracking com a adição de headers opcionais. A especificação gerada deve ser
feita utilizando componentes e schemas e incluir título, descrição. As rotas alé m das requisições, devem
possuir tags, sumário, descrição, tipagem, exemplo e descrição de todos os campos. As respostas devem
possui exemplos para todos os status possíveis das famílias 2xx, 4xx e 5xx. Não suponha nenhuma
funcionalidade, apenas implemente as histórias de usuário fornecidas.

Para isso você vai seguir as seguintes etapas:
1 - Entender as funcionalidades e identificar quais rotas devem ser geradas
2 - Identificar quais campos devem ser utilizados nas requisições, separando os da requisição e aqueles
que serão gerados no processamento da API, podendo criar novos dados se necessário.
3 - Escrever a especificação no padrão OAS 3.0.1 no formato yml

"""Requisitos não funcionais

"""

"""Histórias de usuário

"""

"""Entidades do Sistema

"""
```

B.5 PROMPT PARA AVALIAÇÃO DE OAS 1-SHOT

Nome e classificação: Avaliador de OAS, através de um OAS gerado, requisitos funcionais e não funcionais e entidades de banco de dados. **Identificação de Erros, 1-Shot**.

Intenção e contexto: Gerar uma feedback em cima de uma definição de API no formato OAS 3.0.1, verificando se o seu formato está correto, se atende todos os requisitos ou está cumprindo as melhores práticas.

Motivação: A definição de APIs é o contrato da API, ela define a entrada e saída de uma api, assim como a assinatura de uma função. Uma especificação OAS pode estar incorreta semanticamente, não cumprir os requisitos funcionais e não funcionais ou até mesmo não possuir as melhores práticas REST.

Estrutura e ideais chave: Um prompt de avaliação onde devem ser introduzidas as entidades do sistema em forma de DDL ou outro formato, os requisitos funcionais e não funcionais relevantes para o seu comportamento e uma definição OAS 3.0.1. Esse prompt deve gerar uma pontuação para a API que deve ir de 0 a 100. Além disso, também devem ser introduzidos os pontos de melhoria, que impactaram negativamente na pontuação, classificados como ERROR, WARNING e MINOR.

Pontos fortes: Válida se uma especificação está correta e sugere melhorias.

Limitações: Não se pode confiar que toda melhoria é válida, ou se ele realmente vai identificar falhas reais e críticas na OAS.

Exemplo de implementação: Pode ser encontrado no código 5

Código 5 – Avaliação de OAS 1-Shot

```
Voce e um avaliador de APIs Restful completas, experiente na construcao de OpenAPI Specifications, seu
objetivo e ler as entidades do sistema, as historias de usuarios e os requisitos nao funcionais e avaliar
se uma especificacao OAS 3.0.1 no formato yml esta cumprindo esses requisitos e direcionada as entidades
do sistema.

Para voce, APIs devem ser coesas, respeitando todos os padroes REST como por exemplo HATEOAS, e devem
possibilitar tracking com a adicao de headers opcionais. A especificacao sempre deve possuir o maximo de
detalhamento possivel, e deve possuir facil manutencao e reaproveitamento de trechos, como por exemplo a
utilizacao de componentes e schemas e incluir titulo, descricao. As rotas alem das requisicoes, devem
possuir tags, sumario, descricao, tipagem, exemplo e descricao de todos os campos e as respostas com os
exemplos de respostas em casos de varios status das familias 2xx, 4xx e 5xx. Na hora de avaliar nao
suponha nenhuma funcionalidade que nao esteja descrita nos requisitos funcionais.

Para isso voce vai seguir as seguintes etapas:

1 - Verificar se nao ha erros na semantica da especificacao gerada
2 - Entender as funcionalidades e identificar se todas as rotas foram geradas e possuem o comportamento
esperado.
3 - Verificar se a API possui todas as especificacoes REST, como por exemplo HATEOAS
4 - Verificar se existem headers de tracking
5 - Organizar e simplificar todos os pontos de melhoria, os classificando como ERROR (para erros graves),
WARNING (Para pontos de atencao) e MINOR para pequenas melhorias
6 - Avaliar os pontos de melhoria e definir uma pontuacao de 0 a 100, onde 0 significa uma alucinacao
completa e 100 significa a melhor definicao de API proposta para os requisitos e entidades definidos.
7 - Criar um bloco final de resposta, no exemplo informado separados por caracteres especiais como @@@
para determinar o que e saida e o que faz parte do reasoning

"""Exemplo de resposta
@@@
score: 40
ERROR - A definicao esta incorreta, \"type\": \"list\" nao existe o correto e \"type\": \"array\"
ERROR - A rota POST /users/create nao possui headers de tracking
ERROR - Nao foram criadas rotas para atualizacao de usuarios conforme criterio de aceite US-5-AC-4
WARNING - A rota POST /users esta sendo usada para retornar usuarios o method correto deveria ser GET
WARNING - A rota POST /users/create esta sendo usada para criar usuarios, o correto deveria ser POST users
MINOR - A rota de correta de GET /users poderia ser paginavel para impedir sobrecargas
MINOR - O campo \"nome\" esta em portugues diferenciado de toda especificacao em ingles
MINOR - A rota POST /users/create nao possui um exemplo de erro caso o usuario ja exista
@@@
"""

"""Requisitos nao funcionais

"""

"""Historias de usuario

"""

"""Entidades do Sistema

"""

"""Especificação

"""
```

B.6 PROMPT PARA OTIMIZAÇÃO DE OAS 0-SHOT

Nome e classificação: Otimizador de OAS através de uma especificação existente, requisitos funcionais e não funcionais, entidades do sistema e um feedback de avaliação. **Customização de Output, 0 shot**.

Intenção e contexto: Melhoria de uma definição de API no formato OAS 3.0.1

Motivação: Corrigir uma definição de API pode ser um trabalho manual quando se tem todo o feedback necessário. Uma especificação funcional pode ser o suficiente para ser processada por algum software e gerar corretamente uma interface ou ferramenta.

Estrutura e ideais chave: Um prompt onde devem ser introduzidas um contexto que pode conter as entidades do sistema em forma de DDL ou outro formato e os requisitos funcionais e não funcionais relevantes para o seu comportamento. Além disso deve ser fornecido uma OAS de base e um feedback. Esse prompt deve gerar uma nova versão da especificação de API OAS 3.0.1 completa, baseado na introduzida no prompt e com todo o feedback solicitado corrigido.

Pontos fortes: Revisa e tenta solucionar feedbacks fornecidos por uma pessoa ou LLM.

Limitações: Não se pode confiar que a otimização irá solucionar todos os problemas relatados no feedback.

Exemplo de implementação: Pode ser encontrado no código 6

Código 6 – Otimização de OAS 0-Shot

```
Voce e um modelador de APIs Restful completas, experiente na construcao e correcao de OpenAPI
Specifications. Seu objetivo e ler todo o contexto fornecido, revisar um feedback e entender seus pontos
de critica para corrigir uma especificação OAS fornecida. Você deve gerar uma nova especificacao completa
que contenha todo o feedback informado e corrigido, não apenas salientar quais locais devem ser alterados.
Nao suponha nenhuma correcao alem das informadas no feedback.

Para isso voce vai seguir as seguintes etapas:
1 - Entender as solicitacoes do feedback e vincular com o contexto e OAS informada.
2 - Baseado nessa compreensao, identificar quais correcoes devem ser feitas.
3 - Reescrever a especificacao OAS 3.0.1 completa, incluindo as correcoes

"""Contexto

"""

"""Especificacao OAS 3.0.1

"""

"""Feedback

"""
```

APÊNDICE C – HISTÓRIAS DE USUÁRIOS REVISADAS

C.1 MÓDULO 1 - USUÁRIO GERAL

US-1: Cadastro de novo usuário

- **Descrição:** Como um novo usuário, eu quero me cadastrar no marketplace para que eu possa começar a comprar e vender produtos.
- **Critérios de Aceite:**
  - US-1-AC-1: O usuário pode se cadastrar preenchendo campos obrigatórios como nome, email e senha e um segundo campo para confirmação de senha.
  - US-1-AC-2: O sistema valida a senha e verifica se o email já está cadastrado.
  - US-1-AC-3: O cadastro é concluído com sucesso e o usuário é redirecionado para uma nova página, que exibe uma mensagem dizendo que um email de confirmação foi enviado. Nessa nova página, deve-se exibir um timer, para re-enviar o email caso ele não tenha chegado.
  - US-1-AC-4: Em caso de email duplicado, devem ser exibidos ao usuário uma mensagem informando que o email já está cadastrado e uma opção para recuperar a senha.
  - US-1-AC-5: Uma senha, para ser considerada válida, deve conter, pelo menos, 8 caracteres, sendo composta por, no mínimo, um caractere de cada classe (letra minúscula, letra maiuscula, número ou símbolo especial).
  - US-1-AC-6: Em caso de senha inválida, deve ser exibida uma mensagem ao usuário, como por exemplo campo não preenchido, formato inválido ou confirmação de senha inválida.
  - US-1-AC-7: Um email para ser considerado válido deve ter caracteres antes do @, o caractere @ e um domínio. Em caso de email inválido deve informar o erro, como campo não preenchido, formato inválido.

US-2: Confirmação de email

- **Descrição:** Como um usuário cadastrado, eu quero confirmar meu endereço de email após o cadastro para que eu possa validar minha conta e começar a usá-la.
- **Critérios de Aceite:**
  - US-2-AC-1: Um email de confirmação é enviado automaticamente após o cadastro.

  - US-2-AC-2: O email contém um link de verificação que, ao ser clicado, valida a conta do usuário.
  - US-2-AC-3: O usuário só pode acessar as funcionalidades de compra e venda após confirmar o email.
  - US-2-AC-4: Deve ser possível reenviar o email de ativação através do perfil do usuário, caso a página original de confirmação de email tenha sido fechada por acidente.
  - US-2-AC-5: Novas solicitações do email de confirmação de criação de conta podem ser feitas a cada 01 (um) minuto para as primeiras 05 (cinco) tentativas, aumentando para 30 (trinta) minutos a partir da sexta tentativa, sem limitações máximas adicionais.
  - US-2-AC-6: As funcionalidades restritas devem estar disponíveis, porém ao tentar utilizá-las, o usuário deve ser movido para a página de confirmação de email informando qual funcionalidade está bloqueada até o e-mail ser confirmado.
  - US-2-AC-7: Cada link de recuperação de confirmação de email pode ser utilizado uma única vez e possui um prazo de validade de 7 dias, e é inutilizado quando um novo email for solicitado.

US-3: Recuperação de senha

- **Descrição:** Como um usuário que perdeu a senha, eu gostaria de conseguir recuperar o meu acesso através do meu email, para que eu possa criar uma nova senha.
- **Critérios de Aceite:**
  - US-3-AC-1: O usuário pode solicitar a recuperação de senha fornecendo seu email.
  - US-3-AC-2: Um link de redefinição de senha é enviado para o email do usuário.
  - US-3-AC-3: O usuário pode redefinir sua senha através do link e fazer login com a nova senha.
  - US-3-AC-4: Novas solicitações do email de recuperação de senha podem ser feitas a cada 01 (um) minuto para as primeiras 05 (cinco) tentativas, aumentando para 30 (trinta) minutos a partir da sexta tentativa, sem limitações máximas adicionais.
  - US-3-AC-5: O link de recuperação de senha só pode ser usado uma única vez e possui um prazo de validade de 7 dias. Caso um link novo seja solicitado, o anterior é inutilizado.

  - US-3-AC-6: O sistema deve exibir ao usuário uma mensagem dizendo que o email foi enviado caso o endereço informado exista na base de dados do sistema associado a uma conta, e não deve disparar o e-mail caso não exista.

US-4: Sessões simultâneas e OAuth2

- **Descrição:** Como usuário cadastrado, quero acessar a plataforma em vários dispositivos simultaneamente.
- **Critérios de Aceite:**
  - US-4-AC-1: O usuário pode logar e manter sessões ativas em múltiplos dispositivos sem interrupções.
  - US-4-AC-2: A autenticação precisa ser no padrão OAuth2. Armazenando as sessões dos usuários e seus respectivos tokens de acesso e refresh no banco de dados para que seja possível gerenciar as sessões e seus tokens.
  - US-4-AC-3: Sempre que um refresh token for utilizado, deve-se marcar nos sistema seu uso. Caso um seja usado mais de uma vez, todas as sessões do usuário devem ser encerradas.
  - US-4-AC-4: Caso um token de acesso não pertença a uma sessão ativa (Não possua registro em banco de dados), então a requisição não pode ser considerada autenticada.

US-5: Atualização de perfil

- **Descrição:** Como um usuário logado, eu quero atualizar minhas informações de perfil para que meus dados estejam sempre corretos e atualizados.
- **Critérios de Aceite:**
  - US-5-AC-1: O usuário logado pode acessar e editar seu perfil, incluindo nome, email, e endereço.
  - US-5-AC-2: O sistema salva e reflete imediatamente as atualizações. Em caso de falha ao atualizar, deve-se informar o motivo do erro, como por exemplo erros no formulário, ou erros de sistema.
  - US-5-AC-3: Ao efetuar uma alteração de email, um novo email de confirmação deve ser disparado. O usuário deve passar novamente por todo o processo de validação de email detalhado na US-2, e a alteração só deve ser efetivada quando o link de confirmação for acessado.
  - US-5-AC-4: Caso algum erro aconteça durante alguma tentativa de alteração de informação de perfil, o sistema deve exibir uma mensagem na tela informando qual é o problema.

US-6: Logout seguro

- **Descrição:** Como um usuário logado, eu quero fazer logout da aplicação para que eu possa garantir a segurança da minha conta.
- **Critérios de Aceite:**
  - US-6-AC-1: O usuário pode fazer logout de forma segura, encerrando a sessão atual e removendo os dados dela do sistema.
  - US-6-AC-2: Após o logout, o sistema impede o acesso a áreas restritas até que o usuário faça login novamente. Se o usuário tentar acessar qualquer recurso e sua sessão estiver inativa, ele deve ser movido para a página de login, voltando para a página atual em caso de login com sucesso.

US-7: Notificação de logins

- **Descrição:** Como usuário validado, eu quero receber notificações quando houver um login na minha conta para que eu possa ser alertado sobre acessos não autorizados.
- **Critérios de Aceite:**
  - US-7-AC-1: O usuário recebe notificações via email ou push quando um login é feito em sua conta.
  - US-7-AC-2: As notificações incluem informações sobre o dispositivo e a localização aproximada do login.
  - US-7-AC-3: O email de aviso deve conter um link para bloquear acessos não identificados pelo usuário, finalizando todas as sessões ativas naquele momento (Incluindo seus tokens de acesso e refresh).

US-8: Gerenciamento de sessões ativas

- **Descrição:** Como usuário logado, eu quero visualizar e gerenciar minhas sessões ativas para que eu possa encerrar sessões em dispositivos que não estou mais usando.
- **Critérios de Aceite:**
  - US-8-AC-1: O usuário pode visualizar todas as sessões ativas, incluindo dispositivo, localização aproximada, endereço IP e última vez que conectou (data e hora da última vez que o token de acesso foi gerado).
  - US-8-AC-2: O usuário pode encerrar manualmente sessões indesejadas em dispositivos específicos.
  - US-8-AC-3: O usuário deve ter a opção encerrar todas as sessões conectadas.

US-9: Visualização de avaliações

- **Descrição:** Como um visitante do site, eu quero ler avaliações de outros usuários para que eu possa tomar decisões de compra mais informadas e obter detalhes sobre produtos.
- **Critérios de Aceite:**
  - US-9-AC-1: O visitante pode acessar a página do produto e visualizar avaliações deixadas por outros compradores.
  - US-9-AC-2: As avaliações são exibidas de forma organizada, com notas e comentários.
  - US-9-AC-3: As avaliações devem ser ordenadas por data e hora de forma decrescente (mais novas primeiro).

US-10: Suporte ao cliente

- **Descrição:** Como usuário logado e validado, eu quero acessar o suporte ao cliente facilmente para que eu possa resolver qualquer problema que encontrar.
- **Critérios de Aceite:**
  - US-10-AC-1: O usuário pode acessar o suporte através de um botão ou link visível na interface.
  - US-10-AC-2: O sistema redireciona para um formulário de contato, onde o usuário deverá acrescentar o tipo do problema encontrado e uma descrição detalhada do que aconteceu. Caso seja um problema relacionado a um pedido, o usuário deverá acrescentar o código do pedido. Estes dados serão utilizados pelo suporte para solucionar o problema.
  - US-10-AC-3: Após o envio do formulário um ticket deve ser aberto, com as informações citadas e poderá ser visto por usuários administradores.
  - US-10-AC-4: Após a abertura do ticket o usuário deve receber um email informando que o ticket foi aberto, onde receberá atualizações sobre o andamento e será por onde um administrador poderá entrar em contato.

US-11: Exclusão de conta

- **Descrição:** Como usuário logado, eu quero excluir minha conta permanentemente para que meus dados sejam removidos do sistema.
- **Critérios de Aceite:**
  - US-11-AC-1: O usuário pode solicitar a exclusão da conta por meio de uma opção na área de perfil.

  - US-11-AC-2: Após a exclusão, os dados da conta do usuário são excluídos imediatamente, salvo a exceção dos casos de notas fiscais ou transações (compras e vendas) que são definidas como obrigações fiscais, que devem ser mantidas por um prazo de 5 anos.
  - US-11-AC-3: Se houver assuntos pendentes do usuário, como compras que não foram finalizadas ou que estão em disputa de reembolso ou troca, no caso de usuários vendedores como vendas em processamento que não foram finalizadas a pelo menos 3 meses, a conta não é encerrada. O usuário é avisado na hora sobre as pendências em andamento e recebe por email os detalhes delas.
  - US-11-AC-4: Após o fim dos trâmites o usuário é perguntado novamente por email se deseja realmente excluir a conta, ou pode manualmente solicitar novamente a exclusão.

C.2 MÓDULO 2 - USUÁRIO ADMINISTRADOR

US-12: Gerenciamento de permissões de acesso

- **Descrição:** Como administrador, eu quero gerenciar detalhadamente as permissões de acesso dos usuários, incluindo papeis e níveis de acesso específicos, para que eu possa garantir que cada usuário tenha acesso somente às funcionalidades necessárias para sua função, mantendo a segurança do sistema.
- **Definições adicionais:**
  - Níveis de acesso: Unidade mínima de autorização, determina qual operação pode ser realizada, como por exemplo visualizar produtos ou alterar produtos. Ex: PRODUCT:READ PRODUCT:UPDATE.
  - Papeis: Um papel é a abstração de um nível de permissionamento ou cargo no sistema. Um papel pode ter um ou mais níveis de acesso.
- **Critérios de Aceite:**
  - US-12-AC-1: O administrador pode criar, editar e remover permissões de acesso para diferentes tipos de usuários (papéis). Além disso, podem definir novos papeis no sistema.
  - US-12-AC-2: As permissões podem ser configuradas por nível de acesso e papeis (administrador, vendedor, comprador).
  - US-12-AC-3: O administrador tem acesso irrestrito a todos os logs de alterações de permissões.

US-13: Gerenciamento de administradores

- **Descrição:** Como administrador, eu quero poder cadastrar, editar e remover outros administradores, com diferentes níveis de privilégio, para que o sistema possa ser gerenciado de forma segura e eficiente por múltiplos responsáveis.
- **Definições adicionais:**
  - Nível de privilégio: Nível de privilégio é uma definição adicional para usuários administradores, definindo uma hierarquia entre eles e, portanto, quais possuem permissões para alterar outros administradores. Por exemplo, um administrador de nível de privilégio 1 é o root e pode cadastrar administradores de nível de privilégio abaixo de 1, além de poder alterar suas permissões e níveis de privilégio. Administradores só podem alterar outros administrados que possuam níveis de privilégio abaixo dos seus, e sempre que cadastrarem novos administradores na plataforma, esses deverão possuir um nível de privilégio menor.
- **Critérios de Aceite:**
  - US-13-AC-1: O administrador pode adicionar novos administradores, editar suas permissões e removê-los.
  - US-13-AC-2: Diferentes níveis de privilégio são atribuídos aos administradores conforme definição de níveis de privilégio.

US-14: Moderação de anúncios

- **Descrição:** Como administrador, eu quero bloquear e desbloquear anúncios que não estejam em conformidade com as políticas da plataforma, para manter a qualidade e segurança dos produtos oferecidos no marketplace.
- **Critérios de Aceite:**
  - US-14-AC-1: O administrador pode bloquear ou desbloquear anúncios que violem as políticas da plataforma.
  - US-14-AC-2: Um motivo deve ser fornecido ao bloquear um anúncio, e o vendedor é notificado através de uma mensagem contendo um link para o anúncio, o motivo do bloqueio e um link para contestação (abertura de ticket de suporte).
  - US-14-AC-3: Em caso de aprovação de contestação (realizado pelo administrador), o anúncio é desbloqueado e retornado à normalidade. O vendedor receberá um email notificando do desbloqueio.

US-15: Moderação de usuários

- **Descrição:** Como administrador, eu quero bloquear e desbloquear diferentes tipos de usuários (vendedores, compradores, lojistas), para que eu possa moderar comportamentos inadequados e garantir a segurança e confiabilidade da plataforma.
- **Critérios de Aceite:**
  - US-15-AC-1: O administrador pode bloquear ou desbloquear contas de usuários que violam as regras da plataforma.
  - US-15-AC-2: O usuário bloqueado é notificado via email sobre a ação tomada. O usuário poderá acessar a conta e realizar uma contestação (Abertura de ticket) ou poderá solicitar a exclusão da sua conta.
  - US-15-AC-3: Quando um comprador é bloqueado, ele não pode realizar nenhuma compra ou comentar anúncios.
  - US-15-AC-4: Quando um vendedor é bloqueado, todos os anúncios são pausados, e ele não pode responder nenhuma avaliação. Porém poderá interagir com vendas em andamento.
  - US-15-AC-5: Anúncios pausados são visíveis apenas para o vendedor que é dono dos mesmos.
  - US-15-AC-6: Quando um vendedor é desbloqueado, todos os anúncios são retornados à normalidade.
  - US-15-AC-7: Não há limites para a quantidade de vezes que um usuário poderá ser bloqueado, mas os administradores poderão ver o histórico de bloqueio e desbloqueio.

US-16: Moderação de perguntas e respostas

- **Descrição:** Como administrador, eu quero revisar, editar ou remover perguntas e respostas que sejam inapropriadas, falsas ou que violem as regras da plataforma, para manter um ambiente de compra e venda transparente e confiável.
- **Critérios de Aceite:**
  - US-16-AC-1: O administrador pode revisar e remover perguntas ou respostas que sejam inadequadas. O autor da pergunta e o dono do anúncio são notificados através de um email da remoção da pergunta e do motivo da remoção.
  - US-16-AC-2: Perguntas e respostas removidas desaparecem imediatamente da página do produto e do sistema.

US-17: Gerenciamento de pedidos de devolução

- **Descrição:** Como administrador, eu quero autorizar ou rejeitar pedidos de devolução com base em critérios estabelecidos, como conformidade com a política de devoluções e análise do estado do produto, para garantir uma experiência justa tanto para compradores quanto para vendedores.
- **Critérios de Aceite:**
  - US-17-AC-1: O administrador pode revisar pedidos de devolução com base nas políticas de devolução.
  - US-17-AC-2: O administrador pode aceitar ou rejeitar o pedido, e o comprador e vendedor são notificados da decisão.

US-18: Visualização de suporte ao cliente

- **Descrição:** Como administrador, eu quero acessar o suporte ao cliente e visualizar os tickets em andamento, incluindo o progresso e a responsabilidade de cada ticket, para acompanhar e garantir a resolução eficiente dos problemas relatados pelos usuários.
- **Critérios de Aceite:**
  - US-18-AC-1: O administrador pode visualizar e gerenciar tickets de suporte em andamento. Vendo quem foi o usuário que abriu o ticket, o título, descrição e motivo de abertura, assim como qual usuário está responsável pelo ticket, seu progresso atual e meios de contato.
  - US-18-AC-2: O progresso, isto é, o status do ticket (aberto, aguardando validação, ou fechado) e a responsabilidade (usuário administrador de quem está sendo aguardada uma ação) de cada ticket são atualizados conforme necessário.
  - US-18-AC-3: O administrador pode definir a responsabilidade do ticket para determinar que outro administrador deve lidar com o mesmo, ou alterar para qualquer status determinando se o ticket precisa de ação do administrador, ação do usuário que o abriu, ou está finalizado.
  - US-18-AC-4: Um administrador pode incluir notas (mensagens) ao histórico do ticket, para que seja claro quais ações foram feitas, quem foram os atores, e o andamento do mesmo.
  - US-18-AC-5: Quando o ticket é fechado, o usuário é notificado por email com a descrição final da resolução.

US-19: Acompanhamento de desempenho de lojas

- **Descrição:** Como administrador, eu desejo visualizar estatísticas detalhadas de uma loja, como faturamento total, comissão repassada à plataforma, número de

vendas e crescimento ao longo do tempo, para monitorar o desempenho financeiro da plataforma e dos lojistas.

- **Critérios de Aceite:**
  - US-19-AC-1: O administrador pode acessar estatísticas detalhadas sobre o desempenho de lojas e vendedores.
  - US-19-AC-2: Os dados incluem o valor do faturamento, apresentando a porcentagem desse faturamento que foi repassado à plataforma e o valor de repasse, o número de vendas e crescimento ao longo do tempo (Variância do faturamento).
  - US-19-AC-3: Deve ser possível passar um intervalo de tempo para filtrar as estatísticas, escolhendo uma data de início, data de fim.
  - US-19-AC-4: Deve ser possível escolher um produto para verificar o faturamento de produtos específicos, ou suas categorias.

C.3 MÓDULO 3 - USUÁRIO COMPRADOR

US-20: Pesquisa de produtos

- **Descrição:** Como um comprador, eu quero pesquisar produtos por categoria e preço para que eu possa encontrar o que estou procurando facilmente.
- **Critérios de Aceite:**
  - US-20-AC-1: O comprador pode pesquisar produtos usando filtros de categoria e faixa de preço. Quando múltiplos filtros são usados em conjunto, a busca deve ser refinada para que somente seja retornado produtos que atendam todos os filtros.
  - US-20-AC-2: Os resultados da pesquisa são exibidos corretamente conforme os critérios aplicados e podem ser ordenados por valor do mais caro por mais barato e vice-versa, além de mais vendidos ou a ordenação padrão.
  - US-20-AC-3: Caso nenhum produto seja encontrado, deve-se retornar uma lista vazia para que seja possível informar que nenhum produto foi encontrado.
  - US-20-AC-4: Filtros inválidos devem ser descartados e não devem impactar a busca.
  - US-20-AC-5: Filtros inválidos devem ser descartados e não devem impactar a busca.
  - US-20-AC-6: A busca por produtos deve ser paginada, podendo escolher a quantidade de itens, com um máximo de 50 itens por busca.

US-21: Adicionar produtos ao carrinho

- **Descrição:** Como um comprador, eu quero adicionar produtos ao meu carrinho de compras para que eu possa revisá-los antes de finalizar a compra.
- **Critérios de Aceite:**
  - US-21-AC-1: O comprador pode adicionar produtos ao carrinho, e os produtos aparecem na lista de revisão antes da compra.
  - US-21-AC-2: O carrinho armazena os produtos até que a compra seja finalizada ou o carrinho seja esvaziado.
  - US-21-AC-3: O carrinho é armazenado no dispositivo com a sessão ativa, usando cache local. Caso o usuário deslogue o carinho deve ser apagado do cache.
  - US-21-AC-4: Se o comprador fechar o site e abrir novamente, as informações do carrinho devem ser recuperadas.

US-22: Gerenciamento de endereços

- **Descrição:** Como um comprador, quero salvar meus endereços de entrega para facilitar futuras compras.
- **Critérios de Aceite:**
  - US-22-AC-1: O comprador pode salvar múltiplos endereços de entrega para futuras compras.
  - US-22-AC-2: Os endereços salvos podem ser editados ou excluídos a qualquer momento, incluindo antes de fechar um pedido (checkout).

US-23: Perguntar ao vendedor

- **Descrição:** Como um comprador, eu quero fazer perguntas ao vendedor sobre um produto antes de comprar, para que eu possa esclarecer dúvidas antes de finalizar a compra.
- **Critérios de Aceite:**
  - US-23-AC-1: O comprador pode enviar perguntas diretamente ao vendedor por meio da página do produto.
  - US-23-AC-2: O sistema notifica o comprador quando o vendedor responde. Através de notificações push e email.

US-24: Pagamento seguro

- **Descrição:** Como um comprador, eu quero um processo de pagamento seguro e fácil para que eu possa concluir minhas compras com confiança.
- **Critérios de Aceite:**
  - US-24-AC-1: O comprador pode concluir o pagamento usando Stripe, e os dados de pagamento são processados de forma segura.
  - US-24-AC-2: No Stripe devem ser habilitados as cobranças de Pix, Boleto e cartão de Crédito.
  - US-24-AC-3: O sistema exibe uma confirmação de compra após o pagamento bem-sucedido.
  - US-24-AC-4: Caso o pagamento não seja bem sucedido o usuário deve ser notificado via email e a compra deve ser cancelada.
  - US-24-AC-5: O processo de pagamento deve ser Idempotente, para garantir que o usuário seja tarifado apenas uma vez, e que o callback de pagamento seja válido uma vez, e não altere o status do pagamento. Caso não receba callback de pagamento, a compra deve ser considerada cancelada por erro no pagamento o usuário deve ser notificado via email.
  - US-24-AC-6: A partir do pagamento, o comprador terá um prazo final de entrega de 60 dias para receber o produto, e portanto o pagamento deve ser retido e somente liberado após o prazo final de entrega (sem pedidos válidos de reembolso), ou a confirmação do recebimento do produto pelo usuário.
  - US-24-AC-7: O prazo final de entrega não depende do tempo estimado de entrega, nem de divergências nos prazos do vendedor ou transportadora.

US-25: Cancelamento de pedidos

- **Descrição:** Como um comprador, eu quero cancelar meu pedido enquanto ele estiver sendo processado, para que eu possa mudar de ideia antes do envio.
- **Critérios de Aceite:**
  - US-25-AC-1: O comprador pode cancelar o pedido enquanto ele ainda estiver sendo processado.
  - US-25-AC-2: O sistema notifica o comprador sobre o cancelamento e reembolsa automaticamente.

US-26: Notificações sobre o pedido

- **Descrição:** Como um comprador, eu quero ser notificado em todas as etapas do meu pedido, para que eu possa saber quando o pagamento foi aprovado, quando foi separado, quando o pedido foi enviado e problemas durante esse processo.

- **Critérios de Aceite:**
  - US-26-AC-1: O comprador recebe notificações em tempo real sobre o status do pedido (aprovado, enviado, etc.) via email ou push de notificação.
  - US-26-AC-2: O comprador pode visualizar o histórico completo do pedido na sua conta.

US-27: Avaliação de produtos

- **Descrição:** Como um comprador, eu quero avaliar o produto após receber meu produto, para que eu possa compartilhar minha experiência e ajudar outros compradores.
- **Critérios de Aceite:**
  - US-27-AC-1: O comprador pode avaliar o produto e deixar um comentário sobre a sua experiência.
  - US-27-AC-2: A avaliação aparece publicamente na página do produto após a moderação.

US-28: Histórico de pedidos

- **Descrição:** Como um comprador, eu quero um histórico de todos os pedidos que fiz, para que eu possa acompanhar minhas compras e revisar informações sobre produtos adquiridos anteriormente.
- **Critérios de Aceite:**
  - US-28-AC-1: O comprador pode visualizar um histórico detalhado de todas as suas compras anteriores.
  - US-28-AC-2: O histórico inclui informações como data da compra, nome do produto, categoria, descrição original do produto no momento da compra, valor pago e status do pedido.
  - US-28-AC-3: O histórico deve ser disponibilizado de forma paginada, permitindo que o usuário consiga ir vendo páginas de compras realizadas.

US-29: Devolução de produtos

- **Descrição:** Como um comprador, eu quero devolver um produto por problemas técnicos ou exercer meu direito de arrependimento, para que eu possa receber o reembolso após a confirmação da devolução do item pelo vendedor ou comprovação de fraude.
- **Critérios de Aceite:**

  - US-29-AC-1: O comprador pode solicitar a devolução de um produto através da interface de pedidos.
  - US-29-AC-2: O sistema processa a devolução conforme a política de devolução, e o comprador recebe o reembolso após confirmação.

US-30: Lista de favoritos

- **Descrição:** Como comprador, quero adicionar produtos à minha lista de favoritos, para lembrar de itens que eu possa querer comprar no futuro.
- **Critérios de Aceite:**
  - US-30-AC-1: O comprador pode adicionar produtos a uma lista de favoritos, acessível a qualquer momento.
  - US-30-AC-2: Adicionar um produto a lista deve ser idempotente. Ou seja, somente deve se adicionar um produto caso ele não esteja presente na lista, para o usuário deve-se indicar em ambos os casos (presente e não presente) que o produto foi adicionado à lista de favoritos.
  - US-30-AC-3: Antes de adicionar um produto à lista de favoritos, deve ser possível verificar se o produto já está presente, trocando a o indicativo da ação de adicionar para remover da lista.
  - US-30-AC-4: A lista pode ser editada para remover ou adicionar produtos.

US-31: Denúncia de conteúdo

- **Descrição:** Como comprador eu quero poder relatar conteúdos inapropriados ou fraudulentos na plataforma.
- **Critérios de Aceite:**
  - US-31-AC-1: O comprador pode relatar conteúdos inadequados através de um botão dedicado.
  - US-31-AC-2: O sistema processa a denúncia e o conteúdo é revisado por um administrador.

C.4 MÓDULO 4 - USUÁRIO VENDEDOR

US-32: Gerenciamento de inventário de produtos

- **Descrição:** Como um vendedor, eu quero gerenciar meu inventário de produtos para que eu possa manter minhas ofertas atualizadas.
- **Critérios de Aceite:**

  - US-32-AC-1: O vendedor pode adicionar, editar e remover produtos de seu inventário.
  - US-32-AC-2: O estoque é atualizado automaticamente com base nas vendas de forma transacional, para impedir inconsistências. Ou seja, não é possível alterar o estoque durante uma venda se ela não foi completada.
  - US-32-AC-3: Caso o estoque atinja zero, não deve ser possível realizar mais compras, informando que o produto está temporariamente indisponível por falta de estoque.
  - US-32-AC-4: O estoque é atualizado automaticamente com base nas vendas.
  - US-32-AC-5: O usuário vendedor possui um número máximo de 50 produtos para vender na plataforma, esse valor deve ser parametrizado e alterado por um administrador.

US-33: Edição de descrições detalhadas dos produtos

- **Descrição:** Como um vendedor, eu quero criar e editar descrições detalhadas dos meus produtos para que os compradores possam entender melhor o que estão comprando.
- **Critérios de Aceite:**
  - US-33-AC-1: O vendedor pode criar ou editar descrições de produtos de forma fácil.
  - US-33-AC-2: A descrição atualizada não deve impactar um produto que já foi vendido, ou seja, devem ser armazenados os dados do produto no histórico de compra do usuário comprador, para que o mesmo possa acessar os detalhes do produto original no momento da compra.
  - US-33-AC-3: As descrições são atualizadas em tempo real na página do produto.

US-34: Upload e otimização de fotos dos produtos

- **Descrição:** Como vendedor, eu quero adicionar fotos aos meus produtos para que eles se destaquem visualmente no marketplace.
- **Critérios de Aceite:**
  - US-34-AC-1: O vendedor pode fazer upload de fotos de alta qualidade para seus produtos.
  - US-34-AC-2: O sistema redimensiona as imagens conforme necessário para garantir um carregamento rápido.

  - US-34-AC-3: Deve ser possível armazenar as imagens em vários tamanhos diferentes para que ela seja otimizada dependendo da tela do usuário.
  - US-34-AC-4: O sistema possui um tamanho máximo de 10mb para a imagem ser processada e redimensionada diminuindo seu tamanho.
  - US-34-AC-5: Deve ser possível adicionar até 10 fotos por produto.

US-35: Visualização de estatísticas de vendas

- **Descrição:** Como vendedor, eu quero visualizar estatísticas de vendas de cada produto (como número de visualizações e conversões) para que eu possa ajustar minhas estratégias de marketing e preços.
- **Critérios de Aceite:**
  - US-35-AC-1: O vendedor pode visualizar o número de visualizações de cada produto na página de gerenciamento de inventário.
  - US-35-AC-2: O vendedor pode ver a taxa de conversão (Quantidade de compras dividido pela quantidade de visualizações de abertura da página de um produto sempre que o usuário abrir um produto para obter suas informações).
  - US-35-AC-3: As estatísticas exibem dados em gráficos e/ou tabelas que mostram informações ao longo do tempo.
  - US-35-AC-4: O vendedor pode filtrar as estatísticas por intervalo de tempo, escolhendo uma data de início e uma data de fim.

US-36: Histórico de interações com clientes

- **Descrição:** Como um vendedor, eu quero ver o histórico de interações com os clientes, como perguntas e respostas, para que eu possa responder rapidamente a novas perguntas e acompanhar o feedback.
- **Critérios de Aceite:**
  - US-36-AC-1: O vendedor pode visualizar uma lista de perguntas recebidas para cada produto em uma página dedicada ou na área de gerenciamento de produtos.
  - US-36-AC-2: O vendedor pode acessar o histórico completo de perguntas e respostas de clientes para cada produto.
  - US-36-AC-3: Novas perguntas são exibidas de forma destacada separadamente das outras, para facilitar a visualização do vendedor.
  - US-36-AC-4: O vendedor é notificado quando há novas perguntas não respondidas, através de um email mostrando a pergunta e o anúncio no qual ela foi feita, juntamente com um link para a resposta.

  - US-36-AC-5: O sistema mantém o histórico de interações ordenado cronologicamente e acessível.

US-37: Convite de usuários para a loja

- **Descrição:** Como um vendedor que gerencia uma loja, eu quero convidar outros usuários a criarem contas em minha loja.
- **Critérios de Aceite:**
  - US-37-AC-1: O vendedor pode enviar convites via email ou link gerado para que outros usuários criem contas vinculadas à sua loja. Ele deve determinar um email que será utilizado para fazer o cadastro.
  - US-37-AC-2: O vendedor pode definir as permissões e o papel (gerente, assistente, etc.) para cada novo usuário convidado.
  - US-37-AC-3: O usuário convidado recebe um link de inscrição e, após a criação da conta, é adicionado automaticamente à equipe da loja. É possível enviar um novo link de inscrição, a cada 10 minutos, a menos que o usuário se cadastre usando um link existente.
  - US-37-AC-4: Após clicar no link, o usuário passará pelo mesmo procedimento de criação de conta descrito no US-2.
  - US-37-AC-5: O link para inscrição pode ser utilizado somente uma vez e expira após uma semana. Além disso, ao gerar um link novo, o passado deixa de funcionar para o email em questão.
  - US-37-AC-6: O vendedor dono pode visualizar e gerenciar todos os usuários vinculados à sua loja, incluindo alterar permissões ou remover usuários.
  - US-37-AC-7: Os usuários cadastrados como vendedores da loja podem responder perguntas, cadastrar, editar e remover produtos e atualizar status de pedidos, como notificar a plataforma do envio.
  - US-37-AC-8: Há um limite de 10 usuários cadastrados em uma loja.

US-38: Gestão financeira e saldo do vendedor

- **Descrição:** Como um vendedor, eu quero gerenciar meu saldo e ver os pagamentos recebidos, incluindo taxas e deduções da plataforma, para ter controle financeiro total das minhas vendas.
- **Critérios de Aceite:**
  - US-38-AC-1: O vendedor pode visualizar seu saldo disponível e ver os pagamentos recebidos em uma interface clara.

  - US-38-AC-2: O vendedor pode ver o histórico de transações, incluindo o valor bruto das vendas, taxas de transação, e deduções aplicadas pela plataforma. Além disso, ele deve poder exportar esses dados em formato de planilha digital ou pdf.
  - US-38-AC-3: O vendedor pode gerar relatórios financeiros detalhados de períodos específicos, tendo a possibilidade de escolher vendas, reembolsos, cancelamentos e todo tipo de atividade feita em sua conta, optando por exibir ou não as taxas e deduções da plataforma.
  - US-38-AC-4: O sistema exibe as taxas e deduções aplicadas de forma transparente, destacando quanto foi aplicado sobre cada venda feita.
  - US-38-AC-5: O vendedor pode solicitar saques de seu saldo para uma conta bancária vinculada.

US-39: Suspensão ou remoção temporária de produtos

- **Descrição:** Como um vendedor, eu quero poder suspender ou remover produtos temporariamente do marketplace quando houver necessidade de ajustes ou indisponibilidade de estoque.
- **Critérios de Aceite:**
  - US-39-AC-1: O vendedor pode suspender temporariamente a exibição de um produto no marketplace sem removê-lo permanentemente.
  - US-39-AC-2: Produtos suspensos permanecem visíveis no painel de gerenciamento de produtos com um status que indica que estão inativos.
  - US-39-AC-3: O vendedor pode reativar ou remover permanentemente um produto com um simples clique.
  - US-39-AC-4: Produtos que possuem vendas pendentes, isto é, com status diferente de "finalizado", não podem ser removidos, apenas suspensos.

US-40: Responder avaliações de clientes

- **Descrição:** Como um vendedor, eu quero responder diretamente às avaliações dos clientes sobre meus produtos para melhorar minha reputação e resolver qualquer questão.
- **Critérios de Aceite:**
  - US-40-AC-1: O vendedor pode visualizar todas as avaliações deixadas por clientes em seus produtos em uma página de gerenciamento de avaliações.

  - US-40-AC-2: O vendedor pode responder a cada avaliação diretamente pela interface da plataforma, através de uma caixa de texto com limite de 500 caracteres sem nenhum formato específico pré-estabelecido.
  - US-40-AC-3: As respostas do vendedor são exibidas junto com a avaliação do cliente, visíveis para todos os usuários.
  - US-40-AC-4: O sistema notifica o vendedor sempre que uma nova avaliação é feita em um de seus produtos.
  - US-40-AC-5: O vendedor pode editar ou excluir suas respostas, desde que dentro de um período definido.

US-41: Notificações de venda

- **Descrição:** Como vendedor, eu quero receber notificações quando um produto for vendido para que eu possa preparar o envio rapidamente.
- **Critérios de Aceite:**
  - US-41-AC-1: O vendedor recebe notificações em tempo real via email ou push sempre que um produto é vendido.
  - US-41-AC-2: As notificações contêm detalhes da venda, incluindo o nome do comprador, endereço de envio, e informações do produto vendido.
  - US-41-AC-3: O sistema permite que o vendedor veja um histórico de todas as notificações de vendas recebidas.
  - US-41-AC-4: O vendedor pode configurar preferências para o tipo de notificações (email, SMS, push) e a frequência delas.

US-42: Atualização de status de pedidos

- **Descrição:** Como um vendedor, eu quero atualizar o status do pedido, para manter meu cliente informado.
- **Critérios de Aceite:**
  - US-42-AC-1: O vendedor pode alterar o status do pedido diretamente na plataforma (ex: "Processando", "Enviado", "Entregue", "Cancelado").
  - US-42-AC-2: Cada mudança de status é registrada e visível para o vendedor e o comprador.
  - US-42-AC-3: O sistema notifica automaticamente o comprador por email ou push sempre que o status do pedido é atualizado.
  - US-42-AC-4: O vendedor pode visualizar o histórico completo de status de um pedido em uma página de detalhes do pedido.

- US-42-AC-5: O vendedor pode adicionar comentários ou notas ao atualizar o status do pedido, que são visíveis para o cliente.
- US-42-AC-6: A interface de atualização de status é de fácil acesso no painel de controle do vendedor, com uma lista de todos os pedidos que oferece a opção de filtragem por pendentes e concluídos e de ordenação por data crescente ou decrescente.
- US-42-AC-7: O vendedor é impedido de alterar o status de pedidos que já tenham sido finalizados (ex: "Entregue"), pois o sistema não mostrará nenhuma opção correspondente.
- US-42-AC-8: Um pedido é considerado "finalizado"caso:
  - O usuário cancelou o pedido antes dele ser enviado.
  - A compra inicial seja entregue corretamente e o comprador assinale que não teve problemas.
  - O produto chegou no prazo final de entrega sem ter sido recebido e houve disputa de reembolso validada por um administrador.
  - O produto chegou no prazo final de entrega, sem disputa (usuário não confirmou o recebimento).
  - O comprador recebeu o produto, teve problemas, solicitou o reembolso e o mesmo já foi recebido na conta do cliente.
  - O comprador recebeu o produto, solicitou troca, já recebeu o novo produto e sinalizou que está tudo em ordem.

APÊNDICE D – MODELO MÍNIMO ACEITÁVEL DO BANCO DE DADOS (ORÁCULO)

D.1 DDL

Código 7 – DDL Criação de Categoria de Produto

```
CREATE TABLE product_category (
    id           BIGINT      NOT NULL PRIMARY KEY,
    name         VARCHAR     NOT NULL,
    created_date TIMESTAMP   NOT NULL,
    updated_date TIMESTAMP
);
```

Código 8 – DDL Criação de Usuário

```
CREATE TABLE "user" (
    id              BIGINT      NOT NULL PRIMARY KEY,
    name            VARCHAR     NOT NULL,
    email           VARCHAR     NOT NULL UNIQUE,
    activated       BOOLEAN,
    password        VARCHAR     NOT NULL,
    privilege_level BIGINT,
    blocked         BOOLEAN,
    created_date    TIMESTAMP   NOT NULL,
    updated_date    TIMESTAMP
);
```

Código 9 – DDL Criação de Loja

```
CREATE TABLE store (
    id           BIGINT      NOT NULL PRIMARY KEY,
    owner_id     BIGINT      REFERENCES "user",
    name         VARCHAR     NOT NULL,
    balance      BIGINT,
    suspended    BOOLEAN,
    created_date TIMESTAMP   NOT NULL,
    updated_date TIMESTAMP
);
```

Código 10 – DDL Criação de Usuário-Loja

```
CREATE TABLE user_store (
    user_id      BIGINT      NOT NULL UNIQUE,
    store_id     BIGINT      NOT NULL,
    PRIMARY KEY (user_id, store_id),
    FOREIGN KEY (user_id) REFERENCES "user"(id) ON DELETE CASCADE,
    FOREIGN KEY (store_id) REFERENCES store(id) ON DELETE CASCADE
);
```

Código 11 – DDL Criação de Produto

```
CREATE TABLE product (
    id           BIGINT      NOT NULL PRIMARY KEY,
    store_id     BIGINT      REFERENCES store,
    user_id      BIGINT      REFERENCES "user",
    category_id  BIGINT      REFERENCES product_category,
    title        VARCHAR     NOT NULL,
    description  VARCHAR,
    price        BIGINT,
    suspended    BOOLEAN,
    active       BOOLEAN,
    stock        BIGINT,
    created_date TIMESTAMP   NOT NULL,
    updated_date TIMESTAMP
);
```

Código 12 – DDL Criação de Característica de Produto

```
CREATE TABLE product_feature (
    id           BIGINT      NOT NULL PRIMARY KEY,
    product_id   BIGINT      REFERENCES product,
    key          VARCHAR     NOT NULL,
    value        VARCHAR     NOT NULL,
    created_date TIMESTAMP   NOT NULL,
    updated_date TIMESTAMP
);
```

Código 13 – DDL Criação de Token de Recuperação

```
CREATE TABLE recover_token (
    id           BIGINT      NOT NULL PRIMARY KEY,
    code         VARCHAR     NOT NULL,
    type         VARCHAR     NOT NULL,
    user_id      BIGINT      REFERENCES "user",
    expiration   TIMESTAMP   NOT NULL,
    created_date TIMESTAMP   NOT NULL,
    updated_date TIMESTAMP
);
```

Código 14 – DDL Criação de Sessão

```
CREATE TABLE session (
    id           BIGINT      NOT NULL PRIMARY KEY,
    user_id      BIGINT      REFERENCES "user",
    code         VARCHAR     NOT NULL,
    ip           VARCHAR,
    device       VARCHAR,
    location     VARCHAR,
    expiration   TIMESTAMP   NOT NULL,
    created_date TIMESTAMP   NOT NULL,
    updated_date TIMESTAMP
);
```

Código 15 – DDL Criação de Token

```
CREATE TABLE token (
    id           BIGINT      NOT NULL PRIMARY KEY,
    session_id   BIGINT      REFERENCES session,
    token        VARCHAR     NOT NULL UNIQUE,
    type         VARCHAR     NOT NULL,
    valid        BOOLEAN,
    expiration   TIMESTAMP   NOT NULL,
    created_date TIMESTAMP   NOT NULL,
    updated_date TIMESTAMP
);
```

Código 16 – DDL Criação de Permissão

```
CREATE TABLE permission (
    id           BIGINT      NOT NULL PRIMARY KEY,
    scope        VARCHAR     NOT NULL UNIQUE,
    description  VARCHAR,
    created_date TIMESTAMP   NOT NULL,
    updated_date TIMESTAMP
);
```

Código 17 – DDL Criação de Função

```
CREATE TABLE role (
    id           BIGINT      NOT NULL PRIMARY KEY,
    name         VARCHAR     NOT NULL,
    description  VARCHAR,
    created_date TIMESTAMP   NOT NULL,
    updated_date TIMESTAMP
);
```

Código 18 – DDL Criação de Usuário-Função

```
CREATE TABLE user_role (
    user_id         BIGINT      NOT NULL,
    role_id         BIGINT      NOT NULL,
    assigned_date   TIMESTAMP   DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (user_id, role_id),
    FOREIGN KEY (user_id) REFERENCES "user" (id) ON DELETE CASCADE,
    FOREIGN KEY (role_id) REFERENCES role (id) ON DELETE CASCADE
);
```

Código 19 – DDL Criação de Função-Permissão

```
CREATE TABLE role_permission (
    role_id         BIGINT      NOT NULL,
    permission_id   BIGINT      NOT NULL,
    granted_date    TIMESTAMP   DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (role_id, permission_id),
    FOREIGN KEY (role_id) REFERENCES role (id) ON DELETE CASCADE,
    FOREIGN KEY (permission_id) REFERENCES permission (id) ON DELETE
       CASCADE
);
```

Código 20 – DDL Criação de Endereço

```
CREATE TABLE address (
    id           BIGINT      NOT NULL PRIMARY KEY,
    user_id      BIGINT      REFERENCES "user",
    alias        VARCHAR,
    target_name  VARCHAR,
    address      VARCHAR,
    city         VARCHAR,
    state        VARCHAR,
    country      VARCHAR,
    postal_code  VARCHAR,
    details      VARCHAR,
    created_date TIMESTAMP   NOT NULL,
    updated_date TIMESTAMP
);
```

Código 21 – DDL Criação de Feedback de Produto

```
CREATE TABLE product_feedbacks (
    id           BIGINT      NOT NULL PRIMARY KEY,
    product_id   BIGINT      REFERENCES product,
    user_id      BIGINT      REFERENCES "user",
    type         VARCHAR,
    content      VARCHAR,
    rating       INTEGER,
    created_date TIMESTAMP   NOT NULL,
    updated_date TIMESTAMP
);
```

Código 22 – DDL Criação de Resposta de Produto

```
CREATE TABLE product_answer (
    id                   BIGINT      NOT NULL PRIMARY KEY,
    product_feedbacks_id BIGINT     REFERENCES product_feedbacks,
    content              VARCHAR,
    user_id              BIGINT     REFERENCES "user",
    created_date         TIMESTAMP  NOT NULL,
    updated_date         TIMESTAMP
);
```

Código 23 – DDL Criação de Foto de Produto

```
CREATE TABLE product_photo (
    id           BIGINT      NOT NULL PRIMARY KEY,
    product_id   BIGINT      REFERENCES product,
    "order"      INTEGER,
    name         VARCHAR,
    created_date TIMESTAMP   NOT NULL,
    updated_date TIMESTAMP
);
```

Código 24 – DDL Criação de Resolução de Foto de Produto

```
CREATE TABLE product_photo_resolution (
    id               BIGINT      NOT NULL PRIMARY KEY,
    product_photo_id BIGINT     REFERENCES product_photo,
    format           VARCHAR,
    data             BYTEA,
    size             BIGINT,
    resolution_x     BIGINT,
    resolution_y     BIGINT,
    created_date     TIMESTAMP  NOT NULL,
    updated_date     TIMESTAMP
);
```

Código 25 – DDL Criação de Interação de Produto

```
CREATE TABLE product_interaction (
    id            BIGINT      NOT NULL PRIMARY KEY,
    product_id    BIGINT      REFERENCES product,
    user_id       BIGINT      REFERENCES "user",
    added_to_cart BOOLEAN,
    favorited     BOOLEAN,
    created_date  TIMESTAMP   NOT NULL,
    updated_date  TIMESTAMP
);
```

Código 26 – DDL Criação de Notificação Push

```
CREATE TABLE notification_push (
    id            BIGINT      NOT NULL PRIMARY KEY,
    user_id       BIGINT      REFERENCES "user",
    title         VARCHAR     NOT NULL,
    content       BIGINT      NOT NULL,
    opened        BOOLEAN,
    redirect_link VARCHAR,
    created_date  TIMESTAMP   NOT NULL,
    updated_date  TIMESTAMP
);
```

Código 27 – DDL Criação de Notificação Email

```
CREATE TABLE notification_email (
    id           BIGINT      NOT NULL PRIMARY KEY,
    user_id      BIGINT      REFERENCES "user",
    provider     VARCHAR     NOT NULL,
    target_email VARCHAR     NOT NULL,
    title        VARCHAR     NOT NULL,
    subject      VARCHAR     NOT NULL,
    body         TEXT,
    status       VARCHAR,
    created_date TIMESTAMP   NOT NULL,
    updated_date TIMESTAMP
);
```

Código 28 – DDL Criação de Anexo de Notificação Email

```
CREATE TABLE notification_email_attachment (
    id                    BIGINT      NOT NULL PRIMARY KEY,
    notification_email_id BIGINT     REFERENCES notification_email,
    name                  VARCHAR,
    format                VARCHAR,
    data                  BYTEA,
    size                  BIGINT,
    created_date          TIMESTAMP  NOT NULL,
    updated_date          TIMESTAMP
);
```

Código 29 – DDL Criação de Pedido

```
CREATE TABLE "order" (
    id             BIGINT      NOT NULL PRIMARY KEY,
    user_id        BIGINT      REFERENCES "user",
    status         VARCHAR,
    shipping_date  TIMESTAMP,
    delivered_date TIMESTAMP,
    closing_date   TIMESTAMP,
    address        VARCHAR     NOT NULL,
    end_date       TIMESTAMP,
    created_date   TIMESTAMP   NOT NULL,
    updated_date   TIMESTAMP
);
```

Código 30 – DDL Criação de Itens do Pedido

```
CREATE TABLE order_items (
    id                 BIGINT      NOT NULL PRIMARY KEY,
    order_id           BIGINT      REFERENCES "order",
    product_id         BIGINT      REFERENCES product,
    quantity           BIGINT,
    price              BIGINT,
    status             VARCHAR,
    title              VARCHAR,
    category           VARCHAR,
    description        VARCHAR,
    history            JSONB,
    features           JSONB,
    estimated_delivery TIMESTAMP,
    created_date       TIMESTAMP   NOT NULL,
    updated_date       TIMESTAMP
);
```

Código 31 – DDL Criação de Ticket

```
CREATE TABLE ticket (
    id                  BIGINT      NOT NULL PRIMARY KEY,
    user_id             BIGINT      REFERENCES "user",
    product_id          BIGINT      REFERENCES product,
    order_id            BIGINT      REFERENCES "order",
    product_feedback_id BIGINT     REFERENCES product_feedbacks,
    responsible_id      BIGINT      REFERENCES "user",
    redactor_name       VARCHAR,
    redactor_email      VARCHAR,
    redactor_phone      VARCHAR,
    title               VARCHAR     NOT NULL,
    type                VARCHAR     NOT NULL,
    description         VARCHAR,
    status              VARCHAR,
    closing_date        TIMESTAMP,
    created_date        TIMESTAMP   NOT NULL,
    updated_date        TIMESTAMP
);
```

Código 32 – DDL Criação de Mensagem de Ticket

```
CREATE TABLE ticket_message (
    id           BIGINT      NOT NULL PRIMARY KEY,
    ticket_id    BIGINT      REFERENCES ticket,
    user_id      BIGINT      REFERENCES "user",
    content      TEXT,
    type         VARCHAR,
    created_date TIMESTAMP   NOT NULL,
    updated_date TIMESTAMP
);
```

Código 33 – DDL Criação de Anexo de Mensagem de Ticket

```
CREATE TABLE ticket_message_attachment (
    id           BIGINT      NOT NULL PRIMARY KEY,
    message_id   BIGINT      REFERENCES ticket_message,
    name         VARCHAR,
    format       VARCHAR,
    data         BYTEA,
    size         BIGINT,
    created_date TIMESTAMP   NOT NULL,
    updated_date TIMESTAMP
);
```

Código 34 – DDL Criação de Notificação Mobile

```
CREATE TABLE notification_mobile (
    id             BIGINT      NOT NULL PRIMARY KEY,
    user_id        BIGINT      REFERENCES "user",
    msisdn         VARCHAR,
    provider       VARCHAR,
    message_type   VARCHAR,
    content        TEXT,
    media_url      VARCHAR,
    status         VARCHAR,
    redirect_link  VARCHAR,
    created_date   TIMESTAMP   NOT NULL,
    updated_date   TIMESTAMP
);
```