UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
UNIDADE ACADÊMICA DE SISTEMAS E COMPUTAÇÃO
PROGRAMAÇÃO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

HIAGO NATAN FERNANDES DE SOUSA

UM EXPERIMENTO COMPARATIVO DA EFICÁCIA DE DIFERENTES LLM NA GERAÇÃO DE CENÁRIOS GHERKIN

CAMPINA GRANDE - PB
2025

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Programa de Pós-Graduação em Ciência da Computação

Um Experimento Comparativo da Eficácia de Diferentes LLM na Geração de Cenários Gherkin

Hiago Natan Fernandes de Sousa

Dissertação submetida à Coordenação do Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Engenharia de Software

Prof. Dr. Danilo Freire de Souza Santos (Orientador)
Dr. Mirko Barbosa Perkusich (Coorientador)

Campina Grande, Paraíba, Brasil

S725e Sousa, Hiago Natan Fernandes de.

Um experimento comparativo da eficácia de diferentes LLM na geração de cenários Gherkin / Hiago Natan Fernandes de Sousa. – Campina Grande, 2025.

141 f. : il. color.

Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2025.

"Orientação: Prof. Dr. Danilo Freire de Souza Santos, Prof. Dr. Mirko Barbosa Perkusich".

Referências.

1. Behavior-Driven Development (BDD). 2. Gherkin. 3. Modelos de Linguagem de Grande Escala. 4. Geração Automatizada de Cenários. 5. Avaliação de Qualidade. 6. Análise de Variabilidade. I. Santos, Danilo Freire de Souza. II. Perkusich, Mirko Barbosa. III. Título.

CDU 004.43(043)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECÁRIA MEIRE EMANUELA DA SILVA MELO CRB-15/568

HIAGO NATAN FERNANDES DE SOUSA

UM EXPERIMENTO COMPARATIVO DA EFICÁCIA DE DIFERENTES LLM NA GERAÇÃO DE CENÁRIOS GHERKIN

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande, pertencente à linha de pesquisa de Engenharia de Software e área de Concentração em Ciência da Computação, como requisito para a obtenção do título de Mestre em Ciência da Computação, pela seguinte banca examinadora:

Data de aprovação: 31/01/2025

Prof. Dr. Danilo Freire de Souza Santos (UFCG)
(Orientador)

Dr. Mirko Barbosa Perkusich (UFCG)
(Coorientador)

Prof. Dr. Kyller Costa Gorgônio (UFCG)
(Examinador interno)

Prof. Dr. Danyllo Wagner Albuquerque (IFPB)
(Examinador externo)

CAMPINA GRANDE - PB

2025

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
POS-GRADUACAO EM CIENCIA DA COMPUTACAO
Rua Aprígio Veloso, 882, Edifício Telmo Silva de Araújo, Bloco CG1, - Bairro Universitário, Campina Grande/PB, CEP 58429-900
Telefone: 2101-1122 - (83) 2101-1123 - (83) 2101-1124
Site: http://computacao.ufcg.edu.br - E-mail: secpg@computacao.ufcg.edu.br

FOLHA DE ASSINATURA PARA TESES E DISSERTAÇÕES

HIAGO NATAN FERNANDES DE SOUSA

UM EXPERIMENTO COMPARATIVO DA EFICÁCIA DE DIFERENTES LLM NA GERAÇÃO DE CENÁRIOS GHERKIN

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação como pré-requisito para obtenção do título de Mestre em Ciência da Computação.

Aprovada em: 31/01/2025

Prof. Dr. **Danilo Freire de Souza Santos**, UFCG, Orientador

Prof. Dr. **Mirko Barbosa Perkusich**, Orientador

Prof. Dr. **Kyller Costa Gorgônio**, UFCG, Examinador Interno

Prof. Dr. **Danyllo Wagner Albuquerque**, IFPB, Examinador Externo

Documento assinado eletronicamente por **DANILO FREIRE DE SOUZA SANTOS**, **PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 03/02/2025, às 09:38, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **Danyllo Wagner Albuquerque**, **Usuário Externo**, em 03/02/2025, às 10:09, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **KYLLER COSTA GORGONIO**, **PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 03/02/2025, às 11:08, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **MIRKO BARBOSA PERKUSICH**, **Usuário Externo**, em 03/02/2025, às 22:11, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

A autenticidade deste documento pode ser conferida no site https://sei.ufcg.edu.br/autenticidade, informando o código verificador **5193991** e o código CRC **D028D8F6**.

Resumo

O Behavior-Driven Development (BDD) é essencial no desenvolvimento de software moderno, com a linguagem Gherkin sendo crucial para especificar cenários de teste. No entanto, a criação manual desses cenários é demorada e propensa a erros. Os Grandes Modelos de Linguagem (LLM) surgem como uma solução inovadora para automatizar e otimizar esse processo, oferecendo uma alternativa mais eficiente e confiável.

Neste estudo, investigamos a eficácia de seis LLM (GPT-3.5 Turbo, GPT-4 Turbo, GPT-4o Mini, LLaMA 3, Phi-3 e Gemini) na geração automatizada de cenários Gherkin a partir de 1.286 cenários de teste reais. Aplicamos diferentes técnicas de *prompting*, como *zero-shot*, *one-shot* e *few-shot*, para avaliar a qualidade e a consistência das saídas produzidas. O objetivo foi identificar a técnica e o modelo mais adequados para a criação de cenários BDD.

Para conduzir a análise, foram selecionadas medidas de avaliação de qualidade e variabilidade, que foram correlacionadas com avaliações qualitativas realizadas por especialistas. Isso garantiu a escolha de métricas representativas que refletem adequadamente a qualidade dos cenários gerados. Além disso, análises estatísticas foram realizadas para verificar a existência de diferenças significativas entre os modelos e técnicas aplicadas, assegurando a robustez metodológica do estudo.

A análise de variabilidade apontou que a consistência dos modelos depende da técnica utilizada: em *zero-shot*, o Gemini foi mais consistente, enquanto LLaMA 3 e GPT-3.5 Turbo apresentaram maior variabilidade. Em *one-shot*, GPT-4o Mini e GPT-4 Turbo se destacaram pela estabilidade, ao passo que em *few-shot*, GPT-4o Mini e LLaMA 3 foram os mais estáveis. A análise de desempenho revelou que a técnica *zero-shot* foi a mais eficaz em diversos contextos, especialmente quando aplicada ao modelo Gemini. No entanto, análises estatísticas, como o teste de Kruskal-Wallis, demonstraram que as diferenças observadas entre os modelos não foram estatisticamente significativas.

Palavras-chave: Behavior-Driven Development, Gherkin, Modelos de Linguagem de Grande Escala, geração automatizada de cenários, avaliação de qualidade, análise de variabilidade.

Abstract

Behavior-Driven Development (BDD) is essential in modern software development, with the Gherkin language playing a crucial role in specifying test scenarios. However, the manual creation of these scenarios is time-consuming and error-prone. Large Language Models (LLMs) emerge as an innovative solution to automate and optimize this process, offering a more efficient and reliable alternative.

In this study, we investigated the effectiveness of six LLMs (GPT-3.5 Turbo, GPT-4 Turbo, GPT-4o Mini, LLaMA 3, Phi-3, and Gemini) in the automated generation of Gherkin scenarios from 1,286 real-world test scenarios. We applied different prompting techniques, such as zero-shot, one-shot, and few-shot, to evaluate the quality and consistency of the generated outputs. The goal was to identify the most suitable technique and model for creating BDD scenarios.

To conduct the analysis, we selected quality and variability evaluation measures, which were correlated with qualitative assessments performed by experts. This ensured the choice of representative metrics that adequately reflect the quality of the generated scenarios. Additionally, statistical analyses were performed to verify the existence of significant differences between the models and techniques applied, ensuring the methodological robustness of the study.

The variability analysis indicated that the consistency of the models depends on the technique used: in zero-shot, Gemini was more consistent, while LLaMA 3 and GPT-3.5 Turbo showed higher variability. In one-shot, GPT-4o Mini and GPT-4 Turbo stood out for their stability, whereas in few-shot, GPT-4o Mini and LLaMA 3 were the most stable. The performance analysis revealed that the zero-shot technique was the most effective in various contexts, especially when applied to the Gemini model. However, statistical analyses, such as the Kruskal-Wallis test, demonstrated that the observed differences between the models were not statistically significant.

Keywords: Behavior-Driven Development, Gherkin, Large Language Models, automated scenario generation, quality evaluation, variability analysis.

Agradecimentos

A Deus, força maior que me guiou e sustentou em cada etapa desta jornada, dedico minha eterna gratidão. "Não temas, porque estou contigo; não te assombres, porque eu sou teu Deus; eu te fortaleço, e te ajudo, e te sustento com a destra da minha justiça." (Isaías 41:10)

Aos meus pais, Ana Lúcia e Antonio, por serem pilares inabaláveis de amor e apoio. Agradeço por estarem presentes em todos os momentos, por me impulsionarem a sempre buscar o melhor e por acreditarem em mim, mesmo quando eu duvidava.

À minha irmã, Hiolanda, e meu irmão, Higor, por todo o carinho, cumplicidade e por tornarem essa caminhada mais leve com seus sorrisos e incentivos.

A João Pedro, meu noivo, meu porto seguro. Agradeço por todo o apoio, por estar ao meu lado em cada passo dado, por me amparar nos momentos difíceis e por me ajudar. Seu amor e compreensão foram essenciais para a realização deste sonho.

Aos meus orientadores, Dr. Mirko Perkusich e Dr. Danilo Freire, meus sinceros agradecimentos pela dedicação, paciência e ensinamentos que foram essenciais para a realização desta dissertação. Agradeço especialmente pela confiança em meu potencial e por me inspirarem a trilhar o caminho da pesquisa.

Aos meus amigos, André Andrade, David Medeiros, Gabriela Motta e Gabriel Araujo, que acompanharam essa trajetória de perto. Agradeço pela amizade, pelo companheirismo e por todos os momentos de descontração que me proporcionaram durante esta jornada.

A todos que, direta ou indiretamente, contribuíram para a concretização deste trabalho, meu sincero agradecimento.

Conteúdo

Lista de Símbolos

BDD - *Behavior-Driven Development*

TDD - *Test-Driven Development*

SDLC - *Software Development Life Cycle*

LLM - *Large Language Model*

SLM - *Small Language Model*

GPT-3.5 - *Generative Pre-trained Transformer 3.5*

GPT-4 - *Generative Pre-trained Transformer 4*

CSV - *Comma-Separated Values*

API - *Application Programming Interface*

JSON - *JavaScript Object Notation*

BERT - *Bidirectional Encoder Representations from Transformers*

METEOR - *Metric for Evaluation of Translation with Explicit ORdering*

CRUD - *Create, Read, Update, Delete*

SEED - *Semente aleatória*

top_k - *Número de tokens considerados na geração de texto*

top_p - *Probabilidade acumulada de tokens considerados na geração de texto*

pt - *Código do idioma português*

P - *Precisão*

R - *Revocação (Recall)*

F1 - *Média harmônica entre precisão e revocação*

IQR - *Intervalo interquartil*

Q1 - *Primeiro quartil*

Q3 - *Terceiro quartil*

CV - *Coeficiente de Variação*

DP - *Desvio Padrão*

NLP - *Natural Language Processing*

k - *Número de tokens considerados na técnica top-k*

p - *Probabilidade acumulada de tokens considerados na técnica top-p*

α - *Parâmetro que controla o peso relativo entre precisão e recall na medida METEOR*

γ - *Parâmetro que define a penalidade máxima na medida METEOR*

β - *Parâmetro que ajusta a sensibilidade da penalidade na medida METEOR*

ch - *Número de agrupamentos (chunks) consecutivos de uniagramas alinhados na medida METEOR*

$\mathbf{A}$ - *Vetor* $\mathbf{A}$

a - *Elemento do vetor* $\mathbf{A}$

$\mathbf{B}$ - *Vetor* $\mathbf{B}$

b - *Elemento do vetor* $\mathbf{B}$

$\mathbf{x}$ - *Vetores das representações da sentença de referência gerados pelo modelo BERT*

$\hat{\mathbf{x}}$ - *Vetores da sentença candidata gerados pelo modelo BERT*

m - *Número de uniagramas alinhados na medida METEOR*

t - *Total de uniagramas na tradução gerada na medida METEOR*

r - *Total de uniagramas na tradução de referência na medida METEOR*

n - *Número de dimensões*

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A crescente complexidade dos sistemas modernos exige metodologias que promovam a integração contínua entre requisitos de negócios, desenvolvimento e garantia de qualidade. Nesse contexto, o *Desenvolvimento Guiado por Comportamento* (do inglês *Behavior-Driven Development*, ou BDD) [40, 46] tem se destacado como uma prática essencial. Surgindo como uma evolução do *Desenvolvimento Guiado por Testes* (do inglês *Test-Driven Development*, ou TDD), o BDD estabelece uma linguagem comum que conecta as equipes, permitindo que as especificações de comportamento sejam traduzidas em cenários de teste automatizados. Por outro lado, diferentemente do TDD, que é altamente técnico, o BDD utiliza linguagem natural semi-estruturada, como a Gherkin, facilitando a participação de analistas de negócios e outros profissionais não técnicos na criação e validação dos cenários de teste [46, 53].

Apesar dos benefícios do BDD em termos de colaboração e alinhamento, a criação manual de cenários apresenta desafios, especialmente em projetos complexos. Manter a consistência e a qualidade dos cenários pode ser difícil em equipes distribuídas e com requisitos dinâmicos [34]. Em ambientes de larga escala, o alto volume de especificações torna o processo manual suscetível a erros e ineficiências. A automação, portanto, surge como uma necessidade para maximizar o potencial do BDD.

Nesse sentido, devido à sua capacidade de compreender linguagem natural, adaptar-se a diferentes contextos e gerar textos consistentes e sintaticamente corretos, os *Large Language Models* (LLM) emergiram como ferramentas promissoras para automatizar a criação de cenários de teste em BDD. Essas características permitem que os LLM lidem com a varia-

bilidade e inconsistência inerentes ao processo manual, oferecendo soluções escaláveis para projetos complexos e dinâmicos. Estudos indicam que modelos como GPT-3.5 e GPT-4 [40] são eficazes na geração de cenários que aderem às regras de sintaxe do Gherkin, mesmo com poucos exemplos de treinamento.

Os LLM oferecem vantagens significativas em relação ao BDD tradicional, como a capacidade de capturar requisitos implícitos e alinhar os cenários de teste aos objetivos de negócio. Enquanto o BDD tradicional pode negligenciar restrições de domínio e dependências contextuais, os LLM [30, 40] podem automatizar a identificação dessas nuances, gerando cenários que atendem tanto às prioridades do cliente quanto aos padrões de desenvolvimento. Essa capacidade promove uma integração mais eficiente entre especificações de negócios e testes técnicos, resultando em maior consistência e escalabilidade.

A aplicação de LLM em BDD [24, 54] apresenta desafios significativos, como a validação da qualidade dos cenários de teste gerados e a mitigação de erros de interpretação. Medidas de avaliação automática [59] são essenciais para analisar a precisão desses cenários, permitindo uma comparação objetiva com cenários manuais. Além disso, é crucial desenvolver estratégias para a reutilização e revisão de cenários gerados automaticamente, garantindo sua adaptabilidade a mudanças nos requisitos [35]. Apesar do potencial, o uso de LLM em engenharia de software também introduz ameaças à validade da pesquisa, como o uso de modelos fechados, vazamento de dados e problemas de reprodutibilidade [65]. Pesquisas recentes [40, 43] têm explorado como os LLM podem auxiliar na automação de tarefas relacionadas ao desenvolvimento de software, incluindo a geração de código e cenários de teste.

1.1 Problemática

Um dos principais desafios na aplicação do BDD é a lacuna existente entre a descrição informal de cenários de teste e sua especificação formal no formato Gherkin, padrão do BDD. Essa lacuna exige um esforço considerável para traduzir descrições em linguagem natural para cenários estruturados e executáveis [43, 46]. A título de exemplo, a descrição *"O usuário deve poder fazer login com suas credenciais e acessar sua conta"* precisa ser convertida para o seguinte formato:

Scenario: Login bem-sucedido
Given o usuário está na página de login
When o usuário insere credenciais válidas
Then o usuário deve ser redirecionado para o painel

Esse processo de tradução exige não apenas esforço manual para detalhar cada passo, mas também conhecimento técnico da sintaxe Gherkin. A falta de uma ponte eficiente entre esses dois formatos dificulta a adoção e a escalabilidade do BDD em organizações cujos sistemas possuem requisitos complexos e dinâmicos [9].

Diante desse desafio, os LLM surgem como uma solução promissora. Modelos como o GPT-4 demonstram capacidade de interpretar linguagem natural e gerar código Gherkin com alta precisão e aderência aos padrões [24]. Essa capacidade se mantém mesmo com poucos dados de treinamento ou exemplos, tornando os LLM adequados para automatizar tarefas com alta variabilidade linguística [43].

Os LLM oferecem uma solução automatizada para a geração de cenários BDD, com potencial para reduzir o esforço manual, acelerar o desenvolvimento e melhorar a qualidade dos casos de teste gerados [54]. A capacidade de compreender nuances linguísticas e contextuais torna os LLM ferramentas valiosas para transpor a lacuna entre a descrição informal e a especificação formal de cenários de teste, promovendo consistência e escalabilidade [46, 48].

A automação de cenários BDD tem se mostrado uma estratégia eficaz para otimizar processos de desenvolvimento de software, especialmente em projetos de grande escala. Dados indicam que a adoção de BDD pode reduzir o tempo de desenvolvimento em até 30%, além de diminuir as taxas de erro em testes de aceitação em aproximadamente 25% [5]. Esses resultados evidenciam que a automação de cenários de teste, quando bem implementada, não apenas acelera o ciclo de desenvolvimento, mas também contribui para a entrega de software com maior qualidade. A utilização de LLM para gerar cenários BDD de forma automatizada surge, portanto, como uma solução promissora para superar os desafios de escalabilidade e complexidade, especialmente em sistemas com requisitos dinâmicos e multifacetados.

É crucial reconhecer que a gama de LLM disponíveis é vasta, cada um com características e potencialidades específicas que influenciam os resultados. Já modelos como o GPT-4, reconhecido por sua versatilidade, destacam-se em tarefas que exigem generalização, en-

quanto modelos especializados, como Codex e Flan-PaLM, são otimizados para áreas como geração de código e processamento de instruções [50]. Fatores como tamanho, arquitetura, dados de treinamento e técnicas de ajuste fino também impactam a qualidade dos resultados. Essa diversidade ressalta a importância da escolha criteriosa do modelo, considerando os objetivos e o contexto da aplicação [50]. Além disso, avanços recentes na pesquisa destacaram o uso de LLM para tarefas de engenharia de software, como geração de código, identificação de vulnerabilidades e manutenção de software, mostrando que esses modelos possuem potencial significativo para transformar práticas de desenvolvimento e melhorar a eficiência em diversas áreas de aplicação [37, 73].

Considerando a diversidade de LLM e a complexidade de suas aplicações, o desenvolvimento e a avaliação de uma abordagem para a geração automatizada de cenários de teste em Gherkin foram investigados nesta pesquisa. A base de dados utilizada, composta por 1.286 cenários de teste reais extraídos de sistemas corporativos, abrange funcionalidades como autenticação, controle de acesso e operações CRUD.

Portanto, o problema central desta dissertação é: Quão eficazes são os modelos de linguagem (LLM) na geração automatizada de cenários de teste no formato Gherkin a partir de descrições textuais em linguagem natural?

1.2 Objetivo

O objetivo principal desta dissertação é avaliar a **eficácia** de diferentes LLM na geração automatizada de cenários de teste no formato *Gherkin* (utilizado em *BDD*) a partir de descrições em linguagem natural. Para isso, foi conduzido um experimento comparativo com LLM amplamente reconhecidos, como GPT-3.5 Turbo, GPT-4 Turbo, GPT-4o Mini, LLaMA 3, Phi-3 e Gemini.

Para atingir o objetivo geral, esta pesquisa foca nesses objetivos específicos:

- **Identificar boas práticas para escrita de casos de teste BDD:** Identificar e consolidar boas práticas recomendadas para a escrita de casos de teste BDD no formato Gherkin, com o propósito de orientar a formulação de prompts adequados para experimentos com LLM.

- **Coletar e preparar os casos de teste para o experimento:** Construir uma base de casos de teste composta por descrições livres e cenários no formato Gherkin, validados por especialistas, para servir como referência (*ground truth*) em análises comparativas de eficácia.

- **Selecionar uma medida apropriada para avaliar a eficácia dos LLM:** Identificar e validar medidas adequadas para avaliar o desempenho dos LLM na transformação de descrições livres em casos de teste Gherkin, considerando precisão sintática, semântica e aderência às práticas do BDD.

- **Avaliar a eficácia dos LLM na geração de casos de teste Gherkin:** Avaliar a eficácia de LLMs, como GPT-3.5, GPT-4, Gemini, LLaMA e Phi-3, na geração de cenários Gherkin a partir de descrições livres, com base na qualidade dos resultados gerados.

- **Analisar a variabilidade dos resultados gerados pelos LLM:** Investigar a variabilidade nos cenários gerados em múltiplas execuções, avaliando os impactos dessa variabilidade na consistência dos resultados e sua aplicabilidade prática.

1.3 Metodologia

A pesquisa foi conduzida com uma abordagem experimental e exploratória, visando avaliar a eficácia de seis LLM na geração automatizada de cenários de teste no formato BDD a partir de descrições livres em linguagem natural. O estudo foi estruturado em duas etapas principais: (1) **Construção da Base Experimental** e (2) **Experimento Comparativo**.

Na primeira etapa, foram identificadas boas práticas para a escrita de cenários de teste BDD, seguidas pela coleta e preparação de uma base de casos de teste que serviu como referência (*ground-truth*) para a avaliação dos LLM. Em seguida, foi realizada a seleção da medida mais adequada para mensurar a eficácia dos modelos, considerando critérios como relevância, consistência e capacidade de diferenciar as abordagens de *prompting* (*zero-shot*, *one-shot* e *few-shot*).

A seleção dos LLM para este estudo foi orientada por critérios que visam abranger diversidade em características, arquiteturas e aplicações. Incluíram-se LLM com mais de 10 bilhões de parâmetros (GPT-3.5 Turbo, GPT-4 Turbo e Gemini), reconhecidos pelo estado

da arte em processamento e desempenho em tarefas complexas, além de modelos compactos e eficientes (Phi-3 e GPT-4o Mini), que equilibram desempenho e custo computacional, adequados para cenários com restrições de recursos. A incorporação, por exemplo, de LLM como LLaMA 3 e Gemini permite a comparação entre arquiteturas distintas, desenvolvidas por organizações diferentes no mercado [1, 16, 23, 27, 31, 50].

Na segunda etapa, conduziu-se o experimento comparativo, no qual os LLM foram avaliados quanto à sua capacidade de gerar cenários de teste Gherkin alinhados às boas práticas de BDD. Além disso, analisou-se a variabilidade dos resultados gerados pelos modelos e os desafios enfrentados durante a aplicação dos LLM. Para garantir uma avaliação sistemática, os objetivos específicos da pesquisa foram diretamente vinculados a questões de pesquisa, conforme detalhado na Tabela 1.1.

A Tabela 1.1 apresenta a relação entre os objetivos específicos e as questões de pesquisa, organizada de forma a guiar a execução do estudo. Cada objetivo foi associado a uma ou mais questões, garantindo que todas as etapas da pesquisa fossem alinhadas aos resultados esperados. Por exemplo, o objetivo de *identificar boas práticas para escrita de cenários de teste BDD* está diretamente relacionado às questões Q1 e Q2, que avaliam as técnicas de *prompting* e as medidas de qualidade dos cenários gerados. Da mesma forma, o objetivo de *comparar a eficácia dos LLM* está vinculado às questões Q6 e Q7, que investigam o desempenho dos modelos e o impacto das técnicas de *prompting*. Essa estrutura permite uma análise dos resultados, além de facilitar a interpretação dos dados obtidos.

Para facilitar a reprodutibilidade, disponibilizou-se um repositório com todos os artefatos produzidos na pesquisa, incluindo os cenários de teste utilizados, os *prompts* aplicados e os resultados obtidos[1]. A validação das medidas utilizadas para avaliar a qualidade dos cenários gerados está descrita no Capítulo 3, enquanto a análise comparativa do desempenho dos LLM é apresentada no Capítulo 4.

A organização da pesquisa em torno desses objetivos e questões permitiu uma abordagem metodológica clara e replicável, além de facilitar a interpretação dos resultados obtidos. A Tabela 1.1 serve como um guia para entender como cada etapa do estudo contribui para responder às questões de pesquisa propostas, garantindo que todos os aspectos relevantes fossem cobertos de forma sistemática.

[1]Disponível em: `https://github.com/hiagonfs/bdd-scenario-evaluation/`

Tabela 1.1: Objetivos Específicos e Questões de Pesquisa

Objetivos Específicos	**Questões de Pesquisa**
Identificar boas práticas para escrita de cenários de teste BDD	Q1. Qual técnica de prompting (*zero-shot*, *one-shot* ou *few-shot*) gera cenários Gherkin mais alinhados às boas práticas do BDD? Q2. Quais medidas melhor avaliam a qualidade dos cenários Gherkin gerados automaticamente e diferenciam as abordagens de prompting?
Coletar e preparar os cenários de teste para o experimento	Q3. Quais critérios devem ser considerados para selecionar e organizar uma base de cenários de teste representativa para avaliar os LLM?
Selecionar uma medida apropriada para avaliar a eficácia dos LLM	Q4. Qual medida é mais adequada para avaliar a eficácia dos LLM na transformação de descrições livres em cenários Gherkin? Q5. Quais critérios devem ser aplicados para selecionar a medida mais apropriada, garantindo relevância e consistência?
Comparar a eficácia dos LLM na geração de cenários de teste Gherkin	Q6. Como diferentes LLM se comparam na geração de cenários de teste Gherkin? Q7. Como as técnicas de prompting impactam a qualidade e consistência dos cenários gerados?
Analisar a variabilidade dos resultados gerados pelos LLM	Q8. Como a variabilidade dos resultados dos LLM afeta a consistência dos cenários de teste Gherkin?

1.3.1 Construção da Base Experimental

A construção da base experimental foi estruturada nas seguintes etapas principais:

- **Identificação de boas práticas para escrita de cenários de teste BDD:** Boas práticas foram analisadas e consolidadas a partir da literatura e recomendações do mercado, servindo como base para a formulação de prompts eficazes para os LLM [54].
- **Coleta e preparação dos cenários de teste:** Foram coletados 1.286 cenários de teste escritos de forma livre a partir do estudo de de Souza Filho [21]. Após um processo de limpeza da base – que incluiu a remoção de descrições curtas e prefixos não informativos, a priorização de funcionalidades críticas e a exclusão de redundâncias por similaridade semântica – foram selecionados aleatoriamente 10 cenários. Esses cenários foram transformados por um especialista em BDD no formato Gherkin, seguindo as boas práticas identificadas, estabelecendo uma base de referência (*ground truth*). É importante ressaltar que os domínios e categorias dos testes coletados não foram categorizados neste estudo, e a base foi utilizada em sua forma original, sem divisão por categorias ou domínios específicos.
- **Seleção de medida(s) apropriada(s) para avaliação:** A medida mais adequada foi escolhida com foco em precisão sintática, semântica e aderência às práticas recomendadas do BDD.

É importante destacar que a base de cenários foi utilizada em sua forma original, sem categorização por domínios específicos, o que significa que a diversidade de contextos e funcionalidades presentes na base inicial pode ter sido modificada até a seleção final das descrições dos casos. inclusive, após a seleção dos 10 casos finais, a diversidade de cenários pode ter sido impactada, pois ocorreu uma redução amostral devido à quantidade baixa de recursos para análise.

1.3.2 Experimento Comparativo

A segunda atividade concentrou-se na avaliação e comparação do desempenho dos LLM na geração automatizada de cenários BDD. Essa etapa foi organizada em três etapas principais:

- **Comparação da eficácia dos LLM:** A eficácia dos LLM de linguagem (como GPT-3.5, GPT-4, Gemini, LLaMA e Phi-3) foi avaliada por meio da aplicação de diferentes técnicas de prompting (*zero-shot*, *one-shot* e *few-shot*). Antes disso, uma análise manual foi realizada para selecionar a medida mais adequada para avaliar a eficácia das técnicas e perfomance dos LLM, garantindo a relevância e consistência dos resultados obtidos.

- **Análise da variabilidade dos resultados:** A variabilidade dos cenários gerados foi investigada a partir de múltiplas execuções dos LLM, considerando a natureza probabilística dos LLM e avaliando seus impactos na consistência dos resultados obtidos.

- **Identificação de desafios e limitações:** Os desafios observados durante a geração automatizada incluíram aspectos como adaptação a domínios específicos, escalabilidade e a necessidade de mudanças para garantir que os cenários estivessem alinhados com as práticas recomendadas do BDD.

1.4 Contribuições

Este estudo avaliou o impacto de diferentes técnicas de prompting na geração automatizada de cenários de teste Gherkin, buscando otimizar o processo de desenvolvimento de software.

A técnica *zero-shot* obteve o melhor desempenho médio na geração de cenários de teste, mostrando-se ideal para produção rápida e eficiente, especialmente com LLM como Gemini e GPT-4o Mini. Em contraste, a técnica *one-shot* apresentou maior consistência, crucial para cenários que exigem previsibilidade, com destaque para os LLM GPT-3.5 Turbo e GPT-4 Turbo. A técnica *few-shot* apresentou resultados mais variáveis.

Com base nos resultados obtidos, foram formuladas orientações práticas para a escolha da técnica de *prompting* ideal, considerando o LLM e os requisitos da tarefa. Essas descobertas contribuem para a melhoria da automação de testes e para o aprimoramento das práticas de BDD, facilitando a criação de cenários de teste mais completos e eficazes.

Além disso, este estudo forneceu dados e insights valiosos sobre o desempenho de diferentes LLM em tarefas de geração de código e testes, impulsionando a pesquisa e o desen-

volvimento de novas aplicações para LLM. O compartilhamento das práticas e estratégias de *prompting* aqui apresentadas visa incentivar a exploração do potencial máximo desses LLM.

Adicionalmente, foi realizado o teste estatístico de Kruskal-Wallis para comparar as distribuições das pontuações de desempenho da melhor técnica de *prompting* para cada LLM avaliado. Os resultados indicaram que, embora diferenças nas médias tenham sido observadas, essas variações não foram estatisticamente significativas ($H = 6.44$, $p = 0.27$). Essa análise destaca que, apesar de variações quantitativas entre os modelos, os dados não fornecem evidências robustas para afirmar a superioridade estatística de um modelo em relação aos outros no contexto avaliado, indicando que a escolha do modelo deve considerar outros fatores.

Finalmente, propõe-se um design experimental para estudos comparativos de LLM em Engenharia de Software, considerando a variabilidade dos resultados e a validação das medidas de avaliação. Este estudo abre caminho para novas pesquisas e contribui para o avanço da área.

1.5 Estrutura do Documento

Este trabalho está organizado em cinco capítulos, descritos a seguir:

- **Capítulo 1 - Introdução:** Apresenta o contexto da pesquisa, destacando a problemática que motivou este estudo, os objetivos propostos e a estrutura geral do documento.
- **Capítulo 2 - Fundamentação Teórica:** Explora os conceitos e teorias relevantes para o entendimento da pesquisa, com foco em BDD e LLM. Este capítulo fornece a base conceitual necessária para compreender os aspectos técnicos e metodológicos do estudo.
- **Capítulo 3 - Validação da Medida:** Descreve em detalhes a metodologia utilizada para validar a pesquisa. São abordados aspectos como a seleção da base de dados, o desenvolvimento experimental e a definição das medidas de avaliação.
- **Capítulo 4 - Experimento Comparativo:** Apresenta uma análise detalhada dos LLM avaliados. Este capítulo compara o desempenho e a variabilidade dos LLM sob dife-

rentes técnicas de *prompting*:*zero-shot*, *one-shot* e *few-shot*, destacando os principais achados do estudo.

- **Capítulo 5 - Considerações Finais:** Discute os resultados obtidos e suas implicações práticas, respondendo as questões de pesquisa levantadas, além de abordar as limitações do estudo. Este capítulo também apresenta sugestões para trabalhos futuros, oferecendo um panorama para pesquisas subsequentes.

Capítulo 2

Fundamentação Teórica

Este capítulo apresenta os fundamentos teóricos necessários para a pesquisa, com foco em três eixos principais: o BDD, os LLM e a geração de casos de teste automatizados. Inicialmente, serão discutidos os conceitos do BDD, abrangendo sua definição, histórico, princípios, práticas e a linguagem Gherkin, bem como os benefícios e desafios associados. Em seguida, será explorado o papel dos LLM, abordando seus conceitos básicos, métodos de treinamento, aplicações em engenharia de software e implicações éticas. Por fim, será analisada a geração de casos de teste com LLM, com destaque para a literatura existente, técnicas de *prompting* e medidas de avaliação, estabelecendo a base para compreender o contexto e os objetivos da pesquisa.

2.1 Teste de Software

O teste de software é uma atividade essencial no ciclo de vida do desenvolvimento de software (*Software Development Life Cycle*, ou SDLC), garantindo que os produtos atendam a requisitos funcionais e de qualidade antes de sua liberação. Segundo Baqar e Khanda [8], o principal objetivo do teste de software é identificar defeitos, validar funcionalidades e assegurar que o software se comporte conforme o esperado em diferentes cenários. Além disso, os testes desempenham um papel fundamental na verificação da confiabilidade, eficiência e usabilidade do sistema, assegurando que ele atenda aos requisitos de qualidade estabelecidos. No entanto, o teste de software é uma prática considerada custosa, exigindo tempo e recursos significativos. Métodos tradicionais enfrentam desafios como a criação

manual de casos de teste, erros humanos e cobertura insuficiente, o que pode comprometer a confiabilidade e segurança dos sistemas.

A estrutura do teste de software envolve diferentes níveis e tipos de testes. Testes unitários verificam componentes isolados, enquanto testes de integração analisam a comunicação entre módulos. Além disso, há testes de sistema, que avaliam o comportamento global do software, e testes de aceitação, que garantem que o produto atende às expectativas dos usuários finais. Metodologias tradicionais de geração de testes incluem técnicas baseadas em especificação, código e histórico de falhas. Métodos como particionamento de equivalência e análise de valores-limite permitem cobrir um amplo espectro de entradas de teste. No entanto, Baqar e Khanda ([8]) destacam que a criação manual de testes frequentemente falha em cobrir cenários inesperados. Para enfrentar essa limitação, abordagens modernas como TDD e BDD têm sido amplamente adotadas.

O TDD enfatiza uma abordagem *test-first*, na qual os testes são escritos antes da implementação do código, garantindo que cada unidade do sistema passe por validação contínua. Esse processo segue o ciclo "Red-Green-Refactor", no qual o desenvolvedor escreve um teste que inicialmente falha ("Red"), implementa a mínima mudança necessária para passar no teste ("Green") e, em seguida, refatora o código para garantir sua qualidade e manutenibilidade. Essa abordagem reduz a taxa de defeitos e melhora a modularidade do código, embora possa aumentar o tempo inicial de desenvolvimento [19].

Por outro lado, o BDD amplia os princípios do TDD, focando na especificação do comportamento esperado do software em uma linguagem compreensível para desenvolvedores, testadores e partes interessadas não técnicos. O BDD utiliza uma estrutura padronizada "Given-When-Then" para descrever cenários de teste baseados em requisitos do usuário. Isso promove maior colaboração entre equipes, garantindo que o software esteja alinhado com as expectativas do negócio. No entanto, um dos desafios do BDD está na definição clara de requisitos, pois cenários mal especificados podem gerar ambiguidades na implementação [19].

Ambas as abordagens, TDD e BDD, representam avanços significativos na engenharia de testes, permitindo maior eficiência e confiabilidade no processo de desenvolvimento de software. Enquanto o TDD foca na validação contínua de unidades de código, o BDD promove uma visão mais ampla, alinhando o desenvolvimento às necessidades do negócio e

facilitando a comunicação entre as partes envolvidas. A escolha entre essas metodologias depende das necessidades do projeto e da maturidade da equipe de desenvolvimento, mas ambas contribuem para a entrega de software de alta qualidade.

2.1.1 Behavior-Driven Development (BDD)

BDD representa uma evolução no desenvolvimento de software, originando-se como uma extensão do TDD e focando na colaboração entre desenvolvedores, testadores e partes interessadas não técnicas [72]. BDD é uma prática ágil de desenvolvimento de software que se concentra na entrega do comportamento do sistema, garantindo que os desenvolvedores e especialistas do domínio usem a mesma linguagem [58]. Como metodologia ágil, o BDD visa alinhar os esforços das equipes de desenvolvimento e negócios por meio da criação de cenários de teste baseados em critérios de aceitação claros e acessíveis. Solis e Wang [69] identificaram características distintas do BDD, como a linguagem ubíqua, a decomposição iterativa e a especificação orientada ao comportamento.

Essa abordagem utiliza linguagens estruturadas, como o Gherkin, que permitem a definição de cenários no formato *Given-When-Then*, promovendo a rastreabilidade entre requisitos e testes automatizados [3]. O BDD tem desempenhado um papel crucial na verificação da qualidade de softwares, em diversos setores [12]. Além disso, o BDD utiliza uma linguagem específica do domínio, chamada *Gherkin*, para descrever os comportamentos do software, escondendo seus detalhes de implementação [17]. Esta abordagem coloca ênfase em garantir que o software esteja alinhado com as necessidades reais dos usuários, promovendo uma compreensão clara e compartilhada dos requisitos.

A linguagem comum usada no BDD para descrever o comportamento do software é uma de suas características mais distintivas. Ao expressar requisitos em termos naturais, as equipes podem evitar mal-entendidos e garantir que todos os envolvidos no projeto estejam com conhecimentos alinhados [15]. Estudos apontam que o BDD é particularmente eficaz na redução de discrepâncias entre os requisitos do cliente e a implementação técnica. Por exemplo, Mughal et al. [51] demonstram como o uso de cenários reutilizáveis em frameworks como Cucumber melhora a modularidade e a eficiência das suítes de teste, especialmente em ambientes de desenvolvimento ágil. Além disso, essa linguagem não serve apenas como documentação: as descrições de comportamento são transformadas em testes automatizados,

tornando as especificações executáveis [17].

Ferramentas, incluindo *Cucumber*, *JBehave* e *Selenium*, têm desempenhado um papel fundamental na prática do BDD, facilitando a transformação de especificações escritas em linguagem natural em testes automatizados [12, 17]. No entanto, o estudo [58] destaca a ferramenta *Codeception*, que pode implementar testes de aceitação do usuário com uma perspectiva de desejos e condições do usuário usando o modelo de desenvolvimento BDD. Um aspecto central do BDD é sua integração com ferramentas de automação, como Jenkins e IDEs com suporte a *auto-complete* de definições de passos, que simplificam a escrita e manutenção dos testes [25]. Essas integrações não apenas aceleram o processo de desenvolvimento de testes, mas também aumentam a consistência entre os cenários gerados, o que é essencial para grandes projetos com múltiplas equipes. Através dessas ferramentas, a comunicação entre equipes técnicas e não técnicas é aprimorada, reduzindo as chances de desalinhamento nas expectativas. O trabalho de Barus et al. [55] destaca a aplicabilidade do BDD em sistemas complexos, sugerindo que sua abordagem iterativa e colaborativa facilita a adaptação a mudanças frequentes nos requisitos.

Dan North, criador do BDD, enfatizou a importância da colaboração no processo de desenvolvimento [52]. Ao envolver todos as partes interessadas, o BDD busca construir um entendimento compartilhado do objetivo final do software, garantindo que as necessidades do usuário sejam atendidas. O estudo de Raharjana et al. [58] também ressalta a importância da automação na aceitação de testes, indicando que o teste automatizado de aceitação era mais eficiente em termos de custo e que os desenvolvedores tinham mais confiança ao usar essa abordagem.

Ao contrário de abordagens tradicionais que podem se concentrar em detalhes técnicos, o BDD prioriza o comportamento desejado do software [2]. Esta mudança de foco, do "como" para o "quê", ajuda as equipes a permanecerem centradas nas necessidades do usuário. Por fim, a literatura também destaca desafios na adoção do BDD, como a necessidade de treinamento para equipes pouco familiarizadas com a metodologia e o esforço inicial na definição de cenários abrangentes [14]. Em conclusão, o BDD, com sua ênfase na colaboração, comunicação e comportamento desejado, oferece uma abordagem refrescante ao desenvolvimento de software, especialmente em um cenário onde a complexidade dos projetos continua crescendo [68]. Além disso, a ferramenta para geração de casos de teste BDD, como apresentado

por Raharjana et al. [58], demonstra a crescente influência e aplicação prática do BDD no cenário atual de desenvolvimento de software [58]. Apesar dos desafios, o BDD tem se mostrado uma ferramenta poderosa na entrega de software de alta qualidade, promovendo um ciclo de desenvolvimento mais transparente e eficiente.

2.1.1.1 Linguagem Gherkin

Um dos pilares que sustenta a eficácia do BDD é a linguagem Gherkin, que desempenha um papel central na implementação dessa metodologia. A linguagem Gherkin é amplamente reconhecida como uma ferramenta essencial para a implementação de práticas de BDD. Ela adota uma sintaxe simples e estruturada baseada em texto natural, utilizando palavras-chave como *Given*, *When* e *Then* para definir cenários de teste que são ao mesmo tempo compreensíveis por humanos e passíveis de automação. Segundo Farooq et al. [25], a principal vantagem dessa abordagem é sua capacidade de atuar como uma linguagem ubíqua, unificando o entendimento entre desenvolvedores e partes interessadas não técnicas.

A estrutura simplificada do Gherkin permite a criação de cenários que traduzem diretamente requisitos do cliente em testes de aceitação executáveis, reduzindo ambiguidades e promovendo uma comunicação mais eficaz entre equipes de negócios e desenvolvimento [54]. Essa característica minimiza problemas de documentação incompleta ou inconsistência em requisitos, que frequentemente afetam projetos de software [14].

Além disso, o Gherkin é uma linguagem declarativa, focada em descrever o que o sistema deve fazer (comportamento esperado) em vez de como ele deve fazer (detalhes de implementação). Essa abordagem facilita a adaptação da linguagem a diferentes contextos e ferramentas de automação, como Cucumber e SpecFlow [25]. Ao abstrair os detalhes técnicos, o Gherkin promove a reutilização de cenários e evita redundâncias, contribuindo para a manutenção e escalabilidade dos testes ao longo do ciclo de vida do software [54].

Um dos principais benefícios do Gherkin é seu suporte à colaboração. Ao criar cenários que refletem as expectativas dos clientes e requisitos do sistema, a linguagem serve como um ponto de partida para discussões de refinamento e validação de requisitos. Estudos mostram que essa prática aumenta a clareza e reduz retrabalho em fases posteriores de desenvolvimento, promovendo a entrega de software que atende melhor às necessidades do cliente [14, 54].

Apesar de suas vantagens, a aplicação eficaz do Gherkin exige um entendimento claro de boas práticas, como o uso de passos independentes e a adoção de critérios de qualidade para evitar cenários redundantes ou mal definidos. Essas práticas, se implementadas corretamente, amplificam os benefícios da linguagem, garantindo sua eficácia como parte integral das práticas de BDD [25, 54].

Apesar de suas vantagens, a aplicação eficaz do Gherkin exige um entendimento claro de boas práticas, como o uso de passos independentes e a adoção de critérios de qualidade para evitar cenários redundantes ou mal definidos. Essas práticas, se implementadas corretamente, amplificam os benefícios da linguagem, garantindo sua eficácia como parte integral das práticas de BDD [25, 54].

A linguagem Gherkin organiza-se em componentes fundamentais que simplificam a especificação de requisitos e a automação de testes, promovendo um alinhamento entre equipes técnicas e não técnicas. A seção *Feature* descreve a funcionalidade geral do sistema, oferecendo um resumo do comportamento esperado, como exemplificado por "Funcionalidade de Login". Dentro de uma *Feature*, cada *Scenario* representa um caso de uso específico, detalhando interações particulares do usuário com o sistema, como o cenário "Login Bem-sucedido". Esses cenários são estruturados de maneira lógica e sequencial, permitindo uma compreensão clara do que está sendo testado [72].

Abaixo, um cenário simples em linguagem Gherkin:

Scenario: Cadastro de usuário bem-sucedido
Given o usuário acessa a página de cadastro
And preenche todos os campos obrigatórios com informações válidas
When clica no botão "Cadastrar"
Then o sistema exibe uma mensagem de sucesso "Cadastro realizado com sucesso"
And o usuário é redirecionado para a página de login

Os passos *Given*, *When* e *Then* formam a base narrativa de cada cenário em Gherkin. O passo *Given* define as condições prévias ou o estado inicial, como "o usuário está na página de login", enquanto o passo *When* descreve a ação desencadeadora, como "o usuário insere credenciais válidas". O passo *Then* especifica o resultado esperado, como "o usuário deve ser redirecionado para o painel de controle". De acordo com as práticas descritas em *The*

Cucumber Book: Behaviour-Driven Development for Testers and Developers, essa estrutura narrativa torna os cenários compreensíveis e reutilizáveis, ao mesmo tempo em que facilita sua automação [72].

Cada um desses passos segue uma lógica de narrativa que facilita o entendimento entre diferentes partes interessadas, desde desenvolvedores até analistas de negócios, e permite a automatização do teste por ferramentas compatíveis com Gherkin, como Cucumber ou SpecFlow.

2.2 Large Language Models (LLM)

Os LLM têm se consolidado como uma força motriz no campo do processamento de linguagem natural (NLP) [75]. Estes modelos, impulsionados por arquiteturas de *deep learning*, são capazes de processar e compreender vastas quantidades de dados linguísticos, identificando relações semânticas e padrões complexos [22, 47]. Radford et al. [70] destacam que a capacidade dos LLM de lidar com grandes volumes de texto não estruturado os torna ferramentas inestimáveis para uma ampla gama de tarefas de NLP.

A trajetória dos LLM é rica e diversificada. Desde os primeiros modelos de linguagem que se baseavam em técnicas estatísticas até os avançados modelos que utilizam redes neurais profundas, a evolução tem sido notável [76]. Vaswani et al. [32] foram pioneiros na introdução da arquitetura *Transformer*, que utiliza mecanismos de autoatenção, proporcionando aos LLM a capacidade de tratar eficientemente dependências de longo alcance no texto.

Modelos emblemáticos como o GPT da OpenAI e o BERT do Google, fundamentados na arquitetura *Transformer*, têm estabelecido novos padrões em várias tarefas de NLP [39, 71]. Rillig et al. [62] observam que a habilidade destes modelos de compreender contexto e semântica tem redefinido as fronteiras do que é possível no domínio do NLP.

No entanto, os LLM, apesar de suas capacidades impressionantes, apresentam desafios. A consistência e a estabilidade dos modelos, por exemplo, têm sido áreas de foco na pesquisa [39]. Bender et al. [31] salientam que, à medida que os LLM são aplicados em áreas críticas como medicina e educação, surgem questões éticas e de responsabilidade. Estas preocupações sublinham a necessidade de abordagens mais robustas e transparentes no de-

senvolvimento e aplicação desses modelos.

Bender et al. [31] enfatizam que "os modelos de linguagem são treinados em dados que refletem as desigualdades do mundo, e, portanto, podem reproduzir essas desigualdades". Esta citação destaca a preocupação emergente sobre o potencial de os LLM demonstrarem viés.

Além disso, os autores observam que "os modelos de linguagem podem gerar ou amplificar informações erradas ou enganosas" [31], ressaltando a necessidade de cautela ao aplicar LLM em domínios críticos.

Em conclusão, os LLM, com sua capacidade de processar e compreender grandes volumes de texto, representam uma revolução no campo do NLP. No entanto, é essencial abordar as preocupações éticas e técnicas associadas à sua implementação.

2.2.1 Parâmetros em LLM

Os parâmetros em LLM são elementos fundamentais que governam o processo de geração de respostas, influenciando diretamente a qualidade, a coerência e a confiabilidade das saídas produzidas. Eles atuam como mecanismos de controle, ajustando desde a aleatoriedade e a diversidade do texto gerado até a precisão e a calibração das previsões do modelo. No contexto de geração de respostas, esses parâmetros servem para equilibrar a criatividade e a relevância, adaptando-se a diferentes necessidades de tarefas e estilos de *prompts* [11, 57].

Um dos parâmetros mais utilizados é a **temperatura**, que regula a aleatoriedade das previsões. Valores mais baixos (próximos de 0) resultam em respostas mais determinísticas e conservadoras, enquanto valores mais altos aumentam a diversidade e a criatividade, podendo, no entanto, comprometer a coerência. Outros parâmetros, como *top-k* e *top-p*, limitam o número de *tokens* considerados durante a geração. O *top-k* restringe a seleção aos k *tokens* mais prováveis, enquanto o *top-p* (também conhecido como amostragem de núcleo) considera apenas os *tokens* cuja probabilidade acumulada não excede um limite p. Esses mecanismos são essenciais para evitar respostas irrelevantes ou incoerentes, garantindo que o modelo mantenha um equilíbrio entre diversidade e precisão [11, 57].

A **calibração** dos parâmetros também desempenha um papel crítico, especialmente em aplicações que exigem alta confiabilidade, como medicina e educação. A calibração refere-se ao alinhamento entre a confiança do modelo e a precisão de suas previsões. Um modelo

bem calibrado é aquele cuja confiança nas previsões reflete com precisão a probabilidade de acerto. Estudos como os de Guo et al. [29] e Jiang et al. [36] destacam que a falta de calibração pode levar a superconfiança, onde o modelo atribui probabilidades excessivamente altas a previsões incorretas, comprometendo sua aplicação em cenários críticos.

Para aprimorar a calibração, técnicas como o **Platt Scaling** têm sido empregadas. Esse método recalibra as probabilidades de saída do modelo, ajustando-as para refletir melhor a realidade. Além disso, o estudo de Xia et al. [74] propôs o framework *Calib-n*, que utiliza modelos auxiliares para agregar respostas de múltiplos LLM, melhorando a confiança e a precisão das previsões. O estudo demonstrou que a concordância entre modelos e o uso de funções de perda específicas, como a *focal loss*, são eficazes para aprimorar a calibração. Além disso, destacou que *prompts* do tipo *few-shot* são particularmente adequados para métodos baseados em modelos auxiliares, reforçando a importância da escolha do estilo de *prompt* na qualidade das respostas geradas.

No entanto, a manipulação desses parâmetros não está isenta de desafios. Zhang et al. [77] observaram que os LLM tendem a exibir superconfiança em suas previsões, especialmente em cenários de baixa acurácia. Isso ocorre porque os modelos frequentemente concentram suas probabilidades em um intervalo fixo, independentemente da complexidade do conjunto de dados. Para mitigar esse problema, além do Platt Scaling, outras abordagens têm sido exploradas, como a utilização de funções de perda que penalizam previsões excessivamente confiantes.

Em resumo, os parâmetros em LLM são ferramentas essenciais para controlar o comportamento dos modelos durante a geração de respostas. Eles permitem ajustar a criatividade, a coerência e a confiabilidade das saídas, adaptando-se a diferentes requisitos de tarefas. No entanto, a calibração inadequada e a superconfiança continuam sendo desafios significativos, exigindo técnicas avançadas e uma compreensão profunda dos mecanismos subjacentes para garantir que os LLM sejam aplicáveis e confiáveis em cenários do mundo real.

2.2.2 LLM Utilizados no Estudo

GPT-3.5 Turbo, GPT-4 Turbo e GPT-4o Mini: Desenvolvidos pela OpenAI, os modelos GPT-3.5 Turbo e GPT-4 Turbo representam avanços significativos na área de processamento de linguagem natural, oferecendo capacidades ampliadas para geração de texto, raciocínio

lógico e compreensão contextual. O GPT-3.5 Turbo foca na eficiência de processamento, enquanto o GPT-4 Turbo expande suas capacidades em tarefas mais complexas, permitindo maior precisão e flexibilidade. Ambos suportam estratégias de aprendizado como *Zero-shot*, *One-shot* e *Few-shot*, adaptando-se bem a diversos contextos de aplicação [64].

O GPT-4o Mini, embora menos conhecido, foi avaliado recentemente em tarefas de pontuação de redações no contexto educacional, mostrando limitações em relação à concordância com avaliações humanas e maior tendência a atribuir notas mais baixas. Estudos indicam que o modelo enfrenta desafios de acurácia e questões de justiça, especialmente em aplicações sensíveis, como avaliações educacionais [38]. Apesar disso, seu uso em estratégias de *Zero-shot* permite insights sobre a robustez dos modelos em cenários com dados não estruturados. No entanto, a falta de informações detalhadas sobre sua arquitetura interna limita análises mais técnicas e aprofundadas, além da aplicação em mais estudos, torna limitada a quantidade de informações acerca do modelo.

LLaMA 3: Desenvolvido pela Meta AI, o modelo LLaMA 3 apresenta avanços substanciais em escalabilidade e eficiência. Sua maior versão inclui 405 bilhões de parâmetros e suporta uma janela de contexto de até 128.000 tokens, sendo otimizado para tarefas como raciocínio, codificação e interpretação de ferramentas. Versões menores, como o LLaMA 3.1, apresentam desempenho competitivo em relação a modelos de tamanho similar devido a estratégias avançadas de pós-treinamento [28, 64]. Avaliações indicam que o modelo se compara a outras soluções de ponta, como o GPT-4, em uma ampla gama de tarefas.

Phi-3: Com 3,8 bilhões de parâmetros, o modelo Phi-3 foi projetado para alta eficiência e alinhamento com tarefas específicas. Treinado em 3,3 trilhões de tokens, ele combina compactação com desempenho próximo a modelos maiores, como o GPT-3.5. Além disso, versões derivadas, como o Phi-3.5-MoE, expandem suas capacidades multimodais, incluindo raciocínio visual e textual [1]. Sua aplicação é particularmente útil em dispositivos com recursos limitados, dada sua eficiência operacional.

Gemini 1.5 Pro: Desenvolvido pela Google DeepMind, o modelo Gemini 1.5 Pro integra texto, imagens, áudio e vídeo, demonstrando capacidades multimodais avançadas. Apesar de ser comparável ao GPT-4 em diversos domínios, estudos indicam que ele apresenta desempenho inferior em tarefas educacionais que requerem processamento detalhado de dados visuais e textuais. Contudo, sua versatilidade em múltiplas modalidades o torna uma opção

robusta para aplicações diversificadas [45].

2.2.3 Aplicações em Geração de Testes Automatizados

A aplicação de LLM na geração de testes automatizados representa um avanço significativo no desenvolvimento de software, particularmente no contexto de metodologias ágeis. Esses modelos têm demonstrado capacidade de transformar descrições textuais em cenários estruturados no formato BDD, permitindo a criação automatizada de casos de teste no padrão Given-When-Then. Essa abordagem reduz a dependência de intervenção humana e minimiza erros no processo de escrita de testes [24, 54].

LLM como os analisados nesta pesquisa utilizam técnicas para compreender requisitos textuais e convertê-los em casos de teste alinhados às especificações de aceitação. Além disso, eles possibilitam a reutilização de cenários existentes e a adaptação a novos contextos, garantindo maior eficiência em tarefas de validação automatizada. Esses recursos são especialmente úteis em projetos de grande escala ou com requisitos dinâmicos [24, 40].

Outro benefício destacado na literatura é a capacidade dos LLM de analisar cenários redundantes ou inconsistentes, aprimorando a qualidade do processo de teste. Ferramentas de avaliação são frequentemente utilizadas para medir a precisão sintática e semântica dos cenários gerados, enquanto métodos qualitativos verificam a adequação às práticas de BDD. Essa combinação de métricas quantitativas e qualitativas proporciona uma validação mais abrangente e confiável [40, 54].

Um ponto especialmente relevante é o potencial dos LLM para gerar documentação viva. A capacidade desses modelos de transformar cenários e casos de teste em uma forma compreensível para equipes técnicas e não técnicas permite que a documentação se torne um recurso dinâmico e continuamente atualizado. Essa documentação viva atua como um artefato central durante o desenvolvimento de software, facilitando a comunicação e o alinhamento entre as partes interessadas. Ao utilizar linguagem natural e formatos estruturados, os LLM garantem que requisitos e critérios de aceitação sejam claramente compreendidos e compartilhados, reduzindo ambiguidades e promovendo maior colaboração [24, 40].

Os desafios enfrentados na geração de testes automatizados com LLM incluem a adaptação a domínios específicos e a escalabilidade da solução. A necessidade de refatoração contínua dos cenários gerados também é amplamente mencionada. Apesar dessas limita-

ções, os LLM têm se mostrado eficazes em domínios variados, desde sistemas corporativos até aplicações educacionais, com potencial para serem adaptados a contextos específicos mediante ajustes nas técnicas de prompting [24, 40].

Por fim, a utilização de LLM na automação de testes acelera o desenvolvimento e melhora a comunicação entre equipes técnicas e não técnicas, pois os casos de teste gerados automaticamente no formato BDD também servem como documentação viva do sistema. Essa integração reforça a colaboração entre as partes interessadas, alinhando expectativas e reduzindo ambiguidades durante o ciclo de desenvolvimento [54].

2.2.4 Medidas de Avaliação

A avaliação de modelos de linguagem é um processo fundamental para compreender e quantificar a qualidade das respostas geradas, especialmente em tarefas que envolvem geração de texto e tradução automática. De acordo com os estudos, a complexidade dos modelos do LLM exige o uso de abordagens sistemáticas para avaliar não apenas a precisão das respostas, mas também sua coerência e relevância contextual [63, 78].

A escolha cuidadosa das medidas de avaliação é essencial para alinhar os objetivos da pesquisa aos resultados esperados. Diferentes medidas oferecem perspectivas complementares sobre o desempenho dos modelos, permitindo uma análise mais robusta e detalhada. Estudos ressaltam que avaliar modelos de LLM requer o uso de medidas que considerem tanto a similaridade semântica quanto as características quantitativas dos dados, garantindo que as análises sejam representativas da complexidade dos sistemas avaliados [40].

Além disso, avaliar a saída de LLM é particularmente desafiador devido à variabilidade das respostas possíveis e à natureza subjetiva de muitas tarefas, como a geração de texto. Por isso, a utilização de medidas que combinem análises qualitativas e quantitativas é recomendada, assegurando que aspectos como fluidez, coerência e alinhamento semântico sejam devidamente capturados. Essa abordagem promove uma avaliação mais equilibrada e permite identificar tanto as forças quanto as limitações dos modelos [78].

A escolha das medidas Distância de Manhattan, BERTScore e METEOR foi baseada em sua capacidade de cobrir diferentes aspectos da avaliação de modelos de linguagem. A Distância de Manhattan é uma medida para avaliar a proximidade entre vetores de características, sendo útil para capturar diferenças quantitativas entre textos gerados e referências.

O BERTScore, por sua vez, utiliza embeddings contextuais do BERT para medir a similaridade semântica entre textos, sendo altamente eficaz em capturar nuances linguísticas e alinhamento contextual. Já o METEOR combina precisão, recall e penalizações por desordem, sendo particularmente eficaz em tarefas de tradução e geração de texto, onde a fluidez e a coerência são críticas. Juntas, essas medidas proporcionam uma avaliação abrangente, cobrindo desde aspectos quantitativos até a qualidade semântica e estrutural das respostas geradas [44, 63, 67, 78].

2.2.4.1 Distância de Manhattan

Também conhecida como medida L1, é uma medida amplamente utilizada para calcular a distância entre dois pontos em um espaço vetorial, considerando apenas deslocamentos horizontais e verticais. Essa medida é definida como a soma das diferenças absolutas entre as coordenadas correspondentes de dois vetores. Sua simplicidade e eficiência tornam-na uma escolha comum em diversas aplicações computacionais e científicas [67].

A fórmula matemática para o cálculo da Distância de Manhattan entre dois vetores $A = (a_1, a_2, \ldots, a_n)$ e $B = (b_1, b_2, \ldots, b_n)$ é expressa como:

$$d_{\text{Manhattan}}(A, B) = \sum_{i=1}^{n} |a_i - b_i|$$

Esse cálculo é mais eficiente porque, em espaços com muitas características (alta dimensionalidade), ele evita operações complexas, como elevar números ao quadrado, que são usadas na Distância Euclidiana. Isso torna o processo mais rápido e menos custoso computacionalmente [67].

Os resultados da medida refletem diferenças quantitativas entre representações vetoriais. Em contextos como a avaliação de casos de teste, valores mais baixos de Distância de Manhattan indicam maior similaridade entre os vetores avaliados, enquanto valores mais altos refletem maior disparidade. No entanto, a medida pode ser limitada em situações onde as dimensões apresentam diferentes magnitudes ou pesos, o que pode gerar vieses na interpretação dos resultados [67].

Entre as vantagens da Distância de Manhattan estão sua robustez e simplicidade, particularmente em espaços estruturados, como matrizes e grades. Ela é menos sensível a outliers do que a Distância Euclidiana, sendo útil em aplicações como análise de agrupamentos e

redução de dimensionalidade. No entanto, ele não funciona bem em situações onde a "direção"dos dados importa, pois não leva em conta como esses dados estão alinhados ou se estão apontando para o mesmo sentido [67].

A medida tem sido amplamente utilizada em áreas como aprendizado de máquina, processamento de imagens e sistemas de recomendação. Por exemplo, em estudos de reconhecimento facial, a Distância de Manhattan foi empregada para medir a similaridade entre vetores de características extraídos por técnicas como PCA (Análise de Componentes Principais), demonstrando desempenho superior em determinadas aplicações [49].

Apesar de suas limitações, a Distância de Manhattan continua sendo uma ferramenta valiosa para avaliação e comparação em diversos contextos, desde que utilizada em situações compatíveis com suas características e restrições.

2.2.4.2 BERTScore

E uma medida de avaliação para tarefas de tradução automática e geração de texto que utiliza o modelo pré-treinado BERT para medir a similaridade semântica entre uma sentença candidata e uma sentença de referência. Em vez de depender da sobreposição de n-gramas, como medidas tradicionais (por exemplo, BLEU e METEOR), o BERTScore avalia os pares de sentenças em um espaço vetorial denso, capturando relações semânticas mais profundas [33, 63, 78].

Sejam z os vetores das representações da sentença de referência e $\hat{z}$ os vetores da sentença candidata, gerados pelo modelo BERT. A medida é baseada na similaridade por produto interno máximo, conforme descrito abaixo.

A precisão (P_{BERT}) é calculada como:

$$P_{\text{BERT}} = \frac{1}{|\hat{z}|} \sum_{\hat{z}_j \in \hat{z}} \max_{z_i \in z} z_i^\top \hat{z}_j,$$

e a revocação (R_{BERT}) é definida como:

$$R_{\text{BERT}} = \frac{1}{|z|} \sum_{z_i \in z} \max_{\hat{z}_j \in \hat{z}} z_i^\top \hat{z}_j.$$

O F_1-score é então derivado como:

$$F_{\text{BERT}} = \frac{2 \cdot P_{\text{BERT}} \cdot R_{\text{BERT}}}{P_{\text{BERT}} + R_{\text{BERT}}}.$$

Essas fórmulas demonstram como o BERTScore captura similaridades locais e globais entre a sentença candidata e a referência [78].

Possui diversas vantagens em relação a medidas tradicionais. Ele é capaz de capturar nuances semânticas e sintáticas, sendo robusto a variações lexicais, mudanças na ordem das palavras e diferenças estilísticas [63, 78]. Além disso, estudos mostram que ele apresenta maior correlação com avaliações humanas em tarefas de tradução e sumarização de texto [33, 78].

Entretanto, a medida apresenta limitações. Ele pode ser menos sensível a erros em palavras funcionais, como preposições e conjunções, e pode superestimar traduções incorretas que compartilham alta similaridade lexical com a referência [63]. Além disso, diferenças estilísticas podem levar a interpretações enviesadas, mesmo quando as traduções apresentam alta qualidade semântica [33].

Tem sido amplamente utilizado em tarefas de avaliação de tradução automática, sumarização e outras áreas relacionadas à geração de texto. Em contextos como a tradução, ele se destaca por capturar a similaridade semântica mesmo quando as sentenças apresentam diferenças lexicais significativas. A medida também tem sido aplicada para análise detalhada de traduções, demonstrando sua versatilidade e precisão [33, 78].

2.2.4.3 METEOR

METEOR (Metric for Evaluation of Translation with Explicit ORdering) foi desenvolvido como uma medida para superar limitações observadas em outras abordagens, como o BLEU, proporcionando maior correlação com julgamentos humanos. O principal objetivo do METEOR é avaliar a qualidade de traduções automáticas considerando não apenas a precisão, mas também o recall, resultando em uma medida equilibrada que reflete a qualidade geral da tradução [7, 44].

A medida realiza o alinhamento entre a tradução gerada e a tradução de referência em três etapas principais: (1) alinhamento exato, (2) aplicação de *stemming* para lidar com variações morfológicas e (3) uso de sinonímia por meio de recursos como o WordNet. A partir desses alinhamentos, calcula-se a precisão (P) e o recall (R) com as fórmulas:

$$P = \frac{m}{t}$$
$$R = \frac{m}{r}$$

onde m é o número de uniagramas alinhados, t é o total de uniagramas na tradução gerada, e r é o total de uniagramas na tradução de referência [7].

A pontuação combinada é derivada de uma média harmônica entre precisão e recall, ponderada pelo parâmetro α, que controla o peso relativo de cada componente:

$$F_{\text{mean}} = \frac{P \cdot R}{\alpha \cdot P + (1 - \alpha) \cdot R}$$

Uma característica inovadora do METEOR é a inclusão de uma penalização baseada na fragmentação da correspondência entre as palavras da hipótese e da referência. Essa penalização é calculada pela fórmula:

$$\text{Pen} = \gamma \cdot \left(\frac{\text{ch}}{m} \right)^{\beta}$$

em que ch representa o número de agrupamentos (*chunks*) consecutivos de uniagramas alinhados, γ é um parâmetro que define a penalidade máxima, e β ajusta a sensibilidade da penalidade. A pontuação final é então definida como:

$$\text{Score}_{\text{METEOR}} = (1 - \text{Pen}) \cdot F_{\text{mean}}$$

O METEOR oferece flexibilidade ao permitir o ajuste de seus parâmetros (α, β e γ), adaptando a medida a diferentes idiomas ou critérios de avaliação, como fluência e adequação. Além disso, suporta múltiplas traduções de referência, garantindo que a melhor correspondência seja utilizada para o cálculo da pontuação [44].

Essa medida se destaca por sua capacidade de capturar variações legítimas na tradução, avaliando tanto a similaridade lexical quanto a estrutura sintática. Por isso, tem sido amplamente reconhecida como uma ferramenta robusta para avaliação de sistemas de tradução automática [7, 44].

2.2.4.4 Limitações das Medidas

A distância de Manhattan, muito usada para medir diferenças entre textos, tem problemas quando os dados têm muitas características (alta dimensionalidade). À medida que o número de dimensões n aumenta, a medida se torna menos discriminativa, pois os valores da distância tendem a convergir para distribuições normais previsíveis, limitando sua capacidade de capturar nuances importantes. A dependência exclusiva de diferenças absolutas por dimensão faz com que a medida ignore dependências globais nos dados, tornando-a menos eficaz em contextos com interações complexas entre dimensões. Adicionalmente, o crescimento linear da distância com o aumento de n pode amplificar diferenças irrelevantes entre vetores quando muitos valores são pequenos ou semelhantes. Estas características indicam a necessidade de cautela ao aplicar a distância de Manhattan em contextos de alta dimensionalidade, especialmente quando nuances semânticas ou padrões globais são fundamentais para a análise [67].

O BERTScore, embora utilize embeddings contextuais para avaliar a similaridade entre textos, também apresenta limitações críticas. A medida demonstra alta sensibilidade à sobreposição lexical, atribuindo escores elevados a textos que compartilham palavras semelhantes, mesmo quando seus significados são distintos. Sua vulnerabilidade a exemplos adversariais se manifesta na preferência por sentenças com alta similaridade lexical, mas semanticamente incorretas, em detrimento de paráfrases válidas. Esta limitação está intrinsecamente relacionada ao problema de antônimos, onde palavras com significados opostos, como "bom"e "ruim", apresentam alta similaridade no espaço vetorial devido ao contexto compartilhado, tornando a medida insensível a mudanças semânticas fundamentais. Em aplicações multilíngues, a medida enfrenta desafios adicionais devido ao desalinhamento dos subespaços linguísticos no modelo BERT multilingue, comprometendo sua eficácia na avaliação de traduções. O BERTScore também apresenta dificuldades significativas ao lidar com traduções ou textos de baixa qualidade gramatical, afetando sua robustez em cenários realistas e adversariais [41].

A medida METEOR, apesar de incorporar informações semânticas através da consideração de sinônimos e raízes de palavras, apresenta limitações significativas em sua aplicação. Uma de suas principais deficiências é a permissividade com erros funcionais, particularmente

na ausência de marcadores de negação, que podem alterar substancialmente o significado emocional de uma frase. Por exemplo, traduções que omitem uma negação crucial recebem pontuações apenas ligeiramente inferiores às traduções corretas, evidenciando a inadequada penalização de erros sentimentais críticos pela medida. Além disso, o METEOR demonstra notável dificuldade em distinguir entre traduções que preservam o conteúdo emocional e aquelas que apresentam distorções severas, comprometendo sua precisão em contextos sensíveis, como análise de sentimentos ou avaliação de textos emocionais [63].

Capítulo 3

Validação da Medida

Neste capítulo, são abordadas três atividades principais. Primeiramente, são apresentadas as boas práticas de BDD que orientaram a construção dos *prompts* utilizados no experimento, assegurando-se a aderência às recomendações da literatura. Em seguida, é descrita a base de dados utilizada, composta por casos de teste provenientes de um estudo prévio[21], e o processo de construção da base de referência (*ground truth*), desenvolvido por um especialista. Por fim, é detalhado o processo de validação da medida selecionada para avaliar os cenários gerados.

3.1 Construção do *Prompt* Baseado em Boas Práticas do BDD

Os *prompts* desenvolvidos neste estudo foram estruturados com base nas boas práticas recomendadas por Oliveira et al.[54]. No estudo, boas práticas de qualidade para cenários BDD eficazes foram identificadas por meio de entrevistas com profissionais da área, e uma lista de verificação baseada em perguntas foi criada para auxiliar na avaliação desses cenários. Essas práticas são direcionadas a garantir a clareza, consistência e qualidade nos cenários BDD gerados, de forma a alinhá-los às expectativas de desenvolvedores e partes interessadas.

As recomendações apresentadas a seguir foram identificadas como boas práticas para a escrita de cenários BDD, com o objetivo de assegurar clareza, consistência e alinhamento com os objetivos do negócio na criação de bons cenários de teste. Na Tabela 3.1, essas

diretrizes são resumidas, destacando os principais aspectos a serem considerados durante o desenvolvimento dos cenários.

Tabela 3.1: Recomendações Identificadas para Cenários BDD

Recomendação	**Descrição**
Clareza no Valor de Negócio	Certificar-se de que o valor de negócio ou resultado esperado seja explicitamente declarado em cada cenário.
Foco em Uma Ação e Resultado	Cada cenário deve concentrar-se em uma única ação do usuário e seu respectivo resultado, evitando múltiplas interpretações ou ações combinadas.
Uso de Linguagem Declarativa	Adotar uma linguagem que descreva o comportamento esperado, ao invés de detalhes de implementação técnica.
Consistência Terminológica	Utilizar uma terminologia de negócios consistente, eliminando o uso de jargões técnicos.
Terceira Pessoa como Ponto de Vista	Garantir que os passos sejam escritos na terceira pessoa, promovendo um foco neutro e profissional.

Essas orientações foram incorporadas na formulação dos *prompts*, que foram utilizados para conduzir a transformação de cenários de teste livres em cenários BDD no formato Gherkin. Três abordagens distintas de *prompting* foram empregadas (*zero-shot*, *one-shot* e *few-shot*), sendo os exemplos fornecidos e as instruções adaptados para que a capacidade dos LLM na geração de cenários consistentes e alinhados às práticas descritas fosse explorada.

A importância de um design de *prompt* foi destacada pelos resultados obtidos, demonstrando-se que a aplicação das boas práticas pode influenciar diretamente a qualidade dos cenários gerados. A contribuição prática do estudo é reforçada nesta seção ao serem integradas diretrizes teóricas com a utilização de LLM no contexto da engenharia de software.

3.1.1 Técnicas de *Prompting* Utilizadas

Cada técnica foi estruturada para que os LLM fossem orientados na geração de cenários BDD de maneira clara, precisa e aderente à sintaxe do Gherkin.

Na técnica *zero-shot*, apenas uma instrução detalhada é apresentada no *prompt*, solicitando que a conversão direta de uma descrição de caso de teste em um único cenário BDD seja realizada, sem a inclusão de exemplos prévios. A estrutura padrão do Gherkin é reforçada nas instruções, com a utilização de palavras-chave em inglês e passos descritivos em português, destacando-se a necessidade de clareza e concisão na formulação dos cenários.

Na técnica *one-shot*, um único exemplo de cenário BDD é incluído no *prompt* antes da apresentação da descrição do caso de teste a ser convertido. Esse exemplo é utilizado como guia para que o modelo seja demonstrado na formatação esperada e na aplicação das boas práticas do BDD, como a independência dos cenários e o uso de terminologia consistente. A geração de cenários mais alinhados ao padrão esperado pelos LLM é auxiliada por essa inclusão.

Por fim, na técnica *few-shot*, múltiplos exemplos de cenários BDD são utilizados no início do *prompt*, abrangendo diferentes casos de uso. Nuances importantes da sintaxe do Gherkin, bem como variações no formato e na aplicação das boas práticas, são ilustradas por esses exemplos. Essa abordagem busca que um contexto mais amplo seja fornecido ao modelo, permitindo que ele se adapte melhor às descrições de casos de teste apresentadas, especialmente em tarefas mais complexas ou específicas.

Adicionalmente, as técnicas de *zero-shot*, *one-shot* e *few-shot* apresentam limitações significativas, principalmente em tarefas que exigem raciocínio complexo ou conhecimento específico de domínio. O *zero-shot* depende fortemente da qualidade das descrições semânticas e pode falhar em cenários onde a generalização é necessária para classes não vistas, especialmente em tarefas de reconhecimento visual ou tradução automática [60]. Já o *one-shot* é altamente sensível à escolha do exemplo fornecido, podendo superajustar-se a ele e falhar em generalizar para casos semelhantes, como em reconhecimento de ações ou segmentação de imagens [26]. Por fim, o *few-shot*, embora mais robusto, ainda enfrenta desafios de escalabilidade e custo computacional, além de ser limitado pela seleção de exemplos representativos, o que pode comprometer seu desempenho em tarefas de detecção de objetos ou classificação

de texto [13]. Essas limitações destacam a necessidade de melhorias no design de *prompts* e na seleção de exemplos para maximizar a eficácia dessas técnicas.

O prompt apresentado a seguir foi utilizado para a geração de cenários BDD na técnica *zero-shot*. Ele define as instruções detalhadas para a conversão de descrições de cenários de teste em cenários BDD no formato Gherkin, garantindo aderência às boas práticas e consistência na saída gerada pelos modelos. Os prompts utilizados nas técnicas *one-shot* e *few-shot* estão disponibilizados no Apêndice A para consulta.

Prompt Template

Converta a seguinte descrição de caso de teste em um único cenário BDD, usando a sintaxe estrita de Gherkin. Certifique-se de que a saída contenha apenas a sintaxe de Gherkin para o cenário, sem comentários, explicações ou a palavra "Feature". Use o português para as descrições dos casos de teste e detalhes do cenário, mas mantenha as palavras-chave do Gherkin em inglês.

Agora, converta a seguinte descrição de caso de teste em exatamente um cenário BDD usando a sintaxe estrita de Gherkin. A saída deve seguir exatamente o formato do exemplo fornecido e não conter nada além do cenário BDD. Somente as palavras-chave do Gherkin devem estar em inglês; todo o outro texto deve estar em português.

Descrição do Caso de Teste: {test_case_description}

Para um bom cenário BDD, certifique-se de declarar claramente o valor de negócio ou resultado esperado e mantenha o foco em uma única ação e seu resultado. Use apenas os passos essenciais (Given, When, Then, And) de forma clara e declarativa, evitando detalhes de implementação e repetições desnecessárias. Garanta que os cenários sejam independentes, utilizem uma terminologia de negócios consistente sem jargões técnicos e que os passos sejam escritos em terceira pessoa para evitar múltiplas interpretações.

Certifique-se de que os cenários BDD tenham indentação consistente de dois espaços para cada passo sob 'Scenario', sem linhas em branco entre os passos, e uma linha em branco separando diferentes cenários.

Siga esta estrutura para a saída:

Scenario: [Descrição do Cenário]

    Given [algum contexto inicial]

    And [mais algum contexto, se houver]

    When [uma ação é realizada]

    Then [um conjunto específico de resultados deve ocorrer]

    And [outro resultado, se houver]

A resposta deve conter estritamente apenas a sintaxe válida de Gherkin e evitar qualquer informação ou comentário extra. A resposta deve conter apenas o texto BDD, sem formatação ou texto adicional (por exemplo, sem 'gherkin"'), e deve ser apenas um cenário BDD.

Cada estrutura de *prompt* foi projetada para que fosse avaliado como o nível de contexto e os exemplos fornecidos impactam a eficácia e a consistência dos cenários BDD gerados pelos LLM. As técnicas foram aplicadas de forma comparativa para que fosse determinada

sua influência na qualidade das saídas geradas.

Os outros *prompts* completos utilizados em cada técnica de *prompting* estão disponíveis no Apêndice A, onde detalhes adicionais sobre as instruções e exemplos utilizados no estudo são fornecidos.

3.1.2 Scripts

A implementação foi conduzida com a criação de scripts específicos, a configuração das APIs para que a interação com os modelos fosse realizada, e o uso de ferramentas como o Google Colab[1] para execução e processamento.

Os scripts [2] foram desenvolvidos em Python, com bibliotecas como `OpenAi`, `Replicate` e `GenerativeAi` da Google sendo utilizadas, de acordo com o modelo empregado. Cada script foi configurado para que fossem implementadas técnicas específicas de *prompting*, como *zero-shot*, *one-shot* e *few-shot*, com *prompts* projetados para guiar os modelos na geração de cenários BDD.

O escopo do script inclui funções para:

- Gerar cenários BDD baseados em descrições de cenários de teste, seguindo a sintaxe do Gherkin.
- Configurar parâmetros do modelo, como temperatura, penalidades de repetição e limites de tokens.
- Salvar os cenários gerados em arquivos no formato `.feature`, organizados por execução e caso.

3.1.3 APIs e Modelos Utilizados

A integração com os modelos foi realizada por meio de APIs específicas:

[1] Ferramenta de desenvolvimento colaborativo baseada em nuvem, amplamente utilizada para experimentos em aprendizado de máquina e processamento de linguagem natural. Disponível em: `https://colab.research.google.com/`.

[2] Todos os scripts utilizados para geração dos cenários estão no github. Para acessar: `https://github.com/hiagonfs/bdd-scenario-evaluation`

- **OpenAI API**[3]: Utilizada para interagir com modelos como `GPT-3.5`, `GPT-4` e `GPT-4o-mini` configurados para diferentes técnicas de *prompting*.
- **Replicate API**[4]: Implementada para executar modelos alternativos como `Phi-3` e `Llama 3`, configurados com parâmetros ajustados para controle da qualidade e diversidade das respostas.
- **Google Generative AI API**[5]: Integrada para explorar o modelo `Gemini 1.5 Pro`, com configurações customizadas de geração e suporte a interações específicas via *chat sessions*.

Parâmetros como *penalidade de presença* e *temperatura moderada* foram utilizados nas chamadas às APIs para que respostas repetitivas ou inconsistentes fossem evitadas. Cada execução teve suas saídas organizadas em pastas específicas, contendo informações acerca do modelo e da técnica aplicados.

Em contraste com a busca por consistência através do ajuste de parâmetros, a utilização do parâmetro `SEED` não foi explorada neste estudo, considerando-se que sua aplicabilidade está limitada a modelos como os GPTs à época de execução do estudo. Mesmo nesses casos, ele não elimina completamente a variabilidade nas respostas geradas. Conforme evidenciado no estudo de Blackwell et al. [10], os modelos de linguagem são inerentemente estocásticos, e, mesmo com configurações como temperatura zero e `SEED` fixado, respostas absolutamente determinísticas não podem ser obtidas devido à natureza probabilística dos modelos e à execução paralela em sistemas distribuídos. Por esse motivo, e em linha com os objetivos de garantir a consistência das respostas, o ajuste de parâmetros como `temperature`, `top-p` e `frequency penalty` foi priorizado, buscando-se um equilíbrio entre consistência e criatividade nas saídas geradas.

3.1.4 Configuração dos Parâmetros por Modelo

Os modelos empregados nos experimentos foram:

[3]Mais informações sobre a OpenAI API podem ser encontradas em: `https://openai.com/index/openai-api/`

[4]Mais informações sobre a Replicate API podem ser encontradas em: `https://replicate.com/`

[5]Mais informações sobre a Google Generative AI API podem ser encontradas em: `https://aistudio.google.com/`

- **GPT-3.5 Turbo, GPT-4 Turbo e GPT-4o Mini**: Implementados via API da OpenAI, configurados com os seguintes parâmetros: temperatura de 0,5, penalização de frequência e presença de 1,15, e limite de 2.048 tokens por resposta.

- **LLaMA 3**: Utilizado por meio da plataforma Replicate, configurado com `top_k` ajustado para 40, `top_p` de 0,8 e uma penalização leve de comprimento para respostas mais concisas.

- **Phi-3**: Configurado pela API Replicate, este modelo foi ajustado com configurações semelhantes ao LLaMA 3.

- **Gemini 1.5 Pro**: Implementado via API do Google Generative AI, com temperatura de 0,5, `top_p` de 0,8 e capacidade de gerar até 2.048 tokens por resposta.

No estudo, a definição dos parâmetros e configurações para os modelos de linguagem foi realizada de forma empírica. Os valores para temperatura, penalização de frequência e presença, `top_k`, `top_p` e limite de *tokens* foram escolhidos com base em experimentação e observação, sendo ajustados de acordo com o caso de uso específico e os objetivos do experimento.

A justificativa para essa abordagem empírica está na natureza complexa e estocástica dos LLM, bem como na dificuldade de determinar os valores ideais para esses parâmetros de forma teórica. A experimentação e o ajuste fino foram realizados para que a combinação de valores que produzisse os melhores resultados para cada situação fosse encontrada.

3.1.5 Técnicas Utilizadas

Os *prompts* utilizados para a geração de casos BDD foram projetados para que as especificidades de cada técnica (*zero-shot*, *one-shot* e *few-shot*) fossem atendidas, garantindo que a tarefa de transformar casos de teste livres em cenários estruturados no formato Gherkin fosse compreendida pelos LLM. Em cada configuração, os *prompts* foram ajustados para que instruções claras e contexto suficiente fossem fornecidos, adaptando-se ao nível de dependência de exemplos.

- ***Zero-shot learning***: Nesta técnica, o modelo foi instruído a realizar a tarefa diretamente, sem exemplos fornecidos. O *prompt* descrevia a tarefa de forma objetiva,

explicando que o modelo deveria transformar um caso de teste livre em um cenário no formato BDD, utilizando as palavras-chave "Feature", "Scenario", "Given", "When"e "Then". Adicionalmente, eram incluídas diretrizes para respeitar a sintaxe do Gherkin e capturar aspectos funcionais como pré-condições, ações do usuário e resultados esperados.

- ***One-shot learning***: Para esta técnica, além das instruções gerais descritas no *zero-shot*, foi incluído um exemplo completo de caso de teste livre transformado em cenário no formato Gherkin. Esse exemplo serviu como referência para o modelo, ilustrando o padrão esperado e auxiliando no alinhamento semântico e estrutural. O exemplo foi cuidadosamente selecionado para representar um caso comum e genérico, maximizando a aplicabilidade a outros cenários.

- ***Few-shot learning***: Nesta abordagem, o *prompt* incluía três exemplos de transformação de casos de teste livres para o formato Gherkin, além das instruções gerais. Os exemplos foram selecionados para abranger diferentes tipos de funcionalidades (ex.: autenticação, cadastro, navegação) e demonstrar variações na estrutura de cenários. Essa técnica foi utilizada para explorar a capacidade do modelo de generalizar padrões a partir de múltiplas referências.

A principal diferença entre os *prompts* utilizados para as técnicas *zero-shot*, *one-shot* e *few-shot* foi definida pelo número de exemplos fornecidos para que o modelo fosse guiado na geração dos cenários de teste BDD. Enquanto na técnica *zero-shot* nenhum exemplo foi apresentado, um e três exemplos foram incluídos, respectivamente, nas técnicas *one-shot* e *few-shot*. No entanto, as instruções básicas foram mantidas em todas as técnicas, com a criação de cenários de teste alinhados à sintaxe Gherkin e às boas práticas do BDD sendo enfatizada.

3.2 Base de Dados

A base de dados[6] utilizada neste trabalho foi coletada a partir do estudo intitulado *Uma Abordagem para Recomendação de Cenários de Teste em Projetos Ágeis Baseados no Scrum* [21]. Neste estudo, uma abordagem foi proposta para que cenários de teste em projetos ágeis fossem recomendados, utilizando dados históricos para melhorar a eficiência e a qualidade das suítes de teste.

Essa base foi extraída dos scripts de população de banco de dados empregados no estudo [21]. A extração dos cenários de teste desses scripts foi realizada e, posteriormente, os dados foram convertidos para o formato CSV, facilitando sua manipulação e análise. A base reúne 1.286 cenários provenientes de sistemas corporativos, nos quais valores padrão, como `"Imported from sheet"`, foram identificados em campos como `purpose`, `precondition` e `steps`, indicando limitações na documentação original. Apesar dessas limitações, o objetivo deste trabalho não foi comprometido, pois os nomes dos cenários de teste permanecem descritivos e informativos, abrangendo funcionalidades críticas, como autenticação, controle de acesso, gerenciamento de usuários e operações CRUD.

A rastreabilidade entre os cenários de teste e seus requisitos de negócio foi estabelecida por meio de identificadores que relacionam os cenários a 16 diferentes projetos. Essa relação foi implementada pelos campos `project_id`, `user_story_id` e `reused_id`, facilitando a vinculação dos testes aos seus respectivos contextos e requisitos. Além disso, uma uniformidade na classificação foi observada: todos os cenários possuem prioridade `"M"` e são executados manualmente, mantendo uma estrutura padronizada que inclui os seguintes atributos:

Os principais atributos da base de dados são descritos a seguir:

- **id**: Identificador único do caso de teste.

- **name**: Nome descritivo do caso.

- **purpose**: Objetivo do teste.

- **precondition**: Condições necessárias para execução.

- **execution_type**: Tipo de execução (manual/automatizado).

[6]Disponível em: `https://github.com/hiagonfs/bdd-scenario-evaluation/blob/main/csv_files/test_case.csv`.

- **steps**: Sequência de passos para execução.

- **project_id**: Identificador do projeto associado.

- **user_story_id**: História de usuário relacionada.

- **priority**: Prioridade do caso.

- **reused_id**: Identificador de caso reusado.

- **active**: Status de ativação do caso.

Abaixo um exemplo retirado da base de dados, ilustrando a estrutura uniforme e a padronização dos campos:

- **id:** 1
  name: Cadastrar Entidade com sucesso
  purpose: Imported from sheet
  precondition: Imported from sheet
  execution_type: M
  steps: Imported from sheet
  project_id: 1
  user_story_id: 1
  priority: M
  reused_id: -
  active: True

A uniformidade estrutural da base de dados é destacada por esses exemplos, com a padronização dos campos e a consistência dos identificadores sendo evidenciadas. Apesar de limitações provenientes da coleta inicial, como os valores padrão `"Imported from sheet"`, essas falhas não impactaram significativamente as análises realizadas neste estudo, garantindo que suporte adequado fosse fornecido para a avaliação e adaptação dos cenários de teste às metodologias exploradas.

3.2.1 Seleção de Cenários

A seleção dos cenários de teste foi conduzida com base em critérios estabelecidos para que a relevância, representatividade, qualidade e imparcialidade do conjunto de dados utilizado

no experimento fossem garantidas. Os critérios seguidos foram os seguintes:

- **Remoção de descrições curtas e prefixos não informativos**: cenários de teste cujas descrições possuíam três palavras ou menos foram descartados, devido à sua provável insuficiência para representar um cenário de teste significativo. Além disso, cenários que continham prefixos não informativos, como "[SPEC]", "HWLL-28:"ou nomenclaturas similares, foram removidos. Esses prefixos eram geralmente utilizados para identificação interna, mas não contribuíam para o entendimento do caso de teste, comprometendo a utilidade dos mesmos no contexto do experimento.

- **Priorização de funcionalidades críticas**: A priorização foi realizada com foco em identificar cenários relacionados às funcionalidades mais relevantes e críticas do sistema. Essas funcionalidades incluem operações essenciais, como login, cadastro, autenticação, recuperação de senha e edição de perfis de usuário, as quais são amplamente reconhecidas como fundamentais em sistemas modernos devido à sua importância tanto para a experiência do usuário quanto para a segurança e integridade do sistema [4, 66]. A seleção baseou-se em palavras-chave específicas, como "login", "senha", "autenticação"e "cadastro", permitindo destacar cenários de teste que verificassem diretamente aspectos cruciais de confiabilidade, usabilidade e proteção de dados. Essa abordagem garantiu que os cenários priorizados refletissem cenários de uso de alta frequência e alto impacto, alinhando-se às melhores práticas recomendadas na literatura para validação de requisitos funcionais e não funcionais [66].

- **Exclusão de redundâncias por similaridade semântica**: Para evitar a repetição de cenários semelhantes, foi realizada a análise de similaridade semântica entre as descrições dos cenários de teste. Utilizou-se um modelo de linguagem robusto (*all-mpnet-base-v2*) para calcular a similaridade de cosseno entre os cenários, com um limiar de 97% para detecção de sinônimos. Quando cenários de teste redundantes foram identificados, apenas uma descrição representativa foi mantida, garantindo que o conjunto final fosse diversificado e livre de duplicatas.

Após a aplicação dos critérios descritos, o processo de filtragem resultou em um con-

junto de 105 cenários finais[7], dos quais 10 foram selecionados aleatoriamente[8] para serem utilizados no experimento principal, que consistiu em sua transformação em cenários BDD. Esses cenários representam domínios variados e relevantes, como validação de senhas com regras específicas, operações de cadastro e cancelamento, ações realizadas sem autenticação, edições de dados de perfil ou reservas, além de inconsistências como senhas divergentes ou datas inválidas. Essa seleção foi estratégica para garantir que os cenários escolhidos cobrissem cenários amplamente representativos das funcionalidades críticas de um sistema.

3.2.2 Construção da Base de Referência

A criação do *ground truth* foi realizada com o objetivo de que uma base de referência confiável para a avaliação do desempenho dos LLM na geração de cenários BDD fosse estabelecida. Esse processo envolveu a transformação de cenários de teste selecionados em cenários BDD, conduzida por um especialista para que servisse como referência na avaliação dos casos gerados.

A base de cenários transformados foi construída com atenção às boas práticas, constituindo um *ground truth* para a avaliação comparativa dos modelos de linguagem. Inicialmente, os cenários foram gerados por um especialista e, em seguida, submetidos à avaliação de outros dois especialistas. Cada cenário foi analisado, as considerações foram compartilhadas e um consenso sobre a versão final foi alcançado, garantindo a qualidade e a confiabilidade do *ground truth*. Essa base é utilizada como referência nos experimentos, permitindo que a eficácia dos LLM na geração de cenários e a qualidade dos cenários gerados automaticamente sejam avaliadas.

Abaixo, seguem alguns exemplos dos casos que foram gerados a partir dos cenários de base:

[7]Os cenários de teste finais, após a aplicação dos filtros, podem ser acessados em: `https://github.com/hiagonfs/bdd-scenario-evaluation/blob/main/csv_files/casos_de_teste_filtrados_final.csv`

[8]Os 10 cenários selecionados aleatoriamente após o processo de filtragem estão disponíveis em: `https://github.com/hiagonfs/bdd-scenario-evaluation/blob/main/csv_files/casos_selecionados.csv`

Cenário Original: Validação da regra de senha

Cenário BDD Transformado:

Scenario: Validação da regra de senha

Given uma regra de senha definida

When um usuário informa uma senha

Then o sistema deve validar a senha de acordo com a regra

And o sistema deve informar ao usuário se a senha é válida ou não

And o sistema deve exibir uma mensagem de erro caso a senha seja inválida

Cenário Original: Executar ação sem login

Cenário BDD Transformado:

Scenario: Executar ação sem login

Given um usuário não autenticado

When o usuário tenta executar uma ação que requer autenticação

Then o sistema deve impedir o usuário de executar a ação

And o sistema deve redirecionar o usuário para a página de login

Perfil dos Especialistas

A base de referência utilizada neste estudo foi elaborada por um especialista com experiência em escrita de casos de teste, incluindo a aplicação prática de BDD. O profissional possui 18 anos de experiência na área de testes, dos quais 14 anos foram dedicados a testes mobile, 2 anos a e-commerce e 2 anos a instituições bancárias. Além disso, sua atuação abrange a escrita e execução de cenários BDD, bem como a realização de testes guiados por esses cenários, destacando sua expertise em alinhar especificações com práticas de automação e validação de software. Há 5 anos, o especialista utiliza BDD de forma rotineira.

A revisão da base de referência foi conduzida por dois especialistas com perfis complementares, ambos com experiência no uso rotineiro de BDD em suas atividades profissionais. O primeiro especialista, com formação em Ciência da Computação e mestrado em Engenharia de Software, possui 5 anos de experiência em tecnologia e testes de software, incluindo atuação em projetos de NLP, validação de sistemas biométricos, como reconhecimento fa-

cial e impressões digitais, e testes de aplicações móveis para plataformas Android e iOS. O segundo especialista, graduado em Ciência da Computação, conta com 8 anos de experiência no mercado, com foco em testes de software, pesquisa e automação, abrangendo a automação de testes para os setores de e-commerce e de automotivos.

3.3 Desenvolvimento Experimental

Os experimentos foram conduzidos utilizando o ambiente Google Colab, com scripts implementados em Python, explorando-se sua versatilidade para automação e manipulação de dados. A integração com os LLM foi realizada por meio de APIs oficiais, permitindo que o envio de solicitações e o processamento das respostas fossem executados de forma eficiente. A coleta dos dados gerados foi efetuada diretamente a partir da execução desses scripts, que foram configurados para testar diferentes modelos e configurações de *prompts*. Todo o processo foi projetado para que a validade dos dados gerados e a reprodutibilidade dos experimentos fossem garantidas, estabelecendo-se uma base confiável para as análises subsequentes.

3.3.1 Execução e Coleta de Dados

As execuções foram realizadas no ambiente Google Colab. Cada execução processou um conjunto de descrições de cenários de teste e gerava múltiplos cenários BDD, com tentativas limitadas para garantir resultados válidos. Os dados coletados foram salvos em diretórios organizados, contendo:

- Arquivos de texto com os cenários gerados.
- Um arquivo de texto com informações sobre o modelo, técnica utilizada e horário de execução.
- Logs de execução para monitorar possíveis erros ou inconsistências durante o processo.

Essa abordagem garantiu a reprodutibilidade e organização dos experimentos, permitindo a análise detalhada dos resultados e sua integração com as demais etapas do estudo.

3.3.1.1 Cenários gerados por cada modelo e técnica

A análise estatística foi conduzida com base nos resultados obtidos a partir dos 6 modelos selecionados, sendo cada modelo avaliado utilizando três técnicas de prompt: *zero-shot*, *one-shot* e *few-shot*. Para cada técnica, foram gerados 10 cenários de teste no formato BDD a partir de 10 cenários selecionados aleatoriamente. Este processo foi repetido em 5 execuções independentes para cada técnica, totalizando 50 cenários de teste gerados por técnica e 150 cenários por modelo. Ao todo foram gerados 900 cenários a partir da transformação.

Tabela 3.2: Distribuição de Cenários por Técnica e Modelo

Modelo	**Zero-shot**	**One-shot**	**Few-shot**	**Total**
GPT-3.5 Turbo	50	50	50	150
GPT-4 Turbo	50	50	50	150
GPT-4o Mini	50	50	50	150
LLaMA 3	50	50	50	150
Phi-3	50	50	50	150
Gemini 1.5 Pro	50	50	50	150
Total Geral	**300**	**300**	**300**	**900**

Após a geração dos cenários, as medidas de avaliação foram calculadas para cada medida candidata. Os resultados obtidos foram utilizados para ranquear as medidas, permitindo comparar seu desempenho e adequação ao contexto da validação de cenários de teste em BDD. Mais detalhes sobre o processo de validação das medidas estão descritos na Seção 3.4.

Os LLM avaliados neste estudo, detalhados na Tabela 3.3, incluíram versões compactas, como o *Phi-3 Mini*, e robustas, como o *Meta LLaMA 3* com 70 bilhões de parâmetros. Isso garantiu uma análise do desempenho e da variabilidade entre diferentes configurações e arquiteturas.

Tabela 3.3: Modelos Avaliados e Suas Versões

Modelo	**Versão**
GPT-3.5 Turbo	gpt-3.5-turbo-0125
GPT-4o Mini	gpt-4o-mini
GPT-4 Turbo	gpt-4-turbo
Gemini	gemini-1.5-pro
LLaMA 3	meta-llama-3-70b-instruct
Phi-3 Mini	phi-3-mini-128k-instruct

3.3.1.2 Custos e Limitações

Custos relacionados ao uso das APIs dos modelos foram apresentados durante a execução dos experimentos, variando conforme a plataforma e o modelo utilizado. Os principais custos foram associados à OpenAI e à plataforma Replicate, enquanto o modelo Gemini foi executado utilizando-se créditos de teste gratuitos fornecidos pelo Google Generative AI. A seguir, os custos específicos são detalhados:

- **GPT-3.5 Turbo e GPT-4 Turbo (OpenAI)**: O custo total para a execução desses modelos foi de $9.14, considerando a quantidade de chamadas realizadas e os tokens processados durante os experimentos.
- **LLaMA 3 e Phi-3 (Replicate)**: Os modelos executados na plataforma Replicate apresentaram um custo total de $7.28, devido ao uso de recursos otimizados para processamento de modelos grandes.
- **Gemini (Google Generative AI)**: Este modelo foi executado utilizando créditos de teste gratuitos, com um limite equivalente a R$94, o que permitiu realizar os experimentos sem custos adicionais.

Apesar dessas limitações, os experimentos foram conduzidos com alta confiabilidade, sem que interrupções críticas comprometessem a obtenção dos resultados esperados.

3.4 Seleção de Medida para Avaliação de Cenários BDD

A escolha de medidas adequadas para avaliação foi fundamental para que os resultados obtidos fossem relevantes, consistentes e alinhados aos objetivos da pesquisa. Cada medida foi caracterizada por suas especificidades, capazes de capturar diferentes aspectos dos dados analisados, como similaridade estrutural, semântica ou textual. Todas as medidas foram utilizadas para que a similaridade semântica entre os cenários de teste gerados pelos modelos e os cenários de referência (*ground truth*) elaborados pelo especialista fosse avaliada. Dessa forma, a seleção cuidadosa de medidas permitiu que uma análise direcionada fosse realizada, redundâncias fossem evitadas e a interpretação fosse facilitada, promovendo-se a confiabilidade dos resultados no contexto do estudo.

3.4.1 Medidas Candidatas

No contexto deste trabalho, a avaliação dos cenários BDD gerados pelos LLM em comparação com os cenários de referência foi realizada utilizando-se medidas capazes de capturar diferentes aspectos de qualidade, como discrepância estrutural, similaridade semântica e precisão textual. Para isso, três medidas amplamente reconhecidas foram selecionadas para que essas necessidades específicas fossem atendidas: Distância de Manhattan, METEOR e BERTScore.

A seleção dessas medidas foi baseada em sua capacidade de avaliar dimensões distintas da qualidade dos cenários gerados. A Distância de Manhattan foi considerada eficaz para que discrepâncias estruturais fossem capturadas, sendo particularmente adequada para medir diferenças na organização textual de cenários no formato Gherkin de Carvalho Moraes et al. [20]. Além disso, a Distância de Manhattan é amplamente reconhecida por sua relevância em contextos onde a organização estrutural dos dados desempenha um papel crítico, sendo frequentemente apontada como uma medida confiável para capturar discrepâncias em sistemas baseados em dados categóricos ou estruturados [61]. O METEOR destacou-se por considerar sinônimos e flexões, permitindo que a similaridade semântica fosse avaliada de forma mais robusta, como evidenciado no estudo de Zhang et al. [78]. Por fim, o BERTScore ofereceu uma abordagem baseada em embeddings contextuais, permitindo que nuances semânticas

profundas e relações contextuais fossem capturadas, conforme detalhado por Zhang et al. [78]. Essas características tornaram essas medidas complementares e altamente adequadas para o contexto desta pesquisa, cobrindo tanto aspectos estruturais quanto semânticos dos cenários gerados.

3.4.1.1 Processamento da Distância de Manhattan

O cálculo foi implementado utilizando-se a biblioteca `scipy`, que fornece a funcionalidade necessária para que a distância entre dois vetores numéricos fosse calculada. A seguir, o fluxo principal realizado no script é descrito:

1. **Preparação dos Dados:** Inicialmente, os cenários de referência (*ground truth*) foram descompactados e convertidos para o formato `.feature`. Em paralelo, os cenários gerados pelos modelos foram organizados em pastas de execuções (`exec_1` a `exec_5`), representando-se as diferentes iterações dos modelos para cada técnica de prompt (*zero-shot*, *one-shot* e *few-shot*).

2. **Fluxo de Comparação:** Para cada cenário base presente no *ground truth*, o script foi configurado para iterar sobre as cinco execuções (`exec_1` a `exec_5`) de um modelo específico. Cada par de textos (cenário base e cenário gerado) foi submetido a uma transformação e comparação.

3. **Transformação para Vetores:** Ambos os textos — o cenário base e o cenário gerado — foram convertidos em vetores numéricos, com o método de frequência de palavras (*bag-of-words*) sendo implementado por meio da funcionalidade `CountVectorizer` da biblioteca `scikit-learn`. Cada posição do vetor foi configurada para representar a frequência de uma palavra específica no texto.

4. **Cálculo da Distância de Manhattan:** A distância de Manhattan foi calculada entre os vetores gerados, com a função `cityblock` da biblioteca `scipy` sendo utilizada. Essa etapa produziu, para cada par (cenário base, cenário gerado), um valor que quantificou a diferença estrutural entre os textos. Valores menores foram associados a maior similaridade estrutural.

5. **Organização dos Resultados:** Para cada cenário base, as distâncias calculadas para as cinco execuções foram organizadas em ordem crescente, formando-se um ranking. O menor valor de distância foi identificado como o cenário gerado mais semelhante ao cenário de referência. Os resultados finais foram armazenados em um arquivo CSV, permitindo-se uma análise posterior.

Esse fluxo foi repetido para que cada modelo avaliado fosse testado utilizando os três tipos de prompts (*zero-shot*, *one-shot* e *few-shot*), permitindo que uma análise detalhada da similaridade estrutural dos cenários gerados em relação ao *ground truth* fosse realizada.

3.4.1.2 Processamento do BERTScore

Baseia-se na comparação de embeddings contextuais das palavras, com modelos de linguagem pré-treinados sendo utilizados. O processo foi implementado com a biblioteca `bert-score`, que automatiza o cálculo das pontuações de precisão (P), revocação (R) e $F1$, sendo este último utilizado como medida principal.

A seguir, o fluxo principal do cálculo realizado no script é descrito:

1. **Preparação dos Dados:**
   - Os cenários de referência foram descompactados e convertidos para o formato `.feature`, sendo armazenados no diretório `ground_truth`.
   - Os cenários gerados pelos modelos foram organizados em pastas (`exec_1` a `exec_5`), cada uma representando diferentes rodadas de geração com as técnicas de *prompt* (*zero-shot*, *one-shot* e *few-shot*).
2. **Comparação entre cenários:**
   - Para cada cenário base no *ground truth*, os textos gerados foram associados com base nos nomes dos arquivos.
   - Cada texto gerado foi comparado ao texto correspondente no *ground truth*.
3. **Cálculo do BERTScore:**

- A biblioteca `bert-score` foi utilizada para que as pontuações de similaridade semântica entre os textos fossem calculadas. Essa medida compara os embeddings contextuais gerados por modelos pré-treinados, como o BERT, para que o quanto as palavras nos textos estão semanticamente relacionadas fosse avaliado. Para cada palavra no texto gerado, a similaridade de coseno com as palavras do texto de referência foi calculada, identificando-se as correspondências mais próximas. Essa abordagem vai além da análise lexical tradicional, pois o significado das palavras em seu contexto é considerado.
- A pontuação $F1$, que combina precisão e recall por meio de uma média harmônica, foi utilizada como medida principal. A precisão (P) reflete o quão bem as palavras do texto gerado correspondem semanticamente às do texto de referência, enquanto o recall (R) mede o quanto do texto de referência é coberto pelas palavras do texto gerado. A pontuação $F1$ foi sintetizada para garantir um equilíbrio entre ambas. Essa pontuação foi utilizada para que os cenários gerados fossem ranqueados, destacando-se aqueles mais semanticamente próximos ao *ground truth*.

4. **Organização dos Resultados:**

   - As pontuações $F1$ foram calculadas para cada par de textos (cenário base e cenário gerado) e organizadas em ordem decrescente. Os textos com pontuações mais altas foram considerados semanticamente mais similares ao cenário de referência.
   - Para cada cenário base, as pontuações foram agrupadas por execução e armazenadas em arquivos CSV contendo as seguintes informações: o cenário base, a execução comparada, a pontuação BERTScore ($F1$) e o ranking.

Esse fluxo foi repetido para que cada modelo avaliado fosse testado utilizando as diferentes técnicas de *prompt* (*zero-shot*, *one-shot*, *few-shot*), permitindo que uma análise detalhada da similaridade semântica entre os cenários gerados e os textos de referência fosse realizada.

A escolha do BERTScore como medida foi motivada por sua capacidade de capturar nuances semânticas e dependências contextuais, utilizando-se embeddings de modelos pré-treinados. Essa abordagem supera métodos tradicionais baseados em n-gramas, ao permitir

que a similaridade semântica em nível de palavra e contexto fosse considerada. A biblioteca `bert-score` permitiu que essa medida fosse implementada de forma eficiente, adaptando-se ao idioma utilizado (*pt*) e garantindo que resultados confiáveis fossem alcançados.

3.4.1.3 Processamento do METEOR

Correspondências exatas de palavras, sinônimos, ordem das palavras e diferenças no comprimento do texto são consideradas. No script, a medida foi adaptada para que a expansão de sinônimos fosse incluída por meio da biblioteca `WordNet`, garantindo-se maior sensibilidade às relações semânticas. A seguir, o fluxo do processo é detalhado:

1. **Preparação dos Dados:**
    - Os cenários de referência foram descompactados e convertidos para o formato `.feature`, sendo organizados em um diretório denominado `ground_truth`.
    - Os cenários gerados pelos modelos foram separados em pastas (`exec_1` a `exec_5`), cada uma representando diferentes execuções para as técnicas de *prompt* (*zero-shot*, *one-shot* e *few-shot*).
2. **Comparação entre cenários:**
    - Para cada cenário base presente no *ground truth*, os cenários gerados pelos modelos foram associados com base nos nomes dos arquivos.
    - Antes da comparação, os textos foram tokenizados e expandidos para que sinônimos fossem incluídos, utilizando-se a funcionalidade `synsets` da biblioteca `WordNet`.
3. **Expansão com Sinônimos:**
    - Cada palavra nos textos foi analisada para que seus sinônimos fossem identificados, adicionando-se esses termos ao conjunto de palavras comparadas.
    - Por exemplo, se o texto continha a palavra "car", sinônimos como "automobile"ou "vehicle"foram automaticamente incluídos, aumentando-se as chances de que correspondências semânticas entre os textos fossem identificadas.

4. **Cálculo do METEOR:**

    - A medida foi implementada para que correspondências exatas de palavras e sinônimos entre os textos fossem identificadas. Além disso, diferenças significativas na ordem das palavras foram penalizadas.
    - A precisão (*precision*), que reflete o percentual de palavras no texto gerado que correspondem ao texto de referência (incluindo sinônimos), e o recall (*recall*), que avalia o percentual de palavras no texto de referência cobertas pelo texto gerado, foram consideradas no cálculo.
    - Para que alterações significativas na estrutura fossem penalizadas, um fator de penalização foi incluído com base na fragmentação da ordem das palavras.
    - A pontuação final foi ajustada para que o recall fosse priorizado em relação à precisão, refletindo-se a estratégia padrão do METEOR para favorecer coberturas completas dos textos de referência.

5. **Organização dos Resultados:**

    - Para cada cenário base, as pontuações METEOR foram calculadas para os cenários gerados em todas as execuções. Essas pontuações foram organizadas em ordem decrescente, criando-se um ranking que destacou os textos mais semanticamente similares ao *ground truth*.
    - Os resultados foram exportados em arquivos CSV contendo: o cenário base, a execução comparada, a pontuação METEOR e o ranking.

Esse processo foi repetido para que cada modelo avaliado fosse testado utilizando as diferentes técnicas de *prompt* (*zero-shot*, *one-shot*, *few-shot*). A medida METEOR foi considerada eficaz para que nuances semânticas e estruturais fossem capturadas, ampliadas pelo uso de sinônimos, o que a tornou uma ferramenta poderosa para que textos em contextos com variações lexicais e estruturais fossem comparados.

3.4.2 Validação Qualitativa

A validação de medidas é considerada um processo crítico para que seja garantido que os instrumentos empregados em uma análise realmente mensurem o fenômeno de interesse

de forma precisa e confiável. Em especial, acredita-se que a análise qualitativa humana, combinada com correlações, permita que nuances e contextos sejam identificados, os quais podem ser perdidos em abordagens exclusivamente quantitativas [42].

A incorporação da análise qualitativa é vista como essencial para que a consistência entre os cenários gerados por LLM aplicados à transformação de casos livres em cenários BDD, seja avaliada. A validação mais robusta é possibilitada pela correlação de medidas quantitativas com interpretações qualitativas, assegurando-se que os resultados reflitam não apenas a conformidade sintática, mas também a adequação semântica e contextual às intenções originais.

Portanto, acredita-se que integrar métodos qualitativos ao processo de validação fortaleça a confiabilidade do modelo, promovendo-se uma análise mais abrangente e alinhada às expectativas práticas de sistemas reais.

3.4.2.1 Metodologia da Avaliação Manual

A avaliação manual foi conduzida a partir de um conjunto de 10 casos selecionados entre os resultados gerados pelos LLM, independentemente da técnica de prompt utilizada. A seleção desses casos foi realizada com base em critérios específicos definidos a partir das medidas previamente calculadas, como discrepância entre METEOR e BERTScore, consistência entre medidas, variações significativas e comportamentos diferenciados entre modelos. Essa abordagem garantiu que os casos analisados qualitativamente fossem representativos de cenários de interesse particular e capturassem diferentes padrões de comportamento dos modelos.

Os casos selecionados foram avaliados qualitativamente com base em critérios de avaliação predefinidos, descritos na próxima subseção. A comparação foi realizada em relação a um *ground truth*, previamente definido por um especialista, que estabeleceu cenários ideais para análise. Ao final do processo, cada caso recebeu uma pontuação que foi calculada por meio de uma média ponderada dos critérios avaliados, refletindo-se a performance geral dos modelos no contexto da transformação de casos livres em cenários BDD.

3.4.2.2 Critérios da Avaliação

Os critérios apresentados a seguir foram utilizados para que os cenários de teste analisados fossem avaliados, abrangendo-se aspectos de organização, significado e detalhamento, com

o objetivo de que clareza e precisão fossem asseguradas em cada avaliação.

Estrutura: A organização do cenário é avaliada, incluindo-se a sequência lógica dos passos, a completude das informações e a concisão na escrita. O objetivo é garantir que o cenário seja considerado claro, coeso e fácil de entender.

Semântica: O significado e a intenção do cenário são analisados, verificando-se se a linguagem utilizada é precisa, se o objetivo do teste é considerado claro e se o cenário cobre adequadamente a funcionalidade em teste.

Detalhes: O nível de detalhamento do cenário é avaliado, observando-se se ele contém informações suficientes para a execução do teste, mas sem que seja excessivamente específico ou dependente da interface, garantindo-se que o cenário seja robusto a mudanças.

Os critérios de avaliação utilizados neste estudo foram definidos pelo pesquisador, o que pode ser considerado uma fonte de viés na análise qualitativa dos cenários de teste. A subjetividade inerente à escolha e aplicação desses critérios foi reconhecida como um possível fator de influência nos resultados, mesmo com o objetivo de que a imparcialidade fosse mantida. Além disso, a análise qualitativa foi conduzida exclusivamente pelo pesquisador, reforçando-se a possibilidade de que interpretações individuais tenham impactado as pontuações atribuídas. Apesar dessas limitações, foi assegurado que os critérios fossem cuidadosamente selecionados com base em boas práticas, buscando-se garantir a consistência e a relevância das avaliações realizadas.

3.4.2.3 Resultados da Avaliação

A análise de dez casos de teste, selecionados aleatoriamente, foi realizada manualmente, conforme apresentado na Tabela 3.4. Para cada caso, três critérios fundamentais foram avaliados: **Estrutura**, **Semântica** e **Detalhes**, com pesos atribuídos de acordo com sua relevância para a qualidade de um cenário *BDD*. A **Estrutura** e a **Semântica** receberam peso 4 cada, por serem aspectos críticos que garantem a organização lógica e a clareza do cenário. Já os **Detalhes**, com peso 2, foram considerados importantes, mas menos impactantes na avaliação geral. A nota final de cada caso foi calculada utilizando a média ponderada, conforme a seguinte fórmula:

$$\text{Nota Final} = \frac{(\text{Estrutura} \times 4) + (\text{Semântica} \times 4) + (\text{Detalhes} \times 2)}{10} \tag{3.1}$$

A Tabela 3.4 apresenta a avaliação detalhada de 4 dos 10 casos analisados, com as notas atribuídas a cada critério e a nota final calculada. Por exemplo, no caso *"O campo senha deve seguir regra <regra>"*, a estrutura recebeu nota 7, a semântica 7,5 e os detalhes 7, resultando em uma nota final de 7,2. Essa abordagem permite uma análise equilibrada da qualidade dos cenários, considerando tanto a organização quanto o significado e os aspectos complementares.

Tabela 3.4: Avaliação Final

Caso	Estrutura (Nota) - Peso 4	Semântica (Nota) - Peso 4	Detalhes (Nota) - Peso 2	Nota Final
O campo senha deve seguir regra <regra>	7	7,5	7	7,2
Cadastro de perfil com identificador existente	8	8,5	7	8
Realizar ação sem ter feito login	8,5	9	9	8,8
Editar informações do condomínio com sucesso	7,5	8,3	7	7,72

Além da avaliação qualitativa, foi realizada uma análise quantitativa dos casos, utilizando medidas como a *Distância de Manhattan*, o *METEOR* e o *BERTScore*, conforme apresentado na Tabela 3.5. Essas medidas foram aplicadas para medir a similaridade e a precisão dos cenários gerados em relação aos cenários de referência. A *Distância de Manhattan* avalia discrepâncias estruturais, enquanto o *METEOR* e o *BERTScore* focam na similaridade semântica, considerando aspectos como sinônimos, fluidez textual e *embeddings* contextuais. Por exemplo, no caso *"Cadastro de perfil com identificador existente"*, utilizando o modelo *Llama 3* com técnica *zero-shot*, a *Distância de Manhattan* foi 24, o *METEOR* 0,74 e o *BERTScore* 76,08, indicando um bom alinhamento com o cenário de referência.

Tabela 3.5: Análise Dos 4 Primeiros Casos Com Medidas Quantitativas.

Caso	Modelo	Técnica	Manhattan	METEOR	BERTScore
O campo senha deve seguir regra <regra>	Gemini	few-shot	45	0,69	46,34
Cadastro de perfil com identificador existente	Llama 3	zero-shot	24	0,74	76,08
Realizar ação sem ter feito login	Gpt-3.5-turbo	one-shot	24	0,87	59,59
Editar informações do condomínio com sucesso	Phi-3	zero-shot	38	0,78	54,05

As tabelas completas, contendo a avaliação qualitativa e quantitativa de todos os 10 casos analisados, estão disponíveis no apêndice deste estudo (Tabelas B.1 e B.2). Essa abordagem combinada permite uma análise abrangente, considerando tanto a percepção humana quanto medidas objetivas para avaliar a qualidade dos cenários *BDD* gerados.

3.4.3 Testes Estatísticos Para Escolha da Medida

A escolha da medida ideal para que a similaridade entre os cenários gerados pelos LLM e os cenários de referência fosse avaliada foi fundamentada em um processo que combinou análises quantitativas e qualitativas. Esse processo foi estruturado em etapas, abrangendo-se desde a seleção de cenários representativos até a aplicação de análises descritivas, testes de normalidade e correlações com a avaliação qualitativa. O objetivo central foi que a medida que melhor refletisse os julgamentos humanos sobre a qualidade dos cenários gerados fosse identificada, garantindo-se uma avaliação robusta e alinhada aos objetivos do estudo.

Os resultados das análises descritivas, apresentados na Tabela 3.6, complementam essa abordagem ao serem fornecidas informações detalhadas sobre a posição e dispersão das medidas avaliadas, como Manhattan, METEOR, BERTScore e Avaliação Qualitativa. Entre as medidas, o METEOR foi destacado por sua maior consistência, evidenciada pelo menor desvio padrão (0.06), enquanto a Manhattan foi associada a uma maior amplitude de valores, indicando-se maior variabilidade nos resultados. É importante que seja ressaltado que as me-

didas possuem diferentes escalas: Manhattan não possui escala fixa, BERTScore varia de 0 a 100 e METEOR de 0 a 1. Essa análise foi considerada essencial para que uma compreensão mais clara das características de cada medida e sua adequação ao contexto do estudo fosse alcançada.

Tabela 3.6: Estatísticas Descritivas das Medidas Avaliadas

Medida	**Média**	**Desvio Padrão**	**Mínimo**	**Mediana**	**Máximo**
Manhattan	35,20	7,32	24,00	36,50	45,00
METEOR	0,77	0,06	0,69	0,79	0,87
BERTScore	58,99	8,35	46,34	56,50	76,08
Avaliação Qualitativa	0,79	0,05	0,71	0,78	0,88

Para avaliar a normalidade das distribuições, foi realizado o teste Shapiro-Wilk, cujos resultados indicaram que todas as medidas seguem uma distribuição normal ($p > 0.05$) conforme mostrado na Tabela 3.7. A normalidade observada justifica o uso de correlações para medidas para a análise.

Tabela 3.7: Resultados do Teste Shapiro-Wilk para Normalidade

Medida	**Estatística Shapiro-Wilk**	**p-valor**	**Normalidade**
Manhattan	0,92	0,31	Sim
METEOR	0,92	0,38	Sim
BERTScore	0,93	0,46	Sim
Avaliação Qualitativa	0,97	0,90	Sim

3.4.3.1 Correlação Entre Medidas

Os coeficientes de correlação de Pearson e Spearman são apresentados nas Tabelas 3.8 e 3.9, respectivamente. Em ambas as análises, o METEOR apresentou a maior correlação com a Avaliação Qualitativa ($r = 0,88$ e $\rho = 0,83$), reforçando sua adequação como medida principal para avaliação. A normalização do BERTScore e da Avaliação Qualitativa garantiu que METEOR, BERTScore e Avaliação Qualitativa estivessem na mesma escala, variando de 0 a 1. Para isso, o BERTScore foi dividido por 100 e a Avaliação Qualitativa por 10. A

Manhattan apresentou correlação negativa com as demais medidas, refletindo sua natureza distinta, já que, para essa medida, valores maiores indicam menor similaridade e valores menores indicam maior similaridade. Além disso, a Manhattan não possui uma escala definida e, por essa razão, não foi normalizada.

Tabela 3.8: Correlação de Pearson Entre as Medidas Avaliadas

Medida	**Manhattan**	**METEOR**	**BERTScore**	**Avaliação Qualitativa**
Manhattan	1,00	-0,48	-0,61	-0,52
METEOR	-0,48	1,00	0,26	0,88
BERTScore	-0,61	0,26	1,00	0,39
Avaliação Qualitativa	-0,52	0,88	0,39	1,00

As correlações de Spearman, apresentadas na Tabela 3.9, oferecem uma análise complementar às de Pearson, avaliando associações monotônicas entre as medidas, independentemente de suposições sobre a linearidade. Essa abordagem é especialmente útil para identificar relações consistentes entre as variáveis, mesmo na presença de distribuições não normais.

Tabela 3.9: Correlação de Spearman Entre as Medidas Avaliadas

Medida	**Manhattan**	**METEOR**	**BERTScore**	**Avaliação Qualitativa**
Manhattan	1,00	-0,48	-0,57	-0,44
METEOR	-0,48	1,00	0,34	0,83
BERTScore	-0,57	0,34	1,00	0,43
Avaliação Qualitativa	-0,44	0,83	0,43	1,00

Com base na normalidade dos dados, evidenciada pelos resultados do teste de Shapiro-Wilk ($p > 0,05$ para todas as medidas), a correlação de *Pearson* foi escolhida como a principal abordagem de análise. A correlação de Pearson pressupõe a normalidade das variáveis para capturar relações lineares de forma precisa e confiável, o que é adequado neste estudo devido à distribuição normal observada nos dados. Essa escolha é justificada pela capacidade de Pearson de fornecer uma medida robusta e interpretável das associações lineares entre as

medidas avaliadas e a Avaliação Qualitativa, sendo especialmente relevante em contextos em que os dados seguem uma distribuição normal. A análise de Spearman foi utilizada de forma complementar, mas a normalidade dos dados reforça a preferência por Pearson como a medida principal neste estudo.

3.4.3.2 Escolha Final da Medida

Com base nos resultados das análises de correlação, o METEOR foi identificado como a medida mais adequada para avaliar a similaridade entre os cenários de teste gerados e os cenários de referência (*ground truth*). Conforme apresentado na Tabela 3.8, a correlação de Pearson revelou uma associação muito forte entre o METEOR e a Avaliação Qualitativa ($r = 0,88$), superando as demais medidas avaliadas. Além disso, a consistência observada no METEOR, evidenciada por seu menor desvio padrão (0.06), reforça sua robustez em cenários variados.

De acordo com os intervalos de correlação definidos por Cohen [18], valores de $0,10 \leq |r| < 0,30$ são considerados correlações fracas, $0,30 \leq |r| < 0,50$ são moderadas, $0,50 \leq |r| < 0,70$ são fortes, e $|r| \geq 0,70$ indicam correlações muito fortes. Nesse contexto, a correlação observada entre o METEOR e a Avaliação Qualitativa ($r = 0,88$) é classificada como muito forte, evidenciando a capacidade do METEOR de capturar os julgamentos humanos de similaridade semântica e estrutural.

A escolha do METEOR como medida ideal está alinhada com os princípios descritos por Cohen [18], que destaca a importância de selecionar medidas com maior poder de detecção de efeitos relevantes. O poder estatístico de uma medida depende não apenas de sua sensibilidade a diferenças, mas também de sua correlação com avaliações externas, como julgamentos humanos. Nesse sentido, o METEOR demonstrou a melhor capacidade de capturar as nuances semânticas e estruturais avaliadas qualitativamente, tornando-o a escolha mais apropriada para este estudo.

A decisão de utilizar a correlação de Pearson foi embasada na normalidade dos dados, conforme evidenciado pelos resultados do teste de Shapiro-Wilk (Tabela 3.7), que indicaram que todas as medidas apresentaram distribuições normais ($p > 0,05$). Essa característica reforça a adequação do uso de Pearson, dada sua maior sensibilidade e precisão em dados normalmente distribuídos. Dessa forma, a combinação de alta correlação ($r = 0.88$) e me-

nor variabilidade do METEOR consolidam sua posição como a medida mais confiável para avaliar a similaridade semântica e estrutural no contexto deste estudo.

3.4.4 Ameaças à Validade

A validade interna deste estudo pode ser afetada por várias questões. A seleção aleatória dos cenários BDD, embora realizada com critérios rigorosos, pode não refletir completamente a complexidade real dos cenários existentes. Essa limitação pode introduzir distorções, uma vez que os resultados obtidos podem ser influenciados por uma representação parcial do conjunto de cenários possíveis. Além disso, o viés do pesquisador e dos especialistas representa uma ameaça significativa, pois tanto a elaboração dos cenários de referência quanto sua validação podem ser influenciadas por fatores humanos. Apesar da experiência dos especialistas envolvidos, o viés humano pode impactar, mesmo que de forma inconsciente, a escolha de aspectos avaliados como relevantes ou adequados nos cenários de referência, resultando em uma preferência involuntária que afete a imparcialidade tanto na criação quanto na validação desses cenários.

A operacionalização das variáveis também pode limitar a validade interna. As medidas utilizadas para avaliar a qualidade dos cenários BDD podem não abranger completamente o constructo de interesse. Aspectos qualitativos importantes, que poderiam ser melhor avaliados por humanos, podem ser negligenciados devido à dependência exclusiva de medidas automatizadas. Adicionalmente, o comportamento estocástico dos LLM utilizados pode gerar variabilidade nas respostas, mesmo sob condições semelhantes de execução, o que compromete a reprodutibilidade dos resultados. A falta de documentação consolidada de boas práticas no uso de BDD também representa uma ameaça à validade, uma vez que a maioria das práticas consideradas neste estudo foi retirada de documentos e estudos com base na expertise de profissionais, e há poucas referências amplamente aceitas que possam servir como guia universal.

A validade de construtos é outra categoria que pode ser afetada. As medidas utilizadas, como METEOR, BERTScore e Distância de Manhattan, embora complementares, podem não capturar todos os aspectos relevantes para a qualidade dos cenários BDD gerados. Enquanto essas medidas oferecem perspectivas sobre discrepância estrutural, similaridade semântica e precisão textual, aspectos mais sutis, como nuances semânticas específicas ou

adequação contextual, podem ter sido negligenciados, limitando a análise qualitativa e aprofundada dos resultados. Para mitigar essa ameaça, o estudo combinou avaliações quantitativas com análises qualitativas realizadas por especialistas, garantindo que aspectos mais subjetivos fossem considerados.

A validade externa está condicionada à representatividade da amostra utilizada no estudo. Se a amostra de cenários BDD for pequena ou enviesada, as conclusões obtidas podem não ser aplicáveis a outros cenários ou contextos, dificultando a generalização dos resultados para diferentes domínios ou condições. Neste estudo, os cenários foram coletados do estudo de de Souza Filho [21], porém não foram categorizados quanto ao seu domínio específico. Essa ausência de categorização pode limitar a análise da aplicabilidade dos resultados em diferentes áreas, uma vez que não é possível avaliar se determinados domínios estão sobre ou sub-representados na amostra. Além disso, o processo de filtragem dos cenários iniciais, que incluiu a remoção de descrições curtas ou redundantes, pode ter alterado a composição final da amostra, afetando sua diversidade e representatividade. Para mitigar essa ameaça, o estudo buscou garantir a seleção de cenários representativos, priorizando funcionalidades críticas e excluindo redundâncias por similaridade semântica.

As limitações relacionadas ao custo e ao tempo dos experimentos também representam uma ameaça à validade estatística. LLM mais complexos, como o GPT-4 Turbo e o LLaMA, exigem maior tempo de processamento e recursos computacionais significativos, o que pode restringir a quantidade de execuções realizadas e, consequentemente, a análise de variabilidade e consistência nos resultados. Para mitigar essa questão, o estudo realizou múltiplas execuções para cada técnica de *prompting*, garantindo uma análise mais robusta da variabilidade e consistência dos resultados.

Por fim, o viés na estruturação dos *prompts* utilizados para a geração dos cenários pode impactar os resultados, refletindo as expectativas ou entendimentos prévios dos pesquisadores e, potencialmente, favorecendo determinadas abordagens ou modelos. Para mitigar esses efeitos, diversas estratégias foram implementadas ao longo do estudo. A medida escolhida para avaliar a qualidade das saídas foi selecionada com base em sua correlação com avaliação qualitativa, assegurando maior alinhamento entre medidas objetivas e julgamentos humanos. Adicionalmente, a utilização de amostras aleatórias de cenários de teste visou reduzir o enviesamento nos dados de entrada e garantir maior representatividade. Reconhecem-se, no

entanto, os limites inerentes aos *prompts* utilizados, enfatizando que sua formulação pode influenciar os resultados obtidos, sendo necessário explorar abordagens alternativas em estudos futuros.

Em resumo, embora o estudo tenha adotado diversas estratégias para mitigar as ameaças à validade, como a combinação de avaliações quantitativas e qualitativas, a seleção de cenários representativos e a realização de múltiplas execuções, ainda há espaço para melhorias, especialmente na documentação de boas práticas e na diversificação dos cenários utilizados. Essas melhorias podem contribuir para uma análise mais abrangente e robusta em estudos futuros.

3.5 Considerações Finais do Capítulo

O processo de construção dos *prompts* foi orientado pelas boas práticas de BDD, com o objetivo de que cenários BDD claros e concisos fossem gerados. As técnicas de *prompting zero-shot*, *one-shot* e *few-shot* foram empregadas para que descrições de cenários de teste fossem transformadas em cenários BDD no formato Gherkin. A base de dados utilizada foi extraída de um estudo prévio [21] e submetida a um processo rigoroso de seleção, resultando em um conjunto de 105 cenários relevantes, dos quais 10 foram selecionados de forma aleatória para serem utilizados no estudo. A construção da base de referência (*ground truth*) foi realizada por um especialista e revisada por outros dois, assegurando a qualidade dos cenários e sua utilidade prática. Os experimentos foram conduzidos no ambiente *Google Colab*, utilizando scripts em *Python* e APIs para interação com os LLM. A coleta de dados foi automatizada e organizada para que a reprodutibilidade dos experimentos fosse garantida.

A seleção da medida foi realizada considerando-se a correlação com a avaliação qualitativa fornecida por um especialista, garantindo-se que a medida representasse de forma fiel a qualidade percebida dos cenários de teste gerados. A avaliação qualitativa foi conduzida por um especialista, enquanto os cenários de referência foram elaborados por outro, servindo como base para que a análise e validação das medidas utilizadas no estudo fossem realizadas.

Entre as medidas analisadas, o *METEOR* foi identificado como a mais adequada. Sua seleção foi justificada pela apresentação da maior correlação com a avaliação qualitativa ($r = 0.88$), superando-se outras medidas, como a Distância de Manhattan e o *BERTScore*. Além

disso, foi constatado que o *METEOR* exibiu maior consistência, evidenciada pelo menor desvio padrão em relação às demais medidas, o que reforçou sua robustez para aplicação em diferentes cenários.

Os critérios utilizados para a escolha da medida incluíram a habilidade de que aspectos como discrepância estrutural, similaridade semântica e precisão textual fossem capturados. A validação foi conduzida por meio de análises quantitativas e qualitativas, combinando-se dados estatísticos com as avaliações realizadas pelos especialistas. Essa abordagem garantiu que a medida selecionada fosse robusta e adequada ao contexto do estudo.

Concluiu-se que a escolha cuidadosa da medida foi considerada essencial para que a confiabilidade dos resultados obtidos fosse assegurada. A utilização do *METEOR* como principal medida foi fundamentada em sua alta correlação com os julgamentos humanos, sua baixa variabilidade e sua capacidade de que a qualidade dos cenários de teste gerados pelos LLM fosse representada com precisão. Esses fatores confirmaram sua adequação como medida central para análises relacionadas à geração automatizada de cenários de teste.

Capítulo 4

Estudo Comparativo

Este capítulo apresenta os resultados do experimento comparativo realizado entre os diferentes LLM avaliados (GPT4-Turbo, GPT3.5-Turbo, GPT4o-Mini, Llama 3, Phi-3, Gemini). Inicialmente, foi realizada uma análise de variabilidade dos resultados gerados por cada LLM, considerando-se as diferentes técnicas de *prompting* (*zero-shot*, *one-shot* e *few-shot*). Em seguida, foi conduzida uma análise de desempenho de cada LLM, utilizando-se a medida *METEOR* para que a qualidade dos cenários de teste gerados fosse avaliada. Por fim, foi realizada uma análise comparativa entre os LLM, considerando-se as diferentes técnicas de *prompting*, com o objetivo de que fossem identificados qual LLM e qual técnica de *prompting* foram mais adequados para a geração de cenários de teste.

A análise de variabilidade foi concentrada na avaliação da consistência dos resultados gerados pelos LLM, considerando-se a natureza estocástica desses LLM. Para que a variabilidade dos resultados gerados por cada LLM em cada técnica fosse avaliada, medidas estatísticas como média, desvio padrão e coeficiente de variação (CV) foram calculadas com base em várias execuções para os mesmos cenários de teste.

A análise de desempenho foi direcionada à avaliação da qualidade dos cenários de teste gerados pelos LLM. Para isso, a medida *METEOR* foi utilizada, medindo-se a similaridade entre os cenários gerados e os cenários de referência (*ground truth*). Para que a qualidade dos cenários de teste gerados fosse avaliada, testes estatísticos foram conduzidos para comparar as médias das pontuações *METEOR* entre as diferentes técnicas.

4.1 Análise de Variabilidade

A variabilidade em LLM é um aspecto crucial para que a consistência nos resultados gerados por diferentes técnicas seja compreendida. Foi indicado por estudos que a estabilidade dos LLM não é apenas um desafio técnico, mas também um aspecto essencial para que a confiabilidade de suas aplicações práticas seja assegurada [6, 56]. Conforme destacado por Atil et al. [6], variações nos resultados foram evidenciadas, podendo atingir até 10%, mesmo sob configurações determinísticas, ressaltando-se a importância de que a consistência dos modelos seja avaliada de forma mais rigorosa. Por sua vez, foi enfatizado por Pimentel et al. [56] que a ausência de padronização nas avaliações é considerada uma contribuição significativa para variações que impactam a reprodutibilidade e a comparabilidade entre diferentes técnicas e LLM.

No contexto de cenários de testes gerados automaticamente, a análise de variabilidade é considerada essencial para que seja assegurado que os resultados obtidos sejam suficientemente consistentes para uso prático. LLM, devido à sua natureza probabilística, podem apresentar variações significativas nas respostas dependendo da técnica utilizada, como já postulado anteriormente. Essas variações precisam ser compreendidas para que estratégias que ofereçam um equilíbrio ideal entre estabilidade e flexibilidade sejam selecionadas, especialmente em cenários onde confiabilidade é uma exigência crítica. Também se observa que a análise é útil para que limitações e pontos de melhoria nos LLM avaliados sejam identificados, promovendo-se avanços na geração de testes automáticos. No que se segue, detalha-se a metodologia aplicada para analisar a variabilidade dos LLM no contexto desta pesquisa (4.1.1) e os resultados da análise (4.1.2).

4.1.1 Metodologia

A metodologia teve como objetivo mensurar, de forma quantitativa, a variabilidade das respostas geradas por LLM, considerando as três técnicas de *prompting* estudadas (*zero-shot*, *one-shot* e *few-shot*). De caráter **exploratório** e **experimental**, a pesquisa utilizou medidas estatísticas e testes controlados para avaliar e comparar, de maneira sistemática, a consistência dos resultados obtidos.

As etapas foram estruturadas em um processo que incluiu:

- **Coleta dos dados**: Foi realizada por meio de múltiplas execuções dos LLM, totalizando 5 execuções por técnica, em que pontuações de similaridade foram geradas, organizadas em planilhas e rankeadas.
- **Análise estatística**: Foram calculadas médias, desvios padrão e coeficientes de variação (CV) para que a variabilidade entre os cenários e entre as técnicas fosse avaliada para cada LLM.

4.1.1.1 Dados Utilizados

Para cada LLM, três técnicas distintas de geração foram testadas: *zero-shot*, *one-shot* e *few-shot*. Cada técnica foi aplicada a um conjunto fixo de dez cenários base, totalizando-se 50 execuções por LLM para cada técnica, ao todo o foram 150 cenários gerados por LLM, considerando os 10 selecionados. Os resultados foram armazenados em arquivos no formato CSV, nos quais as seguintes informações principais foram incluídas:

- **Cenário Base**: Identificador do cenário de teste utilizado como entrada.
- **Pontuação METEOR**: Similaridade textual calculada entre a saída do LLM e o cenário base de referência (*ground-truth*.
- **Ranking**: Posição relativa no ranking de similaridade considerando o cenário de referência.

Arquivos de saída de cada LLM foram analisados para que medidas de variabilidade fossem calculadas, como o coeficiente de variação (CV), média, desvio padrão e amplitude, com o objetivo de que padrões de estabilidade e instabilidade entre os cenários e as técnicas fossem identificados.

4.1.1.2 Processo de Análise

O cálculo da variabilidade foi realizado de maneira sistemática para que a consistência e a precisão das medidas fossem garantidas. Após a coleta e organização das planilhas contendo

as pontuações (*Pontuação METEOR*) geradas em cinco execuções para cada técnica (*zero-shot*, *one-shot*, *few-shot*) e para cada modelo, um script[1] em Python foi desenvolvido para que os dados fossem processados e as medidas estatísticas necessárias fossem calculadas. Abaixo, as etapas do cálculo são detalhadas:

1. **Leitura e validação dos dados:**
    - Os arquivos CSV foram lidos utilizando a biblioteca `pandas`, sendo garantida a padronização dos dados.
    - Foi verificada pelo script a presença das colunas essenciais, `Cenário Base` e `Pontuação METEOR`, para que fosse assegurado que os dados estavam no formato correto para análise.
2. **Agrupamento por cenário base:**
    - As pontuações foram agrupadas com base na coluna `Cenário Base`, permitindo-se que uma análise individualizada de cada cenário fosse realizada.
    - Esse agrupamento foi considerado fundamental para que medidas específicas para cada cenário fossem calculadas, isolando-se os efeitos de cada técnica sobre as respostas.
3. **Cálculo das medidas estatísticas por cenário:** Para cada cenário base, as seguintes medidas foram calculadas:
    - **Média:** Representa a pontuação média das cinco execuções do cenário base, sendo calculada como:
    $$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$
    onde x_i é a pontuação de uma execução e $n = 5$ o número de execuções.
    - **Desvio Padrão:** Mede a dispersão das pontuações em relação à média, sendo dado por:
    $$\sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2}$$

[1]Todos os scripts utilizados nesse estudo estão disponíveis em: `https://github.com/hiagonfs/bdd-scenario-evaluation`

onde $\bar{x}$ é a média e $n = 5$.

- **Coeficiente de Variação (CV):** Avalia a relação percentual entre o desvio padrão e a média, sendo definido como:

$$CV = \frac{\sigma}{\bar{x}} \times 100$$

Essa medida é expressa em porcentagem e foi utilizada como principal indicador de variabilidade.

4. **Cálculo das medidas globais:** As medidas globais resumem a variabilidade geral das respostas dos LLM ao considerar o conjunto completo de cenários avaliados. A média dos coeficientes de variação (*Mean CV*) reflete a variabilidade média entre todos os cenários, enquanto o desvio padrão dos CVs (*Standard Deviation of CV*) avalia a dispersão dessa variabilidade, permitindo identificar padrões gerais e consistência nos resultados de forma consolidada.

   - A média dos coeficientes de variação (*Mean CV*) foi calculada como:

$$\text{Média do CV} = \frac{1}{m} \sum_{j=1}^{m} CV_j$$

onde CV_j é o coeficiente de variação de cada cenário base e m é o número total de cenários.

   - O desvio padrão dos coeficientes de variação (*Standard Deviation of CV*) foi calculado para que a dispersão dos CVs entre os cenários fosse medida:

$$\text{Desvio Padrão do CV} = \sqrt{\frac{1}{m-1} \sum_{j=1}^{m} (CV_j - \text{Média do CV})^2}$$

4.1.2 Análise Estatística

A análise estatística da variabilidade dos resultados gerados pelos LLM avaliados utilizou como principais indicadores a média, o desvio padrão (DP) e o coeficiente de variação (CV), aplicados em nível global e individual. Esses indicadores permitiram quantificar a consistência das pontuações METEOR obtidas em cinco execuções para as técnicas *zero-shot*, *one-shot* e *few-shot*, com os resultados consolidados apresentados na Tabela 4.1.

Tabela 4.1: Média, Desvio Padrão E Coeficiente De Variação (CV) Por LLM E Técnica.

Modelo	**Zero-shot**			**One-shot**			**Few-shot**		
	Média	**DP**	**CV (%)**	**Média**	**DP**	**CV (%)**	**Média**	**DP**	**CV (%)**
GPT-4 Turbo	0,77	0,04	5,19	0,73	0,04	5,48	0,75	0,06	8,00
GPT-3.5 Turbo	0,76	0,05	6,58	0,72	0,05	6,94	0,70	0,04	5,71
GPT-4o Mini	0,78	0,03	3,85	0,74	0,04	5,41	0,73	0,03	4,11
Gemini	0,84	0,03	3,57	0,78	0,05	6,41	0,74	0,06	8,11
Llama 3	0,76	0,05	6,58	0,70	0,04	5,71	0,72	0,03	4,17
Phi-3	0,81	0,04	4,94	0,74	0,05	6,76	0,70	0,06	8,57

Todos os dados coletados durante a análise de variabilidade, incluindo tabelas e todas as medidas calculadas por caso, estão disponíveis no Apêndice C. Com base nesses dados, observou-se que, na técnica *zero-shot*, houve uma variabilidade acentuada nos LLM Llama 3 e GPT-3.5 Turbo, ambos apresentando **CV = 6,58%**. Em contraste, o LLM Gemini foi identificado como o mais consistente nesta técnica, com o menor **CV = 3,57%** e desvio padrão **DP = 0,03**, evidenciando sua capacidade de lidar com entradas complexas sem exemplos prévios. O Phi-3 foi destacado por sua boa estabilidade em *zero-shot*, com **CV = 4,94%**.

Na técnica *one-shot*, verificou-se que a introdução de um exemplo reduziu a variabilidade geral. O GPT-4o Mini apresentou o menor **CV = 5,41%** nesta técnica, seguido pelo GPT-4 Turbo, com **CV = 5,48%**. O Gemini apresentou **CV = 6,41%**, mantendo um bom nível de consistência, enquanto o Phi-3 apresentou **CV = 6,76%**. O Llama 3 exibiu **CV = 5,71%**, indicando uma variabilidade maior em comparação aos demais modelos nesta técnica.

Na técnica *few-shot*, foi evidenciada maior variabilidade no Phi-3 (**CV = 8,57%**) e no Gemini (**CV = 8,11%**). O GPT-4 Turbo também apresentou uma variabilidade considerável, com **CV = 8,00%**. Por outro lado, o GPT-4o Mini e o Llama 3 foram destacados por sua excelente estabilidade em *few-shot*, com **CV = 4,11%** e **CV = 4,17%**, respectivamente, sendo considerados os LLM mais consistentes nessa técnica.

Ao analisar os resultados, foi constatado que o *Gemini* demonstrou excepcional estabilidade em *zero-shot*, com um coeficiente de variação (**CV**) de **3,57%**, sendo considerado a escolha ideal quando exemplos não estão disponíveis. O *Phi-3*, por sua vez, emergiu como uma alternativa em *zero-shot*, com **CV = 4,94%**. Embora o *GPT-4 Mini* tenha apresentado

valores de **CV** ligeiramente menores, sua média de desempenho foi inferior à do *Phi-3*, que alcançou uma média mais alta, indicando maior qualidade nos resultados. Essa combinação de alta média e estabilidade relativa posiciona o *Phi-3* como uma opção viável para cenários onde a precisão e a confiabilidade são prioritárias.. Na técnica *few-shot*, o GPT-4o Mini e o Llama 3 apresentaram a maior consistência, com **CV = 4,11%** e **CV = 4,17%**, respectivamente, destacando-se em cenários que demandam adaptação com poucos exemplos. Em *one-shot*, os menores **CVs** foram atribuídos ao GPT-4o Mini (**CV = 5,41%**) e ao GPT-4 Turbo (**CV = 5,48%**).

Em resumo, considerando a estabilidade, temos:

- **Melhor LLM para Zero-Shot:** Gemini, com o menor coeficiente de variação (**CV = 3,57%**) e desvio padrão (**DP = 0,03**), destacando-se como o mais consistente nesta técnica.

- **Melhor LLM para One-Shot:** GPT-4o Mini, com o menor coeficiente de variação (**CV = 5,41%**) e desvio padrão (**DP = 0,04**), evidenciando sua maior estabilidade quando um exemplo é fornecido.

- **Melhor LLM para Few-Shot:** GPT-4o Mini, apresentando novamente o menor coeficiente de variação (**CV = 4,11%**) e desvio padrão (**DP = 0,03**), sendo o mais estável em cenários que utilizam múltiplos exemplos.

Portanto, a seleção do LLM e da técnica, considerando a variabilidade dos resultados, deve ser guiada pela disponibilidade de dados e pela necessidade de consistência. Para tarefas *zero-shot*, o Gemini é destacado como o mais consistente, com o menor coeficiente de variação (**CV = 3,57%**) e desvio padrão (**DP = 0,03**), demonstrando sua superioridade em lidar com entradas complexas sem exemplos prévios. Para tarefas *few-shot*, o GPT-4o Mini é escolhido devido ao menor coeficiente de variação (**CV = 4,11%**) e desvio padrão (**DP = 0,03**), apresentando resultados ligeiramente mais consistentes em comparação ao Llama 3. Já para tarefas *one-shot*, o GPT-4o Mini novamente se destaca com o menor coeficiente de variação (**CV = 5,41%**) e desvio padrão (**DP = 0,04**), superando o GPT-4 Turbo em termos de estabilidade.

4.2 Análise de Desempenho

Nesta seção, uma análise detalhada do desempenho dos seis LLM avaliados neste estudo é apresentada, examinando-se como cada um responde às diferentes técnicas de prompting: *zero-shot*, *one-shot* e *few-shot*. Para cada LLM, seu comportamento e desempenho são analisados através da medida METEOR, permitindo que suas características específicas sejam compreendidas e que se avalie como diferentes estratégias de prompting influenciam seus resultados. Essa estrutura de análise é utilizada para que os pontos fortes e limitações de cada LLM sejam identificados, bem como para que sejam determinadas as técnicas de prompting mais efetivas para cada LLM.

4.2.1 Metodologia

A análise de desempenho foi conduzida utilizando-se uma abordagem estruturada para que as pontuações METEOR de seis LLM fossem avaliadas em três técnicas de prompting: *zero-shot*, *one-shot* e *few-shot*. O processo incluiu a coleta e o processamento dos dados, seguidos pelos cálculos estatísticos e pelas visualizações gráficas, para que os resultados fossem interpretados.

4.2.1.1 Coleta e Organização dos Dados

Os dados analisados foram obtidos a partir da primeira execução dentre várias realizadas. Cada técnica de *prompting* foi aplicada a 10 casos de teste por técnica, resultando em um total de 30 observações por LLM. As pontuações METEOR foram coletadas diretamente das saídas dos LLM e organizadas em arquivos separados para cada técnica.

4.2.1.2 Análise Estatística

Com as pontuações coletadas, as seguintes etapas de análise estatística foram realizadas:

- **Verificação de Normalidade:** O teste de Shapiro-Wilk foi aplicado às pontuações de cada técnica (*zero-shot*, *one-shot* e *few-shot*) para que fosse verificado se os dados seguiam uma distribuição normal. Essa etapa foi considerada crucial para que o método estatístico apropriado para análise fosse determinado.

- **Verificação de Homocedasticidade:** Para os casos em que os dados apresentaram normalidade, a homocedasticidade (igualdade das variâncias entre os grupos) foi verificada utilizando o teste de Levene. A homocedasticidade é uma suposição importante para a aplicação da ANOVA, pois garante que as comparações entre grupos sejam válidas. Caso a homocedasticidade não fosse observada, indicando heterocedasticidade (variâncias desiguais entre os grupos), métodos alternativos ou correções foram considerados para garantir a robustez da análise.

- **Análise de Variância:** Para dados com distribuição normal e homocedasticidade confirmada, o teste ANOVA foi realizado para que as médias das pontuações entre as três técnicas fossem comparadas. Nos casos em que o p-valor foi menor que 0.05, indicando diferenças estatisticamente significativas, o teste pós-hoc de Tukey foi utilizado para que os grupos que diferiam entre si fossem identificados.

- **Testes Não Paramétricos:** Caso os dados não apresentassem normalidade ou homocedasticidade, o teste de Kruskal-Wallis foi aplicado para que diferenças nas distribuições fossem avaliadas. Nos casos de significância estatística, o teste pós-hoc de Dunn com ajuste de Bonferroni foi utilizado para que comparações par a par entre as técnicas fossem realizadas.

4.2.1.3 Cálculo das Estatísticas por LLM

Os valores apresentados nas tabelas por cada LLM foram calculados com base nos seguintes métodos e fórmulas:

- **Média:** Representa o valor médio das pontuações METEOR para cada técnica, calculado a partir das pontuações individuais.

- **Mediana:** Indica o valor central da distribuição das pontuações, ordenadas em ordem crescente.

- **Desvio Padrão:** Mede a dispersão das pontuações em relação à média, refletindo a variabilidade dos dados.

- **Variância:** Quantifica a dispersão das pontuações, sendo o desvio padrão elevado ao quadrado.

- **Mínimo e Máximo:** Representam os menores e maiores valores observados nas pontuações METEOR para cada técnica.
- **Amplitude:** Corresponde à diferença entre o valor máximo e o valor mínimo, indicando a dispersão total das pontuações.

Os resultados de cada cálculo foram implementados utilizando o script em Python descrito, com as bibliotecas `pandas` para manipulação dos dados e `numpy` para realizar os cálculos estatísticos.

4.2.1.4 Visualização dos Resultados

Para que a interpretação dos resultados fosse facilitada, gráficos de caixa (*boxplots*) foram gerados para ilustrar a distribuição das pontuações METEOR em cada técnica. Esses gráficos foram utilizados para que as diferenças de variabilidade e desempenho entre as técnicas e os LLM fossem destacadas, permitindo que tendências gerais e possíveis outliers fossem identificados.

É importante que seja ressaltado que a análise da variação considera exclusivamente as diferenças dentro da mesma execução e técnica, ou seja, avalia-se a dispersão das pontuações dos 10 casos de teste de uma única execução. Não são mensuradas as variabilidades entre diferentes execuções, mas sim como os casos de teste variam em uma única aplicação do LLM com a técnica específica. Esse enfoque foi adotado para que as conclusões refletissem as características da técnica e do LLM avaliados em um contexto controlado.

4.2.1.5 Ferramentas Utilizadas

A análise foi conduzida utilizando-se um conjunto de ferramentas de código aberto:

- **Python:** Foi utilizado para que os dados fossem processados, cálculos estatísticos fossem realizados e gráficos fossem gerados.
- **Bibliotecas Específicas:**
  - `pandas` foi utilizada para que a manipulação de dados fosse realizada.
  - `scipy` foi empregada para que testes estatísticos (Shapiro-Wilk, ANOVA e Kruskal-Wallis) fossem conduzidos.

  - `statsmodels` foi utilizada para que o teste pós-hoc de Tukey fosse aplicado.
  - `scikit-posthocs` foi empregada para que o teste de Dunn com ajuste de Bonferroni fosse realizado.

- **Matplotlib:** Foi utilizada para que os gráficos de caixa (*boxplots*) fossem criados.

4.2.1.6 Delimitações da Análise

A análise foi delimitada à avaliação da performance baseada em uma única execução para cada técnica e LLM, concentrando-se especificamente na comparação entre as saídas obtidas e as saídas esperadas (*ground truth* elaborado por especialista).

4.2.2 Resultados da Análise

Nesta seção, é apresentada a análise de desempenho dos LLM avaliados no estudo, considerando as técnicas de prompting *zero-shot*, *one-shot* e *few-shot*. A análise é conduzida com base nas pontuações METEOR obtidas para os cenários de teste gerados, comparando-as aos cenários de referência (*ground truth*). Também são examinadas as diferenças entre os LLM, destacando as técnicas mais adequadas para a geração de cenários BDD de alta qualidade por LLM. Os padrões de desempenho e possíveis limitações nos modelos avaliados são identificados.

4.2.2.1 Gemini

As estatísticas descritivas das pontuações METEOR para as três técnicas analisadas são apresentadas na Tabela 4.2. A maior média foi registrada para a técnica *zero-shot*, com um valor de 0,84, seguida pela técnica *one-shot*, com 0,78, e pela técnica *few-shot*, com 0,74. As medianas observadas foram de 0,83, 0,79 e 0,79 para as técnicas *zero-shot*, *one-shot* e *few-shot*, respectivamente. A menor variabilidade das pontuações, indicada pelo desvio padrão, foi constatada na técnica *one-shot*, com um valor de 0,07. Em contrapartida, as técnicas *zero-shot* e *few-shot* apresentaram desvios padrão de 0,08 e 0,10, respectivamente.

A maior amplitude das pontuações foi verificada na técnica *few-shot*, variando de 0,57 a 0,87. Por outro lado, amplitudes de 0,71 a 0,94 e de 0,67 a 0,88 foram registradas para as técnicas *zero-shot* e *one-shot*, respectivamente. Foi constatado, portanto, que a maior

concentração das pontuações em torno da média foi exibida pela técnica *zero-shot*, enquanto a maior dispersão foi observada para a técnica *few-shot*.

Técnica	**Média**	**Mediana**	**Desvio Padrão**	**Variância**	**Mínimo**	**Máximo**
Zero Shot	0,84	0,83	0,08	0,0064	0,71	0,94
One Shot	0,78	0,79	0,07	0,0049	0,67	0,88
Few Shot	0,74	0,79	0,10	0,0100	0,57	0,87

Tabela 4.2: Estatísticas Descritivas Das Pontuações METEOR Para Cada Técnica (Gemini).

Os testes de normalidade de Shapiro-Wilk apresentaram *p*-valores superiores a 0,05 para todas as técnicas, indicando que os dados seguem uma distribuição normal. Além disso, o teste de Bartlett para homocedasticidade revelou um *p*-valor de 0,44, confirmando que as variâncias entre os grupos são homogêneas. Esses resultados indicam que os pressupostos para a aplicação da ANOVA foram atendidos.

A análise de variância (ANOVA) revelou um valor de $F = 3,53$ e um p-valor de $0,04$, indicando que há uma diferença estatisticamente significativa entre as médias das três técnicas avaliadas. O teste *post-hoc* de Tukey demonstrou que a diferença entre as técnicas *few-shot* e *zero-shot* é estatisticamente significativa (p-valor $= 0,0347$). Por outro lado, as diferenças entre *few-shot* e *one-shot* (p-valor $= 0,5088$) e entre *one-shot* e *zero-shot* (p-valor $= 0,2961$) não foram significativas. Esses resultados sugerem que a técnica *zero-shot* apresenta um desempenho superior ao *few-shot* nas pontuações METEOR, mas não há evidências suficientes para afirmar que há diferenças significativas entre *zero-shot* e *one-shot*, ou entre *few-shot* e *one-shot*.

A Figura 4.1 apresenta o boxplot das pontuações METEOR para cada técnica. Visualmente, as medianas são mais altas para a técnica *zero-shot*, seguidas pelas técnicas *one-shot* e *few-shot*, confirmando as estatísticas descritivas. A dispersão das pontuações é maior em *few-shot*, consistente com os valores de desvio padrão reportados. Não foram observados *outliers* aparentes que pudessem comprometer a interpretação dos dados. A distribuição mais concentrada da técnica *zero-shot* reflete sua menor variabilidade e reforça seu melhor desempenho.

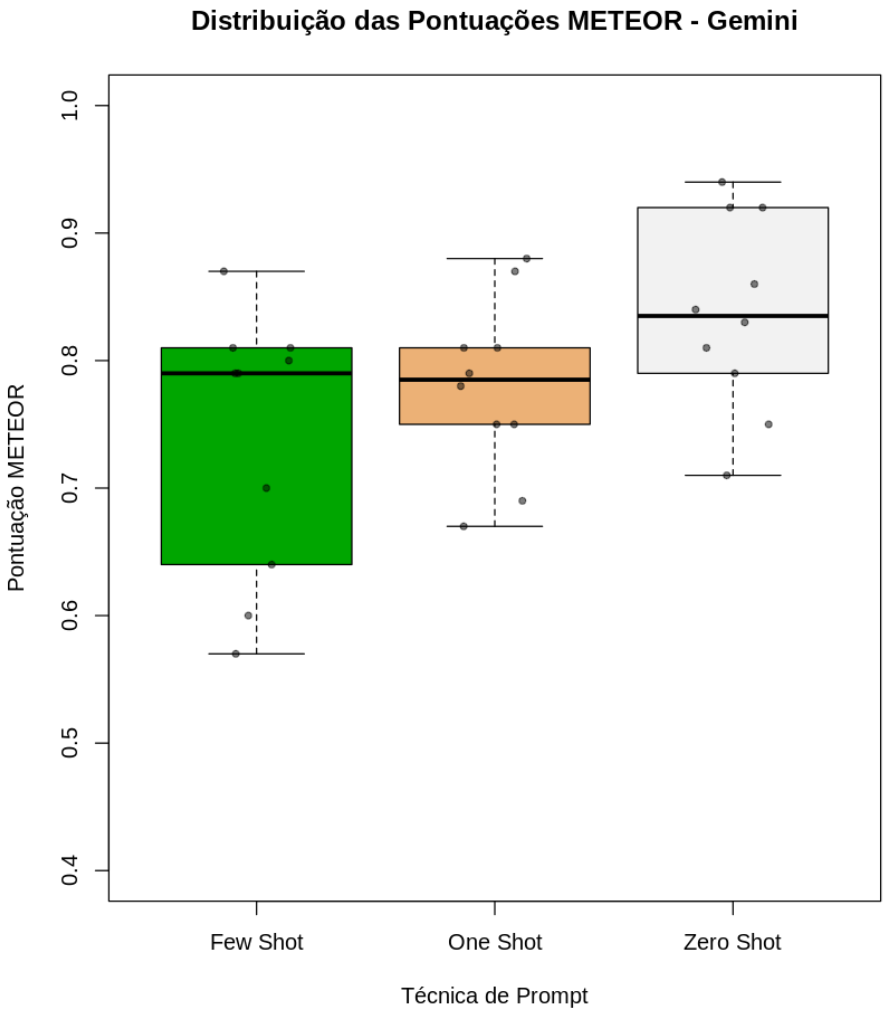

Figura 4.1: Boxplot Das Pontuações METEOR Para As Técnicas Zero Shot, One Shot E Few Shot (Gemini).

Em conclusão, foi indicado pela análise dos dados que a técnica *zero-shot* apresentou a melhor performance na geração de casos BDD, com a média mais alta e uma diferença estatisticamente significativa em relação à *few-shot*. Resultados competitivos também foram observados para a técnica *one-shot*, embora a diferença em relação à *zero-shot* não tenha sido estatisticamente significativa. Assim, a técnica *zero-shot* é recomendada como a mais adequada para que casos BDD de maior qualidade sejam gerados, devido à sua consistência e às suas pontuações superiores.

4.2.2.2 LLama 3

Na Tabela 4.3, foram detalhadas as estatísticas descritivas referentes às pontuações METEOR obtidas pelas três técnicas sob análise. Foi observado que a técnica *zero-shot* alcançou a média mais elevada, de 0,75, bem como a maior mediana, de 0,74. Em seguida, foi registrada pela técnica *few-shot* uma média de 0,72 e uma mediana de 0,70, enquanto pela técnica *one-shot* foram registrados uma média de 0,71 e uma mediana de 0,71.

No que tange à variabilidade das pontuações, expressa pelo desvio padrão, constatou-se que o maior valor, de 0,12, foi obtido pela técnica *zero-shot*, e o menor, de 0,07, pela técnica *few-shot*. A técnica *one-shot*, por sua vez, exibiu um desvio padrão intermediário, de 0,11. Quanto à amplitude, foram apresentados os maiores intervalos pelas técnicas *zero-shot* e *one-shot*, variando de 0,51 a 0,89 e de 0,51 a 0,85, respectivamente. Em contrapartida, uma amplitude menor foi observada para a técnica *few-shot*, com variação de 0,64 a 0,85.

Depreendeu-se, portanto, que as pontuações da técnica *few-shot* demonstraram maior concentração em torno de sua média, enquanto as técnicas *zero-shot* e *one-shot* revelaram maior dispersão em seus resultados.

Técnica	**Média**	**Mediana**	**Desvio Padrão**	**Variância**	**Mínimo**	**Máximo**
Zero Shot	0,75	0,74	0,12	0,0144	0,51	0,89
One Shot	0,71	0,71	0,11	0,0121	0,51	0,85
Few Shot	0,72	0,70	0,07	0,0049	0,64	0,85

Tabela 4.3: Estatísticas Descritivas Das Pontuações METEOR Para Cada Técnica (Llama 3).

Os testes de normalidade de Shapiro-Wilk apresentaram p-valores superiores a 0,05 para todas as técnicas (*zero-shot*, *one-shot* e *few-shot*), indicando que os dados seguem uma distribuição normal. Além disso, o teste de Bartlett para homocedasticidade revelou um p-valor de 0,24, confirmando que as variâncias entre os grupos são homogêneas. Esses resultados indicam que os pressupostos para a aplicação da ANOVA foram atendidos.

A análise de variância (ANOVA) apresentou um valor de F igual a 0,64 e um p-valor de 0,54, indicando que não há diferenças estatisticamente significativas entre as médias das três técnicas. Portanto, não foi necessário realizar testes pós-hoc para comparação entre os grupos.

A Figura 4.2 apresenta o boxplot das pontuações METEOR para cada técnica. Visualmente, as medianas são similares entre as três técnicas (*zero-shot*, *one-shot* e *few-shot*), enquanto a dispersão é maior em *zero-shot* e *one-shot*, consistente com os valores de desvio padrão reportados na Tabela 4.3. A técnica *few-shot* apresenta menor dispersão, sugerindo maior consistência nos resultados.

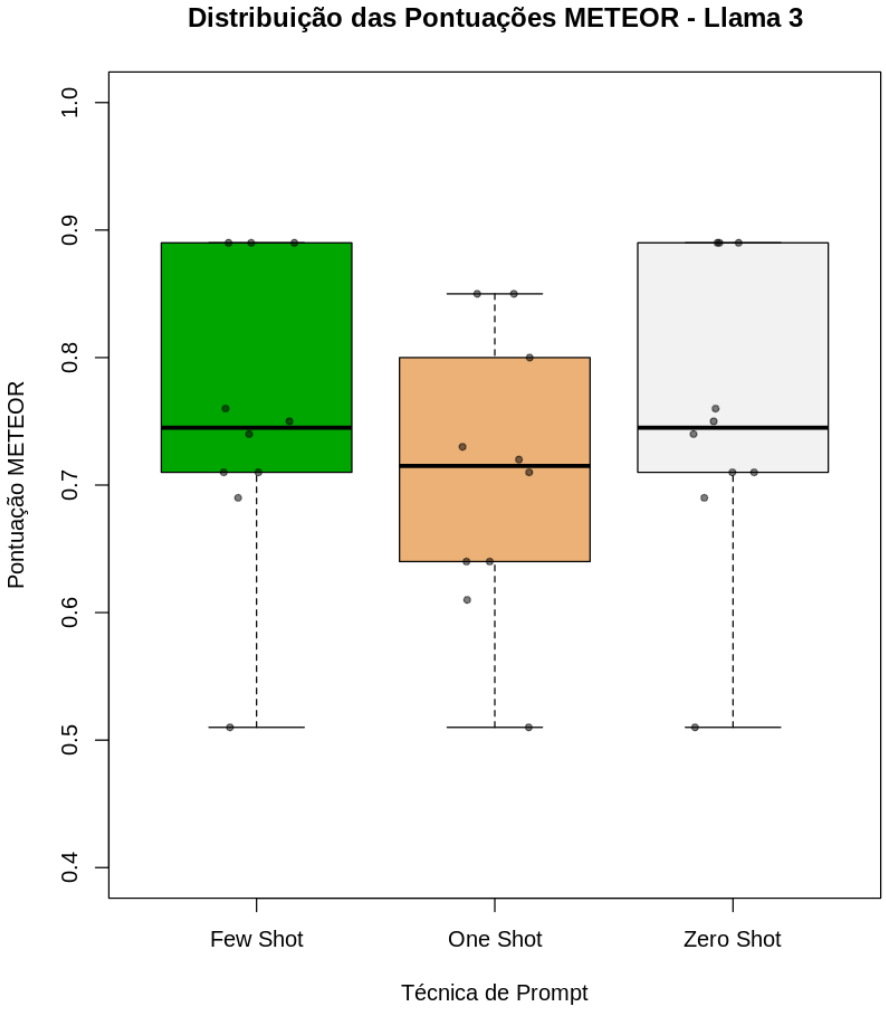

Figura 4.2: Boxplot Das Pontuações METEOR Para As Técnicas Zero Shot, One Shot E Few Shot (Llama 3).

Em conclusão, foi constatado pela análise dos dados que, embora diferenças estatisticamente significativas entre as técnicas *zero-shot*, *one-shot* e *few-shot* no LLM Llama 3 não tenham sido reveladas pela ANOVA ($F = 0{,}64$, p-valor $= 0{,}54$), uma avaliação mais detalhada do desempenho de cada técnica foi possibilitada. Foi destacado que a técnica *zero-shot*, ape-

sar de apresentar a maior dispersão, obteve a média mais elevada, de 0,75, e a maior mediana, de 0,74, sugerindo um potencial para que cenários BDD de alta qualidade sejam gerados. A técnica *few-shot*, por sua vez, foi identificada como a que apresentou a menor dispersão, com DP = 0,07, indicando maior consistência e previsibilidade em seus resultados, o que pode ser considerado um atributo valioso dependendo do contexto de aplicação. A técnica *one-shot*, embora tenha sido associada a resultados intermediários em termos de média e mediana, também apresentou uma dispersão relativamente alta.

Portanto, embora significância estatística não tenha sido alcançada, foi sugerido que a técnica *zero-shot* pode ser a mais promissora para que cenários BDD sejam gerados no LLM Llama 3, devido ao seu potencial para alcançar pontuações METEOR mais altas. No entanto, foi indicado que a escolha da técnica ideal pode depender dos atributos escolhidos. Se consistência e previsibilidade forem mais críticas, a técnica *few-shot* pode ser considerada a mais indicada. Se a busca por cenários de alta qualidade for priorizada, mesmo que com maior variabilidade, a técnica *zero-shot* parece ser a melhor opção.

4.2.2.3 Phi-3

Os resultados referentes às estatísticas descritivas das pontuações METEOR, para as três técnicas em questão, foram compilados na Tabela 4.4. Foi constatado que a técnica *zero-shot* destacou-se com a maior média, de 0,81, e a maior mediana, de 0,83. Subsequentemente, foram registradas médias de 0,74 e 0,70 para as técnicas *one-shot* e *few-shot*, respectivamente, acompanhadas das medianas de 0,74 e 0,69, na mesma ordem.

Em relação ao desvio padrão, indicador da variabilidade dos dados, verificou-se o menor valor, de 0,05, para a técnica *zero-shot*, enquanto valores idênticos, de 0,08, foram apresentados pelas técnicas *one-shot* e *few-shot*. A maior amplitude foi observada para a técnica *few-shot*, com uma variação de 0,58 a 0,84. Em contraste, amplitudes de 0,71 a 0,86 e de 0,60 a 0,86 foram registradas para as técnicas *zero-shot* e *one-shot*, respectivamente.

Concluiu-se, a partir destes dados, que as pontuações associadas à técnica *zero-shot* exibiram uma maior concentração em torno da média, enquanto uma dispersão mais acentuada foi identificada para as técnicas *one-shot* e *few-shot*.

Os testes de normalidade de Shapiro-Wilk apresentaram p-valores superiores a 0,05 para todas as técnicas, indicando que os dados seguem uma distribuição normal. Além disso, foi

Técnica	**Média**	**Mediana**	**Desvio Padrão**	**Variância**	**Mínimo**	**Máximo**
Zero Shot	0,81	0,83	0,05	0,0025	0,71	0,86
One Shot	0,74	0,74	0,08	0,0064	0,60	0,86
Few Shot	0,70	0,69	0,08	0,0064	0,58	0,84

Tabela 4.4: Estatísticas Descritivas Das Pontuações METEOR Para Cada Técnica (Phi-3).

revelado pelo teste de Bartlett para homocedasticidade um p-valor de 0,44, confirmando que as variâncias entre os grupos são homogêneas. Esses resultados indicaram que os pressupostos para a aplicação da ANOVA foram atendidos.

Foi constatado pela análise de variância (ANOVA) um valor de F igual a 5,15 e um p-valor de 0,01, indicando uma diferença estatisticamente significativa entre as médias das três técnicas. O teste pós-hoc de Tukey revelou que a diferença entre *few-shot* e *zero-shot* é estatisticamente significativa (p-valor = 0,0107), enquanto as diferenças entre *few-shot* e *one-shot* (p-valor = 0,5524) e entre *one-shot* e *zero-shot* (p-valor = 0,1086) não foram consideradas significativas. Isso sugere que a técnica *zero-shot* supera a *few-shot* nas pontuações METEOR, mas não foi encontrada evidência suficiente para afirmar uma diferença significativa entre *zero-shot* e *one-shot*, ou entre *few-shot* e *one-shot*.

Na Figura 4.3, o boxplot das pontuações METEOR para cada técnica é apresentado. Visualmente, foi identificado que as medianas são mais altas para a técnica *zero-shot*, seguidas pelas técnicas *one-shot* e *few-shot*, confirmando as estatísticas descritivas. A dispersão das pontuações foi maior nas técnicas *one-shot* e *few-shot*, consistente com os valores de desvio padrão reportados. Não foram observados *outliers* aparentes que pudessem comprometer a interpretação dos dados. Foi observado que a distribuição mais concentrada da técnica *zero-shot* reflete sua menor variabilidade e reforça seu melhor desempenho.

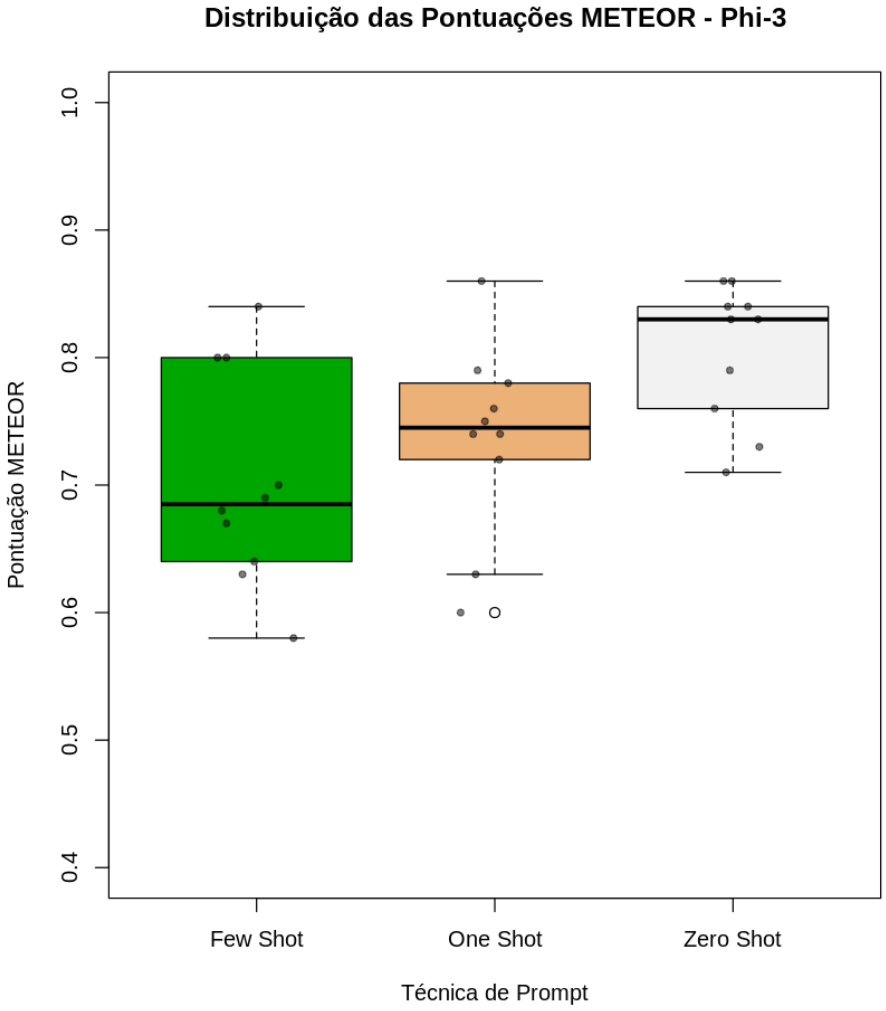

Figura 4.3: Boxplot das pontuações METEOR para as técnicas Zero Shot, One Shot e Few Shot (phi-3).

Em conclusão, foi indicado pela análise dos dados que a técnica *zero-shot* apresenta a melhor performance na geração de cenários BDD, com média e mediana mais altas, além de uma diferença estatisticamente significativa em relação à *few-shot*. Embora a técnica *one-shot* também tenha mostrado resultados competitivos, não foi identificada uma diferença estatisticamente significativa em relação à *zero-shot*. Assim, a técnica *zero-shot* é recomendada como a mais adequada para que cenários BDD de maior qualidade sejam gerados, devido à sua consistência e pontuações superiores.

4.2.2.4 GPT-3.5 Turbo

As estatísticas descritivas obtidas para as pontuações METEOR relativas às três técnicas avaliadas são expostas na Tabela 4.5. Foi constatado que a técnica *zero-shot* sobressaiu-se com a média mais alta, de 0,78, e a mediana mais expressiva, de 0,81. Na sequência, foram observadas as médias de 0,74 e 0,72 para as técnicas *one-shot* e *few-shot*, respectivamente, acompanhadas pelas medianas de 0,72 e 0,70, nesta ordem.

No que concerne ao desvio padrão, uma medida da variabilidade dos dados, foi identificado um valor idêntico, de 0,11, para as técnicas *zero-shot* e *one-shot*, enquanto para a técnica *few-shot* foi constatado um valor inferior, de 0,09. A técnica *zero-shot* revelou a maior amplitude, com uma variação de 0,59 a 0,91, enquanto a técnica *one-shot* apresentou uma amplitude ligeiramente menor, variando de 0,62 a 0,91. Já para a técnica *few-shot*, foi registrada uma amplitude menor, com um intervalo de 0,59 a 0,89.

Infere-se, com base nesses resultados, que as pontuações da técnica *few-shot* apresentaram maior concentração em torno da média, enquanto uma dispersão mais ampla foi verificada para as técnicas *zero-shot* e *one-shot*.

Técnica	**Média**	**Mediana**	**Desvio Padrão**	**Variância**	**Mínimo**	**Máximo**
Zero Shot	0,78	0,81	0,11	0,0121	0,59	0,91
One Shot	0,74	0,72	0,11	0,0121	0,62	0,91
Few Shot	0,72	0,70	0,09	0,0081	0,59	0,89

Tabela 4.5: Estatísticas Descritivas Das Pontuações METEOR Para Cada Técnica (GPT 3.5 Turbo).

Os testes de normalidade de Shapiro-Wilk apresentaram p-valores superiores a 0,05 para todas as técnicas, indicando que os dados seguem uma distribuição normal. Além disso, foi revelado pelo teste de Bartlett para homocedasticidade um p-valor de 0,84, confirmando que as variâncias entre os grupos são homogêneas. Esses resultados indicaram que os pressupostos necessários para a aplicação da ANOVA foram atendidos.

Foi apresentado pela análise de variância (ANOVA) um valor de F igual a 0,67 e um p-valor de 0,52, indicando que diferenças estatisticamente significativas entre as médias das três técnicas não foram identificadas. Portanto, não foi considerada necessária a realização

de testes pós-hoc para comparação entre os grupos.

Na Figura 4.4, o boxplot das pontuações METEOR para cada técnica é apresentado. Visualmente, foi constatado que as medianas são mais altas para a técnica *zero-shot*, seguidas pelas técnicas *one-shot* e *few-shot*, consistente com as estatísticas descritivas. Foi observado que as técnicas *zero-shot* e *one-shot* apresentam maior dispersão, enquanto a técnica *few-shot* demonstra maior consistência nos resultados.

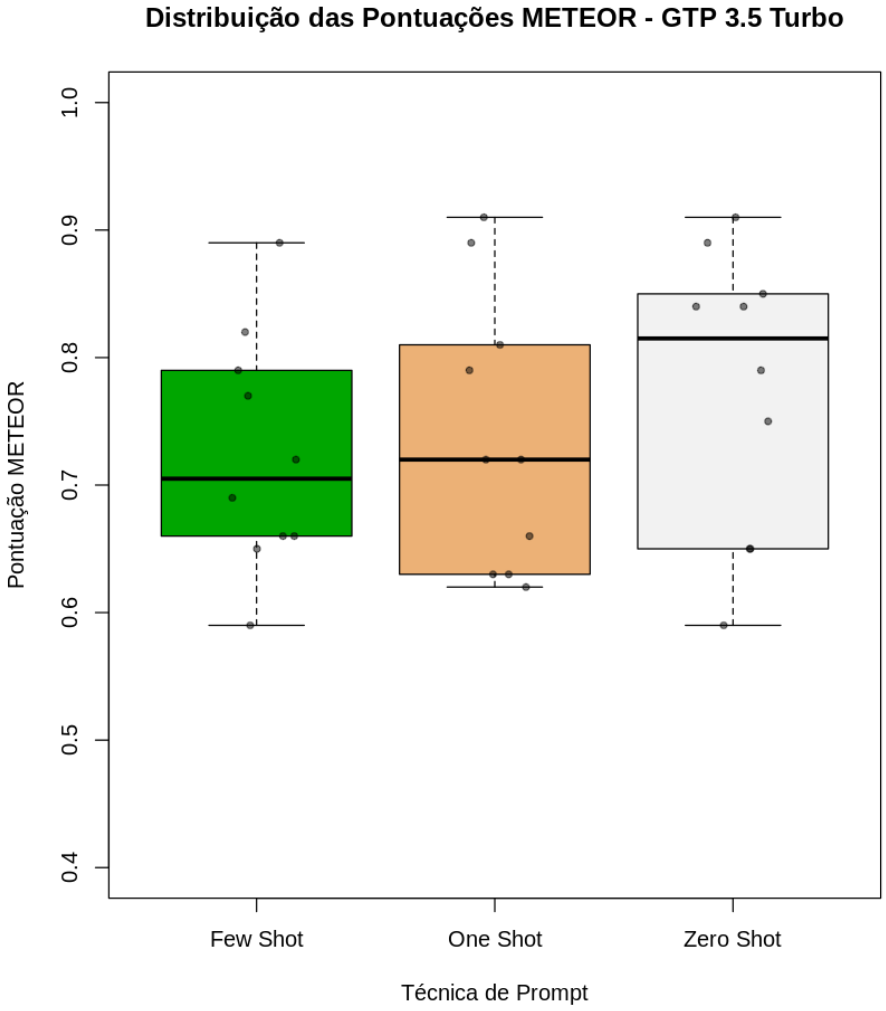

Figura 4.4: Boxplot Das Pontuações METEOR Para As Técnicas Zero Shot, One Shot E Few Shot (GPT 3.5 Turbo).

Embora diferenças estatisticamente significativas entre as técnicas *zero-shot*, *one-shot* e *few-shot* no LLM GPT-3,5 Turbo não tenham sido apontadas pela análise de variância (ANOVA, $F = 0,67$, p-valor $= 0,52$), uma análise descritiva permitiu destacar o desem-

penho superior de uma das técnicas em termos de alta qualidade. Alta qualidade, neste contexto, é definida como a capacidade de gerar cenários BDD com pontuações superiores na métrica METEOR, indicando maior alinhamento com o *ground truth*.

Foi constatado que a técnica *zero-shot* obteve a maior média (0,78) e a mediana mais alta (0,81), sugerindo um potencial significativo para que cenários BDD de alta qualidade sejam gerados. A técnica *one-shot*, embora tenha apresentado desempenho intermediário, também compartilhou características próximas em termos de dispersão e resultados gerais. Por outro lado, a técnica *few-shot*, apesar de sua menor variabilidade, mostrou médias e medianas inferiores, indicando menor alinhamento com o objetivo de alta qualidade.

Portanto, apesar da ausência de significância estatística, foi sugerido que a técnica *zero-shot* seja priorizada para projetos que demandem cenários BDD de alta qualidade, devido às suas pontuações superiores na métrica METEOR.

4.2.2.5 GPT-4 Turbo

Uma análise das estatísticas descritivas para as pontuações METEOR, referentes às três técnicas em estudo, é apresentada na Tabela 4.6. Foi constatado que a técnica *zero-shot* evidenciou a média mais elevada, de 0,74, juntamente com a maior mediana, de 0,78. Em seguida, a técnica *few-shot* foi associada a uma média de 0,73 e a uma mediana de 0,75, enquanto a técnica *one-shot* apresentou uma média de 0,71 e uma mediana de 0,69.

No que diz respeito ao desvio padrão, utilizado como um indicativo da variabilidade dos valores, foi observado o maior valor, de 0,10, para a técnica *few-shot*, um valor intermediário, de 0,09, para a técnica *zero-shot*, e o menor valor, de 0,08, para a técnica *one-shot*. Em relação às amplitudes, foi identificado que estas são similares entre as três técnicas, com a técnica *zero-shot* variando de 0,55 a 0,83, a *one-shot* de 0,61 a 0,86, e a *few-shot* de 0,56 a 0,86.

Técnica	**Média**	**Mediana**	**Desvio Padrão**	**Variância**	**Mínimo**	**Máximo**
Zero Shot	0,74	0,78	0,09	0,0081	0,55	0,83
One Shot	0,71	0,69	0,08	0,0064	0,61	0,86
Few Shot	0,73	0,75	0,10	0,0100	0,56	0,86

Tabela 4.6: Estatísticas Descritivas Das Pontuações METEOR Para Cada Técnica (GPT 4 Turbo).

Os testes de normalidade de Shapiro-Wilk apresentaram p-valores superiores a 0,05 para todas as técnicas, indicando que os dados seguem uma distribuição normal. Além disso, foi revelado pelo teste de Bartlett para homocedasticidade um p-valor de 0,85, confirmando que as variâncias entre os grupos são homogêneas. Com base nesses resultados, foi indicado que os pressupostos necessários para a aplicação da ANOVA foram atendidos.

Foi apresentado pela análise de variância (ANOVA) um valor de F igual a 0,33 e um p-valor de 0,72, indicando que diferenças estatisticamente significativas entre as médias das três técnicas não foram identificadas. Por esse motivo, não foi considerada necessária a realização de testes pós-hoc para comparação entre os grupos.

Na Figura 4.5, o boxplot das pontuações METEOR para cada técnica é apresentado. Visualmente, foi constatado que as medianas são mais altas para as técnicas *zero-shot* e *few-shot*, seguidas pela técnica *one-shot*. Foi observado que as técnicas apresentam dispersões similares, embora a técnica *few-shot* demonstre uma leve tendência a maior variabilidade, consistente com os valores de desvio padrão reportados.

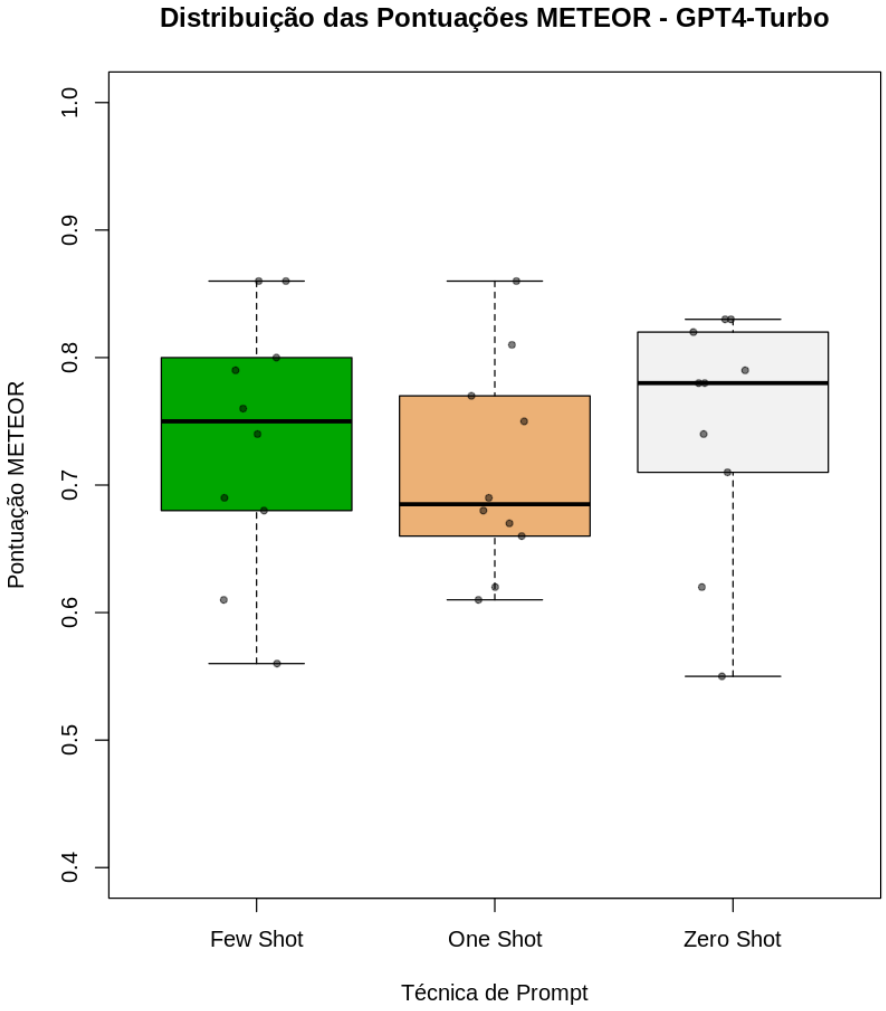

Figura 4.5: Boxplot Das Pontuações METEOR Para As Técnicas Zero Shot, One Shot E Few Shot (GPT 4 Turbo).

Na análise do LLM GPT-4 Turbo, foi constatado pela ANOVA que diferenças estatisticamente significativas entre as técnicas *zero-shot*, *one-shot* e *few-shot* não foram detectadas ($F = 0,33$, p-valor $= 0,72$). Contudo, foi evidenciado por uma avaliação descritiva que as técnicas *zero-shot* e *few-shot* alcançaram médias (0,74 e 0,73, respectivamente) e medianas (0,78 e 0,75, respectivamente) ligeiramente superiores às obtidas pela técnica *one-shot*, que apresentou uma média de 0,71 e uma mediana de 0,69.

Em termos de alta qualidade, definida como a capacidade de gerar cenários BDD mais alinhados às expectativas do *ground truth*, as técnicas *zero-shot* e *few-shot* destacaram-se como as mais promissoras. Apesar da ausência de significância estatística, foi sugerido que

a técnica *zero-shot* seja priorizada para projetos que demandem pontuações METEOR mais altas, enquanto a técnica *few-shot* pode ser considerada uma alternativa viável, devido ao seu desempenho competitivo em relação às medidas centrais em cenários similares.

4.2.2.6 GPT-4o Mini

Os dados estatísticos descritivos, relativos às pontuações METEOR das três técnicas analisadas, estão discriminados na Tabela 4.7. Foi constatado que a técnica *zero-shot* apresentou a maior média, de 0,78, e a maior mediana, de 0,80. Em seguida, foi observada uma média de 0,72 e uma mediana de 0,73 para a técnica *few-shot*, enquanto a técnica *one-shot* obteve uma média de 0,71 e uma mediana de 0,70.

No que se refere ao desvio padrão, que reflete a variabilidade dos dados, o maior valor, de 0,09, foi verificado para a técnica *one-shot*, seguido por um valor intermediário, de 0,08, para a técnica *zero-shot*, e o menor valor, de 0,07, para a técnica *few-shot*. Em termos de amplitude, foi constatado que a maior variação ocorreu para a técnica *zero-shot*, com valores entre 0,61 e 0,87, seguida pela técnica *one-shot*, com variação entre 0,58 e 0,83. A técnica *few-shot*, por sua vez, apresentou a menor amplitude, variando de 0,59 a 0,81.

Esses resultados sugerem que a técnica *few-shot* demonstra maior consistência em suas pontuações quando comparada às demais técnicas.

Técnica	**Média**	**Mediana**	**Desvio Padrão**	**Variância**	**Mínimo**	**Máximo**
Zero Shot	0,78	0,80	0,08	0,0064	0,61	0,87
One Shot	0,71	0,70	0,09	0,0081	0,58	0,83
Few Shot	0,72	0,73	0,07	0,0049	0,59	0,81

Tabela 4.7: Estatísticas Descritivas Das Pontuações METEOR Para Cada Técnica (GPT 4o Mini).

Os testes de normalidade de Shapiro-Wilk apresentaram p-valores superiores a 0,05 para todas as técnicas, indicando que os dados seguem uma distribuição normal. Além disso, foi revelado pelo teste de Bartlett para homocedasticidade um p-valor de 0,90, confirmando que as variâncias entre os grupos são homogêneas. Com base nesses resultados, foi indicado que os pressupostos necessários para a aplicação da ANOVA foram atendidos.

Foi constatado pela análise de variância (ANOVA) um valor de F igual a 2,25 e um p-valor de 0,12, indicando que diferenças estatisticamente significativas entre as médias das três técnicas não foram identificadas. Por esse motivo, não foi considerada necessária a realização de testes pós-hoc para comparação entre os grupos.

Na Figura 4.6, o boxplot das pontuações METEOR para cada técnica é apresentado. Visualmente, foi observado que as medianas são mais altas para as técnicas *zero-shot* e *few-shot*, seguidas pela técnica *one-shot*. Foi identificado que as técnicas apresentam dispersões similares, embora a técnica *few-shot* demonstre maior consistência, alinhada aos valores de desvio padrão mais baixos.

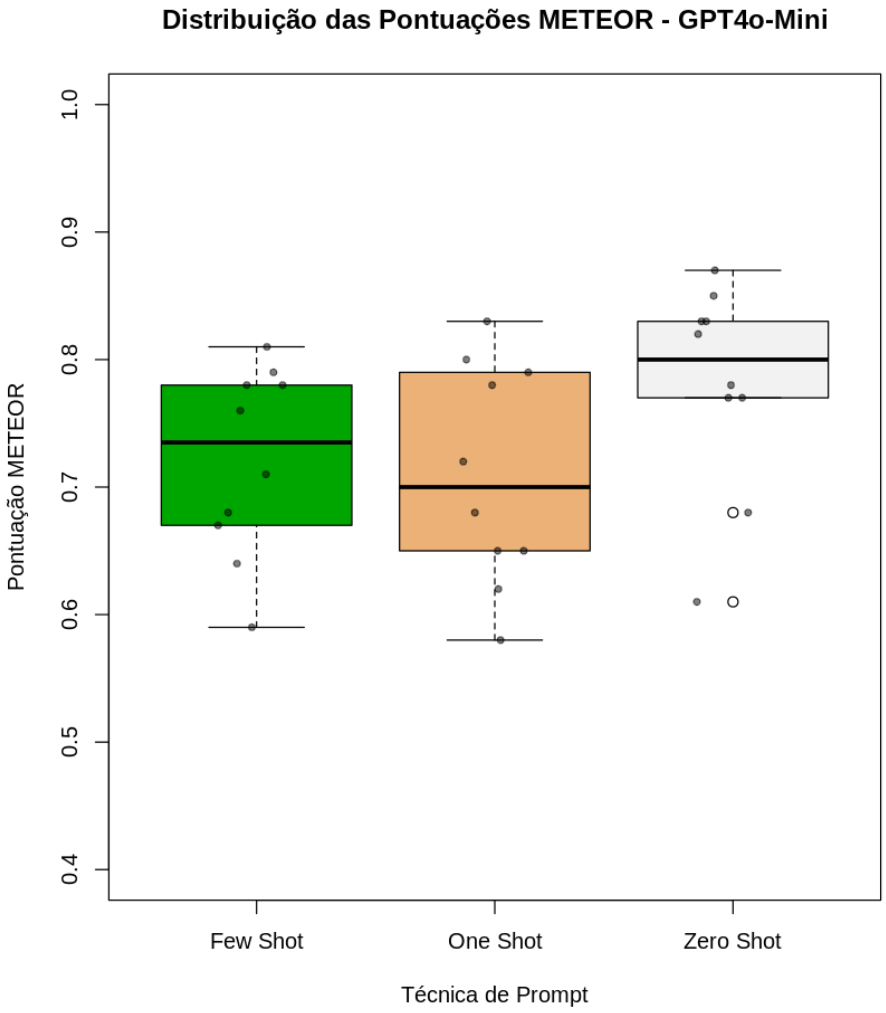

Figura 4.6: Boxplot Das Pontuações METEOR Para As Técnicas Zero Shot, One Shot E Few Shot (GPT 4o Mini).

No LLM GPT-4o Mini, foi constatado pela ANOVA que diferenças estatisticamente significativas entre as técnicas *zero-shot*, *one-shot* e *few-shot* não foram identificadas ($F = 2,25$, p-valor $= 0,12$). Contudo, foi evidenciado por uma análise descritiva que a técnica *zero-shot* apresentou a maior média (0,78) e mediana (0,80), sugerindo um potencial significativo para que casos BDD de alta qualidade sejam gerados.

Apesar da ausência de significância estatística, foi indicado que a técnica *zero-shot* se destaca como a mais adequada para projetos que demandem cenários BDD de alta qualidade, devido às suas pontuações superiores na métrica METEOR. Embora a técnica *few-shot* tenha demonstrado maior consistência em análises anteriores, o critério de alta qualidade justifica a priorização da técnica *zero-shot* neste contexto.

4.3 Análise Comparativa Entre Modelos

Foi realizada uma análise do desempenho dos LLM Gemini, Meta Meta Llama 3, Phi-3, GPT-3.5 Turbo, GPT-4 Turbo e GPT-4o Mini na geração de cenários BDD. Foi observado que o Gemini apresentou o melhor desempenho geral, com a maior média $(0,84)$ e mediana $(0,83)$ nas pontuações METEOR, além de uma baixa variabilidade (desvio padrão de $0,08$). Este modelo destacou-se especialmente na técnica *zero-shot*, sendo identificada alta consistência nos resultados.

Embora o desempenho do Meta Meta Llama 3 tenha sido inferior ao Gemini, foi alcançada uma média de $0,75$ e mediana de $0,74$, sendo notada uma maior variabilidade nos resultados (desvio padrão de $0,12$), o que pode ser interpretado como uma maior sensibilidade às entradas. Os resultados mais consistentes foram apresentados pelo Phi-3, com uma média de $0,81$, mediana de $0,83$ e o menor desvio padrão entre os modelos analisados $(0,05)$. Esse desempenho sugere um alto potencial para a geração de cenários BDD de qualidade com *zero-shot*.

Os LLM da família GPT apresentaram resultados equilibrados. Foi obtida uma média de $0,78$ e desvio padrão de $0,11$ para o GPT-3,5 Turbo, indicando uma leve dispersão nos resultados, mas com consistência suficiente para que o modelo fosse considerado confiável em cenários BDD. No caso do GPT-4o Mini, foi obtida a mesma média $(0,78)$, porém com um desvio padrão menor $(0,08)$, sugerindo uma performance mais estável e controlada, o

que indica que o modelo pode ser preferido em projetos que exijam maior regularidade nas pontuações.

O menor desempenho médio foi apresentado pelo GPT-4 Turbo $(0,74)$, acompanhado de um desvio padrão de $0,09$. Esse desempenho sugere que, embora o modelo consiga gerar cenários de qualidade, maior variabilidade e menor previsibilidade foram observadas nos resultados. Além disso, foram identificados *outliers* no GPT-4 Turbo, o que pode indicar que, em circunstâncias específicas, há dificuldades em manter a consistência na técnica *zero-shot*. Apesar disso, os resultados gerais do modelo permanecem competitivos quando comparados aos demais da família GPT.

Foi indicado pelo teste estatístico Kruskal-Wallis que não houve diferenças estatisticamente significativas entre as distribuições dos modelos ($H = 6,44$, $p = 0,27$). Esse resultado sugere que as variações nas pontuações médias podem ser atribuídas ao acaso, não sendo encontradas evidências suficientes para que uma superioridade clara de um modelo sobre outro fosse afirmada em termos de desempenho quantitativo.

Na Tabela 4.8, que resume os principais resultados da análise, foi possível observar que a escolha do LLM ideal deve depender das prioridades do projeto. Caso a consistência e a qualidade sejam fatores determinantes, os modelos Gemini e Phi-3 foram destacados como as melhores opções na técnica *zero-shot*. No entanto, outros fatores, como custo, disponibilidade e facilidade de implementação, devem ser considerados na tomada de decisão final.

Tabela 4.8: Análise Comparativa dos Modelos

Modelo	**Técnica**	**Média**	**Desvio Padrão**
Gemini	Zero Shot	0,84	0,08
Llama 3	Zero Shot	0,75	0,12
Phi-3	Zero Shot	0,81	0,05
GPT-3.5 Turbo	Zero Shot	0,78	0,11
GPT-4 Turbo	Zero Shot	0,74	0,09
GPT-4o Mini	Zero Shot	0,78	0,08

A Tabela 4.8 apresenta uma visão consolidada das medidas de desempenho de cada LLM na técnica *zero-shot*, destacando a média e o desvio padrão das pontuações METEOR. Para

complementar essa análise, a Figura 4.7 ilustra a distribuição detalhada das pontuações para cada modelo, permitindo uma visualização mais clara das variabilidades individuais e da presença de possíveis *outliers* em seus resultados.

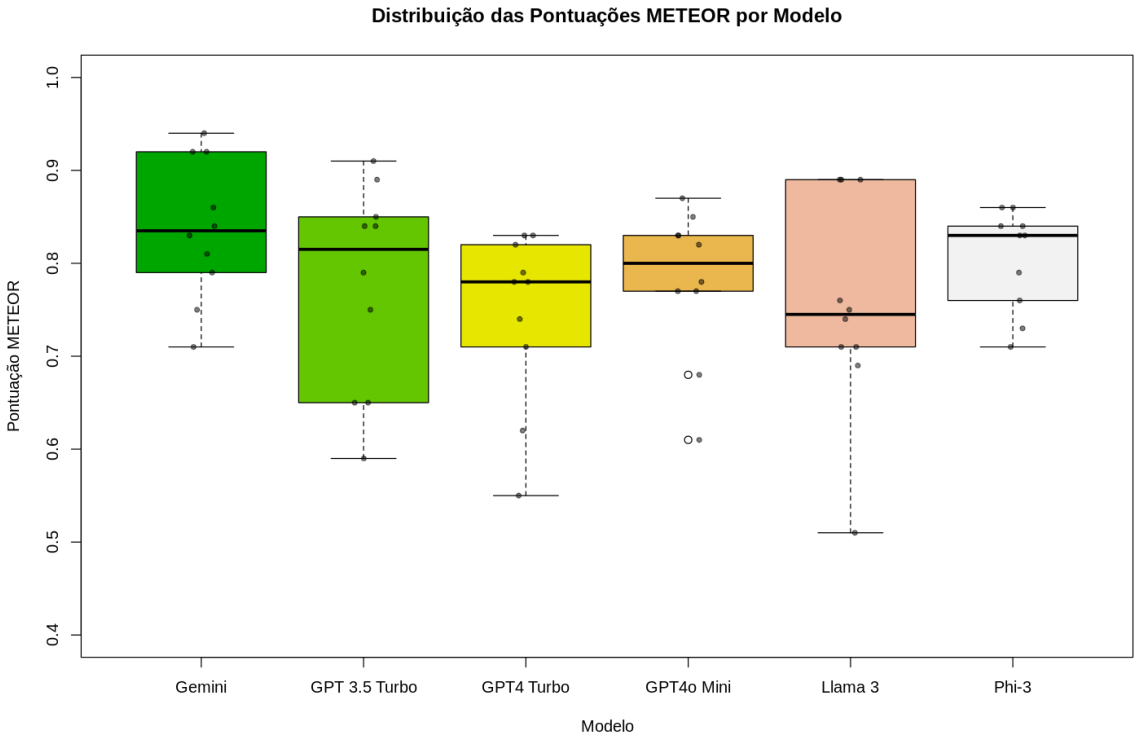

Figura 4.7: Distribuição das Pontuações METEOR - Melhor técnica por LLM

Os resultados do teste de Kruskal-Wallis indicam que não há diferenças estatisticamente significativas entre os modelos avaliados ($H = 6,44$, $p = 0,27$). Portanto, as escolhas entre os LLM devem considerar características específicas como estabilidade e variabilidade observada nas medidas, com os modelos Gemini e Phi-3 mostrando maior consistência. Contudo, não há evidências suficientes para afirmar a superioridade de um modelo sobre os demais com base nas medidas analisadas.

4.4 Ameaças à Validade

As principais ameaças à validade deste estudo incluem validade interna, externa, de construção e de conclusão. Essas ameaças foram analisadas e estratégias foram implementadas para mitigá-las.

No que diz respeito à validade interna, a seleção aleatória dos cenários BDD foi reconhecida como uma possível limitação, pois pode não refletir a complexidade real dos cenários existentes, introduzindo distorções nos resultados. Para mitigar essa ameaça, a base de referência foi construída a partir de um estudo prévio e submetida a um processo de seleção, garantindo que cenários representativos fossem incluídos. Além disso, o viés do pesquisador e dos especialistas envolvidos na elaboração e validação dos cenários foi considerado. Para reduzir esse impacto, a base foi desenvolvida por um especialista com vasta experiência e revisada por outros dois especialistas, buscando assegurar a qualidade e a imparcialidade dos cenários.

A operacionalização das variáveis também foi identificada como uma ameaça, uma vez que as medidas automatizadas utilizadas podem não capturar completamente o construto de interesse. Essa limitação foi mitigada pela escolha do METEOR como medida principal, dada sua alta correlação com a avaliação qualitativa e sua consistência, complementada por análises qualitativas realizadas por especialistas. Adicionalmente, o comportamento estocástico dos LLM, que pode comprometer a reprodutibilidade dos resultados [65], foi abordado por meio da realização de múltiplas execuções para cada técnica, com a aplicação de medidas estatísticas como média, desvio padrão e coeficiente de variação. Por fim, a estruturação dos *prompts* foi reconhecida como uma potencial fonte de viés, mas o uso de boas práticas de BDD buscou minimizar esse problema. Entretanto, é reconhecido que essa limitação não foi completamente eliminada.

Adicionalmente, reconhece-se que a qualidade dos *prompts* desenvolvidos pode ter impactado negativamente os resultados, caso não tenham sido suficientemente claros, completos ou alinhados ao objetivo de geração de cenários Gherkin. Embora boas práticas tenham sido utilizadas na elaboração desses *prompts*, essa ameaça não foi completamente eliminada.

Com relação à validade externa, a representatividade da amostra foi identificada como um ponto crítico. Embora o processo de seleção tenha sido rigoroso, garantindo a inclusão de cenários relevantes e diversificados, reconhece-se que a amostra utilizada pode não ser suficientemente ampla para generalizar os resultados a outros contextos. Além disso, as limitações de custo e tempo, especialmente devido às altas demandas computacionais dos LLM mais complexos, restringiram o número de execuções realizadas. Embora essa limitação não tenha sido completamente superada, os experimentos realizados foram suficientes

para alcançar os objetivos do estudo.

A validade de construção também apresentou desafios, principalmente no que diz respeito às medidas de avaliação utilizadas. Embora o METEOR tenha sido escolhido como medida principal devido à sua alta correlação com a avaliação qualitativa e à sua consistência, é reconhecido que aspectos mais sutis, como nuances semânticas específicas, podem não ter sido capturados. Para mitigar essa limitação, a análise quantitativa foi complementada por avaliações qualitativas realizadas por especialistas, o que permitiu uma interpretação mais abrangente dos resultados.

No que tange à validade de conclusão, a aplicação de testes estatísticos dependia de pressupostos como a normalidade dos dados e a homocedasticidade. Para assegurar o cumprimento desses pressupostos, testes como o de Shapiro-Wilk e o de Bartlett foram aplicados. Nos casos em que esses pressupostos não foram atendidos, testes não paramétricos, como o de Kruskal-Wallis, foram utilizados como alternativa. Além disso, a metodologia de análise de variabilidade considerou apenas as diferenças dentro da mesma execução e técnica, o que limita as conclusões sobre a consistência dos LLM em diferentes contextos. Apesar disso, essa abordagem permitiu uma análise detalhada e controlada da estabilidade dos LLM avaliados. Por fim, a análise de desempenho foi baseada em uma única execução para cada técnica e modelo, o que limita a generalização dos resultados. Essa limitação foi reconhecida, mas justificada pelo foco do estudo em comparar saídas específicas com o *ground truth*.

4.5 Considerações Finais do Capítulo

Uma análise aprofundada foi realizada para avaliar a variabilidade e o desempenho de seis LLM na geração de cenários BDD, considerando diferentes técnicas de prompting. Foram investigados os efeitos das estratégias *zero-shot*, *one-shot* e *few-shot* sobre a consistência e a qualidade dos resultados. Para a avaliação, foram empregados métodos estatísticos, como ANOVA e testes pós-hoc, além da medida METEOR, utilizada para medir a qualidade dos cenários de teste gerados.

Foi constatado pela análise da variabilidade dos resultados que a estabilidade dos LLM varia significativamente de acordo com a técnica de prompting aplicada. O LLM Gemini foi identificado como o mais consistente em cenários *zero-shot*, enquanto o GPT-4o Mini

e o Llama 3 foram destacados em *few-shot*. Na técnica *one-shot*, maior estabilidade foi associada ao GPT-4o Mini e ao GPT-4 Turbo. Esses resultados indicam que a escolha da técnica de prompting deve ser cuidadosamente considerada, levando em conta as características específicas de cada LLM e a disponibilidade de dados.

Foi demonstrado pela avaliação do desempenho dos LLM, com base na medida METEOR, que o Gemini apresentou o melhor desempenho geral, especialmente com a técnica *zero-shot*, evidenciando sua superioridade em qualidade e consistência nas pontuações. No entanto, os resultados do teste de Kruskal-Wallis ($H = 6,44$, $p = 0,27$) indicaram que as diferenças entre os modelos avaliados não são estatisticamente significativas. Isso sugere que a escolha do LLM ideal para a geração de cenários BDD deve levar em conta não apenas o desempenho quantitativo, mas também fatores como custo, disponibilidade, facilidade de implementação e adequação ao contexto específico. Além disso, a análise qualitativa dos cenários gerados por cada LLM continua sendo essencial para compreender melhor a adequação às necessidades do projeto e explorar potenciais melhorias.

Capítulo 5

Considerações Finais

Nesta dissertação, foi proposta uma abordagem inovadora para a geração de cenários de teste no formato Gherkin utilizando LLM, com o objetivo de auxiliar equipes de desenvolvimento na automação de testes, reduzindo o esforço manual e aprimorando a qualidade dos casos de teste. A principal contribuição do trabalho foi a identificação da técnica *zero-shot* como a mais eficaz para gerar cenários BDD, destacando-se por não requerer exemplos prévios, demandar menos poder computacional e permitir a criação de um *benchmark* para validação de cenários gerados por LLM.

Um dos aspectos mais cruciais deste estudo foi a seleção da medida de avaliação. O METEOR foi escolhido como a medida mais adequada para avaliar a similaridade entre os cenários de teste gerados pelos LLM e os cenários de referência (*ground truth*), apresentando a maior correlação com a avaliação qualitativa realizada por especialistas. A medida METEOR demonstrou consistência superior, com o menor desvio padrão em comparação com outras medidas analisadas, além de capturar aspectos como discrepância estrutural, similaridade semântica e precisão textual. Esse processo, além de validar os resultados, estabeleceu um benchmark que pode servir de base para análises futuras no contexto de BDD.

Diversos LLM, como GPT-3.5 Turbo, GPT-4 Turbo, GPT-4o Mini, LLaMA 3, Phi-3 e Gemini, foram avaliados em termos de qualidade e desempenho, considerando diferentes técnicas de prompting (*zero-shot*, *one-shot* e *few-shot*). Foi constatado que a técnica ideal varia de acordo com o LLM avaliado, sendo a técnica *zero-shot* geralmente associada ao melhor desempenho na geração de cenários BDD.

A técnica *zero-shot* destacou-se como a melhor em termos de qualidade para diversos

modelos, incluindo Gemini, Phi-3, GPT-3.5 Turbo e GPT-4o Mini. Especificamente:

- A técnica *zero-shot* destacou-se como a mais eficaz em termos de qualidade para diversos LLM avalidos.
- O Gemini foi identificado como o LLM mais eficiente com a técnica *zero-shot*, superando significativamente a *few-shot* em termos de qualidade.
- O Phi-3 também demonstrou seu melhor desempenho com a técnica *zero-shot*, apresentando uma diferença significativa em relação à *few-shot*.
- O GPT-3.5 Turbo e o GPT-4o Mini revelaram grande potencial para gerar cenários de alta qualidade utilizando a técnica *zero-shot*.
- O Llama 3, apesar de uma maior dispersão nos resultados com a técnica *zero-shot*, demonstrou notável capacidade para gerar cenários BDD de alta qualidade.
- O GPT-4 Turbo mostrou bom desempenho tanto na técnica *zero-shot* quanto na *few-shot*.

Por outro lado, a técnica *few-shot* foi geralmente associada ao pior desempenho em termos de qualidade, quando comparada à técnica *zero-shot*. Apesar de ter demonstrado maior consistência em alguns modelos, como o Llama 3 e o GPT-4o Mini, seu desempenho, medido pela métrica METEOR, foi inferior à técnica *zero-shot* na maioria dos casos. Nos LLM Gemini e Phi-3, evidenciou-se uma diferença estatisticamente significativa, com a técnica *zero-shot* superando a *few-shot* em termos de qualidade.

Adicionalmente, o estudo revelou que a técnica *one-shot* apresentaram resultados intermediários, com desempenho variando conforme o LLM avaliado.

Os resultados evidenciaram também que as técnicas de prompting impactam de maneira diferente o desempenho dos LLM. A técnica *zero-shot* apresentou, em geral, os melhores resultados em termos de qualidade. A técnica *few-shot* destacou-se pela maior consistência em alguns modelos, como LLaMA 3 e GPT-4o Mini, enquanto a técnica *one-shot* apresentou resultados intermediários, dependendo do LLM analisado. Essa análise destaca a necessidade de uma seleção criteriosa da técnica de prompting, considerando tanto as características específicas de cada LLM quanto a disponibilidade de dados.

A validação qualitativa, realizada por especialistas, complementou as análises quantitativas, permitindo que aspectos subjetivos, como estrutura, semântica e detalhes, fossem avaliados. A combinação das análises qualitativas e quantitativas garantiu que os resultados refletissem não apenas a conformidade sintática, mas também a adequação semântica e contextual. Contudo, o viés decorrente da definição dos critérios de avaliação pelo pesquisador foi identificado como uma possível limitação.

Acredita-se que esta pesquisa contribui significativamente para o avanço da área de automação de testes e abre caminho para futuras investigações. A automação completa do processo de teste, incluindo a funcionalidade para gerar o código executável associado a cada passo do cenário Gherkin, foi indicada como uma das direções promissoras. O refinamento e a personalização dos LLM, adaptando-os a domínios específicos e às necessidades dos usuários, também foram considerados cruciais para aprimorar a qualidade e a precisão dos cenários gerados. Outras áreas de pesquisa incluem a expansão da abordagem para lidar com casos de teste complexos, a geração de cenários alternativos e a integração com ferramentas de CI/CD, incorporando a geração automatizada de cenários e código de teste em pipelines para garantir a execução automática dos testes a cada mudança.

5.1 Contribuições

Neste trabalho foram definidas oito questões de pesquisa para investigar a geração de cenários Gherkin com o uso de LLM. A primeira, **RQ1**: *Qual técnica de prompting (zero-shot, one-shot ou few-shot) gera cenários Gherkin mais alinhados às boas práticas do BDD?* Foi identificado que a técnica *zero-shot*, que utiliza apenas instruções sem exemplos, apresentou o melhor desempenho. Esse resultado foi evidenciado pela superioridade do LLM Gemini, que destacou-se em qualidade e consistência, obtendo a maior média $(0,84)$ e o menor desvio padrão $(0,08)$ entre os modelos avaliados. A análise estatística pelo teste de Kruskal-Wallis reforçou a ausência de diferenças significativas entre os modelos, mas o desempenho do Gemini sugere maior alinhamento às boas práticas do BDD e ao *ground truth*.

A segunda questão, **RQ2**: *Quais medidas melhor avaliam a qualidade dos cenários Gherkin gerados automaticamente e diferenciam as abordagens de prompting?* Três medidas foram utilizadas: Distância de Manhattan, para capturar discrepâncias estruturais; ME-

TEOR, para avaliar a similaridade semântica com sinônimos e fluidez textual; e BERTScore, para analisar nuances contextuais com embeddings. Essas medidas foram escolhidas por sua capacidade de avaliar tanto a estrutura quanto a semântica dos cenários, diferenciando com clareza as abordagens de prompting.

Para a **RQ3**: *Quais critérios devem ser considerados para selecionar e organizar uma base de cenários de teste representativa?* Foi estabelecido que cenários curtos ou com descrições não informativas deveriam ser removidos. Cenários relacionados a funcionalidades críticas, como login e cadastro, foram priorizados, enquanto redundâncias foram eliminadas para garantir diversidade. Esses critérios garantiram uma base de dados representativa e adequada para avaliação dos LLM.

RQ4: *Qual medida é mais adequada para avaliar a eficácia dos LLM na transformação de descrições livres em cenários Gherkin?* A escolha do METEOR como a medida mais apropriada foi sustentada por sua alta correlação com a Avaliação Qualitativa ($r = 0,88$), superando significativamente outras medidas, como o BERTScore ($r = 0,39$) e a Manhattan ($r = -0,52$). Além disso, o METEOR demonstrou menor variabilidade nos resultados, evidenciada por seu desvio padrão ($0,06$), reforçando sua consistência. Essa medida destacou-se por capturar com maior precisão tanto os aspectos semânticos quanto os estruturais dos cenários gerados, consolidando-se como a opção ideal para avaliar a eficácia dos LLM neste estudo.

Para a **RQ5**: *Quais critérios devem ser aplicados para selecionar a medida mais apropriada?* A relevância da medida foi associada à sua capacidade de correlacionar-se com avaliações qualitativas, além de capturar discrepâncias estruturais, similaridade semântica e precisão textual de forma consistente.

Na **RQ6**: *Como diferentes LLM se comparam na geração de cenários de teste Gherkin?* Observou-se que o Gemini apresentou o melhor desempenho em qualidade na técnica *zero-shot*, destacando-se como o modelo mais alinhado ao *ground truth*. O GPT-4o Mini e o LLaMA 3, por sua vez, demonstraram maior consistência com a técnica *few-shot*. No entanto, o teste estatístico revelou que não há diferenças significativas entre os modelos em termos de desempenho geral, indicando que as variações observadas podem ser atribuídas ao acaso. De forma geral, a técnica *zero-shot* foi identificada como a abordagem mais eficaz para gerar cenários Gherkin de qualidade, demonstrando sucesso consistente em todos os

LLM analisados.

RQ7: *Como as técnicas de prompting impactam a qualidade e consistência dos cenários gerados?* A técnica *zero-shot* destacou-se como a mais eficaz em termos de qualidade, especialmente em LLM como Gemini e Phi-3, que apresentaram maior similaridade com os cenários de referência, evidenciando uma compreensão da tarefa sem a necessidade de exemplos adicionais. Em contrapartida, as técnicas *one-shot* e *few-shot* mostraram-se úteis para melhorar a consistência dos resultados em alguns LLM, oferecendo maior previsibilidade e estabilidade ao fornecerem um guia mais detalhado nos prompts, mesmo que não tenham superado a qualidade da *zero-shot* nesses modelos.

Por fim, a **RQ8**: *Como a variabilidade dos resultados dos LLM afeta a consistência dos cenários de teste Gherkin?* Foi evidenciado que a variabilidade das técnicas impacta diretamente a confiabilidade dos cenários gerados, com *zero-shot* exibindo maior variabilidade em alguns LLM e *few-shot* mostrando maior estabilidade. A inconsistência nas respostas pode dificultar a manutenção e a interpretação dos testes, exigindo uma escolha criteriosa da técnica e do LLM.

5.2 Trabalhos Futuros

Esta dissertação explorou a geração de cenários de teste Gherkin a partir de descrições em linguagem natural utilizando LLM, abrindo caminho para diversas investigações futuras. Para estruturar melhor as propostas, os trabalhos futuros foram organizados em três horizontes temporais: curto, médio e longo prazo, considerando sua viabilidade e conexão com as limitações identificadas no estudo.

No curto prazo, as investigações podem focar no aprimoramento das técnicas de *prompting* e na comparação de diferentes abordagens para geração de cenários. Como observado no estudo, a técnica *zero-shot* apresentou o melhor desempenho médio, mas a *one-shot* mostrou maior consistência. Portanto, investigar como ajustes finos nos modelos e a combinação de técnicas de *prompting* podem melhorar a qualidade e a consistência dos cenários gerados é uma direção promissora. Além disso, a definição de indicadores específicos, como precisão semântica, cobertura de casos de teste e tempo de execução, pode ajudar a medir o impacto dessas melhorias de forma objetiva e replicável.

Outro ponto relevante é a integração com ferramentas de CI/CD para incorporar a geração automatizada de cenários e código de teste em pipelines de desenvolvimento. Essa abordagem pode ser viabilizada pela integração com frameworks de teste, como Cucumber e Selenium, aliada ao uso de LLM especializados em geração de código.

No médio prazo, o refinamento e a personalização dos LLM para domínios específicos são áreas promissoras. A adaptação às nuances do domínio da aplicação e às necessidades específicas dos usuários pode ser realizada por meio de técnicas de *fine-tuning* com dados de treinamento específicos, além do desenvolvimento de mecanismos de *feedback* para aprendizado contínuo. Por exemplo, a personalização para domínios como aplicações web, jogos ou sistemas embarcados pode aumentar a precisão e a relevância dos cenários gerados.

Além disso, a expansão da abordagem para lidar com casos de teste mais complexos, como fluxos de usuário com múltiplas etapas e cenários alternativos, é uma direção importante. Pesquisas futuras podem investigar técnicas avançadas de processamento de linguagem natural para extrair informações relevantes e métodos eficazes para representar e estruturar casos de teste com maior complexidade.

No longo prazo, a automação completa do processo de teste, desde a especificação até a execução, é uma visão ambiciosa e promissora. Isso inclui a geração de código executável associado a cada passo do cenário Gherkin, bem como a integração com ferramentas de CI/CD para garantir a execução contínua dos testes a cada mudança no sistema.

Outra direção de longo prazo é a geração de testes baseada em modelos, utilizando diagramas como UML como entrada, e a automação de testes para interfaces gráficas de usuário (GUI), gerando cenários de teste para aplicações com interfaces complexas. Essa abordagem pode ser especialmente útil em domínios como desenvolvimento de jogos e sistemas embarcados, onde a interação com a interface é crítica.

Por fim, a geração de cenários alternativos, como cenários de sucesso, falha e borda, é crucial para aumentar a cobertura e robustez dos testes. A integração com ferramentas de CI/CD pode ser explorada para incorporar a geração automatizada de cenários e código de teste em pipelines, garantindo a execução contínua dos testes a cada mudança no sistema.

Bibliografia

[1] Abdin, M., Aneja, J., Awadalla, H., et al. Phi-3 technical report: A highly capable language model locally on your phone, 2024. URL https://arxiv.org/abs/2404.14219.

[2] Adzic, G. *Specification by Example: How Successful Teams Deliver the Right Software*. Manning Publications, 2011. ISBN 978-1617290084. URL https://www.manning.com/books/specification-by-example.

[3] Alferez, M., Pastore, F., Sabetzadeh, M., Briand, L., e Riccardi, J.-R. Bridging the gap between requirements modeling and behavior-driven development. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 239–249, 2019. doi: 10.1109/MODELS.2019.00008.

[4] Alrajeh, D. e Williams, D. Prioritization of test cases in secure systems. In *2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 13–18. IEEE, 2018. doi: 10.1109/ISSREW.2018.8389562.

[5] Arredondo-Reyes, V. M., Domínguez-Isidro, S., Sánchez-García, J., e Ocharán-Hernández, J. O. Benefits and challenges of the behavior-driven development: A systematic literature review. In *2023 11th International Conference in Software Engineering Research and Innovation (CONISOFT)*, pages 45–54, 2023. doi: 10.1109/CONISOFT58849.2023.00016.

[6] Atil, B., Chittams, A., Fu, L., Ture, F., Xu, L., e Baldwin, B. Llm stability: A detailed analysis with some surprises, 2024. URL https://arxiv.org/abs/2408.04667.

[7] Banerjee, S. e Lavie, A. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Goldstein, J., Lavie, A., Lin, C.-Y., e Voss, C., editors, *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL https://aclanthology.org/W05-0909.

[8] Baqar, M. e Khanda, R. The future of software testing: Ai-powered test case generation and validation. *arXiv preprint*, 2409.05808v1, September 2024. URL https://arxiv.org/abs/2409.05808.

[9] Bezsmertnyi, O., Golian, N., Golian, V., e Afanasieva, I. Behavior driven development approach in the modern quality control process. *IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC S&T)*, pages 215–218, 2020. doi: 10.1109/PICST51311.2020.9467891.

[10] Blackwell, R. E., Barry, J., e Cohn, A. G. Towards reproducible llm evaluation: Quantifying uncertainty in llm benchmark scores. *arXiv preprint arXiv:2410.03492*, 2024. URL https://arxiv.org/abs/2410.03492.

[11] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

[12] Cavalcante, M. G. e Sales, J. I. The behavior driven development applied to the software quality test:. In *2019 14th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–4, 2019. doi: 10.23919/CISTI.2019.8760965.

[13] Chamieh, I., Zesch, T., e Giebermann, K. Llms in short answer scoring: Limitations and promise of zero-shot and few-shot approaches. In *Proceedings of the 19th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2024)*. Association for Computational Linguistics, 2024. URL https://arxiv.org/abs/2406.16143.

[14] Chandorkar, A., Patkar, N., Di Sorbo, A., e Nierstrasz, O. An exploratory study on the usage of gherkin features in open-source projects. In *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 1159–1166, 2022. doi: 10.1109/SANER53432.2022.00134.

[15] Chelimsky, D. et al. *The RSpec Book: Behaviour-Driven Development with RSpec, Cucumber, and Friends*. Pragmatic Bookshelf, 2010.

[16] Chen, X., Ye, J., Zu, C., Xu, N., Zheng, R., Peng, M., Zhou, J., Gui, T., Zhang, Q., e Huang, X. How robust is gpt-3.5 to predecessors? a comprehensive study on language understanding tasks, 2023. URL `https://arxiv.org/abs/2303.00293`.

[17] Chiavegatto, R. B., Silva, L. V., Andréia, V., e Malvezzi, W. R. Desenvolvimento orientado a comportamento com testes automatizados utilizando jbehave e selenium. In *Anais do Encontro Regional de Informática e Sistemas de Informação*, 2018. URL `https://www.academia.edu/31492969/Desenvolvimento_Orientado_a_Comportamento_com_Testes_Automatizados_utilizando_JBehave_e_Selenium`.

[18] Cohen, J. *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum Associates, Hillsdale, NJ, 2nd edition, 1988.

[19] Cui, J. A comparative study on the impact of test-driven development (tdd) and behavior-driven development (bdd) on enterprise software delivery effectiveness. *IEEE*, 2024. URL `https://arxiv.org/abs/2411.04141`.

[20] de Carvalho Moraes, L., Silvério, I. C., Marques, R. A. S., de Castro Anaia, B., de Paula, D. F., de Faria, M. C. S., Cleveston, I., de Santana Correia, A., e Freitag, R. M. K. Análise de ambiguidade linguística em modelos de linguagem de grande escala (llms), 2024. URL `https://arxiv.org/abs/2404.16653`.

[21] de Souza Filho, E. D. *Uma Abordagem para Recomendação de Casos de Teste em Projetos Ágeis Baseados no Scrum*. PhD thesis, Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Campina Grande, Campina Grande, Brasil,

2021. URL http://dspace.sti.ufcg.edu.br:8080/jspui/handle/riufcg/21343.

[22] De Wynter, A., Wang, X., Sokolov, A., Gu, Q., e Chen, S.-Q. An evaluation on large language model outputs: Discourse and memorization. *Natural Language Processing Journal*, 4, 2023.

[23] Espejel, J. L., Ettifouri, E. H., Alassan, M. S. Y., Chouham, E. M., e Dahhane, W. Gpt-3.5, gpt-4, or bard? evaluating llms reasoning ability in zero-shot setting and performance boosting through prompts, 2023. URL https://arxiv.org/abs/2305.12477.

[24] Farooq, M. S., Omer, U., Ramzan, A., Rasheed, M. A., e Atal, Z. Behavior driven development: A systematic literature review. *IEEE Access*, 11:88007–88019, 2023. doi: 10.1109/ACCESS.2023.3302356.

[25] Farooq, M. S., Omer, U., Ramzan, A., Rasheed, M. A., e Atal, Z. Behavior driven development: A systematic literature review. *IEEE Access*, 11:88008–88024, 2023. doi: 10.1109/ACCESS.2023.3302356.

[26] Gao, J., Zhang, T., e Xu, C. A generative approach to zero-shot and few-shot action recognition. *arXiv preprint arXiv:1801.09086*, 2018. URL https://arxiv.org/abs/1801.09086.

[27] Grattafiori, A. e et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

[28] Grattafiori, A., Dubey, A., Jauhri, A., et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

[29] Guo, C., Pleiss, G., Sun, Y., e Weinberger, K. Q. On calibration of modern neural networks. *Proceedings of the 34th International Conference on Machine Learning*, 70: 1321–1330, 2017.

[30] Gupta, A., Poels, G., e Bera, P. Generating multiple conceptual models from behavior-driven development scenarios. *Data & Knowledge Engineering*, 145:102141, 2023. doi: 10.1016/j.datak.2023.102141.

[31] Hadi, M. U., Al-Tashi, Q., Qureshi, R., Shah, A., Muneer, A., Irfan, M., Zafar, A., Shaikh, M. B., Akhtar, N., Wu, J., e Mirjalili, S. A survey on large language models: Applications, challenges, limitations, and practical usage, 2023.

[32] Hadi, M. U., Tashi, Q. A., Qureshi, R., Shah, A., Muneer, A., Irfan, M., et al. Large language models: A comprehensive survey of its applications, challenges, limitations, and future prospects. TechRxiv, 2023.

[33] Hanna, M. e Bojar, O. A fine-grained analysis of BERTScore. In Barrault, L., Bojar, O., Bougares, F., Chatterjee, R., Costa-jussa, M. R., Federmann, C., Fishel, M., Fraser, A., Freitag, M., Graham, Y., Grundkiewicz, R., Guzman, P., Haddow, B., Huck, M., Yepes, A. J., Koehn, P., Kocmi, T., Martins, A., Morishita, M., e Monz, C., editors, *Proceedings of the Sixth Conference on Machine Translation*, pages 507–517, Online, November 2021. Association for Computational Linguistics. URL `https://aclanthology.org/2021.wmt-1.59`.

[34] Irshad, M., Britto, R., e Petersen, K. Adapting behavior driven development (bdd) for large-scale software systems. *Journal of Systems and Software*, 177:110944, 2021. doi: 10.1016/j.jss.2021.110944.

[35] Irshad, M., Börstler, J., e Petersen, K. Supporting refactoring of bdd specifications—an empirical study. *Information and Software Technology*, 141:106717, 2022. doi: 10.1016/j.infsof.2021.106717.

[36] Jiang, Z., Araki, J., Ding, H., e Neubig, G. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977, 2021.

[37] Jin, H., Huang, L., Cai, H., Yan, J., Li, B., e Chen, H. From llms to llm-based agents for software engineering: A survey of current, challenges and future, 2024. URL `https://arxiv.org/abs/2408.02479`.

[38] Johnson, M. e Zhang, M. Examining the responsible use of zero-shot ai approaches to scoring essays. *Scientific Reports*, 14:30064, 2024. doi: 10.

1038/s41598-024-79208-2. URL https://www.nature.com/articles/s41598-024-79208-2#citeas.

[39] Kang, S., Yoon, J., e Yoo, S. Large language models are few-shot testers: Exploring llm-based general bug reproduction. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, 2023.

[40] Karpurapu, S., Myneni, S., Nettur, U., Gajja, L. S., Burke, D., Stiehm, T., e Payne, J. Comprehensive evaluation and insights into the use of large language models in the automation of behavior-driven development acceptance test formulation. *IEEE Access*, 12:58715–58730, 2024. doi: 10.1109/ACCESS.2024.3391815.

[41] Kaster, M., Zhao, W., e Eger, S. Global explainability of bert-based evaluation metrics by disentangling along linguistic factors. *arXiv preprint arXiv:2110.04399*, 2021. URL https://arxiv.org/abs/2110.04399.

[42] Kim, D.-W. e Lee, K. H. A new validity measure for fuzzy c-means clustering. *arXiv preprint arXiv:2407.06774*, 2024. URL http://arxiv.org/abs/2407.06774.

[43] Lafi, M., Alrawashed, T., e Hammad, A. M. Automated test cases generation from requirements specification. *2021 International Conference on Information Technology (ICIT)*, pages 851–857, 2021. doi: 10.1109/ICIT52682.2021.9491761.

[44] Lavie, A. e Denkowski, M. J. The meteor metric for automatic evaluation of machine translation. *Machine Translation*, 23(2-3):105–115, 2009.

[45] Lee, G.-G., Latif, E., Shi, L., e Zhai, X. Gemini pro defeated by gpt-4v: Evidence from education, 2023. URL https://arxiv.org/abs/2401.08660.

[46] Lenka, R. K., Kumar, S., e Mamgain, S. Behavior driven development: Tools and challenges. pages 1032–1036, 2018. doi: 10.1109/ICACCCN.2018.8748756.

[47] Liu, Y., Han, T., Ma, S., Zhang, J., Yang, Y., Tian, J., He, H., Li, A., He, M., Liu, Z., Wu, Z., Zhao, L., Zhu, D., Li, X., Qiang, N., Shen, D., Liu, T., e Ge, B. Summary of chatgpt-related research and perspective towards the future of large language models. *Meta-Radiology*, 1, 2023.

[48] Mahalakshmi, G. e Vani, V. Theoretical verification of test cases for behavior driven development. *Second International Conference on Recent Trends and Challenges in Computational Models*, pages 307–310, 2017. doi: 10.1109/ICRTCCM.2017.83.

[49] Malkauthekar, M. D. Analysis of euclidean distance and manhattan distance measure in face recognition. In *Third International Conference on Computational Intelligence and Information Technology (CIIT 2013)*, pages 503–507, 2013. doi: 10.1049/cp.2013.2636.

[50] Minaee, S., Mikolov, T., Nikzad, N., Chenaghlu, M., Socher, R., Amatriain, X., e Gao, J. Large language models: A survey, 2024. URL `https://arxiv.org/abs/2402.06196`.

[51] Mughal, A. H. Advancing bdd software testing: Dynamic scenario re-usability and step auto-complete for cucumber framework, 2024. URL `https://arxiv.org/abs/2402.15928`.

[52] North, D. Introducing bdd. `https://dannorth.net/introducing-bdd/`, 2006. Acesso em: 16 de agosto de 2023.

[53] Oliveira, G. e Marczak, S. On the empirical evaluation of bdd scenarios quality: Preliminary findings of an empirical study. In *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, pages 299–302, 2017. doi: 10.1109/REW.2017.62.

[54] Oliveira, G., Marczak, S., e Moralles, C. How to evaluate bdd scenarios' quality? pages 181–190, 2019. doi: 10.1145/3350768.3351301.

[55] Patkar, N., Chiş, A., Stulova, N., e Nierstrasz, O. Interactive behavior-driven development: a low-code perspective. In *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pages 128–137, 2021. doi: 10.1109/MODELS-C53483.2021.00024.

[56] Pimentel, M. A., Christophe, C., Raha, T., Munjal, P., Kanithi, P. K., e Khan, S. Beyond metrics: A critical analysis of the variability in large language model evaluation frameworks, 2024. URL `https://arxiv.org/abs/2407.21072`.

[57] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., e Sutskever, I. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[58] Raharjana, I. K., Harris, F., e Justitia, A. Tool for generating behavior-driven development test-cases. *Journal of Information Systems Engineering and Business Intelligence*, 6(1):27–36, 2020. doi: 10.20473/jisebi.6.1.27-36. URL https://doi.org/10.20473/jisebi.6.1.27-36.

[59] Raharjana, I. K., Harris, F., e Justitia, A. Tool for generating behavior-driven development test-cases. *Journal of Information Systems Engineering and Business Intelligence*, 6(1):27–36, 2020. doi: 10.20473/jisebi.6.1.27-36.

[60] Rahman, S., Khan, S., e Porikli, F. A unified approach for conventional zero-shot, generalized zero-shot and few-shot learning. *arXiv preprint arXiv:1706.08653*, 2017. URL https://arxiv.org/abs/1706.08653.

[61] Ramos, F. B. A. *Recomendação de Requisitos Não Funcionais em Projetos Ágeis Baseados em Scrum*. PhD thesis, Universidade Federal de Campina Grande, Campina Grande, Brasil, fevereiro 2019. URL http://dspace.sti.ufcg.edu.br:8080/jspui/handle/riufcg/10743.

[62] Rillig, M. C., Ågerstrand, M., Bi, M., Gould, K. A., e Sauerland, U. Risks and benefits of large language models for the environment. *Environmental Science & Technology*, 57:3464–3466, 2023.

[63] Saadany, H. e Orasan, C. Bleu, meteor, bertscore: Evaluation of metrics performance in assessing critical translation errors in sentiment-oriented text. *arXiv preprint arXiv:2109.14250*, 2021. URL https://arxiv.org/abs/2109.14250.

[64] Saka, A., Taiwo, R., Saka, N., Salami, B. A., Ajayi, S., Akande, K., e Kazemi, H. Gpt models in construction industry: Opportunities, limitations, and a use case validation. *Developments in the Built Environment*, 17:100300, 2024. ISSN 2666-1659. doi: https://doi.org/10.1016/j.dibe.2023.100300. URL https://www.sciencedirect.com/science/article/pii/S2666165923001825.

[65] Sallou, J., Durieux, T., e Panichella, A. Breaking the silence: the threats of using llms in software engineering. In *New Ideas and Emerging Results (ICSE-NIER'24)*, pages 1–5. ACM, 2024. doi: 10.1145/3639476.3639764.

[66] Shah, H., Ahmad, A. J., e Khaliq, S. A. R. Prioritizing user-session-based test cases for web applications testing. *IEEE Access*, 10:10475–10488, 2021. doi: 10.1109/ACCESS.2021.9698145.

[67] Silva, E. C. d. M. Asymptotic behavior of the manhattan distance in n-dimensions: Estimating multidimensional scenarios in empirical experiments. *arXiv preprint arXiv:2406.15441*, 2024. URL `https://arxiv.org/abs/2406.15441`.

[68] Smart, J. *BDD in Action*. Manning Publications, 2014. ISBN 9781617291654. URL `https://www.oreilly.com/library/view/bdd-in-action/9781617291654/`.

[69] Solis, C. e Wang, X. A study of the characteristics of behaviour driven development. In *2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 383–387, 2011. doi: 10.1109/SEAA.2011.76.

[70] Sun, Z. A short survey of viewing large language models in legal aspect. Renmin University of China, arXiv, mar 2023. URL `https://arxiv.org/abs/2303.09136`. arXiv:2303.09136v1 [cs.CL].

[71] Tsigkanos, C., Rani, P., Müller, S., e Kehrer, T. Large language models: The next frontier for variable discovery within metamorphic testing? In *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2023.

[72] Wynne, M., Hellesøy, A., e Tooke, S. *The Cucumber Book, Second Edition: Behaviour-Driven Development for Testers and Developers*. Pragmatic Bookshelf, Raleigh, 2 edition, 2017. ISBN 9781680502381.

[73] Xia, C. S., Deng, Y., Dunn, S., e Zhang, L. Agentless: Demystifying llm-based software engineering agents, 2024. URL `https://arxiv.org/abs/2407.01489`.

[74] Xia, Y., Araujo, P. H. L. d., Zaporojets, K., e Roth, B. Influences on llm calibration: A study of response agreement, loss functions, and prompt styles. *arXiv preprint arXiv:2501.03991*, 2024.

[75] Yan, Y., Li, B., Feng, J., Du, Y., Lu, Z., e Huang, M. Research on the impact of trends related to chatgpt. *Procedia Computer Science*, 221:1284–1291, 2023.

[76] Yang, C., Wang, X., Lu, Y., Le, Q. V., Zhou, D., Chen, X., e Liu, H. Large language models as optimizers, sep 2023. URL `https://arxiv.org/abs/2309.03409`. arXiv:2309.03409v1 [cs.LG].

[77] Zhang, M., He, J., Ji, T., e Lu, C.-T. Don't go to extremes: Revealing the excessive sensitivity and calibration limitations of llms in implicit hate speech detection. *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 12073–12086, 2024.

[78] Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., e Artzi, Y. Bertscore: Evaluating text generation with bert, 2020. URL `https://arxiv.org/abs/1904.09675`.

Apêndice A

Prompts utilizados

A.1 One-shot

O prompt apresentado a seguir foi utilizado para a geração de cenários BDD na técnica one-shot. Ele inclui um exemplo detalhado que serve como referência para a estrutura e estilo esperados, garantindo que a saída gerada pelos modelos esteja alinhada às boas práticas do BDD.

Prompt Template (One-Shot)

Converta a seguinte descrição de caso de teste em um único cenário BDD, usando a sintaxe estrita de Gherkin. Certifique-se de que a saída contenha apenas a sintaxe de Gherkin para o cenário, sem comentários, explicações ou a palavra "Feature". Use o português para as descrições dos casos de teste e detalhes do cenário, mas mantenha as palavras-chave do Gherkin em inglês.

Exemplo

Descrição do caso de teste:

Usuário tenta login com credenciais inválidas.

Scenario: Login com senha inválida

Given o usuário está na página de login

And o usuário insere um nome de usuário válido

When o usuário insere uma senha inválida e clica no botão de login

Then o sistema exibe uma mensagem de erro indicando que a senha está incorreta

And o campo de senha é limpo

Agora, converta a seguinte descrição de caso de teste em exatamente um cenário BDD usando a sintaxe estrita de Gherkin. A saída deve seguir exatamente o formato do exemplo fornecido e não conter nada além do cenário BDD. Somente as palavras-chave do Gherkin devem estar em inglês; todo o outro texto deve estar em português.

Descrição do Caso de Teste: {test_case_description}

Para um bom cenário BDD, certifique-se de declarar claramente o valor de negócio ou resultado esperado e mantenha o foco em uma única ação e seu resultado. Use apenas os passos essenciais (Given, When, Then, And) de forma clara e declarativa, evitando detalhes de implementação e repetições desnecessárias. Garanta que os cenários sejam independentes, utilizem uma terminologia de negócios consistente sem jargões técnicos e que os passos sejam escritos em terceira pessoa para evitar múltiplas interpretações. Certifique-se de que os cenários BDD tenham indentação consistente de dois espaços para cada passo sob 'Scenario', sem linhas em branco entre os passos, e uma linha em branco separando diferentes cenários.

Siga esta estrutura para a saída:

Scenario: [Descrição do Cenário]

Given [algum contexto inicial]

And [mais algum contexto, se houver]

When [uma ação é realizada]

Then [um conjunto específico de resultados deve ocorrer]

And [outro resultado, se houver]

A resposta deve conter estritamente apenas a sintaxe válida de Gherkin e evitar qualquer informação ou comentário extra. A resposta deve conter apenas o texto BDD, sem formatação ou texto adicional (por exemplo, sem 'gherkin"'), e deve ser apenas um cenário BDD.

A.2 Few-shot

O prompt apresentado a seguir foi utilizado para a geração de cenários BDD na técnica few-shot. Ele inclui três exemplos detalhados que servem como referência para a estrutura e estilo esperados, garantindo que a saída gerada pelos modelos esteja alinhada às boas práticas do BDD.

Prompt Template (Few-Shot)

Converta a seguinte descrição de caso de teste em um único cenário BDD, usando a sintaxe estrita de Gherkin. Certifique-se de que a saída contenha apenas a sintaxe de Gherkin para o cenário, sem comentários, explicações ou a palavra "Feature". Use o português para as descrições dos casos de teste e detalhes do cenário, mas mantenha as palavras-chave do Gherkin em inglês.

Exemplo 1

Descrição do caso de teste:

Usuário tenta login com credenciais inválidas.

Scenario: Login com senha inválida

Given o usuário está na página de login

And o usuário insere um nome de usuário válido

When o usuário insere uma senha inválida e clica no botão de login

Then o sistema exibe uma mensagem de erro indicando que a senha está incorreta

And o campo de senha é limpo

Exemplo 2

Descrição do caso de teste:

Usuário tenta redefinir a senha esquecida.

Scenario: Redefinição de senha com e-mail válido

Given o usuário está na página de redefinição de senha

And o usuário insere um endereço de e-mail registrado

When o usuário clica no botão de enviar

Then o sistema exibe uma mensagem indicando que um link de redefinição de senha foi enviado para o e-mail do usuário

And o usuário é redirecionado para a página de login

Exemplo 3

Descrição do caso de teste:

Usuário adiciona um item ao carrinho de compras.

Scenario: Adicionar item ao carrinho de compras

Given o usuário está na página de detalhes de um produto

And o produto está disponível em estoque

When o usuário clica no botão "Adicionar ao carrinho"

Then o item é adicionado ao carrinho de compras

And o sistema exibe uma mensagem de confirmação "Item adicionado ao carrinho com sucesso"

And o ícone do carrinho de compras é atualizado para refletir o novo item

Prompt Template (Few-Shot)

Agora, converta a seguinte descrição de caso de teste em exatamente um cenário BDD usando a sintaxe estrita de Gherkin. A saída deve seguir exatamente o formato do exemplo fornecido e não conter nada além do cenário BDD. Somente as palavras-chave do Gherkin devem estar em inglês; todo o outro texto deve estar em português.

Descrição do Caso de Teste: {test_case_description}

Para um bom cenário BDD, certifique-se de declarar claramente o valor de negócio ou resultado esperado e mantenha o foco em uma única ação e seu resultado. Use apenas os passos essenciais (Given, When, Then, And) de forma clara e declarativa, evitando detalhes de implementação e repetições desnecessárias. Garanta que os cenários sejam independentes, utilizem uma terminologia de negócios consistente sem jargões técnicos e que os passos sejam escritos em terceira pessoa para evitar múltiplas interpretações. Certifique-se de que os cenários BDD tenham indentação consistente de dois espaços para cada passo sob 'Scenario', sem linhas em branco entre os passos, e uma linha em branco separando diferentes cenários.

Siga esta estrutura para a saída:

Scenario: [Descrição do Cenário]

    Given [algum contexto inicial]

    And [mais algum contexto, se houver]

    When [uma ação é realizada]

    Then [um conjunto específico de resultados deve ocorrer]

    And [outro resultado, se houver]

A resposta deve conter estritamente apenas a sintaxe válida de Gherkin e evitar qualquer informação ou comentário extra. A resposta deve conter apenas o texto BDD, sem formatação ou texto adicional (por exemplo, sem 'gherkin"'), e deve ser apenas um cenário BDD.

Apêndice B

Tabelas

B.1 Tabela de Avaliação Qualitativa

A Tabela B.1 apresenta a avaliação qualitativa final dos 10 casos de teste selecionados aleatoriamente para serem avaliados, considerando três critérios principais: Estrutura, Semântica e Detalhes, cada um com pesos específicos. A nota final foi calculada utilizando uma média ponderada baseada nesses pesos.

Tabela B.1: Avaliação Final

Caso	Estrutura (Nota) - Peso 4	Semântica (Nota) - Peso 4	Detalhes (Nota) - Peso 2	Nota Final
O campo senha deve seguir regra <regra>	7	7,5	7	7,2
Cadastro de perfil com identificador existente	8	8,5	7	8
Realizar ação sem ter feito login	8,5	9	9	8,8
Editar informações do condomínio com sucesso	7,5	8,3	7	7,72
Cancelar cadastro de área comum com sucesso	7	7,2	7	7,08
Cancelar cadastro de peril com sucesso	7,8	8,5	7	7,92
Executar ação sem ter feito login	8	9	7,5	8,3
Editar reserva deixando os campos obrigatórios do formulário em branco	7,3	8	7	7,52
Registrar com senha diferente da confirmação	7	8	8,5	7,7
Cadastro de sprint com data inicial maior que data final	8,7	8	8	8,28

B.2 Tabela de Pontuações METEOR para casos selecionados

A Tabela B.2 apresenta uma análise quantitativa dos 10 primeiros casos avaliados, destacando as métricas de Manhattan, METEOR e BERTScore, utilizadas para medir a similaridade entre os cenários gerados e os casos de referência.

Tabela B.2: Análise dos 10 Primeiros Casos com Métricas Quantitativas.

Caso	Modelo	Técnica	Manhattan	METEOR	BERTScore
O campo senha deve seguir regra <regra>	Gemini	few-shot	45	0.69	46.34
Cadastro de perfil com identificador existente	Llama 3	zero-shot	24	0.74	76.08
Realizar ação sem ter feito login	Gpt-3.5-turbo	one-shot	24	0.87	59.59
Editar informações do condomínio com sucesso	Gpt-4-turbo	zero-shot	38	0.78	54.05
Cancelar cadastro de área comum com sucesso	Phi-3	few-shot	31	0.70	54.94
Cancelar cadastro de peril com sucesso	Gpt-3.5-turbo	zero-shot	34	0.81	58.06
Executar ação sem ter feito login	Gpt-4o-mini	few-shot	36	0.81	54.70
Editar reserva deixando os campos obrigatórios do formulário em branco	Phi-3	one-shot	45	0.70	54.89
Registrar com senha diferente da confirmação	Gpt-4o-mini	few-shot	37	0.79	67.78
Cadastro de sprint com data inicial maior que data final	Gemini	one-shot	38	0.79	63.48

Apêndice C

Dados de Variabilidade

Este capítulo apresenta todos os dados de variabilidade coletados durante o estudo, considerando as 5 execuções realizadas. A análise de variabilidade é fundamental para compreender a consistência e a dispersão dos resultados obtidos.

Os dados estão organizados em casos numerados de 1 a 10, que correspondem, em ordem, aos cenários descritos no arquivo `casos_selecionados.csv`[1]. Cada caso foi executado múltiplas vezes para garantir a confiabilidade dos resultados, e as medidas de variabilidade foram calculadas com base nessas execuções.

A seguir, são apresentadas tabelas e gráficos que detalham a variabilidade observada em cada caso, incluindo medidas estatísticas como média, desvio padrão e coeficiente de variação. Esses dados fornecem insights sobre a estabilidade e a reprodutibilidade dos cenários testados.

[1]Disponível em: `https://github.com/hiagonfs/bdd-scenario-evaluation/blob/main/csv_files/casos_selecionados.csv`

C.1 Gemini

Tabela C.1: Resultados da Variabilidade METEOR - Técnica Zero-Shot

Caso Base	**Média**	**Desvio Padrão**	**CV (%)**
Caso 1	0.86	0.00	0.00%
Caso 2	0.94	0.00	0.00%
Caso 3	0.92	0.00	0.00%
Caso 4	0.93	0.00	0.48%
Caso 5	0.84	0.00	0.00%
Caso 6	0.71	0.00	0.00%
Caso 7	0.75	0.00	0.00%
Caso 8	0.79	0.01	0.69%
Caso 9	0.81	0.00	0.00%
Caso 10	0.83	0.01	1.08%

Tabela C.2: Resultados da Variabilidade METEOR - Técnica One-Shot

Caso Base	**Média**	**Desvio Padrão**	**CV (%)**
Caso 1	0.75	0.05	6.04%
Caso 2	0.82	0.04	5.36%
Caso 3	0.81	0.04	5.38%
Caso 4	0.84	0.09	10.42%
Caso 5	0.82	0.06	7.11%
Caso 6	0.60	0.07	11.67%
Caso 7	0.76	0.06	8.42%
Caso 8	0.82	0.05	6.04%
Caso 9	0.79	0.05	6.89%
Caso 10	0.78	0.04	5.07%

Tabela C.3: Resultados da Variabilidade METEOR - Técnica Few-Shot

Caso Base	**Média**	**Desvio Padrão**	**CV (%)**
Caso 1	0.68	0.03	3.71%
Caso 2	0.76	0.04	5.01%
Caso 3	0.64	0.13	20.88%
Caso 4	0.82	0.03	3.93%
Caso 5	0.82	0.06	7.41%
Caso 6	0.64	0.04	5.51%
Caso 7	0.65	0.04	6.27%
Caso 8	0.81	0.01	0.68%
Caso 9	0.79	0.01	0.90%
Caso 10	0.84	0.08	9.14%

C.2 Llama 3

Tabela C.4: Resultados da Variabilidade METEOR - LLaMA 3 (Zero-Shot)

Caso Base	**Média**	**Desvio Padrão**	**CV (%)**
Caso 1	0.71	0.02	2.90%
Caso 2	0.72	0.02	3.22%
Caso 3	0.74	0.00	0.00%
Caso 4	0.87	0.02	2.21%
Caso 5	0.88	0.02	2.03%
Caso 6	0.78	0.05	6.15%
Caso 7	0.52	0.03	5.97%
Caso 8	0.88	0.02	1.97%
Caso 9	0.74	0.03	3.98%
Caso 10	0.75	0.01	1.12%

Tabela C.5: Resultados da Variabilidade METEOR - LLaMA 3 (One-Shot)

Caso Base	**Média**	**Desvio Padrão**	**CV (%)**
Caso 1	0.66	0.04	6.70%
Caso 2	0.64	0.04	6.99%
Caso 3	0.72	0.00	0.00%
Caso 4	0.85	0.00	0.00%
Caso 5	0.79	0.01	1.38%
Caso 6	0.72	0.02	2.12%
Caso 7	0.62	0.06	10.36%
Caso 8	0.85	0.00	0.00%
Caso 9	0.50	0.01	2.62%
Caso 10	0.67	0.04	5.74%

Tabela C.6: Resultados da Variabilidade METEOR - LLaMA 3 (Few-Shot)

Caso Base	**Média**	**Desvio Padrão**	**CV (%)**
Caso 1	0.66	0.01	1.66%
Caso 2	0.71	0.00	0.00%
Caso 3	0.74	0.01	0.74%
Caso 4	0.84	0.05	5.65%
Caso 5	0.75	0.06	8.47%
Caso 6	0.69	0.00	0.00%
Caso 7	0.64	0.03	4.75%
Caso 8	0.85	0.01	1.05%
Caso 9	0.68	0.02	3.03%
Caso 10	0.63	0.02	2.62%

C.3 Phi-3

Tabela C.7: Resultados da Variabilidade METEOR - Phi-3 (Zero-Shot)

Caso Base	**Média**	**Desvio Padrão**	**CV (%)**
Caso 1	0.80	0.03	3.57%
Caso 2	0.68	0.22	32.09%
Caso 3	0.82	0.08	9.75%
Caso 4	0.75	0.07	9.66%
Caso 5	0.82	0.06	7.58%
Caso 6	0.75	0.09	11.94%
Caso 7	0.81	0.09	11.66%
Caso 8	0.76	0.06	7.46%
Caso 9	0.71	0.05	7.57%
Caso 10	0.75	0.02	2.78%

Tabela C.8: Resultados da Variabilidade METEOR - Phi-3 (One-Shot)

Caso Base	**Média**	**Desvio Padrão**	**CV (%)**
Caso 1	0.69	0.12	17.62%
Caso 2	0.68	0.12	17.02%
Caso 3	0.78	0.03	3.48%
Caso 4	0.81	0.02	2.69%
Caso 5	0.75	0.05	7.24%
Caso 6	0.67	0.09	12.96%
Caso 7	0.75	0.07	9.61%
Caso 8	0.79	0.09	11.04%
Caso 9	0.66	0.12	17.67%
Caso 10	0.70	0.05	7.35%

Tabela C.9: Resultados da Variabilidade METEOR - Phi-3 (Few-Shot)

Caso Base	**Média**	**Desvio Padrão**	**CV (%)**
Caso 1	0.62	0.07	11.40%
Caso 2	0.69	0.06	8.00%
Caso 3	0.70	0.12	17.00%
Caso 4	0.77	0.04	4.89%
Caso 5	0.70	0.06	8.63%
Caso 6	0.61	0.08	13.53%
Caso 7	0.69	0.09	13.07%
Caso 8	0.81	0.06	6.93%
Caso 9	0.72	0.07	10.34%
Caso 10	0.70	0.07	9.56%

C.4 GTP4-Turbo

Tabela C.10: Resultados da Variabilidade METEOR - GPT-4 Turbo (Zero-Shot)

Caso Base	**Média**	**Desvio Padrão**	**CV (%)**
Caso 1	0.77	0.04	4.59%
Caso 2	0.73	0.08	11.13%
Caso 3	0.81	0.12	14.29%
Caso 4	0.80	0.05	5.84%
Caso 5	0.81	0.03	3.75%
Caso 6	0.59	0.06	10.91%
Caso 7	0.77	0.05	6.34%
Caso 8	0.79	0.03	3.77%
Caso 9	0.75	0.11	14.67%
Caso 10	0.71	0.03	4.70%

Tabela C.11: Resultados da Variabilidade METEOR - GPT-4 Turbo (One-Shot)

Caso Base	**Média**	**Desvio Padrão**	**CV (%)**
Caso 1	0.75	0.07	9.57%
Caso 2	0.62	0.07	10.87%
Caso 3	0.67	0.08	11.22%
Caso 4	0.80	0.01	1.63%
Caso 5	0.80	0.02	2.41%
Caso 6	0.62	0.08	13.05%
Caso 7	0.65	0.10	15.58%
Caso 8	0.83	0.02	2.49%
Caso 9	0.71	0.06	7.92%
Caso 10	0.65	0.04	5.61%

Tabela C.12: Resultados da Variabilidade METEOR - GPT-4 Turbo (Few-Shot)

Caso Base	**Média**	**Desvio Padrão**	**CV (%)**
Caso 1	0.69	0.04	5.61%
Caso 2	0.57	0.07	11.92%
Caso 3	0.64	0.11	17.00%
Caso 4	0.85	0.02	2.68%
Caso 5	0.79	0.03	3.80%
Caso 6	0.73	0.05	6.93%
Caso 7	0.67	0.07	10.38%
Caso 8	0.84	0.03	3.45%
Caso 9	0.59	0.03	4.84%
Caso 10	0.70	0.03	4.04%

C.5 GPT3.5-Turbo

Tabela C.13: Resultados da Variabilidade METEOR - GPT-3.5 Turbo (Zero-Shot)

Caso Base	**Média**	**Desvio Padrão**	**CV (%)**
Caso 1	0.75	0.01	1.97%
Caso 2	0.67	0.06	8.53%
Caso 3	0.84	0.09	10.55%
Caso 4	0.87	0.04	4.13%
Caso 5	0.82	0.04	4.67%
Caso 6	0.63	0.05	7.20%
Caso 7	0.71	0.12	17.28%
Caso 8	0.85	0.02	2.43%
Caso 9	0.65	0.04	5.54%
Caso 10	0.73	0.04	5.08%

Tabela C.14: Resultados da Variabilidade METEOR - GPT-3.5 Turbo (One-Shot)

Caso Base	**Média**	**Desvio Padrão**	**CV (%)**
Caso 1	0.71	0.04	5.20%
Caso 2	0.68	0.08	11.95%
Caso 3	0.64	0.01	1.72%
Caso 4	0.89	0.01	1.66%
Caso 5	0.82	0.03	4.04%
Caso 6	0.65	0.03	4.28%
Caso 7	0.78	0.06	7.42%
Caso 8	0.87	0.03	3.49%
Caso 9	0.66	0.04	6.21%
Caso 10	0.72	0.01	1.57%

Tabela C.15: Resultados da Variabilidade METEOR - GPT-3.5 Turbo (Few-Shot)

Caso Base	**Média**	**Desvio Padrão**	**CV (%)**
Caso 1	0.77	0.04	4.71%
Caso 2	0.73	0.04	5.56%
Caso 3	0.67	0.05	7.21%
Caso 4	0.79	0.14	17.64%
Caso 5	0.80	0.02	2.98%
Caso 6	0.65	0.05	8.00%
Caso 7	0.77	0.07	8.45%
Caso 8	0.89	0.00	0.00%
Caso 9	0.65	0.04	6.44%
Caso 10	0.73	0.03	3.55%

C.6 GPT-4o Mini

Tabela C.16: Resultados da Variabilidade METEOR - GPT-4o Mini (Zero-Shot)

Caso Base	**Média**	**Desvio Padrão**	**CV (%)**
Caso 1	0.75	0.05	6.25%
Caso 2	0.68	0.01	0.80%
Caso 3	0.81	0.04	4.62%
Caso 4	0.78	0.01	1.14%
Caso 5	0.84	0.05	6.42%
Caso 6	0.65	0.04	6.25%
Caso 7	0.82	0.02	1.93%
Caso 8	0.81	0.01	1.61%
Caso 9	0.70	0.10	14.10%
Caso 10	0.80	0.05	6.25%

Tabela C.17: Resultados da Variabilidade METEOR - GPT-4o Mini (One-Shot)

Caso Base	**Média**	**Desvio Padrão**	**CV (%)**
Caso 1	0.67	0.05	7.59%
Caso 2	0.64	0.02	2.79%
Caso 3	0.77	0.06	7.35%
Caso 4	0.83	0.02	1.84%
Caso 5	0.78	0.03	3.82%
Caso 6	0.69	0.01	1.96%
Caso 7	0.66	0.03	4.29%
Caso 8	0.83	0.01	1.08%
Caso 9	0.63	0.05	7.75%
Caso 10	0.78	0.01	0.70%

Tabela C.18: Resultados da Variabilidade METEOR - GPT-4o Mini (Few-Shot)

Caso Base	**Média**	**Desvio Padrão**	**CV (%)**
Caso 1	0.70	0.05	6.52%
Caso 2	0.72	0.05	6.60%
Caso 3	0.67	0.07	10.31%
Caso 4	0.77	0.07	9.67%
Caso 5	0.79	0.03	4.04%
Caso 6	0.67	0.03	5.09%
Caso 7	0.70	0.06	8.91%
Caso 8	0.85	0.02	2.80%
Caso 9	0.57	0.04	7.46%
Caso 10	0.76	0.04	5.44%